Quasar Company
9401 W. Grand Ave., Franklin Park, Illinois 60131
Made in U.S.A.

Quasar®

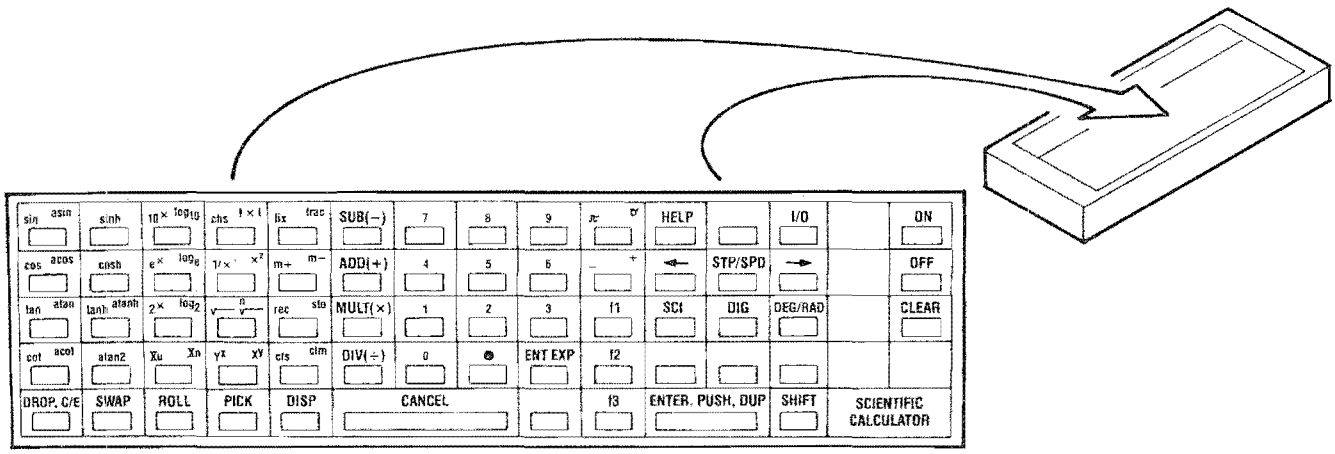
T.M.

HHC

Scientific Calculator

AN ADVANCED MULTI-FUNCTION CALCULATOR FOR SCIENCE AND ENGINEERING

LK330TES



Scientific Calculator

an advanced multi-function calculator for science and engineering

TABLE OF CONTENTS

SECTION 1. INTRODUCTION TO THE SCIENTIFIC CALCULATOR

SCIENTIFIC CALCULATOR FEATURES 1-1
REQUIREMENTS FOR OPERATION 1-2
HOW TO USE THIS MANUAL 1-2

SECTION 2. GETTING STARTED

GETTING SET UP 2-1
BECOMING ACQUAINTED WITH THE STACK 2-1
INPUTTING NUMBERS 2-3
CONTROLLING THE OUTPUT FORMAT 2-5
USING MEMORY 2-6
ERRORS AND WARNINGS 2-7

SECTION 3. USING THE OPERATION KEYS

CONTROL OPERATIONS 3-1
  Memory Control 3-1
  Stack Control 3-3
OPERATIONAL CONTROL 3-5
  Inputting Numbers 3-7
ELEMENTARY OPERATIONS 3-7
MATHEMATICAL FUNCTIONS 3-10
  Trigometric, Inverse Trigometric, Hyperbolic, Inverse Hyperbolic 3-10
  Power, Logarithm, and Root Functions 3-11
STATISTICAL FUNCTIONS 3-12
PROGRAMMABLE KEYS 3-13

SECTION 4. PROBLEM-SOLVING EXAMPLES

TRIGONOMETRIC FUNCTIONS 4-1
LOGARITHMIC FUNCTIONS 4-2
ENGINEERING FUNCTIONS 4-3
  FIX and FRAC 4-3
  Modulo (y mod x) 4-4
  Degrees/Radians 4-5
  Percentages and Conversion Factors 4-6
  Factorials 4-6

STATISTICAL FUNCTIONS 4-7
Summations 4-7
Arithmetic Mean 4-9
Harmonic Mean 4-9
Standard Deviation 4-10
Means and Sums of Squares 4-11
An Investment Problem 4-14

SECTION 5. FLOATING-POINT NUMBERS AND ALGORITHMS

FLOATING-POINT NUMBERS AND ALGORITHMS . 5-1
MATHEMATICAL FUNCTIONS 5-3
GENERATION OF RANDOM DEVIATES 5-3
Uniform Psedo-random Deviates 5-3
Normal Psedo-random Deviates 5-4

SECTION 6. MACHINE PARAMETERS

SECTION 7. OPERATION KEY PARAMETERS

GENERAL INFORMATION 7-1
SYMBOLS 7-2
FLOATING-POINT ERRORS 7-2
ARITHMETIC FUNCTIONS 7-3
TRIGONOMETRIC FUNCTIONS 7-3
HYPERBOLIC FUNCTIONS 7-5
LOGARITHMS AND EXPONENTIALS 7-6
ROOTS AND POWERS 7-7
RANDOM NUMBER GENERATORS 7-8
ADDITIONAL FUNCTIONS 7-9
MEMORY REGISTER HANDLING 7-10
STACK MANIPULATION AND CONTROL 7-10
CONSTANTS 7-12
OPERATIONAL CONTROL 7-12
INPUT AND EDITING 7-13
PROGRAMMING KEYS 7-13

SECTION 8. ERROR MESSAGES

SECTION 9. HHC AND SCIENTIFIC CALCULATOR SYMBOLS

APPENDIX A. REFERENCES

APPENDIX B. USING PERIPHERALS WITH THE SCIENTIFIC CALCULATOR

APPENDIX C. USING THE SCIENTIFIC CALCULATOR AS A LIBRARY CAPSULE FOR OTHER PROGRAMS

APPENDIX D. HHC OVERLAY

SECTION 1. INTRODUCTION TO THE SCIENTIFIC CALCULATOR

The Scientific Calculator capsule satisfies a broad spectrum of needs — from hand calculations requiring extreme versatility and accuracy to the support, as a function library, for programs you may write using other HHC capsules.

SCIENTIFIC CALCULATOR FEATURES

The Scientific Calculator is a read-only-memory (ROM) capsule containing 4096 bytes of program for the HHC. It provides a set of abilities based on an arithmetic architecture that previously was available only in much larger and more costly computing systems.

- A complete floating-point arithmetic system for numbers from -10^{1024} to $+10^{1024}$. It includes the basic operations add, subtract, multiply, and divide as well as the mathematical functions listed below. A minimum of ten decimal digits of accuracy is maintained, with most calculations achieving twelve digits.
- A complete set of mathematical functions built upon this floating-point system that provides accurate approximations and precise error control: sine/cosine/tangent/cotangent, arcsine/arccosine/arctangent/arctangent (y/x), logarithm and exponentiation for three bases (2, e, 10), the hyperbolics sinh/cosh/ tanh/arctanh, and four versions of the power function (x^y, y^x, nth root, and square root) as well as $|x|$, x^2, and 1/x.
- Optional standard or scientific notation for input and output. Either kind of input is always allowed. However, for large numbers (i.e., $|x| > 10^5$), output is always in scientific notation, which allows entry, display, and calculation of numbers in the full range of -10^{1024} to $+10^{1024}$.
- User-selectable memory for saving intermediate results and frequently used constants.
- Complete error control, with error messages displayed on the HHC.
- The ability to function as a library for use by other HHC programs.

- Input/output ability that allows selection of printer or display units for recording calculations.
- Continuous memory that allows the retention of results for later calculations. Memory is lost, however, when you go out of the calculator to use another HHC application.
- Three HHC user-definable function keys that allow you to program sequences of up to 15 keystrokes on each that will execute when the programmed key is pressed.

This manual will guide both the casual user who needs the calculator occasionally and the user who needs to define the preciseness of results in some larger calculations. Full information is given in Sections 6 through 8 on the accuracy of calculations performed by this capsule.

REQUIREMENTS FOR OPERATION

The capsule requires 96 free bytes of HHC internal RAM. If the I/O menu shows less than this amount, you must delete a file or detach a peripheral before entering the calculator program.

HOW TO USE THIS MANUAL

Read Section 2 to become familiar with the keyboard and the stack concept. Section 3 gives you an explanation of how each key functions. Later you will want to examine Section 4, which leads you through examples of varying complexity. Section 5 explains the algorithms used in the mathematical and statistical areas; they are designed to provide accurate results over the large argument domains.

Finally, Sections 6 through 8 give complete specification on the boundaries and parameters associated with the Scientific Calculator's floating-point number system, argument domains and result ranges for each function, and error messages that are displayed when appropriate.

SECTION 2. GETTING STARTED

GETTING SET UP

To get started, turn the HHC OFF, then plug the Scientific Calculator capsule into the back of the HHC, but do not yet put the keyboard overlay in place. Press ON and then press the CLEAR key once or twice. The primary programs menu will be displayed; press the number of SCIENTIFIC CALCULATOR. Notice the momentary display confirming that you have selected the desired capsule. Now you can put the keyboard overlay in place and use the calculator.

BECOMING ACQUAINTED WITH THE STACK

The stack is basic to the Scientific Calculator. It is a section of memory that can contain up to ten standard or floating-point numbers. The first number (top of the stack) is displayed at all appropriate times. The STACK is under user control through the operation keys.

Reverse Polish Notation is used to enter arguments for calculations. This concise "programming" notation is used in many computers and calculators, particularly those that are stack-oriented.

As an example, use your calculator to follow the calculation of adding 4 to 6. A standard statement would be 6 + 4 = 10. However, using the Scientific Calculator and Reverse Polish Notation, you press

6 ENTER 4 ADD; 10 is displayed as the answer.

In this example, keys affect the stack as follows (assuming that only stack levels 1 and 2 have numbers in them).

Key in	*Stack Level*	*Value*	*Display*
6 ENTER	1:	6	6
4 ADD	1:	4	4 (the 6 is displayed until ADD is pressed, then the addition occurs yielding the next condition)
	2:	6	

1:	10	10
2:		(note that level 2 is now empty.)

Now press DROP to remove the one remaining stack entry and try a more complex example.

Calculate the square root of $1 - x^2$. Let x be 0.6. Note that PUSH and ENTER are the same key. The term PUSH is the more "stack-oriented", but we use both interchangeably.

Key in	*Stack*
1 PUSH	1: 1
.6 PUSH	1: .6 2: 1
SHIFT x^2	1: .36 2: 1
SUB	1: .64
$\sqrt{\ }$	1: .8

In the Scientific Calculator, all **one-operand** operations (keys such as x^2) use the first stack entry (the top) as the argument, and replace that entry with the result. Thus, to both save and use an argument, you should first store it in memory or duplicate it (DUP) on the stack so that it is on both level 1 and level 2. Then you can perform the operation and still retain the original number.

Two-operand operations like addition (ADD) combine the two numbers nearest the top of the stack, placing the result at level 1, eliminating level 2 and shifting levels 3,4,...n up one level each.

NOTE: Keying in a number does not put it on the stack automatically. For two-operand operations, if the first number is keyed in, it must be PUSHed onto the stack. If the first number is the result of a preceding operation or if it has been recalled from memory, it will be on the top of the stack already and need not be PUSHed.

There are also **no-operand** keys; for example, generation of a random number adds a result to the top of the stack. There are **control keys** that give precise control of the stack. All this, as well as what happens in case of an out-of-bounds result, is discussed in detail in Section 3 and in Section 7.

