OPERATING INSTRUCTIONS

sinclair
Enterprise Programmable

Sinclair Radionics Ltd.
London Road
St Ives
Huntingdon
Cambs PE17 4HJ

Sinclair Radionics Inc
Galleria
115 East 57th Street
New York
NY 10022
U.S.A.

Part NO 48584 630

INTRODUCTION

The Enterprise Programmable is the latest in a long line of Sinclair calculator 'firsts'. It is our third generation *programmable* calculator – and the most sophisticated yet.

Just take a look at the total package.

As a scientific calculator, the Enterprise Programmable has logs, trigs, six convenience functions, and works in scientific or normal notation. But it also has *seven* addressable memories and *two* levels of brackets. If it stopped there, you'd still have pretty good value for money.

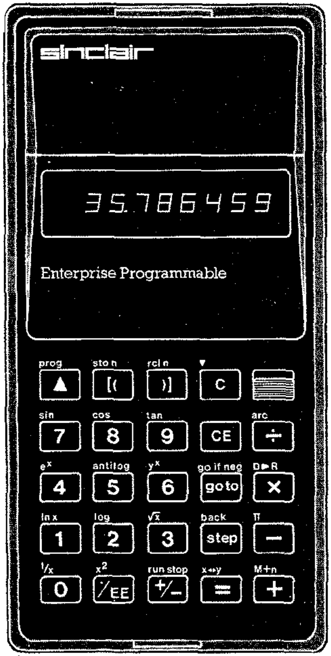
However, you'll only realise the full potential of your new calculator by using the 79-step program facility. With forward and backward stepping, plus conditional and unconditional branching, the Enterprise Programmable becomes a true computer.

And this is where the rest of the package comes in. Together with this instruction book, the AC mains adaptor, and a carrying case, you'll also find a comprehensive program library. It contains over 300 programs, covering everything from discounted cash flow through co-ordinate geometry to special relativity. For ease of everyday use, the library is split up into separate sections and frequently used programs can be detached from the main library if desired.

Whether you use the program library or write your own programs (page 28 of the instructions shows you how); you need never again be without your own, personal computer power.

Finally, of course, the Enterprise Programmable is covered by the famous Sinclair no-quibble guarantee.

CONTENTS



FUNCTIONS AND FEATURES

On/off switch

Display
8 digits plus sign fully floating decimal point, or fixed point scientific notation format. Leading zero suppression.

Arithmetic Functions
$+ - \times \div =$

C key
Clears calculator completely (except program or memory)

CE key
Clears last entry

▲ *key*
Selects upper case functions

Convenience functions
$1/x$, x^2, $\sqrt{x}$, π, $x \leftrightarrow y$, $+/-$.

Brackets
Two sets of brackets provided.

Memories
Seven independent, 3 function memories *(sto n, rcl n, M+ n)*

Logarithmic functions
Common logs and antilogs, Natural (naperian) logs and antilogs (e^x) together with y^x.

Trigonometric functions
Sin, cos, tan and their inverses, with degree to radian and *radian to degree conversions.*

Programmability
Full 79 step program with merged upper case functions, forward and backward step facility, conditional and unconditional branches.

Battery
A battery of the manganese alkaline type, such as Mallory Duracell Mn 1604, is required. No other type of battery may be used.

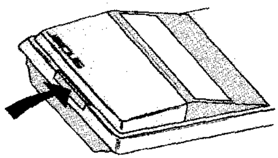
Ac Mains Adaptor
For continuous on-desk use the Sinclair mains adaptor supplied should be used.
Use of any other adaptor invalidates the guarantee

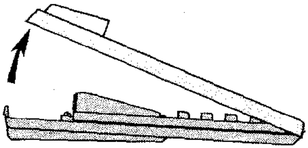
PART 1 MANUAL OPERATION

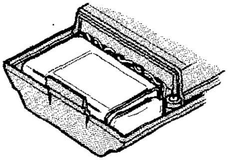
Power Supply

Fitting the battery

Depress the top catch and hinge the case top away from the base. Clip connector to battery and lay battery in the base. Snap on the case top, bottom catch first.







It is essential that only a 9 Volt battery of the Manganese Alkaline type is fitted such as Mallory Duracell Mn 1604. Remove exhausted batteries to prevent any possible damage through leakage.

Ac mains adaptor

It is recommended that wherever possible the Enterprise *Programmable* is powered by the AC mains adaptor supplied, to conserve the battery. *Use of any other adaptor invalidates the guarantee.*

When the adaptor jack is plugged into the socket on the side of the calculator the battery is automatically disconnected.

Keyboard functions and display format — manual operation

+, –, x, ÷, =	Normal arithmetic operators
C	Clear all key
CE	Clear last entry key
+/–	Change sign of number in the display
·/EE	Decimal point/exponent entry key
▲	Upper function key
[( )]	Open and close brackets — 2 sets
1/x	Reciprocal key
x^2	Square key
$\sqrt{x}$	Square root key
π	Display constant pi. 3.1415927
x ◂▸ y	Exchanges number displayed with the last partial answer calculated.
sto n	Stores the number displayed in memory n (2 key steps required.)
rcl n	*Recalls the number stored in memory n* (2 key steps required.)
M+ n	Adds the number displayed to the contents of memory n (2 key steps required.)
log	Common logarithm — Base 10
antilog	Common antilog — Base 10
ln x	Natural logarithm — Base e
e^x	Natural antilog — Base e
sin	sine of number displayed
cos	cosine of number displayed
tan	tangent of number displayed
arc	inverse trigonometric key — used in conjunction with sin, cos or tan
D ▸ R	Converts displayed degrees into radians, or when used with 'arc' from radians to degrees
▼	Cancel upper function key

The keys used for programmable operation are described in Part 2.

The display is used in two ways, floating point format and fixed point, scientific notation format.

Floating point format

The full eight digits are used when the display is holding a floating point number. This will be when the number is in the range of ± 1 to ± 99999999. Results obtained by the use of the arithmetic operators are always displayed in floating point format, unless they fall outside the 'normal' range, even if entries are made in scientific notation.

Fixed point, scientific notation, format

For entry or display of numbers outside the 'normal' floating point range, scientific notation is used.

Any number can be divided into two parts. The first, the mantissa, lies between 1 and 10. The second, the exponent is that 'power of 10' that the mantissa must be multiplied by to give the original number. For example, 123 can be written 1.23×10^2 ie 1.23×100. Similarly, 0.123 can be written 1.23×10^{-1} or 1.23×0.1. This format, known as scientific notation, allows any number between 10^{99} and 10^{-99} to be displayed on an eight digit display.

The display is partitioned as follows:

Position in display	1	2	3	4	5	6	7	8	9
	Sign and symbol		Five digit mantissa				sign	Exponent	

Manual operation

Arithmetic functions, +, –, ÷, x, +/–, ·/EE, C, CE, [(,)].
Normal algebraic rules of entry are used. The expression is entered in a similar way to writing it down.

