HP 35s scientific calculator

user's guide

invent

Edition 1

HP part number F2215AA-90001

Notice

REGISTER YOUR PRODUCT AT: www.register.hp.com

THIS MANUAL AND ANY EXAMPLES CONTAINED HEREIN ARE PROVIDED "AS IS" AND ARE SUBJECT TO CHANGE WITHOUT NOTICE. HEWLETT-PACKARD COMPANY MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.

HEWLETT-PACKARD CO. SHALL NOT BE LIABLE FOR ANY ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MANUAL OR THE EXAMPLES CONTAINED HEREIN.

Hewlett-Packard Company
16399 West Bernardo Drive
MS 8-600
San Diego, CA 92127-1899
USA

Printing History

Edition 1 February 2007

Contents

Part 1. Basic Operation

Part 2. Programming

Part 1

Basic Operation

1

Getting Started

✓ **Watch for this symbol in the margin. It identifies examples or keystrokes that are shown in RPN mode and must be performed differently in ALG mode.**
Appendix C explains how to use your calculator in ALG mode.

Important Preliminaries

Turning the Calculator On and Off

To turn the calculator on, press C. ON is printed on the bottom of the C key.

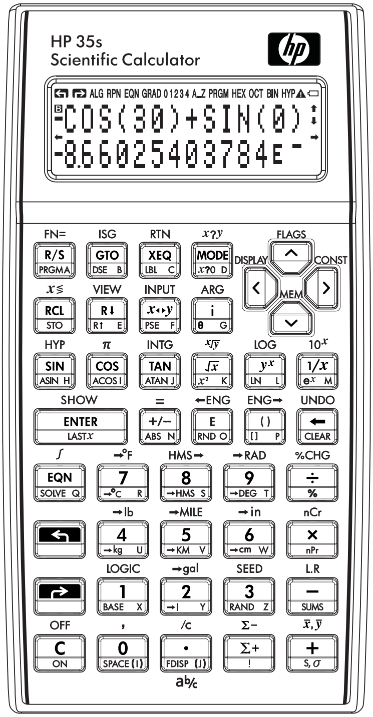
To turn the calculator off, press ↰ C. That is, press and release the ↰ shift key, then press C (which has OFF printed in yellow above it). Since the calculator has *Continuous Memory*, turning it off does not affect any information you've stored.

To conserve energy, the calculator turns itself off after 10 minutes of inactivity. If you see the low–power indicator (▭) in the display, replace the batteries as soon as possible. See appendix A for instructions.

Adjusting Display Contrast

Display contrast depends on lighting, viewing angle, and the contrast setting. To increase or decrease the contrast, hold down the C key and press + or −.

Highlights of the Keyboard and Display

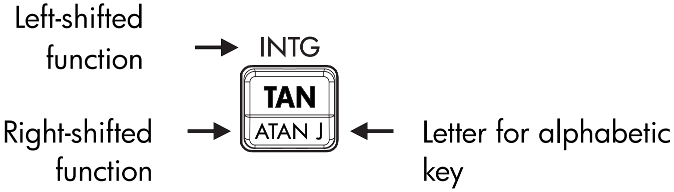

HP 35s
Scientific Calculator
ALG RPN EQN GRAD 01234 A..Z PRGM HEX OCT BIN HYP
COS(30)+SIN(0)
-8.66025403784E-
FN= ISG RTN x?y FLAGS
R/S PRGMA GTO DSE B XEQ LBL C MODE x?0 D
DISPLAY CONST MEM
x≶ VIEW INPUT ARG
RCL STO R↓ R↑ E x↔y PSE F i θ G
HYP π INTG x√y LOG 10^x
SIN ASIN H COS ACOS I TAN ATAN J √x x² K y^x LN L 1/x e^x M
SHOW = ←ENG ENG→ UNDO
ENTER LASTx +/− ABS N E RND O ( ) [ ] P ← CLEAR
∫ →°F HMS→ →RAD %CHG
EQN SOLVE Q 7 →°C R 8 →HMS S 9 →DEG T ÷ %
→lb →MILE →in nCr
4 →kg U 5 →KM V 6 →cm W × nPr
LOGIC →gal SEED L.R
1 BASE X 2 →l Y 3 RAND Z − SUMS
OFF , /c Σ− x̄,ȳ
C ON 0 SPACE ( ) • FDISP ( ) Σ+ ! + s,σ
a b/c

Shifted Keys

Each key has three functions: one printed on its face, a left–shifted function (yellow), and a right–shifted function (blue). The *shifted* function names are printed in yellow above and in blue on the bottom of each key. Press the appropriate shift key (↰ or ↱) before pressing the key for the desired function. For example, to turn the calculator off, press and release the ↰ shift key, then press C.

Pressing ↰ or ↱ turns on the corresponding ↰ or ↱ *annunciator* symbol at the top of the display. The annunciator remains on until you press the next key. To cancel a shift key (and turn off its annunciator), press the same shift key again.

Alpha Keys



Most keys display a letter in their bottom right corner, as shown above. Whenever you need to type a letter (for example, a variable or a program *label*), the **A..Z** annunciator appears in the display, indicating that the alpha keys are "active".

Variables are covered in chapter 3; labels are covered in chapter 13.

Cursor Keys

Each of the four cursor direction keys is marked with an arrow. In this text we will use the graphics >, <, ∧ and ∨ to refer to these keys.

Backspacing and Clearing

Among the first things you need to know are how to clear an entry, correct a number, and clear the entire display to start over.

Keys for Clearing

Key	Description
←	*Backspace.* If an expression is in the process of being entered, ← erases the character to the left of the entry cursor (_). Otherwise, with a completed expression or the result of a calculation in line 2, ← replaces that result with a zero. ← also clears error messages and exits menus. ← behaves similarly when the calculator is in program-entry and equation-entry modes, as discussed below: ■ Equation–entry mode: If an equation is in the process of being entered or edited, ← erases the character immediately to the left of the insert cursor; otherwise, if the equation has been entered (no insert cursor present), ← deletes the entire equation. ■ Program-entry mode: If a program line is in the process of being entered or edited, ← erases the character to the left of the insert cursor; otherwise, if the program line has been entered, ← deletes the entire line.
C	*Clear* or *Cancel.* Clears the displayed number to zero or cancels the current situation (such as a menu, a message, a prompt, a catalog, or Equation–entry or Program–entry mode).

Keys for Clearing (continued)

Key	Description
⮫ CLEAR	*The CLEAR menu* (x VARS ALL Σ STK CLVARx) contains options for clearing x (the number in the X-register), all direct variables, all of memory, all statistical data, all stacks and indirect variables. If you press 3(3ALL), a new menu CLR ALL? Y N is displayed so you can verify your decision before erasing everything in memory. During program entry, 3ALL is replaced by 3PGM. If you press 3 (3PGM), a new menu CLR PGMS? Y N is displayed, so you can verify your decision before erasing all your programs. During equation entry, 3ALL is replaced by 3EQN. If you press 3 (3EQN), the CLR EQN? Y N menu is displayed, so you can verify your decision before erasing all your equations. When you select 6 (CLVARx), the command is pasted into the command line with three placeholders. You must enter a 3-digit number in the placeholder blanks. Then all the indirect variables whose addresses are greater than the address entered are erased. For example: CLVAR056 erases all indirect variables whose address is greater than 56.

Using Menus

There is a lot more power to the HP 35s than what you see on the keyboard. This is because 16 of the keys are menu keys. There are 16 menus in all, which provide many more functions, or more options for more functions.

HP 35s Menus

Menu Name	Menu Description	Chapter
	Numeric Functions	
L.R.	$\hat{x}$ $\hat{y}$ r m b Linear regression: curve fitting and linear estimation.	12
$\bar{x}$, $\bar{y}$	$\bar{x}$ $\bar{y}$ $\bar{x}w$ Arithmetic mean of statistical *x*– and *y*–values; weighted mean of statistical *x*–values.	12
s,σ	sx sy σx σy Sample standard deviation, population standard deviation.	12
CONST	Menu to access the values of 41 physics constants—refer to "Physics constants" on page 4–8.	4
SUMS	n Σx Σy Σx^2 Σy^2 Σxy Statistical data summations.	12
BASE	DEC HEX OCT BIN d h o b Base conversions (decimal, hexadecimal, octal, and binary).	12
INTG	SGN INT÷ Rmdr INTG FP IP Sign value, integer division, remainder from division, greatest integer, fractional part, integer part	4,C
LOGIC	AND XOR OR NOT NAND NOR Logic operators	11

	Programming Instructions	
FLAGS	`SF CF FS?` Functions to set, clear, and test flags.	14
x?*y*	≠ ≤ < > ≥ = Comparison tests of the X–and Y–registers.	14
x?0	≠ ≤ < > ≥ = Comparison tests of the X–register and zero.	14
	Other functions	
MEM	`VAR PGM` Memory status (bytes of memory available); catalog of variables; catalog of programs (program labels).	1, 3, 12
MODE	`DEG RAD GRAD ALG RPN` Angular modes and operation mode	4, 1

DISPLAY	`FIX SCI ENG ALL. , 1,000 1000 xiy`
	`x+yi rθa`
	Fixed, scientific, engineering, full floating point numerical display formats; radix symbol options (. or ,); complex number display format (in RPN mode, only xiy and rθa are available)
		1

R↓ R↑	`X Y Z T` Functions to review the stack in ALG mode –X–, Y–, Z–, T–registers	C
CLEAR	Functions to clear different portions of memory—refer to ↱ CLEAR in the table on page 1–5.	1, 3, 6, 12

To use a menu function:

1. Press a menu key to display a set of menu items.
2. Press ▷ ◁ △ ▽ to move the underline to the item you want to select.
3. Press ENTER while the item is underlined.

With numbered menu items, you can either press ENTER while the item is underlined, or just enter the number of the item.

Some menus, like the CONST and SUMS, have more than one page. Entering these menus turns on the ⬆ or ⬇ annunciator. In these menus, use the ⟩ and ⟨ cursor keys to navigate to an item on the current menu page; use the ⌄ and ⌃ keys to access the next and previous pages in the menu.

Example:

In this example, we use the DISPLAY menu to fix the display of numbers to 4 decimal places and then compute 6÷7. The example closes using the DISPLAY menu to return to full floating point display of numbers.

Keys:	Display:	Description:
	0 0	Initial display
↰ DISPLAY	1FIX 2SCI 3ENG 4ALL	Enter the DISPLAY menu
1 or ENTER	FIX ______	The Fix command is pasted to line 2
4	0.0000 0.0000	Fix to 4 decimal places
6 ENTER 7 ÷	0.0000 0.8571	Perform the division
↰ DISPLAY 4	0 8.57142857143E-	Return to full precision

Menus help you execute dozens of functions by guiding you to them. You don't have to remember the names of all the functions built into the calculator nor search through the functions printed on the keyboard.

Exiting Menus

Whenever you execute a menu function, the menu automatically disappears, as in the above example. If you want to leave a menu *without* executing a function, you have three options:

- Pressing ⬅ backs out of the 2–level CLEAR or MEM menu, one level at a time. Refer to ↱ CLEAR in the table on page 1–5.
- Pressing ⬅ or C cancels any other menu.

Keys:	Display:		
1 2 3 . 5 6 7 8	123.5678_		
↰ DISPLAY	1FIX	2SCI	⬇
	3ENG	4ALL	
⬅ or C	123.5678_		

- Pressing another menu key replaces the old menu with the new one.

Keys:	Display:		
1 2 3 . 5 6 7 8	123.5678_		
↰ DISPLAY	1FIX	2SCI	⬇
	3ENG	4ALL	
↱ CLEAR	1X	2VARS	⬇
	3ALL	4Σ	
C	123.5678		

RPN and ALG Modes

The calculator can be set to perform arithmetic operations in either RPN (Reverse Polish Notation) or ALG (Algebraic) mode.