Sometimes you may want to view the current contents of the stack. If you are not in the middle of an operation, you may do so by pressing the DISP key. The first use of this key results in a display of the quantity of items on the stack.

```
HEIGHT=5
```

After this, each time you press any key except CLEAR or CANCEL, the next level of the stack will be displayed, starting with level 1.

```
1:  2.56E9
```

After the last item is shown, the calculator returns to keyboard input mode, displaying the value on the top of the stack. To leave the display mode while it is cycling , press CANCEL; this will return you to keyboard mode so that you can continue normal operations.

To try out this display feature, press DISP now. Then press CANCEL to return to keyboard mode and DROP the .8 put in level 1 at the end of the example above. Work your way through this example, using DISP to view the stack after each step. You should be able to verify the stack level and contents as they are shown. Remember to press CANCEL to return to keyboard mode.

INPUTTING NUMBERS

The number displayed on the LCD (liquid crystal display) is usually the number on the top of the stack. When you key in another number, it will become the new top entry as soon as you press PUSH. The number you enter may contain as many as seven components.

	Keys	*Example*
1. a sign	+/−	−
2. a string of integers	0,1,...9	12
3. a decimal point	.	.
4. a fractional part	0,1,...,9	34
5. an exponent symbol	ENT EXP	E
6. a sign for the exponent	+/−	−
7. an integer for the exponent	0,1,...,9	56

This example yields the number -12.34×10^{-56}; the calculator stores this internally as -1.234×10^{-55}. Examples of acceptable numeric entries are:

Integers such as:

0
1
650
-442
+17

Decimal values:

0.0
.00096
−.2
+643.33333333333
12.00

Values with exponents:

6E20	$(=6\times10^{20})$
4.12E−1	$(=4.12\times10^{-1}=$ .412)
−1.0009E−6	$(=-1.0009\times10^{-6})$
4E+16	$(=4\times10^{16})$

The overall sign and the exponent's sign may be omitted with positive or zero values. You should use the − key to enter a negative sign (do not use the SUB key). The + key can be used for correcting improper minus signs during input and for indicating sign on a printed output.

The magnitude of the mantissa (e.g., 12.34 above) need not contain all three components, but must include at least one numeric digit. For example, 12, 12., and 012.0 all yield the same result. Although thirteen significant digits of accuracy are retained, more digits may be entered.

The exponent (e.g., E−56 above) may be omitted for numbers that can be represented by no more than twenty-six characters without an exponent. If the ENT EXP key is used, then at least one digit must appear for the exponent's integer. This integer cannot contain more than four digits; within this constraint, leading zeros are allowed.

The number input is limited to twenty-six characters and cannot exceed the range of numbers the calculator can store.

Before the PUSH key is pressed, the number may be completely reedited. Use the left- and right-arrow keys to place the cursor for editing; there is no space bar. After a number has been entered with the PUSH key, it cannot be changed.

CONTROLLING THE OUTPUT FORMAT

The SCI and DIG keys control calculator output format (the input format is not affected). Their default values are fixed-point notation (SCI key toggled off) and display of 10 significant digits (DIG).

To have your results presented in scientific notation, press the SCI key once to turn it on; the DELETE blip shows on the display to indicate that SCI is active. Key in the following demonstration.

Key in	*Display*
123456 PUSH	123456
SCI DUP	1.23456E5

To return to standard notation, press SCI again to turn it off; the DELETE blip disappears:

Key in	*Display*
SCI DUP	123456

The DIG key controls the number of significant digits presented on the display. The default value for this function is 10 digits; however,the maximum value of 12 digits allows you to take full advantage of the Scientific Calculator's 13-digit arithmetic. Calculations use all the digits entered (up to 13), and results will be rounded to the number of significant digits you have chosen. Continuing with the example above:

Key in	*Display*
	123456
DIG	TYPE NUMBER (1-12):
3 ENTER	TYPE NUMBER (1-12): 3
DUP	123000
295 ADD	124000 (123456 + 295 = 123751, which is rounded to 3 significant digits.)

To verify that all digits entered have been used in the calculation and put onto the stack, reset DIG to a larger number and then DUP to see the top of the stack.

Key in	*Display*
DIG 7 ENTER	124000
DUP	123751 (= 123456 + 295)

The DIG setting will remain until changed deliberately or until the CLEAR key is pressed, which causes the setting to revert to default. Turning the calculator off does not affect the setting.

USING MEMORY

Every number used in a calculation is put on the stack automatically, and the stack must emptied when it becomes full. However, numbers must be *specifically* stored in one of the 12 memory locations and they stay there until *specifically* cleared with either CLM, which clears all memory locations, or CLS, which clears a specified memory storage location.

Contents of memory can be recalled at any time either by you or by a program that you have entered in the f1, f2, and f3, user-definable function keys.

Data stored in memory is not affected by turning off the HHC or by pressing CLEAR once. However, pressing CLEAR twice to access the HHC primary menu and other HHC applications will destroy the contents of all 12 memory locations.

To familiarize yourself with the storage function, run through the following example. Either key in a new number to store or use the number on top of the stack that is displayed.

Key in	*Display*
4	4
π	3.141592654
MUL	12.56637061
SHIFT STO	TYPE NUMBER (1-12):
6	TYPE NUMBER (1-12):6
ENTER	12.56637061

To verify entry, recall the number from storage:

Key in	*Display*
REC	TYPE NUMBER (1-12)
6 ENTER	12.56637061

ERRORS AND WARNINGS

Attempting calculations with invalid parameters will result in error messages. These can occur in the following two situations.

Illegal operation — if you violate the constraints of the floating-point system in your calculations or if you use mathematical functions with incorrect arguments. For example, raising 2 to the power 10000 is prohibited, because 10000 is too large an argument, and the result exceeds the maximum machine-representable number (Section 6). Similarly, division by zero is not tolerated.

Accuracy Bound — if an argument presented to a routine results in a function value considered to be too inaccurate. For example, sine routine arguments should be less than 102942 radians in absolute value. This is because n times π is used to reduce the argument; if the argument is large, it is nearly n times π (being identical to it to a number of significant digits). Then, in subtraction, these identical digits are lost. The sine approximation is tailored for arguments in the (0,π/2) range, so that the reduced argument will be less significant than was the original argument by too large an amount; the result cannot be as accurate as desired, since the argument was slightly altered during reduction. The bounds on the argument domains presented in Section 7 were chosen so that the floating-point arithmetic system could be fully exploited by the algorithms used. Users are protected from results which may be "less than accurate" because of arguments that are too large to be accurately reduced.

In both overflow and accuracy bound,

- an "ERROR n" message is displayed. Tables 7 and 8 discuss the meanings for the various values of n. Use the CANCEL key to remove the message from the LCD;
- the same number of inputs is dropped from the stack as would be appropriate if the operation succeeded;
- the resultant top stack entry (an invalid-floating point number) is automatically dropped; and
- other stack entries and user memory are unchanged.

There are other situations in which warning messages are given. For example, an attempt to execute an instruction that would underflow the stack causes a beep; neither the display nor the stack is changed. When the maximum stack height of 10 is reached, the HEIGHT WARNING message is dis-

played. No more stack entries will be accepted. You can use the DISP key to display stack entries and can diminish the stack by using a combination of the ROLL and DROP keys. You can also press CLEAR once; however, this will cause the DIG and SCI keys to revert to their default values.

SECTION 3. USING THE OPERATION KEYS

This section discusses the operation of each key you may use while operating the Scientific Calculator. Simple examples are given that include key strokes, stack contents, and the HHC display for each step where this information will be helpful. The HHC will BEEP when you enter an illegal keystroke.

The discussion is organized by category: control operations, elementary operations, mathematical functions, statistical functions, and programmable keys. The more complex functions are explained with problem-solving examples in Section IV.

> NOTE: When two functions are assigned to one key, the upper function printed on the keyboard overlay is accessed by pressing and releasing the SHIFT key before using the function key. If you change your mind after pressing SHIFT, press SHIFT again to cancel it.
>
> CAUTION: All the HHC keys are still functional when you are using the Scientific Calculator. Therefore, you must be careful not to press the HHC 2nd SFT key. If it is used with an HHC alphabetic key that has a second shift value, the calculator will BEEP; if used with an alphabetic key without a second shift value, the calculator will treat the keystroke as CANCEL.

CONTROL OPERATIONS

Memory Control

Memory contents are not affected by turning the HHC off or by pressing the CLEAR key once. However, pressing CLEAR twice to return to the HHC primary menu will destroy the contents of locations 1 through 12.

- STO (Store Memory). This key allows you to store any valid HHC number (one you enter for this purpose or one already on the top of the stack) in a storage location 1 through 12. The location number is entered after the prompt asking for one appears on the display. Location 12 is used to store the seed for X_u and X_n in statistical calculations; it is the only location that can have its contents altered by the capsule without an explicit request from the user.

Key in	*Display*
1234	1234
SHIFT	
STO	TYPE NUMBER (1-12):
9	TYPE NUMBER (1-12):9
ENTER	1234

The number entered into memory is also put on the top of the stack.

- REC (Recall Memory). Use this key to recall to the display the contents of any storage location 1 through 12; the number recalled is also put on the top of the stack.

Key in	*Display*
REC	TYPE NUMBER (1-12):
9	TYPE NUMBER (1-12):9
ENTER	1234

When a number has been recalled and is shown on the display, it can be used in 1- and 2-operand calculations because it has been put on the top of the stack automatically, pushing the preceding entry down to level 2.

- CLM (Clear Memory to zero). Pressing this key clears all memory storage locations to zero.

- CLS (Clear Storage Location n). The prompt asks for the number of the memory storage location to clear.

- M−, M+ (Memory subtract and add). These two keys operate on the contents of a specified memory location 1 through 12. The number on the top of the stack is used; this can be the result of the preceding operation or a number that you have just keyed in.

Key in	*Display*
(assume that the contents of 9 are 68)	
24	24
M−	TYPE NUMBER (1-12):
9	TYPE NUMBER (1-12):9
ENTER	24

To verify the subtraction, you can recall from memory:

Key in	*Display*
REC	TYPE NUMBER (1-12):
9	TYPE NUMBER (1-12):9
ENTER	44 (= 68 - 24) The number keyed in is subtracted from the contents of the memory location specified.

Stack Control

Every number that you key in must be pushed onto the top of the stack by you or by an operation key. A HEIGHT WARNING message will be displayed when the stack is full; no new numbers will be accepted. If you receive this warning message, you can store your last entry in a memory storage location, then use DROP or a combination of ROLL and DROP to remove numbers from the stack to continue operation. You can also press CLEAR once.

> CAUTION: If you press the CLEAR key for any reason, stack contents are destroyed (and the SCI and DIG keys are turned off, returning to their default values).

- PUSH. The ENTER,PUSH,DUP key is used to put your keyboard entry on the top of the stack. Used several times in succession, it will put the same entry on several levels.

- DROP,C/E. This key is used to delete the entry on the top of stack and display the next level as the new top level. When the stack is empty, the cursor is displayed. You must be in keyboard mode to do this; the key will not operate while you are cycling through the stack display (press CANCEL to abort the DISP cycle).

Key in	*Stack*
	1: 8.3E2
	2: 4.683E−9
	3: 3.1415
	4: 6
DROP	
	1: 4.683E−9
	2: 3.1415
	3: 6

The DROP,C/E key is also used to CLEAR ENTRY from the

display ***before*** it has been PUSHed onto the stack. In this use, the entry on top of the stack is not dropped.

- SWAP. Use this key to exchange the top two entries on the stack. Neither number is destroyed.

Key in	*Stack*	*Display*
	1: 789 2: 476	789
SWAP	1: 476 2: 789	476

- ROLL. This key will remove the number from the specified stack level and push it onto the top of the stack. The prompt requests the number of the stack level containing the number you want moved.

