ATARI® PROGRAM EXCHANGE

JUNE 1982

CALCULATOR

DISKETTE (APX-20130)
REQUIRES: 24K RAM

User-Written Software for ATARI Home Computers

JUNE 1982

CALCULATOR

DISKETTE (APX-20130)
REQUIRES: 24K RAM

TRADEMARKS OF ATARI

****

Distributed by

The ATARI Program Exchange
P. O. Box 427
155 Moffett Park Drive, B-1
Sunnyvale, CA 94086

To request an APX Software Catalog, write to the address above, or call toll-free:

800/538-1862 (outside California)
800/672-1850 (within California)

Or call our Sales number, 408/745-5535.

****

CALCULATOR
INSTRUCTION MANUAL

ATARI®

A Warner Communications Company

CONTENTS

1 INTRODUCTION

Your **ATARI® Personal Computer System** with its 145-function **CALCULATOR** diskette program combines the best qualities of a calculator and a computer. The diskette program contains 145 functions ranging from simple arithmetic operations to programming commands that allow you to write your own programs. The typewriter-like keyboard permits you to enter data with less chance of "finger-error" and the display—your own television set—shows you what you have entered in readable characters, how the computer handles the data, and the calculation results. So, not only can you work complex conversions at the touch of one or two keys, but you can watch the stack and memory displays as well. In addition, the computer will let you know if you've made a mistake, or if you are asking the impossible.

If you do not know how to hook up your ATARI computer or attach the peripheral equipment you wish to use, read the appropriate Operator's Manual. This manual was written to explain the CALCULATOR diskette functions. You must have 24K RAM to use this diskette.

LOADING THE CALCULATOR DISKETTE PROGRAM

1. Remove cartridge from cartridge slot and close console cover.
2. Turn on television set.
3. Turn on disk drive unit. The BUSY light will stay on until the drive is initialized.
4. Turn on any other desired peripheral devices.
5. Hold diskette with label in the lower right corner and the arrow pointing toward the disk drive (see Figure 1).
6. Insert diskette into drive and close disk drive door.
7. Turn on computer console.

The CALCULATOR diskette program will load automatically and the display will appear on the screen.

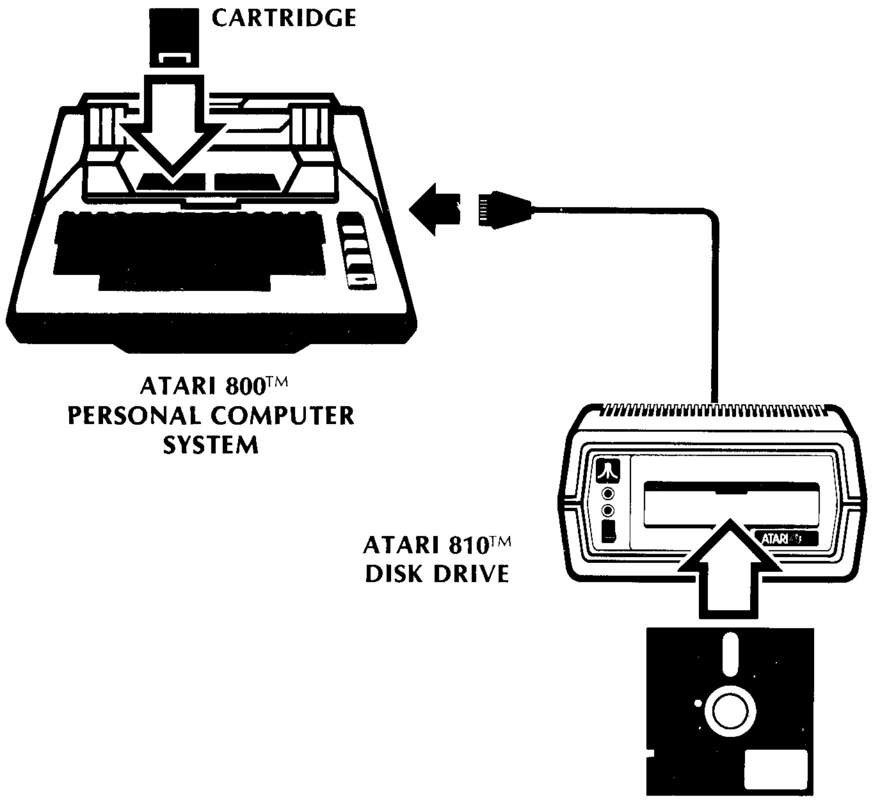

CARTRIDGE
ATARI 800^{TM} PERSONAL COMPUTER SYSTEM
ATARI 810^{TM} DISK DRIVE

Figure 1 Diskette Insertion

CONVENTIONS USED IN THIS MANUAL

KEYBOARD REPRESENTATION

Each named key is illustrated as a keycap: BREAK, SHIFT, CTRL, CLEAR, DELETE BACK S, INSERT, RETURN.

The space bar is also represented as a keycap. However, SPACE and SPACE BAR are used interchangeably.

Alphanumeric characters and symbols are illustrated in bold typeface: **12, 3, +, RPN.**

Commands are illustrated in bold typeface as single keystroke entries whenever possible: **A** for Absolute; otherwise they are illustrated in their most abbreviated form: **LOADM** for Load Memory from File. These commands are not shown as separate keytops.

Double key commands which entail one key being pressed and held while pressing a second key are illustrated as being side by side. These double key commands usually require the SHIFT or control (CTRL) key to be pressed and held; e.g.

Press and Hold	**Press**	
SHIFT	CLEAR	to clear the display
SHIFT	9	for an open parenthesis
CTRL	∧	for the power symbol

TERMINOLOGY

The words command, function, and instruction are used interchangeably throughout this manual to refer to the 145 functions that are available. However, instruction is used primarily for functions used in programs. Operators refer to symbols used in solving problems; e.g., +,*,−, /,∧.

Unless otherwise specified, the word *stack* refers to the number stack. RAM refers to Random Access Memory.

2 KEYBOARD

DISPLAY CONTROL KEYS

The CALCULATOR screen format is displayed in upper case, non-inverse video. This means that all the alphabetic characters are capitalized and that the characters appear on the screen as light characters on a dark background. In the ATARI BASIC computer language you had the option of displaying the characters in upper and lower case by using the CAPS LOWR key. That function has been nullified for the CALCULATOR program as you do not need it. Also, by using the ATARI logo key (⧄) in BASIC, you could print dark characters on a light background. That function has also been nullified. The following paragraphs describe the keys that do control the screen display.

SHIFT FUNCTION

On a typewriter, the SHIFT key is used to print a capital letter or an "upper case" symbol. This CALCULATOR program uses the SHIFT key to print an upper case symbol. The following is a list of all the symbols that require a SHIFT key preceding them and the functions they represent.

SYMBOL	KEYS USED	FUNCTION
!	SHIFT 1	Factorial
"	SHIFT 2	Reset
#	SHIFT 3	Program mode
$	SHIFT 4	End
%	SHIFT 5	Modulo
&	SHIFT 6	Logical AND
'	SHIFT 7	Run
@	SHIFT 8	Continue
—	SHIFT –	Change sign
|	SHIFT =	Logical OR
∧	SHIFT *	Raise to a power
[	SHIFT ,	Push contents of X register
]	SHIFT .	Pop contents of X register
?	SHIFT /	Print

CTRL FUNCTION

This key is similar in function to the SHIFT key. On your ATARI keyboard, there are four keys that have triple functions. The arrows are displayed when the CTRL key is pressed first. The following examples illustrate the symbols that must be preceded by the CTRL key.

SYMBOL	KEYS USED	FUNCTION
↑	CTRL −	Back Step
↓	CTRL =	Single Step
←	CTRL +	Exchange X and Y registers
→	CTRL *	Insert number

CLEAR CURRENT ENTRY FUNCTION

To clear the prompt line, you can use either the DELETE BACK S key or the SHIFT DELETE BACK S keys. This is valuable when you mistype a number or misspell a command name.

CLEAR PROMPT LINE AND STACK DISPLAY FUNCTION

Press either SHIFT CLEAR or CTRL CLEAR. This causes the computer to erase the prompt line, clear the number stack display, and place a 0 in the X register.

END OF COMMAND FUNCTION

To indicate the end of a command entry, press RETURN or the SPACE BAR. The RETURN and SPACE keys may be used interchangeably in this program. When either of these keys or a number is pressed, the computer program clears the prompt line and displays the function name in the scroll area, then performs the necessary operation.

SYSTEM RESET FUNCTION

The SYSTEM RESET key initiates a "warm start" in that it will return the CALCULATOR program to its original ON state. However, the program memory and memory registers are not cleared. This operation must be done separately using either **CLPROG** or **CLMEM** instructions (to be discussed in the **PROGRAMMING INSTRUCTIONS AND EXAMPLES** section).

FUNCTION ENTRY FORMATS

This CALCULATOR diskette program contains two different ways to enter functions: token and abbreviated. Some functions can be entered either way. A complete summary of commands with their tokens or abbreviated forms is given in Appendix C.

TOKEN ENTRY

A token entry is a single symbol entry. Examples of token entries are the mathematical operators; +, −, *, /, =. Other token entries were discussed in the SHIFT function and CTRL function paragraphs. They required pressing two keys simultaneously to print the symbol. Refer back to these paragraphs.

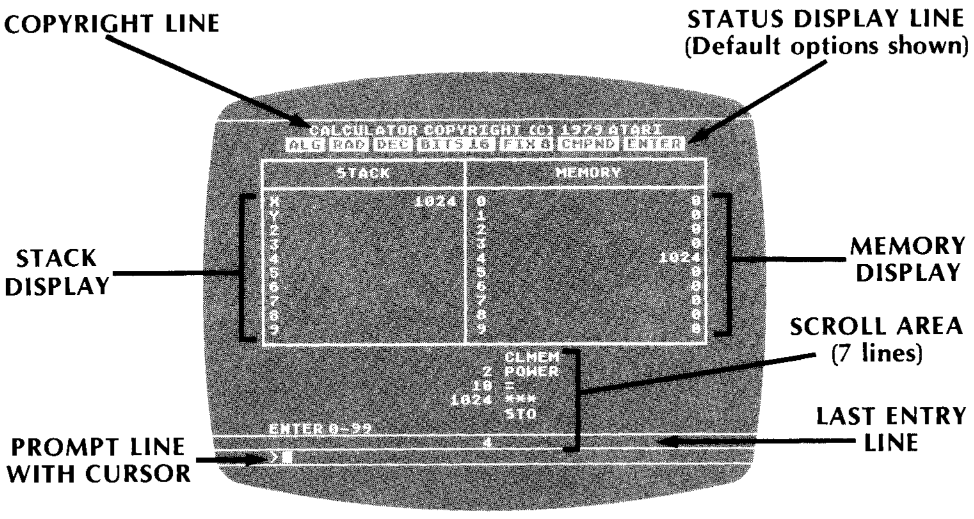
ABBREVIATED ENTRY

Other functions have "abbreviated" forms ranging from a single letter to six letters. Examples of these are **O** for Octal, **NOT** or **NOTRC** for No Trace, **INSN** or **INSNUM** for Insert Number, and **ALGN** for Algebraic Notation with No Operator Precedence.

As this manual progresses, you will use the display control keys and functions so that you will become completely familiar with them. But for now, look at the screen display.

SCREEN DISPLAY

As you can see, the display on the screen consists of several divisions, each denoted by a different color. Figure 2 illustrates the screen and its various divisions with sample entries.



```
    CALCULATOR COPYRIGHT (C) 1979 ATARI
ALG RAD DEC BITS 16 FIX 8 CMPND ENTER
          STACK               MEMORY
X              1024  0                 0
Y                    1                 0
2                    2                 0
3                    3                 0
4                    4              1024
5                    5                 0
6                    6                 0
7                    7                 0
8                    8                 0
9                    9                 0
                       CLMEM
                     2 POWER
                    10 =
                  1024 ***
                       STO
ENTER 0-99
                     4
>
```

Figure 2 Example of Calculator Screen Display

PROMPT AND SCROLL AREA

The scroll area, in which all keyboard and computational actions are reflected, is divided into three fields. The left field is reserved for computer-generated messages. In the above figure, the computer has displayed the message, ENTER 0–99. The program also uses this field to display error messages. Appendix A gives a complete list of error messages while Appendix B gives a list of the "helpful" messages the program uses to prompt you. The middle field is reserved for numbers. Your entries and the computer program's calculated results are displayed in this field. Any computer-calculated results will have *** after them. In Figure 2, the number field contains 2, 10, and 1024. The rightmost field displays function names, operator symbols and asterisks. In Figure 2, this field displays **CLMEM, POWER,** =, ***, and **STO.**

The prompt symbol (>) is located on the last line of the scroll area. It is followed by the cursor which shows you where your next entry will begin. To familiarize yourself with the way your CALCULATOR displays data you enter, type **1**. The 1 appears next to the prompt symbol and the cursor moves one space to the right. Now press SPACE BAR. The 1 moves up one line into the scroll area number field and into the X register (see **NUMBER STACK**). Now type the numbers on the next page.

2 SPACE BAR
3 SPACE BAR
4 SPACE BAR
5 SPACE BAR
6 SPACE BAR
7 SPACE BAR
8

The scroll area now looks like this:

```
                    1
                    2
                    3
                    4
                    5
                    6
                    7
> 8█
```

Notice that each time you entered a number by pressing the SPACE BAR, the entire scroll moved up one line.

Type

CTRL CLEAR

The 8 that was on the prompt line disappears as does the 7 that was in the X register of the Stack Display.

NUMBER STACK

Located directly above the scroll area, the stack provides temporary storage locations (or registers) for 42 numbers. Of these 42 locations, only the first 10 are displayed on the screen. Although the other 32 locations are "invisible," they are nonetheless available for your use. The first two registers are designated as the X and Y registers. The X register is the location where your results appear; therefore, it is also known as the accumulator register.

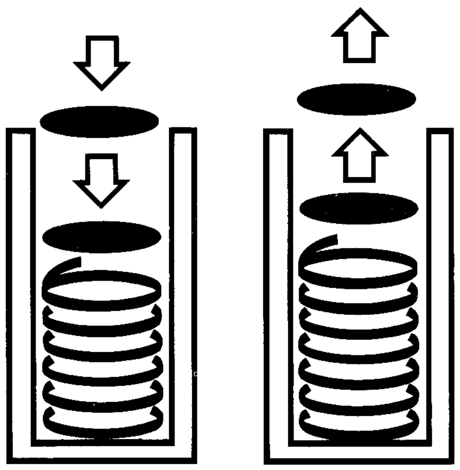


Figure 3 Stack Analogy

The stack works very simply. If you have ever been in a cafeteria that uses an automatic spring-loaded plate dispenser, you have seen a practical example of a stack.

The plate dispenser can only hold a certain number of plates. Therefore, it has a certain number of places or locations. When the busboy puts one plate on the dispenser, he has "loaded" one location. When he places a second plate on top of the first one, the first plate moves down one location. This move is called a **PUSH** and the Number Stack works the same way.

Type	**Stack Display**	**Comments**
5 SPACE BAR	X 5	This places a 5 in the X register.

To move this 5 to the Y register (one location down), you must use a **PUSH** command.

PUSH COMMAND

SHIFT [(left bracket) or **PUSH** SPACE BAR

You can either type the word **PUSH** followed by pressing the SPACE BAR, or you can press SHIFT [to use the token entry. The following example uses the token entry.

Type	**Stack Display**	**Comments**
SHIFT [	X 5 Y 5	This command causes the content of the X register to move down to the Y register, but it does does not disappear from the X register.
6 SHIFT [	X 6 Y 6 2 5	This PUSH places the 6 in the X register and immediately moves it down to the Y register. The 5 that was in the Y register is "pushed" to the 2 register.
7 SHIFT [	X 7 Y 7 2 6 3 5	Again, this PUSH places the new entry, 7, into the X register and immediately moves it to the Y register. This causes the former contents of the X and Y registers and the 2 register to be "pushed" down also.

Each time you enter a value followed by a PUSH command, the value will appear in the X register and will be immediately duplicated into the Y register. If you do not enter a value preceding a PUSH command, the number that is already in the X register will be duplicated into the Y register. In either of these cases, the contents of the other registers will move down one register.

If you fill all the registers (42) with values then try to enter another number, the screen will display the message ERROR—STACK FULL. The program will not allow your last number to be entered and the stack will remain unchanged.

POP COMMAND

SHIFT] (right bracket) or **POP** SPACE BAR

Just as you can remove plates from the cafeteria plate dispenser, you can remove numbers from the stack register. To do this, you either type **POP,** then press the SPACE BAR, or you press SHIFT].

Type	Stack Display	Comments
SHIFT]	X 7 Y 6 2 5	This command removes the number that was in the X register and replaces it with the number that used to be in the Y register. This causes all other register contents to move up one location.
SHIFT]	X 6 Y 5	Now you can see more clearly that the 7 was removed from the X register, replaced by the 6 that was in the Y register, and that the 5 that was in the 2 register is now moved up to the Y register. The "popped" number (7) shows in the scroll area.

Notice that in removing numbers from the stack, you didn't remove the first number you entered, but the last. The same is true of the plate dispenser. The busboy doesn't take the first plate he loaded onto the stack, but the last one he placed there. This is called a **LIFO** (Last In, First Out) structure. You can only remove the last number you entered (the number in the X register).

Currently you have two numbers left in the stack—a 6 in the X register and a 5 in the Y register.

Type	Stack Display	Comments
SHIFT]	X 5	This POP command removes the 6 from the X register and moves the 5 from the Y register to the X register.

This leaves only one number in the stack. If you try to enter another POP command, the screen will display the message ERROR—STACK EMPTY. The computer program will not allow you to empty the stack completely.

CLEAR X REGISTER COMMAND

CLX SPACE BAR

Although the computer program will not allow you to empty the stack completely, you can use this **CLX** command to replace the content of the X register with 0.

Type	Stack Display		Comments
CLX SPACE BAR	X	0	This command removes the 5 from the X register and replaces it with a 0.

CLEAR STACK COMMANDS

CLR SPACE BAR
SHIFT CLEAR
CTRL CLEAR

Any one of the above three commands will clear the stack and set the contents of the X register to 0 as well as clearing the current entry on the prompt line.

EXCHANGE X AND Y REGISTERS COMMAND

CTRL ← or **XCHGY** SPACE BAR

Sometimes you will find it necessary to switch the contents of the X and Y registers in the stack. To do this, you either type **XCHGY** and press the SPACE BAR, or you press CTRL ← .

Type	Stack Display		Comments
5 SHIFT [	X	5	
	Y	5	
6 SPACE BAR	X	6	
	Y	5	
CTRL ←	X	5	
	Y	6	The 5 that was in the Y register is now in the X register and vice versa.

The following example uses all the commands explained in this section.

Type	X Register Display	Y Register Display
CTRL CLEAR		
3 SPACE BAR	3	
SHIFT [	3	3
CLX SPACE BAR	0	3
CTRL ←	3	0
CLR SPACE BAR	0	
5 SHIFT [	5	5
2 SPACE BAR	2	5
SHIFT]	5	
CTRL CLEAR	0	

MEMORY

You have 100 memory locations (or registers) into which you can store numbers. When you first turn the power on, all memory registers are initialized to 0. The screen display shows the first 10 memory locations (labeled 0–9). The following six commands are exclusively associated with memory. The other two commands associated with memory, **SAVEM** and **LOADM,** are described in **I/O COMMANDS FOR PERIPHERAL DEVICES.**

STORE COMMAND

STO SPACE BAR

This command stores whatever is in the X register of the stack into a memory location that you specify. After you type **STO** SPACE BAR, the screen displays the message ENTER 0–99. Then you enter the memory location in which you wish to place the content of the X register.

Type	Stack Display		Memory Display		Scroll Message	Comments
5 SPACE BAR	X	5				
STO SPACE BAR	X	5			ENTER 0–99	
0 SPACE BAR	X	5	0	5		The 5 in the X register is now stored in memory location 0.
23 SPACE BAR	X	23	0	5		
STO SPACE BAR	X	23	0	5	ENTER 0–99	
1 SPACE BAR	X	23	0	5		The 23 in the X register is now stored in memory location 1.
			1	23		
17 SPACE BAR	X	17	0	5		
			1	23		
STO SPACE BAR	X	17	0	5	ENTER 0–99	
			1	23		
22 SPACE BAR	X	17	0	5		The 17 in the X register is now stored in memory location 22 (not visible on display).
			1	23		
			22	17		

RECALL COMMAND

RCL SPACE BAR

To retrieve a number from a memory location, use the RCL command. This command takes the number in the memory location you specify and places it in the X register.

Type	Stack Display		Memory Display		Scroll Message	Comments
RCL SPACE BAR			0	5	ENTER 0–99	
			1	23		
0 SPACE BAR	X	5	0	5		The 5 in memory location 0 is now loaded into the X register.
			1	23		

Note: In Reverse Polish Notation (RPN), an automatic PUSH is performed. See **Reverse Polish Notation** computation mode later in this section.

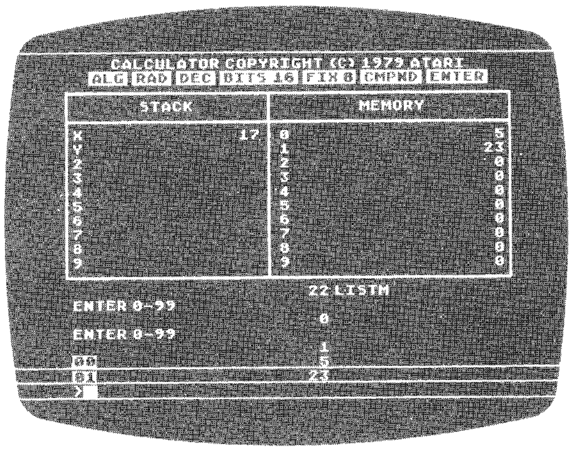
Type	Stack Display		Memory Display		Scroll Message	Comments
RCL SPACE BAR			0	5	ENTER 0–99	
			1	23		
22 SPACE BAR	X	17	0	5		The 17 in memory location 22 (not visible on screen) is now loaded into the X register.
			1	23		

LIST MEMORY COMMAND

LISTM SPACE BAR

This command displays the contents of the requested memory locations in the scroll area. After you enter the command, the screen displays the message ENTER 0–99 for you to enter the first memory location to be displayed. After you enter the number (which must be a decimal number), the screen displays a second message ENTER 0–99 for the last memory location to be displayed. Remember, the scroll area displays only seven lines at a time. If you request more than seven memory locations, you can press the BREAK key to stop the listing. However, if you do, you cannot continue the listing. If you want to halt the listing temporarily, press CTRL **1** to stop it and press CTRL **1** again to restart it.

Type	Stack Display		Memory Display		Scroll Message	Comments
LISTM SPACE BAR	X	17	0	5	ENTER 0–99	
			1	23		
0 SPACE BAR					ENTER 0–99	
1 SPACE BAR						Lists data for memory locations 0 and 1.



```
    CALCULATOR COPYRIGHT (C) 1979 ATARI
  ALG RAD DEC BITS 16 FIX 8 CMPND ENTER
 ┌─────────────────┬──────────────────────┐
 │      STACK      │        MEMORY        │
 ├─────────────────┼──────────────────────┤
 │X              17│0                    5│
 │Y                │1                   23│
 │2                │2                    0│
 │3                │3                    0│
 │4                │4                    0│
 │5                │5                    0│
 │6                │6                    0│
 │7                │7                    0│
 │8                │8                    0│
 │9                │9                    0│
 └─────────────────┴──────────────────────┘
                      22 LISTM
 ENTER 0-99
                         0
 ENTER 0-99
                         1
 00                      5
 01                     23
 >
```

Figure 4 LISTM Display

The memory location numbers are displayed in inverse video in the left field of the scroll area and the contents of each location are displayed in the center field of the scroll area.

Remember, you can stop and restart the listing by pressing CTRL **1.**

Type	Stack Display		Memory Display		Scroll Message	Comments
LISTM SPACE BAR	X	17	0	5	ENTER 0-99	
			1	23		
0 SPACE BAR	X	17	0	5	ENTER 0-99	
			1	23		
25 SPACE BAR	X	17	0	5		Begins displaying memory locations 0-25 as soon as SPACE BAR is pressed.
			1	23		
CTRL **1**	X	17	0	5		Stops listing.
			1	23		
CTRL **1**	X	17	0	5		Restarts listing.
			1	23		

SUM TO MEMORY COMMAND

SUM SPACE BAR

This command adds the content of the X register to a memory location that you specify. It then stores the resulting sum into the same specified memory location.

Note: It is advisable to store a number in memory before doing a series of SUM commands to make sure the memory register is initialized correctly.

If the numbers exceed the allowed range, which for decimal is ±1E −98 to ±9E +97 (Figure 5), the message ERROR—ARITHMETIC OVERFLOW appears in the scroll area and the X register will be set to 0. (See NUMBER BASES.) The contents of the memory register will be displayed in the scroll area followed by the last number you attempted to enter.

Base	**Range in Base**	**Range in Decimal**
Decimal	± −98 to ±9E+97	
Octal	20000000000 to 17777777777	−2147483648 to +2147483647
Hexadecimal	80000000 to 7FFFFFFF	−2147483648 to +2174783647

Figure 5 Maximum Range of Number Bases

Type	**Stack Display**		**Memory Display**		**Scroll Message**	**Comments**
15 SPACE BAR	X	15	0	5		
			1	23		
SUM SPACE BAR	X	15	0	5	ENTER 0-99	
			1	23		
0 SPACE BAR	X	15	0	20		The 15 in the X register was added to the 5 in memory location 0 and the resultant sum of 20 is stored in memory location 0.
			1	23		

EXCHANGE NUMBER IN X REGISTER WITH MEMORY COMMAND

XCHM SPACE BAR

Using this command, you can switch the number in the X register with the content of a memory location that you specify.

Type	Stack Display		Memory Display		Scroll Message	Comments
XCHM						
SPACE BAR	X	15	0	20	ENTER 0–99	
			1	23		
1 SPACE BAR	X	23	0	20		The 15 that was in the X register goes to memory location 1 and the 23 that was in memory location 1 is now loaded into the X register.
			1	15		

CLEAR MEMORY COMMAND

CLMEM SPACE BAR

This command puts a 0 in each memory location.

Type	Stack Display		Memory Display		Scroll Message	Comments
CLMEM SPACE BAR	X	23	0	0		
			1	0		

STATUS DISPLAY

Each category in the Status Display has options that you can change to suit the type of calculations you are doing. When the CALCULATOR program is first inserted (or you press SYSTEM RESET), the default option for each category appears on the screen. (A default option is the option selected by the CALCULATOR.) Figure 6 defines the categories and the following paragraphs explain the options for each category.

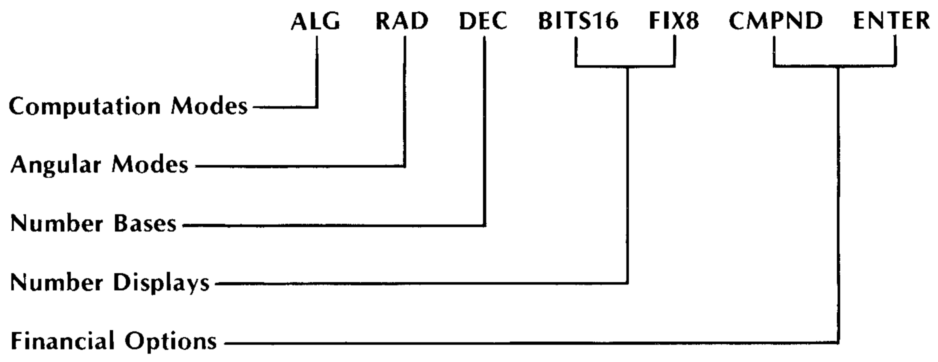


Figure 6 Status Display Category Definition

COMPUTATION MODES

Your CALCULATOR program can operate in any one of three different computation modes.

ALG	— Algebraic Notation with Operator Precedence
ALGN	— Algebraic Notation with No Operator Precedence
RPN	— Reverse Polish Notation

Algebraic Notation With Operator Precedence (ALG)

This is the default option and so far, all of your entries have been made in this mode. This notation uses operator precedence. Operator precedence simply means that the CALCULATOR, when in this mode, performs certain operations before it does other operations. Figure 7 lists the operator precedence.

Highest Priority	
( )	Left and Right Parentheses. If parentheses are "nested" (or placed within another set of parentheses), the CALCULATOR will perform the operation contained with the innermost set first.
LSHF, RSHF, MOD	Left Shift, Right Shift, Modulo. These operations will be discussed in the **PROGRAMMING INSTRUCTIONS AND EXAMPLES** section.
∧, **ROOT**	Raise a number to a power (exponentiation): Take the root of a number.
*, /	Multiplication, Division
+, −	Addition, Subtraction
AND	Logical AND
OR, XOR	Logical OR, Exclusive OR
Lowest Priority	

Figure 7 Operator Precedence

In Figure 7, the operations listed on the same line are of equal precedence and the CALCULATOR program will perform them in the order of their appearance in the problem (from left to right). For example, in the problem

21 −9*2/3 =

the CALCULATOR program would perform the multiplication and division operations before the subtraction operation. This is the same method you learned in elementary mathematics. To have the CALCULATOR solve this problem, follow the steps listed below.

Type	Stack Display		Comments
21 −	X	21	Subtraction requires two numbers so the CALCULATOR puts the first number in both the X and Y registers (does an immediate PUSH).
	Y	21	
9*	X	9	The CALCULATOR must perform the multiplication first, so it leaves the 21 in the 2 register for later use.
	Y	9	
	2	21	
2/	X	18	The CALCULATOR performs the multiplication 9*2=18.
	Y	18	
	2	21	
3 =	X	15	The CALCULATOR divided the 3 into the 18 in the X register, then subtracted the resulting 6 from the 21 in the Y register. It then placed the final answer in the X register.

In ALG, the equals symbol indicates that any remaining operations should be completed.

If you need to "save" a number for later use, you will need to do a PUSH command by pressing SHIFT [or typing **PUSH** SPACE BAR.

Algebraic Notation With No Operator Precedence (ALGN)

Type

ALGN SPACE BAR

The computation mode in the Status Display line changes from **ALG** to **ALGN** to reflect your new choice. ALGN mode differs from ALG in that it ignores the operator precedence and performs each operation as you enter it. Using the same problem, 21 −9*2/3 =,

Type	Stack Display		Comments
21 –	X Y	21 21	The X and Y registers both display the number you entered.
9*	X Y	12 12	The CALCULATOR subtracted 9 from the 21 in the X register and stored the result in both the X and Y registers.
2/	X Y	24 24	The CALCULATOR multiplied the 2 times the 12 in the X register and placed the resulting 24 in both the X and Y registers.
3 =	X	8	The CALCULATOR divided the 24 stored in the X register by 3 and placed the answer in the X register.

In this mode, you determine which operation you want the CALCULATOR to perform first by using the parentheses. Otherwise, the CALCULATOR will perform each operation as it is entered.

Type	Stack Display		Comments
SHIFT [	X X	8 8	PUSH the answer from the last problem to save it.
21 –	X Y 2	21 21 8	The CALCULATOR places the 21 you entered in both the X and Y registers and moves the 8 to register 2.
(9*	X Y 2 3	9 9 21 8	Because of the parentheses, the CALCULATOR cannot subtract the 9 from the 21, so it stores the 21 in register 2 and pushes the 8 to register 3.
2/	X Y 2 3	18 18 21 8	The CALCULATOR performs the multiplication but still does not subtract the result from the 21 in register 2. It must wait for the second part of the division operation.
3 =	X Y	15 8	The CALCULATOR performs the division, then subtracts the resulting 6 from 21 that was stored in register 2 and places the answer of 15 in the X register. You still have the answer from the last problem in the Y register so you can compare the difference the parentheses made in the answer.

In ALGN, you have the option of placing parentheses wherever you need them.

Both ALG and ALGN modes are *infix* notations. You type the number followed by an operator and use an equals symbol to indicate where the problem ends. But there are times when this type of notation is inconvenient—especially in scientific programming functions. At those times, there is a third computation mode available to you on the CALCULATOR.

Reverse Polish Notation (RPN)

Reverse Polish Notation is a *postfix* notation which excludes the use of parentheses and the equals symbol. Therefore, it does not use operator precedence and you must enter the problems a little differently. For example, in a simple problem such as 3+2=, you could not place the operation symbol between the 3 and the 2 and end with an equals symbol after the 2. IF you use an equals symbol, the CALCULATOR will respond with the message ERROR—NOT VALID COMMAND OR MESSAGE.

To work the simple problem in RPN,

Type	**Stack Display**		**Comments**
CTRL CLEAR	X Y	0	Clears the X and Y registers.
RPN SPACE BAR			Status Display now reflects RPN mode.
3 SPACE BAR	X Y	3 0	The 0 that was in the X register is automatically pushed into the Y register.
2 SPACE BAR	X Y 2	2 3 0	Entering the 2 pushes the 0 into the 2 register and the 3 into the Y register.
+	X Y	5 0	The CALCULATOR adds the 2 to the 3 in the X register, then places the resulting sum of 5 in the X register.

Each time you enter a number, it is displayed in the X register and causes the previous contents to be pushed into Y and the previous contents of Y to be pushed into the 2 register, etc. RPN operators work only with the contents of the X and Y registers. The result is always placed in the X register and the stack is always popped. In the above problem, each entry was followed by a space, so that the number stack activity could be more clearly seen. It is not necessary to separate numbers and operator entries with a space. In the succeeding problems, the appropriate operator will be placed immediately following the number entry. However, in RPN a space must **always** separate the first two numbers.

PROBLEM: 1 + 5 * 2 / 3 − 2 =

Type	Stack Display		Comments
CTRL CLEAR	X Y	0	Clears the X and Y registers.
1 SPACE BAR	X Y	1 0	The X register displays your first entry and pushes the 0 that was in the X register to the Y register.
5+	X Y	6 0	The CALCULATOR adds the 5 to the 1 in the X register and stores the results in the X register.
2*	X Y	12 0	The CALCULATOR multiplies the 2 times the 6 in the X register and places the resulting 12 in the X register.
3/	X Y	4 0	Again, the CALCULATOR performs the division operation using your entry of 3 and the 12 in the X register. It then stores the 4 in the X register.
2−	X Y	2 0	The CALCULATOR performs the last operation using the 4 in the X register and the 2 you entered. The final result of 2 is placed in the X register.

Note that in algebraic (ALG) notation, you'd see this problem written as ((1+5)*2)3−2=. In ALG, parentheses determine which operations are performed first. In RPN, you determine the order by the sequence in which you enter the numbers and operator symbols. In ALG, the equals sign performs the totaling task and determines when the problem is finished. The equals symbol is not needed because totaling is done continuously into the X register.

If you do not want a number operated on immediately you must use a space instead of the operation symbol. The number will be placed in the stack for future computation. Using the same problem, solve for the multiplication operation first, then the division, then the subtraction, and finally the addition.

Type	Stack Display		Comments
CTRL CLEAR	X Y	0	Clears the X and Y registers.
1 SPACE BAR	X Y	1 0	This problem starts out the same as the last one.
5 SPACE BAR	X Y 2	5 1 0	Since there have been no operation symbols entered yet, the 1 that was in the X register is pushed down into the Y register and the 5 is placed in the X register.