In Reverse Polish Notation (RPN) mode, the intermediate results of calculations are stored automatically; hence, you do not have to use parentheses.

In Algebraic mode (ALG), you perform arithmetic operations using the standard order of operations.

To select RPN mode:

Press MODE 5 (5RPN) to set the calculator to RPN mode. When the calculator is in RPN mode, the **RPN** annunciator is on.

To select ALG mode:

Press MODE 4 (4ALG) to set the calculator to ALG mode. When the calculator is in ALG mode, the **ALG** annunciator is on.

Example:

Suppose you want to calculate 1 + 2 = 3.

In RPN mode, you enter the first number, press the ENTER key, enter the second number, and finally press the arithmetic operator key: +.

In ALG mode, you enter the first number, press +, enter the second number, and finally press the ENTER key.

RPN mode	ALG mode
1 ENTER 2 +	1 + 2 ENTER

In ALG mode, the results and the calculations are displayed. In RPN mode, only the results are displayed, not the calculations.

Note

You can choose either ALG (Algebraic) or RPN (Reverse Polish Notation) mode for your calculations. Throughout the manual, the "✓" in the margin indicates that the examples or keystrokes in RPN mode must be performed differently in ALG mode. Appendix C explains how to use your calculator in ALG mode.

Undo key

The Undo Key

The operation of the Undo key depends on the calculator context, but serves largely to recover from the deletion of an entry rather than to undo any arbitrary operation. See *The Last X Register* in Chapter 2 for details on recalling the entry in line 2 of the display after a numeric function is executed. Press ↰ UNDO immediately after using ← or C to recover:

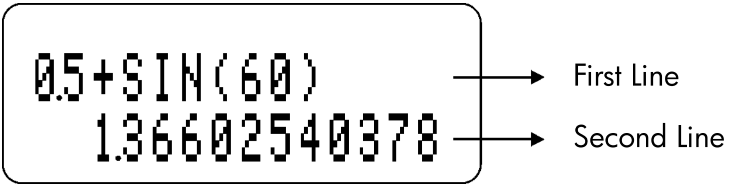
- an entry that you deleted
- an equation deleted while in equation mode
- a program line deleted while in program mode

In addition, you can use Undo to recover the value of a register just cleared using the CLEAR menu. The Undo operation must immediately follow the delete operation; any intervening operations will keep Undo from retrieving the deleted object. In addition to retrieving an entire entry after its deletion, Undo can also be used while editing an entry. Press ↰ UNDO while editing to recover:

- a digit in an expression that you just deleted using ←
- an expression you were editing but cleared using C
- a character in an equation or program that you just deleted using ← (while in equation or program mode)

Please note also that the Undo operation is limited by the amount of available memory.

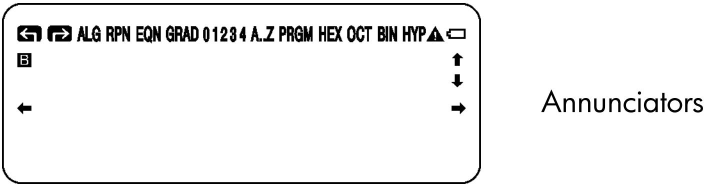
The Display and Annunciators



```
0.5+SIN(60)
  1.36602540378
```

The display comprises two lines and *annunciators*.

Entries with more than 14 characters will scroll to the left. During input, the entry is displayed in the first line in ALG mode and the second line in RPN mode. Every calculation is displayed in up to 14 digits, including an E sign (exponent), and exponent value up to three digits.



The symbols on the display, shown in the above figure, are called *annunciators*. Each one has a special significance when it appears in the display.

HP 35s Annunciators

Annunciator	Meaning	Chapter
B	The "B (Busy)" annunciator appears while an operation, equation, or program is executing.	
▲ ▼	When in Fraction–display mode (press ↱ FDISP), only one of the "▲" or "▼" halves of the "▲▼"' annunciator will be turned on to indicate whether the displayed numerator is slightly less than or slightly greater than its *true* value. If neither part of "▲▼" is on, the *exact* value of the fraction is being displayed.	5
↰	Left shift is active.	1
↱	Right shift is active.	1
RPN	Reverse Polish Notation mode is active.	1, 2
ALG	Algebraic mode is active.	1, C
PRGM	Program–entry is active.	13
EQN	Equation–entry mode is active, or the calculator is evaluating an expression or executing an equation.	6
0 1 2 3 4	Indicates which flags are set (flags 5 through 11 have no annunciator).	14
RAD or **GRAD**	Radians or Grad angular mode is set. DEG mode (default) has no annunciator.	4
HEX OCT BIN	Indicates the active number base. DEC (base 10, default) has no annunciator.	11
HYP	Hyperbolic function is active.	4, C

HP 35s Annunciators (continued)

Annunciator	Meaning	Chapter
⬅, ➡	There are more characters to the left or right in the display of the entry in line 1 or line 2. Both of these annunciators may appear simultaneously, indicating that there are characters to the left and right in the display of an entry. Entries in line 1 with missing characters will show an ellipsis (…) to indicate missing characters. In RPN mode, use the ⟩ and ⟨ keys to scroll through an entry and see the leading and trailing characters. In ALG mode, use ↱ ⟩ and ↱ ⟨ to see the rest of the characters.	1, 6
⬆, ⬇	The ⌄ and ⌃ keys are active for stepping through an equation list, a catalog of variables, lines of a program, menu pages, or programs in the program catalog.	1, 6, 13
A..Z	The alphabetic keys are active.	3
⚠	Attention! Indicates a special condition or an error.	1
▭	Battery power is low.	A

Keying in Numbers

The minimum and maximum values that the calculator can handle are $\pm 9.99999999999^{499}$. If the result of a calculation is beyond this range, the error message "OVERFLOW" appears momentarily along with the ⚠ annunciator. The overflow message is then replaced with the value closest to the overflow boundary that the calculator can display. The smallest numbers the calculator can distinguish from zero are $\pm 10^{-499}$. If you enter a number between these values, the calculator will display 0 upon entry. Likewise, if the result of calculation lies between these two values, the result will be displayed as zero. Entering numbers beyond the maximum range above will result in an error message "INVALID DATA"; clearing the error message returns you to the previous entry for correction.

Making Numbers Negative

✓ The +/– key changes the sign of a number.

- To key in a negative number, type the number, then press +/–,
- In ALG mode, you may press +/– key before or after typing the number.
- To change the sign of a number that was entered previously, just press +/–. (If the number has an exponent, +/– affects only the *mantissa* — the *non–*exponent part of the number.)

Exponents of Ten

Exponents in the Display

Numbers with explicit powers of ten (such as 4.2×10^{-5}) are displayed with an **E** preceding the exponent of 10. Thus 4.2×10^{-5} is entered and displayed as 4.2**E**-5.

A number whose magnitude is too large or too small for the display format will automatically be displayed in exponential form.

For example, in FIX 4 format for four decimal places, observe the effect of the following keystrokes:

Keys:	Display:	Description:
0 · 0 0 0 0 6 2	0.000062_	Shows number being entered.
ENTER	0.0001	Rounds number to fit the display format.
0 · 0 0 0 0 4 2 ENTER	4.2000E-5	Automatically uses scientific notation because otherwise no significant digits would appear.

Keying in Powers of Ten

The E key is used to enter powers of ten quickly. For example, instead of entering one million as 1000000 you can simply enter 1 E 6. The following example illustrates the process as well as how the calculator displays the result.

Example:

Suppose you want to enter Planck's constant: 6.6261×10^{-34}

Keys:	Display:	Description
6 · 6 2 6	0	Enter the mantissa
1	6.6261_	
E	0	Equivalent to $\times 10^{x}$
	6.621E_	
3 4 +/− ENTER	6.621E‾34	Enter the exponent
	6.621E-34	

For a power of ten without a multiplier, as in the example of one million above, press the 1 E key followed by the desired exponent of ten.

Other Exponent Functions

To calculate an exponent of ten (the base 10 antilogarithm), use ⬑ 10^x. To calculate the result of *any* number raised to a power (exponentiation), use y^x (see chapter 4).

Understanding Entry Cursor

As you key in a number, the cursor (_) appears and blinks in the display. The cursor shows you where the next digit will go; it therefore indicates that the number is not complete.

Keys:	Display:	Description:
1 2 3	`123_`	Entry *not* terminated: the number is not complete.

If you *execute a function* to calculate a *result*, the cursor disappears because the number is complete —entry has been terminated.

✓ √x	`11.0905`	Entry is terminated.

Pressing ENTER terminates entry. To separate two numbers, key in the first number, press ENTER to terminate entry, and then key in the second number

✓ 1 2 3 ENTER	`123.0000`	A completed number.
✓ 4 +	`127.0000`	Another completed number.

If entry is *not* terminated (if the cursor is present), ← backspaces to erase the last digit. If entry is terminated (no cursor), ← acts like C and clears the entire number. Try it!

Range of Numbers and OVERFLOW

The smallest number available on the calculator is $-9.99999999999 \times 10^{499}$,while the largest number is $9.99999999999 \times 10^{499}$.

- If a calculation produces a result that exceeds the largest possible number, $-9.99999999999 \times 10^{499}$ or $9.99999999999 \times 10^{499}$ is returned, and the warning message `OVERFLOW` appears.

Performing Arithmetic Calculations

The HP 35s can operate in either RPN mode or in Algebraic mode (ALG). These modes affect how expressions are entered. The following sections illustrate the entry differences for single argument (or unary) and two argument (or binary) operations.

Single Argument or Unary Operations

Some of the numerical operations of the HP 35s require a single number for input, such as 1/x, x^2, LN and SIN. These single argument operations are entered differently, depending on whether the calculator is in RPN or ALG mode. In RPN mode, the number is entered first and then the operation is applied. If the ENTER key is pressed after the number is entered, then the number appears in line 1 and the result is shown in line 2. Otherwise, just the result is displayed in line 2 and line 1 is unchanged. In ALG mode, the operator is pressed first and the display shows the function, followed by a set of parentheses. The number is entered between the parentheses and then the ENTER key is pressed. The expression is displayed in line 1 and the result is shown in line 2. The following examples illustrate the differences.

Example:

Calculate 3.4^2, first in RPN mode and then in ALG mode.

Keys:	Display:	Description:
MODE 5 (5RPN)		Enter RPN mode (if necessary)
3 . 4	0 3.4	Enter the number
↱ x^2	0 11.56	Press the square operator
MODE 4 (4ALG)		Switch to ALG mode
↱ x^2	SQ()	Enter the square operation
3 . 4	SQ(3.4)	Insert the number between the parentheses
ENTER	SQ(3.4) 11.56	Press the Enter key to see the result

In the example, the square operator is shown on the key as x^2 but displays as SQ(). There are several single argument operators that display differently in ALG mode than they appear on the keyboard (and differently than they appear in RPN mode as well). These operations are listed in the table below.

Key	In RPN,RPN Program	In ALG, Equation, ALG Program
x^2	X^2	SQ()
$\sqrt{x}$	√ x	SQRT()
e^x	e^x	EXP()
10^x	10^x	ALOG()
$1/x$	1/x	INV()

Two Argument or Binary Operations

Two argument operations, such as +, ÷, y^x, and nCr, are also entered differently depending on the mode though the differences are similar to the case for single argument operators. In RPN mode, the first number is entered, then the second number is placed in the x-register and the two argument operation is invoked. In ALG mode, there are two cases, one using traditional infix notation and another taking a more function-oriented approach. The following examples illustrate the differences.

Example

Calculate 2+3 and ${}_6C_4$, first in RPN mode and then in ALG mode.