Key in	*Stack*	*Display*
	1: 9 2: 8 3: 7	9
ROLL		TYPE NUMBER (1-3):
3		TYPE NUMBER (1-3):3
ENTER	1: 7 2: 9 3: 8	7

Note that the number 7 is no longer in its original relative position in the stack.

- PICK. This key operates like ROLL except that the number at the specified stack level is left in its original place as well as being placed on the top of the stack. Press PICK. After the prompt, key in the level number and press ENTER. The duplicate is pushed onto the top of the stack; the original retains its initial relative position in the stack sequence, but is one level lower.

- DISP. Use this key to see how many entries there are on the stack and what their values are. The first time you press the key, the display will tell you the height of the stack — for example

```
HEIGHT = 4
```

If you press again, the first level of the stack will be displayed.

```
1 : 3.56E4
```

Press again to see level 2, and so on. When all levels have been displayed, the calculator returns to keyboard input mode, displaying the value on the top of the stack. To stop the cycle at any point and return to keyboard input mode, press CANCEL.

DISP will not show a number that has not been PUSHed.

OPERATIONAL CONTROL

- ENTER. The ENTER,PUSH,DUP key is used to complete the entering of data requested by a prompt on the display, such as for a number of digits or memory storage number.

- CANCEL. Use this key to cancel operations that ask for a response (such as REC, STO, and DIG) if you decide not to activate the operation, to return to input mode when the DISP key is active, and to cancel an error message so that you can continue operation.

- DIG. This key is used to control the number of ***significant*** digits in the numbers displayed. The key is effective until pressed again and the setting is changed. Pressing CLEAR for any reason, however, will release the setting and will return to the default value of ten digits. Although the number is displayed rounded to the specified number of digits, it is in memory as entered and can be recalled in its original form by respecifying a larger number of digits.

Key in	*Stack*	*Display*
DIG		TYPE NUMBER (1-12):
2		TYPE NUMBER (1-12):2
ENTER		
123		123
PUSH	1:123	123
456		456
ADD	1:579	580 (note the effect of rounding in the display)

Up to here all calculations have been displayed rounded to 2 significant digits. Now, if you specify more digits you can obtain the result of the calculation unrounded unless your input was more than 12 digits.

Key in	*Stack*	*Display*
DIG		TYPE NUMBER (1-12):
4		TYPE NUMBER (1-12):4
ENTER		
DISP	1:579	579 (= 123 + 456)

Note that although a large number of digits may be input and thirteen are retained, twelve digits of accuracy at most will be displayed.

- SCI. This key operates as an on-off switch; the ON mode is disabled when CLEAR is pressed; default value is standard, fixed- point notation. The SCI key ON will cause all numbers to be ***displayed*** in scientific notation, the number of digits being controlled by the DIG key (default is ten digits). The DELETE blip is displayed when the SCI key is active.

- DEG/RAD. In the Scientific Calculator, trigonometric functions can be used with angles specified in either decimal degrees or radians. The DEG/RAD key is a toggle switch that ***must be set to match the mode of your inputs***; its default value is radian mode. Press the key to cause the mode (but not the data currently on the display) to change to degree mode. Press the key again to revert to radian mode. The current mode is indicated by the INSERT blip — blip showing means degrees, blip not showing means radians.

The DEG/RAD key does not convert previous calculations. You can use the following conversions to do this. If it is a frequent calculation, program one of the f1,f2,f3 keys to do it for you.

n degrees on stack: enter

π 180 DIV MULT

(n degrees × π/180 = radians equivalent)

n radians on stack: enter

180 π DIV MULT

(n radians × 180/π = degrees equivalent)

Angles in degrees, minutes, seconds must be converted to decimal format before entry. Section 4 shows you how to program the calculator to do this.

Inputting Numbers

- ±. Use this key to enter a sign for a number or an exponent. This is not a function key and will not add or subtract.

- 0,1,...,9. The numeral keys are used to enter the numbers for your calculations or in response to prompts from the display. Press the ENTER, PUSH,DUP key to complete the entry. To cancel an entry rather than entering it, press DROP,C/E to clear argument entries and press CANCEL to clear the response to a prompt for storage location, number of digits, etc.

- .(Decimal). This key is used to enter the decimal point in numbers.

- ENT EXP (Enter Exponent). Use this key to enter an exponent with your number. For example, 123E−5 PUSH stores 1.23×10^{-3} (.00123 in standard notation) on the top of the stack; pressing the ENT EXP key after the 3 key causes "E" to be displayed.

- ←, →. Use these arrow keys to space the cursor forward and backward to change a number before you have PUSHed or ENTERed it. Note, however, that you cannot correct an error in the f1, f2, f3 keys this way; you must start over.

ELEMENTARY OPERATIONS

- π and γ. Keying in either of these constants automatically pushes your previous entry to stack level two. Use the procedure below to find the circumference of a circle with a radius of 2.6 feet (or meters).

(π = 3.14159265359)
(γ = 0.5772156649)

Key in	*Display*
2	2
π	3.14159265359
MULT	6.28318530718
2.6	2.6
MULT	16.33628 ft (m)

- ADD, SUB, MULT, DIV. These functions require two arguments on the stack. Pressing the operation key automatically pushes the second argument onto the stack if it has just been keyed in; then it performs the operation, deleting both arguments and putting the result on the top of the stack.

Key in	*Stack*	*Display*
360 PUSH	1:360	360
2 MULT	1:720	720

Any numbers within the HHC range can be used for these two-number operations. However, because the HHC truncates at the thirteenth digit, adding or subtracting numbers whose exponents differ by more than 12 will result in display of an answer identical, except possibly in sign, to the input number that is the larger in magnitude. As an example, consider trying to add 1.E13 and 2.

```
10000000000000
          +  2
--------------

10000000000002
              ↑
              └── Truncation at the thirteenth digit.
```

If an out-of-bounds result is produced, an error message is given, and the result is automatically dropped. As an example, enter

9E1023 PUSH DUP MULT

The display will show the message ERROR 2 (floating-point overflow). Press CANCEL to cancel the error message, then press DISP. Note that none of the above calculation is present on the stack.

Any number less than 10^{-1024} is considered to be 0 and could result in a zero-divide error.

- 1/X, X^2. These two functions operate on the number on the top of the stack or a number just keyed in. 1/X will give the reciprocal of any number in the HHC maximum range; X^2 will give the square of any number no larger than the square root of the plus-or-minus maximum HHC number (see Section 6).

Key in	*Display*	*Key in*	*Display*
25	25	25	25
SHIFT		1/x	.04
X^2	625		

- FRAC. Used to obtain the fractional part of the number on the top of the stack and to drop the integer. Answers will be in the range >-1 to $<+1$. The fraction is placed on the top of the stack with the original number being destroyed.

Key in	*Display*	*Key in*	*Display*
123.456	123.456	−123.456	-123.456
SHIFT		SHIFT	
FRAC	4.56E-1	FRAC	-4.56E-1
	or .456		or -.456

- FIX. The opposite of FRAC; the fractional part of the number is dropped and the integer is placed on top of the stack and on the display.

Key in	*Display*
−123.456	-123.456
FIX	-123

NOTE: Rounding can seriously affect FIX and FRAC, as shown by the following example. Assume that DIG has been set to 3.

Key in	*Stack*	*Display*
	9.999	10
FIX	9	9

Key in	*Stack*	*Display*
	9.999	10
FRAC	.999	.999

or worse,

	9.9995	10
FRAC	.9995	1

- |X| [ABS(X)]. This key will change the number on the top of the stack to its absolute value.

Key in	*Display*
−12.4	-12.4
SHIFT	
|X|	12.4

- CHS. This key changes the sign of the number being displayed (mantissa only, exponent is not affected). If the number being changed is already on the top of the stack, it will be replaced with the changed value. If the number has not yet been PUSHed onto the stack, pressing CHS will

cause stack contents to be pushed down one level, and the changed number to be put on top.

Key in	*Stack*	*Display*
		25E-9
CHS	−2.5E8	-2.5E8
CHS	2.5E8	2.5E8

MATHEMATICAL FUNCTIONS

Note that for functions using two stacked arguments, x denotes the item on top of the stack at level 1 and y denotes the item at stack level 2.

Trigometric, Inverse Trigometric, Hyperbolic, Inverse Hyperbolic

The Scientific Calculator is initialized to expect angles to be entered in radians. If you are using degrees, press the DEG/RAD key to set degree mode. The INSERT blip will show when degrees mode is active.

- SIN, COS, TAN, COT; ASIN, ACOS, ATAN, ACOT; SINH, COSH, TANH, ATANH, and ATAN2 automatically control over- and underflow. Examples are given in Section 4; detailed parameter information is contained in Sections 6 and 7.

Except for the ATAN2 function, all of these functions operate on the value on the top of the stack. ATAN2, the two-argument arctangent computes [$\tan^{-1}$ (y/x)], x being the value at stack level 1 and y the value at stack level 2.

- Find the angle whose tangent is 36.75:

Key in	*Display*
36.75	36.75
SHIFT ATAN	1.54359 radians

- Two-argument arctangent: x = 14.57; y = 36.4

Key in	*Display*
36.4 PUSH	36.4
14.57	
ATAN2	1.19005

- Cosine of 36.75 degrees:

Key in	*Display*
DEG/RAD	(INSERT blip goes on for degree mode)
36.75	36.75
COS	.801254

Power, Logarithm, and Root Functions

Most of these keys operate on the value at the top of the stack. However, several require two arguments. For the one-operand functions, just key in a value, then press the function key. The result of the calculation will be displayed.

- x^y, y^x. These keys allow you to raise any number to any power, keeping within the Scientific Calculator range. Y is the value at stack level 2 and X is the value at stack level 1.

Key in	*Stack*	*Key in*	*Stack*
3 PUSH	1: 3	3 PUSH	1: 3
2	1: 2 2: 3	2	1: 2 2: 3
SHIFT x^y	1: 8 (2^3)	y^x	1: 9 (3^2)

- 2^x, e^x, 10^x. These keys allow you to raise these three constants to any power, keeping within the Scientific Calculator range. x is the value on the top of the stack. The value of e is 2.71828182846. Key in the value of x, then press one of the power keys. The result appears on the display and is automatically put onto the top of the stack.

- Log_2, Log_e, Log_{10}. These keys compute the log of x, the value on the top of the stack, to base 2, base e (2.71828182846), and base 10, respectively. x, the value on the top of the stack, is replaced by the result of the operation.

- $\sqrt{\ }$. Gives the square root of the value on the top of the stack.

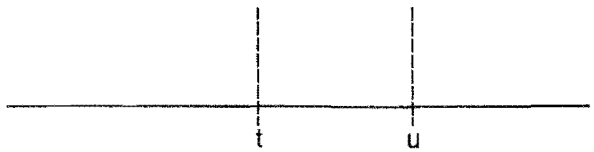
- $\sqrt[n]{\ }$. This key will give you the nth root of the value on top of the stack. When you press this key (SHIFT first), the display will ask you for the value of n — an integer from 1 to 50. Key in the number and press ENTER.