Example: Calculate 3 + 5

Key	Display	Comment
3	3	enter 3
[+]	3	
5	5	enter 5
[=]	8	answer

Example: Calculate 10 — 8

Key	Display	Comment
10	10	enter 10
[–]	10	
8	8	enter 8
[=]	2	answer

Example: Calculate 25 ÷ 5

Key	Display	Comment
25	25	enter 25
[÷]	25	
5	5	enter 5
[=]	5	answer

Example: Calculate 33 x 6

Key	Display	Comment
33	33	enter 33
[x]	33	
6	6	enter 6
[=]	198	answer

In the previous examples the numbers were all integers and positive. The [./EE] key allows decimal numbers to be entered, with the [+/−] key changing the sign of the number displayed. Pressing the [./EE] key once only enters the decimal point.

Example: Calculate 3.2 x (−5.6)

Key	Display	Comments
3	3.	enter 3
[./EE]	3.	initiate decimal point
2	3.2	3.2 entry completed
[x]	3.2	
5	5.	
[./EE]	5.	initiate decimal point
6	5.6	5.6 entry completed
[+/−]	− 5.6	change sign
[=]	− 17.92	answer

In further examples the key sequence 3/./EE/2 is shown as 3.2

Pressing the [./EE] key twice allows numbers to be entered in scientific notation.

Example: Calculate $(1.3 \times 10^7) \div (3 \times 10^{-3})$

Key	Display	Comment
1.3	1.3	enter 1.3
[./EE]	1.3 00	initiate exponent
[7]	1.3 07	enter exponent
[÷]	13000000	Number less than 99999999 – displayed in floating point format
3	3	enter 3
[./EE]	3	initiate decimal point – not needed
[./EE]	3 00	initiate exponent
3	3 03	enter exponent
[+/−]	3 −03	change sign of exponent
[=]	4.3333 09	answer

In further examples the key sequence 1.3/./EE/7 is shown as 1.3×10^7.

An error made when entering a number can be cleared by pressing the clear last entry key, [CE], once. The display shows the previous subtotal which, together with the last operator entered, is retained for further calculation: enter the correct number to continue. Pressing [CE] a second time clears the whole calculation and display. An error made when entering an operator can be corrected simply by 'overwriting' the wrong operator with the correct one.

Example: Calculate 2.362 x 1.945

Key	Display	Comment
2.362	2.362	enter 2.362
[−] [x]	2.362	correct operator overwrites previous incorrect operator
1.955	1.955	1.955 entered instead of 1.945
[CE]	2.362	clears erroneous entry
1.945	1.945	enter correct number
[=]	4.59409	answer

The clear key, [C], clears the whole calculation and display when pressed after an operator (+, −, x, ÷, =). It is not essential to press [C] after [=], however, because any new number entry after [=] automatically clears the previous calculation and starts a new one.

More complex arithmetical operations can be carried out using the brackets facility. The Enterprise Programmable has two sets of brackets, and these are opened and closed using keys [(] and [)] respectively. It is important to remember that an arithmetic operator must be used *before* any brackets are opened – none are implied.

Example: Calculate 2 x (3 + 5)

Key	Display	Comment
2	2	enter 2
[x]	2	
[(]	2	open brackets
3	3	enter 3
[+]	3	
5	5	enter 5
[)]	8	close brackets – 3 + 5 calculated
[=]	16	answer

Note that the closing of a bracket completes the calculation within that bracket.

Example: Calculate $\frac{(2+3)}{(4+5)}$

Key	Display	Comment
2	2	
[÷]	2	
3	3	
[+]	5	(2 + 3)
[I(]	5	open brackets
4	4	
[+]	4	
5	5	
[)I]	9	close brackets (4 + 5)
[=]	5.5555 −01	answer

Example: Calculate 2 + (3(4 + 5))

Key	Display	Comment
2	2	
[+]	2	
[I(]	2	open brackets 1
3	3	
[×]	3	
[I(]	3	open brackets 2
4	4	
[+]	4	
5	5	
[)I]	9	close brackets 2 (4 + 5)
[)I]	27	close brackets 1 3 × (4 + 5)
[=]	29	answer

The Enterprise Programmable has an *automatic* constant. This allows repetitive operations to be performed without using the memory, and without needing a 'constant' key.

Example: Calculate 2^2, 2^3, 2^4, 2^5 etc

Key	Display	Comment
2	2	
[×]	2	
[=]	4	$2 \times 2 = 2^2$
[=]	8	$2 \times 2 \times 2 = 2^3$
[=]	16	$2 \times 2 \times 2 \times 2 = 2^4$
[=]	32	$2 \times 2 \times 2 \times 2 \times 2 = 2^5$

The constant had taken the operation × 2 and performed that calculation each time the [=] key was pressed. This happens with all four arithmetic operators.

Example: Convert $1.50, $2.30, $35, $50 into pounds if £1 = $1.95

Key	Display	Comment
1.5	1.5	enter first number
[÷]	1.5	
1.95	1.95	constant
[=]	7.6923 −01	$1.5 = £0.77
2.3	2.3	enter second number
[=]	1.1794872	$2.3 = £1.18
35	35	enter third number
[=]	17.948718	$35 = £17.95
50	50	enter fourth number
[=]	25.641026	$50 = £25.64

Example: Convert 35 ins, 23 ins, 100 ins, into centimetres if 1 in = 2.54 cm.

Key	Display	Comment
35	35	First number
[×]	35	
2.54	2.54	constant
[=]	88.9	35 ins = 88.9 cm
23	23	second number
[=]	58.42	23 ins = 58.42 cm
100	100	third number
[=]	254	100 ins = 254 cm

Note that in every case the automatic constant is the last number entered before the [=] and its preceding operator.

Algebraic Functions 1/x, x^2, $\sqrt{x}$, π, x ↔ y

The functions 1/x, x^2 and $\sqrt{x}$ operate on the number in the display. All these functions are "upper case" functions and must be preceded by keying [▲].

Example: Calculate $\frac{1}{3}$, $\frac{1}{4+5}$

Key	Display	Comment
3	3	
[▲] [0] (1/x)	3.3333 -01	answer
4	4	
[+]	4	
5	5	
[=]	9	
[▲] [0] (1/x)	1.1111 -01	answer

It was necessary to use the [=] key before 1/x was used so that 4 + $\frac{1}{5}$ was not calculated. Remember that these functions work on the displayed number and do not complete any arithmetic operation.

Example: Calculate $\sqrt{3^2 + 4^2}$

Key	Display	Comment
3	3	
[▲] [1/x EE] (x^2)	9	3^2
[+]	9	
4	4	
[▲] [1/x EE] (x^2)	16	4^2
[=]	25	$3^2 + 4^2$
[▲] [3] ($\sqrt{x}$)	5	answer

Example: Calculate $\frac{1}{\sqrt{3} + \sqrt{4}}$

Key	Display	Comment
3	3	
[▲] [3] ($\sqrt{x}$)	1.7320508	$\sqrt{3}$
[+]	1.7320508	
4	4	
[▲] [3] ($\sqrt{x}$)	2	$\sqrt{4}$
[=]	3.7320508	$\sqrt{3} + \sqrt{4}$
[▲] [0] (1/x)	2.6794 -01	answer

The constant pi is displayed when the [π] key is pressed. The value of pi used is 3.1415927.