Keys:	Display:	Description:
MODE 5 (5RPN)		Switch to RPN mode (if necessary)
2 ENTER 3	2 3_	Enter 2, then place 3 in the x-register. Note the flashing cursor after the 3; don't press Enter!
+	0 5	Press the addition key to see the result.
6 ENTER 4	6 4_	Enter 6, then place 4 in the x-register.
↰ nCr	5 15	Press the combinations key to see the result.
MODE 4 (4ALG)		Switch to ALG mode
2 + 3 ENTER	2+3 5	Expression and result are both shown.
↰ nCr	nCr(,)	Enter the combination function.
6 > 4	nCr(6,4)	Enter the 6, then move the edit cursor past the comma and enter the 4.
ENTER	nCr(6,4) 15	Press Enter to see the result.

In ALG mode, the infix operators are +, − , ×, ÷, and y^x. The other two argument operations use function notation of the form f(x,y), where x and y are the first and second operands in order. In RPN mode, the operands for two argument operations are entered in the order Y, then X on the stack. That is, y is the value in the y-register and x is the value in the x-register.

The x^{th} root of y ($\sqrt[x]{y}$) is the exception to this rule. For example, to calculate $\sqrt[3]{8}$ in RPN mode, press 8 ENTER 3 ↰ $\sqrt[x]{y}$. In ALG mode, the equivalent operation is keyed in as ↰ $\sqrt[x]{y}$ 3 > 8 ENTER.

As with the single argument operations, some of the two argument operations display differently in RPN mode than in ALG mode. These differences are summarized in the table below.

Key	In RPN, RPN Program	In ALG, Equation, ALG Program
y^x	y^x	^
$\sqrt[x]{y}$	x √ y	XROOT(,)
INT÷	INT÷	IDIV(,)

For commutative operations such as + and ×, the order of the operands does not affect the calculated result. If you mistakenly enter the operand for a noncommutative two argument operation in the wrong order in RPN mode, simply press the x↔y key to exchange the contents in the *x*- and *y*-registers. This is explained in detail in Chapter 2 (see the section entitled *Exchanging the X- and Y-Registers in the Stack*).

Controlling the Display Format

All numbers are stored with 12-digit precision; however, you may control the number of digits used in the display of numbers via the options in the Display menu. Press ↰ DISPLAY to access this menu. The first four options (FIX, SCI, ENG, and ALL) control the number of digits in the display of numbers. During some complicated internal calculations, the calculator uses 15–digit precision for intermediate results. The displayed number is *rounded* according to the display format.

Fixed–Decimal Format (FIX)

FIX format displays a number with up to 11 decimal places (11 digits to the *right* of the "." or "," radix mark) if they fit. After the prompt FIX_, type in the number of decimal places to be displayed. For 10 or 11 places, press · 0 or · 1.

For example, in the number 123,456.7089, the "7", "0", "8", and "9" are the decimal digits you see when the calculator is set to FIX 4 display mode.

Any number that is too large (10^{11}) or too small (10^{-11}) to display in the current decimal–place setting will automatically be displayed in scientific format.

Scientific Format (SCI)

SCI format displays a number in scientific notation (one digit before the "." or "," radix mark) with up to 11 decimal places and up to three digits in the exponent. After the prompt, SCI_, type in the number of decimal places to be displayed. For 10 or 11 places, press [.] [0] or [.] [1]. (The mantissa part of the number will always be less than 10.)

For example, in the number 1.2346E5, the "2", "3", "4", and "6" are the decimal digits you see when the calculator is set to SCI 4 display mode. The "5" following the "E" is the exponent of 10: 1.2346 × 10^5.

If you enter or calculate a number that has more than 12 digits, the additional precision is not maintained.

Engineering Format (ENG)

ENG format displays a number in a manner similar to scientific notation, except that the exponent is a multiple of three (there can be up to three digits before the "." or "," radix mark). This format is most useful for scientific and engineering calculations that use units specified in multiples of 10^3 (such as micro–, milli–, and kilo–units.)

After the prompt, ENG_, type in the number of digits you want after the first significant digit. For 10 or 11 places, press [.] [0] or [.] [1].

For example, in the number 123.46E3, the "2", "3", "4", and "6" are the significant digits after the first significant digit you see when the calculator is set to ENG 4 display mode. The "3" following the "E" is the (multiple of 3) exponent of 10: 123.46 x 10^3.,

Pressing [↰] [←ENG] or [↰] [ENG→] will cause the exponent display for the number being displayed to change in multiples of 3, with the mantissa adjusted accordingly.

Example:

This example illustrates the behavior of the Engineering format using the number 12.346E4. It also shows the use of the [↰] [←ENG] and [↰] [ENG→] functions. This example uses RPN mode.

Keys:	Display:	Description:
[↰] [DISPLAY] [3] (3ENG)	ENG_	Choose Engineering format
[4]	0.0000E0 0.0000E0	Enter 4 (for 4 significant digits after the 1st)
[1] [2] [.] [3] [4] [6] [E] [4] [ENTER]	123.46E3 123.46E3	Enter 12.346E4
[↰] [←ENG] or	123.46E3	
[↰] [ENG→]	123.46E3	
[↰] [←ENG]	123.46E3 0.12346E6	Increases the exponent by 3
[↰] [ENG→]	123.46E3 123.46E3	Decreases the exponent by 3

ALL Format (ALL)

The All format is the default format, displaying numbers with up to 12 digit precision. If all the digits don't fit in the display, the number is automatically displayed in scientific format.

Periods and Commas in Numbers (.) (,)

The HP 35s uses both periods and commas to make numbers easier to read. You can select either the period or the comma as the decimal point (radix). In addition, you can choose whether or not to separate digits into groups of three using thousand separators. The following example illustrates the options.

Example

Enter the number 12,345,678.90 and change the decimal point to the comma. Then choose to have no thousand separator. Finally, return to the default settings. This example uses RPN mode.

Keys:	Display:	Description:
⬑ DISPLAY 4 (4ALL)		Select full floating point precision (ALL format)
1 2 3 4 5 6 7 8 · 9 ENTER	12,345,678.9 12,345,678.9	The default format uses the comma as the thousand separator and the period as the radix.
⬑ DISPLAY 6 (6,)	12.345.678,9 12.345.678,9	Change to use the comma for the radix. Note that the thousand separator automatically changes to the period.
⬑ DISPLAY 8 (81000)	12345678,9 12345678,9	Change to having no comma separator.
⬑ DISPLAY 5 (5.)	12,345,678.9	Return to the default format.
⬑ DISPLAY 7 (71,000)	12,345,678.9	

Complex number display format (xiy, x+yi, rθa)

Complex numbers can be displayed in a number of formats: xiy, x+yi, and rθa, although x+yi is only available in ALG mode. In the example below, the complex number 3+4i is displayed in all three ways.

Example

Display the complex number 3+4i in each of the different formats.

Keys:	Display:	Description:
MODE 4 (4ALG)		Enable ALG mode
3 i 4 ENTER	3i4 3i4	Enter the complex number. It displays as 3i4, the default format.
⬑ DISPLAY ·	3i4	Change to x+yi format.
1 (11x+yi)	3+4i	
⬑ DISPLAY ·	3i4	Change to rθa format. The radius is
0 (10rθa) or	5θ53.1301023542	5 and the angle is approximately
⬑ DISPLAY ^		53.13°.
^ > ENTER		

SHOWing Full 12–Digit Precision

Changing the number of displayed decimal places affects what you see, but it does not affect the internal representation of numbers. Any number stored internally always has 12 digits.

For example, in the number 14.8745632019, you see only "14.8746" when the display mode is set to FIX 4, but the last six digits ("632019") are present internally in the calculator.

To temporarily display a number in full precision, press ⬑ SHOW. This shows you the *mantissa* (but no exponent) of the number for as long as you hold down SHOW.

Keys:	Display:	Description:
✓ 4 5 ENTER 1 · 3 ×	58.5000	Four decimal places displayed.
⬑ DISPLAY 2 (2SCI) 2	5.85E1	Scientific format: two decimal places and an exponent.
⬑ DISPLAY 3 (3ENG) 2	58.5E0	Engineering format.

↰ DISPLAY 4 (4ALL)	58.5	All significant digits; trailing zeros dropped.
↰ DISPLAY 1 (1FIX)	58.5000	Four decimal places, no exponent.
4		
✓ 1/x	0.0171	Reciprocal of 58.5.
↰ SHOW (hold)	170940170940	Shows full precision until you release SHOW

Fractions

The HP 35s allows you to enter and operate on fractions, displaying them as either decimals or fractions. The HP 35s displays fractions in the form a b/c, where a is an integer and both b and c are counting numbers. In addition, b is such that 0≤b<c and c is such that 1<c≤4095.

Entering Fractions

Fractions can be entered onto the stack at any time:

1. Key in the integer part of the number and press ⊡. (The first ⊡ separates the integer part of the number from its fractional part.)
2. Key in the fraction numerator and press ⊡ again. The second ⊡ separates the numerator from the denominator.
3. Key in the denominator, then press ENTER or a function key to terminate digit entry. The number or result is formatted according to the current display format.

The *a b/c* symbol under the ⊡ key is a reminder that the ⊡ key is used twice for fraction entry.

The following example illustrates the entry and display of fractions.

Example

Enter the mixed numeral 12 3/8 and display it in fraction and decimal forms. Then enter ¾ and add it to 12 3/8. This example uses RPN mode.

Keys:	Display:	Description:
1 2 · 3	0 12.3	The decimal point is interpreted in the normal way.
· 8	0.0000 12 3/8_	When · is pressed the 2nd time, the display switches to fraction mode.
ENTER	12.3750 12.3750	Upon entry, the number is displayed using the current display format.
↱ FDISP	12 3/8 12 3/8	Switch to fraction display mode.
· 3 · 4	12 3/8 0 3/4_	Enter ¾. Note you start with · because there is no integer part (you could type in 0 ¾).
+	0 13 1/8	Add ¾ to 12 3/8.
↱ FDISP	0 13.1250	Switch back to the current display mode.

Refer to chapter 5, "Fractions," for more information about using fractions.

Messages

The calculator responds to error conditions by displaying the ⚠ annunciator. Usually, a message will accompany the error annunciator as well.

- To clear a message, press C or ←; in RPN mode, you will return to the stack as it was before the error. In ALG mode, you will return to the last expression with the edit cursor at the position of the error so that you can correct it.

- Any other key also clears the message, though the key function is not entered

If no message is displayed, but the ⚠ annunciator appears, then you have pressed an inactive or invalid key. For example, pressing [·] [·] will display ⚠ because the second decimal point has no meaning in this context.
All displayed messages are explained in appendix F, "Messages".

Calculator Memory

The HP 35s has 30KB of memory in which you can store any combination of data (variables, equations, or program lines).

Checking Available Memory

Pressing [↰] [MEM] displays the following menu:

```
1VAR  2 PGM
nnn    mm,mmm
```

Where

nnn is the amount of used indirect variables.

mm,mmm is the number of bytes of memory available.

Pressing the [1] (1VAR) displays the catalog of direct variables (see "Reviewing Variables in the VAR Catalog" in chapter 3). Pressing the [2] (2PGM) displays the catalog of programs.

1. To enter the catalog of variables, press [1] (1VAR); to enter the catalog of programs, press [2] (2PGM).
2. To review the catalogs, press [∨] or [∧].
3. To delete a variable or a program, press [↱] [CLEAR] while viewing it in its catalog.
4. To exit the catalog, press [C].

Clearing All of Memory

Clearing all of memory erases all numbers, equations, and programs you've stored. It does not affect mode and format settings. (To clear settings as well as data, see "Clearing Memory" in appendix B.)

To clear all of memory:

1. Press 4 (4ALL). You will then see the confirmation prompt CLR ALL? Y N, which safeguards against the unintentional clearing of memory.
2. Press < **(Y)** ENTER.