Key in	*Stack*	*Display*
10.3		10.3
SHIFT $\sqrt[y]{\ }$		TYPE NUMBER (1-50):
3 ENTER	1: 2.176	2.176 (the cube root of 10.3)

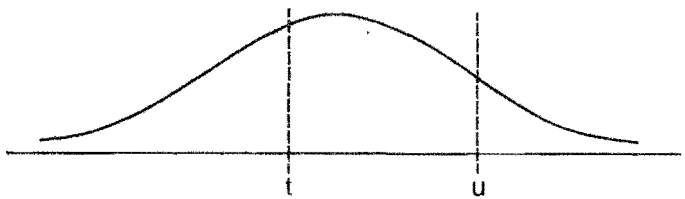
STATISTICAL FUNCTIONS

• $\overline{X}_n$, $\overline{X}_u$. These keys are used to generate pseudo-random normal or uniform deviates, respectively. Storage location 12 is used to store the seed for calculations. If no seed is placed there, a specific seed will be chosen by the generator. For a given seed, the pseudo-random numbers are always generated in the same sequence; therefore, if you restart with the same seed, you can duplicate your calculations. The use of these keys is demonstrated in Section 4 in An Investment Problem; parameters are given in Section 7. A detailed explanation of the algorithms used is given in Section 5.

The uniform deviate, $\overline{X}_u$, has a uniform built-in distribution preference and can generally be represented by a straight line (the range of results is from zero to one). You are as likely to get .001 as a result as .799; this is analogous to picking numbers out of a hat, replacing each number after picking it. More correctly, the deviate is distributed uniformly on the (0,1) line.



The normal deviate, $\overline{X}_n$, has a built-in distribution that can be represented by a normal curve (the "bell" curve).



In a normal distribution your result is more likely to be t than u (because the curve is higher at t), whereas in a uniform distribution t and u are equally probable.

Normal distributions occur very often in real-life situations:

IQ—most people have an IQ between 85 and 115; far fewer have an IQ below 60 or above 140.

dice—throwing two dice, you are more likely to throw 7 than 2 or 12.

PROGRAMMABLE KEYS

These are the three user-definable keys (f1,f2,f3) described in the HHC user manual, INSTRUCTIONS FOR USE. You can define each key to represent a sequence of up to 15 keystrokes that will be executed when you press the defined key. For instance, you can program them to function as a constant recurring value or to execute a series of calculations.

To enter a program, press HELP, then the desired program key (e.g., f1). Enter the required keystrokes, then finish the programming by pressing the same f key again. (When 15 keystrokes are entered, the program terminates automatically.) Mistakes can not be edited; you must cancel the program by pressing the f key and then starting the procedure again from the beginning.

The use of these keys is demonstrated extensively in Section 4. Section 9 shows the correlation between the Scientific Calculator keyboard and the HHC characters that appear on the display when you are programming one of these keys.

SECTION 4. PROBLEM-SOLVING EXAMPLES

Now that we have seen the operation of the individual keys, we can begin combining the various elements available in the Scientific Calculator to solve problems.

TRIGONOMETRIC FUNCTIONS

The following trigonometric examples assume that the stack is initially empty and that the SCI (standard, fixed-point notation), DIG (10 digits), and DEG/RAD (radians) keys are at their default values.

- Calculate the sine of 0.5 radians.

1. Select the desired number of output digits, say 9:

Key in	*Stack*	*Display*
DIG		TYPE NUMBER (1-12):
9 ENTER		TYPE NUMBER (1-12):9

2. Enter 0.5 and execute SIN

0.5	1: .5	0.5 (in this case, PUSH is optional)
SIN	1:.479425539	.479425539

If you want to enter degree arguments, press the DEG/RAD key before you start entering them. Thus, the calculator assumes that arguments for the trigonometric functions are being given in degrees, and results of inverse trigonometric functions will be presented in degrees. Pressing the key again will cause the calculator to revert to radians mode, but will **not** convert data already entered.

- Calculate the arctangent of 1.0/−1.0. This operation places the result in the proper quadrant.

Key in	*Stack*	*Display*
1 PUSH	1: 1	1
DUP	1: 1 2: 1	1 1

CHS	1: −1 2: 1	-1 1
ATAN2	1: 2.35619449	2.35619449

LOGARITHMIC FUNCTIONS

• Calculate the logarithm with base 5 of 25.

Several formulas can be used — note that the Scientific Calculator allows logarithm calculations to bases 2, e, and 10. We select

$\log_5(25) = \log_e(25)/\log_e(5)$

Key in	*Stack*	*Display*
25	1: 25	25
LOG_e	1: ln(25)	3.218875825
5	1: 5 2: ln(25)	5
LOG_e	1: ln(5) 2: ln(25)	1.609437912
DIV	1: 2	2 [$=\log_5(25)$]

For retention of that result for later calculations, you can call on user storage (STO), or can use the higher levels of the stack and then use PICK or ROLL. Say that subsequent calculation results in the stack arrangement below,

Level 1:	result
Level 2:	intermediate result
Level 3:	intermediate result
Level 4:	$\log_5(25)$

and we wished to multiply the result by $\log_5(25)$. PICK 4 ENTER would result in a stack of

Level 1:	$\log_5(25)$
Level 2:	result
Level 3:	intermediate result
Level 4:	intermediate result
Level 5:	$\log_5(25)$

(ROLL 4 would have deleted the last stack level.) Now when you press MULT, stack levels 1 and 2 are multiplied, and the product of [result] × [$\log_5(25)$] is placed on top of the stack. Other values move up one level.

Level 1:	[result) ($\log_5(25)$]
Level 2:	intermediate result
Level 3:	intermediate result
Level 4:	$\log_5(25)$

ENGINEERING FUNCTIONS

When programming the user-defineable keys, you must pay attention to the stack during each step to be sure that the program will use the proper value in its calculations. Always press HELP, then begin and end the program by pressing the f key you are using. When you are calculating, the program will execute each time you press that f key again.

CAUTION: When using an f key during calculations, press the PUSH key before keying in f1, f2, or f3 if the programmed sequence starts with a number. This is necessary to put the new number just keyed in onto the top of the stack. For instance, if the program sequence in f1 starts with 3 and you key in "12 f1", the 3 will be appended to the 12, resulting in 123 and causing an erroneous answer. "12 PUSH f1" will give the correct answer. If you don't remember exactly how your programmed sequence starts, press PUSH before the f key just to be sure of correct operation.

Also remember to check stack levels. If a function key's program uses three stack levels, it won't work if you start with eight levels on the stack already.

FIX and FRAC

• Calculate x − [x] (find the value of a number less the largest integer in that number). Let x = 5.2345.

Key in	*Stack*	*Display*
5.2345 PUSH	1: 5.2345	5.2345
DUP	1: 5.2345 2: 5.2345	5.2345
FIX	1: 5 2: 5.2345	5

SUB	1: .2345	.2345

Or, better,

5.2345	1: 5.2345	5.2345
FRAC	1: .2345	.2345

NOTE: See DIG in Section 3 for an explanation of rounding that takes place in the Scientific Calculator.

Modulo (y mod x)

Enter the y value first, then the x value. y is divided by x and the remainder is multiplied by x and put on top of the stack, replacing x and y.

Key in	*Stack*	*Display*
(y) 275 PUSH	1: 275	275 (y value)
(x) 3 DIV	1: 91.66666667	91.66666667 (x value)
SHIFT FRAC	1: .6666666667	.6666666667
(x) 3 MULT	1: 2	2 (reenter the x value=3)

You can program this routine into an f key as follows.

HELP f3	
SWAP	Brings y value to top of stack.
PICK 2 ENTER	Duplicates x on top of stack so that x at level 1 can be used to divide y at level 2, and x at level 3 will move up with each calculation and be available for multiplication at the end of the calculation.
DIV	Divides y by x.
SHIFT FRAC	Drops integer and displays remainder.
MULT f3	Multiplies the remainder by the x value.

To use the program, key in:

y value PUSH x value f3

Degrees/Radians

- Convert degrees to radians: n degrees x (pi/180) = radians

  Program: HELP f1 π 180 DIV MULT f1

- Convert radians to degrees: n radians(180/pi) = degrees

  Program: HELP f2 180 π DIV MULT f2

Example: Convert 20 degrees to radians equivalent and then convert back to degrees.

Key in	*Display*
20 PUSH f1	.3490658504 (radians equivalent)
f2	20 (converted back to degrees)

- Convert degrees or hours, minutes, seconds to decimal format. In this example, f1 and f2 are programmed to calculate the decimal equivalent:

  1 minute = .01666667 degree
  1 second = .00027778 degree

  Program:

  HELP f1 .01666667 MULT ADD f1

  (converts minutes and adds to degrees already entered)

  HELP f2 .00027778 MULT ADD f2

  (converts seconds and adds to degrees and minutes)

Example: Convert 3 16'25" to decimal format.

Key in	*Stack*	*Display*
3 PUSH	1: 3	3
16 PUSH f1	1: 3.2666656	3.2666656 (deg+min)
25 PUSH f2	1: 3.2736081	3.2736081 (deg+min+sec)

NOTE: If there are no degrees, key in "0 PUSH" before using f1 or f2.

Percentages and Conversion Factors

- Compute a sales tax of 6% and add to total purchase.

Program:

HELP f1 DUP	Amount of purchase must be duplicated
.06 MULT	Calculates 6% sales tax
ADD f1	and adds to subtotal for grand total

The value you key in must be duplicated on the stack by the program so that it will be available for both MULT and ADD.

Example: Calculate total bill for purchases of $25.96 and 6% sales tax.

Key in	*Display*
25.96	25.96
PUSH f1	27.5176

- Convert from liters to quarts. The conversion factor can be stored in a memory location from 1 to 12, but, if it is a frequently used conversion, it is more convenient to program it into an f key along with the multiplication.

Example: Convert 25 liters to the quarts equivalent. Conversion factor is 0.946.

Program: HELP f3 .946 MULT f3

Key in	*Display*	
25 PUSH f3	23.65	(equivalent of 25 liters)

Factorials

- Find how many ways seven symbols can be arranged on a straight line.

Program:

HELP f1	
REC 1 ENTER	Recalls the total number of factors stored here when starting calculation.
1 SUB	Decrease number of factors by 1.
SHIFT STO 1	Store new total for recall.
ENTER	
MULT f1	Multiplies stack levels 1 and 2.

Key In to Use	*Stack*	
7 SHIFT STO		
1 ENTER	1: 7	(Store total number of factors)
f1	1: 42	(7 x 6)
f1	1: 210	(42 x 5)
f1	1: 840	(210 x 4)
f1	1: 2520	(840 x 3)
f1	1: 5040	(2520 x 2)
f1	1: 5040	(5040 x 1)

You can press REC 1 ENTER to see which factor you have just multiplied by. If you do this during calculation, be sure to press DROP after recalling the factor; if you don't, the next multiplication will be wrong because the wrong number will be on the top of the stack.

STATISTICAL FUNCTIONS

Summations

In the following example, pairs of numbers are entered. The f1 and f2 keys are programmed to accumulate in memory locations 1 through 6 the values x, x^2, y, y^2, xy, and the number of paired entries. Several examples follow that are based on these values.

Location	*Value*
1	Σ x
2	Σ x^2
3	Σ y
4	Σ y^2
5	Σ xy
6	number of pairs calculated

Program the keys:

HELP f1	
M+ 1 ENTER	
DUP	Adds x to total in loc. 1.
SHIFT X^2 M+ 2	Adds x^2 to total in loc. 2.
ENTER DROP	Brings y to top of stack.
SWAP M+ 3	
ENTER DUP	Adds y to total in loc. 3.
f1	To terminate.
HELP f2	
SHIFT x^2	Computes y^2.
M+ 4 ENTER	
DROP	Adds y^2 to total in loc. 4.
MULT M+ 5	Multiplies x and y, which are in
ENTER DROP	levels 1 and 2, and adds the product to the total in 5.
1 M+ 6 ENTER	Adds 1 to the number of pairs in 6.
DROP	Clears stack so it won't fill up during your calculation.
(f2)	Because there have been 15 keystrokes, the program exits automatically.