Example: Calculate the area of a circle of diameter 5 cm.
The area $= \pi r^2 = \pi \times 2.5^2 cm^2$

Key	Display	Comment
2.5	2.5	
[▲] [1/x EE] (x^2)	6.25	r^2
[×]	6.25	
[▲] [−] (π)	3.1415927	π
[=]	19.634954	answer 19.63 cm^2

It is possible to exchange the number in the display with the last partial answer stored in the calculator. The exchange function x ↔ y is used to do this.

Example: Calculate $\frac{2}{3+5}$

Key	Display	Comment
3	3	
[+]	3	
5	5	
[+]	8	(3 + 5)
2	2	
[▲] [=] (x ↔ y)	8	exchange 2 ↔ 8
[=]	2.5 -01	answer

The exchange function allows calculations to be made without use of brackets or memory. This becomes important when programming or calculating complex functions.

Memory Functions sto n, rcl n, M+ n

The Enterprise Programmable has seven addressable memories. Three keys, including [▲], are pressed for each memory operation. The first two define the memory function, the third the memory address or number.

Example: Store the number 1.35 in memory 3

Key	Display	Comment
1.35	1.35	
[▲] [((] (sto n)	1.35	initiate memory function – Store
3	1.35	define memory number

To display the contents of a memory a similar key sequence is required.

Example: Clear the display and recall the contents of memory 3.

Key	Display	Comment
[C]	0	Clear Display
[▲] [))] (rcl n)	0	initiate memory function – Recall
3	1.35	define memory number – answer

Displayed numbers can be accumulated in an addressed memory using the M+ n key.

Example: Convert 3 miles, 10 miles and 50 miles to kilometres and sum the results in memory 0. (1 mile = 1.6 km)

Key	Display	Comment
3	3	
[×]	3	
1.6	1.6	
[=]	4.8	first conversion
[▲] [+] (M+n)	4.8	
0	4.8	Store 4.8 in Memory 0
10	10	
[=]	16	second conversion
[▲] [+] (M+n)	16	
0	16	Add 16 to memory 0
50	50	
[=]	80	third conversion
[▲] [+] (M+n)	80	
0	80	Add 80 to memory 0
[▲] [))] (rcl n)	80	
0	100.8	accumulated answer

Note that the memories are numbered from 0 to 6.

Example: In the following table it is necessary to total a set of numbers in both the horizontal and vertical directions. (The use of the memories can make such calculations relatively easy.)

	A	B	C
a	33.2	12.4	6.8
b	20.4	6.3	2.8
c	10.3	13.3	15.5

Totals for columns A, B and C and rows a, b and c are required.

Key	Display	Comment
33.2	33.2	
[▲] [+] (M+n) 0	33.2	First number of Σa stored in Memory 0
[+]	33.2	
20.4	20.4	
[▲] [+] (M+n) 1	20.4	First number of Σb stored in Memory 1
[+]	53.6	(33.2 + 20.4)
10.3	10.3	
[▲] [+] (M+n) 2	10.3	First number of Σc stored in Memory 2
[=]	63.9	ΣA
12.4	12.4	
[▲] [+] (M+n) 0	12.4	Second number of Σa stored in Memory 0
[+]	12.4	
6.3	6.3	
[▲] [+] (M+n) 1	6.3	Second number of Σb stored in Memory 1
[+]	18.7	(12.4 + 6.3)
13.3	13.3	
[▲] [+] (M+n) 2	13.3	Second number of Σc stored in Memory 2
[=]	32.	ΣB

Key	Display	Comment
6.8	6.8	
[▲] [+] 0 (M+n)	6.8	Third number of Σa stored in Memory 0
[+]	6.8	
2.8	2.8	
[▲] [+] 1 (M+n)	2.8	Third number of Σb stored in Memory 1
[+]	9.6	(6.8 + 2.8)
15.5	15.5	
[▲] [+] 2 (M+n)	15.5	Third number of Σc stored in Memory 2
[=]	25.1	ΣC
[▲] [)] (rcl n)	52.4	Σa
[▲] [)] (rcl n)	29.5	Σb
[▲] [)] (rcl n)	39.1	Σc

Therefore the sum of column A is 63.9
the sum of column B is 32
the sum of column C is 25.1
the sum of row a is 52.4
the sum of row b is 29.5
and the sum of row c is 39.1

Logarithmic functions. log, antilog, ln x, e^x, y^x

Logarithms and antilogarithms to both base 10 and base e are directly obtainable on the Enterprise Programmable. A power function, y^x, is also available. All these functions are "upper case" and the [▲] *key must be pressed before the function.* As with the algebraic functions the log. functions operate on *the displayed number only*, except y^x which has special entry key sequences.

Example: Calculate log 5, log 0.3, ln 12, ln 0.05

Key	Display	Comment
5	5	
[▲] [2] (log)	6.9897 -01	answer
0.3	0.3	
[▲] [2] (log)	-5.2287 -01	*answer (the log of any number less than 1 is negative)*
12	12	
[▲] [1] (ln x)	2.4849067	*answer*
0.05	0.05	
[▲] [1] (ln x)	-2.9957323	answer (the log of any number less than 1 is negative)

Example: Calculate antilog 0.5, antilog (−7.1), $e^{3.2}$, $e^{-0.1}$

Key	Display	Comment
0.5	0.5	
[▲] [5] (antilog)	3.1622777	answer
7.1	7.1	
[+/-]	−7.1	
[▲] [5] (antilog)	7.9432 −08	answer
3.2	3.2	
[▲] [4] (e^x)	24.53253	answer
0.1	0.1	
[+/-]	−0.1	
[▲] [4] (e^x)	9.0483 −01	*answer*

As the log and algebraic functions operate on the display "chaining" is possible.

Example: Calculate $e^{1/3} + e^{\sqrt{2}}$

Key	Display	Comment
3	3	
[▲] [0] ($1/x$)	3.3333 −01	$\frac{1}{3}$
[▲] [4] (e^x)	1.3956124	$e^{1/3}$
[+]	1.3956124	
2	2	
[▲] [3] ($\sqrt{x}$)	1.4142136	$\sqrt{2}$
[▲] [4] (e^x)	4.1132505	$e^{\sqrt{2}}$
[=]	5.5088629	answer

Brackets can be used with the functions also.