2

RPN: The Automatic Memory Stack

This chapter explains how calculations take place in the automatic memory stack in RPN mode. *You do not need to read and understand this material to use the calculator*, but understanding the material will greatly enhance your use of the calculator, especially when programming.

In part 2, "Programming", you will learn how the stack can help you to manipulate and organize data for programs.

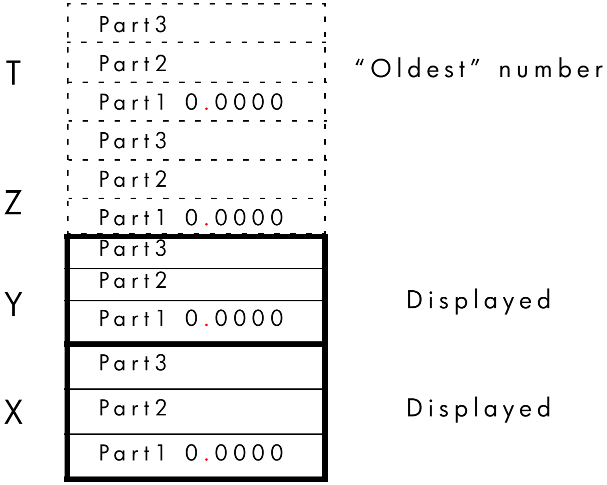
What the Stack Is

Automatic storage of intermediate results is the reason that the HP 35s easily processes complex calculations, and does so without parentheses. The key to automatic storage is the *automatic, RPN memory stack*.

HP's operating logic is based on an unambiguous, *parentheses–free* mathematical logic known as "Polish Notation," developed by the Polish logician Jan Łukasiewicz (1878–1956).

While conventional algebraic notation places the operators *between* the relevant numbers or variables, Łukasiewicz's notation places them *before* the numbers or variables. For optimal efficiency of the stack, we have modified that notation to specify the operators *after* the numbers. Hence the term *Reverse Polish Notation*, or RPN.

The stack consists of four storage locations, called *registers*, which are "stacked" on top of each other. These registers — labeled X, Y, Z, and T — store and manipulate four current numbers. The "oldest" number is stored in the T– (*top*) register. The stack is the work area for calculations.



The most "recent" number is in the X–register: *this is the number you see in the second line of the display.*

Every register is separated into three parts:

- A real number or a 1-D vector will occupy part 1; part 2 and part 3 will be null in this case.
- A complex number or a 2-D vector will occupy part 1 and part 2; part 3 will be null in this case.
- A 3-D vector will occupy part 1, part 2, and part 3.

In programming, the stack is used to perform calculations, to temporarily store intermediate results, to pass stored data (variables) among programs and subroutines, to accept input, and to deliver output.

The X and Y–Registers are in the Display

The X and Y–Registers are what you see *except* when a menu, a message, an equation line ,or a program line is being displayed. You might have noticed that several function names include an *x* or *y*.

This is no coincidence: these letters refer to the X– and Y–registers. For example, ↰ 10^x raises ten to the power of the number in the X–register.

Clearing the X–Register

Pressing ↱ CLEAR 1 (x) *always* clears the X–register to zero; it is also used to program this instruction. The C key, in contrast, is context–sensitive. It either clears or cancels the current display, depending on the situation: it acts like ↱ CLEAR 1 (x) only when the X–register is displayed. ← also acts like ↱ CLEAR 1 (x) when the X–register is displayed *and* digit entry is terminated (no cursor present).

Reviewing the Stack

R↓ (Roll Down)

The R↓ *(roll down) key* lets you review the entire contents of the stack by "rolling" the contents downward, one register at a time. You can see the numbers as they roll through the *x*- and *y*-registers.

Suppose the stack is filled with 1, 2, 3, 4. (press 1 ENTER 2 ENTER 3 ENTER 4) Pressing R↓ four times rolls the numbers all the way around and back to where they started:

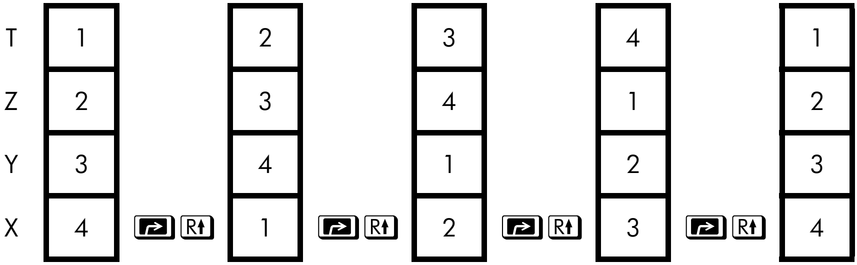
T	1		4		3		2		1
Z	2		1		4		3		2
Y	3		2		1		4		3
X	4	R↓	3	R↓	2	R↓	1	R↓	4

What was in the X–register *rotates* into the T–register, the contents of the T–register rotate into the Z–register, etc. Notice that only the *contents* of the registers are rolled — the registers themselves maintain their positions, and only the X– and Y–register's contents are displayed.

R↑ (Roll Up)

The ↱ R↑ (*roll up*) key has a similar function to R↓ except that it "rolls" the stack contents upward, one register at a time.

The contents of the X–register rotate into the Y–register; what was in the T–register rotates into the X–register, and so on.



Exchanging the X– and Y–Registers in the Stack

Another key that manipulates the stack contents is x↔y (*x exchange y*). This key swaps the contents of the X– and Y–registers without affecting the rest of the stack. Pressing x↔y twice restores the original order of the X– and Y–register contents.

The x↔y function is used primarily to swap the order of numbers in a calculation. For example, one way to calculate 9 ÷ (13 × 8):
Press 1 3 ENTER 8 × 9 x↔y ÷.
The keystrokes to calculate this expression from *left–to–right* are:
9 ENTER 1 3 ENTER 8 × ÷.

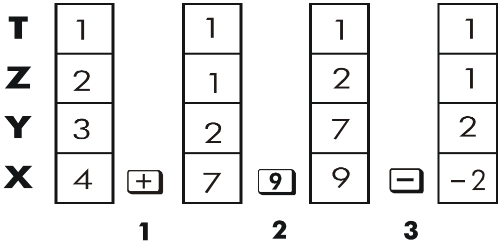
Note Understand that there are no more than four numbers in the stack at any given time – the contents of the T-register (the top register) will be lost whenever a fifth number is entered.

Arithmetic – How the Stack Does It

The contents of the stack move up and down automatically as new numbers enter the X–register *(lifting the stack)* and as operators combine two numbers in the X– and Y–registers to produce one new number in the X–register (*dropping the stack*).

Suppose the stack is filled with the numbers 1, 2, 3, and 4. See how the stack drops and lifts its contents while calculating

3+4−9



1. The stack "drops" its contents. The T–(top) register *replicates* its contents.

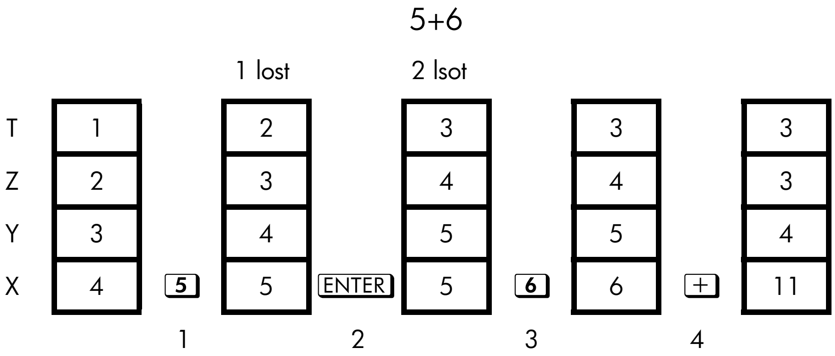
2. The stack "lifts" its contents. The T–register's contents are *lost.*

3. The stack drops.

- Notice that when the stack lifts, it replaces the contents of the T– (top) register with the contents of the Z–register, and that the *former* contents of the T–register are lost. You can see, therefore, that the stack's memory is limited to four numbers.
- Because of the automatic movements of the stack, you do *not* need to clear the X–register before doing a new calculation.
- Most functions prepare the stack to lift its contents *when the next number enters the X–register.* See appendix B for lists of functions that disable stack lift.

How ENTER Works

You know that ENTER separates two numbers keyed in one after the other. In terms of the stack, how does it do this? Suppose the stack is again filled with 1, 2, 3, and 4. Now enter and add two new numbers:



1. Lifts the stack.
2. Lifts the stack and replicates the X–register.
3. Does *not* lift the stack.
4. Drops the stack and replicates the T–register.

ENTER replicates the contents of the X–register into the Y–register. The next number you key in (or recall) *writes over* the copy of the first number left in the X–register. The effect is simply to separate two sequentially entered numbers.

You can use the replicating effect of ENTER to clear the stack quickly: press 0 ENTER ENTER ENTER. All stack registers now contain zero. Note, however, that you don't *need* to clear the stack before doing calculations.

Using a Number Twice in a Row

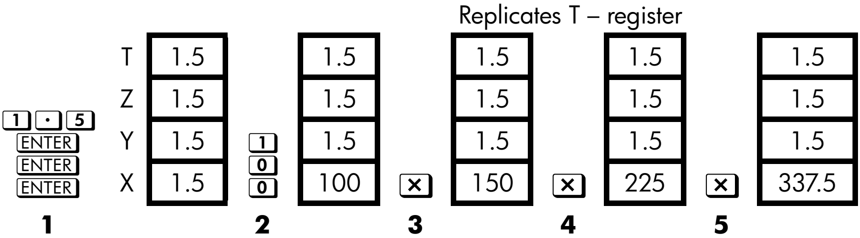
You can use the replicating feature of ENTER to other advantages. To add a number to itself, press ENTER +.

Filling the stack with a constant

The replicating effect of ENTER together with the replicating effect of stack drop (from T into Z) allows you to fill the stack with a numeric constant for calculations.

Example:

Given bacterial culture with a constant growth rate of 50% per day, how large would a population of 100 be at the end of 3 days?



Replicates T – register

	1	2	3	4	5
	1 · 5 ENTER ENTER ENTER	1 0 0	×	×	×
T	1.5	1.5	1.5	1.5	1.5
Z	1.5	1.5	1.5	1.5	1.5
Y	1.5	1.5	1.5	1.5	1.5
X	1.5	100	150	225	337.5

1. Fills the stack with the growth rate.
2. Keys in the initial population.
3. Calculates the population after 1 day.
4. Calculates the population after 2 days.
5. Calculates the population after 3 days.

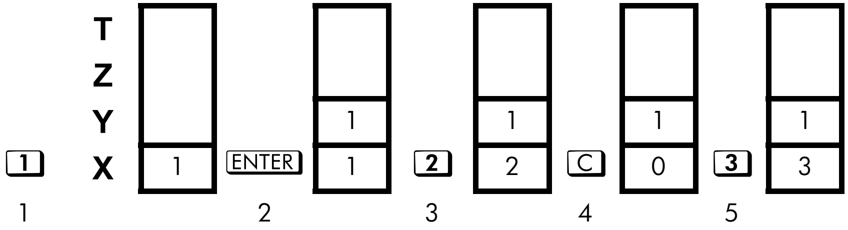
How to Clear the Stack

Clearing the X–register puts a zero in the X–register. The next number you key in (or recall) *writes* over this zero.

There are four ways to clear the contents of the X–register, that is, to clear *x*:

1. Press C
2. Press ←
3. Press ↱ CLEAR 1 (1x) (Mainly used during program entry.)
4. Press ↱ CLEAR 5 (5STK) to clear the X-, Y-, Z-, and T-registers to zero.