Type	Stack Display		Comments
2*	X	10	The CALCULATOR multiplies the 5 in the X register by the 2 that you entered and stores the results in the X register. The 1 in the Y register still does nothing.
	Y	1	
	2	0	
3/	X	3.3333333	The CALCULATOR divides the 10 that was in the X register by the 3 you entered and places the result in the X register. Notice that the 1 in the Y register is still waiting for an operation symbol.
	Y	1	
	2	0	
2−	X	1.3333333	
	Y	1	
	2	0	
+	X	2.3333333	By entering an operator symbol by itself (without a preceding number), you are telling the CALCULATOR to add what is in the Y register to the number in the X register and display the results in the X register.
	Y	0	

In algebraic notation, this problem would be written as 1+((5*2)3)−2=. Any "lone" operator symbol you enter will work with the contents of the X and Y registers. If you want to solve another problem (or another part of a problem) and then combine the two answers, don't clear the stack. For instance, you have a 2.3333333 in the X register that we will assume is a partial answer to one part of a problem. Now, to solve the second part; e.g., 2+16/3 (or in algebraic notation (2+16)/3),

Type	Stack Display		Comments
2 SPACE BAR	X	2	The 2.3333333 is automatically pushed to the Y register.
	Y	2.3333333	
	2	0	
16+	X	18	The 2 in the X register and the 16 are added together and the result placed in the X register. The Y register still stores the 2.3333333.
	Y	2.3333333	
	2	0	
3/	X	6	The CALCULATOR performs the division and places the result in the X register. Now, to combine the X and Y registers to get a final result, enter a "lone" operator.
	Y	2.3333333	
	2	0	
+	X	8.3333333	The result of adding the contents of the X and Y registers is placed in the X register.
	Y	0	

The *automatic push* feature of RPN can be very advantageous if you are doing a problem in several sections or several related problems. However, it also fills your stack more quickly. Since it takes slightly longer to display 10 numbers in the stack than it does to display one, you might want to clear away any unwanted data occasionally. You already know how to clear the X register (CLX SPACE BAR) or clear the entire stack (CTRL CLEAR or SHIFT CLEAR). To clear all registers except the X register, you can type **RPN** SPACE BAR. This leaves the content of the X register unchanged.

ANGULAR MODES

Your CALCULATOR program contains a variety of trigonometric functions, many of which involve angle calculations. Angles are usually measured in degrees or radians. In solving electrical engineering problems, it may be more convenient to work in radians than degrees. Notice that the default option for this mode is **RAD** (radians). To change the Status Display from radians to degrees, type the command **DEG** SPACE BAR. The Status Display adjusts accordingly. Changing this option does not affect the X and Y registers immediately; however, it is a good idea to check to make sure you are in the correct mode before solving problems or writing programs dealing with angular measurement.

NUMBER BASES

So far, all the problems and examples have been given in decimal mode (**DEC**). However, the CALCULATOR is capable of working in two other bases: octal (**OCT**) and hexadecimal (**HEX**). Although all three number bases are stored inside the computer in BCD (Binary Coded Decimal) format, each number base is entered and displayed somewhat differently.

Decimal Base and the Floating Point Notation

The most well-known number base is decimal which uses the 10 numbers (0–9). The CALCULATOR allows you to enter 9 digits if the exponent is odd or 10 digits if the exponent is even. Two digits are allowed for an exponent. If you try to enter more digits the program treats the additional digits as zeroes—up to 15 numbers. If you try to enter 15 digits, the program displays a message ERROR—TOO MANY CHARACTERS. But there are times when you will need to work with very large or very small numbers. Your CALCULATOR provides the means of entering these numbers using a system known as Floating Point Notation.

Suppose you want to enter the number one billion two hundred thirty-four million.

Type	Stack Display		Comments
CTRL CLEAR	X	0	Clears the stack registers and sets the X register to 0.

Type	Stack Display		Comments
1234000000 SPACE	X Y	1.234E+09 0	The X register displays the number in floating point notation. The 1.234 portion of the notation is the mantissa. The E stands for exponent and is the main indicator of a number in floating point notation. The operation symbol alternate indicates whether you move the decimal point to the right (+) or to the left (−). The 09 tells you to move the decimal point 9 places.
.0000001234 SPACE	X Y 2	1.234E−07 1.234E+09 0	This is an example of how an extremely small number would appear in floating point notation. Notice the operation symbol indicates that you move the decimal point to the left.
CTRL CLEAR	X	0	Clears the stack and sets the X register to 0.

Octal Base
In this mode, numbers are entered and displayed in base 8. In other words, you use only the numbers 0 through 7. When the CALCULATOR is in OCT (octal) mode, you cannot use decimal points or exponents in the numbers you enter. If you attempt to do so, the message ERROR—NOT VALID COMMAND OR NUMBER will be displayed on the screen.

Octal numbers can be up to 10 decimal digits long. However, only eight digits can be displayed in DEC mode so 17777777777 in octal would actually be displayed as 2.1474836E+09 in decimal. Addition, subtraction, multiplication, division, and bit manipulation functions are accurate in the full range of octal numbers.

To obtain negative numbers in octal, enter the absolute value of the number and use the **CHGSGN** function. (See **FUNDAMENTAL FUNCTIONS** section.) Alternatively, the two's complement (see **PROGRAMMING INSTRUCTIONS AND EXAMPLES** section) form of the number in either 32-bit format (**BITS32**) or with the number of bits specified by **BITS** may be entered (see **DISPLAYING NUMBERS**).

To convert the number 14 from decimal to octal,

Type	Stack Display		Comments
CTRL CLEAR	X	0	Clears the stack display.
14 SPACE BAR	X Y	14 0	Places a 14 in X register.
OCT SPACE BAR	X Y	16 0	The decimal number 14 is equal to 16 in octal.
DEC SPACE BAR	X Y	14 0	Converts the number back to decimal.

Hexadecimal Base

Numbers are entered and displayed in base 16. This means that you use the digits 0 through 9 and the alphabetic characters A through F.

Hexadecimal	Decimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15
10	16
11	17

As in the octal mode, hexadecimal (HEX) numbers can be up to 8 decimal digits long. But, the same decimal limitation exists for hexadecimal as for octal so the hexadecimal value 7FFFFFFF would be displayed in decimal as 2.1474836E+09. Addition, subtraction, multiplication, division, and bit manipulation functions are accurate in the full range of hexadecimal numbers.

In entering a hexadecimal number beginning with A through F, you must precede it by the digit 0 to distinguish it from a function name. Similarly, if a hexadecimal number ends with A through F, you must follow it with a separator (RETURN or SPACE BAR).

To obtain negative numbers in hexadecimal, enter the absolute value of the number and use the **CHGSGN** function. (See **FUNDAMENTAL FUNCTIONS** section.) Alternatively, you can enter the two's complement (see **PROGRAMMING INSTRUCTIONS AND EXAMPLES** section) form of the number in either 32 bit format (BITS32) or with the number of bits specified by BITS.

To convert the number 14 from decimal to hexadecimal,

Type	Stack Display		Comments
	X	14	You already have the 14 in the X register.
	Y	0	
HEX SPACE BAR	X	E	The decimal number 14 is equal to E in hexadecimal.
	Y	0	
DEC SPACE BAR	X	14	Converts the number back to decimal.
	Y	0	

Be sure to check the Status Display before solving any problems or writing any programs to see whether or not you have entered the number base you want.

DISPLAYING NUMBERS

Fix Command

FIX SPACE BAR

The FIX command defines the number of digits you wish to have to the right of the decimal point. This command applies only in the decimal mode, but not numbers in hexadecimal or octal modes. After you enter the command, the message ENTER 0–8 will be displayed as a reminder. The number you enter is always assumed to be in the decimal mode and is rounded off to the nearest integer. If it is outside of the specified range, you will get another message, ERROR—NUMBER OUT OF RANGE and there will be no change to the FIX setting.

If you enter FIX and then decide you don't really want to do a FIX after all, type **NOP** for No Operation. You will get ERROR—NUMBER OUT OF RANGE since the CALCULATOR was expecting a number, but the FIX setting will not be changed. The same holds true for any command that requests that a number be entered after the command is entered. The numbers entered for these commands are special in that they are always in decimal, are always rounded to the nearest integer and are not placed in the X register.

A value from 0 through 7 in the FIX command selects the number of digits that will be displayed to the right of the decimal point. Zeroes will be displayed at the end of a number as necessary to obtain the proper number of digits. The maximum number of digits that can be displayed, not including the exponent, is 8. This can reduce the number of places possible after the decimal point. FIX affects the display only; the internal form of the number remains unchanged. The numbers are rounded to the number of digits to be displayed. Zero (0) through 4 rounds down; 5 through 9 rounds up. FIX8 selects floating point decimal mode. This is the initial setting of the CALCULATOR. Displayed numbers are always rounded to 8 digits. In FIX8, however, zeroes are not added after the decimal point to make 8 digits. Numbers with a magnitude less than .01 or greater than or equal to 1E9 will be displayed in floating point notation.

Type	X Register		Left Field of Scroll Area
DEC SPACE BAR			
CTRL CLEAR	X	0	
FIX SPACE BAR	X	0	ENTER 0–8
8 SPACE BAR	X	0	
.12345678 SPACE BAR	X	0.12345678	
FIX SPACE BAR	X	0.12345678	ENTER 0–8
7 SPACE BAR	X	1.2345678E-01	
FIX SPACE BAR	X	1.2345678E-01	ENTER 0–8
0 SPACE BAR	X	1E-01	
1.9 SPACE BAR	X	2.	

Type	X Register Display		Left Field of Scroll Area
FIX SPACE BAR	X	2.	ENTER 0–8
1 SPACE BAR	X	1.9	
FIX SPACE BAR	X	1.9	ENTER 0–8
4 SPACE BAR	X	1.9000	
1234.5678 SPACE BAR	X	1234.5678	
FIX SPACE BAR	X	1234.5678	ENTER 0–8
7 SPACE BAR	X	1234.5678	
1.234E+10 SPACE BAR	X	1.2340000E+10	
FIX SPACE BAR	X	1.2340000E+10	ENTER 0–8
8 SPACE BAR	X	1.234E+10	

Bits Command

BITS SPACE BAR

This command is used only in hexadecimal or octal modes. It selects the number of bits for numbers in hex or octal modes. When **BITS** SPACE BAR is entered, the message ENTER 1-32 is displayed. This tells you to enter the number size you desire for HEX or OCT mode displays and the bit manipulation instructions AND, OR, XOR, LSHF, and RSHF (see **PROGRAMMING INSTRUCTIONS AND EXAMPLES**). The number you enter must be between 1 and 32 inclusive. The range of numbers that can be entered and displayed is:

BITS Setting*	Range (in decimal)
8	−128 to +127
16	−32768 to +32767
32	−3147483648 to +2147483647

Type	X Register Display		Left Field of Scroll Area
CTRL CLEAR	X	0	
HEX SPACE BAR	X	0	
BITS SPACE BAR	X	0	ENTER 1–32
16 SPACE BAR	X	0	
1 SPACE BAR	X	1	
CHGSGN SPACE BAR	X	FFFF	
0FFFF SPACE BAR	X	FFFF	
0FFFFFFFF SPACE BAR	X	FFFF	
BITS SPACE BAR	X	FFFF	ENTER 1–32
32 SPACE BAR	X	FFFFFFFF	

*There are of course other BITS settings, but these are the most commonly used.

If a number to be displayed in hexadecimal or octal is within the range allowed for decimal numbers, but is not within the range specified by the BITS command, then the message ERROR—HEX/OCT OVRFLW will be displayed in the stack or memory location where the number was to be displayed.

The number will remain unchanged internally so switching to decimal mode will allow it to be displayed. If a program is executing without display and no bit manipulation functions are used, then HEX/OCT OVRFLW will not occur.

As mentioned before, negative numbers are displayed in two's complement form in hexadecimal and octal modes. Two's complement is the complement of the representation of the absolute value of the number, plus one.

The BITS command may be used to specify the word size or address space of the computer you are working with. On a byte-oriented machine, you may want to use BITS8. If the machine has a 16-bit address length, then you may use BITS16 to do address calculations. BITS16 is the initial (default) setting.

FINANCIAL OPTIONS

Although financial options are part of the Status Display, these options are discussed in the **FINANCIAL** section of this manual.

4 CONVERSIONS

This section includes the conversions that can be performed on the CALCULATOR by entering a number followed by the "old" units as well as a table showing the conversions factors for all other measurements.

For example, to convert 68 degrees Fahrenheit to degrees Celsius:

FAHRENHEIT ↔ CELSIUS

Type	Stack Display	Scroll Message
DEC SPACE BAR		
68F SPACE BAR	X 20	TO C

The left field of the scroll area "reminds" you that you are converting to C (Celsius). To convert 100 degrees Celsius to degrees Fahrenheit:

Type	Stack Display	Scroll Message
100C SPACE BAR	X 212	TO F

Again, the left field of the scroll area "reminds" you that you are converting to degrees F (Fahrenheit).

MASS, LENGTH, AND VOLUME CONVERSIONS

In mass, length, and volume conversions, an intermediate conversion is performed internally from the old units (the one you enter) to kilograms, meters, or fluid ounces. Then the message ENTER DESIRED UNITS is displayed. Suppose you want to find out how many kilograms it takes to make 3¼ pounds.

Type	Stack Display	Scroll Message
3.25 LB SPACE	X 3.25	ENTER NEW UNITS
KG SPACE	X 1.4741752	

The conversion groupings are as follows:

Function	**Description**	**Conversion Constants**
MASS		
KG	Kilograms	1 KG = 1 KG
GM	Grams	1 GM = .001 KG
OZ	Ounces (Av.)	1 OZ = .0283495231 KG (approx.)
LB	Pounds (Av.)	1 LB = .45359237 KG (approx.)
LENGTH		
M	Meters	1 M = 1 M
CM	Centimeters	1 CM = .01 M
KM	Kilometers	1 KM = 1000 M
IN	Inches	1 IN = .0254 M
FT	Feet	1 FT = .3048 M
YD	Yards	1 YD = .9144 M
MI	Miles (statute)	1 MI = 1609.344 M

You will notice that ounces (OZ) and pounds (LB) are listed under the MASS grouping. (This is correct as they are units of mass and are independent of the force of gravity. In engineering, a pound is generally used for force or weight. The pound-force is the force that gives a standard pound-mass an acceleration equal to the standard acceleration of gravity (32.1740 ft/sec/sec).)

VOLUME		
FLOZ	Fluid ounces	1 FL = FL
TSP	Teaspoons	1 TSP = .1666666667 FL (⅔)
TBSP	Tablespoons	1 TBSP = .5 FL
CUP	Cups	1 CUP = 8 FL
QT	Quarts	1 QT = 32 FL
GAL	Gallons (U.S.)	1 GAL = 128 FL
L	Liters	1 L = 33.81492266 FL (approx.)

If you do not choose a "new units" entry from the same grouping, you will get a message ERROR—UNIT MISMATCH and the result will be in the intermediate units. To convert the number in new units to different units, you must repeat the same process of entering old, then new units. All conversion constants given above are accurate to at least 9 digits. In most cases, the exact value can be expressed in fewer than 9 digits.

DEGREES ↔ RADIANS CONVERSIONS

CD SPACE BAR or **CDEG** SPACE BAR
CR SPACE BAR or **CRAD** SPACE BAR

CDEG assumes that the X register contains an angle in degrees and returns an angle in radians. CRAD takes an angle in radians and converts it to degrees. To remind you which is which, CDEG displays the message TO RAD and CRAD displays the message TO DEG. CDEG and CRAD are not affected by RAD and DEG.

To convert from degrees to gradians (400 GRAD = 360 DEG) divide by .9. To convert gradians (GRAD) to DEG, multiply by .9. The following problems illustrate these two conversions:

Type	**X Display**	
270 CDEG SPACE BAR	4.712389	

What is the angle in gradians whose sine is .3?

DEG SPACE BAR		
.3ASIN SPACE BAR	17.457603	
.9/=	19.397337	

What is 325 gradians in radians?

***.9=325** SPACE BAR	325	(grad to deg)
CDEG SPACE BAR	5.1050881	(deg to rad)

Appendix D gives a summary of all the weights and measurements and the factors by which you must multiply them to convert to the desired measurement or weight.

POLAR ↔ RECTANGULAR CONVERSIONS

PO SPACE BAR or **POLAR** SPACE BAR

Polar takes an angle Θ (theta) in the Y register using the current angular mode and a radius R in the X register and converts them from polar to rectangular coordinates. The y-coordinate is put in the Y register and the x-coordinate is put in the X register. The angle is entered first, then the radius. In ALG and ALGN modes, a PUSH (SHIFT [) must be performed after entering the angle to get it into the Y register. In RPN, a push is performed automatically after each number is entered. The message TO RECT Y=ANGLE, X=R TO Y, X will be displayed to remind you which value goes in which register. It means that an angle in Y and a radius in X have been converted to x and y rectangular coordinates in the X and Y registers. Be sure to set the angular mode correctly by entering DEG or RAD.

The procedure used for each calculation mode is as follows:

ALG and **ALGN**	**RPN**
DEG or RAD	DEG or RAD
Enter Θ	Enter Θ
PUSH or SHIFT [	
Enter radius	Enter radius
POLAR or PO	POLAR or PO

X register = x-coordinate
Y register = y-coordinate

This function uses the formula:

$$y = R*SIN(\Theta)$$
$$x = R*COS(\Theta)$$

Since SIN and COS are accurate to 7 digits (over most of their range), polar is also accurate to 7 digits in most cases.

If overflow occurs then ERROR—ARITHMETIC OVERFLOW will be displayed and the coordinate whose value overflowed will be set to 0.

To illustrate this function, convert R = 8, Θ = 60 degrees to rectangular coordinates.

Type	**Stack Display**		
CTRL CLEAR			
ALG SPACE BAR			
DEG SPACE BAR			
60 SPACE BAR	X		60
SHIFT [	X		60
	Y		60
8 SPACE BAR	X		8
	Y		60
POLAR SPACE BAR	X	=	4.0000001
	Y	=	6.9282033
RPN SPACE BAR			
DEG SPACE BAR			
60 SPACE BAR	X		60
	Y		4.0000001
8 SPACE BAR	X		8
	Y		60
POLAR SPACE BAR	X	=	4.0000001
	Y	=	6.9282033

To convert rectangular coordinates to polar coordinates, use the following function:

RECT SPACE BAR

Rectangular takes a y-coordinate in the Y register and an x-coordinate in the X register and converts them to an angle Θ in the Y register using the current angular mode and a radius R in the X register. Angle Θ ranges −Pi radians or −180 degrees to +Pi radians or +180 degrees. The x value is entered first. In ALG and ALGN modes a PUSH must be performed after entering the y value (see **POLAR**). The message TO POLAR Y, X TO Y=ANGLE, X=R will be displayed to remind you which value goes in which register. It means that values in the X and Y registers have been converted to an angle in the Y register and a radius in the X register.

The procedure for each calculation mode is as follows:

ALG & ALGN	**RPN**
DEG or RAD SPACE BAR	DEG or RAD
Enter y	Enter y
PUSH or SHIFT [	
Enter x	Enter x
RECT	RECT
x = R	
y = Θ	

This function uses:

$\Theta = \text{ATAN}\ (y/x)$
$R = 1\ x/\text{COS}\ (\Theta)1$

Since ATAN and COS are accurate to 7 digits (over most of their range), RECT is also accurate to 7 digits in most cases. Errors may accumulate if RECT and POLAR are applied repeatedly.

To illustrate this function, convert x = 20, y = 45 to polar coordinates with the angle in radians.

Type	**Stack Display**		**Comments**
CTRL CLEAR	X	0	
ALG SPACE BAR			
DEG SPACE BAR			
45 SPACE BAR	X	45	
SHIFT [	X	45	
	Y	45	
20 SPACE BAR	Y	20	
	Y	45	
RECT SPACE BAR	X	49.244289	Radius is displayed in X register.
	Y	66.037511	Angle Θ is displayed in Y register.

Type	**Stack Display**	
CTRL CLEAR	X	0
RPN SPACE BAR		
DEG SPACE BAR		
45 SPACE BAR	X	45
	Y	0
20 SPACE BAR	X	20
	Y	45
RECT SPACE BAR	X	49.244289
	Y	66.037511

5 FUNDAMENTAL FUNCTIONS

These functions plus the single-variable functions found in the next section are performed immediately upon entry. Each (with the exception of Pi) operates on the value in the X register and the result is immediately placed into the X register.

ABSOLUTE VALUE FUNCTION

A SPACE BAR or **ABS** SPACE BAR

This function makes a number positive.

Examples:

Type	X Display
CTRL CLEAR	0
4 CHGSGN SPACE BAR	−4
ABS SPACE BAR	4
0ABS SPACE BAR	0
1E-10A SPACE BAR	1E-10

CHANGE SIGN FUNCTION

CH SPACE BAR or **CHGSGN** SPACE BAR or SHIFT −

This function changes the sign of a number in the X register from positive to negative or from negative to positive. However, 0 is left unchanged if it is the number in the X register.

Type	X Display
CTRL CLEAR	
6CH SPACE BAR	−6

FRACTION FUNCTION

FR SPACE BAR or **FRAC** SPACE BAR

This function keeps the fractional part of a number. All digits before the actual decimal point location are removed. This function is equivalent to x-TRUNC(x).

Type	X Display
CTRL CLEAR	
4.5 SPACE BAR	4.5
FRAC SPACE BAR	0.5

INTEGER FUNCTION

INT SPACE BAR

This command takes the greatest integer less than or equal to that number.

Type	X Display
CTRL CLEAR	0
4.4 INT SPACE BAR	4
4.5 INT SPACE BAR	4
5.6CHGSGN SPACE BAR	−5.6
INT SPACE BAR	−6
2 INT SPACE BAR	2

PI FUNCTION

PI SPACE BAR

The value of Pi, computed to an accuracy of 8 digits, is displayed in the X register. Actually, Pi is internally computed to an accuracy of 9 digits. If you are in RPN mode, the previous value in the X register is pushed on the stack just as if you had entered a number.

Type	Stack Display	
CTRL CLEAR	X	0
ALG SPACE BAR	X	0
PI SPACE BAR	X	3.1415927
RPN SPACE BAR	X	3.1415927
4 SPACE BAR	X	4
	Y	3.1415927
PI SPACE BAR	X	3.1415927
	Y	4
	Z	3.1415927

RECIPROCAL FUNCTION

RE SPACE BAR or **RECIP** SPACE BAR

This function replaces a number with 1 divided by the same number. The reciprocal of 2 would be ½. If the number is 0, then the message ERROR—ARITHMETIC OVERFLOW is displayed on the screen and the result will appear as 0.

Type	X Display
CTRL CLEAR	0
2RE SPACE BAR	0.5
RE SPACE BAR	2

ROUND A NUMBER FUNCTION

ROU SPACE BAR or **ROUND** SPACE BAR

This command rounds a number to the nearest integer. For a positive number, if the fractional part is .5 or larger, then the number is rounded up and if the fractional part is less than .5, then the number is rounded down.

Type	Stack Display	
CTRL CLEAR	X	0
4.4ROU SPACE BAR	X	4
4.5ROU SPACE BAR	X	5
5.6 SHIFT –	X	−5.6
ROU SPACE BAR	X	−6
2ROU SPACE BAR	X	2

SQUARE FUNCTION

SQU SPACE BAR **SQUARE** SPACE BAR

This function computes the square of a number by multiplying the number by itself. If the absolute value of the number is equal to or greater than 10 to the 49th power, then the message ERROR—ARITHMETIC OVERFLOW will be displayed and the result will be 0.

Type	Stack Display		Comments
4.7063787 SPACE	X	4.7063787	Note you can square a fractional number easily.
SQU SPACE BAR	X	22.15	
5SQU SPACE BAR	X	25	

SQUARE ROOT FUNCTION

SQ SPACE BAR or **SQRT** SPACE BAR

This function takes the square root of a number. If the number is negative, the message ERROR—NUMBER OUT OF RANGE is displayed and the square root of the absolute value of the number is computed.

Type	Stack Display	
25SQ SPACE BAR	X	5
22.15SQ SPACE BAR	X	4.7063787

TRUNCATE FUNCTION

TRU SPACE BAR or **TRUNC** SPACE BAR

This command removes the fractional part from a number; that is, it keeps the integer part. All digits after the actual decimal point location are removed. Note that when floating point notation is used in the display, the decimal point is always shown to the right of the first digit. The fractional part is to the right of where the decimal point would be displayed if this notation were not used.

Type	Stack Display	
CTRL CLEAR	X	0
4.5TRU SPACE BAR	X	4
5.6 SHIFT −	X	−5.6
TRUNC SPACE BAR	X	−5
2TRU SPACE BAR	X	2
.001234567 SPACE BAR	X	1.234567E−03
TRU SPACE BAR	X	0

6

ALGEBRAIC AND TRIGONOMETRIC FUNCTIONS

This section describes the algebraic and trigonometric functions included in the CALCULATOR diskette program.

ALGEBRAIC FUNCTIONS

EXPONENTIATION BASE e FUNCTION

EX SPACE BAR or **EXPE** SPACE BAR

This command computes the natural antilogarithm of a number where e is approximately 2.7182818. If the absolute value of a number is greater than 255 (approximately), then ERROR—ARITHMETIC OVERFLOW will be displayed and the result will be 0.

EXPONENTIATION BASE 10 FUNCTION

EXPT SPACE BAR or **EXPTEN** SPACE BAR

This command computes the common antilogarithm of a number. If the absolute value of the number is greater than or equal to 98, then ERROR—ARITHMETIC OVERFLOW will be displayed and the result will be 0.

FACTORIAL FUNCTION

FA SPACE BAR or **FACT**SPACE BAR or SHIFT **!**

This command computes the factorial of a number using the formula x! = 1*2*3*...*(x−1)*x, which is the product of all integers from 1 to x (the number). If the number is negative, then ERROR—NUMBER OUT OF RANGE will be displayed and the factorial of the absolute value of the number will be computed. If the number is 0!, the factorial will be 1. If the number is not an integer, it will be rounded to the nearest integer (see **ROUND**) before the factorial is computed. If the number is greater than 68, then ERROR—ARITHMETIC OVERFLOW will be displayed and the result will be 0.

LOGARITHM BASE 10 FUNCTION

LOG SPACE BAR or **LOGTEN** SPACE BAR

This command computes the common logarithm (base 10) of a number. Error conditions are the same as for LN (see below). Note that ATARI BASIC uses LOG and CLOG instead of LN and LOGTEN. LN and LOG are used here to be consistent with common mathematical notation.

Type	X Display	
CTRL CLEAR	X	0
10		
LOG SPACE	X	1
SHIFT [	X	1
	Y	1

NATURAL LOGARITHM FUNCTION

LN SPACE BAR

This command computes the natural logarithm (base e) of a number. If the number is less than 0, then ERROR—NUMBER OUT OF RANGE will be displayed and the result will be the natural log of the absolute value of the number. If the number is equal to 0, then ERROR—ARITHMETIC OVERFLOW will be displayed and the result will be 0.

Type	Stack Display	
	X	10
10	Y	1
LN SPACE	X	2.3025851
	Y	1

POWER AND ROOT FUNCTIONS

POW SPACE BAR or **POWER** SPACE BAR or SHIFT ∧
RO SPACE BAR or **ROOT** SPACE BAR

The POWER function computes a number (y) to the xth power using the formula:

y∧x = EXPTEN(x*LOGTEN(y)).

The ROOT function computes the xth root of y using the formula:

y ROOT x = y∧(1/x).

For both the POWER and ROOT functions, if y is negative, then ERROR—NUMBER OUT OF RANGE will be displayed and the absolute value of y will be used to compute the result. If x and y are both exact positive integers (not just rounded to integers in the display), then the result of the POWER function will always be an exact integer. If y is an integer and x is a negative integer, then the result will be the reciprocal of an exact integer. Figure 8 shows what the result and error message (if any) will be for various values and −y indicates numbers less than 0.

y	x	y∧x	y∧x Error	yROOTx	yROOTx Error
0	0	1	—	1	1
0	x	0	—	0	—
0	−x	0	1	0	1
y	0	1	—	1	1
−y	0	1	2	1	1
+y	+x	y∧x	—	y∧(1/x)	—
+y	−x	1/(y∧x)	—	1(y∧(1/x))	—
−y	+x	y∧x	2	y∧(1/x)	2
−y	−x	y∧−x	2	1/(y(1/x))	2

Note: The dashes indicate no error unless magnitude of result is too large.
Error 1 indicates ARITHMETIC OVERFLOW.
Error 2 indicates NUMBER OUT OF RANGE.

Figure 8 Root and Power Table

As an example, raise 35 to the cube root of 5.3. In math books this would be written as $35^{\sqrt[3]{5.3}}$

Type	**X Display**
CTRL CLEAR	0
ALG SPACE BAR	0
35 SHIFT ∧	35.00
(5.3 **ROOT** 3) **1.7435134**	
=	492.15656

Since the ROOT and POWER functions are of equal precedence, the computer would perform the POWER function first if no parentheses were included. This answer is correct to 7 digits rounding up, since the eighth digit is 6.

As a second example, using the power and root functions and the exponentiation functions, compute 10∧(LN(5) + $e^{(-4.3)}$*LOG(97)).

Type	Stack Display	
CTRL CLEAR	X	0
ALG SPACE BAR	X	0
10 SHIFT ^	X	10
	Y	10
(5LN SPACE BAR	X	1.6094379
	Y	10
+	X	1.6094379
	Y	1.6094379
	2	10
4.3 SHIFT −	X	−4.3
	Y	1.6094379
	2	10
EXPE SPACE BAR	X	0.013568559
	Y	1.6094379
	2	10
*	X	0.013568559
	Y	0.013568559
	2	1.6094379
	3	10
97 SPACE BAR	X	97
	Y	0.013568559
	2	1.6094379
	3	10
LOG)	X	1.6363955
	Y	10
=	X	43.290791

In RPN, you would enter the problem as follows:

Type	Stack Display	
CLR SPACE BAR	X	0
RPN SPACE BAR	X	0
4.3 SHIFT −	X	−4.3
	Y	0
EXPE SPACE BAR	X	0.013568559
	Y	0

Type	Stack Display	
97 SPACE BAR	X	97
	Y	0.013568559
	2	0
LOG SPACE BAR	X	1.9867717
	Y	0.013568559
	2	0
*	X	0.02695763
	Y	0
5LN SPACE BAR	X	1.6094379
	Y	0.02695763
	2	0
+	X	1.6363955
	Y	0
EXPTEN SPACE BAR	X	43.290791

MODULO FUNCTION

MO SPACE BAR **MOD** SPACE BAR or SHIFT **%**

This function performs the MOD function using the formula:

yMODx = y − (x*INT(y/x))

This function is most often used with positive integers, but x and y may have any value (within range). The value returned by the function may be thought of as the remainder of y divided by x. If y is greater than or equal to 1E10, then ERROR—NUMBER OUT OF RANGE will be displayed and the result will be the original value of y.

An example of this function would be:

Type	X Display	Comments
CTRL CLEAR	0	
ALG SPACE BAR	0	
14MOD3=	2	
5 SHIFT −	−5	
MOD2=	1	
1E10MOD500=	1E+10	ERROR—NUMBER OUT OF RANGE
RPN SPACE BAR	1E+10	
7.6 SPACE BAR	7.6	
3 SHIFT **%**	1.6	

TRIGONOMETRIC FUNCTIONS

SINE, COSINE, AND TANGENT FUNCTIONS

SI SPACE BAR or **SIN** SPACE BAR

COS SPACE BAR

T SPACE BAR or **TAN** SPACE BAR

The sine, cosine, and tangent functions all assume that the value in the X register is an angle in radians if RAD mode is selected and in degrees if DEG mode is selected. If the angle is greater than or equal to 1E + 10 (10,000,000,000), then ERROR—NUMBER OUT OF RANGE will be displayed and the result will not be accurate. The tangent of +90 degrees, −90 degrees, PI/2 radians, and −PI/2 radians is undefined so ERROR—ARITHMETIC OVERFLOW will be displayed and the result will be meaningless (not necessarily 0). Like most of the other functions in this calculator, sine, cosine, and tangent are generally accurate to 7 digits. Near extreme points, such as near 90 degrees where the tangent is undefined, or very close to 0, there is a loss of accuracy. For example, the tangent of 89.99 degrees is only accurate to 4 digits.

ARC SINE, ARC COSINE, AND ARC TANGENT FUNCTIONS

AS SPACE BAR or **ASIN** SPACE BAR

AC SPACE BAR or **ACOS** SPACE BAR

AT SPACE BAR or **ATAN** SPACE BAR

These three functions are the inverse functions corresponding to sine, cosine, and tangent, respectively. They return an angle in degrees or radians, depending on the current mode. For ACOS and ASIN, if the absolute value of the number is greater than 1, then ERROR—NUMBER OUT OF RANGE will be displayed and the result will be meaningless (not necessarily 0).

Function	**Range of Result**	
SIN, COS	−1 less than or equal to x less than or equal to 1	
	Degrees	**Radians**
ASIN, ATAN	−90 < = x < = 90	−PI/2 < =x <PI/2
ACOS	0 < = x < =180	0 < = x < = PI

Type	**X Display**
CTRL CLEAR	0
DEG SPACE BAR	0
45 SPACE BAR	45

Type	X Display
SIN SPACE BAR	0.70710678
ASIN SPACE BAR	45
RAD SPACE BAR	45
PI/6 SHIFT −	−6
= SPACE BAR	−0.52359878
COS SPACE BAR	0.86602541
ACOS SPACE BAR	.52359878

The other trigonometric functions can be computed using the following functions:

Function	Abbreviation	Enter
Cotangent	COT	TAN RECIP
Cosecant	CSC	SIN RECIP
Secant	SEC	COS RECIP
Arc Cotangent	ARCCOT	RECIP ATAN
Arc Cosecant	ARCCSC	RECIP ASIN
Arc Secant	ARCSEC	RECIP ACOS

COMPUTING HYPERBOLIC FUNCTIONS

This CALCULATOR does not have hyperbolic functions built in; however, they may be calculated using the other functions. If you use them a lot, you could write subroutines to do them for you.

Function	Description	Derivation
SINH(x)	Hyperbolic SINe	(EXPE(x)−EXPE(−x))/2
COSH(x)	Hyperbolic COSine	(EXPE(x)+EXPE(−x))/2
TANH(x)	Hyperbolic TANgent	SINH(x)/COSH(x) or −EXPE(−x)/(EXPE(x)+EXPE(−x))*2+1
SECH(x)	Hyperbolic SECant	1/COSH(x) or 2/(EXPE(x)+EXPE(−x))
CSCH(x)	Hyperbolic CoSeCant	1/SINH(x) or 2/(EXPE(x)−EXPE(−x))
COTH(x)	Hyperbolic COTangent	1/TANH(x) or EXPE(−x)/(EXPE(x)−EXPE(−x))*2+1
ASINH(x)	Arc Hyp Sine	LN(x+SQRT(x*x+1))
ACOSH(x)	Arc Hyp COSine	LN(x+SQRT(x*x−1))
ATANH(x)	Arc Hyp TANgent	LN((1+x)/(1−x))/2
ASECH(x)	Arc Hyp SECant	LN((SQRT(−x*x+1)+1)/x)
ASCH(x)	Arc Hyp CoSeCant	x>=0LN(SQRT(x*x+1)+1)/x x<0LN(−SQRT(x*x+1)+1)/x
ACOTH(x)	Arc Hyp COTangent	LN((x+1)/(−1))/2

The following two formats illustrate a simpler method of calculating SINH(x) in both ALG and RPN.

Type

```
ALG((EXPE(x)−RECIP(x))/2
RPN EXP(x) PUSH RECIP(x)−2/
```

This method uses the fact that RECIP(x) = 1/EXPE(x). The value of EXPE(x) is needed twice, so in ALG mode the − is entered before taking the reciprocal, to get EXPE(x) into both the X and Y registers. Parentheses are used here so that SINH may be found in the middle of a computation. Otherwise, = could be used instead.

7

STATISTICS FUNCTIONS

STATISTICAL FUNCTIONS INCLUDING LINEAR REGRESSION

Before doing statistical operations you must enter the command **CLSTAT** SPACE BAR to clear memory registers 3-9 and display headings next to the memory numbers. These memory locations must not be used for other purposes while you are doing statistical calculations. These functions operate on two sets of variables, x and y. The numbers are entered by putting a y value in the Y register and an x value in the X register and issuing the command SPLUS which adds to the sums in various registers as shown below. Mistakes can be corrected by using SMINUS to remove unwanted number pairs. Once a set of numbers has been entered, a variety of statistical functions may be performed, including linear regression.

The Greek symbol Σ (Sigma) is used to indicate summation. Σ (x) is used here to indicate the sum of the x values. Σ (x*y) indicates that each x and y pair is multiplied together and the resulting products are summed together.