• Example 1: Calculate the above values for four pairs of numbers. Remember that the first number entered for each pair becomes the y value because it ends up on stack level 2.

y:	19	24	20	17	ENTER Y FIRST, THEN X
x:	10	16	15	14	

Key in:

SHIFT CLM	Zero locations 1 through 12
19 PUSH 10 f1 f2	Do not press f2 until f1
24 PUSH 16 f1 f2	has stopped calculating.
20 PUSH 15 f1 f2	
17 PUSH 14 f1 f2	

Check the registers:

REC 1 ENTER 55	Σ x
REC 2 ENTER 777	Σ x^2
REC 3 ENTER 80	Σ y
REC 4 ENTER 1626	Σ y^2
REC 5 ENTER 1112	Σ xy
REC 6 ENTER 4	number of pairs

Arithmetic Mean

• Example 2: Calculate the arithmetic mean, $\bar{x}$: (Σ x)/n and (Σ y)/n based on the input in Example 1 above.

Key in	*Display*	
REC 1 ENTER	55	Fetches Σ x
REC 6 ENTER	4	Fetches number of entries
DIV	13.75	Arithmetic mean of x values

Use REC 3 to compute mean for y values.

Harmonic Mean

• Example 3: Calculate the harmonic mean (the reciprocal of the arithmetic mean of the reciprocals of a finite set of numbers). You can program f3 to calculate Σ 1/x and Σ 1/y while you are executing f1 and f2.

Program:

HELP f3	Enter y value first, then x.
PICK 2 ENTER	Brings y to top of stack.
1/x M+ 8	
ENTER	Puts reciprocal of y value in loc. 8.
DROP DUP	Brings x value to top of stack and duplicates it for use by f1.
1/x M+ 7	
ENTER	Puts reciprocal of x value in loc. 7.
DROP f3	x and y are now in their original positions on the stack for the next calculations.

Key in to Execute:

```
19  PUSH  10  f3  f1  f2
24  PUSH  16  f3  f1  f2
20  PUSH  15  f3  f1  f2
17  PUSH  14  f3  f1  f2
```

Locations 1 through 6 will contain the same values as those in the previous example. Check the new calculations.

Key in	*Display*	
REC 7 ENTER	.3005952381	$\Sigma(1/x)$
REC 8 ENTER	.203121775	$\Sigma(1/y)$

Now you can calculate the harmonic mean using the values in locations 7 and 8.

Key in	*Display*	
REC 8 ENTER	.203121775	$\Sigma(1/y)$
REC 6 ENTER	4	number of entries
DIV	.0507804437	arithmetic mean of $\Sigma(1/y)$
1/x	19.69262035	harmonic mean of Σ y

Using the same procedure but RECalling 7, you can get the harmonic mean of Σ x, which is 13.30693069.

Standard Deviation

• Example 4: Using the data in memory from running Example 1 or 3, find the standard deviation (sigma—σ) for the x and y values. These calculations are based on the formula

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

Note that some calculations use (n) in the divisor; however, we use (n−1). To work through this example, be sure that the registers have the following data (produced in the SUMMATIONS section).

Key in	*Display*	
REC 1 ENTER	55	Σx_i
REC 2 ENTER	777	Σx_i^2
REC 6 ENTER	4	n

Calculate sigma as follows.

REC 2 ENTER	777	Σx_i^2
REC 6 ENTER		
MULT	3108	$n(\Sigma x_i^2)$
REC 1 ENTER	55	Σx_i
SHIFT x^2	3025	$(\Sigma x_i)^2$
SUB	83	$[n(\Sigma x_i^2)] - [(\Sigma x_i)^2]$
REC 6 ENTER		
DUP	4	DUPed to use twice
1 SUB MULT	12	$n(n-1)$
DIV	6.916666667	
$\sqrt{}$	2.62995564	

Means and Sums of Squares

This example demonstrates another approach to these calculations. Full retention of accuracy in computing means and the sums of squares requires the use of techniques which are somewhat inscrutable. The main problem is that accuracy is lost when the range of numbers is large unless adjustment of the entered values by their means (or approximations to their means) takes place at the correct time. In most cases, the mean of a set of numbers can be calculated by the formula

$$\overline{x} = \left(\sum_{i=1}^{n} x_i\right)/n,$$

If memory is available for total array storage, then the most accurate method is to sort the numbers into numerical order, and then to sum in longer precision than that of the individual numbers. If the range of the numbers is large, and small memory is available, then the mean is best calculated by the recurrence relationship

$$\overline{x}_{k+1} = \overline{x}_k + (x_{k+1} - \overline{x}_k)/(k+1)$$

where

x_{k+1} is the (k+1)st entry and $\overline{x}_1 = x_1$ is the first entry.

Similarly (although we will not exemplify with a programmed example), the recurrence relationship for an accurate sum of squares may be of value.

$$s = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

should use

$$s_{k+1} = s_k + k(k+1)\,[(x_{k+1} - \bar{x}_k)/(k+1)]^2,$$

where $s_1 = 0$.

(Note that if the total array of x's could be held in memory, then the actual mean could be found directly and could be used for adjustment in calculating the sum of squares.)

In calculating the mean, we will

1. Set k to 0 in user memory location 6; set memory location 7 to zero, for use in retaining the approximations to the mean.

2. Enter x_{k+1}

3. Calculate $\bar{x}_{k+1}$ and increase k to k+1 and iterate steps 2 and 3 as many times as required.

1. To start by setting k to zero in register 6 and the result to zero in register 7, type

```
CLS 6  ENTER
CLS 7  ENTER
```

2. The original step 2 (entering x_{k+1}) cannot be programmed but the function keys can be set up to compute the new mean as each new value is entered. Because more than fifteen keystrokes are required, the task will be split between the f1 and f2 keys.

2a. The f1 key will calculate the factor $(x_{k+1} - \bar{x}_k)/(k+1)$ and will increment k to k+1 in register 6. It is assumed that the user has typed in x_{k+1} so that it appears on the display (i.e., is on top of the stack) and that k is in register 6 and $\bar{x}_k$ is in register 7. To set up the functions, type

```
HELP f1            start program
REC 7 ENTER        get x̄k
SUB                get x(k+1) − x̄k
REC 6 ENTER
  1 ADD            get k and add 1
SHIFT STO 6
  ENTER            save k+1
                     Note: k+1 still on stack
DIV                get (x(k+1) − x̄k)/k+1
f1                 to terminate
```


2b. Now all that remains is to add the original mean $\bar{x}_k$ to the modification $(x_{k+1} - \bar{x}_k)/(k+1)$. It will be saved in register 7. Type

```
HELP  f2           start
REC 7  ENTER       get x̄k
ADD                add modification
SHIFT STO 7
  ENTER            save it
DROP               remove it from the stack to
                   make room for x(k+2)
f2                 to terminate
```

Example: compute the mean of 4, 8, and 7.

```
CLS 6  ENTER       set k = 0
CLS 7  ENTER       set x̄0 = 0
4  f1  f2          do x1 = 4
8  f1  f2          do x2 = 8
7  f1  f2  ENTER   do x3 = 7
REC 7  ENTER       to display results: 6.333333333
```

3. A considerable portion of the previous sample is devoted to the use of register 7. If you just want to calculate a mean in one sitting (that is, you don't need to save the result for adding in more factors later), then this shorter program is adequate. Here you just leave $\bar{x}_k$ on the stack as you type x_{k+1}.

```
HELP  f1
  PICK 2 ENTER     x̄k is on the stack
SUB                xk − x̄k
REC 6  ENTER
  1  ADD           get k, add 1
SHIFT STO 6
  ENTER            save k+1
DIV                [x(k+1) − x̄k]/k+1
ADD                add x̄k
f1
```

Example: compute mean of 4, 8, and 7.

```
1 SHIFT STO 6
  ENTER DROP       set k = 1
4  PUSH            since x̄1 = x1
8  f1              do x2 = 8
7  f1              do x3 = 7
```

Note that PUSH is not necessary after 7 and 8 before pressing f1 because the f1 program does not start with a number.

An Investment Problem

Over the past several years, the annual inflation rate has varied in a range from four to fourteen percent. Statisticians say that that rate has averaged nine percent, with a variation of two percent; that is the rate distributed normally with mean 9 and variance 2.

If this is true, and a randomly selected group of investors were able to invest so as to keep pace with inflation, what will be the value of their portfolios at the end of a specific time period?

Let us assume that each invests in a fixed-interest-rate account for a two-year period starting at a randomly selected time during the period through which the inflation rate was tracked. It is assumed for this calculation that interest paid was fixed at the inflation rate at the time of the investment. We calculate the value of each account at the end of its specific two-year period.

To examine this situation, we examine hypothetical investors by simulation.

First, we generate a random inflation rate using the normal generation operation $\overline{X}_n$ with mean 0 and variance 1. Then we transform it to a rate with mean 9 and variance 2. Next, considering that we invest and review our situation monthly, we change the random rate to a monthly rate. And, finally, we calculate the amount one dollar would be worth if invested for n months at that rate.

Mathematically, the problem can be stated as follows.

- Generate a normal (0,1) random deviate r'.
- Transform it to $r = \sqrt{2}\ r' + 9$ (r has mean 9 and variance 2 at this point).
- Change r to a percent and then to a monthly inflation rate m = r/12.
- Then $(1 + m)^n$ is the value of one dollar invested for n months at m% compounded monthly.

First, we will write the program, and in doing so, initialize certain program elements so that easy repetition of the simulation is possible. Then, by using the programmable keys f1, f2, and/or f3, we will simplify the operations, reducing them to a few keystrokes. We assume a clear stack at the beginning.

To put the square root of 2 (the desired standard deviation) in register 2 for future use type:

2 $\sqrt{\ }$ SHIFT STO 2 ENTER

Let us put 7325 in register 12 to serve as a seed for the random number generator ($\overline{X}_n$), type;

7325 SHIFT STO 12 ENTER

At this point, the values square root of 2 and 7352 remain on the stack. To discard them, type

DROP DROP

Now the HHC is properly initialized for the investment problem. We will run through this problem step by step.

1. Generate the normal random deviate, r' by pressing

$\overline{X}_n$	(.4278677851)

2. Now transform r' to the desired range by the formula $(r' \times \sqrt{2}\ ) + 9$. Key in

REC 2 ENTER	to get $\sqrt{2}$	(1.41423562)
MULT	$r' \times \sqrt{2}$	(.6050964246)
9 ADD	+ 9	(9.605096425)

3. Change to percent by multiplying by .01. Key in

.01 MULT	(.09605096425)

4. Now get the monthly inflation rate by dividing the yearly rate you have just calculated by twelve. Key in

12 DIV	(.008004247021)

(Note: $(r \times .91)/12 = r/1200$ in steps 3 and 4 can be combined as 1200 DIV)

5. You now have the value m (the monthly inflation rate). Recalling that $(1+m)^n$ is the value of one dollar invested for n months at m% compounded each month, we are now prepared for the final computation. If n = 24 months, type

1 ADD	1 + m	(1.008004247)
24	24	
y^x	to raise 1+m to the 24th power	(1.210867677)

The final result is now displayed. It represents the value of 1 dollar invested for 24 months at a 9.6% annual rate compounded monthly.

If you want to save steps 1 through 4 for future reference, the function keys f1, f2, and f3 can be used.

SECTION 5. FLOATING-POINT NUMBERS AND ALGORITHMS

FLOATING-POINT NUMBERS AND ALGORITHMS

A floating-point number is constructed as follows. (REMEMBER that all "construction" takes place automatically when you enter a number through the keyboard. This detailed information addresses those who may wish to use the scientific capsule as a library capsule, and who require precise knowledge for use in other programming environments).

The floating-point number requires eight bytes (each containing eight bits) of memory.