For example, if you intended to enter 1 and 3 but mistakenly entered 1 and 2, this is what you should do to correct your error:



1. Lifts the stack

2. Lifts the stack and replicates the X–register.

3. Overwrites the X–register.

4. Clears *x* by overwriting it with zero.

5. Overwrites *x* (replaces the zero.)

The LAST X Register

The LAST X register is a companion to the stack: it holds the number that was in the X–register before the last numeric function was executed. (A numeric function is an operation that produces a result from another number or numbers, such as √x.) Pressing ⮫ LAST*x* returns this value into the X–register.

This ability to retrieve the "last x" has two main uses:

1. Correcting errors.

2. Reusing a number in a calculation.

See appendix B for a comprehensive list of the functions that save *x* in the LAST X register.

Correcting Mistakes with LAST X

Wrong Single Argument Function

If you execute the wrong single argument function, use ↱ LASTx to retrieve the number so you can execute the correct function. (Press C *first* if you want to clear the incorrect result from the stack.)

Since ↱ % and ↰ %CHG don't cause the stack to drop, you can recover from these functions in the same manner as from single argument functions.

Example:

Suppose that you had just computed ln 4.7839 × (3.879 × 10^5) and wanted to find its square root, but pressed e^x by mistake. You don't have to start over! To find the correct result, press ↱ LASTx $\sqrt{x}$.

Mistakes with Two Argument Functions

If you make a mistake with a two argument operation (such as +, y^x, or nCr), you can correct it by using ↱ LASTx and the inverse of the two argument operation.

1. Press ↱ LASTx to recover the second number (*x* just before the operation).
2. Execute the inverse operation. This returns the number that was originally first. The second number is still in the LAST X register. Then:
   - If you had used the *wrong function*, press ↱ LASTx again to restore the original stack contents. Now execute the correct function.
   - If you had used the *wrong second number*, key in the correct one and execute the function.

If you had used the *wrong first number*, key in the correct first number, press ↱ LASTx to recover the second number, and execute the function again. (Press C *first* if you want to clear the incorrect result from the stack.)

Example:

Suppose you made a mistake while calculating

$$16 \times 19 = 304$$

There are three kinds of mistakes you could have made:

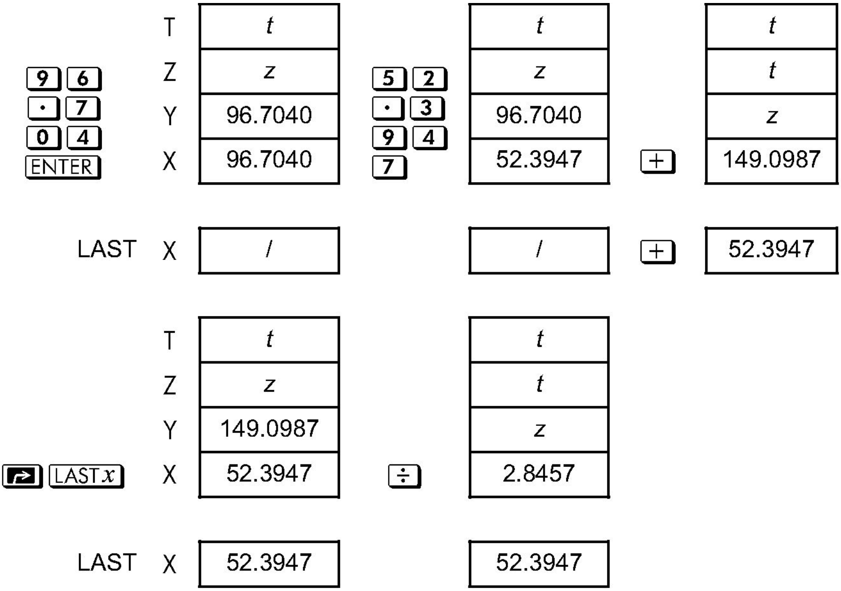
Wrong Calculation:	Mistake:	Correction:
1 6 ENTER 1 9 −	Wrong function	↱ LASTx + ↱ LASTx ×
1 5 ENTER 1 9 ×	Wrong first number	1 6 ↱ LASTx ×
1 6 ENTER 1 8 ×	Wrong second number	↱ LASTx ÷ 1 9 ×

Reusing Numbers with LAST X

You can use ↱ LASTx to reuse a number (such as a constant) in a calculation. Remember to enter the constant second, just before executing the arithmetic operation, so that the constant is the last number in the X–register, and therefore can be saved and retrieved with ↱ LASTx.

Example:

$$\text{Calculate } \frac{96.704+52.3947}{52.3947}$$



Keys:	Display:	Description:
9 6 . 7 0 4 ENTER	`96.7040`	Enters first number.
5 2 . 3 9 4 7 +	`149.0987`	Intermediate result.
↱ LASTx	`52.3947`	Brings back display from before +.
÷	`2.8457`	Final result.

Example:

Two close stellar neighbors of Earth are Rigel Centaurus (4.3 light–years away) and Sirius (8.7 light–years away). Use *c*, the speed of light (9.5 × 10^{15} meters per year) to convert the distances from the Earth to these stars into meters:

To Rigel Centaurus: 4.3 yr × (9.5 × 10^{15} m/yr).
To Sirius: 8.7 yr × (9.5 × 10^{15} m/yr).

Keys:	Display:	Description:
4 · 3 ENTER	4.3000	Light–years to Rigel Centaurus.
9 · 5 E 1 5	9.5E15_	Speed of light, *c*.
×	4.0850E16	Meters to R. Centaurus.
8 · 7 ↱ LASTx	9.5000E15	Retrieves *c*.
×	8.2650E16	Meters to Sirius.

Chain Calculations in RPN Mode

In RPN mode, the automatic lifting and dropping of the stack's contents let you retain intermediate results without storing or reentering them, and without using parentheses.

Work from the Parentheses Out

For example, evaluate (12 + 3) × 7.

If you were working out this problem on paper, you would first calculate the intermediate result of (12 + 3) ...

$$(12 + 3) = 15$$

... then you would multiply the intermediate result by 7:

$$(15) \times 7 = 105$$

Evaluate the expression in the same way on the HP 35s, starting inside the parentheses.

Keys:	Display:	Description:
1 2 ENTER 3 +	15.0000	Calculates the intermediate result first.

You don't need to press ENTER to save this intermediate result before proceeding; since it is a *calculated* result, it is saved automatically.

Keys:	Display:	Description:
7 ×	105.0000	Pressing the function key produces the answer. This result can be used in further calculations.

Now study the following examples. Remember that you need to press ENTER only to separate sequentially-entered numbers, such as at the beginning of an expression. The operations themselves (+, −, etc.) separate subsequent numbers and save intermediate results. The last result saved is the first one retrieved as needed to carry out the calculation.

Calculate 2 ÷ (3 + 10):

Keys:	Display:	Description:
3 ENTER 1 0 +	13.0000	Calculates (3 + 10) first.
2 x◂▸y ÷	0.1538	Puts 2 *before* 13 so the division is correct: 2 ÷ 13.

Calculate 4 ÷ [14 + (7 × 3) – 2] :

Keys:	Display:	Description:
7 ENTER 3 ×	21.0000	Calculates (7 × 3).
1 4 + 2 −	33.0000	Calculates denominator.
4 x◂▸y	33.0000	Puts 4 *before* 33 in preparation for division.
÷	0.1212	Calculates 4 ÷ 33, the answer.

Problems that have multiple parentheses can be solved in the same manner using the automatic storage of intermediate results. For example, to solve (3 + 4) × (5 + 6) on paper, you would first calculate the quantity (3 + 4). Then you would calculate (5 + 6). Finally, you would multiply the two intermediate results to get the answer.

Work through the problem the same way with the HP 35s, except that you don't have to write down intermediate answers—the calculator remembers them for you.

Keys:	Display:	Description:
3 ENTER 4 +	7.0000	First adds (3+4)
5 ENTER 6 +	11.0000	Then adds (5+6)

×	`77.0000`	Then multiplies the intermediate answers together for the final answer.

Exercises

Calculate:

$$\frac{\sqrt{\text{[redacted]}05x5)}}{0.05} = 181.0000$$

Solution:

1 6 · 3 8 0 5 ENTER 5 × √x · 0 5 ÷

Calculate:

$$\sqrt{\text{[redacted]}\times(4+5)]} + \sqrt{\text{[redacted]}\times(8+9)]} = 21.5743$$

Solution:

2 ENTER 3 + 4 ENTER 5 + × √x 6 ENTER 7 + 8 ENTER
9 + × √x +

Calculate:

(10 – 5) ÷ [(17 – 12) × 4] = 0.2500

Solution:

1 7 ENTER 1 2 – 4 × 1 0 ENTER 5 – x◂▸y ÷
or
1 0 ENTER 5 – 1 7 ENTER 1 2 – 4 × ÷

Order of Calculation

We recommend solving chain calculations by working from the innermost parentheses outward. However, you can also choose to work problems in a left–to–right order.

For example, you have already calculated:

$$4 \div [14 + (7 \times 3) - 2]$$

by starting with the innermost parentheses (7 × 3) and working outward, just as you would with pencil and paper. The keystrokes were 7 ENTER 3 × 1 4 + 2 − 4 x↔y ÷.

If you work the problem from left–to–right, press

4 ENTER 1 4 ENTER 7 ENTER 3 × + 2 − ÷.

This method takes one additional keystroke. Notice that the first intermediate result is still the innermost parentheses (7 × 3). The advantage to working a problem left–to–right is that you don't have to use x↔y to reposition operands for *noncommutative* functions (− and ÷).

However, the first method (starting with the innermost parentheses) is often preferred because:

- It takes fewer keystrokes.
- It requires fewer registers in the stack.

Note

When using the *left–to–right* method, be sure that no more than *four* intermediate numbers (or results) will be needed at one time (the stack can hold no more than four numbers).

The above example, when solved *left–to–right,* needed all registers in the stack at one point:

Keys:	Display:	Description:
4 ENTER 1 4 ENTER	`14.0000`	Saves 4 and 14 as intermediate numbers in the stack.
7 ENTER 3	`3_`	At this point the stack is full with numbers for this calculation.
×	`21.0000`	Intermediate result.
+	`35.0000`	Intermediate result.

[2] [–]	33.0000	Intermediate result.
[÷]	0.1212	Final result.

More Exercises

Practice using RPN by working through the following problems:

Calculate:

(14 + 12) × (18 – 12) ÷ (9 – 7) = 78.0000

A Solution:

[1] [4] [ENTER] [1] [2] [+] [1] [8] [ENTER] [1] [2] [–] [×] [9] [ENTER] [7] [–] [÷]

Calculate:

$23^2 - (13 \times 9) + 1/7 = 412.1429$

A Solution:

[2] [3] [↱] [x^2] [1] [3] [ENTER] [9] [×] [–] [7] [$1/x$] [+]

Calculate:

$\sqrt{\text{[redacted]} \div (12.5 - 0.7^3)} = 0.5961$

Solution:

[5] [·] [4] [ENTER] [·] [8] [×] [·] [7] [ENTER] [3] [y^x] [1] [2] [·] [5] [x◂▸y] [–] [÷] [$\sqrt{x}$]

or

[5] [·] [4] [ENTER] [·] [8] [×] [1] [2] [·] [5] [ENTER] [·] [7] [ENTER] [3] [y^x] [–] [÷] [$\sqrt{x}$]

Calculate:

$$\sqrt{\frac{8.33 \times (4 - 5.2) \div [(8.33 - 7.46) \times 0.32]}{4.3 \times (3.15 - 2.75) - (1.71 \times 2.01)}} = 4.5728$$

A Solution:

4 ENTER 5 · 2 − 8 · 3 3 × ↱ LASTx 7 · 4 6 −
0 · 3 2 × ÷ 3 · 1 5 ENTER 2 · 7 5 − 4 · 3 ×
1 · 7 1 ENTER 2 · 0 1 × − ÷ $\sqrt{x}$

3

Storing Data into Variables

The HP 35s has 30 KB of memory, in which you can store numbers, equations, and programs. Numbers are stored in locations called *variables*, each named with a letter from *A* through *Z*. (You can choose the letter to remind you of what is stored there, such as *B* for *bank balance* and *C* for the speed of light.)