CLEAR MEMORY REGISTERS 3-9 FOR STATISTICS CALCULATIONS MODE

CLS SPACE BAR or **CLSTAT** SPACE BAR

The contents of memory locations 3, 4, 5, 6, 7, 8, and 9 are set to 0 and the following headings are displayed in the memory area of the screen.

Memory and Heading	Description
3NWT	N Weight. 0 means N weighting. −1 means N−1 weighting
4N	N = numbers of x and y entries
5X	(x)
6X*X	(x*x)
7Y	(y)
8Y*Y	(y*y)
9X*Y	(x*y)

DESCRIPTION OF FUNCTIONS

SIGMA PLUS FUNCTION

SP SPACE BAR or **SPLUS** SPACE BAR

This function takes the values in the X and Y registers and sums into memory locations 5–9 as specified on the preceding page. It increments memory location 4 (N) by 1 to indicate that one more pair of coordinates has been entered. The first step in using SPLUS is to enter the number to be placed in the Y register. In ALG and ALGN this value must be pushed from the X register to the Y register. In RPN, the PUSH is done automatically when the second value is entered. Next, enter the value for the X register. Finally, enter SPLUS or the abbreviation SP and press RETURN or SPACE BAR. You will see that the statistics memory locations have been changed. The X register value is removed from the stack and the original Y register value is now in the X register. If you wish to use the same y value for the next point, you do not need to reenter it. Simply do a PUSH (in ALG and ALGN) and enter the new value for the X register. Continue entering x and y values and using SPLUS until all of the points have been entered. Then you can use the functions described below to analyze your data. If you have only one variable, x, instead of both x and y variables, the best thing to do is to enter x, PUSH, and enter SPLUS. This uses the same value for x and y, so the value left in the X register is the original x, making it easy to enter the same number several times. The alternative is to enter x and SPLUS. The problem with this is that you will get ERROR—STACK EMPTY because the CALCULATOR expects two values on the stack. Also, the X register value will be x*x instead of x.

SIGMA MINUS FUNCTION

SM SPACE BAR or **SMINUS** SPACE BAR

This function is commonly used to correct mistakes when entering points using SPLUS. Simply reenter the incorrect values the same way you did for SPLUS. Then enter SMINUS or SM and press RETURN or SPACE BAR. The number of entries, N, will be decremented by 1 and the sums will be changed. If you discover the mistake immediately after entering SPLUS, then the y value will still be in the X register, and all you have to do is PUSH (in ALG and ALGN), enter the incorrect x, and SMINUS.

N WEIGHTING FUNCTION

NW SPACE BAR or **NWT** SPACE BAR

This function selects the weighting to be used for standard deviation and variance. N weighting and N−1 weighting are the ones that are commonly used. However, if you should have some reason to use a different weighting, you may do so. The value in the x register is stored in memory location 3 (NWT). This value is added to N to determine the weighting to be used. Since CLSTAT initializes NWT to 0, N weighting will be used until you issue an NWT command. A value of −1 for NWT will select N−1 weighting.

MEAN OF X FUNCTION

XM SPACE BAR or **XMEAN** SPACE BAR

This command computes the mean of the previously entered x values using the formula:

MEAN(x) = Σ (x) / N

and puts the result in the X register.

MEAN OF Y FUNCTION

YM SPACE BAR or **YMEAN** SPACE BAR

This command computes the mean of the previously entered y values using the formula:

MEAN(y) = Σ (y) / N

and puts the result in the X register.

VARIANCE OF X FUNCTION

XV SPACE BAR or **XVAR** SPACE BAR

This function computes the variance of the previously entered x values using the formula:

VAR(x) = (Σ(x*x) − SQUARE (Σ(x)) / N) / (N + NWT)

VARIANCE OF Y FUNCTION

YV SPACE BAR or **YVAR** SPACE BAR

This command computes the variance of the previously entered y values using the formula:

VAR(y) = (Σ (y*y)−SQUARE (Σ (y))/ N)/(N+NWT)

and the result is placed in the X register.

STANDARD DEVIATION OF X FUNCTION

XS SPACE BAR or **XSD** SPACE BAR

This command computes the standard deviation of x using the formula:

SD(x) = SQRT (VAR(x))

and the result is displayed in the X register.

STANDARD DEVIATION OF Y FUNCTION

YS SPACE BAR or **YSD** SPACE BAR

This command computes the standard deviation of y using the formula:

SD(y) = SQRT (VAR(y))

and the result is displayed in the X register.

SLOPE FUNCTION

SL SPACE BAR or **SLOPE** SPACE BAR

This function computes the slope of the line which has the closest fit to the x- and y-coordinates. The least squares method, which minimizes the sum of the squares of the distance of each point from the line, is used.

SLOPE = m = (Σ(x*y) − Σ(x)*Σ(y) / N) / (Σ(x*x) − SQU (Σ(x)) / N)

The result is displayed in the X register and the scroll area.

Y-INTERCEPT FUNCTION

YI SPACE BAR or **YINT** SPACE BAR

This function computes the y-intercept of the least squares fit line through the previously entered points. This is the value of y when x is 0.

YINT = b = (Σ(y) − m*Σ(x))/N

The result is displayed in both the X register and the scroll area.

CORRELATION COEFFICIENT FUNCTION

R SPACE BAR

The function computes the correlation coefficient, R, of the variables x and y. This is a measure of the linear dependence of x on y. The maximum magnitude of R is 1, which indicates complete linear dependence. A value of 0 for R indicates that there is no linear dependence of the two variables. However, they may be dependent in a nonlinear fashion.

R = m * SD(x) / SD(y)

RETURN Y GIVEN X AND RETURN X GIVEN Y FUNCTIONS

X SPACE BAR

Y SPACE BAR

X takes the value in the X register as the x-coordinate of a point on the least squares fit line through the previously entered points (see **SLOPE** and **YINT**). It computes the corresponding y-coordinate using the formula:

y = m * x + b

X displays the message TO Y and Y displays the TO X to remind you which is which.

The following example is a linear regression "word problem" illustrating the above functions:

Twenty students are given a homework assignment and are graded on a scale of 0 to 10. These grades are to be converted to letter grades using the definition that a B is from the mean to one standard deviation (SD) above the mean, an A is from one SD above to two SD's above and an A+ is anything above that. On the other end, a C is from the mean to one SD below the mean, a D is one SD below to two SD's below the mean; and an F is anything below that. Using the test scores shown below, find the mean and standard deviation and the distribution of letter grades.

Number of Correct Answers	Number of Students
0	0
1	0
2	1
3	1
4	3
5	5
6	4
7	3
8	1
9	1
10	1

Type	X Display		Comments
CTRL CLEAR	X	0	
ALG SPACE BAR	X	0	
CLSTAT SPACE BAR	X	0	
1 SHIFT [	X	1	
	Y	1	
SP SPACE BAR	X	1	
SHIFT [	X	1	Mistake in entry
	Y	1	
SMINUS SPACE BAR	X	1	Correct mistake
2 SHIFT [	X	2	
	Y	2	
SP SPACE BAR	X	2	
3 SHIFT [	X	3	
	Y	3	
SP SPACE BAR	X	3	
4 SHIFT [	X	4	
	Y	4	
SP SPACE BAR	X	4	
SHIFT [	X	4	
	Y	4	
SP SPACE BAR	X	4	
SHIFT [	X	4	
	Y	4	
SP SPACE BAR	X	4	
5 SHIFT [	X	5	
	Y	5	

Type	X Display		Comments
SP SPACE BAR	X	5	
SHIFT [	X	5	
	Y	5	
SP SPACE BAR	X	5	
SHIFT [	X	5	
	Y	5	
SP SPACE BAR	X	5	
SHIFT [	X	5	
	Y	5	
SP SPACE BAR	X	5	
SHIFT [	X	5	
	Y	5	
SP SPACE BAR	X	5	
6 SHIFT [	X	6	
	Y	6	
SP SPACE BAR	X	6	
SHIFT [	X	6	
	Y	6	
SP SPACE BAR	X	6	
SHIFT [	X	6	
	Y	6	
SP SPACE BAR	X	6	
SHIFT [	X	6	
	Y	6	
SP SPACE BAR	X	6	
7 SHIFT [	X	7	
	Y	7	
SP SPACE BAR	X	7	
SHIFT [	X	7	
	Y	7	
SP SPACE BAR	X	7	
SHIFT [	X	7	
	Y	7	
SP SPACE BAR	X	7	
8 SHIFT [	X	8	
	Y	8	
SP SPACE BAR	X	8	
9 SHIFT [	X	9	
	Y	9	
SP SPACE BAR	X	9	
10 SHIFT [	X	10	
	Y	10	
SP SPACE BAR	X	10	Finished entering numbers

Type	Stack Display		Comments
XMEAN +	X	5.7	
	Y	5.7	
XSD SPACE BAR	X	1.9	
	Y	5.7	
+	X	7.6	
	Y	7.6	
XSD =	X	9.5	
XMEAN −	X	5.7	
	Y	5.7	
XSD −	X	3.8	
	Y	3.8	
XSD =	X	1.9	

To use RPN instead of ALG you can enter the number segment in the same way you did above. But first, you need to change the Status Display to RPN.

Type	Stack Display		Comments
RPN SPACE BAR			Type **CLSTAT**. Enter numbers again. The number stack will have the numbers 10 through 1 in registers X through 9.
XMEAN			
SPACE BAR	X	5.6666667	Save XMEAN for later use.
SHIFT [	X	5.6666667	
	Y	5.6666667	
SHIFT [	X	5.6666667	
	Y	5.6666667	
	2	5.6666667	
XSD SPACE BAR	X	1.8601929	
	Y	5.6666667	
	2	5.6666667	
+	X	7.5268595	
	Y	5.6666667	
SHIFT [	X	7.5268595	
	Y	7.5268595	
	2	5.6666667	
XSD +	X	9.3870524	
	Y	5.6666667	
XSD −	X	3.8064738	
SHIFT [	X	3.8064738	
	Y	3.8064738	
XSD −	X	1.946281	

Note: The **PUSH** (SHIFT [ ) before each XSD is necessary because XSD is a function, not a number, so an automatic PUSH is not performed.

Grade	Score
A+	10
A	8-9
B	6-7
C	4-5
D	2-3
F	0-1

The following examples allow you to apply all of the built-in statistics functions to the set of 8 points using both N and N−1 weighting for standard deviation and variance. Remember to enter first the Y value, then the X value.

X	Y
−3.5	−5
−2	−4.3
.1	−2
3	3
5.6	4.4
10.3	9.1
12.5	11.2
20.0	17.9

Type	Stack Display		Comments
CTRL CLEAR			
ALG SPACE BAR			
CLSTAT SPACE BAR			
5 SHIFT −	X	−5	
SHIFT [	X	−5	
	Y	−5	
3.5 SHIFT −	X	−3.5	
	Y	−5	
SP SHIFT	X	−5	
4.3 SHIFT −	X	−4.3	
SHIFT [	X	−4.3	
	Y	−4.3	
2 SHIFT −	X	−2	
	Y	−4.3	
SP SPACE BAR	X	−4.3	
2 SHIFT −	X	−2	
SHIFT [	X	−2	
	Y	−2	
.1 SPACE BAR	X	0.1	
	Y	−2	
SP SPACE BAR	X	−2	
3 SHIFT [	X	3	
	Y	3	

Type	Stack Display		Comments
SP SPACE BAR	X	3	
4.4 SHIFT [	X	4.4	
	Y	4.4	
5.6 SPACE BAR	X	5.6	
	Y	4.4	
SP SPACE BAR	X	4.4	
9.1 SHIFT [	X	9.1	
	Y	9.1	
10.3 SP SPACE BAR	X	9.1	
11.2 SHIFT [	X	11.2	
	Y	11.2	
12.5 SPACE BAR	X	12.5	
	Y	11.2	
SP SPACE BAR	X	11.2	
17.9 SHIFT [	X	17.9	
	Y	17.9	
20 SPACE BAR	X	20	
	Y	17.9	
SP SPACE BAR	X	17.9	
XMEAN SPACE BAR	X	5.75	
XSD SPACE BAR	X	7.537075	
XVAR SPACE BAR	X	56.8075	
YMEAN SPACE BAR	X	4.2875	
YSD SPACE BAR	X	7.5618182	
YVAR SPACE BAR	X	57.181094	
SLOPE SPACE BAR	X	0.99908683	
YINT SPACE BAR	X	−1.4572493	
R SPACE BAR	X	0.9958177	
1 SHIFT −	X	−1	
NWT SPACE BAR	X	−1	N−1 weighting
XSD SPACE BAR	X	8.0574721	
XVAR SPACE BAR	X	64.922857	
YSD SPACE BAR	X	8.0839236	
YVAR SPACE BAR	X	65.349821	

To enter the numbers in RPN:

CLMEM SPACE BAR		
CTRL CLEAR		
RPN SPACE BAR		
CLSTAT SPACE BAR		
5 SHIFT −	X	−5
3.5 SHIFT −	X	−3.5
	Y	−5

Type	Stack Display		Comments
SP SPACE BAR	X	−5	
	X	−4.3	
4.3 SHIFT −	Y	−5	
2 SHIFT −	X	−2	
	Y	−4.3	
SP SPACE BAR	X	−4.3	
	Y	−5	
2 SHIFT −	X	−2	
	Y	−4.3	
.1 SPACE BAR	X	0.1	
	Y	−2	
SP SPACE BAR	X	−2	
	Y	−4.3	
3 SPACE BAR	X	3	
	Y	−2	
3 SPACE BAR	X	3	
	Y	3	
SP SPACE BAR	X	3	
	Y	−2	
4.4 SPACE BAR	X	4.4	
	Y	3	
5.6 SPACE BAR	X	5.6	
	Y	4.4	
SP SPACE BAR	X	4.4	
	Y	3	
9.1 SPACE BAR	X	9.1	
	Y	4.4	
10.3 SPACE BAR	X	10.3	
	Y	9.1	
SP SPACE BAR	X	9.1	
	Y	4.4	
11.2 SPACE BAR	X	11.2	
	Y	9.1	
12.5 SPACE BAR	X	12.5	
	Y	11.2	
SP SPACE BAR	X	11.2	
	Y	9.1	

What is x if y is 9.8?

Type	X Display
9.8Y SPACE BAR	11.267538 10.851547

Reenter the above program and compute y if x is −3.

The X register should display −4.4545097.

If you are entering coordinates with many digits which only differ in the last few digits, the variance will not be very accurate because the CALCULATOR is limited to 9 or 10 digits when storing the sums and computing the variance. To improve the accuracy, subtract a constant amount from each value as it is entered (or add for negative numbers). To find the mean, add the constant which you subtracted to the computed mean. The standard deviation and variance are computed in the normal way.

Suppose you have the following coordinates:

X	Y
3010987	1000001
3013900	1000004

Instead of entering all those "big" numbers, subtract 3010000 from each x and 1000000 from each y.

Type	Stack	Display	Comments
CLSTAT SPACE BAR			
CLX SPACE BAR	X	0	
ALG SPACE BAR			
1 SHIFT [	X	1	
	Y	1	
987 SPACE BAR	X	987	
	Y	1	
SP SPACE BAR	X	1	
4 SHIFT [	X	4	
	Y	4	
3900 SPACE BAR	X	3900	
	Y	4	
SP SPACE BAR	X	4	
5 SHIFT [	X	5	
	Y	5	

Type	Stack	Display	Comments
6706 SPACE BAR	X	6706	
	Y	5	
SP SPACE BAR	X	5	
XMEAN +	X	3864.3333	
	Y	3864.3333	
3010000 =	X	3013864.3	Actual Mean
XSD SPACE BAR	X	2334.9082	Accurate SD
YMEAN +	X	3.3333333	
	Y	3.3333333	
1000000 =	X	1000003.3	Actual Mean
YSD SPACE BAR	X	1.6996732	Accurate SD

The **PROGRAMMING INSTRUCTIONS AND EXAMPLES** section will show you how to write a short program that will scale the coordinates for you.

Other types of regression may be performed by transforming x or y or both before entering them. Different variations may be obtained by using log, root, power, reciprocal, or exponentiation. A semilogarithmic curve fit is obtained by taking the log of one of the variables.

For example, population growth is usually exponential, and can be modeled with the equation y = ae∧ (bx). This is equivalent to LN(y) = LN(a)*bx. A plot of x vs. LN(y) should give a straight line. This is the same as plotting x vs. y on semilog paper. Using the following population data for a town, project the population in 1980 and determine when the population will reach 55000. Determine the correlation coefficient for x and LN(y) to see how close a straight line the semilog curve is.

Year X	Population Y
1955	9305
1960	12036
1965	15398
1970	20801
1975	27509

Type	Stack	Display	Comments
CLSTAT SPACE BAR			
CLX SPACE BAR	X	0	
9305 SPACE BAR	X	9305	
LN SHIFT [	X	9.1383072	
	Y	9.1383072	
1955 SPACE BAR	X	1955	
	X	9.1383072	
SP SPACE BAR	X	9.1383072	
12036 SPACE BAR	X	12036	
	Y	9.3956574	

Type	Stack	Display	Comments
LN SHIFT [	X	9.3956574	
	Y	9.3956574	
1960 SPACE BAR	X	1960	
	Y	9.3956574	
SP SPACE BAR	X	9.3956574	
15398			
LN SHIFT [	X	9.6419929	
	Y	9.6419929	
1965 SPACE BAR	X	1965	
	Y	9.6419929	
SP SPACE BAR	X	9.6419929	
20801			
LN SHIFT [	X	9.9427563	
	Y	9.9427563	
1970 SPACE BAR	X	1970	
	Y	9.9427563	
SP SPACE BAR	X	9.9427563	
27509			
LN SHIFT [	X	10.222268	
	Y	10.222268	
1975 SPACE BAR	X	1975	
	Y	10.222268	
SP SPACE BAR	X	10.222268	
FIX SPACE BAR	X	10.222268	Computer requests ENTER 0–8.
0 SPACE BAR	X	10.	
1980 SPACE BAR	X	1980	
X SPACE BAR	X	10.	Computer displays TO Y.
EXPE SPACE BAR	X	35692	Projected population in 1980.
55000 LN SPACE BAR	X	11.	
Y SPACE BAR	X	1988	Computer displays TO X. Population will equal 55000 in 1988.
FIX SPACE BAR			Computer displays ENTER 0–8.
3 SPACE BAR	X	1987.963	
R SPACE BAR	X	9.993E-01	R = .9993

8

FINANCIAL FUNCTIONS

MODE OPTIONS

Before doing compound interest and annuity calculations, enter the **CLINT** SPACE BAR command to clear memory registers 4–9 and to display headings, then type **ENTER** SPACE BAR. Memories 4–9 should not be used for other purposes while doing compound interest calculations. This means that statistics and compound interest cannot be done at the same time unless the memory registers are saved and restored.

You can select the five types of interest computation: Compound Interest, Future Value of an Annuity Due, Future Value of an Ordinary Annuity, Present Value of an Annuity Due, or Present Value of an Ordinary Annuity. The abbreviations for these are CMPND, FVDUE, FVORD, PVDUE, and PVDRD, respectively. Values are input by typing **ENTER** SPACE BAR. This puts the CALCULATOR in ENTER (as opposed to FIND) mode. Type the value and enter the appropriate variable name; e.g., I. This causes this value to be stored in the appropriate memory location. After all the values have been entered, type **FIND** SPACE BAR followed by the unknown variable, e.g., PV, and the computed value will be displayed. Note that FIND I can only be used in Compound Interest mode.

CLEAR MEMORY LOCATIONS 4-9 FOR INTEREST CALCULATIONS MODE

CL SPACE BAR or **CLINT** SPACE BAR

The contents of memory locations, 4, 5, 6, 7, 8, and 9 are set to 0. The CALCULATOR is set to **ENTER** mode (see **ENTER** and **FIND**) and headings are displayed in the memory area of the screen.

Memory and Heading	**Description**
4BAL	BALloon payment
5FV	Future Value
6i	I/100 (interest rate per period as a fraction)
7N	Number of periods
8PMT	PayMenT
9PV	Present Value

SELECT ENTER MODE

ENT SPACE BAR or **ENTER** SPACE BAR

All subsequent BAL, FV, I, N, PMT, and PV statements will be used to ENTER values until the next **FIND** command is typed.

SELECT FIND MODE

FI SPACE BAR or **FIND** SPACE BAR

All subsequent BAL, FV, I, N, PMT, and PV statements will be used to FIND values until the next **ENTER** command is typed.

COMPOUND INTEREST

BALLOON PAYMENT FUNCTION

B SPACE BAR or **BAL** SPACE BAR

A balloon payment is sometimes made at the end of a loan to pay off the remainder of the loan. In **ENTER** mode, the value of the X register is stored in memory register 4 (BAL). In **FIND** mode (PVDUE and PVORD only), the balloon payment is computed using the values in the other registers and stored in the X register and memory register 4.

FUTURE VALUE FUNCTION

FV SPACE BAR

The Future Value is the value of the investment or loan at the end of the last period. In **ENTER** mode, the value in the X register is stored in memory location 5 (FV). In **FIND** mode, the Future Value is computed and stored in the X register and memory location 5.

INTEREST RATE PER PERIOD IN PERCENT FUNCTION

I SPACE BAR

In **ENTER** mode, i=x/100 is stored in memory location 6 (i). In **FIND** mode, I is computed and stored in the X register and i=1/100 is stored in memory location 6. When entering I (in ENTER mode), if interest is compounded quarterly, divide the nominal annual interest rate by 4 to get the interest rate per period. If interest is compounded monthly, divide the annual rate by 12. When finding I (in FIND mode), reverse the process and multiply by 4 or 12 to get the annual interest rate in percent.

NUMBER OF PERIODS FUNCTION

N SPACE BAR

In **ENTER** mode, X is stored in memory location 7, (N). In **FIND** mode, the number of periods is computed and stored in the register and memory location 7. When entering N, if interest is compounded quarterly for a number of years, multiply the number of years by 4 to get the number of periods. If interest is compounded monthly, or monthly payments are to be made, multiply the number of years by 12. When finding N, reverse the process, multiplying by 4 or 12 to convert the number of periods to years.

PAYMENT PER PERIOD FUNCTION

PM SPACE BAR or **PMT** SPACE BAR

In **ENTER** mode, the content of the X register is stored in memory location 8 (PMT). In **FIND** mode, the payment is computed and stored in the X register and memory location 8.

PRESENT VALUE FUNCTION

PV SPACE BAR

The Present Value is the value of the investment or loan at the *beginning* of the first period. In **ENTER** mode, the content of the X register is stored in memory location 9 (PV). In **FIND** mode, the Present Value is computed and stored in the X register and memory location 9.

Note that the Status Display has a FIX2 for BAL, FV, N, PMT, and PV so that dollars and cents will be displayed. In **ENTER** mode, these five variables are displayed in the memory area. In **FIND** mode, these five variables are displayed in the memory area and the computed value is also displayed in the scroll area. In **ENTER** mode, issuing the I command causes i = I/100 to be displayed in the memory area in FIX8 and I in the scroll area in FIX3. In **FIND** mode, i is displayed first in the scroll area in FIX8.

SELECT COMPOUND INTEREST MODE

CMP SPACE BAR or **CMPND** SPACE BAR

This mode is used for situations where a sum of money is invested, earns interest, and the interest is compounded to the account at the *end* of each interest period. This interest is now part of the principal and will earn more interest in the next period. Examples are a savings account in a bank or savings and loan and a share draft account in a credit union. Money is put in the account and left there to accumulate interest with no deposits or withdrawals. The values that may be entered and found are FV, I, N, and PV.

This mode uses the equation:

```
FV = PV * (1+1) ∧N
```

The following problems illustrate the method of computing interest.

1. How much money will be accumulated in a credit union share draft account if $8000 is deposited and left for 2 years at an annual interest rate of 7%, compounded quarterly? What if the money is placed in a bank savings account at 5¼ % interest, compounded quarterly?

Type	**Stack Display**		**Comments**
CLM SPACE BAR			Clear memory.
CTRL CLR			Clear stack.
CMPND SPACE BAR			
CLINT SPACE BAR			
8000 PV SPACE BAR	X	8000.00	FIX8 changes to FIX2 in status display. Present value also displayed at memory location 9.
2*	X	2.00	
	Y	2.00	
4=	X	8.00	Number of quarters
N SPACE BAR	X	8.00	Memory location 7
7/	X	7.00	
	Y	7.00	
4=	X	1.75	Interest rate per quarter
I SPACE BAR	X	1.750	
FIND SPACE BAR	X	1.750	
FV SPACE BAR	X	9191.05	
ENTER SPACE BAR	X	9191.05	
5.25/	X	5.25	
	Y	5.25	
4=	X	1.31	
I SPACE BAR	X	1.313	
FIND SPACE BAR	X	1.313	
FV SPACE BAR	X	8879.62	

The credit union account would have $9191.05 after two years and the bank account would contain $8879.62.

2. Using the same data, what annual interest rate is needed to have $10,000.00 at the end of the two years?

Type	**Stack Display**		**Comments**
ENTER SPACE BAR			
10000 FV SPACE BAR	X	10000.00	
FIND SPACE BAR			
I SPACE BAR	X	2.829	Rate per quarter
	X	2.829	
*	X	2.829	
4=	X	11.314	Annual interest rate

3. If the annual inflation rate from 1979 to 1982 is 10%, what is $10,000 in 1982 dollars worth in 1979 dollars?

Type	Stack Display		Comments
CTRL CLEAR			Clear stack.
CLMEM SPACE BAR			Clear memory.
CMPND SPACE BAR			
CLINT SPACE BAR			
10000 FV SPACE BAR	X	10000.00	Note FIX2 mode
10 I SPACE BAR	X	10.000	Annual inflation rate
3 N SPACE BAR	X	3.00	Number of years
FIND SPACE BAR	X	3.00	
PV SPACE BAR	X	7513.15	

What if the annual inflation rate is 13%?

Type	Stack Display		Comments
ENTER			
13	X	13.00	Annual inflation rate
I SPACE BAR	X	13.000	
FIND SPACE BAR			
PV SPACE BAR	X	6930.50	

The Annual Effective Rate (AER) of interest takes into account the compounding of interest. It tells what annual interest rate (in percent) with compounding annually is equivalent to the nominal annual rate with compounding done more often. This is based on the assumption that the interest is left in the account. The AER is always at least as large as the nominal annual interest rate. To compute the AER from the nominal annual interest rate, use the formula:

```
FV = 1 + AER/100 = PV * (1 + I/100) ΛN
AER = (FV−1) * 100
```

Plug in PV=1, N=number of periods per year, and I=nominal interest rate per period. Solve for FV. Then solve for the AER. Once you know the AER, you can use this for I and use the number of years for N, since each period is one year. The procedure is as follows:

```
CLINT
CMPND
Enter number of periods per year
N
1PV
Enter annual interest rate
/
Enter number of periods per year
=
I
FIND FV
−1=*100=
```

The following problems illustrate how to solve for AER:

1. If the nominal annual interest rate is 18% compounded monthly, what is the AER?

Type	Stack Display		Comments
CTRL CLEAR			
CLMEM SPACE BAR			
CLI SPACE BAR			
CMP SPACE BAR			
12 SPACE BAR	X	12	
N SPACE BAR	X	12.00	
1 SPACE BAR	X	1.00	
PV SPACE BAR	X	1.00	
18/	X	18.00	
	Y	18.00	
12=	X	1.50	
I SPACE BAR	X	1.500	
FIND SPACE BAR	X	1.500	
FV SPACE BAR	X	1.20	
–	X	1.20	
	Y	1.20	
1=	X	1.96E-01	
*	X	1.96E-01	
	Y	1.96E-01	
100=	X	19.56	
FIX SPACE BAR	X	19.56	
8 SPACE BAR	X	19.561816	Annual Effective Rate

2. To find the nominal annual rate from the AER use the following procedure which is similar to the procedure used to find AER:

```
CLINT
CMPND
Enter periods per year
N
1PV
1+
Enter AER
/100=FV
FIND I       (rate per period)
*
Enter periods per year
=            (nominal annual rate)
```

Reverse the calculation in the previous AER example.

Type	Stack Display		Comments
CTRL CLEAR			
CLMEM SPACE BAR			
CLI SPACE BAR			
CMP SPACE BAR			
12 SPACE BAR	X	12.000	
N SPACE BAR	X	12.00	
1 SPACE BAR	X	1.00	
PV SPACE BAR	X	1.00	Stored in memory location 9
1+	X	1.00	
	Y	1.00	
19.561816/	X	19.56	
	Y	19.56	
	2	1.00	
100=	X	1.20	
FV SPACE BAR			Stored in memory location 5
FIND SPACE BAR			
I SPACE BAR	X	1.500	Rate per period
*	X	1.500	
	Y	1.500	
12=	X	18.000	Nominal annual rate

Note: If your answer does not agree with your bank's answer, it may be that they are using a different number of periods per year or are putting the interest into a different account.

The formula for continuous compounding for one year is:

FV = PV * e ∧ (I/100) = 1 + AER/100

where I is the nominal interest rate per period in percent. The AER may be computed from this and used in subsequent calculations. Continuous compounding may also be approximated by using a large number of periods per year.

3. As an example, compute the AER if the nominal annual rate is 9.255% and interest is compounded continuously.

Type	Stack Display		Comments
CLMEM SPACE BAR			
9.255/	X	9.2555	
	Y	9.2555	
100	X	0.9255	
EXPE SPACE BAR	X	1.096968	
—			
1	X	0.09696797	
*			
100=	X	9.696797	

ANNUITIES

The word annuity is used here for a situation where fixed payments are made each period and interest is compounded at the end of each period. In an annuity due, payments are made at the beginning of each period. In an ordinary annuity, payments are made at the end of each period.

FUTURE VALUE OF AN ANNUITY DUE FUNCTION

FVD SPACE BAR or **FVDUE** SPACE BAR

An example of an annuity due is a savings account where equal payments are made at the beginning of each interest period. Selecting future value means that I, N, and PMT will be calculated using the value in the FV register, not the PV register (Present Value). The values that may be entered are FV, I, N, and PMT. The values that may be found are FV, N, and PMT. PV may also be computed in this mode. In this case, the formula for PVDUE will be used, so if BAL is not 0, a balloon payment will be included.

FV = PMT * ((1 + i) ΛN - 1) * (1 + i) / i

As an example, compute how much money will be accumulated in a credit union share draft account after two years if $1000 is deposited at the beginning of each quarter and the annual interest rate is 7%, compounded quarterly.

Type	Stack	Display	Comments
CTRL CLEAR			
CLINT SPACE BAR			
FVDUE SPACE BAR			
2*	X	2.000	
	Y	2.000	
4=	X	8.000	
N SPACE BAR	X	8.00	Placed in memory location 7
1000 SPACE BAR	X	1000.00	
PMT SPACE BAR			Placed in memory location 8
7/	X	7.00	
	Y	7.00	
4=	X	1.75	
I SPACE BAR	X	1.750	Placed in memory location 6
FIND SPACE BAR			
FV SPACE BAR	X	8656.41	

What if the money is in a bank account at 5¼%?

Type	Stack	Display	Comments
ENTER	X	8656.41	
5.25/	X	5.25	
	Y	5.25	
4=	X	1.31	

Type	Stack Display		Comments
I SPACE BAR	X	1.313	
FIND SPACE BAR			
FV SPACE BAR	X	8487.26	

Note: Compare this with the first example in CMPND.

FUTURE VALUE OF AN ORDINARY ANNUITY FUNCTION

FVO SPACE BAR or **FVORD** SPACE BAR

An ordinary annuity is similar to an annuity due except that payments are made at the end of each period rather than at the beginning. An example is a sinking fund which is a savings fund that will accumulate a specific amount of money at a future date. The values that may be entered are FV, I, N, and PMT. The values which may be computed are FV, N, and PMT. PV may also be computed while in FVORD mode. However, the equation will be the one used by PVORD, which includes a balloon payment.

FV *5 PMT * ((1 + i) ∧N −1) / i

Examples:

1. Using the same example as for FVDUE on the preceding page, this time make the deposits at the end of each quarter.

Type	Stack Display		Comments
CTRL CLEAR			
CLM SPACE BAR			
CLI SPACE BAR			
FVO SPACE BAR			
2*	X	2.00	
	Y	2.00	
4=	X	8.00	
N SPACE BAR			Stored in memory location 7
1000			
PMT SPACE BAR	X	1000.00	Stored in memory location 8
7/	X	7.00	
	Y	7.00	
4=	X	1.75	
I SPACE BAR	X	1.750	
FIND SPACE BAR			
FV SPACE BAR	X	8507.53	Future value stored in memory location 5
ENTER			
5.25/	X	5.25	
	Y	5.25	

Type	Stack Display		Comments
4=	X	1.31	
I SPACE BAR	X	1.313	
FIND SPACE BAR			
FV SPACE BAR	X	8377.31	

2. What do the monthly payments have to be to accumulate $100,000 in 25 years at 9% annual interest?

Type	Stack Display		Comments
CTRL CLEAR			
CLM SPACE BAR			
CLI SPACE BAR			
FVO SPACE BAR			
100000 SPACE BAR	X	100000	
FV SPACE BAR	X	100000.00	Automatic FIX2: contents of X register stored in memory location 5.
25*	X	25.00	
	Y	25.00	
12=	X	300.00	
N SPACE BAR			Stored in memory location 7
9/	X	9.00	
	Y	9.00	
12=	X	7.50E-01	
I SPACE BAR	X	7.500E-01	i computed and stored in memory location 6
FIND SPACE BAR			
PMT SPACE BAR	X	89.20	Stored in memory location 8

PRESENT VALUE OF AN ANNUITY DUE FUNCTION

The difference between this mode and FVDUE is that PV is used in the calculations rather than FV. This is an annuity where payments are made over a fixed period of time. A balloon payment may be made at the end of the last period. The values which may be found are BAL, N, PMT, I, N, PMT, and PV. FV may also be found in PVDUE mode, however you will use the formula from FVDUE, which does not include a balloon payment.

PV = PMT * (1 −(1+i) ΛN) / i * (1+i) + BAL * (1+i) Λ−N

The following problem solves for monthly payments using PVDUE mode:

A company leases its equipment, which costs $52,000, to a customer for 3 years and then sells it for $15,000. An annual yield of 25% is desired. What should the lease payments be (paid at the beginning of each month)?

Type	Stack Display		Comments
CTRL CLEAR			
CLM SPACE BAR			
CLI SPACE BAR			
PVD SPACE BAR			
52000 SPACE BAR	X	52000.00	Automatic FIX2
PV SPACE BAR	X	52000.00	Stored in memory location 9
3*	X	3.00	
	Y	3.00	
12=	X	36.00	Stored in memory location 7
N SPACE BAR			
15000 BAL	X	150000.00	
25/	X	25.00	
	Y	25.00	
12=	X	2.08	
I SPACE BAR	X	2.083	i computed and stored in memory location 6
FIND SPACE BAR			
PMT SPACE BAR	X	1747.21	Stored in memory location 8

What should the monthly payments be if they sell the equipment for $12,000?

Type	Stack Display		Comments
ENTER SPACE BAR			
12000 SPACE BAR	X	12000.00	
BAL SPACE BAR	X	12000.00	Stored in memory location 4
FIND SPACE BAR			
PMT SPACE BAR	X	1802.83	

PRESENT VALUE OF AN ORDINARY ANNUITY FUNCTION

PVO SPACE BAR or **PVORD** SPACE BAR

This mode is useful for doing computations involving loans. The Present Value is the amount of the loan, PMT is the amount of money to be paid each month (or other period of time), N is the number of periods, and I is the interest rate per period. A balloon payment may be made at the end to pay off the remainder of the loan. The values that may be entered are BAL, FV, I, N, and PMT. The values that may be found are BAL, FV, N, and PMT. PV may also be completed, however, the formula from FVORD will be used so no balloon payment will be considered.

PV = PMT * (1 −(1 + i) ∧N) / i + BAL * (1 + i) ∧−

The following problem solves for monthly payments using PVORD mode.

What will the monthly payments be on a $150,000 loan for 25 years if the annual interest rate is 13%? What if the loan runs for 50 years?