Example: Calculate $20\,(1 + e^{3/2})$

Key	Display	Comment
20	20	
[×]	20	
[[( ]	20	
1	1	
[+]	1	
[[( ]	1	
3	3	
[÷]	3	
2	2	
[ )]]	1.5	
[▲] [4] (e^x)	4.4816891	$e^{3/2}$
[ )]]	5.4816891	$1 + e^{3/2}$
[=]	109.63378	answer

It was possible to have solved this problem working from inside the brackets first. However, in a complex problem it is often more helpful to calculate in the same order as writing down.

The power function, y^x, needs two number entries. Consequently, this function acts like an arithmetic operator and completes any previous calculation when it is pressed. It must therefore be used at the beginning of a calculation, or to complete a partial calculation that will be raised to a power.

Example: Calculate $2.5^{3.2}$, $3 + 2^3$

Key	Display	Comment
2.5	2.5	enter y
[▲] [6] (y^x)	2.5	
3.2	3.2	enter x
[=]	18.767569	answer
2	2	enter y
[▲] [6] (y^x)	2	
3	3	enter x
[+]	8	2^3
3	3	
[=]	11	answer

If the above example had been keyed as 3/ +/ 2/ y^x/ 3/ = the answer would be 125, which is 5^3 i.e. $(3 + 2)^3$, and is not what was required.
"Chaining" of functions is possible.

Example: Calculate $e^{(3+2)^{1.5}}$

Key	Display	Comment
3	3	
[+]	3	
2	2	(3 + 2)
[▲] [6] (y^x)	5	
1.5	1.5	
[=]	11.18034	$(3 + 2)^{1.5}$
[▲] [4] (e^x)	71706.739	answer

Note that the [=] key was needed before e^x was used. This was because the e^x function does not complete any pending operations, and operates on the displayed number only.

The automatic constant can be used with y^x.

Example: Calculate $1^{3.5}$, $2^{3.5}$, $3^{3.5}$, $4^{3.5}$, $5^{3.5}$, etc

Key	Display	Comment
1	1	
[▲] [6] (y^x)	1	
3.5	3.5	
[=]	1	first answer
2	2	
[=]	11.313708	second answer
3	3	
[=]	46.765372	third answer
4	4	
[=]	128	fourth answer
5	5	
[=]	279.5085	fifth answer

Trigonometric functions sin, cos, tan, arc, D ➢ R

The trigonometric functions operate on the displayed number only, in the same manner as do the log and algebraic functions. The Enterprise Programmable operates in degrees, but has the facility to change degrees into radians and vice versa.

Example: Calculate sin 30°, cos 135°, tan 350°, sin (−110°)

Key	Display	Comment
30	30	
[▲] [7] (sin)	5 −01	answer
135	135	
[▲] [8] (cos)	−70710 −01	answer
350	350	
[9] (tan)	−1.7632 −01	answer
110	110	
[+/−]	−110	
[▲] [7] (sin)	−9.3969 −01	answer

To obtain the inverse of a trigonometric function arc is used. This is an "upper case" function; however the [▲] key only needs pressing once for the whole operation.

Example: Calculate arc sin 0.5, arc cos 0.8, arc tan 10.

Key	Display	Comment
0.5	0.5	
[▲] [+] (arc)	5 −01	
[7] (sin)	30	answer
0.8	0.8	
[+] (arc)	8. −01	
[8] (cos)	36.869898	answer
10	10	
[▲] [+] (arc)	10	
[9] (tan)	84.289407	answer

Example: Convert the following angles from degrees to radians. 100°, 25°, −31°

Key	Display	Comment
100	100	
[▲] [×] (D ➢ R)	1.7453293	answer
25	25	
[▲] [×] (D ➢ R)	4.3633 −01	answer
31	31	
[+/−]	−31	
[▲] [×] (D ➢ R)	−5.4105 01	answer

To convert from radians to degrees arc is used.

Example: Convert 1.5 rad, 2.3 rad, −10 rad, to degrees

Key	Display	Comment
1.5	1.5	
[▲] [+] (arc)	1.5	
[×] (D ➢ R)	85.943669	answer
2.3	2.3	
[▲] [+] (arc)	2.3	
[×] (D ➢ R)	131.78029	answer
10	10	
[+/−]	−10	
[▲] [+] (arc)	−10	
[×] (D ➢ R)	−572.95779	answer

As the trigonometric functions operate on the display only it is possible to chain such calculations.

Example: Calculate sin 30° + cos 40°

Key	Display	Comment
30	30	
[▲] [7] (sin)	5 −01	sin 30°
[+]	5 01	
40	40	
[▲] [8] (cos)	7.6604 −01	cos 40°
[=]	1.2660444	answer

Example: Calculate $e^{\sin\alpha}$ (α = 0.3 radians), log sin 20°

Key	Display	Comment
0.3	0.3	
[▲] [+] (arc)	3 −01	
[×] (D ➢ R)	17.188734	0.3 rad in degrees
[▲] [7] (sin)	2.9552 −01	sin 0.3°
[▲] [4] (e^x)	1.3438252	answer
20	20	
[▲] [7] (sin)	3.4202 −01	sin 20°
[▲] [2] (log)	−4.6594 −01	answer

Input ranges, accuracy and error indication

Any attempted calculation that would result in an answer outside the range of the calculator, or an illegal algebraic operation – dividing by zero for example – will result in the display of "Error". The input and output range is $9.9999999 \times 10^{99} \geq |x| \geq 1.0 \times 10^{-99}$. An "Error" indication will therefore result from attempts to use any of the functions outside the range indicated below.

$1/x$	$\|x\| > 0$
x^2	$\|x\| \leq 9.9999 \times 10^{49}$
$\sqrt{x}$	$x \geq 0$
y^x	$y > 0$ $\|x\|$ is such that any answer is within the normal working range.
lnx, log	$0 < x \leq 9.9999999 \times 10^{99}$
sin, cos	$0° < \|x\| \leq 8999°$
tan	$0° \leq \|x\| \leq 8999°$ but *not* multiples of 90°
arc sin, arc cos	$10^{-50} < \|x\| \leq 1$
arc tan	$0 \leq \|x\| \leq 9.9999999 \times 10^{99}$
antilog	$0 \leq \|x\| \leq 99$
e^x	$-227 < x < 230$

"Error" in the display indicates that the calculation has been aborted; all pending operations are cleared and the calculator is conditioned for new entries. The next keystroke clears "Error" from the display and enters the number or function normally.

The Enterprise Programmable is generally accurate to ± 1 in the eighth digit within the ranges above. However, the accuracy becomes less very near the range limits indicated, especially for the trigonometric functions. For example, the user is expected to make allowances for some inaccuracy in sin 0.001 or tan 89.999.

PART 2 – PROGRAMMABLE OPERATION

Introduction

Programming is essentially a very simple exercise. Basically, as applied to a calculator such as the Enterprise Programmable, it consists of getting the calculator to remember a series of key strokes, so that it can work a problem out by itself.

The type of programming used by the Enterprise Programmable is very easy to use, and bears little resemblance to the more complicated forms of programming used in computers.