The first 12 bits (i.e., low memory address) are used for the

- sign bit (0 for positive numbers, 1 for negative numbers)
- exponent bits (eleven bits). These allow an exponent range of from −1023 to 1023; the exponent is considered to be a power of ten; a bias of 1024 is added; an exponent of zero (ten to the zero power, or 1) is bits 10000000000, binary, or 1024 decimal.

The remaining 52 bits are used to hold the mantissa in BCD format as follows.

- The first 4 bits contain the integer part of the floating-point number. A decimal point is implied after this byte.
- Each of the last thirteen half bytes (4-bit groupings) is considered to be one decimal number by the arithmetic componentry.

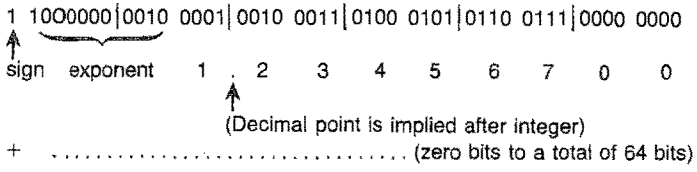
This system has many useful features.

- Excellent symmetry properties exist (see Section 6). For example, the largest number multiplied by the smallest number is very nearly unity (1). The largest positive number is the negation of the largest negative number.
- All valid floating-point numbers are "normalized". That is, the integer part is non-zero for all non-zero values.
- The actual number is taken to be the thirteen-digit integer +fraction multiplied by ten raised to the power (unbiased) stated in the exponent area.

- All other types of floating-point numbers are INVALID, and are generally the result of calculations which should not have been performed (divide by zero, logarithm of a negative number,...,). A result of this type is detected by the error-handling system, and an error message is displayed.

To use the Scientific Calculator effectively, you need not be concerned with the actual construct of floating-point numbers. One example will be enough to demonstrate the concept.

If you enter the number −123.4567, using −123.4567 PUSH, the top of the stack will contain that number in floating-point form as shown below (in bits).



The exponent above is 1026 in decimal form, which implies that ten is raised to the power 2 (1026—the biased form of 2), and that result is multiplied by the integer+fractional part of the number.

The Scientific Calculator allows entry of numbers using scientific notation. The number above can be entered as −1234567E3 (E3 implying multiply by 10 cubed), or as −1234.567E−1. Both entries result in the internal representation shown above.

Section 6 summarizes the boundaries of this floating-point system, and provides information on such numbers as m (machine infinity, the largest representable number) and eps, the smallest non-zero number.

REMEMBER:

- Just as in standard operation, the numbers 10000 and 0.0000000001 cannot be added in 13-digit arithmetic if full significance is desired in the result. If two operands having exponents that differ by more than 12 are used in addition or in subtraction, the result will be identical to the number with the larger absolute value (10000, in the above case, in addition).
- The results of operations cannot exceed the range of the floating-point system. As an example, consider

$$10^{1000}/10^{-1000} = 10^{2000}.$$

The maximum floating-point number (Section 6) is less than that result; in this case overflow (ERROR 2) would be signalled. The operation 1.0/0.0 will cause a divide check error, as a limiting case of overflow.

A review of Section 6 can help you design calculations that avoid these situations. If calculations cause violation of the floating-point system boundaries (Section 6) or use the more complicated mathematical functions in a way that makes adequate results impossible (Section 2 and Section 7), a warning message (Section 8) is displayed. Section 7 defines error types by function, Section 8 by error number.

MATHEMATICAL FUNCTIONS

For arguments in the domains noted in Section 7, the following functions are approximated to give the maximal accuracy. This means ten significant digits in many cases, since thirteen-digit arithmetic is available. Exceptions that may exist are noted in Section 7.

Consider the following situation: Calculate tan(0.6 radians). Then calculate arctan (the result).

Tan(0.6) = 0.6841368083416, correct to less than one digit in the thirteenth place.

Arctan (0.6841368083416) = 0.5999999999999, correct to within one digit in the thirteenth place.

In practice, a rounded result is displayed in the two cases, so that if ten digits were requested, then 0.6841368083 and 0.6 would be displayed.

Consider now the result of the following operations:

tan(0.6) atan (result) tan (that result)
atan (that result) ...

After many of the above iterations, the error of one digit in the thirteenth place would cause final results to stray from 0.6 because the result is 0.5999999999999 after the first iteration.

GENERATION OF RANDOM DEVIATES

Uniform Pseudo-random Deviates

A sequence of positive integers x is generated by the recursion

$$x(i+1) = 16807x(i) \bmod 2147483647.$$

The seed x(1) is presented to the program as an integer in floating-point form in the range [1,2147483646]. Subsequent x's [x(i+1)] can be used on the next program usage.

Before use of the recursion, x is scaled to a floating-point number in the range (0, 1), and returned. The floating point-integer x(i+1) is stored for use in the next generation.

Each integer from 1 to $(2^{31} - 2)$ is generated once in each cycle, and the deviates generated are transportable. That is, deviates thus produced can be reproduced by using the same seed, x, in a correct implementation of the algorithm for another machine in this language or in another one.

For uniform deviates in the range [a,b], the above generator can be used, and the resultant deviate u then transformed by

$$u' = (b-a)u + a.$$

Normal Pseudo-random Deviates (Reference 3)

For normal deviates with mean 0 and standard deviation 1, the uniform(0,1) generator above is used, and two deviates are generated, u and v. Then,

$x = \text{sqrt}\ (8/e)\ (v-0.5)/u$ is obtained.

These operations are repeated until

$x^2 \leq -4/\text{Ln}(u)$; x is the desired deviate.

For normal deviates with mean u and standard deviations, the above normal [0,1] generator can be used, and the resultant deviate x transformed by

$$x' = sx + u.$$

SECTION 6. MACHINE PARAMETERS

The floating point numbers given below represent parameters which define the boundaries of the floating point system used in the HHC. Violation of correct number usage will lead to a floating point operation error (See Sections 7 and 8).

Notation	Parameter	Number
m:	largest floating-point number	$9.9999999999 \times 10^{1023}$
−m:	largest negative floating-point number	$-9.999999999999 \times 10^{1023}$
eps_1:	smallest positive number which, when added to or subtracted from 1, gives a result not equal to 1.	10^{-12}
eps:	smallest positive number	10^{-1024}
−eps:	negative number largest in absolute value	-10^{-1024}
eps':	smallest positive number such that it and its inverse can be inverted, and such that the inverted inverse is the original number.	$1.000000000001\,E-1024$
$1-eps_1$:	largest number less than 1	.9999999999999

SECTION 7. OPERATION KEY PARAMETERS

GENERAL INFORMATION

Section 7 summarizes pertinent information about the Scientific Calculator operation keys. The codes used for argument domain and function range are explained in Section 6. Stack control notation is explained below.

Before Operation		After Operation	Example of applicable execution
x	--->		DROP — drop top item from the stack
	--->	x	REC — recall from memory and put on stack
x	--->	sin(x)	SIN — replace top of stack with its sine
y,x	--->	yx	MUL — replace top two entries with their product

Our convention for noting stack values is that level 1 value x is always presented to the right of level 2 value y. x represents the number on the top of the stack — level 1. y represents the next number on the stack — level 2. When an input key has the letters x and y on it, x represents the number on the top of the stack (it appears on the display), and y represents the next number down.

The x or y,x on the left of the ---> arrow shows stack contents **before** the operation being discussed; the group on the right of the ---> arrow shows stack contents **after** the operation. At most, only the top two levels are considered. The stack holds the argument or arguments for the operations; the result of an operation is always placed on the top of the stack (level 1) and in memory if the M+ and M− keys are used (or in memory location 12 when generating random deviates).

Numbers keyed in will show on the display but do **not** go onto the stack automatically. In two-argument operations, the first number, after being keyed in, must be pushed onto the top of the stack by pressing the ENTER,PUSH,DUP key. When the operation key is pressed (such as MULT) after the second argument is keyed, the second argument is momentarily pushed onto the stack and then both arguments are replaced

by the result of the calculation, which is put in level 1. It is important to know what is at levels 1 and 2 in the ATAN2, SUB, DIV, x^y operations.

In one-argument operations, pressing an operation key (such as COT) after the argument is keyed in will push the argument onto the stack momentarily (pushing present stack contents down one level) before it is replaced with the result of the calculation. However, no harm is done if the number is PUSHed before the operation is pressed.

SYMBOLS

- m, eps, ...: refer to Section 6
- c(n) symbolizes "the contents of floating point storage location n"
- m, as noted in Section 6, is machine infinity
- na is used to imply "non-applicable in this case"
- [x] is the symbol for "the greatest integer in x"
- ( ) implies that the interval is open
- [ ] implies a closed interval
- > greater than
- < less than

FLOATING-POINT ERRORS

error 2: overflow; result < −m or result > m
error 4: divide check; division by zero

These errors can occur if incorrect operation is requested. In the underflow case, when a result is less than 10^{-1024} in absolute value, but not absolutely zero, our implementation of the floating point arithmetic operations sets the result to zero, and does not trigger an underflow error. However, if the capsule is being used as a library program (see Appendix C), underflow can occur.

While the basic mathematical functions discussed below protect against the above errors, operations preceding or succeeding use of these functions can result in such errors. For example, in raising 10 to the power x, first ln(10) is multiplied by x. Then the exponential routine, e to the power x, is entered. The initial multiplicaiton can cause an overflow if x is too large (for example, m). Then, ERROR 2 (overflow) will be displayed.

If the result of any calculation is "ERROR N", (error discussions are given in this section and Section 8), then the answer that would have been the top stack element is automatically dropped.

BEEP Warning: (Any stack conflict will cause a beep.)

- if a number is entered which cannot be converted to an internal floating point counterpart
- if an operation key which does not have a counterpart in the Scientific Calculator is pressed
- if the user addressed a non-existent memory register (location)
- if PICK or ROLL is requested and an n out of range is selected
- if store memory (STO) is requested and the stack is empty
- if DROP is requested and the stack is empty
- if SWAP is requested with less than two numbers on the stack
- if +, −, etc. are used with one operand

ARITHMETIC FUNCTIONS

ADD (+) y,x--->y+x
SUB (−) y,x--->y−x
MULT (×) y,x--->yx
DIV (/) y,x--->y/x

Domain: [−m,m] for both x and y

Range: [−m,m]

ERROR 2: overflow; result < −m or result > m

ERROR 4: divide check; division by zero (DIV only)

Note that if two floating point numbers are added or subtracted, and their exponents differ by more than 12, the result will be identical to the number with the larger absolute value. If the absolute value of the result of an operation is less than 10^{-1024}, it is set to zero.

TRIGONOMETRIC FUNCTIONS

SIN x--->sin(x)

Domain: [−102942,102942] radians
[−5898142,5898142] degrees

Range: [−1,1]

ERROR 9: x is out of the domain noted above

COS x--->cos(x)

Domain: [−102942−π/2,102924−π/2] radians
[−5898052,5898052] degrees

Range: [−1,1]

ERROR 9: x is not in the domain noted above

TAN x--->tan(x)

Domain: [−51471,51471] radians
[−2949071,2949071] degrees

Range: [−m,m]

ERROR 7: x is out of the domain noted above

COT x--->cot(x)

Domain: [−51471,51471] radians
[−2949071,2949071] degrees

Range: [−m,m]

ERROR 6: x < eps'

ERROR 7: |x| > 51471 radians

ASIN x--->arcsin(x)

Domain: [−1,1]

Range: [−π/2,π/2] radians
[−90,90] degrees

ERROR 12: |x| > 1

ACOS x--->arccos(x)

Domain: [−1,1]

Range: [0,π] radians
[0,180] degrees

ERROR 8: |x| > 1

ATAN x--->arctan(x)

Domain: [−m,m]

Range: [−π/2,π/2] radians
[−90,90] degrees

ATAN2 x,y--->arctan (y/x)

Domain: x in [−m,m]
y in [−m,m]

Range: [−π,π] radians
[−180,180] degrees

ERROR 11: x = y = 0 or x = 0

ACOT x--->arccot(x)

Domain: [−m,m]

Range: [0,π] radians
[0,180] degrees

NOTE: Reduction of the arguments used in trigonometric functions can cause accuracy degradation, since x is modified to lie in the principal domain (e.g. [−π/2,π/2] for SIN). The program produces ERROR 9 when the argument is so large that accuracy will degrade more than is acceptable. This situation applies to SIN, COS, TAN, and COT.