Type	Stack Display		Comments
CTRL CLEAR			
CLM SPACE BAR			
CLI SPACE BAR			
PVO SPACE BAR			
150000 SPACE BAR	X	150000.00	
PV SPACE BAR	X	150000.00	Stored in memory location 9
25*	X	25.00	
	Y	25.00	
12=	X	300.00	
N SPACE BAR			Stored in memory location 7
13/	X	13.00	
	Y	13.00	
12=	X	1.08	
I SPACE BAR	X	1.083	i computed and stored in memory location 6
FIND SPACE BAR			
PMT SPACE BAR	X	1691.75	25-year loan payment is stored in memory location 8.
ENTER			
50*	X	50.00	
	Y	50.00	
12=	X	600.00	
N SPACE BAR			Stored in memory location 7
FIND SPACE BAR			
PMT SPACE BAR	X	1627.53	50-year loan payment is stored in memory location 8.

The payments for the 25-year loan, although it runs for half the time of the 50-year loan, are not very much larger. This is because most of the money for the 50-year loan goes into interest payments rather than reducing the principal.

9
PROGRAMMING INSTRUCTIONS AND EXAMPLES

MODE SELECT

Now that you are at least acquainted with the functions of the CALCULATOR program, you are now ready to learn the instructions that will allow you to use these functions in programs. A program is a series of *logical* instructions to be executed or "run" later. A program is said to be entered in *indirect* mode because the instructions are not performed immediately as they are entered. Until now, all of your examples have been executed in *direct* mode because each operation was performed immediately.

In program (or indirect) mode, most instructions are stored in the program memory, which contains 3072 locations (bytes). The address of the first location is 0000 and the last is 3071.

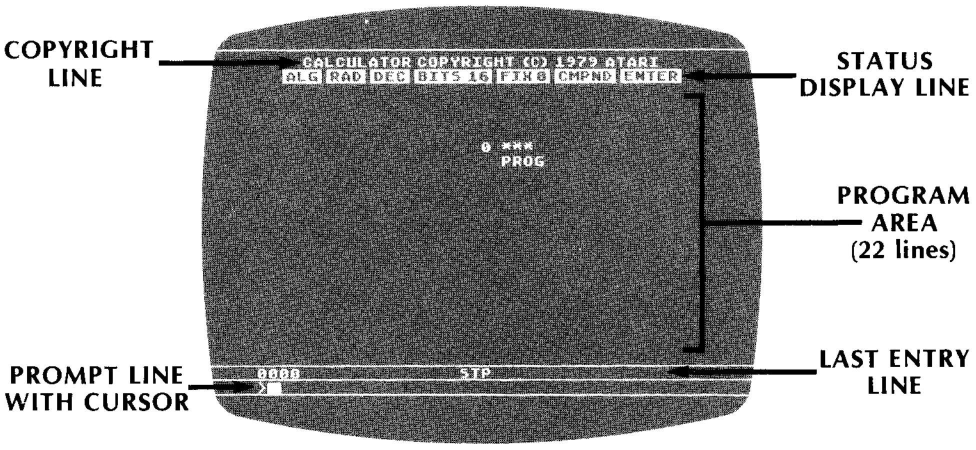
Note: This address is not the same as the real RAM address. The real RAM address varies.

A pointer, called the Program Counter (PC), displays the location that is currently being accessed. After you enter each instruction, the program counter is advanced. When you finish entering your program by typing the **END** instruction you enter direct mode and can execute your program. But right now you need to know the instructions that allow you to write a program.

ENTER PROGRAM MODE INSTRUCTION

PRO SPACE BAR or **PROG** SPACE BAR or SHIFT #

When you enter this instruction, the entire display scrolls upward, expanding the scroll area from 7 lines to 21 lines. This allows you to display 20 lines of your program at a time.



```
CALCULATOR COPYRIGHT (C) 1979 ATARI
ALG RAD DEC BITS 16 FIX 8 CMPND ENTER

                        0 ***
                          PROG

 0000              STP
>
```

Figure 9 Program Mode Screen Display

In the programs included in this section, the format will be shown as follows:

SCROLL AREA

Type	**Address Location**	**Old Contents**	**New Contents**	**Comments**
SYSTEM RESET				Clears number stack, scroll area, and returns status display line to default options.
SHIFT #	0000	STP		Enter program mode. Scroll area is now 21 lines and the first location 0000 is displayed. The STP instruction is in the "Old Contents" field as it is for every location before you enter an instruction.

This PRO instruction is not stored in the program memory and is only executed in the program mode. Other instructions that have the same limitation are listed in Figure 10.

Instruction	**Abbreviation**	**Token**
Back Step	BST	CTRL ↑
Clear Program	CLP or CLPROG	
Delete	DEL	CTRL DELETE BACK S
Disk Operating System	DOS	
End Program	END	SHIFT $
Insert Characters	INS	CTRL INSERT
Insert Number	INSNUM	CTRL →
List Program	LIST	
Load Program	LOAD	
Load Memory	LOADM	
Single Step	SST	CTRL ↓
Reset	RST	SHIFT "
Save Program	SAVE	
Save Memory	SAVEM	

Figure 10 Instructions Not To Be Stored in Memory

These instructions are discussed in the succeeding paragraphs.

CLEAR PROGRAM MEMORY INSTRUCTION

CLP SPACE BAR or **CLPROG** SPACE BAR

Use this instruction when you are ready to enter a new program. It fills the entire program memory with **STP** (stop) instructions and sets the program counter to 0000, so you can begin a new program.

Note: If you want to keep a previous program, make sure you have saved it on a diskette or cassette tape before entering a **CLPROG** instruction (see **PERIPHERAL INPUT/OUTPUT COMMANDS**).

Write a program to solve the problem 3 + 2 =. Use ALG mode.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
CLP SPACE BAR	0000	STP	CLPROG	
CLR SPACE BAR	0000	STP	CLR	Clears program of accumulated results.
ALG SPACE BAR	0001	STP	ALG	An instruction uses only one location so the next address will be 0002.
3 SPACE BAR	0002	STP	3	A number entry uses eight locations so the next address will be 0010.
+	0010	STP	+	
2 SPACE BAR	0011	STP	2	
=	0019	STP	=	
	0020	STP		

END PROGRAM MODE INSTRUCTION

E SPACE BAR or **END** SPACE BAR or SHIFT **$**

This instruction returns the screen to the direct mode in which commands are executed immediately. (Refer to Figure 2.)

Note: If you press BREAK immediately after entering an END instruction, the display will be messed up. If this does happen, enter **PROG** SPACE BAR and **END** SPACE BAR to straighten out the display.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0020	STP		
SHIFT **$**				This instruction is not stored in program memory. It takes you back to direct mode.

Now you are back in direct mode.

RU SPACE BAR or **RUN** SPACE BAR or SHIFT '

This instruction clears the *Call Stack* (See **CALL SUBROUTINE AT LOCATION n**) and executes the current program in memory beginning with address location 0000. You cannot enter ordinary commands while the program is running, but you can press the BREAK key to stop your program. If an error occurs during program execution, you'll see an error message and your program will stop.

Type

SHIFT '

You should have a 5 in the X register and in the scroll area.

LIST PROGRAM INSTRUCTION

LIST SPACE BAR

This instruction lists the requested address locations and contents of those locations.

SCROLL AREA

Type	**Comments**
SHIFT #	Return to program mode.
LIST SPACE BAR	Enter the first location to be listed.
ENTER 0-3071	
0 SPACE BAR	Enter the last location to be listed.
ENTER 0-3071	
20 SPACE BAR	

You should now see your program listed on the screen in program mode. The program counter is at location 0020—the last address location you requested. Note that this line is displayed twice.

On larger programs, you may only want to list part of the program. Make sure you do not enter a starting location that is in the middle of a number entry. If you do, you will get several locations listed and then the message ERROR—NOT VALID COMMAND OR NUMBER. If the ending location is in the middle of a number, then the program will display all the locations of the number and the program counter will point to the next instruction. In program mode, the last requested location is listed twice. This happens because the current location is always displayed after each command. If you want to list only one instruction, enter that instruction's location as the first and second requested addresses. Or you can type that instruction's address as the first requested address and 0 as the second requested address.

BACK STEP INSTRUCTION

BS SPACE BAR or **BST** SPACE BAR or CTRL ↑

Entering this instruction moves the program counter back to the previous instruction's address location and displays the location and its contents. However, it never executes the instruction in either direct mode or program mode. If the program counter is at address location 0 and you enter a BST instruction, the screen will display the message ERROR—END OF MEMORY and the program counter will remain at address location 0.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0020	STP		
CTRL ↑	0020	STP	BST	The preceding location is displayed with its contents on the current entry line. This instruction is not placed in program memory.
	0019	=		

SINGLE STEP INSTRUCTION

SS SPACE BAR or **SST** SPACE BAR or CTRL ↓

In program mode, this instruction moves the program counter to the next instruction's location and displays both the address location and its contents, but does not execute it. If the content is not a number, the program counter moves ahead one address location.

If the content is a number, the program counter moves ahead eight locations. In direct mode, entering **SST** SPACE BAR will both display and execute the instruction.

If the program counter is pointing to the last location in memory and you try to enter a single step instruction, the screen will display the message ERROR—END OF MEMORY and the program counter will not move.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0019	=		
CTRL ↓	0019	=	SST	The next location and its contents are displayed on the current entry line. This instruction is not placed in program memory.
	0020	STP		

In direct mode, it is possible to execute your program one step at a time using this instruction.

INSERT CHARACTER(S) INSTRUCTION (ONE BYTE)

INS SPACE BAR or CTRL INSERT

This instruction is used to make room for a new instruction to be inserted in the middle of a program. When you enter an **INS,** all the instructions from the current program counter location move ahead one address location (one byte). This instruction also inserts an **STP** (stop) instruction at the current address location. The instruction at the last address location (3071) is lost.

Suppose you want to enter a PUSH command to put a 3 in both the X and Y registers.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0020	STP		
CTRL ↑	0020	STP	BST	Moves back one address location
	0019	=		
CTRL ↑	0019	=	BST	Moves back eight address locations
	0011	2		

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
CTRL ↑	0011	2	BST	
	0010	+		
INS SPACE BAR	0010	+	INS	
	0010	STP		Clears address location
SHIFT [	0010	STP	PUSH	
	0011	+		

Now, to see the change, enter:

LIST SPACE BAR
ENTER 0-3071
0
ENTER 0-3071
20

The screen should show the following:

```
0000    CLR
0001    ALG
0002    3
0010    PUSH
0011    +
0012    2
0020    =
```

When you execute this program, you will have a 5 in the X register and a 3 in the Y register.

INSERT NUMBER INSTRUCTION (EIGHT BYTES)

INSN SPACE BAR or **INSNUM** SPACE BAR or CTRL →

Use this instruction to make room for a number to be inserted in the middle of a program. All succeeding instructions will be moved *eight* address locations. Therefore, if you have an instruction that requires a number entry (like a CALL or GOTO), but you do not know what the number will be, go ahead and enter a 0 so the program will leave room for a number entry. This will save you many later modifications.

In your current program, you decide to add another number so the problem will read 3+2*6=.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0020	=		
INS SPACE BAR	0020	=	INS	
	0020	STP		Clears address location
*	0020	STP	*	
	0021	=		
INSN SPACE BAR	0021	=	INSNUM	
	0021	STP		Clears address location
6 SPACE BAR	0021	STP	6	
	0029	=		

Now list the changed program:

Type	Address Location	Old Contents	New Contents
LIST SPACE BAR	0029	=	LIST
ENTER 0-3071			
0 SPACE BAR			
ENTER 0-3071			
30 SPACE BAR			

PROGRAMMING INSTRUCTIONS

The program now looks like the following:

```
0000    CLR
0001    ALG
0002    3
0010    PUSH
0011    +
0012    2
0020    *
0021    6
0029    =
0030    STP
```

Type

SHIFT **$**
CLR SPACE BAR
RUN SPACE BAR

You should have a 15 in both the X register and in the scroll area and a 3 in the Y register.

NO OPERATION INSTRUCTION

NOP SPACE BAR

This instruction is used in a program to allow room to add commands later or to delete commands without moving the address locations of the rest of the instructions. In direct mode, you can use a NOP command to terminate a command which requires an input. If, for instance, you have entered FIX and decide not to change the current option, type **NOP.** You'll get an error message, but the Status Display line remains unchanged.

In some cases, however, using a NOP instruction does not just take up space and leave everything as it was. For instance, in ALG mode, you cannot use two binary operations in a row or you get an error. If you want to repeat the value that is in the X register, you can substitute a NOP command as in the problem 3+(3+2)*6=. This is the same as writing 3+(NOP+2)6=. To program this change:

	SCROLL AREA			
Type	**Address Location**	**Old Contents**	**New Contents**	**Comments**
SHIFT #	0031	STP		
LIST SPACE BAR				
ENTER 0-3071				
12 SPACE BAR				Displays address location 12 and its contents

PROGRAMMING INSTRUCTIONS

Type	Address Location	Old Contents	New Contents
ENTER 0–3071			
12 SPACE BAR			
	0012	2	
INS SPACE BAR	0012	2	INS
	0012	STP	
SHIFT (	0012	STP	(
	0013	2	
INS SPACE BAR	0013	2	INS
	0013	STP	
NOP SPACE BAR	0013	STP	NOP
	0014	2	
INS SPACE BAR	0014	2	INS
	0014	STP	
+	0014	STP	+
	0015	2	
SS SPACE BAR	0015	2	SST
	0023	*	
INS SPACE BAR	0023	*	INS
	0023	STP	
SHIFT)	0023	STP	)
	0024	*	

Now, list the program to make sure you have the same data on your screen as is listed below:

Type

LIST

ENTER 0–3071

0 SPACE BAR

ENTER 0–3071

33 SPACE BAR

```
0000     CLR
0001     ALG
0002     3
0010     PUSH
0011     +
0012     (
0013     NOP
0014     +
0015     2
0023     )
0024     *
0025     6
0033     =
```

Now, run the program.

Type

SHIFT **$**
CLR SPACE BAR
RUN SPACE BAR

You should have a 33 in the X register and in the center field of the scroll area, and a 3 in the Y register.

DELETE CURRENT INSTRUCTION COMMAND

DEL SPACE BAR or CTRL DELETE BACK S

This instruction deletes the current instruction pointed to by the program counter. The succeeding instructions each move *back* one address location. If the deleted instruction is a number, succeeding instructions move back *eight* locations (eight bytes). STP instructions will be automatically added at the end of memory.

Remember that the numbers referring to address locations following a GOTO or CALL command will not be automatically modified to reflect the address location change. You must remember to change it manually (see **CALL** and **GOTO** instructions).

To delete the PUSH command from your program, enter the following:

Type	Address Location	Old Contents
SHIFT #		
	0035	STP

The PUSH instruction is at address location 0010, so list that line and delete it.

	SCROLL AREA			
Type	**Address Location**	**Old Contents**	**New Contents**	**Comments**
LIST SPACE BAR				
ENTER 0-3071				
10 SPACE BAR				
ENTER 0-3071				
10 SPACE BAR	0010	PUSH		
CTRL DELETE BACK S	0010	PUSH	DEL	Deletes PUSH instruction
	0010	+		

The + that was in address location 0011 is now in 0010 and the rest of the instructions have moved up accordingly.

List the program from 0 through 32.

STOP PROGRAM INSTRUCTION

STP SPACE BAR

As you have seen, the program memory is initially filled with STP instructions. As you entered the new instruction, it appeared in the right field. When you listed the program, the STP instruction had been replaced by the new instruction. But this STP instruction has other uses. If on a CALL or GOTO instruction you make a mistake and enter an address location *outside* your program, the STP instruction in that location will prevent your program from becoming a "runaway." The program will simply stop.

This instruction is also used in the middle of a program to stop execution and wait for you to input data (a number, in most cases). When you execute the program and it reaches the STP instruction, you will hear a "beep" (pitched higher than the error beep) and the prompt symbol appears.

You've used specific numbers in your program until now. Your program is only good to solve the one specific problem 3+(NOP+2)*6. By using STP instructions in place of the numbers, we can generalize the program to accommodate any numbers written as a general equation:

A+(NOP+B)*C

Back to the program and begin by listing location 2.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
LIST SPACE BAR				
ENTER 0-3071				
2 SPACE BAR				
ENTER 0-3071				
2 SPACE BAR				
	0002	3		
DEL SPACE BAR	0002	3	DEL	You *must* delete the number with a DELETE command before doing an INS.
	0002	+		
INS SPACE BAR	0002	+	INS	
	0002	STP		
STP SPACE BAR	0002	STP	STP	Puts a STP command in place of 3.
	0003	+		
SS SPACE BAR	0003	+	SST	
	0004	(		
SS SPACE BAR	0004	(	SST	
	0005	NOP		
SS SPACE BAR	0005	NOP	SST	
	0006	+		
SS SPACE BAR	0006	+	SST	
	0007	2		

Type	Address Location	Old Contents	New Contents	Comments
DEL SPACE BAR	0007	2	DEL	
	0007	)		
INS SPACE BAR	0007	STP	INS	Puts a STP command in place of 2.
	0007	)		
STP SPACE BAR	0007	STP	STP	
	0008	)		
SS SPACE BAR	0008	)	SST	
	0009	*		
SS SPACE BAR	0009	*	SST	
	0010	6		
DEL SPACE BAR	0010	6	DEL	
	0010	=		
INS SPACE BAR	0010	=	INS	Puts a STP command in place of the 6.
	0010	STP		
STP SPACE BAR	0010	STP	STP	
	0011	=		
SS SPACE BAR	0011	=	SST	
	0012	STP		

This completes the changes in the program. But before you can execute it, you need to know how to restart it after you have entered a number.

CONTINUE PROGRAM FROM CURRENT LOCATION INSTRUCTION

CON SPACE BAR or **CONT** SPACE BAR or SHIFT @

When the program counter reaches a stop (STP) instruction, it stops executing the program. After you have entered your number, press SHIFT @ to continue execution. List the first 12 locations to verify your program matches the above example.

Type

LIST SPACE BAR
ENTER 0–3071
0 SPACE BAR
ENTER 0–3071
12 SPACE BAR
END SPACE BAR
RUN SPACE BAR

Enter the following set of numbers: 2, 4, 8.

Note: Don't forget to press SHIFT @ after you enter *each* number. Otherwise, you could spend hours wondering why the program accepted the numbers, but didn't solve the problem.

When your program finishes executing, the answer of 50 appears in the X register and in the scroll area.

PROGRAMMING INSTRUCTIONS

GOTO INSTRUCTION

GO SPACE BAR or **GOTO** SPACE BAR

This instruction, followed by an address location number, is called an unconditional branching instruction. In your program, when the program counter comes to a GOTO instruction, it immediately jumps to the address location specified by the location following the GOTO statement. When you enter a GOTO instruction in direct mode, the screen displays the message ENTER 0–3071. You enter an address location and press the SPACE BAR. In program mode you of course do not get the prompt message. By putting a GOTO 2 at the end of your program, you can create an endless loop. The program will execute, waiting for you to supply the values and press SHIFT @. If you insert a PUSH command before the GOTO command, you can push your answers down into the stack.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
SHIFT #	0013	STP		Puts you in program mode
SHIFT [	0013	STP	PUSH	Each program solution will be saved in the stack.
	0014	STP		
GOTO SPACE BAR	0014	STP	GOTO	Program will return to address location 0002
	0015	STP		
2 SPACE BAR	0015	STP	2	
	0023	STP		

Run the program three times using the following sets of numbers:

1) 27, 16, 32	Answer is 1403
2) 1E+08, 2006, 1E+06	Answer is 1.0000211E+14
3) 3.456, 76.1, .0037	Answer is 3.7503572

When you have run the program three times, press SHIFT # to return to the program mode.

CONDITIONAL BRANCHING INSTRUCTIONS

XE SPACE BAR or **XEQ** SPACE BAR
XG SPACE BAR or **XGE** SPACE BAR
XL SPACE BAR or **XLT** SPACE BAR
XN SPACE BAR or **XNE** SPACE BAR

In a conditional branch, the GOTO portion of the instruction is not performed unless the first part of the instruction is true. For instance, the first conditional branch listed above is XE. It simply means if the number in the X register is equal to a number in a specified memory location, then you jump (or branch) to another address location.

Example:

XEQ	Testing to see if number in the X register is the same as the
1	number in memory location 1. If it is, the program branches to
22	address location 22.

To modify your program to incorporate this change, make the following changes:

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0003	+		Use back step instructions to get back to location 0000.
BST SPACE BAR	0003	+	BST	
	0002	STP		
BST SPACE BAR	0002	STP	BST	
	0001	ALG		
BST SPACE BAR	0001	ALG	BST	
	0000	CLR		
INSN SPACE BAR	0000	CLR	INSNUM	INSN used to insert number (58)
	0000	STP		
58 SPACE BAR	0000	STP	58	Now you've got a number to store in memory
	0008	CLR		
INS SPACE BAR	0008	CLR	INS	Use INS to enter STORE instruction.
	0008	STP		
STO SPACE BAR	0008	STP	STO	Enter STORE command.
	0009	CLR		
INSN SPACE BAR	0009	CLR	INSNUM	Use INSN to enter which memory location.
	0009	STP		
1 SPACE BAR	0009	STP	1	
	0017	CLR		

Single step to the STP instruction at address location 0029.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0029	STP		
DEL SPACE BAR	0029	STP	DEL	
	0029	PUSH		
SS SPACE BAR	0029	PUSH	SST	
	0030	GOTO		
INS SPACE BAR	0030	GOTO	INS	
	0030	STP		

Type	Address Location	Old Contents	New Contents	Comments
XE SPACE BAR	0030	STP	XEQ	Conditional branching instructions
	0031	GOTO		
INSN SPACE BAR	0031	GOTO	INSNUM	
	0031	STP		
1 SPACE BAR	0031	STP	1	
	0039	GOTO		
INSN SPACE BAR	0039	GOTO	INSNUM	
	0039	STP		
0 SPACE BAR	0039	STP	0	Enter a 0 because you don't know the program's last location yet.
	0047	GOTO		

Now that the address locations have changed, you'll have to change the location after the GOTO instruction; otherwise, the program will loop back in the middle of a number and you'll get an error.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
SS SPACE BAR	0047	GOTO	SST	
	0048	2		
17 SPACE BAR	0048	2	17	Enter new GOTO address location (17)
	0056	STP		

Now you know exactly where the program ends. You can now back step to address 40 and put in the correct number.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
BST SPACE BAR	0056	STP	BST	Remember, BST's are not stored in program memory. This is the number you need to change.
	0048	17		
BST SPACE BAR	0048	17	BST	
	0047	GOTO		
BST SPACE BAR	0047	GOTO	BST	
	0039	0		
56 SPACE BAR	0039	0	56	Entering 56 puts it in the new contents field and, on the next LIST, it will appear in place of the 0.
	0047	GOTO		

PROGRAMMING INSTRUCTIONS

Type **LIST** SPACE BAR and the numbers 0 and 56 upon request. The listing should look like this:

```
0000    58
0008    STO
0009    1
0017    CLR
0018    ALG
0019    STP
0020    +
0021    (
0022    NOP
0023    +
0024    STP
0025    )
0026    *
0027    STP
0028    =
0029    PUSH
0030    XEQ
0031    1
0039    56
0047    GOTO
0048    17
0056    STP
```

Now you have a program that stores a number into a memory location, checks your totals against that number and, if they don't match, loops back for another set of values. If the result in the X register *does* match the number in the memory location, the program stops. You can now put in a "test" to make sure the program does stop.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0056	STP		
22222 SPACE BAR	0056	STP	22222	Enter a test number to store in memory location.
	0064	STP		
STO SPACE BAR	0064	STP	STO	
	0065	STP		
0 SPACE BAR	0065	STP	0	
	0073	STP		

Now press SHIFT **$** to return to direct mode. Run the program using the following sets of values when the prompt symbol appears.

6, 9, 3	(First run)
2, 5, 8	(Second time through the program)

The first set of numbers did not total 58, so the program counter ignored the address of the **XE** instruction, dropped through to the GOTO instruction and looped back.

The second time through, the total is 58. Therefore, a 22222 appears in memory location 0 and the X register.

The other conditional instructions work very much the same way.

XGE states that if the number in the X register is greater than or equal to the number stored in a specified memory location, the program counter will go to the address location you have specified.

XLT states that if the number in the X register is less than the number stored in a specified memory location, the program counter will go to the address location you have specified.

XNE states that if the number in the X register is *not* equal to the number stored in a specified memory location the program counter will go to the address location you have specified.

The format for entering these three conditional branching instructions is the same as for the XEQ instruction.

CALL SUBROUTINE AT LOCATION n INSTRUCTION

CA SPACE BAR or **CALL** SPACE BAR

This instruction is very similar to a GOSUB instruction in BASIC in that it calls a subroutine. It requests that a program memory address (n) from 0–3071 be entered (if you're in direct mode). If you're in program mode, you enter the CALL instruction and at the next address location enter the number for the program memory address location.

When the program is executing and the program counter reaches a CALL instruction, the address of the instruction *following* the CALL's n is stored in a special stack known as the Call Stack. This Call Stack holds this number as a return address so that when the subroutine has completed execution, the program counter will know what location to go back to in the program. This Call Stack can contain up to 64 return addresses, so you can have 64 subroutines, each calling another. If you try to enter a 65th return address, you'll get an error message, ERROR—STACK FULL. The Call Stack is not visible on the CALCULATOR display.

RETURN FROM SUBROUTINE INSTRUCTION

RET SPACE BAR or **RETURN** SPACE BAR

This instruction is used to let the CALCULATOR know when it has reached the end of a subroutine. When the CALCULATOR "reads" this instruction, it pops the last CALL instruction off the Call Stack and stores it in the program counter. In direct mode the program counter is simply restored to the value it had before the CALL was issued. A message ERROR—STACK EMPTY is displayed if the Call Stack is empty when a Return is executed. In other words, you entered an RET instruction without a matching CALL instruction.

PROGRAMMING INSTRUCTIONS

You can change the first three lines of your program so that it will become a subroutine.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
SHIFT #				Enter program mode.
LIST SPACE BAR				
ENTER 0–3071				
0 SPACE BAR				
ENTER 0–3071				
0 SPACE BAR	0000	58		
DEL SPACE BAR	0000	58	DEL	Delete number entry.
	0000	STO		Moves contents up.
CALL SPACE BAR	0000	STO	CALL	
	0001	1		
0 SPACE BAR	0001	1	0	Enter a 0 because you don't know where the subroutine is going to be yet.
	0009	CLR		

Now list the program from 0 to 70. If everything is where it should be, then go on. If not, recheck your entries.

Now list line 65.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0065	STP		
SHIFT]	0065	STP	POP	This gets the 22222 out of the X register.
	0066	STP		
SHIFT [	0066	STP	PUSH	This pushes your total into the X register.
	0067	STP		

Single step to location 70 to begin subroutine entry.

SCROLL AREA

Type	Address Location	Old Contents	New Contents
	0070	STP	
58 SPACE BAR	0070	STP	58
	0078	STP	

Type	Address Location	Old Contents	New Contents
STO SPACE BAR	0078	STP	STO
	0079	STP	
1 SPACE BAR	0079	STP	1
	0087	STP	
RET SPACE BAR	0087	STP	RETURN
	0088	STP	

Since your address locations have changed, you must change the address references on your CALL and GOTO instructions.

List location 1 and make the necessary changes.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0001	0		
70 SPACE BAR	0001	0	70	
	0009	CLR		
LIST SPACE BAR				List address location 31.
ENTER 0-3071				
31 SPACE BAR				
ENTER 0-3071				
31 SPACE BAR				
	0031	56		
48 SPACE BAR	0031	56	48	
	0039	GOTO		
SS SPACE BAR	0039	GOTO	SST	
	0040	17		
11 SPACE BAR	0040	17	11	
	0048	22222		

List 0 through 90 to ensure you made the correct changes by typing the **LIST** command.

LIST SPACE BAR
ENTER 0-3071
0 SPACE BAR
ENTER 0-3071
90 SPACE BAR

PROGRAMMING INSTRUCTIONS

The listing on your screen should match the following.

```
0000        CALL
0001        70
0009        CLR
0010        ALG
0011        STP
0012        +
0013        (
0014        NOP
0015        +
0016        STP
0017        )
0018        *
0019        STP
0020        =
0021        PUSH
0022        XEQ
0023        1
0031        48
0039        GOTO
0040        11
0048        22222
0056        STO
0057        0
0065        POP
0066        PUSH
0067        STP
0068        STP
0069        STP
0070        58
0078        STO
0079        1
0087        RETURN
0088        STP
0089        STP
0090        STP
```

To run the program:

Type

SHIFT **$**
CLMEM SPACE BAR
RUN SPACE BAR

Use the values 2, 5, and 8. If a 22222 appears in memory location 0, the program worked perfectly.

POP CALL STACK INSTRUCTION

POPC SPACE BAR

This instruction is similar to the POP instruction. It pops one return address out of the Call Stack and discards it. Use it when you have more than one subroutine in a program—if one subroutine is inside another subroutine.

PROGRAMMING INSTRUCTIONS

If your program counter is working through the steps of an "inner" subroutine and you don't want to go back to the "outer" subroutine, then you program a POPC and the counter will discard the return to the first (outer) subroutine address and the program counter will go directly to the main program. If you have three subroutines "nested" one within another, the POPC will discard the address to the subroutine immediately preceding the one on which the program counter is working.

To see how this instruction works, you'll have to enter another subroutine. This second subroutine will allow you to test a number against a number in a memory location to see whether or not the program counter should return to the first subroutine or to the main program.

SCROLL AREA

Type	**Address Location**	**Old Contents**	**New Contents**	**Comments**
SHIFT #				
LIST SPACE BAR				
ENTER 0–3071				
78 SPACE BAR				
ENTER 0–3071				
78 SPACE BAR				
	0078	STO		
INS SPACE BAR	0078	STO	INS	
	0078	STP		
CALL SPACE BAR	0078	STP	CALL	Beginning of second (inner) subroutine.
	0079	STO		
INSN SPACE BAR	0079	STO	INSNUM	
	0079	STP		
0 SPACE BAR	0079	STP	0	Put a 0 in to hold eight bytes (actual number to be inserted later).
	0087	STO		
SS SPACE BAR	0087	STO	SST	
	0088	1		Store 58 + contents of address 0107 in memory location 1.
SS SPACE BAR	0088	1	SST	
	0096	STP		
RET SPACE BAR	0096	STP	RETURN	End of first (outer) subroutine.
	0097	STP		
STP SPACE BAR	0097	STP	STP	Put a STP to keep subroutines from "running into" each other.
	0098	STP		
+	0098	STP	+	Will add a number you enter to the number in the X register.
	0099	STP		In execution, insert number you want to enter at address 0107.

Type	Address Location	Old Contents	New Contents	Comments
11111 SPACE BAR	0099	STP	11111	Indicator to show you must enter a number after the next instruction.
	0107	STP		
STP SPACE BAR	0107	STP	STP	You will enter a number here that will determine whether the subroutine goes back into first subroutine or main program.
	0108	STP		
=	0108	STP	=	This adds the numbers in the X and Y registers.
	0109	STP		
STO SPACE BAR	0109	STP	STO	Stores the sum of those 2 numbers in memory location 3.
	0110	STP		
3 SPACE BAR	0110	STP	3	
	0118	STP		
65 SPACE BAR	0118	STP	65	Puts a 65 in X register.
	0126	STP		
XLT SPACE BAR	0126	STP	XLT	Tests to see if the 65 in the X register is less than the number stored in memory location you specify.
	0127	STP		
3 SPACE BAR	0127	STP	3	Specify you want to compare X register with memory location 3. Must be the same location specified at 0110.
	0135	STP		
0 SPACE BAR	0135	STP	0	If true, you'll send the program counter to a location further in the program.
	0143	STP		
RETURN SPACE BAR	0143	STP	RETURN	If false, return to first subroutine.
	0144	STP		
STP SPACE BAR	0144	STP	STP	Separator.
	0145	STP		
85 SPACE BAR	0145	STP	85	Put 85 in X register. This is the jump point for procounter.
	0153	STP		
STO SPACE BAR	0153	STP	STO	Store 85 in memory location.
	0154	STP		
1 SPACE BAR	0154	STP	1	Specify memory location 1
	0162	STP		
POPC SPACE BAR	0162	STP	POPC	Pops the Call Stack which will skip the first subroutine and go back to main program when RET is executed.
	0163	STP		
RET SPACE BAR	0163	STP	RETURN	
	0164	STP		

PROGRAMMING INSTRUCTIONS

Before you can run this program, you still have to insert the proper number entries at 0079 and 0135. List address location 0079.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
LIST SPACE BAR				
ENTER 0-3071				
79 SPACE BAR				
ENTER 0-3071				
79 SPACE BAR				
	0079	0		
98 SPACE BAR	0079	0	98	
	0087	STO		
LIST SPACE BAR				List address location 135.
ENTER 0-3071				
135 SPACE BAR				
ENTER 0-3071				
135 SPACE BAR				
	0135	0		
145 SPACE BAR	0135	0	145	
LIST SPACE BAR	0143	RETURN		List the entire program.
ENTER 0-3071				
0 SPACE BAR				
ENTER 0-3071				
165 SPACE BAR				

Your list should look like this.

```
0000        CALL
0001          70
0009         CLR
0010         ALG
0011         STP
0012           +
0013           (
0014         NOP
0015           +
0016         STP
0017           )
0018           *
0019         STP
0020           =
0021        PUSH
0022         XEQ
0023           1
0031          48
0039        GOTO
0040          11
```

```
0048        22222
0056          STO
0057            0
0065          POP
0066         PUSH
0067          STP
0068          STP
0069          STP
0070           58
0078         CALL
0079           98
0087          STO
0088            1
0096       RETURN
0097          STP
0098            +
0099        11111
0107          STP
0108            =
0109          STO
0110            3
0118           65
0126          XLT
0127            3
0135          145
0143       RETURN
0144          STP
0145           85
0153          STO
0154            1
0162         POPC
0163       RETURN
0164          STP
0165           ON
```

DISPLAY PROGRAM TRACE INSTRUCTION

TR SPACE BAR or **TRACE** SPACE BAR

This instruction displays the program memory address location, the contents of each location, and all results in the scroll area during execution. It also updates the stack and memory displays continuously. As the program you have entered now contains subroutines and the counter is going to be jumping around a bit, you can use the Trace Instruction to see exactly where you are in the program.

Type

SHIFT **$**
TRACE SPACE BAR
RUN SPACE BAR

If you enter a number less than 7 at address location 0107, you'll see a 65 in memory location 1 and 58 plus the number you entered at memory location 3. The program compares the two, and "sees" that the 65 is larger.

Then the program counter returns to the first subroutine at line 87, completes that path, then returns to the main program at address location 0009. Now, you are ready to enter your three entries. If they total 65, you will see 22222 in memory location 0. If not, the program performs the GOTO and returns to address location 0011. Then you enter the second set of numbers.

Example:

Type

6 SHIFT @ at address location 0107
3 SHIFT @ at address location 0011
2 SHIFT @ at address location 0016
7 SHIFT @ at address location 0019

This does not total 65, so the program returns to address location 0011.

Type

5 SHIFT @ at address location 0011
1 SHIFT @ at address location 0016
10 SHIFT @ at address location 0019

This does total 65, so the 22222 appears in memory location 0.

Now, to see if the other path works properly, you'll enter a number larger than 7 at address location 0107.

Type

CLM SPACE BAR
RUN SPACE BAR
12 SHIFT @ at address location 0107

This causes the program counter to jump from address location 0135 to 0145 and to store 70 (58 plus the 12 you entered) in memory location 3 and an 85 in memory location 1. It then performs the POPC which skips the first subroutine and returns directly to address location 0009, but remember, the number set must now total 85.

Type

9 SHIFT @ at address location 0011
2 SHIFT @ at address location 0016
14 SHIFT @ at address location 0019

This total is too large, so the program counter loops back to wait for another number set to be entered.

Type

5 SHIFT @ at address location 0011
3 SHIFT @ at address location 0016
10 SHIFT @ at address location 0019

This totals 85, so 22222 appears in memory location 0.