When the calculator is put into the program 'mode' each step that is keyed in is remembered by the machine. To enable this to happen, each remembered step is given a number – the step number. This allows the calculator to go through a program in the right order. The Enterprise Programmable has 80 steps – numbered from 00 to 79. Once the calculator has reached *79 it goes back to 00 automatically. If not told to* 'stop' then it would count from 00 to 79 continuously and never stop! Consequently, a permanent 'stop' command is stored at step 00. This leaves 79 steps available for programming.

Any 'upper' function key will only take up one step, as the 'upper' function command, ▲, is 'merged' with the function – thus saving considerable space.

Modes of Operation

The Enterprise Programmable works in two modes. When initially switched on the calculator automatically goes into the *'calculate' mode – and thus needs to be informed when* a program is about to be entered.

Program mode

Two presses of the [▲] key are all that is needed to initiate the program mode. The display will then indicate the step *number – and a code on the right that indicates the* instruction stored at that step number. Unless instructed otherwise the calculator will always start at step 00.

After entry into program mode each keystroke made will be stored by the calculator, the step number 'clocking' on by one with each keystroke entered. All step numbers have two *digits, those below 10 starting with zero eg 02, 05.*

Reversion to calculate mode

The calculator will revert back to calculate mode if the [▲] key is again depressed twice. In the calculate mode, normal manual operation is possible regardless of any program stored by the calculator. However, if any of the program control keys are pressed, manual operation is suspended, and the program takes over.

Keyboard functions and display format — programmable operation

run stop	In program mode: Enters a 'stop' instruction In calculate mode: Initiates program
step	In program mode: Steps program through, one step at a time, for checking. In calculate mode: Allows a calculation, under program control, to be stepped through, displaying partial answers.
back	In program mode: Steps program backwards, one step at a time, allowing corrections to program. In calculate mode: No operation.
goto	In program mode: Instructs program to jump to the step number following the 'goto' instruction In calculate mode: Instructs program to jump to the step number following the 'goto' instruction.
go if neg	In program mode: Instructs program to jump to the step number following the 'go if neg' instruction if the partial answer is negative. If not, the program continues operation, ignoring the instruction and step number. In calculate mode: No operation.

NB If runstop is keyed during the operation of a program, the program will directly enter the calculate mode at that point, i.e. it halts execution. During the execution of a program, no data entry is allowed whilst the display is off. This indicates that a program is being executed.

The display format in program mode consists of three sections.

1	1			F		3	2

The digits on the far left indicate the step number, in this case step number 11. The middle digit, indicating F shows that an upper case function has been stored at step 11, and this function has a code 32, which indicates cos.
For reference the codes are given below.

function	code	function	code
+/—	12	run-stop	F 12
goto	13	go if neg	F 13
[(	22	sto n	F 22
)]	23	rcl n	F 23
÷	24	arc	F 24
7	31	sin	F 31
8	32	cos	F 32
9	33	tan	F 33
x	34	D ▶ R	F 34
4	41	e^x	F 41
5	42	antilog	F 42
6	43	y^x	F 43
—	44	π	F 44
1	51	ln	F 51
2	52	log	F 52
3	53	$\sqrt{x}$	F 53
+	54	M + n	F 54
0	61	1/x	F 61
·/EE	62	x^2	F 62
C	63	▼	F 63
=	64	x ◂▸ y	F 64

(Although these codes seem random, they are easily learnt — and the user will soon become familiar with them).

Using a program from the library.

The simplest method of using the Enterprise Programmable is to take a program from the extensive library provided. For example, the program on page 39 calculates the day of the week for years between 1900 and 2100.

Important note:

In the Program Library the names of some of the functions have been abbreviated as follows:

abbreviation	*function*
sto	sto n
rcl	rcl n
M+	M+ n
a log	antilog
ln	ln x
gin	go if neg
run	run stop
stop	run stop
(	[(
)	])

No reference is made to the [▲] key. Remember to use it when the upper case of a key is required; for instance, for /gin/ key [▲] [go if neg].

To enter this program the program mode is initiated by pressing [▲] twice. Once this has been done, and the display shows 00, the program is entered by pressing keys in the sequence shown in the program library. In this example, the key sequence is:-

sto/0/stop/sto/1/stop/sto/2/rcl/1/—/3/=/gin/2/2/+/4/goto/ 3/1/1/+/—/M +/2/rcl/1/+/1/3/x/2/ 7EE/6/+/1/ 7EE/ 7EE/9/ —/1/ 7EE/ 7EE/9/+/(/rcl/2/x/1/ 7EE/2/5/)/—/2/7/1/1/+/rcl/ 0/+/7/=/gin/6/5/+/1/ 7EE/ 7EE/9/—/1/ 7EE/ 7EE/9/=

Once the program has been entered it is possible to check that no entry errors have been made.

Checking a library program.

The calculator is instructed to return to the calculate mode by pressing [▲] twice. The calculator must then be told to go to the beginning of the program so that the program can be checked. This is achieved by keying goto/0/1. (The key sequence goto/0/0 will insert the halt instruction at 00, and necessitates the use of runstop twice to start the program the first time it is used. The key sequence goto/0/1 avoids this.) Next the program mode is entered again by pressing [▲] twice. Now if [step] is pressed the display will show the key step number and the instruction code.

Example.

Key	Display	Comment
[▲] [▲ prog]	01 F 22	sto n
[step]	02 61	0
[step]	03 12	run stop
[step]	04 22	sto n
[step]	05 51	1
[step]	06 F 12	run stop
[step]	07 F 22	sto n
[step]	08 52	2
[step]	09 F 23	rcl n
[step]	10 51	1
[step]	11 44	—
[step]	12 53	3
[step]	13 64	=
	etc	

If a step was missed, i.e. keyed but not noticed, using the back step function will allow the above sequence to be carried out in the reverse order.

Key	Display	Comment
[step]	09 F 23	rcl n — not seen
[step]	10 51	1 — not seen
[step]	11 44	—
[▲] [back step]	10 51	1 ok!
[▲] [back step]	09 F 23	rcl n ok!
[step]	10 51	1
	etc	

Correcting a library program.

If the above procedure has been followed and no entry errors found, then the program is ready to run. However, if a mistake is found then the program must be corrected.

The easiest way of doing this is to overwrite the step that is wrong. If this is done whilst checking the program it is very simple. When the step containing the error is encountered the back step function is used to step the program back one. The correct instruction may then be entered. Assume step 21 is incorrect:

Key	Display	Comment
[step]	20 53	3 – ok
[step]	21 F 51	an upper function is indicated by mistake – ln x
[▲] [back step]	20 53	3 – ok!
1	21 51	step 21 now correct
[step]	22 51	ok!

If the program had been stepped right through to 79, or 00, however, it would be rather inconvenient to [step] twenty times! The [goto] key can be used here. If calculate mode is re-entered and goto/2/0 keyed then the program will automatically advance to step 20. If program mode is now entered the display will read: 20 53. The correct instruction can now be keyed in. Any steps not overwritten will be unaffected by the procedures shown above.