Example:

This example shows you how to store the value 3 in the variable A, first in RPN mode and then in ALG mode.

Keys:	Display:	Description:
MODE 5 (5 RPN)		Switch to RPN mode (if necessary)
3	0.0000 3_	Enter the value (3)
↱ STO	STO_	The Store command prompts for a letter; note the A…Z annunciator.
A	0.0000 3.0000	The value 3 is stored in A and returned to the stack.
MODE 4 (4 ALG)	3.0000	Switch to ALG mode (if necessary)
3 ↱ STO A	3▶A_	Again, the Store command prompts for a letter and the A…Z annunciator appears.
ENTER	3▶A 3.0000	The value 3 is stored in A and the result is placed in line 2.

In ALG mode, you can store an expression into a variable; in this case, the value of the expression is stored in the variable rather than the expression itself.

Example:

Keys:	Display:	Description:
1 + 3 ÷ 4	1+3÷4▮G	Enter the expression, then
↱ STO G ENTER	1.7500	proceed as in the previous
		example.

Each pink letter is associated with a key and a unique variable. (The **A..Z** annunciator in the display confirms this.)

Note that the variables, *X*, *Y*, *Z* and *T* are *different* storage locations from the X–register, Y–register, Z–register, and T–register in the stack.

Storing and Recalling Numbers

Numbers and vectors are stored into, and recalled from, lettered variables by means of the Store (↱ STO) and Recall (RCL) commands. Numbers may be real or complex, decimal or fraction, base 10 or other as supported by the HP 35s.

To store a copy of a displayed number (X–register) to a direct variable:

Press ↱ STO *letter–key* ENTER.

To recall a copy of a number from a direct variable to the display:

Press RCL *letter–key* ENTER.

Example: Storing Numbers.

Store Avogadro's number (approximately 6.0221 × 10^{23}) in *A*.

Keys:	Display:	Description:
6 . 0 2 2 1 E 2 3	6.0221E23_	Avogadro's number.
↱ STO A	6.0221E23▶A_	"▶" prompts for variable.
ENTER	6.0221E23▶A 6.0221E23	Stores a copy of Avogadro's number in *A*. This also terminates digit entry .
C	_	Clears the number in the display.
RCL	**A..Z**	The **A..Z** annunciator Turns on
A ENTER	A= 6.0221E23	Copies Avogadro's number from *A* the display.

To recall the value stored in a variable, use the Recall command. The display of this command differs slightly from RPN to ALG mode, as the following example illustrates.

Example:

In this example, we recall the value of 1.75 that we stored in the variable G in the last example. This example assumes the HP 35s is still in ALG mode at the start.

Keys:	Display:	Description:
RCL G ENTER	G 1.7500	Pressing RCL simply activates A…Z mode; no command is pasted into line 1.

In ALG mode, Recall can be used to paste a variable into an expression in the command line. Suppose we wish to evaluate 15-2×G, with G=1.75 from above.

Keys:	Display:	Description:
1 5 − 2 ×	15-2×G	
RCL G ENTER	11.5000	

We now proceed to switch to RPN mode and recall the value of G.

Keys:	Display:	Description:
MODE 5 (5RPN)		Switch to RPN mode
RCL	RCL _	In RPN mode, RCL pastes the command into the edit line.
G	1.7500 1.7500	No need to press ENTER.

Viewing a Variable

The VIEW command (↰ VIEW) displays the value of a variable without recalling that value to the x-register. The display takes the form Variable=Value. If the number has too many digits to fit into the display, use ↱ > or ↱ < to view the missing digits. To cancel the VIEW display, press ← or C. The VIEW command is most often used in programming but it is useful anytime you want to view a variable's value without affecting the stack.

Using the MEM Catalog

The MEMORY catalog (↰ MEM) provides information about the amount of available memory. The catalog display has the following format:

```
1.VAR   2.PGM

nnn     mm,mmm
```

where *mm,mmm* is the number of bytes of available memory and *nnn* is the amount of used indirect variables.
For more information on indirect variables, see Chapter 14.

The VAR catalog

By default, all direct variables from A to Z contain the value zero. If you store a non-zero value in any direct variable, that variable's value can be viewed in the VAR Catalog (↰ MEM 1(1VAR)).

Example:

In this example, we store 3 in C, 4 in D, and 5 in E. Then we view these variables via the VAR Catalog and clear them as well. This example uses RPN mode.

Keys:	Display:	Description:
↱ CLEAR 2 (2VARS)		Clear all direct variables
3 ↱ STO C 4 ↱ STO D 5 ↱ STO E	4 5	Store 3 in C, 4 in D, and 5 in E.
↰ MEM 1 (1VAR)	C= 3	Enter the VAR catalog.

Note the ⬆ and ⬇ annunciators indicating that the ⌄ and ⌃ keys are active to help you scroll through the catalog; however, if Fraction Display mode is active, the ▲ and ▼ annunciators will not be active to indicate accuracy unless there is only one variable in the catalog. We return to our example, illustrating how to navigate the VAR catalog.

⌄	D= 4	Scroll down to the next direct variable with non-zero value: D=4.
⌄	E= 5	Scroll down once more to see E=5.

While we are in the VAR catalog, let's extend this example to show you how to clear the value of a variable to zero, effectively deleting the current value. We'll delete E.

↱ CLEAR	C= 3	E is no longer in the VAR catalog, as its value is zero. The next variable is C as shown.

Suppose now that you wish to copy the value of C to the stack.

ENTER	5 3	The value of C=3 is copied to the x-register and 5 (from defining E=5 previously) moves to the y-register.

To leave the VAR catalog at any time, press either ← or C. An alternate method to clearing a variable is simply to store the value zero in it. Finally, you can clear all direct variables by pressing ↱ CLEAR 2 (2VARS). If all direct variables have the value zero, then attempting to enter the VAR catalog will display the error message "ALL VARS = 0".

If the value of a variable has too many digits to display completely, you can use › and ‹ to view the missing digits.

Arithmetic with Stored Variables

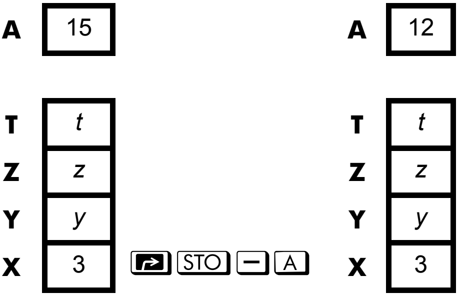
Storage arithmetic and *recall arithmetic* allow you to do calculations with a number stored in a variable *without recalling the variable into the stack.* A calculation uses one number from the X–register and one number from the specified variable.

Storage Arithmetic

Storage arithmetic uses ↱ STO +, ↱ STO −, ↱ STO ×, or ↱ STO ÷ to do arithmetic in the variable itself and to store the result there. It uses the value in the X–register and does not affect the stack.

New value of variable = Previous value of variable {+, –, ×, ÷} *x*.

For example, suppose you want to reduce the value in *A*(15) by the number in the X–register (3, displayed). Press ↱ STO − A. Now A = 12, while 3 is still in the display.



Result: 15 – 3

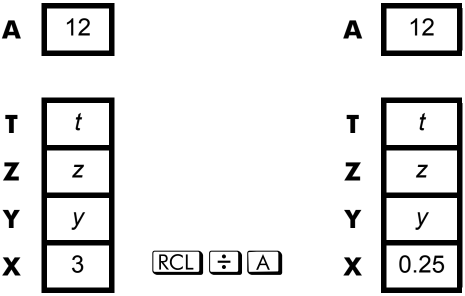
that is, A – x

✓ Recall Arithmetic

Recall arithmetic uses RCL +, RCL −, RCL ×, or RCL ÷ to do arithmetic in the X–register using a recalled number and to leave the result in the display. Only the X–register is affected. The value in the variable remains the same and the result replaces the value in the x-register.

New *x* = Previous *x* {+, –, ×, ÷} Variable

For example, suppose you want to divide the number in the X–register (3, displayed) by the value in *A*(12). Press RCL ÷ A. Now *x* = 0.25, while 12 is still in *A*. Recall arithmetic saves memory in programs: using RCL + *A* (one instruction) uses half as much memory as RCL A, + (two instructions).



Result: 3 ÷ 12

that is, x ÷ 12

✓ **Example:**

Suppose the variables *D, E,* and *F* contain the values 1, 2, and 3. Use storage arithmetic to add 1 to each of those variables.

Keys:	Display:	Description:
1 ↱ STO D 2 ↱ STO E 3 ↱ STO F	1.0000 2.0000 3.0000	Stores the assumed values into the variable.
1 ↱ STO + D ↱ STO + E ↱ STO + F	1.0000	Adds1 to *D, E,* and *F.*
↰ VIEW D	D= 2.0000	Displays the current value of *D.*
↰ VIEW E	E= 3.0000	
↰ VIEW F	F= 4.0000	
←	1.0000	Clears the VIEW display; displays X-register again.

Suppose the variables *D, E,* and *F* contain the values 2, 3, and 4 from the last example. Divide 3 by *D,* multiply it by *E,* and add *F* to the result.

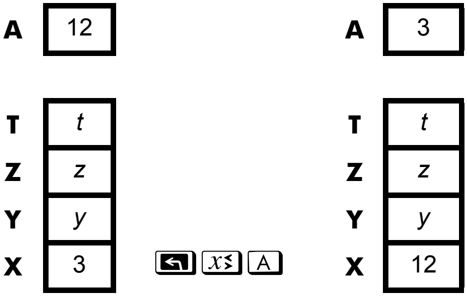
Keys:	Display:	Description:
3 RCL ÷ D	1.5000	Calculates 3 ÷ *D.*
RCL × E	4.5000	3 ÷ *D* × *E.*
RCL + F	8.5000	3 ÷ *D* × *E* + *F.*

Exchanging x with Any Variable

The ↰ x≶ key allows you to exchange the contents of *x* (the displayed X – register) with the contents of any variable. Executing this function does not affect the Y–, Z–, or T–registers.

Example:

Keys:	Display:	Description:
1 2 ↱ STO A ENTER	`12.0000`	Stores 12 in variable A.
3	`3_`	Displays *x*.
↰ x⪋ A	`12.0000`	Exchanges contents of the X–register and variable A.
↰ x⪋ A	`3.0000`	Exchanges contents of the X–register and variable A.



The Variables "I" and "J"

There are two variables that you can access directly: the variables I and J. Although they store values as other variables do, I and J are special in that they can be used to refer to other variables, including the statistical registers, using the (I) and (J) commands. (I) is found on the 0 key, while (J) is on the . key. This is a programming technique called indirect addressing that is covered under “Indirectly Addressing Variables and Labels” in chapter 14.

4

Real–Number Functions

This chapter covers most of the calculator's functions that perform computations on real numbers, including some numeric functions used in programs (such as ABS, the absolute–value function). These functions are addressed in groups, as follows:

- Exponential and logarithmic functions.
- Quotient and Remainder of Divisions.
- Power functions. (y^x and ↰ $\sqrt[x]{y}$)
- Trigonometric functions.
- Hyperbolic functions.
- Percentage functions.
- Physics constants
- Conversion functions for coordinates, angles, and units.
- Probability functions.
- Parts of numbers (number–altering functions).

Arithmetic functions and calculations were covered in chapters 1 and 2. Advanced numeric operations (root–finding, integrating, complex numbers, base conversions, and statistics) are described in later chapters. The examples in this chapter all assume the HP 35s is in RPN mode.

✓ Exponential and Logarithmic Functions

Put the number in the display, then execute the function- there is no need to press ENTER.