PROGRAMMING INSTRUCTIONS

TURN OFF TRACE (NO TRACE) INSTRUCTION

NOT SPACE BAR or **NOTRC** SPACE BAR

This instruction turns off the Trace function and the program will run at normal speed. When the computer is initialized, it is in No Trace.

Type

NOT SPACE BAR

PAUSE INSTRUCTION

P SPACE BAR or **PAUSE** SPACE BAR

This instruction causes the program to stop for ½ second. You can use this instruction to see what is happening in your program without stopping it completely. For instance, you could put a Pause Instruction at address location 0126 to look at the total of the number you entered and the 58 in the program.

SCROLL AREA

Type	**Address Location**	**Old Contents**	**New Contents**	**Comments**
SHIFT #	0067	STP		Enter program mode.
LIST SPACE BAR				
ENTER 0-3071				
126 SPACE BAR				
ENTER 0-3071				
126 SPACE BAR				
	0126	XLT		
INS SPACE BAR	0126	XLT	INS	
	0126	STP		
P SPACE BAR	0126	STP	PAUSE	The insert causes the rest of the address locations to move down one location.
	0127	XLT		
LIST SPACE BAR				
ENTER 0-3071				You can single step to the 0136 address location if you choose.
136 SPACE BAR				
ENTER 0-3071				
136 SPACE BAR				
	0136	145		
146 SPACE BAR	0136	145	146	
	0144	RETURN		

RESET INSTRUCTION

RST SPACE BAR or SHIFT "

This is one of the instructions (Figure 10) that is not held in program memory. You use this instruction to set the program counter back to address location 0000 and clear the Call Stack.

SCROLL AREA

Type	Address Location	Old Contents	New Contents	Comments
	0144	RETURN		
SHIFT "	0144	RETURN	RST	You still have a RETURN at this location, not a RESET.
	0000	CALL		

If you put a CLM instruction at location 0000, you won't have to type it in each time you restart the program. Then, so you won't have to change any more address locations, delete the STP from location 68 as shown below:

SCROLL AREA

Type	Address Location	Old Contents	New Contents
	0000	CALL	
INS SPACE BAR	0000	CALL	INS
	0000	STP	
CLM SPACE BAR	0000	STP	CLMEM
	0001	CALL	
LIST SPACE BAR			
ENTER 0-3071			
68 SPACE BAR			
ENTER 0-3071			
68 SPACE BAR			
	0068	STP	
DEL SPACE BAR	0068	STP	DELETE
	0068	STP	

PROGRAMMING INSTRUCTIONS

The final program should look like the following listing. Check yours against the listing by pressing CTRL **1** to stop and start the program.

```
0000        CLMEM
0001         CALL
0002           70
0010          CLR
0011          ALG
0012          STP
0013            +
0014            (
0015          NOP
0016            +
0017          STP
0018            )
0019            *
0020          STP
0021            =
0022         PUSH
0023          XEQ
0024            1
0032           48
0040         GOTO
0041           11
0049        22222
0057          STO
0058            0
0066          POP
0067         PUSH
0068          STP
0069          STP
0070           58
0078         CALL
0079           98
0087          STO
0088            1
0096       RETURN
0097          STP
0098            +
0099        11111
0107          STP
0108            =
0109          STO
0110            3
0118           65
0126          XLT
0127            3
0135          145
0143       RETURN
0144          STP
0145           85
0153          STO
0154            1
0162         POPC
0163       RETURN
0164          STP
0165          STP
0166          STP
0167          STP
```

The next part of this section contains program examples demonstrating the instructions you have learned in this section plus the functions and commands in the previous sections.

PROGRAMMING EXAMPLES USING FUNDAMENTAL FUNCTIONS

EXAMPLE 1. IN SUMMATION...

This program does no more than add a string of input numbers and subtotal it to memory location 0. You can use it to balance your checkbook, compute total resistance, or just add long columns of numbers with a little less effort.

This program will add whatever plus or minus numbers you enter to memory location 0. It will keep track of how many numbers you have entered in memory location 1, and will total to memory location 3 and clear the accumulated amount out of memory location 0 anytime the number "99999" (five 9's) is entered. To continue a new accumulation to memory location 0 after subtotaling, type in:

CONT SPACE BAR or **CON** SPACE BAR or SHIFT @.

```
IN SUMMATION

              Type          Comments

0000           ALG
0001           CLR
0002         CLMEM
0003         99999
0011           STO  ┐      ┌ Set terminator recognition
0012            99  ┘      └ in memory 99.
0020           STP ───────── Wait for input
0021          PUSH         ┌ Discontinue
0022           XEQ  ┐      │ accumulation to
0023            99  ├──────┤ memory 0 if
0031            76  ┘      └ input is 99999.
0039             1 ──┐
0047           SUM   └────── Add 1 to entry count.
0048             1
0056           POP
0057           SUM
0058             0
0066          GOTO  ┐
0067            20  ┘─────── Loop
0075           STP
0076           RCL  ┐      ┌ Get accumulated sum from
0077             0  ┘      └ memory location 0.
0085           SUM  ┐        Add it to
0086             3  ┘─────── memory location 3
0094           CLX
0095           STO
0096             0
0104           STP
0105          GOTO
0106            20
0114           STP
```

EXAMPLE 2. IN ALL PROBABILITY, I'M READY FOR VEGAS

This problem calculates the probability of your rolling two 4's in five tries using one die (a half of a pair of dice). To do this, the program uses the factorial and power functions and is written in RPN.

The equation you use to solve this problem is:

$$P(r) = \frac{r!}{u!v!w!}\left(\frac{a^u b^v c^w}{n^r}\right)$$

$$= \frac{r!}{u!v!w!}\left(\frac{a}{n}\right)^u\left(\frac{b}{n}\right)^v\left(\frac{c}{n}\right)^w$$

where r is the number of rolls of the die—in this case 5,

n is the number of faces or sides on the die (6),

a is the number of faces you're interested in—only one, because you're trying for a 4,

b is the remaining number of faces in which you're not interested—in this case, the other 5,

u is the number of possible outcomes for all rolls (within your predefined interest range): You want to know the probability of rolling two 4's, so this number is 2,

v is the number of other outcomes for all rolls (3),

c and w are 0 since in this problem we are not interested in any third possibility.

So, the probability of rolling two 4's and three of anything else in a total of five rolls would be:

$$P(r) = \left(\frac{5!}{2!3!}\right)\left(\frac{1}{6}\right)^2\left(\frac{5}{6}\right)^3$$

$$= \frac{120(125)}{12(36)(216)}$$

$$= \frac{15000}{93312}$$

$$= .16075102$$

As you can see, the probability is **NOT** in your favor.

The example that follows illustrates how to program this problem.

```
IN ALL PROBABILITY, I'M READY FOR VEGAS

              Type               Comments

0000           RPN
0001         CLMEM
0002           CLR
0003             5 ------------- Number rolls
0011          PUSH ------------- Copy into Y register
0012          FACT ------------- Also generate factorial
0013         XCHGY ------------- Swap X & Y registers
0014             6 -┐   ┌------- Calculate (5/6)^3
0022             /  |---┘
0023             3  |
0031         POWER -┘
0032             1 -┐
0040             6  |
0048             /  |----------- Calculate (1/6)^2
0049             2  |
0057         POWER -┘
0058             * -┐----------- Calculate 5! (1/6)^2 (5/6)^3
0059             * -┘
0060             3 -┐----------- 3!
0068          FACT -┘
0069             2 -┐----------- 2!
0077          FACT -┘
0078             * ------------- (2!) (3!)
0079             / ------------- Result
0080           STP ------------- Calculation complete
```

PROGRAMMING EXAMPLES USING ALGEBRAIC AND TRIGONOMETRIC FUNCTIONS

EXAMPLE 3. OFF ON A TANGENT

This program illustrates the trigonometric function, tangent.

One of the things that many people say about mathematics above add, subtract, multiply, and divide, is "Sure, trigonometry (for example) is fine in school, but what do you do with it in everyday life?" Unless you are an engineer or scientist, you probably haven't seen a sine or cosine since then—not that you thought of them as trig, anyway.

Suppose you have a house, and nearby is a large (and heavy) tree that must be removed. Your problem is that no matter which way it falls, if it is tall (long) enough, it could land on something—like your house, or your neighbor's house.

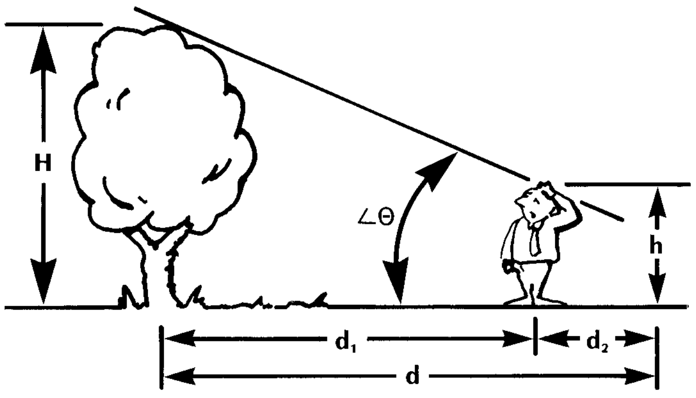
You could hire someone to remove it for you, but that is expensive and they also charge by the size of the tree.

But you can measure the ground distance from the tree to the closest structure in any direction. Say you measure the distance and it is 50 feet, give or take a few inches.

But how tall is the tree? It's not too practical to climb it with a tape measure in hand.

You walk a distance of 38 feet from the tree (for example). From where you are standing, use a level, a protractor and a ruler to measure an angle of 46 degrees from your sight line to the top of the dead tree.

You know you are 6 feet tall, so all you need to use is the tangent function of trigonometry to solve your problem.



H	$=$	? ft
d	$=$	$d_1 + d_2$
d_2	$=$	?
h	$=$	6 ft
Θ	$=$	46 degrees
d_1	$=$	38 feet

The tangent of the angle, called theta (Θ), which you carefully measured is defined as:

$$\tan \Theta = H/d$$

But, how do you find d_2 if you don't know what d_2 is?

$$d = d_1 + d_2$$

You can compute d_2 using the following equation.

$$\tan \Theta = \frac{h}{d_2}, \text{ and } d_2 = h/\tan \Theta$$

$$\tan 46 = 1.0355303$$

$$d_2 = \frac{6}{1.0355303} = 5.7941327 \text{ feet}$$

$d = 38 + 5.79$ feet $= 43.7941327$

$H = d \tan \Theta = (43.7941327 \text{ ft}). \, 1.0355303 = 45.350152$ feet or $= 45.35$ feet

Since the tree height is less than 50 feet, you can safely cut it down without sectioning it.

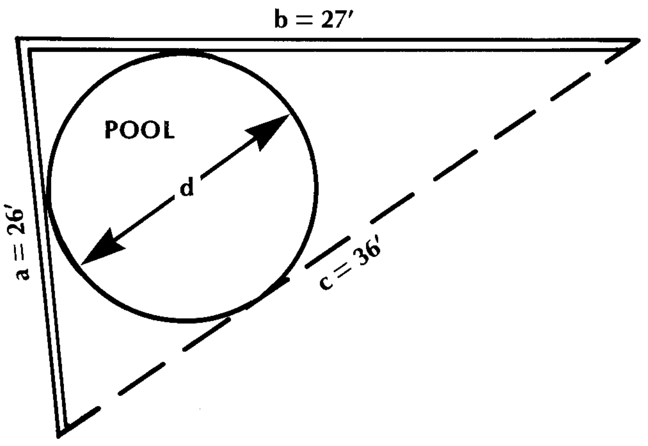
```
OFF ON A TANGENT
```

	Type	Comments
0000	ALG	
0001	CLR	
0002	CLMEM	
0003	11111	
0011	STO	
0012	5	
0020	STP	Wait for h entry
0021	STO	Store h entry in location 4
0022	4	
0030	PUSH	
0031	22222	
0039	STO	
0040	7	
0048	STP	Wait for angle Θ entry
0049	DEG	
0050	TAN	Compute tangent of <Θ
0051	STO	Store tangent Θ in location 6
0052	6	
0060	/	Divide h by tangent Θ
0061	POP	
0062	=	Compute d_2
0063	STO	
0064	8	
0072	PUSH	
0073	33333	
0081	STO	
0082	3	
0090	STP	Wait for d_1 entry
0091	STO	
0092	2	
0100	+	
0101	POP	
0102	=	
0103	*	d computed
0104	RCL	
0105	6	
0113	=	H computed
0114	STP	
0115	GOTO	
0116	0	
0124	STP	

EXAMPLE 4. PUTTING A ROUND "PEG" IN A THREE-CORNERED HOLE

This program illustrates the Reverse Polish Notation Mode, and the square root function.

Suppose you want to place an above-ground swimming pool (or hot tub) in a corner of your yard, and you don't know and can't readily measure the angle at which the fences meet at the corner. You want to know how big a circular pool will fit into that corner without sticking out past a chosen line (c).



Pool must not protrude past line "c" into yard.

What is the largest possible diameter (d) of a circular pool that will fit this space? Find the area (A) using the following formula.

$$A = \frac{a+b+c}{2}$$

To find the diameter, use the following.

$$d = 2r$$

Since you don't have a number for d or r, you can use the following equation to find d.

$$d = \left\{2\,\frac{\sqrt{A(A-a)\,(A-b)\,(A-c)}}{A}\right\}$$

d = 15.727668 feet, or = 15 feet, 8¾ inches.

The following program stops three times; once each for a, b, and c, to be entered. It will then compute d and display it in memory location 0.

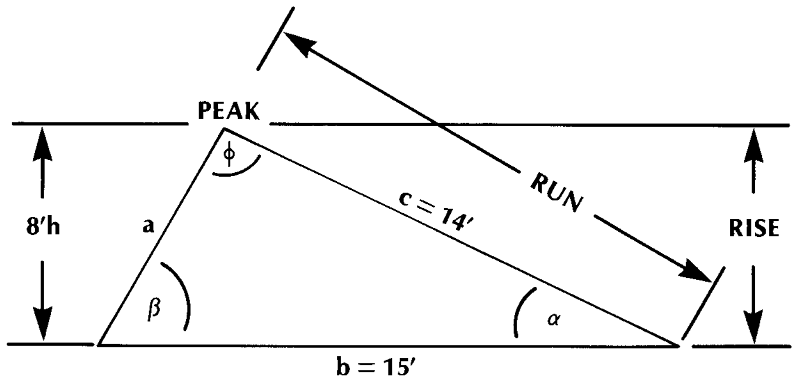
PUTTING A ROUND PEG IN A THREE-CORNERED HOLE

	Type	**Comments**
0000	RPN	
0001	CLR	
0002	CLMEM	
0003	STP	Wait for "a" entry
0004	STO	Save "a" in memory 1
0005	1	
0013	STP	Wait for "b" entry
0014	STO	Save "b" in memory 2
0015	2	
0023	STP	Wait for "c" entry
0024	STO	Save "c" in memory 3
0025	3	
0033	+	Add b to c
0034	+	Add a to sum in X register
0035	2	
0043	/	Divide by 2 to calculate A
0044	PUSH	Push A into stack for future use
0045	PUSH	
0046	RCL	
0047	1	Calculate A-a term
0055	-	
0056	XCHGY	
0057	PUSH	
0058	RCL	
0059	2	Calculate A-b term
0067	-	
0068	XCHGY	
0069	PUSH	
0070	RCL	
0071	3	Calculate A-c term
0079	-	
0080	*	Perform multiplication
0081	*	of A(A-a) (A-b) (A-c)
0082	*	
0083	SQRT	Take square root
0084	XCHGY	
0085	/	Divide by A
0086	2	Multiply by 2 to change radius to
0094	*	diameter
0095	STO	Display result in memory location 0.
0096	0	
0104	STP	

EXAMPLE 5. PUTTING A ROOF OVER YOUR HEAD

This program illustrates RPN and DEG modes and the cosine, arc cosine, power, and square functions.

Suppose you want to build a doll house (or a real one, for that matter). You've read somewhere that to determine the roof pitch, the amount of rise should be about equal to or greater than one-half the run of the rafter (see diagram). Your problem is complicated because on the reverse slope the pitch will have to be steeper. The problem is that you don't know exactly how steep it will have to be.



You know you want the peak of the roof to be 8 feet (or inches, on a smaller scale) and that the longest run for your rafters will be 14 feet. Knowing this, you can figure out what angle alpha (α) is using the following:

$\angle\alpha$ = arcsine h/c = 34.849905 degrees

Now, since 15 feet is the longest board that you have, you can use 15 for length b also. Now you need to know the following:

How long is a? What's the peak angle phi (ϕ), and what's the pitch of the reverse slope, angle beta (β).

There is a way to find all of this with only the information you have. The law of cosines gives us the following three equations:

$$a^2 = b^2 + c^2 - 2bc \cos \alpha$$

$$c^2 = a^2 + b^2 - 2ab \cos \beta$$

$$b^2 = a^2 + c^2 - 2ac \cos \phi$$

Using these equations, you can find the length of side a, and angles ϕ and β:

$$a = \sqrt{b^2+c^2-2bc \cos \alpha} = 8.7364889$$

$$\angle\phi = \arccos\left(\frac{a^2+c^2-b^2}{2ac}\right) = 78.844813 \text{ degrees}$$

$$\angle\beta = \arccos\left(\frac{a^2+b^2-c^2}{2ab}\right) = 66.305284$$

The following program will calculate any side if the other two sides and the angle at which they meet are given, or it will calculate the angle when the two sides adjacent to the angle and the opposing side are given.

When you run the program it will come to a stop with 11111 in the X register. When this happens and you want to calculate an unknown side, enter in degrees the angle that is opposite the unknown side. At stops 22222 and 33333, enter the two sides that form this angle.

If you want to calculate an angle, enter 99999 and type **CON** SPACE BAR or SHIFT @ to continue. The program will go into the second part of the program and stop again with 11111 in the X register. At this stop, enter the values for one of the sides which form the angle to be calculated. At the second stop, enter the value of the second side which forms the angle.

PUTTING A ROOF OVER YOUR HEAD

	Type	Comments
0000	CLMEM	
0001	RPN	
0002	CLR	
0003	DEG	
0004	99999	
0012	STO	
0013	99	
0021	CLX	
0022	11111	
0030	STP	Enter angle or '99999'
0031	XEQ	
0032	99	If entry is 99999, jump to line 201.
0040	201	
0048	XCHGY	
0049	POP	
0050	STO	
0051	0	
0059	COS	Calculate cosine of angle
0060	22222	
0068	STP	Enter side
0069	XCHGY	
0070	POP	
0071	STO	
0072	1	
0080	PUSH	
0081	33333	
0089	STP	Enter adjacent side
0090	XCHGY	
0091	POP	
0092	STO	
0093	2	
0101	XCHGY	
0102	RCL	Recall adjacent side from location 2
0103	2	
0111	*	Multiply the two sides together
0112	2	

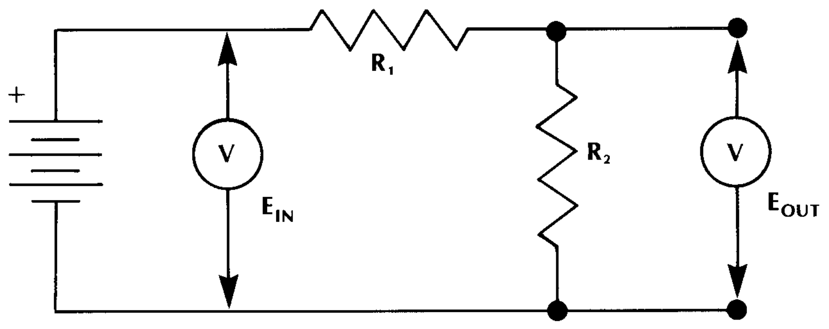
	Type	Comments
0120	*	
0121	STO	
0122	98	
0130	POP	
0131	2	Square one side
0139	POWER	
0140	XCHGY	
0141	2	Square second side
0149	POWER	
0150	+	Add squared sides
0151	XCHM	
0152	98	
0160	*	
0161	RCL	
0162	98	
0170	XCHGY	
0171	-	
0172	SQRT	Computes side a
0173	STO	Display result in memory 3
0174	3	
0182	RCL	
0183	99	
0191	GOTO	
0192	325	
0200	STP	Enter side of unknown angle
0201	POP	
0202	STP	
0203	XCHGY	
0204	POP	
0205	STO	
0206	1	
0214	22222	
0222	STP	Enter other side of unknown name
0223	XCHGY	
0224	POP	
0225	STO	
0226	2	
0234	2	
0242	*	
0243	*	
0244	33333	
0252	STP	Enter side opposite unknown angle
0253	XCHGY	
0254	POP	
0255	STO	Display result in memory 3
0256	3	
0264	RCL	
0265	2	
0273	RCL	
0274	1	
0282	2	
0290	POWER	
0291	XCHGY	
0292	2	

	Type	Comments
0300	POWER	
0301	+	
0302	XCHGY	
0303	2	
0311	POWER	
0312	-	
0313	XCHGY	
0314	/	
0315	ACOS	
0316	STO	
0317	0	
0325	STP	
0326	GOTO	
0327	2	
0335	STP	

EXAMPLE 6. DIVIDE AND CONQUER

This example program uses Algebraic Notation With No Operation Precedence Mode and the Tangent function.

In electricity or electronics the use of resistance ratios to derive a particular voltage at a particular point is fundamental. A simple resistive network consists of two resistors connected in such a way that one is in series with, and one is in parallel with the electrical load. More complex networks can usually be reduced mathematically to this same form:



This fundamental network is referred to as a *voltage divider* or just *divider.* R_1 is the total resistance in series with the voltage source and the load, and R_2 is the total resistance in parallel with the source and the external load. If the net resistance of the external load is high, relative to the resistance of R_2, then the voltage at point 1 becomes for the most part, a function of the ratio of R_1 to R_2. Expressed as an equation, it would be:

$$E_{out} = \frac{E_{in}R_2}{R_1 + R_2}$$

The following program calculates E_{out} for any combination of R_1, R_2 and E_{in}.

```
DIVIDE AND CONQUER
```

	Type	Comments
0000	ALGN	
0001	CLR	
0002	CLMEM	
0003	11111	
0011	STP	Wait for R_1 entry
0012	STO	Display R_1
0013	1	
0021	22222	
0029	STP	Wait for R_2
0030	STO	Display R_2
0031	2	
0039	33333	
0047	STP	Wait for E_{in} Entry
0048	STO	Display E_{in}
0049	3	
0057	*	
0058	RCL	
0059	2	
0067	=	Calculate $R_2 * E_{in}$
0068	/	
0069	(	
0070	RCL	Divide by
0071	1	the quantity
0079	+	$R_1 + R_2$
0080	RCL	
0081	2	
0089	)	$(R_1 + R_2)$
0090	=	Resultant E_{out}
0091	STO	Display E_{out}
0092	0	
0100	STP	
0101	GOTO	Loop
0102	0	program
0110	STP	

EXAMPLE 7. PERMUTATIONS!

This example illustrates the factorial function and is programmed in RPN mode.

When any group of objects is considered, there are often two areas of interest:

1. How many possible sub-groupings can be made of this number of things if the order of the objects in each sub-group is not considered (PERMUTATIONS)?

2. How many sub-groups are available if ordering within each sub-group is considered and not allowed to repeat (COMBINATIONS)? (See Example 8.)

The equation for Permutations is:

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

P = Number of Permutations
n = Number of total items
r = Number of items in each sub-group.

For example, if you have five letters (a, b, c, d, e), how many three-letter groups can you make of these five, using groups that can contain the same letters, but in a different arrangement?

$$P = \frac{n!}{(n-r)!} = \frac{5!}{(5-3)!} = \frac{120}{2} = 60$$

The following program illustrates this problem, allowing you to choose n and r. When the program stops with 11111 in the X register, enter n. When it stops again with 22222 in the X register, enter r.

When the program completes execution, memory location 0 will contain the result.

PERMUTATIONS!

	Type	Comments
0000	RPN	
0001	CLMEM	
0002	CLR	
0003	11111	
0011	STP	Enter n
0012	PUSH	Push entry into stack
0013	22222	
0021	STP	Enter r
0022	XCHGY	Get rid of
0023	POP	22222
0024	-	n-r
0025	FACT	Calculate denomination
0026	XCHGY	Swap X and Y register contents
0027	FACT	Calculate numerator
0028	XCHGY	Swap back X and Y
0029	/	Perform divide
0030	STO	Display
0031	0	result in Memory Ø
0039	STP	On CONT
0040	GOTO	Loop
0041	2	Program
0049	STP	

EXAMPLE 8. COMBINATIONS!

This problem also uses the factorial function and is programmed in RPN.

This example is based on a similar premise. You want to know how many sub-groups, each containing a specified number of items, you can make from a large group. This time, however, you cannot use the same number twice. The equation to find the number of possible combinations is:

$$C = \frac{n!}{r!\,(n\text{-}r)!}$$

where C is the number of combinations,
r is the number of items in each sub-group, and
n is the total number of items.

Using the same data from the last problem (a, b, c, d, e), how many non-repeated three-letter combinations can you make from these five letters?

$$C = \frac{5!}{3!\,(5\text{-}3)!} = \frac{120}{6\,(2)} = 10$$

You can substitute your own values for r and n in the following program, or you can use the above values 3 and 5.

When 11111 appears in the X register, enter 5 (or the number you want) and when 22222 appears in the X register, enter 3 (or the number you want).

```
COMBINATIONS!

               Type                 Comments

0000            RPN
0001          CLMEM
0002            CLR
0003          11111
0011            STP ————————————— Wait for n
0012           PUSH
0013          22222
0021            STP ————————————— Wait for r
0022          XCHGY
0023            POP
0024           PUSH
0025           FACT ————————————— r!
0026            STO
0027             98
0035            POP              ┌ n-r
0036              - ——————————————┤
0037           FACT              └ (n-r)!
0038            RCL
```

```
                Type                 Comments

0039             98
0047              * ──────────────── r! (n-r)!
0048          XCHGY ──────────────── Get n
0049           FACT ──────────────── n!
0050          XCHGY ┐                  n!
                    ├─────────────── ─────────
0051              / ┘                r! (n-r)
0052            STO
0053              0 ──────────────── Display result in memory
0061            STP ──────────────── Stop execution;
0062           GOTO ────────────┬─── On continue, loop program
0063              2             └─── (Do not clear Memory)
0071            STP
```

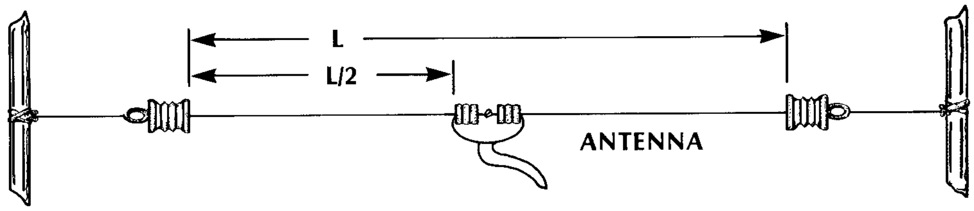
PROGRAMMING EXAMPLES FOR RADIO NUTS

EXAMPLE 9. I CAN'T HEAR YOU—GET A BIGGER ANTENNA

This sample program demonstrates a practical use for the following CALCULATOR functions:

Clear Memory (**CLM**)
PUSH
POP
GOTO

In ham radio (or CB radio), it may be necessary to calculate the length of a half-wave dipole antenna if you are going to install one. The formula below is for a center-fed, harmonically-operated doublet with a 5% correction for "end-effect" for frequencies under 30 megahertz.



$$L = \frac{(K - .05K)*492}{f}$$

where L is in feet
f is in megahertz, and
K is the number of half-waves on the antenna.

The following program calculates L for any number of half-waves (K) at any frequency (f) for this type of antenna.

```
I CAN'T HEAR YOU. GET A BIGGER ANTENNA
```

	Type	Comments
0000	ALG	Set ALG mode
0001	CLR	Clear stack
0002	CLMEM	Clear memory
0003	11111	Set input K indicator
0011	STP	Wait for K input
0012	PUSH	Push entry into stack
0013	*	Multiply by 0.05
0014	0.05	
0022	=	End-effect and total
0023	-	Subtract end-effect
0024	POP	POP to get entry out of stack
0025	=	Total
0026	*	Multiply by 492
0027	492	
0035	=	Total
0036	PUSH	Push-Save partial result
0037	22222	Set input f input
0045	STP	Wait for f input
0046	/	Divide by f
0047	POP	POP stack to divide correctly
0048	=	Total
0049	STO	Save L result in memory
0050	0	location 0
0058	33333	Set run end indicator
0066	STP	
0067	GOTO	End of run; If CONT typed,
0068	1	loop program.
0076	STP	

EXAMPLE 10. FASTER THAN A SPEEDING BULLET

This example program illustrates a practical use for the Mean of X CALCULATOR function.

Many times in radio, you need to find the transmission frequency when the band or wavelength is known or, conversely, to find the band or wavelength when the frequency is known. Radio waves travel at the speed of light. The equation is a simple one, but for some reason many people have trouble remembering the form and units so it is presented here:

$$f = \frac{c}{\lambda}$$

where f is the frequency in kilohertz, and λ is the wavelength in meters.

$$\lambda = \frac{c}{f}$$

FASTER THAN A SPEEDING BULLET

	Type	Comments
0000	ALG	
0001	CLR	
0002	CLMEM	
0003	STP	Enter value for f or λ
0004	PUSH	
0005	CLX	Clear X register
0006	STP	Enter a 1 if λ value was entered; 2 if f value was entered
0007	XCHM	
0008	98	
0016	1	Program checks to see if you entered value for λ
0024	XEQ	
0025	98	
0033	86	If true, then jumps to line 0086.
0041	2	Program checks to see if you entered value for f.
0049	XEQ	
0050	98	
0058	135	If true, then jumps to line 0135.
0066	33333	If number other than 1 or 2 entered at line 0006, program displays 33333 in X register and stops.
0074	STP	
0075	GOTO	Type SHIFT @ to restart program.
0076	0	
0084	STP	
0085	STP	
0086	11111	
0094	STO	
0095	1	
0103	300000	Light speed in kilometers per second.
0111	/	
0112	POP	
0113	=	
0114	STO	Result stored in location 0.
0115	0	
0123	STP	
0124	GOTO	
0125	1	
0133	STP	
0134	STP	
0135	22222	
0143	GOTO	
0144	94	
0152	STP	

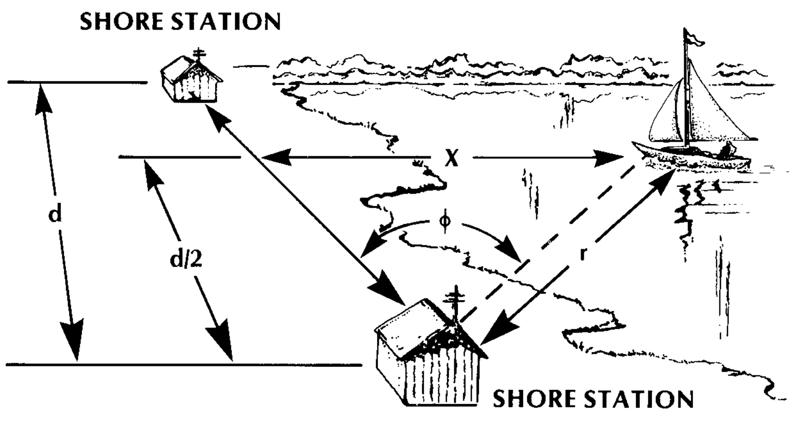
If you are on a boat with a radio and its maximum range is 300 miles, how far from the shore can the boat travel and still maintain contact if there are shore radio stations along the coast every 225 miles?

EXAMPLE 11. SOS...I THINK I'M LOST

This program illustrates a practical use for the following trigonometric functions available in the CALCULATOR:

- Arc Cosine
- Sine

You can solve this problem easily using trigonometry.



r = range of boat radio
d = distance between shore stations
x = unknown maximum distance from shore

Observe that the distance to be computed (x) forms one side of a right triangle. The other two sides of the triangle are r, the boat's radio range, and d/2, or half the distance between shore stations.

If we can find angle theta (Θ), we can use the trigonometric relationship:

$$\sin \Theta = \frac{x}{r} \text{ or } x = r(\sin \Theta)$$

to find the unknown distance (x). However, we don't know what angle Θ is—yet.

We can find angle Θ if we remember that we do know the values of d, d/2, and r. The trigonometric cosine function will give us angle Θ. Using the ratio,

$$\cos \Theta = \frac{d/2}{r}\text{, and}$$

$\cos^{-1}\Theta$ (or arc cosine Θ)

will give the required angle Θ, which you can then use to find the sine of Θ. You then use sin Θ to calculate the unknown distance, d.

S.O.S. ... I THINK I'M LOST.

	Type	Comments
0000	ALG	
0001	CLR	
0002	CLMEM	
0003	225	Distance between shore stations, d.
0011	/	
0012	2	Calculate one-half of d.
0020	=	
0021	/	Calculate the cosine value.
0022	300	
0030	=	

	Type	Comments
0031	DEG	Set degrees
0032	ACOS	Find the arc cosine.
0033	PAUSE	Pause to display the angle.
0034	SIN	Calculate the sine.
0035	*	Multiply by the boat's radio range, r.
0036	300	
0044	=	This is the maximum offshore distance, x.
0045	STP	

EXAMPLE 12. DECIBELS, SHMECIBELS. TURN IT DOWN!

This program illustrates the use of the logarithmic base 10 function to calculate the gain in an amplifier.

Gain in an amplifier is often measured in decibels to allow plotting a gain/frequency response curve on logarithmic graph paper. The equation for this calculation is:

$$G_{(db)} = 20\left(\log\left(\frac{E_O}{E_i}\right)\right)$$ where E_O is the output voltage and E_i is the input voltage

Note: E_O and E_i must be measured across the same impedance.

For example, if you measure 20 volts at the output of the final stage of an amplifier and you measure 10 millivolts of the input to its first stage, then the voltage gain is:

$$A = E_O = \frac{20}{10 \times 10^{-3}} = 2 \times 10^{3} = 2000$$

This gain expressed in decibels is:

$$G_{db} = 20\left(\log\left(\frac{E_O}{E_i}\right)\right) = 20\,(\log[2000]) = 66.0206$$

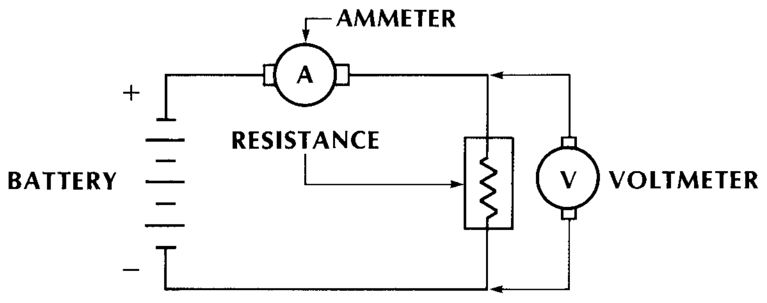
The following example illustrates how to program this problem:

DECIBELS, SHMECIBELS. TURN IT DOWN!

	Type	Comments
0000	ALG	
0001	CLMEM	
0002	CLR	
0003	20	E_O
0011	/	Ratio
0012	0.01	E_2
0020	=	Voltage gain
0021	LOGTEN	
0022	*	Convert to decibels
0023	20	
0031	=	Result complete
0032	STP	

EXAMPLE 13. THE EVER-POPULAR OHM'S LAW

In electronics, there are several formulas that are used often. The most general is Ohm's Law which describes the relationships of voltage, current, and resistance in a circuit. It is usually stated:



E = IR where E is electromotive force (or voltage in volts,) I is the current in amperes and R is the resistance of the circuit where the voltage is observed.

Knowing any two variables allows you to determine the third by simply rearranging the form of the equation.

E = IR
I = E/R
R = E/I

This program will calculate any one of the variables, if the remaining two are supplied.

E is entered followed by an identifier of 1.
I is entered followed by an identifier of 2.
R is entered followed by an identifier of 3.

The answer will be stored in memory location 0, with its identifier number (1, 2, or 3) stored in location 1.