NB Always go to the step *preceding* the one that is to be corrected.

Running a library program

If the program has been successfully entered it can now be run. Calculate mode must be entered first (key [▲] twice). The library shows the key steps to be entered – usually with an example. It is always useful to run the example shown, to check that the program runs correctly. For the Universal Calendar, the execution instructions show: day/run/month/run/year in full/run/*day of the week*. The black type is the data to be entered, the coloured one being the answer.

If the day of the week for 27th June 1979 is to be calculated the keying is:

27/run/6/run/1979/run. The display shows 3, i.e. a Wednesday.

The first variable is keyed directly into the display. The run instruction initiates the program until a stop is encountered. More data is entered at this point, and run is keyed again. The program will automatically return to step 00 because of the stop instruction permanently stored there. New data may be entered as soon as any previous answer is obtained.
All library programs operate in this way. It is not necessary to know how to program – or even how the program *works – to use the library.*

Remember that abbreviations for keys are used in the Program Library – for instance /gin/ for /go if neg/ – see the list on page 25. Remember also to press [▲] first when a key is to be used in its upper case.

Writing a program

A user-generated program is one written by the user specifically for a calculation that is not in the program library. It could be a modification of one in the library or a *totally new program.*

As the Enterprise Programmable simply 'remembers' the steps that are needed in a calculation, programming from 'scratch' is relatively easy. The first stage in the process is to write down the steps needed to calculate the problem manually. Consider Pythagoras' Theorem as an example. The formula is $c = \sqrt{a^2 + b^2}$. *If this was calculated on the calculator* whilst in the calculate mode the key sequence would be:

enter a/x^2/+/enter b/x^2/=/$\sqrt{x}$

As any variable is treated as input data the program must be stopped at that point to allow entry. This is the only thing *that makes the key sequence different* from that in calculate mode.

Thus the key sequence for programming the above example is:

▲/prog/x^2/+/stop/x^2/=/$\sqrt{x}$

The program mode is entered using the ▲/prog sequence. As the first variable, a, is assumed to be in the display a stop command is not needed. Variable b is entered whilst the program is stopped by the stop instruction. The program will run through to step 00 as any unused steps are automatically filled by code 01 – which is a 'no instruction' code. However, the program execution time can be shortened by telling the program to go to 00 after the $\sqrt{x}$. As the program then only takes 9 steps, time is saved. The program then becomes:

▲/prog/x^2/+/run stop/x^2/=/$\sqrt{x}$/goto/0/0

Checking a user program

The procedure for checking a user-generated program is the same as that for a library program. Taking the example above, the check routine would be:

Key	Display	Comment
[▲] prog [▲]	0	revert to calculate mode
[goto] 00	0	set step counter to 00
[▲] prog [▲]	00	program mode — step 00
[step]	01 F 62	x^2
[step]	02 54	+
[step]	03 F 12	run stop
[step]	04 F 62	x^2
[step]	05 64	=
[step]	06 F 53	$\sqrt{x}$
[step]	07 13	goto
[step]	08 61	0
[step]	09 61	0

If the above sequence is obtained then all is correct.

Correcting a user program

The method of correcting a user program is identical to that for a library program, and is given on Page 26.

Running a user program

A user program is run exactly as a library program. However the step instruction in the calculate mode is useful for checking user programs, as the partial answers at various steps in the program are usually known.

To check the above program with this method it is necessary to go into the calculate mode by pressing [▲] twice, and setting the program step to 00.

If data is now entered from the keyboard, and the step key is pressed instead of run stop, the program will be executed one step at a time.

Key	Display	Comment
[▲] prog [▲]	0	enter calculate mode
[goto]	0	
0	0	
0	0	goto step 00
3	3	let a = 3
[step]	3	enter 3
[step]	9	3^2
[step]	9	$3^2 +$
4	4	let b = 4 (stop command here)
[step]	4	enter 4
[step]	16	4^2
[step]	25	$3^2 + 4^2 =$
[step]	5	$\sqrt{3^2 + 4^2}$

Each partial answer is correct, therefore the program is correct. This ability to step through the program execution is one of the major advantages of the Enterprise Programmable. For a simple example, like the one shown, this is not really evident. But if a more complicated program were used as an example, the power of this facility would be seen.

Once the program is correctly entered, the data required may be entered. The procedure is identical to that on page 25.

Example:
calculate the following using the program above.

a) $\sqrt{1.1^2 + 3.3^2}$ b) $\sqrt{10^7 + 10^8}$ c) $\sqrt{0.01^2 + 0.02^2}$

(Assume that the calculator is in calculate mode)

Key	Display	Comment
[goto]	0	
0	0	
1	0	goto 01 (avoiding run stop at 00)
1.1	1.1	
[▲][+/-]	1.21	1.1^2
3.3	3.3	
[▲][+/-]	3.4785054	answer a)
1 × 10^7	1 07	
[▲][+/-]	1 14	$(10^7)^2$
1 × 10^8	1 08	
[▲][+/-]	1.0049 08	answer b)
0.01	0.01	
[▲][+/-]	1 −04	$(0.01)^2$
0.02	0.02	
[▲][+/-]	2.2360 −02	answer c)

Programming techniques.

A series of worked examples will illustrate some of the techniques used in programming a calculator like the Enterprise Programmable.

As we have seen, a program is the sequence of steps needed to perform a calculation. This sequence can be learnt by the Programmable and performed automatically, with pauses to enter data for the program or to display an answer. The first stage in writing your own program is to arrange your problem in a calculable form bearing in mind the capabilities and requirements of the machine.

It is therefore useful to rearrange an equation so that the least number of steps are used in the program. This makes the execution time quicker — and therefore more acceptable.

Example: Write a program to calculate tanh x, and calculate tanh x for a) x = 2, b) x = 3.5, c) x = −1

The equation to be used is $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

It would be possible to calculate tanh x directly — but this would involve calculating e^x and e^{-x}. The equation can be rewritten as $1 - \frac{2}{e^{2x}+1}$. This is far easier to program:-

▲/prog/+/=/e^x/+/1/÷/2/x ↔ y/−/1/=/±/−/goto/0/0

(The constant is used to generate 2x)

Thus for a) x = 2, b) x = 3.5 and c) x = 1, the results are:-

Key	Display	Comment
[▲][▲ prog]	0	enter calculate mode
[goto] 01	0	avoiding halt at 00
2	2	
[▲][+/- run stop]	9.6402 −01	answer a)
3.5	3.5	
[▲][+/- run stop]	9.9817 −01	answer b)
1	1	
[+/-]	−1	
[▲][+/- run stop]	−7.6519 −01	answer c)

It is also possible to combine two calculations into one program, especially if there is common material. A typical case is a program to calculate the area and circumference of a circle.

Example: Write a program to calculate the area and circumference of circles with radius a) 3 cm b) 5 cm c) 10 cm.