To Calculate:	Press:
Natural logarithm (base *e*)	↱ LN
Common logarithm (base 10)	↰ LOG
Natural exponential	↱ e^x
Common exponential (antilogarithm)	↰ 10^x

Quotient and Remainder of Division

You can use ↰ INTG 2(2INT÷)and ↰ INTG 3(3Rmdr) to produce the integer quotient and integer remainder, respectively, from the division of two integers.

1. Key in the first integer.
2. Press ENTER to separate the first number from the second.
3. Key in the second number. (Do *not* press ENTER.)
4. Press the function key.

Example:

To display the quotient and remainder produced by 58 ÷ 9

Keys:	Display:	Description:
5 8 ENTER 9 ↰ INTG 2(2INT÷)	6.0000	Displays the quotient.
5 8 ENTER 9 ↰ INTG 3(3Rmdr)	4.0000	Displays the remainder.

Power Functions

In RPN mode, to calculate a number *y* raised to a power *x*, key in *y* ENTER *x*, then press y^x. (For y>0, x can be any number; for y<0, x must be positive.)

To Calculate:	Press:	Result:
15^2	1 5 ↱ x^2	225.0000
10^6	6 ↰ 10^x	1,000,000.0000
5^4	5 ENTER 4 y^x	625.0000
$2^{-1.4}$	2 ENTER 1 · 4 +/– y^x	0.3789
$(-1.4)^3$	1 · 4 +/– ENTER 3 y^x	-2.7440

In RPN mode, to calculate a root *x* of a number *y* (the *x*th root of *y*), key in *y* ENTER *x*, then press ↰ $\sqrt[x]{y}$. For $y < 0$, *x* must be an integer.

To Calculate:	Press:	Result:
$\sqrt{196}$	1 9 6 $\sqrt{x}$	14.0000
$\sqrt[3]{-125}$	1 2 5 +/– ENTER 3 ↰ $\sqrt[x]{y}$	-5.0000
$\sqrt[4]{625}$	6 2 5 ENTER 4 $\sqrt[x]{y}$	5.0000
$\sqrt[-1.4]{.37893}$	· 3 7 8 9 3 ENTER 1 · 4 +/– ↰ $\sqrt[x]{y}$	2.0000

Trigonometry

Entering π

Press ↰ π to place the first 12 digits of π into the X–register.

(The number displayed depends on the display format.) Because ↰ π is a function that returns an approximation of π to the stack, it is not necessary to press ENTER.

Note that the calculator cannot *exactly* represent π, since π is a transcendental number.

Setting the Angular Mode

The angular mode specifies which unit of measure to assume for angles used in trigonometric functions. The mode does *not* convert numbers already present (see "Conversion Functions" later in this chapter).

360 degrees = 2π radians = 400 grads

To set an angular mode, press MODE. A menu will be displayed from which you can select an option.

Option	Description	Annunciator
DEG	Sets degree mode, which uses decimal degrees rather than hexagesimal degrees (degrees, minutes, seconds)	none
RAD	Sets radian mode	**RAD**
GRAD	Sets gradient mode	**GRAD**

✓ Trigonometric Functions

With *x* in the display:

To Calculate:	Press:
Sine of *x*.	SIN
Cosine of *x*.	COS
Tangent of *x*.	TAN
Arc sine of *x*.	↰ ASIN
Arc cosine of *x*.	↰ ACOS
Arc tangent of *x*.	↰ ATAN

Note Calculations with the irrational number π cannot be expressed *exactly* by the 15–digit internal precision of the calculator. This is particularly noticeable in trigonometry. For example, the calculated sin π (radians) is not zero but –2.0676 × $10^{–13}$, a very small number close to zero.

Example:

Show that cosine (5/7)π radians and cosine 128.57° are equal (to four significant digits).

Keys:	Display:	Description:
MODE 2 (2RAD)		Sets Radians mode; **RAD** annunciator on.
· 5 · 7 ENTER	0.7143	5/7 in decimal format.
↰ π × COS	-0.6235	Cos (5/7)π.
MODE 1 (1DEG)	-0.6235	Switches to Degrees mode (no annunciator).
1 2 8 · 5 7 COS	-0.6235	Calculates cos 128.57°, which is the same as cos (5/7)π.

Programming Note:

Equations using inverse trigonometric functions to determine an angle θ, often look something like this:

$$\theta = \arctan\,(y/x).$$

If $x = 0$, then y/x is undefined, resulting in the error: `DIVIDE BY 0`.

Hyperbolic Functions

With *x* in the display:

To Calculate:	Press:
Hyperbolic sine of *x* (SINH).	↰ HYP SIN
Hyperbolic cosine of *x* (COSH).	↰ HYP COS
Hyperbolic tangent of *x* (TANH).	↰ HYP TAN
Hyperbolic arc sine of *x* (ASINH).	↰ HYP ↱ ASIN
Hyperbolic arc cosine of *x* (ACOSH).	↰ HYP ↱ ACOS
Hyperbolic arc tangent of *x* (ATANH).	↰ HYP ↱ ATAN

✓ Percentage Functions

The percentage functions are special (compared with × and ÷) because they preserve the value of the base number (in the Y–register) when they return the result of the percentage calculation (in the X–register). You can then carry out subsequent calculations using both the base number and the result without reentering the base number.

To Calculate:	Press:
x% of *y*	*y* ENTER *x* ↱ %
Percentage change from *y* to *x*. ($y \neq 0$)	*y* ENTER *x* ↰ %CHG

Example:

Find the sales tax at 6% and the total cost of a $15.76 item.

Use FIX 2 display format so the costs are rounded appropriately.

Keys:	Display:	Description:
↰ DISPLAY 1 (1FIX) 2		Rounds display to two decimal places.
1 5 · 7 6 ENTER	15.76	
6 ↱ %	0.95	Calculates 6% tax.
+	16.71	Total cost (base price + 6% tax).

Suppose that the $15.76 item cost $16.12 last year. What is the percentage change from last year's price to this year's?

Keys:	Display:	Description:
1 6 · 1 2 ENTER	16.12	
1 5 · 7 6 ↰ %CHG	-2.23	This year's price dropped about 2.2% from last year's price.
↰ DISPLAY 1 (1FIX) 4	-2.2333	Restores FIX 4 format.

Note

The order of the two numbers is important for the %CHG function. The order affects whether the percentage change is considered positive or negative.

Physics Constants

There are 41 physics constants in the CONST menu. You can press ↰ CONST to view the following items.

CONST Menu

Items	Description	Value
c	Speed of light in vacuum	299792458 m s^{-1}
g	Standard acceleration of gravity	9.80665 m s^{-2}
G	Newtonian constant of gravitation	6.673×10 $^{-11}$ m^3 kg^{-1}s^{-2}
Vm	Molar volume of ideal gas	0.022413996 m^3 mol^{-1}
NA	Avogadro constant	6.02214199×10 ^{23} mol^{-1}
R∞	Rydberg constant	10973731.5685 m^{-1}
eV	Elementary charge	1.602176462×10^{-19} C
me	Electron mass	9.10938188×10^{-31} kg
mP	Proton mass	1.67262158×10^{-27} kg
mn	Neutron mass	1.67492716×10^{-27} kg
mμ	Muon mass	1.88353109×10^{-28} kg
k	Boltzmann constant	1.3806503×10^{-23} J K^{-1}
h	Planck constant	6.62606876×10^{-34} J s
$\overline{h}$	Planck constant over 2 pi	1.054571596×10^{-34} J s
ϕ_o	Magnetic flux quantum	2.067833636×10^{-15} Wb
a_o	Bohr radius	5.291772083×10^{-11} m
ε_o	Electric constant	8.854187817×10^{-12} F m^{-1}
R	Molar gas constant	8.314472 J mol^{-1} k^{-1}
F	Faraday constant	96485.3415 C mol^{-1}
u	Atomic mass constant	1.66053873×10^{-27} kg
μ_o	Magnetic constant	1.2566370614×10^{-6} NA^{-2}
μB	Bohr magneton	9.27400899×10^{-24} J T^{-1}
μN	Nuclear magneton	5.05078317×10^{-27} J T^{-1}
μP	Proton magnetic moment	1.410606633×10^{-26} J T^{-1}
μe	Electron magnetic moment	–9.28476362×10^{-24} J T^{-1}
un	Neutron magnetic moment	–9.662364×10^{-27} J T^{-1}

Items	Description	Value
μμ	Muon magnetic moment	$-4.49044813\times10^{-26}$ J T^{-1}
re	Classical electron radius	$2.817940285\times10^{-15}$ m
Z_0	Characteristic impendence of vacuum	376.730313461 Ω
λc	Compton wavelength	$2.426310215\times10^{-12}$ m
λcn	Neutron Compton wavelength	$1.319590898\times10^{-15}$ m
λcp	Proton Compton wavelength	$1.321409847\times10^{-15}$ m
α	Fine structure constant	7.297352533×10^{-3}
σ	Stefan–Boltzmann constant	5.6704×10^{-8} W m^{-2} K^{-4}
t	Celsius temperature	273.15
atm	Standard atmosphere	101325 Pa
γ p	Proton gyromagnetic ratio	267522212 $s^{-1}T^{-1}$
C1	First radiation constant	374177107×10^{-16} W m^2
C2	Second radiation constant	0.014387752 m K
G_0	Conductance quantum	7.748091696×10^{-5} S
e	The base number of natural logarithm(natural constant)	2.71828182846

Reference: Peter J.Mohr and Barry N.Taylor, CODATA Recommended Values of the Fundamental Physical Constants: 1998, Journal of Physical and Chemical Reference Data,Vol.28, No.6,1999 and Reviews of Modern Physics,Vol.72, No.2, 2000.

To insert a constant:

1. Position your cursor where you want the constant inserted.
2. Press ⬑ CONST to display the physics constants menu.
3. Press > < ^ v (or, you can press ⬑ CONST to access the next page, one page at a time) to scroll through the menu until the constant you want is underlined, then press ENTER to insert the constant.

Note that constants should be referred to by their names rather than their values, when used in expressions, equations, and programs.

Conversion Functions

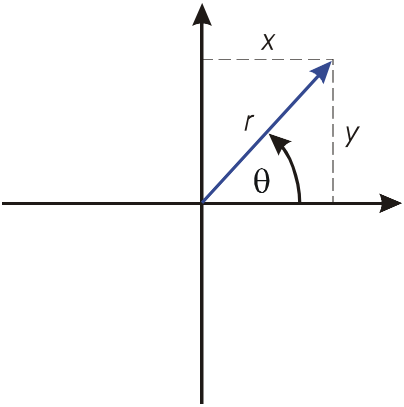
The HP 35s supports four types of conversions. You can convert between:

- rectangular and polar formats for complex numbers
- degrees, radians, and gradients for angle measures
- decimal and hexagesimal formats for time (and degree angles)
- various supported units (cm/in, kg/lb, etc)

With the exception of the rectangular and polar conversions, each of the conversions is associated with a particular key. The left (yellow) shift of the key converts one way while the right (blue) shift of the same key converts the other way. For each conversion of this type, the number you entered is assumed to be measured using the other unit. For example, when using [→°F] to convert a number to Fahrenheit degrees, the number you enter is assumed to be a temperature measured in Celsius degrees. The examples in this chapter utilize RPN mode. In ALG mode, enter the function first, then the number to convert.

Rectangular/Polar Conversions

Polar coordinates (*r*,*θ*) and rectangular coordinates (*x*,*y*) are measured as shown in the illustration. The angle *θ* uses units set by the current angular mode. A calculated result for *θ* will be between –180° and 180°, between –π and π radians, or between –200 and 200 grads.