A 33333 value appearing in memory location 7 means that the same identifier was entered twice.

```
THE EVER-POPULAR OHM'S LAW

                Type            Comments

0000             RPN
0001             CLR
0002           CLMEM          ┌Type value for E, I, or R and RETURN, then
0003             STP ─────────└type 1 for E, 2 for I, or 3 for R and SHIFT @.
0004             STO ──────────Save and display the
0005               3           identifier number
0013           XCHGY ──────────Swap X and Y registers
0014             STO ┐        ┌Save and display the
0015               2 ┘────────└variable value entered
0023             POP ──────────POP stack
0024             STP ──────────Wait for second entry.
```

```
          Type        Comments
0025       0
0033     STO  ]------ Save and display the identifier number.
0034       7  ]
0042     POP        [ Check for mistake same
0043     XEQ -------[ ID entered twice.
0044       3 -------- If yes, go to Error Stop.
0052     236
0060     STO
0061       5
0069   XCHGY
0070     STO
0071       4
0079     POP
0080       +
0081    PUSH
0082       5 -------- Is this calculator E?
0090       -          (I=2, R=3)-5=0
0091     XEQ -------- Check for zero result.
0092      99        [ If yes, go do multiply,
0100     312 -------[ else do divide
0108       1        [ Is voltage variable in
0116     XEQ -------[ memory location 2?
0117       3        [ If yes go get it.
0125     284 -------[ Else it is in memory
0133     RCL        [ location 4.
0134       4
0142     RCL
0143       2
0151       /
0152     STO
0153       0
0161     POP
0162     POP
0163     POP
0164       3
0172       -
0173     XEQ
0174      99
0182     218
0190       2
0198     STO
0199       1
0207     STP
0208    GOTO
0209       0
0217     STP
0218       3
0226    GOTO
0227     198
0235     STP
0236     STO
0237       5
0245   XCHGY
0246     STO
```

	Type	Comments
0247	4	
0255	33333	
0263	STO	
0264	7	
0272	POP	
0273	POP	
0274	GOTO	
0275	24	
0283	STP	
0284	RCL	
0285	2	
0293	RCL	
0294	4	
0302	GOTO	
0303	151	
0311	STP	
0312	RCL	
0313	2	
0321	RCL	
0322	4	
0330	*	
0331	STO	
0332	0	
0340	1	
0348	GOTO	
0349	198	
0357	STP	

EXAMPLE 14. CHARGE IT

This program illustrates how to use the function of Pi and Reciprocal.

The reactance of a capacitor acts like a resistance that changes. This change in reactance is inverse to the frequency; i.e., as the frequency goes higher, the capacitive reactance gets smaller unlike inductive reactance. The formula for calculation of capacitive reactance (X_C) is:

$X_C = \frac{1}{2\pi fc}$, where f is the frequency in hertz and c is the capacitance in farads.

For example, X_C for a capacitive of 1 microfarad at a frequency of 7.14 megahertz is:

$X_C = 1/(6.28)(1 \times 10^{-5})(7.14 \times 10^{6}) = .002290608$ ohms

CHARGE IT

	Type	Comments
0000	RPN	
0001	CLR	
0002	CLMEM	
0003	11111	
0011	STP	Wait for frequency input
0012	XCHGY	
0013	11111	

	Type	Comments
0021	+	
0022	STP	Wait for capacitance input
0023	XCHGY	
0024	POP	
0025	2	
0033	PI	
0034	*	
0035	*	
0036	*	
0037	RECIP	Generate reciprocal
0038	STO	Display result, X_C, in memory 0.
0039	0	
0047	33333	
0055	STP	Indicate completion. To loop program, press SHIFT @.
0056	GOTO	
0057	0	
0065	STP	

EXAMPLE 15. PERSONAL MAGNETISM

Coils of wire or inductors as they are called, exhibit a form of resistance which changes its value when the frequency of the voltage and current applied to them changes. This property of coils is called reactance and the equation to calculate its value is:

Inductive reactance, $X_L = 2\pi fL$, where f is frequency in hertz and L is the inductance in henries

The following program will calculate X_L for any inductance at any frequency.

For example, the reactance of a coil having an inductance of 7 millihenries at a frequency of 300 kilohertz is:

$X_L = 2\pi\,(7\times10^{-3})\,(300\times10^{3}) = 6.28\,(2100) = 13194.689$ ohms or
13.194689 Kohms.

PERSONAL MAGNETISM

	Type	Comments
0000	RPN	
0001	CLR	
0002	CLMEM	
0003	11111	
0011	STP	Wait for frequency entry
0012	XCHGY	
0013	11111	
0021	+	
0022	STP	Wait for inductance input
0023	XCHGY	
0024	POP	
0025	2	
0033	PI	

```
          Type        Comments

0034         *
0035         *
0036         *
0037       STO ┐────────Display result, XL in memory
0038         0 ┘
0046     33333 ──────── Indicate end of run
0054       STP ──────── To loop program, type CONT SPACE BAR
0055      GOTO
0056         0
0064       STP
```

EXAMPLE 16. WHY MAKE IT DIFFICULT?

This example, using RPN mode, illustrates the square and square root functions.

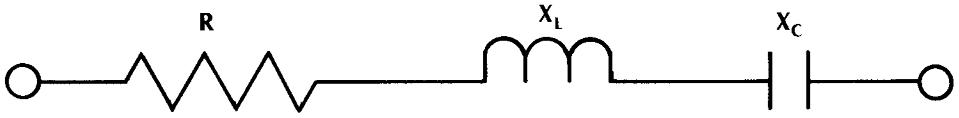
You can solve for the total impedance of a series circuit using the equation:

$Z = \sqrt{R^2 + X^2}$

Notice that in this case, it does not matter whether the reactance is capacitive or inductive, because a negative quantity squared becomes positive.

The following program solves an impedance problem as an example.

The circuit impedance calculated is for the circuit below:



$R = 75$ ohms
$X_L = 150$ ohms
$X_C = 50$ ohms

It is assumed in this case that the net series reactance of X_L and X_C has already been calculated to be 100 ohms (inductive).

$z = 125$ ohms

```
WHY MAKE IT DIFFICULT?

          Type        Comments

0000       RPN ──────── Set RPN
0001     CLMEM ──────── Clear Memory
0002       CLR ──────── Clear Stack
0003        75 ──────── Resistance
0011    SQUARE
0012       150 ┐
0020        50 ├─────── Calculate net
0028         - ┘        reactance
```

```
          Type                   Comments

0029    SQUARE ——————————— Square net reactance
             + ——————————— Add R²
          SQRT ——————————— Take square root of sum
           STP ——————————— Complete, result in X register
```

EXAMPLE 17. SAY IT WITH VECTORS

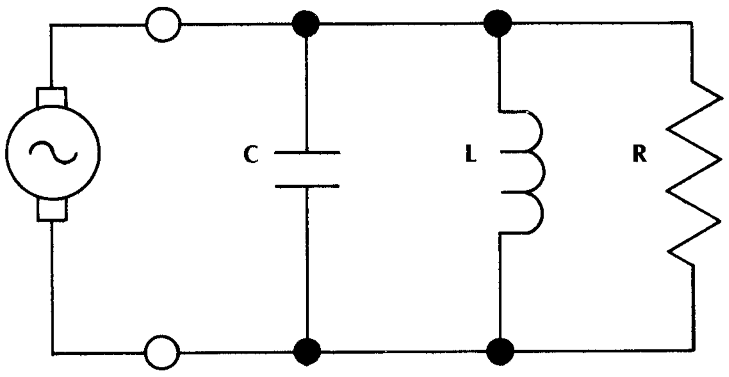
This example also illustrates the square and square root functions and is solved using the RPN mode.

The case of impedance calculation, when resistance and reactance are connected in parallel, is given by the equation:

$$Z = \frac{RX}{\sqrt{R^2 + X^2}}$$

In this case, the net impedance must be considered as inductive or capacitive to determine the sign of the X term in the numerator, which is not squared.

The following program calculates the total impedance for a parallel circuit shown as:



$$X_C = 50$$
$$X_L = 150$$
$$R = 75$$

Remember that the formula for X_C and X_L in parallel is

$$X = \frac{X_L X_C}{X_L - X_C}$$

So this must be substituted for X in the equation for Z. The resultant equation is:

$$Z = \frac{R\left(\frac{X_L X_C}{X_L - X_C}\right)}{\sqrt{R^2 + \left(\frac{X_L X_C}{X_L - X_C}\right)^2}}$$

This is the equation used by the program.

SAY IT WITH VECTORS

	Type	Comments
0000	RPN	
0001	CLMEM	
0002	CLR	
0003	11111	Set enter X_L indicator
0011	STP	Wait for entry
0012	STO	Store X_L entry in location 1
0013	1	
0021	XCHGY	
0022	POP	Get rid of 11111
0023	PUSH	
0024	SQUARE	Square X_L entry
0025	22222	Set enter X_C indicator
0033	STP	Wait for entry
0034	STO	Store X_C entry in location 2
0035	2	
0043	XCHGY	
0044	POP	Get rid of 22222
0045	PUSH	
0046	SQUARE	Square X_C entry
0047	XCHGY	
0048	RCL	Get memory location 1 content.
0049	1	
0057	XCHGY	
0058	-	Subtract $X_L - X_C$
0059	SQUARE	
0060	33333	Set enter R indicator
0068	STP	Wait for entry
0069	STO	Store R entry in location 3
0070	3	
0078	XCHGY	
0079	POP	Get rid of 33333
0080	SQUARE	Square R entry
0081	*	Multiply R^2 times $(X_L - X_C)^2$
0082	STO	Store $R^2 (X_L - X_C)^2$ into location 4
0083	4	
0091	POP	

	Type	Comments
0092	*	Multiplies to get $X_L{}^2X_C{}^2$
0093	SUM	Adds this product to contents of location 4
0094	4	
0102	POP	
0103	RCL	
0104	2	
0112	RCL	
0113	3	
0121	*	
0122	*	$R(X_LX_C)$
0123	RCL	
0124	4	
0132	SQRT	$\sqrt{X_L{}^2X_C{}^2 + R^2(X_L-X_C)^2}$
0133	/	
0134	STO	
0135	0	Z, the total impedance
0143	STP	
0144	GOTO	
0145	0	
0153	STP	

EXAMPLE 18. SIDE BY SIDE

You can solve for the net reactance of series and parallel connections of reactive elements, inductors, and capacitors by using the equations:

Series: $X = X_L - X_C$ Where X_L = inductive reactance
X_C = capacitive reactance

Parallel: $$\frac{X_LX_C}{X_L-X_C}$$

The following program will calculate the parallel circuit case since it is the more complex.

When executive stops with 11111 in the X register, enter X_C. When the program stops again with 22222 in the X register, enter X_L. The third stop is completion of the calculation and the result is displayed in both the X register and in memory location 0.

Both X_L and X_C entries must be in compatible units; i.e., ohms (or both in Kohms, or both in megohms, etc.)

If X_L = 150 ohms and X_C = 50 ohms, the series case is simply 150−50=100 ohms.

What is the parallel solution?

$$X = \frac{X_LX_C}{X_L-X_C} = \frac{(150)(50)}{150-50} = 75 \text{ ohms}$$

SIDE BY SIDE

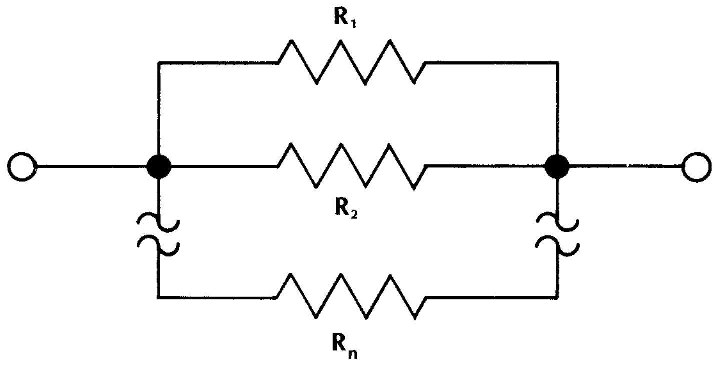
```
                 Type              Comments

0000              ALG ┐
0001            CLMEM ├──────────── Set-up
0002              CLR ┘
0003              150 ──────────── XL
0011                *
0012               50 ──────────── XC
0020                =
0021           CHGSGN
0022                /
0023                (
0024              150 ──────────── XL
0032                -
0033               50 ──────────── XC
0041                )
0042                = ──────────── Net reactance
0043              STP
```

EXAMPLE 19. WIRING YOUR PAD

In electrical or electronics work, calculation of total resistance, R_T, can be a pain if there are more than two resistors to consider. R_T is defined as the reciprocal of the sums of the reciprocals of the parallel resistances. If you didn't quite catch that, look carefully at the following equation.

$$R_T = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \quad \ldots \quad + \frac{1}{R_n}}$$



The program that follows calculates the total resistance for any number of parallel resistors and displays R_T (total resistance) in memory location 0.

As you enter resistances, memory location 1 keeps track of the number. Since the stack is being used, duplicate entries require only **PUSH** (SHIFT [) and a **CONT** (SHIFT @).

You can get out of the loop by entering a 0.

The counter value in memory location 1 is then used to keep track of how many input resistor reciprocals must be calculated and summed.

A sample run might be to calculate the total resistance of a parallel network of three resistors.

$R_1 = 5000$ ohms
$R_2 = 25000$ ohms
$R_3 = 100000$ ohms

$$R_T = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}} \quad = 4000 \text{ ohms}$$

Notice that R_T is less than the smallest; this is always true of parallel-connected circuit elements (except for capacitors.) This same program can be used for inductors in parallel and capacitors in series.

WIRING YOUR PAD

	Type	Comments
0000	RPN	
0001	CLR	
0002	CLMEM	
0003	STP	Wait for resistance input
0004	XEQ	If input is 0
0005	99	
0013	59	Go to calculate R_T
0021	RCL	
0022	1	Count inputs
0030	1	into memory
0038	+	location 1
0039	STO	
0040	1	
0048	POP	POP out counter value
0049	GOTO	Loop, accepting
0050	3	inputs
0058	STP	
0059	POP	POP out input ending 0.
0060	RECIP	Take reciprocal of a resistance
0061	SUM	and add it to memory
0062	0	location 0.
0070	RCL	
0071	1	Reduce resistor input
0079	1	counter by 1
0087	-	
0088	XEQ	All resistances input
0089	99	summed?
0097	125	If yes, go take reciprocals
0105	STO	of the sum for R_T
0106	1	else, save new counter
0114	POP	value and
0115	GOTO	loop till done.
0116	59	

	Type	Comments
0124	STP	
0125	RCL	Get summed reciprocals of resistance and
0126	0	
0134	RECIP	calculate reciprocal of sum for R_T.
0135	STO	Display R_T input memory location 0.
0136	0	
0144	STP	
0145	STP	

EXAMPLE 20. HEAT WAVE

A transformer with a resistance of 1.3 ohms when it is cold experiences a rise in temperature after some period of operation. The warm temperature of a transformer is proportional to the change in its resistance. The equation is:

$$\frac{R_h}{R_c} = \frac{234.5 + T_h}{234.5 + T_c}$$ Where R_h, R_c are the hot and cold resistances, and T_h, T_c are the hot and cold temperatures.

Rearranging the equation to isolate T_h gives:

$$T_h = \frac{R_h(234.5 + T_c)}{R_c} - 234.5 = 40.96923 \text{ degrees}$$

The program that follows calculates T_h for a transformer, which has been in operation in a room with an ambient temperature of 25°C long enough for its resistance to rise from 1.3 ohms to 1.38 ohms.

HEAT WAVE

	Type
0000	ALGN
0001	CLMEM
0002	CLR
0003	234.5
0011	PUSH
0012	+
0013	25
0021	*
0022	1.38
0030	/
0031	1.3
0039	-
0040	POP
0041	XCHGY
0042	=
0043	STP

EXAMPLE 21. THE AIR IS GETTING THICK

This program, written in RPN, uses the Natural Logarithm Function.

The work done during an isothermal energy change in a gas is given by equation:

$W = w*R*T*LN(r)$

W	=	work in foot-pounds
R	=	gas constant in foot-pounds compatible units
T	=	temperature in degrees Fahrenheit, absolute
w	=	weight of gas in pounds
r	=	ratio of expansion, i.e., final volume divided by initial
LN	=	natural logarithm

If 3 pounds of air at 32°F and at atmospheric pressure are compressed to 4 atmospheres, what is the work done during compression?

$$r = V_2 = \frac{P_1}{P_2} = \frac{1}{4}$$

$$W = w*R*T*LN(1/4) = (3)(53.3)(460+32)(LN(1/4))$$
$$= -109060.89$$, the minus sign indicates compression.

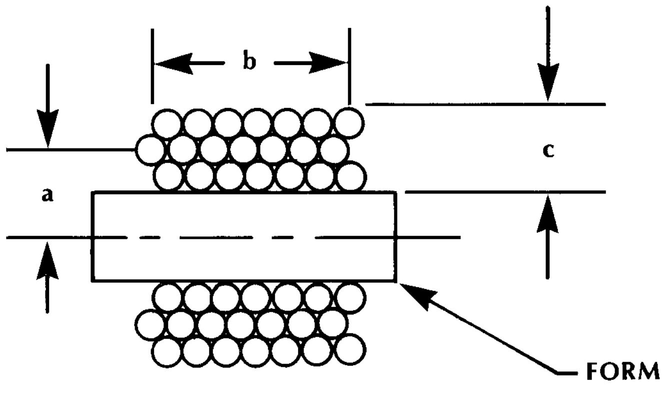
The following program calculates this result:

```
THE AIR IS GETTING THICK.
```

	Type	Comments
0000	RPN	
0001	CLMEM	
0002	CLR	
0003	3	Number of pounds of air (w)
0011	53.3	Gas constant (r)
0019	460	
0027	32	460 + 32 equals absolute temperature
0035	+	
0036	1	New volume
0044	4	Old volume
0052	/	Formation of ratio (r)
0053	LN	Natural log of ratio
0054	*	Multiply by temperature
0055	*	Multiply by R
0056	*	Multiply by w
0057	STP	Run complete

EXAMPLE 22. WINDING IT UP

Radio equipment and other electronic equipment use inductors in filters, solenoids, coupling circuits, and in other applications. The inductance of a given coil depends on several factors; the mean radius of the coil (a), the winding length (b), the winding depth (c), and the number of turns (n). The equation for the inductance of a multi-layer, air core coil is:



$$L = \frac{0.8a^2n^2}{6a+9b+10c}$$

a,b, and c in inches
L in microhenrys

For example, an air core coil with 36 turns and dimensions of a = .5 inch, b = .4 inch, and c = .5 inch with a form radius of .25 inch will have an inductance of:

$$L = \frac{(0.8)\,(.5)^2\,(36)^2}{6(.5)\ +9\,(.4)\ +10\,(.5)} \qquad = 22.244828 \text{ microhenrys}$$

The program solves this problem for an a,b,c, and n. When it stops with 11111, enter a. On each successive stop (22222, 33333, etc.), enter the next variable in order (b,c,n). Remember, these units are in inches, except for n, which is in turns.

```
WINDING IT UP.
```

```
              Type

0000          ALGN
0001         CLMEM
0002           CLR
0003         11111
0011           STP
0012           STO
0013             1
0021             *
0022             6
0030             =
0031          PUSH
0032         22222
0040           STP
0041           STO
0042             2
0050             *
0051             9
0059             =
0060          PUSH
```

```
            Type

0061        33333
0069          STP
0070          STO
0071            3
0079            *
0080           10
0088            =
0089            +
0090          POP
0091            +
0092          POP
0093            =
0094         PUSH
0095          RCL
0096            1
0104       SQUARE
0105            *
0106          0.8
0114            =
0115         PUSH
0116        44444
0124          STP
0125          STO
0126            4
0134       SQUARE
0135            *
0136          POP
0137            =
0138          STP
0139        XCHGY
0140            /
0141          POP
0142            =
0143          STO
0144            0
0152          STP
0153         GOTO
0154            0
0162          STP
```

PROGRAMMING EXAMPLES USING STATISTICS

EXAMPLE 23. HOW MANY MILES PER GALLON DOES YOUR CAR GET?

This example problem demonstrates a practical use for the following CALCULATOR functions:

- Correlation Coefficient
- Slope
- Standard Deviation of Y
- Mean of Y
- Clear Statistics Mode
- Absolute Value
- POP

In these days of energy consciousness and high gasoline prices, it might be advisable to keep track of how efficient your auto is with the fuel it uses. It is simple to do, but does require that you keep track of how many gallons of gasoline you purchase and the mileage from your odometer at each purchase. Knowing what this usual number is can alert you to potential problems too; many problems show first as a drop off in fuel use efficiency. To calculate your usage, you must fill the tank each time you buy fuel and use this number of gallons (g) as a divisor into the current mileage (M_2) minus the mileage reading at the previous fill up (M_1). The formula you need to use is shown below.

$$e = \frac{M_2 - M_1}{g}$$

Obviously, this value is not exact, since the degree to which your tank is filled varies with the cut-off point of the particular gas pump used each time, but it is an indicator in that it should not be radically wrong. When all of these values are averaged over a period of time, the average value is close to a true value.

When this program has completed execution, the following information will be displayed.

Y register = Correlation Coefficient
This is an indicator of how well the data fits the straight line or linear curve which is its trend line. The nearer this value is to 1, the better the fit and the more meaningful the result. If this value is less than .5, then you should probably plot each point on graph paper and see what the curve actually looks like.

Memory location 0 = The median or average value of all y terms.
For our specific problem, this is the average miles per gallon value.

Memory location 1 = The slope of the trend line, or linear curve.
This number represents how the Y values (miles/gallon) tend to change in respect to the X values, or number of entries, in this case. A 0 value means that there is no trend; that the average from beginning to end is the same. Any positive value indicates an increasing trend for fuel efficiency and any negative number indicates that fuel efficiency is worsening; i.e., miles per gallon is falling off. The larger this number is, the steeper the trend (the faster it's getting better or worse).

Memory location 2 = The standard deviation of the data from the trend line.
If this is a large value, it means that some data values, at least, were very far from the general trend line, and the data points should be plotted on graph paper and examined for entry errors or actual significant deviation. Either something very good or very bad happened, or you made a mistake.

HOW MANY MILES PER GALLON DOES YOUR CAR GET?

```
                 Type                Comments

0000            CLMEM ---------------Clear Memory
0001              CLR ---------------Clear Stack
0002              RPN ---------------Set RPN mode to use Stack
0003           CLSTAT ---------------Clear statistic counters and set
0004            99999                statistic mode
0012              STO ---------------Generate and save an input terminator
0013               99                recognition character in memory
0021              STP ------+        location 99
0022              XEQ ----+ +--------Type in mileage, M1.
0023               99     +----------Check to see if this is end of entries.
```

	Type	Comments
0031	80	If yes, go to ending sequence
0039	RCL	otherwise use memory location
0040	98	98 to count the number of
0048	1	periodic fuel efficiency
0056	+	Terms computed will use as statistic
0057	STO	"X" term of X, Y pairs
0058	93	
0066	POP	POP stack, get rid of counter in X
0067	-	Compute M_2-M_1
0068	STP	Type in gallons (G) of gasoline
0069	/	purchased at mileage M_2.
0070	GOTO	Divide M_2-M_1 difference by gallons:
0071	21	generate e term.
0079	STP	Loop to get next M_2,M_1
0080	POP	Ending sequence-POP stack
0081	XEQ	Check for terminator value from stack.
0082	99	
0090	136	If yes, go do final totaling and displaying.
0098	RCL	Else now use memory location
0099	98	98 as a "down" counter to keep
0107	1	"X" terms and "Y" (computed) terms
0115	-	in proper order.
0116	STO	
0117	98	
0125	SPLUS	Do statistic pair summing
0126	GOTO	Loop to get next X, Y pair.
0127	80	
0135	STP	
0136	YMEAN	Final totaling and display:
0137	STO	Display median or overall average
0138	0	value miles per gallon in memory location 0.
0146	SLOPE	Get slope of linear curve and display it in
0147	STO	memory location 1.
0148	1	
0156	YSD	Get standard deviation value and
0157	STO	display it in memory location 2
0158	2	
0166	R	Get correlation coefficient
0167	ABS	
0168	RCL	
0169	99	
0177	STP	
0178	CLR	Put 99999 in X register to indicate end
0179	RCL	Get calculated median value
0180	0	from memory location 0.
0188	CLMEM	Clear memory and reset statistics counters
0189	CLSTAT	
0190	1	
0198	STO	Set memory location 98
0199	98	with 1 for median value.
0207	POP	POP stack, get rid of 1.
0208	99999	Put terminator value in
0216	STO	X register and memory location 99.
0217	99	
0225	XCHGY	Exchange X and Y registers to get
0226	GOTO	terminator first in stack.
0227	21	Loop back to stop for first new input M_2.
0235	STP	

EXAMPLE 24. IS SPEED COSTING YOU MONEY?

This program illustrates the following statistical functions:

- Clear Statistics Mode
- Mean of Y
- Slope
- Standard Deviation of Y
- Correlation Coefficient

It also makes use of the Absolute function.

You might want to check out what happens to your auto's fuel efficiency as a function of your average speed of travel. It does require more effort to gather the raw data than just religiously making notes of travel duration for every trip (even if the car is not moving, but has the motor running). If the car is using fuel, it counts! Collect all of the data in groups associated with each time period between fill-ups. Find an average speed for each period by adding all of the trip times together and dividing this number (expressed in hours) into the mileage between this fill-up and the last. This gives you an average miles-per-hour value for each interval between refueling stops.

Use this value as the X term in the following statistics program, and use the miles-per-gallon value as the Y term, for each interval pair. The resultant Y median in memory location 0 will be your average fuel efficiency for the range of speeds used.

The slope in memory location 1 will indicate that fuel efficiency falls off at higher speeds if it's minus, and its magnitude will be an indication of how fast the efficiency changes as a function of speed. If, however the slope is plus, then you actually improve your fuel efficiency at higher speeds for your auto.

The standard deviation value in memory location 2 and the correlation coefficient are indicators of how well a straight line or linear curve actually fits the data; if the fit is less than .5 or the standard deviation is large, you may actually want to plot the data on graph paper and see what the real curve looks like. You can also determine how much scattering there is of the plotted points.

As an example, the program was run using the following as data:

Y		**X**
MPG	vs	MPH
24		65
26		60
28		55
30		50
31		45
31.5		40
31.3		35
31.2		30
31.1		25

The entries can be entered in any sequence as long as the pairs are kept associated.

The results were:

Y Median (average MPG) = 29.34

Slope $\left(\frac{\Delta y}{\Delta x}\right)$ = .1875
YSD = .9875
R = 7.00E-01

IS SPEED COSTING YOU MONEY?

	Type	Comments
0000	RPN	
0001	CLMEM	
0002	CLR	
0003	CLSTAT	
0004	99999	
0012	STO	
0013	99	
0021	11111	Set indicator for Y term entry
0029	STP	Wait for miles-per-gallon entry
0030	XEQ	
0031	99	
0039	72	
0047	XCHGY	
0048	11111	
0056	+	Set indicator for X term entry
0057	STP	Wait for speed (miles per hour) entry
0058	XCHGY	
0059	POP	POP out indicator
0060	SPLUS	Sum term pairs.
0061	CLX	
0062	GOTO	
0063	48	
0071	STP	
0072	YMEAN	Generate median value, miles per entry
0073	STO	Display in memory 0
0074	0	
0082	SLOPE	Generate slope, MPG vs. speed
0083	STO	Display in memory 1
0084	1	
0092	YSD	Generate MPG standard deviation
0093	STO	Display in memory 2
0094	2	
0102	R	Get correlation coefficient
0103	ABS	
0104	33333	Indicate run end.
0112	STP	Loop program by typing **CON** SPACE BAR
0113	GOTO	
0114	0	
0122	STP	

EXAMPLE 25. IF JOHNNY GOT AN "A," WHY CAN'T HE READ?

This problem is a variation of the grades problem presented in the Statistics Section.

Thirty-five students in a class are given an examination and graded on an absolute scale of 0 to 10. The entire class did very poorly on the examination and the instructor decided that perhaps the test was too hard. The instructor decided to modify the grades by applying a "curve" to the grades. He figured that by taking the class median grade and then making a grading scale relative to it, he could assure himself that most of the class would pass. He decided that a "B" grade would be from the mean to one standard deviation above, an "A" would be from one standard deviation above the median (mean) to two standard deviations above, and an A+ would be anything above two standard deviations above the median. Going downward, he decided that a "C" would be anything below the median to one standard deviation below it, a "D" would be anything below one standard deviation to two standard deviations below, and an "F" would be anything below that. The test scores and the original grade assignments looked like the table below:

Number of Correct Answers	Number of Students	Original Grade Scale
0	0	↑
1	4	│
2	6	│
3	12	│
4	5	│
5	7	F
6	0	D
7	1	C
8	0	B
9	0	A
10	0	A+

Where is the median?

What does the new scale look like?

The program below illustrates a solution for this problem. The new scale will look like this:

Number of Correct Answers	Number of Students	New Grade Scale
0	0	F
1	4	D
2	6	C
3	12	C
4	5	B
5	7	A
6	0	A+
7	1	│
8	0	│
9	0	│
10	0	↓

The actual results are:

		F
X mean − 2 SD_S =	.45539411	D
X mean − 1 SD_S =	1.8562685	C
X mean =	3.2571429	B
X mean + 1 SD_S =	4.6580172	
X mean + 2 SD_S =	6.0588916	A
		A+

```
IF JOHNNY GOT AN A, WHY CAN'T HE READ?

                Type           Comments

0000             ALG
0001           CLMEM
0002             CLR
0003          CLSTAT
0004               1 ————————— Score level 1
0012            PUSH
0013           SPLUS ————————— 1 student
0014            PUSH
0015           SPLUS ————————— 2 students
0016            PUSH
0017           SPLUS ————————— 3 students
0018            PUSH
0019           SPLUS ————————— 4 students
0020               2 ————————— Score level 2
0028            PUSH
0029           SPLUS ————————— 1 student
0030            PUSH
0031           SPLUS ————————— 2 students
0032            PUSH
0033           SPLUS ————————— 3 students
0034            PUSH
0035           SPLUS ————————— 4 students
0036            PUSH
0037           SPLUS ————————— 5 students
0038            PUSH
0039           SPLUS ————————— 6 students
0040               3 ————————— Score level 3
0048            PUSH
0049           SPLUS ————————— #1 student
0050            PUSH
0051           SPLUS ————————— #2 student
0052            PUSH
0053           SPLUS ————————— #3 student
0054            PUSH
0055           SPLUS ————————— #4 student
0056            PUSH
0057           SPLUS ————————— #5 student
0058            PUSH
0059           SPLUS ————————— #6 student
0060            PUSH
0061           SPLUS ————————— #7 student
0062            PUSH
0063           SPLUS ————————— #8 student
```

	Type	Comments
0064	PUSH	
0065	SPLUS	#9 student
0066	PUSH	
0067	SPLUS	#10 student
0068	PUSH	
0069	SPLUS	#11 student
0070	PUSH	
0071	SPLUS	#12 student
0072	4	Score level 4
0080	PUSH	
0081	SPLUS	#1 student
0082	PUSH	
0083	SPLUS	#2 student
0084	PUSH	
0085	SPLUS	#3 student
0086	PUSH	
0087	SPLUS	#4 student
0088	PUSH	
0089	SPLUS	#5 student
0090	5	Score level 5
0098	PUSH	
0099	SPLUS	#1 student
0100	PUSH	
0101	SPLUS	#2 student
0102	PUSH	
0103	SPLUS	#3 student
0104	PUSH	
0105	SPLUS	#4 student
0106	PUSH	
0107	SPLUS	#5 student
0108	PUSH	
0109	SPLUS	#6 student
0110	PUSH	
0111	SPLUS	#7 student
0112	7	Score level 7
0120	PUSH	
0121	SPLUS	# students
0122	XMEAN	X mean
0123	PUSH	Save it
0124	-	
0125	XSD	X mean − 1 SD
0126	=	
0127	PUSH	
0128	-	
0129	XSD	X mean − 2 SD's
0130	=	
0131	XCHGY	Put them in ascending order
0132	PUSH	
0133	XMEAN	X mean
0134	PUSH	
0135	+	
0136	XSD	X mean + 1 SD
0137	=	
0138	PUSH	

	Type	Comments
0139	+	
0140	XSD	X mean + 2 SD's
0141	=	
0142	STP	Execution complete
0143	GOTO	On continue, loop and restart program
0144	0	
0152	STP	

EXAMPLE 26. ITERATIVE—AGAIN AND AGAIN

The preceding example took quite a few steps. However, with a little forethought and some knowledge of the looping capability of the CALCULATOR Diskette Program, you can reduce the number of steps considerably. Using the same problem regarding the student's poor test results, you can reduce the number of steps involved using an iterative loop and the test function.

The program, on execution, shows a 11111 in the X register and waits for you to enter the first number of correct answers—in this case, 1. Then press SHIFT @. A 22222 then appears in the X register. When it does, enter the number of students who only got one question correct—in this case, 4. Then press SHIFT @. When 11111 again appears in the X register, enter the second score (2) and press SHIFT @. Continue entering until you have all the test scores and the number of students that made each. (If no students made a score, you don't enter it.) When you finish your entries, 11111 will again appear in the X register. Enter 99999. This notifies the program that there are no more entries. The program will display the results of its calculations in the stack in descending order.

When you have completed this program's execution, compare the results with those in the preceding problem. They should be the same.

RESULTS:	X Mean − 2 Standard Deviations	= .45539411
	X Mean − 1 Standard Deviation	= 1.8562685
		= 3.2571429
	X Mean + 1 Standard Deviation	= 4.6580172
	X Mean + 2 Standard Deviations	= 6.0588916

ITERATIVE--AGAIN AND AGAIN

	Type	Comments
0000	ALG	
0001	CLMEM	Clear memory
0002	CLSTAT	Clear and set statistics
0003	99999	Remember entry termination character
0011	STO	
0012	98	
0020	CLR	
0021	11111	Set entry identifier
0029	STP	Wait for score value
0030	XEQ	Is this 99999, the entry loop terminator?
0031	98	
0039	119	If yes, go calculate result
0047	PUSH	else, save entry in stack.

```
            Type         Comments

0048        PUSH
0049        22222 ---------- Set entry identifier
0057        STP ------------ Wait for number of students at
0058        *                this score.
0059        POP ------------ Form a calculation loop counter.
0060        =                Counter = (X*n)
0061        CHGSGN
0062        STO ------------ Save counter value
0063        1
0071        POP ------------ Get counter off stack
0072        PUSH ----------- Propagate X value into stack
0073        SPLUS ---------- Statistical summation
0074        PUSH ----------- PUSH X value into stack
0075        SUM ------------ Count down counter
0076        1
0084        RCL ------------ Get current counter value
0085        1
0093        XEQ ------------ Check if counter is zero
0094        99
0102        20 ------------ [ If yes, loop back for
0110        GOTO ]          [ next input
0111        71   ]---------- Else, loop—keep summing
0119        XMEAN ---------- X mean
0120        -
0121        XSD ------------ X mean—1 SD
0122        =
0123        PUSH
0124        -
0125        XSD ------------ X mean—2 SDs
0126        =
0127        XCHGY ---------- Swap places, XM−2 SDs on bottom
0128        PUSH ----------- PUSH into stack
0129        XMEAN ---------- X mean
0130        PUSH ----------- Save it
0131        +
0132        XSD ------------ X mean + 1 SD
0133        =
0134        PUSH
0135        +
0136        XSD ------------ X mean + 2 SDs
0137        =
0138        STP ------------ Stop, all done
0139        GOTO ----------- On CONT, rerun program
0140        0
0148        STP
```

PROGRAMMING EXAMPLES USING INTEREST

EXAMPLE 27. BEFORE YOU BUY THAT FURNITURE ON TIME...

This program illustrates the use of the CLINT function.

Whenever you buy anything on time via an installment credit plan, you should know exactly what your true interest rate is. The CALCULATOR cartridge has financial calculations built into it, so you don't have to know the equations to use them.