The two formulae used are

$$\text{circumference} = 2\pi r$$
$$\text{area} = \pi r^2$$

The common factor is πr. If the circumference is calculated first the area is then found by the relationship

$$\text{area} = \left(\frac{\text{circumference}}{2}\right) \times r$$

The program for this is:

▲/prog/sto/1/x/2/x/π/=/stop/÷/2/x/rcl/1/=/goto/0/0

To calculate the required areas and circumferences the key sequences are:

Key	Display	Comment
▲ ▲ (prog)	0	enter calculate mode
goto 01	0	*avoiding halt at 00*
3	3	
▲ +/- (run stop)	18.849556	circumference a)
▲ +/- (run stop)	28.274334	area a)
5	5	
▲ +/- (run stop)	31.415927	*circumference b)*
▲ +/- (run stop)	78.539818	area b)
10	10	
▲ +/- (run stop)	62.831854	circumference c)
▲ +/- (run stop)	314.15927	area c)

It is possible to take this program one stage further and develop a single program to calculate the radius, area or circumference of a circle if only one out of the three variables is known.

In this example the goto instruction is used to partition the program.

The formulae used are:

$C = 2\pi r = 2\sqrt{A\pi}$
$A = \pi r^2 = C^2/\pi^4$
$r = C/2\pi = \sqrt{A/\pi}$

Each calculation is programmed in turn and exists as a separate segment of the program.

Thus if the circumference is needed and the radius is known, this calculation is performed starting at 01. Similarly, if the area is needed this calculation is performed starting at 10, and so on. The program then becomes:

▲/prog/x/2/x/π/=/stop/goto/0/1 . . .	$C = 2\pi r$
. . . x²/x/π/=/stop/goto/1/0 . . .	$A = \pi r^2$
. . . ÷/2/÷/π/=/stop/goto/1/8 . . .	$r = C/2\pi$
. . . x²/÷/4/÷/π/=/stop/goto/2/7 . . .	$A = C^2/4\pi$
. . . ÷/π/=/√x/stop/goto/3/7 . . .	$r = \sqrt{A/\pi}$
. . . x/4/x/π/=/√x/stop/goto/4/5 . . .	$C = \sqrt{4A\pi}$

(See sample program section, page 42)

Thus we can now solve the following:

Example:

Calculate a) the area of a circle if the radius is 2 cm
b) the circumference of a circle if the area is 20 cm²
c) the radius of a circle if the circumference is 30 cm
d) the area of a circle if the circumference is 25 cm
e) the circumference of a circle if the radius is 5 cm
f) the radius of a circle if the area is 35 cm²

Key	Display	Comment
▲ ▲ (prog)	0	enter calculate mode
goto 10	0	goto 10 $A = \pi r^2$
2	2	
▲ +/- (run stop)	12.566371	answer a)
C	0	
goto 45	0	goto 45 $C = \sqrt{4A\pi}$
20	20	
▲ +/- (run stop)	15.853309	answer b)
C	0	
goto 18	0	goto 18 $r = C/2\pi$
30	30	
▲ +/- (run stop)	4.7746482	answer c)
C	0	
goto 27	0	goto 27 $A = C^2/4\pi$
25	25	
▲ +/- (run stop)	49.735919	*answer d)*
C	0	
goto 01	0	goto 01 $C = 2\pi r$
5	5	
▲ +/- (run stop)	31.415927	answer e)
C	0	
goto 37	0	goto 37 $r = \sqrt{A/\pi}$
35	35	
▲ +/- (run stop)	3.3377906	answer f)

Another example, showing the use of both the 'goto' and 'go if neg' instructions, is the solving of quadratic equations. In this case the expression $(b^2 - 4ac)$ must be investigated by the program. If the calculator is to solve for both real and imaginary roots, a separate memory must be allocated for each answer. In the program below, one root is put into memory 1 and one root into memory 2 if the roots are real; if the roots are complex, the real part is put into memory 3 and the imaginary part into memory 4.

Example: write a program to solve a quadratic equation and *hence solve a)* $x^2 + 2x - 1 = 0$, *b)* $x^2 + 2x + 1 = 0$, c) $x^2 + 2x + 3 = 0$

The equation $x = \frac{-b^2 \pm \sqrt{b^2 - 4ac}}{2a}$

is re-written as $x = \frac{-b}{2a} \pm \sqrt{\left(\frac{-b}{2a}\right)^2 - \frac{c}{a}}$

This allows the program to be written in less number of steps, thus

▲/prog/sto/0/stop/sto/1/stop/÷/rcl/0/=/sto/5/rcl/1/÷/rcl/0/÷/2/=/+/-/sto/3/x^2/—/rcl/5/=/gin/5/3/$\sqrt{x}$/sto/4/+/rcl/3/=/sto/1/rcl/3/—/rcl/4/=/sto/2/stop/goto/0/1/+/-/$\sqrt{x}$/sto/4/0/sto/1/sto/2/stop/goto/0/1

It should be possible to write this program using less steps — but the above program shows the simplest method employed for solving the problem.

The problems posed can now be solved:

	Key	Display	Comment
a)	▲ ▲ (log)	0	enter calculate mode
	goto 01	0	
	1	1	a
	▲ +/- (run stop)	1	enter a
	2	2	b
	▲ +/- (run stop)	2	enter b
	1	1	
	+/-	−1	c
	▲ +/- (run stop)	−2.4142136	enter c
	▲)((rcl n) 1	4.1421 −01	x_1
	▲)((rcl n) 2	−2.4142136	x_2
	c	0	

	Key	Display	Comment
b)	1	1	a
	▲ +/- (run stop)	1	enter a
	2	2	b
	▲ +/- (run stop)	2	enter b
	1	1	c
	▲ +/- (run stop)	−1	
	▲)((rcl n) 1	−1	x_1
	▲)((rcl n) 2	−1	x_2
	c	0	
c)	1	1	a
	▲ +/- (run stop)	1	enter a
	2	2	b
	▲ +/- (run stop)	2	enter b
	3	3	c
	▲ +/- (run stop)	0	
	▲)((rcl n) 1	0	} no real roots
	▲)((rcl n) 2	0	
	▲)((rcl n) 3	−1	real part of complex roots
	▲)((rcl n) 4	1.4142136	imaginary part of complex roots

the answers are therefore:

a) $x_1 = 0.41421$, $x_2 = -2.4142136$
b) $x_1 = x_2 = -1$
c) $x_1 = -1 + j1.4142136$, $x_2 = -1 - j1.4142136$

PART 3 – SAMPLE PROGRAMS

In this section are five programs. Some are from the program library, others from the text of this instruction manual. Some explanation of the program is given, although not if the program is simple. The user should read through page 25 before loading any programs.

Points worth remembering:-

1. After finishing entering the program always enter C/goto/0/1 before the pre-execution or the execution.
2. During execution always wait till the display lights up before entering a new number.
3. If you make a mistake during execution, in general, you must enter C/goto/0/1 and repeat the pre-execution and execution from the beginning.
4. The easiest way to check a program is to test it with some numerical data for which you know what the correct answer should be.
5. A list of the abbreviations for keys used in the program library is found on page 25. Remember to key [▲] before a key is used in its upper case.