HYPERBOLIC FUNCTIONS

SINH x--->sinh(x)

Domain: [−m,m]

Range: [−m,m]

If x is such that sinh(x) >= m, then sinh(x) is set to m.

COSH x--->cosh(x)

Domain: [−m,m]

Range: [1,m]

If x is such that cosh(x) >= m, then cosh(x) is set to m.

TANH x--->tanh(x)

Domain: [−m,m]

Range: [−1,1]

ATANH x--->arctanh(x)

Domain: (−1,1)

Range: [−c,c] where c=15.313376694

ERROR 10: |x| >= 1

LOGARITHMS AND EXPONENTIALS

LOG_2 x--->$\log_2(x)$

Domain: (0,m]

Range: [−m,3401.654369165]

ERROR 14: x <= 0

LOG_{10} x--->$\log_{10}(x)$

Domain: (0,m]

Range: [−m,1024]

ERROR 14: x <= 0

LOG_e x--->ln(x)

Domain: (0,m]

Range: [−m,2357.847135225]

ERROR 14: x <= 0

2^x x--->2^x

Domain: [−m,3401.654369165]

Range: [0,m]

ERROR 13: x > 3401.654369165

10^x x--->10^x

Domain: [−m,1024]

Range: [0,m]

ERROR 2: overflow in ln(10)x prior to e^x entry

ERROR 13: x > 1024

e^x x--->e^x

Domain: [−m,2357.847135225]

Range: [0,m]

ERROR 13: x > 2357.847135225

ROOTS AND POWERS

x^y y,x--->x^y

Domain: x in [−m,m]
y in [−m,m]

Range: [−m,m]

ERROR 2: the multiplication xln(y) caused an overflow

ERROR 13: xln(y) is too large an argument for the e^x program

ERROR 15: x = 0 and y <= 0

ERROR 16: x < 0 and y is not an integer

See the note describing the nth root, below.

y^x y,x--->y^x

Domain: x in [−m,m]
y in [−m,m]

Range: [−m,m]

ERROR 2: the multiplication yln(x) caused an overflow

ERROR 13: yln(x) is too large an argument for the e_x program

ERROR 15: y = 0 and x <= 0

ERROR 16: y < 0 and x is not an integer

See the note describing the nth root, below.

x^2 x--->x^2

Domain: [−sqrt(m),sqrt(m)]

Range: [0,m]

ERROR 2: overflow; result < −m or result > m

1/x x--->1/x

Domain: [−m,m]

Range: [−m,m]

ERROR 4: divide check; division by zero

sqrt x--->sqrt(x)

Domain: [0,m]

Range: [0,sqrt(m)]

ERROR 3: x < 0

nth root x--->$x^{1/n}$

Domain of x: [0,m]

Domain of n: integers from 1 to 50 inclusive

Range: [0,$m^{1/n}$]

ERROR 16: x < 0

In the HHC, x^y and the nth root are calculated using $e^{x\ln(y)}$. Because thirteen-digit arithmetic is performed, if y and x are accurate to machine precision, then the result should provide ten or more digits of accuracy.

In the nth root, if n is large and x is near 1, the result will be 1. n is limited to 50 in the nth root, but the user may consider the other power routines as an alternative for larger values of n.

RANDOM NUMBER GENERATORS

X_u ---->u(0,1)

Domain: seed in [1,2147483646] at memory #12

Range: (0,1)

u(0,1) is a pseudo-random uniform deviate in the exclusive range 0 to 1.

X_n ---->n(0,1)

Domain: seed in [1,2147483646] at memory #12

Range: [−m,m]

n(0,1) is a pseudo-random normal deviate with mean 0 and standard deviation 1.

Prior to use of any generator, a floating point integer seed in the range [1,2147483646] must have been stored in storage location 12.

If the user violates this seed requirement, a specific seed is selected by the generator. Thus, the same deviate stream will be produced on any seed range violation.

ADDITIONAL FUNCTIONS

FRAC x--->x-{x} where {x} is the integer portion of x

Domain: [−m,m]

Range: [0,1)

The significand (mantissa; fractional part of the floating point x) is returned.

FIX x--->[x]

Domain: [−m,m]

Range: [−m,m]

The integer part of x is returned.

|x| x--->|x|

Domain: [−m,m]

Range: [0,m]

CHS x--->-x

Domain: [−m,m]

Range: [−m,m]

MEMORY REGISTER HANDLING

CLM -----

Clears all floating point storage locations to zero.

CLS -----

Clears a user-selected storage location.

STO x--->x

Stores x at user-selected location n.

REC ---->x

The recalled number is moved to the top of the stack.

M− x--->x

Memory is changed; c(n) − x replaces c(n). Note that the memory contents can cause an overflow (ERROR 2).

M+ x--->x

Memory is changed; c(n) + x replaces c(n). Note that the memory contents can cause an overflow (ERROR 2).

NOTE: If n is out of the range 1-12, the HHC will beep. In this case, the prompt for reentry of n is displayed. The user can reenter n or press CANCEL to omit the operation.

STACK MANIPULATION AND CONTROL

DROP C/E y,x--->y

BEEP: the floating point stack was empty

(This presentation is non-standard to emphasize the fact that y, the level 2 value, becomes the new level 1 value.)

SWAP y,x--->x,y

BEEP: fewer than two numbers were on the stack

(This presentation is non-standard to emphasize the fact that the contents of the top two stack locations are swapped.)

PUSH --->x

ERROR 5: the number entered is out of the range [− m,m]

If the floating point stack is filled, HEIGHT WARNING appears on the display, and new numbers are not accepted. Store to memory and drop, or drop elements from the stack.

After number entry or operation, PUSH duplicates the number on top of the stack, leaving it in positions 1 and 2.

If any operation key is pressed after keying a number, that number is automatically PUSHed onto the stack momentarily and is then replaced by the result of the operation.

DISP -----

When this key is pressed, first the stack height (number of elements on the stack) is displayed. Then, repeated pressing causes elements on the stack to be displayed in top-down order until the last element has been displayed. Then the display reverts to keyboard mode, and the value on top of the stack is displayed. Press CANCEL to stop the display cycle in mid cycle to return to keyboard mode.

PICK xk,..,xn,..,x1--->,xk,..,xn,..,x1,xn

BEEP: n is out of range. In this case, the prompt for reentry of n is displayed. The user can reenter n or press CANCEL to omit the operation.

PICK prompts you for the value of n, and then pulls the nth stack element to the top, retaining it at position n.

ROLL xk,..,xn,..,x1--->xk,...x1,xn

ROLL prompts you for the value of n, and then pulls the nth stack element to the top, deleting it from its original position.

CONSTANTS

γ ---> euler's constant

Range: 0.5772156649015

π ---> π

Range: 3.141592653590

OPERATIONAL CONTROL

DIG

BEEP: n is out of range

After pressing the DIG key, the user enters n. Thereafter, n digits rounded are displayed. More than 12 digits will not be accepted. The HHC uses 13-digit arithmetic; mathematical functions can reasonably provide 10 digits in most cases. Exceptions are discussed in this section.

SCI

Toggle switch, initialized to cause standard notation to be used on output. Pressing the key causes output to be displayed in scientific notation (that is, as x.xxxxxxxxEyy) where the number of digits is defined by using the DIG key, and the exponent of 10 is signed. The effect of the key is retained until it is pressed again. The DELETE blip shows on the display when SCI is on.

DEG/RAD

Toggle switch, initialized to expect radian arguments in trigonometric functions and to present radian results in inverse trigonometric functions.

Pressing the key causes conversion to degreees mode; that is, the calculator expects arguments to be presented in degrees and returns results in degrees. The effect of the key is retained until it is pressed again. When the key is pressed, it has no effect on numbers already on the stack.

The INSERT blip shows on the display when in DEG mode.

CANCEL

Cancels the stack display cycle and returns to keyboard mode. Also, this key is used to cancel operations that prompt you for a number (such as DIG, CLS, and STO) before you enter the number with the ENTER, PUSH, DUP key.

CLEAR

To clear stack and keyboard without destroying contents of memory locations 1 to 12 or program keys f1, f2, and f3, press CLEAR once. To return to the HHC primary menu, press CLEAR twice (NOTE that this will destroy the contents of memory locations 1 through 12). CLEAR will cause the SCI and DIG functions to revert to their default values.

INPUT AND EDITING

ENT EXP

Used in conjunction with scientific notation entry.

.

Used to enter decimal points.

+/–

Used to enter the sign of a number. + can be used to emphasize a positive number on a printout, for instance.

→ and ←

Used to move the cursor forward and backward.

PROGRAMMING KEYS

f1, f2, f3

User-definable keys, each of which can represent a sequence of up to 15 keystrokes that will be executed when the key is pressed.

SECTION 8. ERROR MESSAGES

This section presents error conditions in numerical order. Full descriptions of error situations are given by key name in Section 7. Section 6 defines eps and the other boundaries of the floating point system.

ERROR	DISCUSSION	OCCURRENCES
2	floating point overflow	ADD,SUB,MULT,DIV, $\underline{M}-$,M+,10^x,x^y,y^x
3	$x < 0$	$\sqrt{}$
4	zero divide	DIV,1/x,$\sqrt[y]{}$
5	number entered outside of the possible range	keyboard entry
6	$\|x\| < eps$	cot
7	x out of range	cot,tan
8	$\|x\| > 1$	acos
9	x out of range	sin,cos
10	$\|x\| \geq 1$	atanh
11	$x = y = 0$	atan2
12	$\|x\| > 1$	asin
13	x out of range	2^x, e^x, 10^x, x^y, y^x
14	$x \leq 0$	$\log_2$, $\log_{10}$, $\log_e$
15	$x = 0, y < 0$	x^y
	$x < 0, y = 0$	y^x
16	$x < 0$, y not an integer	x^y
	$y < 0$, x not an integer	y^x

NOTE: Press CANCEL to cancel the error message and return to keyboard input mode.

SECTION 9. HHC AND SCIENTIFIC CALCULATOR SYMBOLS

HHC	Calculator	HHC	Calculator	HHC	Calculator
SPACE*	CANCEL	a	TAN	A	ATAN
!	ASIN	b	CLS	B	CLM
"	SINH	c	$\overline{X}_{u(0,1)}$	C	$\overline{X}_{n(0,1)}$
#	LOG_{10}	d	2^x	D	LOG_2
$	|X|	e	e^x	E	LOG_e
%	FRAC	f	$\sqrt{\ }$	F	$\sqrt{\ }$
&	SUB	g	REC	G	STO
'	7	h	MUL	H	MUL
(	8	i	5	I	y
)	9	j	1	J	1
—	γ	k	2	K	2
,	.	l	3	L	3
?	DISP	m	0	M	0
.	ENT EXP	n	DIV	N	DIV
:	ENT EXP	o	6	O	6
1	SIN	p	-	P	+
2	SINH	q	COS	Q	ACOS
3	10^x	r	1/x	R	x^2
4	CHS	s	TANH	S	ATANH
5	FIX	t	M+	T	M−
6	SUB	u	4	U	4
7	7	v	y^x	V	x^y
8	8	w	COSH	W	COSH
9	9	x	ATAN2	X	ATAN2
0	π	y	ADD	Y	ADD
		z	COT	Z	COT
		↓			
C1	DROP/CE		DIG		
C2	SWAP	ROT	DEG/RAD		
C3	ROLL	SEARCH	SCI		
C4	PICK	ENTER	ENTER/ PUSH/DUP		

TECHNICAL NOTE: There are ways to produce a CANCEL other than using the CANCEL key. For example, "2nd SFT" follow by "a" produces a "space" character. The same is true of "2nd SFT" plus any alphabetic key not having an assigned punctuation character for its 2nd SFT interpretation. All such "SPACES" are equivalent to the SPACE bar "SPACE" as far as the Scientific Calculator is concerned and will be considered to be CANCEL.