For those who are interested, however, the following program makes use of the equation:

$$i = \frac{2nf}{A(p+1)}$$

i = interest rate
n = number of payments per year
f = finance charge
A = Amount actually financed (price minus down payment)
p = Total number of scheduled payments.

The program example operates on a problem which supposes a $6000 price with $550 paid down and $5450 financed. The term of the contract is 5 years and the payments are $114.50 per month—every month. The true interest rate on the credit amount is 10.25 per cent (approximately).

```
BEFORE YOU BUY THAT FURNITURE ON TIME...

                 Type                 Comments

0000              ALG
0001            ENTER
0002            CLINT
0003              STP --------------- Enter full cash price
0004                -
0005              STP --------------- Enter down payment made
0006                =
0007              STO
0008                1
0016              STP --------------- Enter total number of months of contract
0017                *
0018             PUSH
0019              STP --------------- Enter monthly payment amount
0020                =
0021                -
0022              RCL
0023                1
0031                = --------------- Total finance charge
0032                *
0033              STP --------------- Enter number of payments per year
0034                =
0035                *
0036                2
0044                =
0045              STO
0046                2
0054                1
0062                +
0063              POP
0064                =
0065                *
0066              RCL
0067                1
0075                =
0076             PUSH
0077              RCL
```

```
            Type                     Comments

0078          2
0086          /
0087        POP ---------------- POP stack back out.
0088      XCHGY ---------------- Rearrange for correct divide.
0089          =
0090          *
0091        100 ---------------- Express as percent
0099          =
0100        STO ---------------- True interest percentage is
0101          0                  in memory location 0.
0109        STP
0110       GOTO ---------------- Loop back on CONT; rerun program
0111          0
0119        STP
0120        STP
```

EXAMPLE 28. WHAT WILL IT BE WORTH?

This program uses the following financial functions:

- Clear Interest Mode
- Present Value
- Number of Periods
- Interest
- Find and Enter Modes
- Future Values

Anyone who is still able to have money invested in this time of high inflation is naturally very interested in how fast it is growing. This program takes any amount, over any number of months using any specified annual percentage rate (considered to be compounding periodically) for any stated interest compounding period (in months), and calculates the future value at the end of the investment period and the earned amount.

This example uses an amount of $8000 invested for a period of 24 months, at a 7 percent annual interest rate, compounded quarterly (every 3 months).

```
WHAT WILL IT BE WORTH?

            Type                     Comments

0000        ALG
0001      ENTER ---------------- Set ENTER mode
0002        CLR
0003      CLMEM
0004      CLINT ---------------- Set financial operation
0005        STP ---------------- Enter amount invested
0006         PV
0007        STP ---------------- Enter number of months
0008          /
0009         12
```

```
              Type                Comments

0017            =
0018          STO
0019            3
0027           12
0035            /
0036          STP ─────────────── Enter compounding period (in months)
0037            =
0038          STO
0039            2
0047            *
0048          RCL
0049            3
0057            =
0058            N
0059          STP
0060            /
0061          RCL
0062            2
0070            =
0071            I
0072         FIND ─────────────── Set FIND mode
0073           FV
0074          STO
0075            0
0083            -
0084           PV
0085            =
0086          STO
0087            1
0095          STP
0096         GOTO ─────────────── Loop program
0097            0
0105          STP
```

PROGRAMMING EXAMPLE USING BIT MANIPULATIONS

EXAMPLE 29. "SOME" CHECK

The Exclusive OR function (Section 10) is often used to generate check characters in digital equipment; e.g., a tape unit or telephone interface. If you need to check the validity of information contained in a serial stream of digital bits, you can ex*clusive OR* each character with each succeeding character in the stream. (A character is nominally 8 bits.) This accumulated character is then written at the end of the information stream as a "check" character. Then, when you read the data back, you again exclusive OR each character. The final result should be all zeros. If any bit in the accumulated check character remains 'on' ($\neq 0$) after it has been exclusive ORed to the end result, then you know you've got an error.

The following example illustrates the Exclusive OR function using 5 data characters and a generated check character. The data characters will be displayed in memory locations 1 through 5 and the generated check character will be in memory location 0. The program halts when the check character is displayed so that you can inspect it. When you type **CON** SPACE BAR or SHIFT @, the program recalls the data from memory and performs the check using the check character. This result is stored in memory location 6.

Note: If you want to create an error, change one of the characters in memory while the program is halted and before you press SHIFT @.

This method of checking does have some weaknesses. The check character may not maintain its integrity during transmission or writing. In addition, it will not detect an error which involves 2 or any even number of characters changed in the same way; i.e., the same bit "on" or "off." That's why this method is rarely used by itself. It is usually combined with other detection methods.

The following example illustrates how to write a program for this checking problem:

```
SOME CHECK

                Type                      Comments

0000            RPN  ------------------ Set RPN
0001          CLMEM
0002            CLR
0003            HEX  ------------------ Hexadecimal mode
0004            197 ┐
0012            163 │
0020             50 │                   Generate check character by
0028            127 │                   exclusive OR'ing 5
0036             93 ├------------------ Hex characters in
0044            XOR │                   succession
0045            XOR │
0046            XOR │
0047            XOR ┘
0048            STO  ------------------ Save check character
0049              1
0057          PAUSE
0058            CLR
0059            197 ┐
0067            163 ├------------------ Read back data character stream
0075             50 │
0083            127 │
0091             93 ┘
0099            RCL
0100              1  ------------------ Add check character
0108            XOR                     to stream
0109            XOR
0110            XOR
0111            XOR
0112            XOR  ------------------ Last result should be 0.
0113            STP
```

PROGRAMMING EXAMPLES USING CONVERSIONS

EXAMPLE 30. HAVE YOU CONVERTED YET?

While some conversions are built into this CALCULATOR program, it is still worthwhile to review the logic of the process. For instance, what steps must you perform to convert miles per hour to feet per second?

You might know that there are 5280 feet in a mile. You might also know that there are 60 minutes in an hour. So how can you use this knowledge? A process of logical cancellation of units is one of the simplest ways.

For example: Convert 30 miles per hour to feet per second.

$$\left(\frac{30\text{ mi}}{\text{hr}} * \frac{5280\text{ ft}}{\text{mi}} \Big/ \frac{60\text{ min}}{\text{hr}}\right) \Big/ \frac{60\text{ sec}}{\text{min}}$$

$$= \frac{30\text{ mi}}{\text{hr}} * \frac{5280\text{ ft}}{\text{mi}} * \frac{\text{hr}}{60\text{ min}} * \frac{\text{min}}{60\text{ sec}}$$

Now you can cancel out like units.

$$= \frac{30\ \cancel{\text{mi}}}{\cancel{\text{hr}}} * \frac{5280\text{ ft}}{\cancel{\text{mi}}} * \frac{\cancel{\text{hr}}}{60\ \cancel{\text{min}}} * \frac{\cancel{\text{min}}}{60\text{ sec}}$$

This leaves you with 44 feet per second.

Using this process will not only help you think logically, but it will also provide a check to make sure that you are really going to end up with the units you want. For the conversion units, refer to Appendix D.

The following programmed example performs the same type of conversion as above.

HAVE YOU CONVERTED YET?

	Type	Comments
0000	RPN	RPN mode
0001	CLR	Clear stack
0002	CLMEM	Clear memory
0003	11111	Set indicator for entry—in meters
0011	STP	Wait for entry
0012	XCHGY	Get rid of 11111 indicator
0013	POP	Meters to centimeters
0014	100	
0022	*	
0023	2.54	Centimeters to inches
0031	/	
0032	12	Inches to feet
0040	/	
0041	STO	Display result in
0042	0	memory location 0
0050	22222	Set end indicator and
0058	STO	display it in memory
0059	1	location 1
0067	STP	
0068	GOTO	Loop to beginning of program
0069	0	
0077	STP	

EXAMPLE 31. CIRCLES AND OTHER PLOTS

When you have a series of corresponding points of data, there are at least two coordinate systems in which to plot them. There is the Cartesian (or X vs. Y) coordinate system, and the Polar (or r, Θ) coordinate system (r = radius; Θ = angle theta).

Which one you use is largely a matter of individual choice, but sometimes the equation you need to use contains trigonometric functions. Depending on the function, you might want to convert from one coordinate system to the other because the data would be better represented.

The equations to translate from one form to the other are relatively simple, but of course they must be applied repetitively to all of the data pairs. For convenience, the transformation process has been programmed into your CALCULATOR program.

For your information, however, here are the equations:

From Cartesian to Polar...

$$r = X^2 + Y^2$$

$$\Theta = \text{atan}\left(\frac{Y}{X}\right)$$

From Polar to Cartesian...

$$X = r(\cos\Theta)$$
$$Y = r(\sin\Theta)$$

The program that follows converts the X, Y point 3,5 to polar (r = 5.8309517, Θ = 59.036243 degrees) and then converts Polar r, Θ point 7, 35 degrees to Cartesian (X = 5.7340644, Y = 4.0150351).

CIRCLES AND OTHER PLOTS

	Type	Comments
0000	ALGN	
0001	CLMEM	
0002	CLR	
0003	DEG	
0004	5	Y = 5
0012	PUSH	
0013	3	X = 3
0021	RECT	Convert to polar
0022	STO	Save R
0023	0	
0031	XCHGY	
0032	STO	Save ∠ Θ
0033	1	
0041	PAUSE	Idle a while
0042	PAUSE	
0043	CLR	

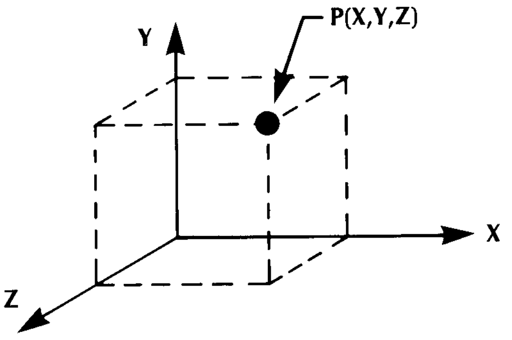
```
           Type              Comments

0044         35 ──────────── Convert
0052       PUSH              35 degrees = ∠ Θ
0053          7 ──────────── R = 7
0061      POLAR ──────────── to rectangular
0062        STP ──────────── complete, X register contains X-coordinate;
0063       GOTO ┐            Y register contains the Y-coordinate
0064          0 ┘─────────── Loop
0072        STP
```

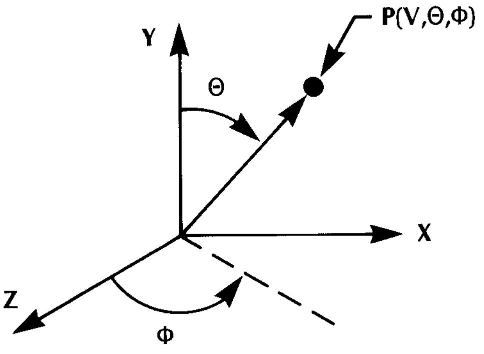
EXAMPLE 32. A 3-D PLOT

This problem is similar to the last except that this problem deals with three dimensions instead of two.

To represent a point in real space requires a coordinate system that is three dimensional. Just as there are equations to convert from Cartesian to Polar and vice-versa, there are equations to convert between these three-dimensional coordinate systems. Take a point (X, Y, Z) in real space:



This point may also be represented as:



p (V, Θ, ϕ)	Where V is the magnitude of the straight line from the origin to the point.

This system of representation is called the Spherical Coordinate system.

Conversion between these two systems is done via the following equations:

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

$$\Theta = \text{acos}\left(\frac{Y}{V}\right) = \frac{Y}{\sqrt{X^2 + Y^2 + Z^2}}$$

$$\phi = \text{atan}\left(\frac{X}{Z}\right)$$

These translate from an extended Cartesian to SPHERICAL.

Use the following to translate back.

$$X = V*\sin\Theta*\sin\phi$$
$$Y = V*\cos\Theta$$
$$Z = V*\sin\Theta*\cos\phi$$

The following program will perform this translation from spherical to extended Cartesian on rectangular coordinates for a point in space observed as $V = 7.2$, $\Theta = 53°$, $\phi = 71°$

$$X = (7.2) \sin(53) \sin(71)$$
$$Y = (7.2) \cos(53)$$
$$Z = (7.2) \sin(53) \cos(71)$$

A 3-D PLOT

	Type	Comments
0000	ALG	
0001	CLMEM	
0002	CLR	
0003	DEG	
0004	53	$\Theta = 53$ degrees
0012	SIN	
0013	*	
0014	71	$\Phi = 71$ degrees
0022	SIN	
0023	*	
0024	7.2	V = 7.2 miles
0032	=	
0033	STO	Save X
0034	0	
0042	53	
0050	COS	
0051	*	
0052	7.2	
0060	=	
0061	STO	Save Y
0062	1	
0070	71	

	Type	Comments
0078	COS	
0079	*	
0080	7.2	
0088	*	
0089	53	
0097	SIN	
0098	=	Save Z
0099	STO	
0100	2	
0108	STP	Run complete

BIT MANIPULATION FUNCTIONS

These functions are used mostly by programmers working in assembly language. In direct mode, they are useful if you are reading a storage dump to calculate the addresses in octal or hexadecimal and also to track the logical operation of instructions executed by the computer that produced the dump. In debugging a computer, you place the program into the calculator in the same sequence with Trace and Pause commands to display what is happening in the program.

The five bit manipulations are **AND, OR, XOR, LSHF,** and **RSHF**. The numbers involved in these functions first must be truncated (automatically), then converted from their normal BCD internal format to a binary number of the length specified by the last BITS command. If the magnitude of a number is too large the ERROR—HEX/OCT OVRFLW will be displayed and the number will be set to 0. The requested operation is performed and the result is converted back to BCD. These functions are intended for use in OCT and HEX modes, but may be used in DEC as well.

LOGICAL AND FUNCTION

AN SPACE BAR or **AND** SPACE BAR or SHIFT **&**

This command computes the logical AND of two numbers, x and y.

Enter	**X Display**
ALG SPACE BAR	
HEX SPACE BAR	
0FFF5 AN SPACE BAR	FFF5
0F=	5

LOGICAL OR FUNCTION

OR SPACE BAR or SHIFT **I**

This command computes the logical OR of two numbers, x and y.

Enter	**X Display**
RPN SPACE BAR	
OCT SPACE BAR	
13 SPACE BAR	13
5 OR SPACE BAR	17

EXCLUSIVE OR FUNCTION

XO SPACE BAR or **XOR** SPACE BAR

This command computes the logical exclusive OR of two numbers, x and y.

Enter	**X Display**
ALG SPACE BAR	
5 XOR 3 =	6

The following table illustrates the binary bit patterns for each of these functions:

y	**x**	**y AND x**	**y OR x**	**y XOR x**
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Figure 11 Binary Bit Pattern

LOGICAL LEFT SHIFT FUNCTION

LS SPACE BAR or **LSHF** SPACE BAR

If x=0 then y is not changed. Otherwise, y is truncated and converted to a binary number of the length specified by the BITS command. ERROR—HEX/OCT OVRFLW will be displayed if y is out of range. Y is shifted left x bits. Zeroes are shifted into the right of y, x is assumed to be in the current base. If x<0 then the absolute value of x is used and a **RSHF** (right shift) is performed instead. X is truncated, but not converted to binary as y is, so x can take on any possible decimal value without HEX/OCT OVRFLW error.

Enter	**X Display**
HEX SPACE BAR	
7F LSHF 1=	FE
DEC SPACE BAR	
5 3RSHF0=	53
HEX SPACE BAR	
BITS32 SPACE BAR	
1 LSHF 10=	10000
7F LSHF 1SHIFT – =	3F

In the above example, the current base is HEX, so the 1 is shifted left 10 (base 16) bits or 16 bits in base 10.

LOGICAL RIGHT SHIFT FUNCTION

RS SPACE BAR or **RSHF** SPACE BAR

If x=0 then y is not changed. Otherwise, y is truncated and shifted right x bits. Zeroes are shifted into the left of y. If x<0 then the absolute value of x is used and a **LSHF** (left shift) is performed instead.

Enter	**X Display**
OCT SPACE BAR	
177777 RSHF 1=	77777
5RSHF1 SHIFT −=	12

11

INPUT/OUTPUT COMMANDS FOR PERIPHERAL DEVICES

The following commands may be used to do Input and Output (I/O) if you have an ATARI printer, disk drive, Program Recorder. If an I/O error occurs, the message ERROR—followed by a number will be displayed. The meanings of these numbers are listed under **ERROR MESSAGES** in Appendix A.

PRINTER FUNCTIONS

PRINTER ON COMMAND

ON SPACE BAR

This command causes the printer to print everything that is displayed in the scroll area of the screen until the OFF command is issued. There are some cases where the printer output will not exactly match the display. If you make an error while entering a command, the characters you entered will be scrolled up in the display, preceded by the > prompt. However, they will not be printed on the printer. In program mode, the next command issued after an error will be further to the left on the printer than it is on the screen.

To print a program listing, you must enter the TRACE command before entering ON and the LIST commands.

PRINTER OFF COMMAND

OFF SPACE BAR

This command turns off the printer. The printer is always off when the CALCULATOR is turned on and after SYSTEM RESET.

When you have completed a program listing, enter OFF and the NOT (NO TRACE) command.

PRINT X COMMAND

PR SPACE BAR or **PRINT** SPACE BAR or SHIFT **?**

This command causes the printer to print the value of any number in the scroll area followed by *** even if the printer is OFF. It also displays that number in the scroll area. This is useful for printing values computed by programs.

ADVANCE PRINTER COMMAND

AD SPACE BAR or **ADV** SPACE BAR

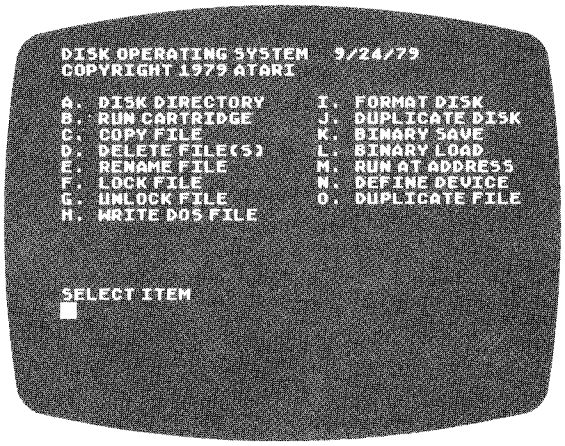
This command puts one blank line on the printer even if the printer is OFF.

USING THE DOS MENU

If you are not familiar with the items listed in the Disk Operating System (DOS) Menu, you should read the *Disk Operating System Manual* that was included with your disk drive. However, you need to know that you can call up and display the DOS Menu **only** as the first operation you perform after you insert the CALCULATOR diskette. So if you want to format a blank diskette on which to save your programs, type the DOS command before doing anything else.

DISPLAYING THE DOS MENU

DOS RETURN



```
DISK OPERATING SYSTEM   9/24/79
COPYRIGHT 1979 ATARI

A. DISK DIRECTORY      I. FORMAT DISK
B. RUN CARTRIDGE       J. DUPLICATE DISK
C. COPY FILE           K. BINARY SAVE
D. DELETE FILE(S)      L. BINARY LOAD
E. RENAME FILE         M. RUN AT ADDRESS
F. LOCK FILE           N. DEFINE DEVICE
G. UNLOCK FILE         O. DUPLICATE FILE
H. WRITE DOS FILE


SELECT ITEM
█
```

Figure 12 DOS Menu

When you type DOS RETURN, the television screen goes blank for a second or two before the DOS Menu appears. You should see a prompt message, SELECT ITEM. Type the letter for the operation you want and press RETURN. For example, to format a blank diskette, you would take the following steps:

1. Remove CALCULATOR diskette from disk drive.
2. Insert blank diskette and close drive door.
3. Type I RETURN.
4. When prompt message reappears, remove the now formatted diskette and insert the CALCULATOR diskette.

You now have a formatted diskette on which you can store programs.

REDISPLAYING THE CALCULATOR SCREEN DISPLAY

To reload the CALCULATOR program, use the Binary Load option on the DOS Menu.

1. Make sure CALCULATOR diskette is inserted in disk drive.
2. Type **L** RETURN.
3. Prompt message LOAD FROM WHAT FILE? appears.
4. Type **AUTO.SYS** RETURN.
5. After a short wait, the CALCULATOR screen display appears.

CASSETTE AND DISK DRIVE FUNCTIONS

SAVE AND LOAD COMMANDS

SAVE filespec
LOAD filespec
SAVEM filespec
LOADM filespec

The message ENTER FILESPEC will be displayed when you enter any of these four commands. The format for a filespec (file specification) is the same as for ATARI BASIC:

Format: Device Name Device Number: Filename.Extension

Examples: D1: CALCZ.INS
C:

Then Atari Program Recorder™ requires only the device name. Consult the Operator's manual for the device you wish to use for more information. Enter a **C** for the Program Recorder and **D** for disk drive device names. If you have more than one disk drive, you must specify device number. The CALCULATOR will check for E:, S:, and K:, in filespecs and display ERROR—NOT VALID COMMAND OR NUMBER if they are used. Other errors in the filespec will be caught by the Operating System built into the computer and an I/O error number will be displayed.

Save Program in File Instruction
S filespec or **SAVE** filespec

This instruction allows you to save all 3072 bytes of program memory in the specified file. Saving on the printer (P:) will produce an unintelligible listing. Stop it by pressing BREAK.

Load Program From File Instruction
LO filespec or **LOAD** filespec

This instruction allows you to load 3072 bytes of program memory from the specified file.

Save Memory in File Instruction
SAVEM filespec

This instruction allows you to save all 100 memory registers (600 bytes) in the specified file. Saving memory on the printer (P:) will produce an unintelligible listing. Stop it by pressing BREAK.

Load Memory From File Instruction
LOADM filespec

This instruction allows you to load all 100 memory registers from the specified file.

A beep (the same as the beep at the end of a program) will sound when SAVE, LOAD, SAVEM, or LOADM is done and the prompt symbol will reappear.

The LOAD and LOADM commands do not check the files they load from to see if they contain valid data, so it is up to you to keep track of what is stored where. For disk files, it is a good idea to use file extensions such as ".MEM" for memory and ".PRG" for programs. If the wrong file is loaded, the program or memory will probably contain garbage and should be cleared using CLPROG or CLMEM. An error message such as ARITHMETIC OVERFLOW or NOT VALID COMMAND OR NUMBER is displayed when incorrect data is displayed or a garbage program is executed. If the file is too short then ERROR—136 (End of File) will be displayed.

The following examples show how to enter these instructions:

Enter	**Comments**
SAVE C:	Save program on cassette
LOAD D:FACT.PRG	Load program (FACT. PRG) from floppy disk file
SAVEM D:TEMP.MEM	Save memory (TEMP.MEM) in floppy disk file
LOADM C:	Load memory from cassette

Data can be passed between CALCULATOR programs and ATARI BASIC programs using the SAVEM and LOADM commands in the CALCULATOR and the GET, POKE, PEEK, and PUT commands in BASIC (see *the ATARI BASIC Reference Manual).* The internal 6-byte BCD representation of each number is stored in the SAVEM file. This is not the format used by PRINT and INPUT in BASIC, so PUT and GET must be used. An example of how to read a SAVEM file into a BASIC array and then write a BASIC array into a file to be loaded with LOADM is given below. The BASIC program could generate data for the CALCULATOR or it could format data produced by the CALCULATOR and print it in some fancy way.

```
10 DIM A$(20),A(99)
20 A=ADR(A$)+20:REM ADDRESS OF ARRAY A
100 REM PROGRAM FRAGMENT TO READ
110 REM CALCULATOR SAVEM FILE INTO ARRAY A
120 ? "ENTER INPUT FILESPEC";:INPUT A$
130 OPEN #1, 4, 0, A$
140 FOR I=0 TO 599:GET #1,B:POKE A+ I,B:NEXT I
150 CLOSE #1
200 REM PROGRAM FRAGMENT TO WRITE
210 REM ARRAY A OUT TO CALCULATOR SAVEM FILE
220 ? "ENTER OUTPUT FILESPEC";:INPUT A$
230 OPEN #1, 8, 0, A$
240 FOR I=0 TO 599:PUT #1, PEEK(A+I):NEXT I
250 CLOSE #1
```

APPENDIX A
ERROR MESSAGES

The following error messages are all preceded by a "beep" sound and the message ERROR— If ERROR—is followed by a number, then it is an I/O (Input/Output) error and the number is the same as the ATARI BASIC error number.

Message	Cause
ARITHMETIC OVERFLOW	Result of a calculation is outside of the range allowed for decimal numbers.
END OF MEMORY	Attempt to access program memory outside the range 0–3071. When the computer is turned on, this error indicates there is not enough RAM (Random Access Memory) in the system.
HEX/OCT OVRFLW	Number is within range allowed for decimal numbers, but is outside the range allowed by BITS setting.
NOT VALID COMMAND OR NUMBER	Entry of one or more characters that are not valid in the current CALCULATOR state. Most common errors of this type: (1) Using an equals symbol in RPN mode. (2) Using a decimal point in OCT or HEX. (3) Entering FIND I when not in CMPND mode.
NUMBER OUT OF RANGE	Number is not in correct range for BITS, FIX, CALL, STO, etc.
STACK EMPTY	Attempted to POP an empty Number Stack. Attempted to RETURN to an empty Call Stack. ALG mode has an empty Operator Stack.
STACK FULL	Attempted to PUSH into a full Number Stack. Attempted to CALL a full Call Stack. Operator Stack full. (This error is unlikely because the Number Stack fills up first.)
TOO MANY CHARACTERS	Attempted to enter number or filespec name that is more than 14 characters long.
TWO OPS IN A ROW	Two binary operators in a row in ALG or ALGN mode.
UNIT MISMATCH	Mixed units in conversion; e.g., attempted to convert kilograms (KG) to miles (MI).

I/O ERRORS

Message	Cause
128	BREAK key abort
130	Nonexistent device
136	End of file (file too small)
138	Peripheral device time out (device disconnected)
139	Device does not acknowledge command
140	Serial bus framing error
142	Serial bus data overrun
143	Serial bus checksum error
144	Device done error (operation not complete)
146	Function not implemented in handler (LOAD from Printer, for example)

DISK I/O ERRORS

Message	Cause
160	Illegal drive number
162	Disk full
163	Fatal I/O error: system error
164	File number mismatch. Disk may be damaged.
165	File name error: system error
167	File locked
169	Directory full
170	File not found

APPENDIX B

NON-ERROR MESSAGES

The following messages are given by the computer either to remind you what to enter or to give helpful information (usually during conversions). They are always displayed when the indicated commands are executed.

Message	**Displayed by**
ENTER 0-8	FIX
ENTER 0-99	All commands that require a memory register (like STO, RCL, etc.)
ENTER 0-3071	All commands that require a program memory address
ENTER 1-32	BITS
ENTER FILESPEC	LOAD, SAVE, LOADM, and SAVEM
ENTER NEW UNITS	Mass, volume, or length commands
TO C	F (Fahrenheit)
TO DEG	CRAD (Convert Radians)
TO F	C (Celsius)
TO POLAR Y, X TO Y=ANGLE, X=R	RECT (Rectangular to Polar)
TO RAD	CDEG (Convert Degrees)
TO RECT Y=ANGLE,X=R TO Y,X	POLAR (Polar to Rectangular)
TO X	Y (Statistics)
TO Y	X (Statistics)

APPENDIX C
FUNCTION SUMMARY

Command Token(s)	Command	Definition
SHIFT (		Left parenthesis
SHIFT)		Right parenthesis
*		Multiplication
/		Division
+		Addition
−		Subtraction
=		Equals
	A or **ABS**	Absolute value
	AC or **ACOS**	Arc cosine
	AD or **ADV**	Advance printer
	AL or **ALG**	Algebraic notation with operator precedence
	ALGN	Algebraic notation with No operator precedence
SHIFT &	**AN** or **AND**	Logical AND
	AS or **ASIN**	Arc sine
	AT or **ATAN**	Arc tangent
	B or **BAL**	Balloon payment
	BI n or **BITS** n	Number of BITS in hex and octal numbers
CTRL ↑	**BST**	Back step
	C	Celsius to Fahrenheit
	CA n or **CALL** n	Call subroutine
	CD or **CDEG**	Convert degrees to radians
SHIFT −	**CH** or **CHGSGN**	Change sign
	CL or **CLINT**	Clear memory for interest calculations
	CLM or **CLMEM**	Clear all of memory
	CLP or **CLPROG**	Clear program memory
	CLR	Clear stack
	CLS or **CLSTAT**	Clear memory for statistics calculations
	CLX	Clear X register
	CM	Centimeters
	CMP or **CMPND**	Compound
	CO or **COMP**	Complement
SHIFT @	**CON** or **CONT**	Continue
	COS	Cosine
	CR or **CRAD**	Convert radians to degrees
	CU or **CUP**	Cups
	D or **DEC**	Decimal base
	DEG	Degree mode
CTRL DELETE BACK S	**DEL**	Delete
SHIFT $	**E** or **END**	End program mode
	ENT or **ENTER**	Enter mode
	EX or **EXPE**	Exponentiation base e
	EXPT or **EXPTEN**	Exponentiation base 10

Command Token(s)	Command	Definition
	F	Fahrenheit to Celsius
SHIFT !	**FA** or **FACT**	Factorial
	FI or **FIND**	Find mode
	FIX n	Fix number of digits to right of decimal point
	FL or **FLOZ**	Fluid ounces
	FR or **FRAC**	Take fractional part
	FT	Feet
	FV	Future value
	FVD or **FVDUE**	Future value, Annuity Due mode
	FVO or **FVORD**	Future value, ordinary annuity mode
	G or **GAL**	Gallons
	GM	Grams
	GO n or **GOTO** n	Go to line number
	H or **HEX**	Hexadecimal base
	I	Interest per period in percent
	IN	Inches
CTRL INSERT	**INS**	Insert
CTRL →	**INSN** or **INSNUM**	Insert number
	INT	Take integer part
	K or **KG**	Kilograms
	KM	Kilometers
	L	Liters
	LB	Pounds
	LI n1 n2 or **LIST** n1 n2	List program
	LISTM r1 r2	List memory
	LN	Natural logarithm
	LO filespec or **LOAD** filespec	Load program from file
	LOADM filespec	Load memory from file
	LOG or **LOGTEN**	Logarithm base 10
	LS or **LSHF**	Logical left shift
	M	Meters
	MI	Miles
SHIFT %	**MO** or **MOD**	Modulo
	N	Number of periods
	NO or **NOP**	No operation
	NOT or **NOTRC**	No trace
	NW or **NWT**	N weighting
	O or **OCT**	Octal base
	OFF	Turn printer off
	ON	Turn printer on
SHIFT I	**OR**	Logical inclusive OR
	OZ	Ounces
	P or **PAUSE**	Pause for ½ second
	PI	Pi
	PM or **PMT**	Payment
	PO or **POLAR**	Polar to rectangular
SHIFT]	**POP**	Pop number stack
	POPC	Pop call stack
SHIFT ∧	**POW** or **POWER**	Exponentiation
SHIFT ?	**PR** or **PRINT**	Print x on printer
SHIFT #	**PRO** or **PROG**	Program mode
SHIFT [	**PU** or **PUSH**	Push X register contents on number stack
	PV	Present value
	PVD or **PVDUE**	Present value, annuity due

Command Token(s)	Command	Definition
	PVO or **PVORD**	Present value, ordinary annuity
	QT	Quarts
	R	Correlation coefficient
	RA or **RAD**	Radian mode
	RCL r	Recall
	RE or **RECIP**	Reciprocal
	RECT	Rectangular to polar
	RO or **ROOT**	Take root of a number
	ROU or **ROUND**	Round off a number
	RP or **RPN**	Reverse Polish Notation
	RS or **RSHF**	Right shift
SHIFT "	**RST**	Reset program counter
SHIFT '	**RU** or **RUN**	Run program
	S filespec or **SAVE** filespec	Save program in file
	SAVEM filespec	Save memory in file
	SI or **SIN**	Sine
	SL or **SLOPE**	Slope
	SM or **SMINUS**	Sigma minus
	SP or **SPLUS**	Sigma plus
	SQ or **SQRT**	Take the square root of a number
	SQU or **SQUARE**	Multiply a number by itself
CTRL ↓	**SS** or **SST**	Single step
	ST r or **STO** r	Store in memory
	STP	Stop
	SU r or **SUM** r	Sum to memory
	T or **TAN**	Tangent
	TB or **TBSP**	Tablespoons
	TR or **TRACE**	Trace program
	TRU or **TRUNC**	Truncate number
	TS or **TSP**	Teaspoons
	X	X to Y
CTRL ←	**XC** or **XCHGY**	Exchange X and Y registers
	XCHM r	Exchange X register and memory
	XE r n or **XEQ** r n	If X equals memory (r) then goto n
	XG r n or **XGE** r n	If X is greater than or equal to memory (r) then goto n
	XL r n or **XLT** r n	If X is less than memory (r) then goto n
	XM or **XMEAN**	Mean of X
	XN r n or **XNE** r n	If X is not equal to memory (r) then goto n
	XO or **XOR**	Logical exclusive OR
	XS or **XSD**	Standard deviation of X
	XV or **XVAR**	Variance of X
	Y	Y to X
	YD	Yards
	YI or **YINT**	Y-intercept
	YM or **YMEAN**	Mean of Y
	YS or **YSD**	Standard deviation of Y
	YV or **YVAR**	Variance of Y

APPENDIX D

CONVERSION FACTORS

The customary units of weight and mass are avoirdupois units unless designated otherwise. The symbol (§) represents the density of a material expressed as a decimal fraction; g equals 980.7 centimeters per second per second.