UNIVERSAL CALENDAR

This program finds the day of the week given the date. It is set for dates from 1st March 1900 to *28th February 2100. For dates* from 1st March 2100 to 28th February 2200 substitute 2943 for steps 56–59 in the program. For Western European dates from 1st March 1800 to 28th February 1900 substitute 2591, for 14th September 1752 to 28th February 1800 substitute 2471. For dates before 1752 in England (and for some dates after that in other *countries) historical methods must* be used to find the day of the week, *because of the variations in* calendars and the date of New Year's day.

Execution:

day/run/month/run/year in full/run/day of week

Jan = 1, Feb = 2 etc.

in answer Sun = 0, Mon = 1, Tue = 2, . . . , Sat = 6.

Example:

26/run/12/run/1976/0

So 26th December 1976 was a Sunday.

KEY	#	KEY	#
HALT	00	—	40
sto	01	1	41
0	02	./EE	42
stop	03	./EE	43
sto	04	9	44
1	05	+	45
stop	06	(	46
sto	07	rcl	47
2	08	2	48
rcl	09	×	49
1	10	1	50
—	11	./EE	51
3	12	2	52
=	13	5	53
gin	14	)	54
2	15	—	55
2	16	2	56
+	17	7	57
4	18	1	58
goto	19	1	59
3	20	+	60
1	21	rcl	61
1	22	0	62
+/—	23	+	63
M+	24	7	64
2	25	=	65
rcl	26	gin	66
1	27	6	67
+	28	5	68
1	29	+	69
3	30	1	70
×	31	./EE	71
2	32	./EE	72
./EE	33	9	73
6	34	—	74
+	35	1	75
1	36	./EE	76
./EE	37	./EE	77
./EE	38	9	78
9	39	=	79

PYTHAGORAS' THEOREM

$c = \sqrt{a^2 + b^2}$

This program is explained on page 28 of this instruction booklet.

Example:

Find c, if a = 2, b = 3 cm.

2/run/3/run/3.6055513

KEY	#	KEY	#
HALT	00		40
x^2	01		41
÷	02		42
stop	03		43
x^2	04		44
=	05		45
$\sqrt{x}$	06		46
goto	07		47
0	08		48
0	09		49
	10		50
	11		51
	12		52
	13		53
	14		54
	15		55
	16		56
	17		57
	18		58
	19		59
	20		60
	21		61
	22		62
	23		63
	24		64
	25		65
	26		66
	27		67
	28		68
	29		69
	30		70
	31		71
	32		72
	33		73
	34		74
	35		75
	36		76
	37		77
	38		78
	39		79

TANH X

This program computes the value of tanh x using the formula

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

as described on page 32 of this booklet.

Example:

Calculate tanh x if x = 3

3/run/9.9505 × 10^{-1}

KEY	#	KEY	#
HALT	00		40
+	01		41
=	02		42
e^x	03		43
+	04		44
1	05		45
÷	06		46
2	07		47
x ↔ y	08		48
−	09		49
1	10		50
=	11		51
+/−	12		52
goto	13		53
0	14		54
0	15		55
	16		56
	17		57
	18		58
	19		59
	20		60
	21		61
	22		62
	23		63
	24		64
	25		65
	26		66
	27		67
	28		68
	29		69
	30		70
	31		71
	32		72
	33		73
	34		74
	35		75
	36		76
	37		77
	38		78
	39		79

CIRCLE

This program is described on page 33 of this instruction booklet.

Circumference

$C = 2\pi r = 2\sqrt{A\pi}$

Area

$A = \pi r^2 = C^2/\pi 4$

Radius

$r = C/2\pi = \sqrt{A/\pi}$

Execution:

(a) goto/0/1/r/run/C
(b) goto/1/0/r/run/A
(c) goto/1/8/C/run/r
(d) goto/2/7/C/run/A
(e) goto/3/7/A/run/r
(f) goto/4/5/A/run/C

It is not necessary if you want to repeat the use of program (a) *to start again with* goto/0/1/,

to repeat program (b), you need not re-enter goto/1/0/,

similarly for all the program segments.

The program segments on the right can be used on their own — it is not necessary to enter the *entire program — only those parts* which are required.

KEY	#	KEY	#
HALT	00	$\sqrt{x}$	40
×	01	stop	41
2	02	goto	42
×	03	3	43
π	04	7	44
=	05	×	45
stop	06	4	46
goto	07	×	47
0	08	π	48
1	09	=	49
x^2	10	$\sqrt{x}$	50
×	11	stop	51
π	12	goto	52
=	13	4	53
stop	14	5	54
goto	15		55
1	16		56
0	17		57
÷	18		58
2	19		59
÷	20		60
π	21		61
=	22		62
stop	23		63
goto	24		64
1	25		65
8	26		66
x^2	27		67
÷	28		68
4	29		69
÷	30		70
π	31		71
=	32		72
stop	33		73
goto	34		74
2	35		75
7	36		76
÷	37		77
π	38		78
=	39		79

QUADRATIC EQUATIONS

The solution of the equation

$ax^2 + bx + c = 0$

is given by

$$x = \frac{-b}{2a} \pm \sqrt{\left(\frac{-b}{2a}\right)^2 - \frac{c}{a}}$$

This program is described on p 35 of this instruction manual.

Execution:

a/run/b/run/c/run

for real roots recall memories 1 and 2

for imaginary roots recall memories 3 and 4

(memories 1 and 2 contain 0 if roots are imaginary)

Example:

$x^2 + 2x + 3 = 0$

1/run/2/run/3/run

rcl 1/0/rcl 2/0/(roots imaginary)

rcl 3/−1/rcl 4/1.4142136

Thus the roots are

$-1 + j\sqrt{2}$ and $-1 - j\sqrt{2}$

NB A more "efficient" program is given in the program library.

KEY	#	KEY	#
HALT	00	1	40
sto	01	rcl	41
0	02	3	42
stop	03	—	43
sto	04	rcl	44
1	05	4	45
stop	06	=	46
÷	07	sto	47
rcl	08	2	48
0	09	stop	49
÷	10	goto	50
sto	11	0	51
5	12	1	52
rcl	13	+/—	53
1	14	√x	54
÷	15	sto	55
rcl	16	4	56
0	17	0	57
÷	18	sto	58
2	19	1	59
=	20	sto	60
+/—	21	2	61
sto	22	stop	62
3	23	goto	63
x^2	24	0	64
—	25	1	65
rcl	26		66
5	27		67
=	28		68
gin	29		69
5	30		70
3	31		71
√x	32		72
sto	33		73
4	34		74
+	35		75
rcl	36		76
3	37		77
=	38		78
sto	39		79

SERVICE & GUARANTEE

For guarantee details, please see separate insert contained in packaging.