APPENDIX A. REFERENCES

In the construction of part of the programs in the Scientific Calculator capsule, Friends Amis used algorithms from the following references:

1. Cody, W. J. Jr., and Waite, W. SOFTWARE MANUAL FOR THE ELEMENTARY FUNCTIONS, 1980, Prentice-Hall, Inc.

2. Kahan, W. "Why do we need a floating-point arithmetic standard?" Unpublished Memorandum, U. California, Berkeley, 1981.

3. Knuth, D. E. THE ART OF COMPUTER PROGRAMMING, Addison- Wesley Publishing Company, 1981, p. 125 (Algorithm, Monahan et al).

APPENDIX B. USING PERIPHERALS WITH THE SCIENTIFIC CALCULATOR

Peripheral devices such as a video monitor, an RS-232 printer, or the microprinter may be used to remind you of prior steps while you are in the middle of a calculation. In general, numbers shown on the HHC display are sent to the output peripherals you select, but arithmetic and function symbols are not.

1. If necessary, consult the appropriate instruction manual for each peripheral you want to use.

2. Press the HHC OFF key and

3. Connect the peripheral(s) to the HHC or to an I/O adaptor. Press the HHC ON key, then the I/O key to get to the I/O menu.

4. After making the appropriate I/O settings, exit from the I/O menu to the primary menu and select the SCIENTIFIC CALCULATOR.

If your peripherals are already physically connected, you can change peripheral selection and preserve the current status of your calculations by pressing the I/O key while you are in the Scientific Calculator. When you have completed your selection, press the I/O key again to return directly to the calculator. The I/O key menu responds to the standard HHC keys on the uppermost row of the keyboard — remove the overlay while using the I/O key.

Because the microprinter has a maximum capacity of 15 characters per line, numbers longer than 15 characters will print on two lines, being split at the E. For example, –.123456789012E–123 will print as

–1.23456789012
E–124

APPENDIX C. USING THE SCIENTIFIC CALCULATOR AS A LIBRARY

One of the basic concepts underlying the FORTH and SNAP computer languages is that they behave as if they were the machine code of an abstract computer — the FORTH Machine. By the use of tags (i.e., specific numbers assigned to individual operations), SNAP object code begins to resemble a machine language in appearance. But unlike most machine languages, SNAP is able to redefine itself and alter its functional capabilities considerably. For example, one application might need advanced math and another might need file-handling utilities.

In this instance the mathematical routines in the scientific calculator are the extensions to SNAP being considered. In the following discussion, directions will be given showing how to add functions like SIN and LOG to the SNAP vocabulary. (The following discussion is not needed to implement the library; it merely explains what is going on. Readers can skip to the sample and copy it directly.)

SNAP has a potential vocabulary of over two thousand operations (or "words"). About two hundred are permanently defined inside the HHC and the remaining operations can be assigned by the programmer either by compiling his or her own code in the HHC or by making use of library capsules like the Scientific Calculator. All of the tags in SNAP can be thought of as three- digit hexidecimal numbers. The first of these three digits is the most important one, because it is required that all tags with the same first digit must all reside in the same area (note that there is an exception if the first digit is zero). In other words, if you are using tags numbered 5xx, you cannot have half of them coming from one library capsule and the other half from a different capsule.

In the HHC, the usage of tags is defined in an area of internal RAM called TVECT0. It is set up as follows.

ADDRESS	3 BYTES	APPLIES TO
TVECT0		0-PREFIX (i.e., short tags)
TVECT0+3		1-PREFIX

TVECT0+6	2-PREFIX
.	.
.	.
.	.
TVECT0+24	8-PREFIX

The first two bytes of each three-byte set contain the address of the tag table (see the HHC SNAP programmer's manual) for the tags assigned a specific prefix. The third byte contains a code specifying where these tags reside — library capsules, RAM, control ROMs, etc.

Example: the following code will locate the library capsule and set the TVECT0 area to access it.


```
&LBUF ROMID ROMADDR - + == IDIMAGE
&LBUF ROMVECT ROMADDR - + == VECTIMAGE
&LBUF ROMEXT ROMADDR - + == EXTIMAGE
STRING" SCIENTIFIC CALCULATOR" SCALC
HEX
: GETMATH (S --- )
   0
   BEGIN
      DUP FLIP &EXTRINSIC C@ OR
      ROMADDR &LBUF %LLEN BIGMOVE
      IDIMAGE COUNT SCALC COUNT S=
      IF
         80 OR
         EXTIMAGE C@ 3 * TVECT0 +
         VECTIMAGE @ OVER !
         2+ C!
         EXIT
      ENDIF
      1+ DUP 80 =
   UNTIL
   CR ," NOT FOUND"
   KEY CLEAR
;
```



```
/ CAPSULE NAME

/ BEGINNING CODE FOR ROM SEARCH
/ LOOP THROUGH CAPSULES
/ NEW ROM, SAME RAM
/ READ NEW CAPSULE HEADER
/ COMPARE CAPSULE ID'S

/ IT'S THE SCIENTIFIC CALC
/ ADDRESS IN TVECT0
/ STORE TAGTABLE ADDRESS

/ CAPSULE FOUND - RETURN

/ NOT FOUND
```

NOTE: this is more general than necessary since it is known that ROMEXT contains 5 in the Scientific Calculator.

After GETMATH has been executed, the only question is how to call a function like COSINE. Assume that the tag value of cosine is 525 (hex), then you would define

```
HEX
   T: COS 5 C,-V 25 C,-V T;
   (in HHC Development system)
```

or

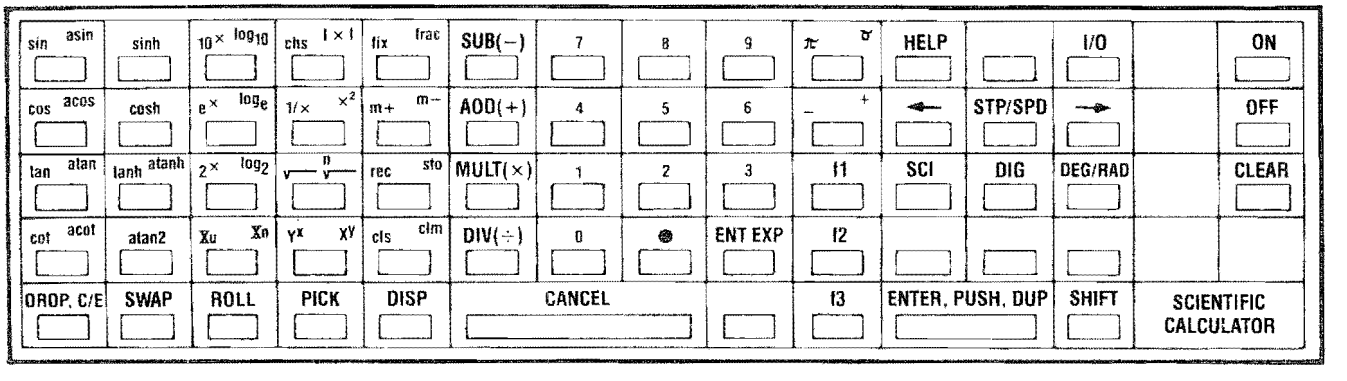
```
   : COS 5 C, 25 C, ; IMMEDIATE
   (in SnapFORTH)
```

before you call COS. Thereafter, COS will compile properly.

Unless stated otherwise, all inputs and results are floating point numbers.

Name	Tag (in hex)	Input	Result	Comment
SCICALC	500	—	—	The Scientific
CAP.INIT	501	—	—	Calculator
FIX	503	X	Integer part of X	sets up
0.0	504	—	0	TVECT0
1.0	505	—	1.0	
.5	506	—	.5	
PI/2	507	—	1.570796326795	
PI	509	—	3.141592653590	
F−	50B	X Y	X−Y	
F+	50C	X Y	X+Y	
F*	50D	X Y	X×Y	
F/	50E	X Y	X/Y	
F0<	50F	X	BOOL (X<0)	
FCMP	510	X Y	X BOOL (ABS[X]> Y)	
ADJUST. SIGN	512	X Y	.SIGN(X)xY	
FLOAT	514	N	X	N is one-word
INT	515	X	N	integer, X is
FINVERT	51C	X	1/X	floating point
ATAN	51E	X	ARCTAN (X)	reverse of float
ATAN2	51F	X Y	ARCTAN (Y/X)	
SQRT	520	X		
EXP	521	X	E^X	
LOG	522	X	LOG_{10} (X)	
LN	523	X	LOGe (X)	
SIN	524	X	SIN (X)	
COS	525	X	COS(X)	
TAN	526	X	TAN(X)	
COT	527	X	COT(X)	
U(0,1)	528	ADDR X		Address of 8-byte seed
N(0,1)	529	ADDR X		Address of 8-byte seed
TANH	52A	X	TANH(X)	
SINH	52B	X	SINH(X)	
COSH	52C	X	COSH(X)	
ASIN	52D	X	ARCSIN(X)	
ACOS	52E	X	ARCCOS(X)	
ATANH	52F	X	ARCTANH(X)	
POWER	530	X Y	x^y	

APPENDIX D. HHC KEYBOARD OVERLAY



FRIENDS AMIS, INC.

The program described in this document is furnished under a license and may be used, copied and disclosed only in accordance with the terms of such license.

FRIENDS AMIS, INC. ("FA") EXPRESSLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE FOR A PARTICULAR PURPOSE RESPECTING THE HHC SOFTWARE PROGRAM AND MANUAL. THE PROGRAM AND MANUAL ARE SOLD "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR USE FOR A PARTICULAR PURPOSE AS TO THE MEDIUM ON WHICH THE SOFTWARE IS RECORDED ARE LIMITED TO SIXTY (60) DAYS FROM THE DATE OF LICENSING BY THE INITIAL USER OF THE PRODUCT AND ARE NOT EXTENDED TO ANY OTHER PARTY.

USER AGREES THAT ANY LIABILITY OF FA HEREUNDER, REGARDLESS OF THE FORM OF ACTION, SHALL NOT EXCEED THE LICENSE FEE PAID BY USER TO FA. FA SHALL NOT BE LIABLE FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, SUCH AS, BUT NOT LIMITED TO, LOSS OR INJURY TO BUSINESS, PROFITS, GOODWILL, OR FOR EXEMPLARY DAMAGES, EVEN IF FA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

FA will not honor any warranty when the product has been subjected to physical abuse or used in defective or non-compatible equipment.

The user shall be solely responsible for determining the appropriate use to be made of the program and establishing the limitations of the program in the user's own operation.

An important note: Good data processing procedure dictates that the user test the program, run and test sample sets of data, and run the system in parallel with the system previously in use for a period of time adequate to insure that results of operation of the computer or programs are satisfactory.