MULTIPLY	BY	TO OBTAIN
abamperes	10	amperes
abamperes	3×10^{10}	statamperes
abamperes per square centimeter	64.52	amperes per square inch
abampere-turns	10	ampere-turns
abampere-turns	12.57	gilberts
abampere-turns per centimeter	25.40	ampere-turns per inch
abcoulombs	10	coulombs
abcoulombs	3×10^{10}	statcoulombs
abcoulombs per square centimeter	64.52	coulombs per square inch
abfarads	10^{9}	farads
abfarads	10^{15}	microfarads
abfarads	9×10^{20}	statfarads
abhenries	10^{-9}	henries
abhenries	10^{-6}	millihenries
abhenries	$1/9 \times 10^{-20}$	stathenries
abmhos per centimeter cube	$10^{5}/§$	mhos per meter-gram
abmhos per centimeter cube	1.662×10^{2}	mhos per mil foot
abmhos per centimeter cube	10^{3}	megmhos per centimeter cube
abohms	10^{-15}	megohms
abohms	10^{-3}	microhms
abohms	10^{-9}	ohms
abohms	$1/9 \times 10^{-20}$	statohms
abohms per centimeter cube	10^{-3}	microhms per centimeter cube
abohms per centimeter cube	6.015×10^{-3}	ohms per mil foot
abohms per centimeter cube	$10^{-5}§$	ohms per meter-gram
abvolts	$1/3 \times 10^{-10}$	statvolts
abvolts	10^{-8}	volts
acres	43,560	square feet
acres	6,272,640	square inches
acres	4047	square meters
acres	1.562×10^{-3}	square miles
acres	4840	square yards
acre-feet	43,560	cubic-feet
acre-feet	3.259×10^{5}	gallons
amperes	1/10	abamperes
amperes	3×10^{9}	statamperes
amperes per square centimeter	6.452	amperes per square inch
amperes per square inch	0.01550	abamperes per square centimeter
amperes per square inch	0.1550	amperes per square centimeter
amperes per square inch	4.650×10^{8}	statamperes per square centimeter

MULTIPLY	BY	TO OBTAIN
ampere-turns	1/10	abampere-turns
ampere-turns	1.257	gilberts
ampere-turns per centimeter	2.540	ampere-turns per inch
ampere-turns per inch	0.03937	abampere-turns per centimeter
ampere-turns per inch	0.3937	ampere-turns per centimeter
ampere-turns per inch	0.4950	gilberts per centimeter
ares	0.02471	acres
ares	100	square meters
atmospheres	76	centimeters of mercury
atmospheres	29.92	inches of mercury
atmospheres	33.90	feet of water
atmospheres	10,332	kilograms per square meter
atmospheres	14.70	pounds per square inch
atmospheres	1.058	tons per square foot
bars	0.9869	atmospheres
bars	1	dynes per square centimeter
bars	1.020×10^4	kilograms per square meter
bars	2,089	pounds per square foot
bars	14.50	pounds per square inch
board-feet	144 sq. in. × 1 in.	cubic inches
British thermal units (Btu)	778.2	foot-pounds
British thermal units	3.930×10^{-4}	horsepower-hours
British thermal units	1055	joules
British thermal units	0.2520	kilogram-calories
British thermal units	107.6	kilogram-meters
British thermal units	2.930×10^{-4}	kilowatt hours
Btu per minute	12.97	foot-pounds per second
Btu per minute	0.02358	horsepower
Btu per minute	0.01758	kilowatts
Btu per minute	17.58	watts
Btu per square feet per minute	0.1221	watts per square inch
bushels	1.244	cubic feet
bushels	2150	cubic inches
bushels	0.03524	cubic meters
bushels	4	pecks
bushels	64	pints (dry)
bushels	32	quarts (dry)
centares	1	square meters
centigrams	0.01	grams
centiliters	0.01	liters
centimeters	3.281×10^{-2}	feet
centimeters	0.3937	inches
centimeters	0.01	meters
centimeters	6.214×10^{-6}	miles
centimeters	10	millimeters
centimeters	393.7	mils
centimeters	1.094×10^{-2}	yards
centimeter-dynes	1.020×10^{-3}	centimeter-grams
centimeter-dynes	1.020×10^{-8}	meter-kilograms
centimeter-dynes	7.376×10^{-8}	pound-feet
centimeter-grams	980.7	centimeter-dynes
centimeter-grams	10^{-5}	meter-kilograms
centimeter-grams	7.233×10^{-5}	pound-feet

MULTIPLY	BY	TO OBTAIN
centimeters of mercury	0.01316	atmospheres
centimeters of mercury	0.4461	feet of water
centimeters of mercury	136.0	kilograms per square meter
centimeters of mercury	27.85	pounds per square foot
centimeters of mercury	0.1934	pounds per square inch
centimeters per second	1.968	feet per minute
centimeters per second	0.03281	feet per second
centimeters per second	0.036	kilometers per hour
centimeters per second	0.6	meters per minute
centimeters per second	0.02237	miles per hour
centimeters per second	3.728×10^{-4}	miles per minute
centimeters per second per second	0.03281	feet per second per second
centimeters per second per second	0.036	kilometers per hour per second
centimeters per second per second	0.02237	miles per hour per second
circular mils	5.067×10^{-6}	square centimeters
circular mils	7.854×10^{-7}	square inches
circular mils	0.7854	square mils
cord-feet	4 ft × 4 ft × 1 ft	cubic feet
cords	8 ft × 4 ft × 4 ft	cubic feet
coulombs	1/10	abcoulombs
coulombs	3×10^{9}	statcoulombs
coulombs per square inch	0.01550	abcoulombs per square centimeter
coulombs per square inch	0.1550	coulombs per square centimeter
coulombs per square inch	4.650×10^{8}	statcouls, per square centimeter
cubic centimeters	3.531×10^{-5}	cubic feet
cubic centimeters	6.102×10^{-2}	cubic inches
cubic centimeters	10^{-6}	cubic meters
cubic centimeters	1.308×10^{-6}	cubic yards
cubic centimeters	2.642×10^{-4}	gallons
cubic centimeters	10^{-3}	liters
cubic centimeters	2.113×10^{-3}	pints (liquid)
cubic centimeters	1.057×10^{-3}	quarts (liquid)
cubic feet	2.832×10^{4}	cubic centimeters
cubic feet	1728	cubic inches
cubic feet	0.02832	cubic meters
cubic feet	0.03704	cubic yards
cubic feet	7.481	gallons
cubic feet	28.32	liters
cubic feet	59.84	pints (liquid)
cubic feet	29.92	quarts (liquid)
cubic feet per minute	472.0	cubic centimeters per second
cubic feet per minute	0.1247	gallons per second
cubic feet per minute	0.4720	liters per second
cubic feet per minute	62.4	pounds of water per minute
cubic inches	16.39	cubic centimeters
cubic inches	5.787×10^{-4}	cubic feet
cubic inches	1.639×10^{-5}	cubic meters
cubic inches	2.143×10^{-5}	cubic yards
cubic inches	4.329×10^{-3}	gallons
cubic inches	1.639×10^{-2}	liters
cubic inches	1.061×10^{5}	mil-feet
cubic inches	0.03463	pints (liquid)
cubic inches	0.01732	quarts (liquid)
cubic meters	10^{6}	cubic centimeters
cubic meters	35.31	cubic feet

MULTIPLY	BY	TO OBTAIN
cubic meters	61,023	cubic inches
cubic meters	1.308	cubic yards
cubic meters	264.2	gallons
cubic meters	10^3	liters
cubic meters	2113	pints (liquid)
cubic meters	1057	quarts (liquid)
cubic yards	7.646×10^5	cubic centimeters
cubic yards	27	cubic feet
cubic yards	46,656	cubic inches
cubic yards	0.7646	cubic meters
cubic yards	202.0	gallons
cubic yards	764.6	liters
cubic yards	1616	pints (liquid
cubic yards	807.9	quarts (liquid)
cubic yards per minute	0.45	cubic feet per second
cubic yards per minute	3.367	gallons per second
cubic yards per minute	12.74	liters per second
days	24	hours
days	1440	minutes
days	86,400	seconds
decigrams	0.1	grams
deciliters	0.1	liters
decimeters	0.1	meters
degrees (angle)	60	minutes
degrees (angle)	0.01745	radians
degrees (angle)	3600	seconds
degrees per second	0.01745	radians per second
degrees per second	0.1667	revolutions per minute
degrees per second	0.002778	revolutions per second
dekagrams	10	grams
dekaliters	10	liters
dekameters	10	meters
drams	1.772	grams
drams	0.0625	ounces
dynes	1.020×10^{-3}	grams
dynes	7.233×10^{-5}	poundals
dynes	2.248×10^{-6}	pounds
dynes per square centimeter	1	bars
ergs	9.480×10^{-11}	British thermal units
ergs	1	dyne-centimeters
ergs	7.378×10^{-8}	foot-pounds
ergs	1.020×10^{-3}	gram-centimeters
ergs	10^{-7}	joules
ergs	2.389×10^{-11}	kilogram-calories
ergs	1.020×10^{-8}	kilogram-meters
ergs per second	5.688×10^{-9}	British thermal units per minute
ergs per second	4.427×10^{-6}	foot-pounds per minute
ergs per second	7.378×10^{-8}	foot-pounds per second
ergs per second	1.341×10^{-10}	horsepower
ergs per second	1.433×10^{-9}	kilogram-calories per minute
ergs per second	10^{-10}	kilowatts

MULTIPLY	BY	TO OBTAIN
farads	10^{-9}	abfarads
farads	10^{6}	microfarads
farads	9×10^{11}	statfarads
fathoms	6	feet
feet	30.48	centimeters
feet	12	inches
feet	0.3048	meters
feet	1.894×10^{-4}	miles
feet	1/3	yards
feet of water	0.02950	atmospheres
feet of water	0.8826	inches of mercury
feet of water	304.8	kilograms per square meter
feet of water	62.43	pounds per square foot
feet of water	0.4335	pounds per square inch
feet per minute	0.5080	centimeters per second
feet per minute	0.01667	feet per second
feet per minute	0.01829	kilometers per hour
feet per minute	0.3048	meters per minute
feet per minute	0.01136	miles per hour
feet per second	30.48	centimeters per second
feet per second	1.097	kilometers per hour
feet per second	0.5921	knots
feet per second	18.29	meters per minute
feet per second	0.6818	miles per hour
feet per second	0.01136	miles per minute
feet per 100 feet	1	percent grade
feet per second per second	30.48	centimeters per second per second
feet per second per second	1.097	kilometers per hour per second
feet per second per second	0.3048	meters per second per second
feet per second per second	0.6818	miles per hour per second
foot-pounds	1.285×10^{-3}	British thermal units
foot-pounds	1.356×10^{7}	ergs
foot-pounds	5.050×10^{-7}	horsepower-hours
foot-pounds	1.356	joules
foot-pounds	3.238×10^{-4}	kilogram-calories
foot-pounds	0.1383	kilogram-meters
foot-pounds	3.766×10^{-7}	kilowatt-hours
foot-pounds per minute	1.285×10^{-3}	British thermal units per minute
foot-pounds per minute	0.01667	foot-pounds per second
foot-pounds per minute	3.030×10^{-5}	horsepower
foot-pounds per minute	3.238×10^{-4}	kilogram-calories per minute
foot-pounds per minute	2.260×10^{-5}	kilowatts
foot-pounds per second	7.712×10^{-2}	British thermal units per minute
foot-pounds per second	1.818×10^{-3}	horsepower
foot-pounds per second	1.943×10^{-2}	kilogram-calories per minute
foot-pounds per second	1.356×10^{-3}	kilowatts
furlongs	40	rods
gallons	3785	cubic centimeters
gallons	0.1337	cubic feet
gallons	231	cubic inches
gallons	3.785×10^{-3}	cubic meters
gallons	4.951×10^{-3}	cubic yards
gallons	3.785	liters

MULTIPLY	BY	TO OBTAIN
gallons	8	pints (liquid)
gallons	4	quarts (liquid)
gallons per minute	2.228×10^{-3}	cubic feet per second
gallons per minute	0.06308	liters per second
gausses	6.452	lines per square inch
gilberts	0.07958	abampere-turns
gilberts	0.7958	ampere-turns
gilberts per centimeter	2.021	ampere-turns per inch
gills	0.1183	liters
gills	0.25	pints (liquid)
grains	1	grains (av.)
grains	0.06480	grams
grains	0.04167	pennyweights (troy)
grams	980.7	dynes
grams	15.43	grains
grams	10^{-3}	kilograms
grams	10^{3}	miligrams
grams	0.03527	ounces
grams	0.03215	ounces (troy)
grams	0.07093	poundals
grams	2.205×10^{-3}	pounds
gram-calories (IT)	3.968×10^{-3}	British thermal units
gram-centimeters	9.297×10^{-8}	British thermal units
gram-centimeters	980.7	ergs
gram-centimeters	7.235×10^{-5}	foot-pounds
gram-centimeters	9.807×10^{-5}	joules
gram-centimeters	2.343×10^{-8}	kilogram-calories
gram-centimeters	10^{-5}	kilogram-meters
grams per centimeter	5.600×10^{-3}	pounds per inch
grams per cubic centimeter	62.43	pounds per cubic foot
grams per cubic centimeter	0.03613	pounds per cubic inch
grams per cubic centimeter	3.405×10^{-7}	pounds per mil-foot
hectares	2.471	acres
hectares	1.076×10^{5}	square feet
hectograms	100	grams
hectoliters	100	liters
hectometers	100	meters
hectowatts	100	watts
hemispheres (solid angle)	0.5	sphere
hemispheres (solid angle)	4	spherical right angles
hemispheres (solid angle)	6.283	steradians
henries	10^{9}	abhenries
henries	10^{3}	millihenries
henries	$1/9 \times 10^{-11}$	stathenries
horsepower	42.40	British thermal units per minute
horsepower	33,000	foot-pounds per minute
horsepower	550	foot-pounds per second
horsepower	1.014	horsepower (metric)
horsepower	10.68	kilogram-calories per minute
horsepower	0.7457	kilowatts
horsepower	745.7	watts
horsepower (boiler)	33.520	British thermal units per hour
horsepower (boiler)	9.804	kilowatts
horsepower-hours	2544	British thermal units

MULTIPLY	BY	TO OBTAIN
horsepower-hours	1.98×10^6	foot-pounds
horsepower-hours	2.684×10^6	joules
horsepower-hours	641.1	kilogram-calories
horsepower-hours	2.737×10^5	kilogram-meters
horsepower-hours	0.7455	kilowatt-hours
hours	4.167×10^{-2}	days
hours	60	minutes
hours	3600	seconds
hours	5.952×10^{-3}	weeks
inches	2.540	centimeters
inches	8.333×10^{-2}	feet
inches	1.578×10^{-5}	miles
inches	10^3	mils
inches	2.778×10^{-2}	yards
inches of mercury	0.03342	atmospheres
inches of mercury	1.133	feet of water
inches of mercury	345.3	kilograms per square meter
inches of mercury	70.73	pounds per square foot
inches of mercury	0.4912	pounds per square inch
inches of water	0.002458	atmospheres
inches of water	0.07355	inches of mercury
inches of water	25.40	kilograms per square meter
inches of water	0.5781	ounces per square inch
inches of water	5.204	pounds per square foot
inches of water	0.03613	pounds per square inch
joules (Int.)	9.480×10^{-4}	British thermal units
joules (Int.)	10^7	ergs
joules (Int.)	0.7378	foot-pounds
joules (Int.)	2.389×10^{-4}	kilogram-calories
joules (Int.)	0.1020	kilogram-meters
joules (Int.)	2.778×10^{-4}	watt-hours
kilograms	980,665	dynes
kilograms	10^3	grams
kilograms	70.93	poundals
kilograms	2.205	pounds
kilograms	1.102×10^{-3}	tons (short)
kilogram-calories	3.968	British thermal units
kilogram-calories	3088	foot-pounds
kilogram-calories	1.560×10^{-3}	horsepower-hours
kilogram-calories	4186	joules
kilogram-calories	427.0	kilogram-meters
kilogram-calories	1.163×10^{-3}	kilowatt-hours
kilogram-calories per minute	51.47	foot-pounds per second
kilogram-calories per minute	0.09358	horsepower
kilogram-calories per minute	0.06977	kilowatts
kilogram-centimeters squared	2.373×10^{-3}	pounds-feet squared
kilogram-centimeters squared	0.3417	pounds-inches squared
kilogram-meters	9.294×10^{-3}	British thermal units
kilogram-meters	9.804×10^7	ergs
kilogram-meters	7.233	foot-pounds
kilogram-meters	9.804	joules
kilogram-meters	2.342×10^{-3}	kilogram-calories

MULTIPLY	BY	TO OBTAIN
kilogram-meters	2.723×10^{-6}	kilowatt-hours
kilograms per cubic meter	10^{-3}	grams per cubic centimeter
kilograms per cubic meter	0.06243	pounds per cubic foot
kilograms per cubic meter	3.613×10^{-5}	pounds per cubic inch
kilograms per cubic meter	3.405×10^{-10}	pounds per mil foot
kilograms per meter	0.6720	pound per foot
kilograms per square meter	9.678×10^{-5}	atmospheres
kilograms per square meter	98.07×10^{-6}	bars
kilograms per square meter	3.281×10^{-3}	feet of water
kilograms per square meter	2.896×10^{-3}	inches of mercury
kilograms per square meter	0.2048	pounds per square foot
kilograms per square meter	1.422×10^{-3}	pounds per square inch
kilograms per square millimeter	10^{6}	kilograms per square meter
kilolines	10^{3}	maxwells
kiloliters	10^{3}	liters
kilometers	10^{5}	centimeters
kilometers	3281	feet
kilometers	3.937×10^{4}	inches
kilometers	10^{3}	meters
kilometers	0.6214	miles
kilometers	1094	yards
kilometers per hour	27.78	centimeters per second
kilometers per hour	54.68	feet per minute
kilometers per hour	0.9113	feet per second
kilometers per hour	0.5396	knots
kilometers per hour	16.67	meters per minute
kilometers per hour	0.6214	miles per hour
kilometers per hour per second	27.78	centimeters per second per second
kilometers per hour per second	0.9113	feet per second per second
kilometers per hour per second	0.2778	meters per second per second
kilometers per hour per second	0.6214	miles per hour per second
kilometers per minute	60	kilometers per hour
kilowatts	56.88	British thermal units per minute
kilowatts	4.427×10^{4}	foot-pounds per minute
kilowatts	737.8	foot-pounds per second
kilowatts	1.341	horsepower
kilowatts	14.33	kilogram-calories per minute
kilowatts	10^{3}	watts
kilowatt-hours	3413	British thermal units
kilowatt-hours	2.656×10^{6}	foot-pounds
kilowatt-hours	1.341	horsepower-hours
kilowatt-hours	3.6×10^{6}	joules
kilowatt-hours	860	kilogram-calories
kilowatt-hours	3.672×10^{5}	kilogram-meters
knots (length)	6080	feet
knots (length)	1.853	kilometers
knots (length)	1.152	miles
knots (length)	2027	yards
knots (speed)	51.48	centimeters per second
knots (speed)	1.689	feet per second
knots (speed)	.1853	kilometers per hour
knots (speed)	1.152	miles per hour
lines per square centimeter	1	gausses
lines per square inch	0.1550	gausses

MULTIPLY	BY	TO OBTAIN
links (engineer's)	12	inches
links (surveyor's)	7.92	inches
liters	10^3	cubic centimeters
liters	0.03531	cubic feet
liters	61.02	cubic inches
liters	10^{-3}	cubic meters
liters	1.308×10^{-3}	cubic yards
liters	0.2642	gallons
liters	2.113	pints (liquid)
liters	1.057	quarts (liquid)
liters per minute	5.885×10^{-4}	cubic feet per second
liters per minute	4.403×10^{-3}	gallons per second
$\log_{10}N$	2.303	log e N or *ln* N
log e N or *ln* N	0.4343	$\log_{10}N$
lumens per square foot	1	foot-candles
maxwells	10^{-3}	kilolines
megalines	10^6	maxwells
megmhos per centimeter cube	10^{-3}	abmhos per centimeter cube
megmhos per centimeter cube	2.540	megmhos per inch cube
megmhos per centimeter cube	10^2/§	mhos per meter-gram
megmhos per centimeter cube	0.1662	mhos per mil foot
megmhos per inch cube	0.3937	megmhos per centimeter cube
megohms	10^6	ohms
meters	100	centimeters
meters	3.281	feet
meters	39.37	inches
meters	10^{-3}	kilometers
meters	6.214×10^{-4}	miles
meters	10^3	millimeters
meters	1.094	yards
meter-kilograms	9.807×10^7	centimeter-dynes
meter-kilograms	10^5	centimeter-grams
meter-kilograms	7.233	pound-feet
meters per minute	1.667	centimeters per second
meters per minute	3.281	feet per minute
meters per minute	0.05468	feet per second
meters per minute	0.06	kilometers per hour
meters per minute	0.03728	miles per hour
meters per second	196.8	feet per minute
meters per second	3.281	feet per second
meters per second	3.6	kilometers per hour
meters per second	0.06	kilometers per minute
meters per second	2.237	miles per hour
meters per second	0.03728	miles per minute
meters per second per second	3.281	feet per second per second
meters per second per second	3.6	kilometers per hour per second
meters per second per second	2.237	miles per hour per second
mhos per meter-gram	10^{-5}§	abmhos per centimeter cube
mhos per meter-gram	10^{-2}§	megmhos per centimeter cube
mhos per meter-gram	2.540×10^{-2}§	megmhos per inch cube
mhos per meter-gram	1.662×10^{-3}§	mhos per mil foot
mhos per mil foot	6.015×10^{-3}	abmhos per centimeter cube
mhos per mil foot	6.015	megmhos per centimeter cube
mhos per mil foot	15.28	megmhos per inch cube

MULTIPLY	BY	TO OBTAIN
mhos per mil foot	601.5/§	mhos per meter-gram
microfarads	10^{-15}	abfarads
microfarads	10^{-6}	farads
microfarads	9×10^{5}	statfarads
micrograms	10^{-6}	grams
microliters	10^{-6}	liters
microhms	10^{3}	abohms
microhms	10^{-12}	megohms
microhms	10^{-6}	ohms
microhms	$1/9 \times 10^{-17}$	statohms
microhms per centimeter cube	10^{3}	abohms per centimeter cube
microhms per centimeter cube	0.3937	microhms per inch cube
microhms per centimeter cube	10^{-2}§	ohms per meter-gram
microhms per centimeter cube	6.015	ohms per mil foot
microhms per inch cube	2.540	microhms per centimeter cube
microns	10^{-6}	meters
miles	1.609×10^{5}	centimeters
miles	5280	feet
miles	6.336×10^{4}	inches
miles	1.609	kilometers
miles	1760	yards
miles per hour	44.70	centimeters per second
miles per hour	88	feet per minute
miles per hour	1.467	feet per second
miles per hour	1.609	kilometers per hour
miles per hour	0.8684	knots
miles per hour	26.82	meters per minute
miles per hour per second	44.70	centimeters per second per second
miles per hour per second	1.467	feet per second per second
miles per hour per second	1.609	kilometers per hour per second
miles per hour per second	0.4470	meters per second per second
miles per minute	2682	centimeters per second
miles per minute	88	feet per second
miles per minute	1.609	kilometers per minute
miles per minute	52.10	knots
miles per minute	60	miles per hour
mil-feet	9.425×10^{-6}	cubic inches
milliers	10^{3}	kilograms
milligrams	10^{-3}	grams
millihenries	10^{6}	abhenries
millihenries	10^{-3}	henries
millihenries	$1/9 \times 10^{-14}$	stathenries
milliliters	10^{-3}	liters
millimeters	0.1	centimeters
millimeters	3.281×10^{-3}	feet
millimeters	0.03937	inches
millimeters	6.214×10^{-7}	miles
millimeters	39.37	mils
millimeters	1.094×10^{-3}	yards
mils	2.540×10^{-3}	centimeters
mils	8.333×10^{-5}	feet
mils	10^{-3}	inches
mils	2.540×10^{-8}	kilometers
mils	2.778×10^{-5}	yards

MULTIPLY	BY	TO OBTAIN
miner's inches	1.5	cubic feet per minute
minutes	6.944×10^{-4}	days
minutes	1.667×10^{-2}	hours
minutes	9.921×10^{-5}	weeks
minutes (angle)	2.909×10^{-4}	radians
minutes (angle)	60	seconds (angle)
months	30.42	days
months	730	hours
months	43.800	minutes
months	2.628×10^{6}	seconds
myriagrams	10	kilograms
myriameters	10	kilometers
myriameters	10	kilowatts
ohms	10^{9}	abohms
ohms	10^{-6}	megohms
ohms	10^{6}	microhms
ohms	$1/9 \times 10^{-11}$	statohms
ohms per meter-gram	10^{5}/§	abohms per centimeter cube
ohms per meter-gram	10^{2}/§	microhms per centimeter cube
ohms per meter-gram	39.37/§	microhms per inch cube
ohms per meter-gram	601.5/§	ohms per mil foot
ohms per mil-foot	166.2	abohms per centimeter cube
ohms per mil-foot	0.1662	microhms per centimeter cube
ohms per mil-foot	0.06524	microhms per inch cube
ohms per mil-foot	1.662×10^{-3}§	ohms per meter-gram
ounces	16	drams
ounces	437.5	grains
ounces	28.35	grams
ounces	0.0625	pounds
ounces (fluid)	1.805	cubic inches
ounces (fluid)	0.02957	liters
ounces (troy)	480	grains
ounces (troy)	31.10	grams
ounces (troy)	20	pennyweights (troy)
ounces (troy)	0.08333	pounds (troy)
ounces per square inch	0.0625	pounds per square inch
pennyweights (troy)	24	grains
pennyweights (troy)	1.555	grams
pennyweights (troy)	0.05	ounces (troy)
perches (masonry)	24.75	cubic feet
pints (dry)	33.60	cubic inches
pints (liquid)	473.2	cubic centimeters
pints (liquid)	1.671×10^{-2}	cubic feet
pints (liquid)	28.87	cubic inches
pints (liquid)	4.732×10^{-4}	cubic meters
pints (liquid)	6.189×10^{-4}	cubic yards
pints (liquid)	0.125	gallons
pints (liquid)	0.4732	liters
poundals	13,826	dynes
poundals	14.10	grams
poundals	0.03108	pounds
pounds	444,823	dynes

MULTIPLY	BY	TO OBTAIN
pounds	7000	grains
pounds	453.6	grams
pounds	16	ounces
pounds (troy)	32.17	poundals
pounds	0.8229	pounds (av.)
pound-feet	1.356×10^{7}	centimeter-dynes
pound-feet	13,825	centimeter-grams
pound-feet	0.1383	meter-kilograms
pounds-feet squared	421.3	kilograms-centimeters squared
pounds-feet squared	144	pounds-inches squared
pounds-inches squared	2.926	kilograms-centimeters squared
pounds-inches squared	6.945×10^{-3}	pounds-feet squared
pounds of water	0.01602	cubic feet
pounds of water	27.68	cubic inches
pounds of water	0.1198	gallons
pounds of water per minute	2.669×10^{-4}	cubic feet per second
pounds per cubic foot	0.01602	grams per cubic centimeter
pounds per cubic foot	16.02	kilograms per cubic meter
pounds per cubic foot	5.787×10^{-4}	pounds per cubic inch
pounds per cubic foot	5.456×10^{-9}	pounds per mil foot
pounds per cubic inch	27.68	grams per cubic centimeter
pounds per cubic inch	2.768×10^{4}	kilograms per cubic meter
pounds per cubic inch	1728	pounds per cubic foot
pounds per cubic inch	9.425×10^{-6}	pounds per mil foot
pounds per foot	1.488	kilograms per meter
pounds per inch	178.6	grams per centimeter
pounds per mil foot	2.306×10^{6}	grams per cubic centimeter
pounds per square foot	4.725×10^{-4}	atmospheres
pounds per square foot	0.01602	feet of water
pounds per square foot	1.414×10^{-2}	inches of mercury
pounds per square foot	4.882	kilograms per square meter
pounds per square foot	6.944×10^{-3}	pounds per square inch
pounds per square inch	0.06804	atmospheres
pounds per square inch	2.307	feet of water
pounds per square inch	2.036	inches of mercury
pounds per square inch	703.1	kilograms per square meter
pounds per square inch	144	pounds per square foot
quadrants (angle)	90	degrees
quadrants (angle)	5400	minutes
quadrants (angle)	1.571	radians
quarts (dry)	67.20	cubic inches
quarts (liquid)	946.4	cubic centimeters
quarts (liquid)	3.342×10^{-2}	cubic feet
quarts (liquid)	57.75	cubic inches
quarts (liquid)	9.464×10^{-4}	cubic meters
quarts (liquid)	1.238×10^{-3}	cubic yards
quarts (liquid)	0.25	gallons
quarts (liquid)	0.9463	liters
quintals	100	pounds
quires	25	sheets
radians	57.30	degrees
radians	3438	minutes
radians	0.6366	quadrants

MULTIPLY	BY	TO OBTAIN
radians per second	57.30	degrees per second
radians per second	9.549	revolutions per minute
radians per second	0.1592	revolutions per second
radians per second per second	573.0	revolutions per minute per minute
radians per second per second	9.549	revolutions per minute per second
radians per second per second	0.1592	revolutions per second per second
reams	500	sheets
revolutions	360	degrees
revolutions	4	quadrants
revolutions	6.283	radians
revolutions per minute	6	degrees per second
revolutions per minute	0.1047	radians per second
revolutions per minute	0.01667	revolutions per second
revolutions per minute per minute	1.745×10^{-3}	radians per second per second
revolutions per minute per minute	0.01667	revolutions per minute per second
revolutions per minute per minute	2.778×10^{-4}	revolutions per second per second
revolutions per second	360	degrees per second
revolutions per second	6.283	radians per second
revolutions per second	60	revolutions per minute
revolutions per second per second	6.283	radians per second per second
revolutions per second per second	3600	revolutions per minute per minute
revolutions per second per second	60	revolutions per minute per second
rods	16.5	feet
seconds	1.157×10^{-5}	days
seconds	2.778×10^{-4}	hours
seconds	1.667×10^{-2}	minutes
seconds	1.654×10^{-6}	weeks
seconds (angle)	4.848×10^{-6}	radians
spheres (solid angle)	12.57	steradians
spherical right angles	0.25	hemispheres
spherical right angles	0.125	spheres
spherical right angles	1.571	steradians
square centimeters	1.973×10^{5}	circular mils
square centimeters	1.076×10^{-3}	square feet
square centimeters	0.1550	square inches
square centimeters	10^{-4}	square meters
square centimeters	3.861×10^{-11}	square miles
square centimeters	100	square millimeters
square centimeters	1.196×10^{-4}	square yards
square centimeters-centimeters squared	0.02402	square inches-inches squared
square feet	2.296×10^{-5}	acres
square feet	1.833×10^{8}	circular mils
square feet	929.0	square centimeters
square feet	144	square inches
square feet	0.09290	square meters
square feet	3.587×10^{-8}	square miles
square feet	1/9	square yards
square feet-feet squared	2.074×10^{4}	square inches-inches squared
square inches	1.273×10^{6}	circular mils
square inches	6.452	square centimeters
square inches	6.944×10^{-3}	square feet
square inches	645.2	square millimeters
square inches	10^{6}	square mils
square inches	7.716×10^{-4}	square yards

MULTIPLY	BY	TO OBTAIN
square inches-inches squared	41.62	square centimeters-centimeters squared
square inches-inches squared	4.823×10^{-5}	square feet-feet squared
square kilometers	247.1	acres
square kilometers	10.76×10^{6}	square feet
square kilometers	1.550×10^{9}	square inches
square kilometers	10^{6}	square meters
square kilometers	0.3861	square miles
square kilometers	1.196×10^{6}	square yards
square meters	2.471×10^{-4}	acres
square meters	10.76	square feet
square meters	1550	square inches
square meters	3.861×10^{-7}	square miles
square meters	1.196	square yards
square miles	640	acres
square miles	27.88×10^{6}	square feet
square miles	2.590	square kilometers
square miles	3.098×10^{6}	square yards
square millimeters	1.973×10^{3}	circular mils
square millimeters	0.01	square centimeters
square millimeters	1.550×10^{-3}	square inches
square mils	1.273	circular mils
square mils	6.452×10^{-6}	square centimeters
square mils	10^{-6}	square inches
square yards	2.066×10^{-4}	acres
square yards	9	square feet
square yards	1296	square inches
square yards	0.8361	square meters
square yards	3.228×10^{-7}	square miles
statamperes	$1/3 \times 10^{-10}$	abamperes
statamperes	$1/3 \times 10^{-9}$	amperes
statcoulombs	$1/3 \times 10^{-10}$	abcoulombs
statcoulombs	$1/3 \times 10^{-10}$	coulombs
statfarads	$1/9 \times 10^{-20}$	abfarads
statfarads	$1/9 \times 10^{-11}$	farads
statfarads	$1/9 \times 10^{-5}$	microfarads
stathenries	9×10^{20}	abhenries
stathenries	9×10^{11}	henries
stathenries	9×10^{14}	millihenries
statohms	9×10^{20}	abohms
statohms	9×10^{5}	megohms
statohms	9×10^{17}	microhms
statohms	9×10^{11}	ohms
statvolts	3×10^{10}	abvolts
statvolts	300	volts
steradians	0.1592	hemispheres
steradians	0.07958	spheres
steradians	0.6366	spherical right angles
steres	10^{3}	liters
temperature (degrees Celsius) +273	1	absolute temperature (degrees Celsius)
temperature (degrees Celsius) +17.8	1.8	temperature (degrees Fahrenheit)
temperature (degrees Fahrenheit) +460	1	absolute temperature (degrees Celsius)
temperature (degrees Fahrenheit) −32	5/9	temperature (degrees Celsius)
tons (long)	1016	kilograms
tons (long)	2240	pounds

MULTIPLY	BY	TO OBTAIN
tons (metric)	10^3	kilograms
tons (metric)	2205	pounds
tons (short)	907.2	kilograms
tons (short)	2000	pounds
tons (short) per square foot	9765	kilograms per square meter
tons (short) per square foot	13.89	pounds per square inch
tons (short) per square inch	1.406×10^6	kilograms per square meter
tons (short) per square inch	2000	pounds per square inch
volts	10^8	abvolts
volts	1/300	statvolts
volts per inch	3.937×10^7	abvolts per centimeter
volts per inch	1.312×10^{-3}	statvolts per centimeter
watts	0.05688	British thermal units per minute
watts	10^7	ergs per second
watts	44.27	foot-pounds per minute
watts	0.7378	foot-pounds per second
watts	1.341×10^{-3}	horsepower
watts	0.01433	kilogram-calories per minute
watts	10^{-3}	kilowatts
watt-hours	3.413	British thermal units
watt-hours	2656	foot-pounds
watt-hours	1.341×10^{-3}	horsepower-hours
watt-hours	0.860	kilogram-calories
watt-hours	367.2	kilogram-meters
watt-hours	10^{-3}	kilowatt-hours
webers	10^8	maxwells
weeks	168	hours
weeks	10,080	minutes
weeks	604,800	seconds
yards	91.44	centimeters
yards	3	feet
yards	36	inches
yards	0.9144	meters
yards	5.682×10^{-4}	miles
years (common)	365	days
years (common)	8760	hours
years (leap)	366	days
years (leap)	8784	hours

INDEX

LIMITED WARRANTY ON MEDIA AND HARDWARE ACCESSORIES.

We, Atari, Inc., guarantee to you, the original retail purchaser, that the medium on which the APX program is recorded and any hardware accessories sold by APX are free from defects for thirty days from the date of purchase. Any applicable implied warranties, including warranties of merchantability and fitness for a particular purpose, are also limited to thirty days from the date of purchase. Some states don't allow limitations on a warranty's period, so this limitation might not apply to you. If you discover such a defect within the thirty-day period, call APX for a Return Authorization Number, and then return the product along with proof of purchase date to APX. We will repair or replace the product at our option.

You void this warranty if the APX product: (1) has been misused or shows signs of excessive wear; (2) has been damaged by use with non-ATARI products; or (3) has been serviced or modified by anyone other than an Authorized ATARI Service Center. Incidental and consequential damages are not covered by this warranty or by any implied warranty. Some states don't allow exclusion of incidental or consequential damages, so this exclusion might not apply to you.

DISCLAIMER OF WARRANTY AND LIABILITY ON COMPUTER PROGRAMS.

Most APX programs have been written by people not employed by Atari, Inc. The programs we select for APX offer something of value that we want to make available to ATARI Home Computer owners. To offer these programs to the widest number of people economically, we don't put APX products through rigorous testing. Therefore, APX products are sold "as is", and we do not guarantee them in any way. In particular, we make no warranty, express or implied, including warranties of merchantability and fitness for a particular purpose. We are not liable for any losses or damages of any kind that result from use of an APX product.

ATARI PROGRAM EXCHANGE

REVIEW FORM

We're interested in your experiences with APX programs and documentation, both favorable and unfavorable. Many software authors are willing and eager to improve their programs if they know what users want. And, of course, we want to know about any bugs that slipped by us, so that the software author can fix them. We also want to know whether our documentation is meeting your needs. You are our best source for suggesting improvements! Please help us by taking a moment to fill in this review sheet. Fold the sheet in thirds and seal it so that the address on the bottom of the back becomes the envelope front. Thank you for helping us!

1. Name and APX number of program ______________________________

2. If you have problems using the program, please describe them here.

____

____

____

3. What do you especially like about this program?

____

____

____

4. What do you think the program's weaknesses are?

____

____

____

5. How can the catalog description be more accurate and/or comprehensive?

____

____

6. On a scale of 1 to 10, 1 being "poor" and 10 being "excellent", please rate the following aspects of this program:

______ Easy to use
______ User-oriented (e.g., menus, prompts, clear language)
______ Enjoyable
______ Self-instructive
______ Useful (non-game software)
______ Imaginative graphics and sound

7. Describe any technical errors you found in the user instructions (please give page numbers).

8. What did you especially like about the user instructions?

9. What revisions or additions would improve these instructions?

10. On a scale of 1 to 10, 1 representing "poor" and 10 representing "excellent", how would you rate the user instructions and why?

11. Other comments about the software or user instructions:

```
 -----
|     |
|STAMP|
|     |
 -----
```

ATARI Program Exchange
Attn: Publications Dept.
P.O. Box 50047
60 E. Plumeria Drive
San Jose, CA 95150

[seal here]