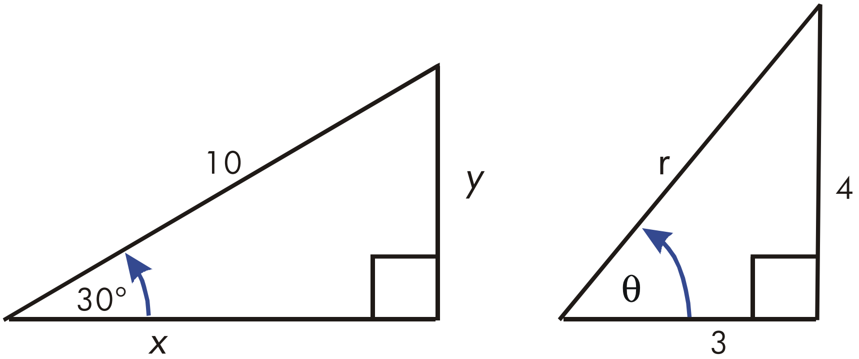
To convert between rectangular and polar coordinates:

The format for representing complex numbers is a mode setting. You may enter a complex number in any format; upon entry, the complex number is converted to the format determined by the mode setting. Here are the steps required to set a complex number format:

1. Press ↰ DISPLAY and then choose either 9 (9x·iy) or · 0 (10rθa) in RPN mode (in ALG mode, you may also choose · 1 (11x+y·i)
2. Input your desired coordinate value (x i y, x + y i or r ↱ θ a)
3. press ENTER

Example: Polar to Rectangular Conversion.

In the following right triangles, find sides *x* and *y* in the triangle on the left, and hypotenuse *r* and angle *θ* in the triangle on the right.



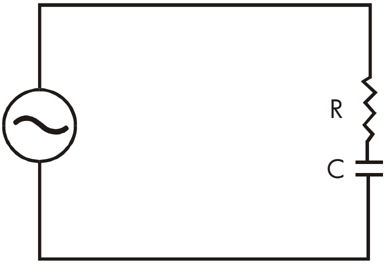
Keys:	Display:	Description:
MODE 1(1DEG) ↰ DISPLAY 9 (9x·iy)		Sets Degrees and complex coordinate mode.
1 0 ↱ θ 3 0 ENTER	8.6603i5.0000	Convert rθa (polar) to xiy (rectangular).

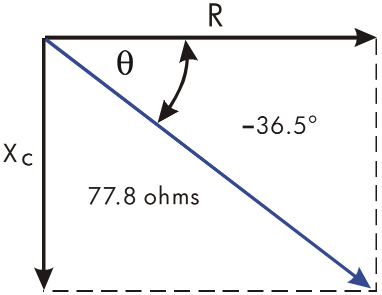
↰ DISPLAY · 0 (10rθa)	10.0000θ30.0000	Sets complex coordinate mode.
3 i 4 ENTER	5.0000θ53.1301	Convert xiy (rectangular) to r*θ* a (polar).

Example: Conversion with Vectors.

Engineer P.C. Bord has determined that in the RC circuit shown, the total impedance is 77.8 ohms and voltage lags current by 36.5°. What are the values of resistance R and capacitive reactance X_C in the circuit?

Use a vector diagram as shown, with impedance equal to the polar magnitude, *r*, and voltage lag equal to the angle, *θ*, in degrees. When the values are converted to rectangular coordinates, the *x*–value yields *R*, in ohms; the *y*–value yields X_C , in ohms.





Keys:	Display:	Description:
MODE 1 (1DEG)		Sets Degrees and complex coordinate mode.
↰ DISPLAY 9 (9x.iy)		
7 7 · 8 ↱ θ	77.8θ-36.5	Enters *θ*, degrees of voltage lag.
3 6 · 5 +/–		Enters *r*, ohms of total impedance.
ENTER	62.5401i-46.2772	Calculates *x*, ohms resistance, *R*. Calculates *y*, ohms reactance, X_C

Time Conversions

The HP 35s can convert between decimal and hexagesimal formats for numbers. This is especially useful for time and angles measured in degrees. For example, in decimal format an angle measured in degrees is expressed as D.ddd…, while in hexagesimal the same angle is represented as D.MMSSss, where D is the integer pat of the degree measure, ddd… is the fractional part of the degree measure, MM is the integer number of minutes, SS is the integer part of the number of seconds, and ss is the fractional part of the number of seconds.

✓ **To convert between decimal format and hours minutes, and seconds:**

1. Enter the number you wish to convert
2. Press ↰ →HMS to convert to hours/degrees, minutes, and seconds or press ↱ HMS→ to convert back to decimal format.

Example: Converting Time Formats.

How many minutes and seconds are there in 1/7 of an hour? Use FIX 6 display format.

Keys:	Display:	Description:
↰ DISPLAY 1 (1FIX) 6		Sets FIX 6 display format.
. 1 . 7	0.000000 0 1/7	1/7 hour as a decimal fraction.
↱ →HMS	0.000000 0.083429	Equals 8 minutes and 34.29 seconds.
↰ DISPLAY 1 (1FIX) 4	0.000000 0.0834	Restores FIX 4 format.

✓ Angle Conversions

When converting to radians, the number in the x–register is assumed to be degrees; when converting to degrees, the number in the x–register is assumed to be radians.

To convert an angle between degrees and radians:

Example

In this example, we convert an angle measure of 30° to π/6 radians.

Keys:	Display:	Description:
3 0	0.0000 30_	Enter the angle in degrees.
↰ →RAD	0.0000 0.5236	Convert to radians. Read the result as 0.5236, a decimal approximation of π/6.

Unit Conversions

The HP 35s has ten unit–conversion functions on the keyboard: →kg, →lb, →°C, →°F, →cm, →in, →l, →gal, →MILE,→KM

To Convert:	To:	Press:	Displayed Results:
1 lb	kg	1 ↱ →kg	0.4536 (kilograms)
1 kg	lb	1 ↰ →lb	2.2046 (pounds)
32 °F	°C	3 2 ↱ →°C	0.0000 (°C)
100 °C	°F	1 0 0 ↰ →°F	212.0000 (°F)
1 in	cm	1 ↱ →cm	2.5400 (centimeters)
100 cm	in	1 0 0 ↰ →in	39.3701 (inches)
1 gal	l	1 ↱ →l	3.7854 (liters)
1 l	gal	1 ↰ →gal	0.2642 (gallons)
1 MILE	KM	1 ↱ →KM	1.6093(KMS)
1 KM	MILE	1 ↰ →MILE	0.6214(MILES)

Probability Functions

✓ Factorial

To calculate the *factorial* of a displayed non-negative integer *x* ($0 \le x \le 253$), press ↱ ! (the right–shifted Σ+ key).

✓ Gamma

To calculate the *gamma function* of a noninteger *x*, $\Gamma(x)$, key in ($x - 1$) and press ↱ !. The *x*! function calculates $\Gamma(x + 1)$. The value for *x* cannot be a negative integer.

Probability

✓ *Combinations*

To calculate the number of possible sets of *n* items taken *r* at *a* time, enter *n* first, ↰ nCr, then *r* (nonnegative integers only). No item occurs more than once in a set, and different orders of the same *r* items are not counted separately.

✓ *Permutations*

To calculate the number of possible *arrangements* of *n* items taken *r* at a time, enter *n* first, ↱ nPr, then *r* (nonnegative integers only). No item occurs more than once in an arrangement, and different orders of the same *r* items *are* counted separately.

✓ *Seed*

To store the number in *x* as a new seed for the random number generator, press ↰ SEED.

✓ *Random number generator*

To generate a random number in the range $0 < x < 1$, press ↱ RAND. (The number is part of a uniformly–distributed pseudo–random number sequence. It passes the spectral test of D. Knuth, *The Art of Computer Programming,* vol. 2, *Seminumerical Algorithms*, London: Addison Wesley, 1981.)

The RANDOM function uses a seed to generate a random number. Each random number generated becomes the seed for the next random number. Therefore, a sequence of random numbers can be repeated by starting with the same seed. You can store a new seed with the SEED function. If memory is cleared, the seed is reset to zero. A seed of zero will result in the calculator generating its own seed.

Example: Combinations of People.

A company employing 14 women and 10 men is forming a six–person safety committee. How many different combinations of people are possible?

Keys:	Display:	Description:

2 4 ENTER 6	`24`
	`6_`
		Twenty–four people grouped six at a time.

↰ nCr	`134,596.0000`	Total number of combinations possible.

If employees are chosen at random, what is the probability that the committee will contain six women? To find the *probability* of an event, divide the number of combinations *for that event* by the total number of combinations.

Keys:	Display:	Description:

1 4 ENTER 6	`14`
	`6_`
		Fourteen women grouped six at a time.

↰ nCr	`3,003.0000`	Number of combinations of six women on the committee.
x◂▸y	`134,596.0000`	Brings total number of combinations back into the X–register.
÷	`0.0223`	Divides combinations of women by total combinations to find probability that any one combination would have all women.

Parts of Numbers

These functions are primarily used in programming.

Integer part

✓ To remove the fractional part of *x* and replace it with zeros, press ↰ INTG 6 (6IP). (For example, the integer part of 14.2300 is 14.0000.)

Fractional part

✓ To remove the integer part of *x* and replace it with zeros, press ↰ INTG 5 (5FP). (For example, the fractional part of 14.2300 is 0.2300)

Absolute value

To replace a number in the x-register with its absolute value, press ↱ ABS. For complex numbers and vectors, the absolute value of:

1. a complex number in rθa format is r
2. a complex number in xiy format is $\sqrt{x^2 + y^2}$
3. a vector [A1,A2,A3, …An] is $|A| = \sqrt{A_1{}^2 + A_2{}^2 + \cdots + A_n{}^2}$

Argument value

To extract the argument of a complex number, use ↰ ARG. The argument of a complex number:

1. in rθa format is a
2. in xiy format is Atan(y/x)

Sign value

✓ To indicate the sign of *x*, press ↰ INTG 1(1SGN). If the *x* value is negative, –1.0000 is displayed; if zero, 0.0000 is displayed; if positive, 1.0000 is displayed.

Greatest integer

✓ To obtain the greatest integer equal to or less than given number, press [↰] [INTG] [4] (4INTG).

Example:

This example summarizes many of the operations that extract parts of numbers.

To calculate:	Press:	Display:
The integer part of 2.47	[2] [.] [4] [7] [↰] [INTG] [6] (6IP)	2.0000
The fractional part of 2.47	[2] [.] [4] [7] [↰] [INTG] [5] (5FP)	0.4700
The absolute value of –7	[7] [+/_] [↱] [ABS]	7.0000
The sign value of 9	[9] [↰] [INTG] [1] (1SGN)	1.0000
The greatest integer equal to or less than –5.3	[5] [.] [3] [+/_] [↰] [INTG] [4] (4INTG)	-6.0000

The RND function ([↱] [RND]) rounds *x* internally to the number of digits specified by the display format. (The internal number is represented by 12 digits.) Refer to chapter 5 for the behavior of RND in Fraction–display mode.

5

Fractions

In Chapter 1, the section *Fractions* introduced the basics of entering, displaying, and calculating with fractions. This chapter gives more information on these topics. Here is a short review of entering and displaying fractions:

- To enter a fraction, press · twice: once after the integer part of a mixed number and again between the numerator and denominator of the fractional part of the number. To enter 2 3/8, press 2 · 3 · 8. To enter 5/8, press either · 5 · 8 or 0 · 5 · 8.
- To toggle Fraction-display mode on and off, press ↱ FDISP. When Fraction-display mode is turned off, the display reverts to the previous display format set via the Display menu. Choosing another format via this menu also turns off Fraction-display mode, if active.
- Functions work the same with fractions as they do with decimal numbers – except for RND, which is discussed later in this chapter.

The examples in this chapter all utilize RPN mode unless otherwise noted.

Entering Fractions

You can type almost any number as a fraction on the keyboard — including an improper fraction (where the numerator is larger than the denominator).

Example:

Keys:	Display:	Description:
↱ FDISP		Turns on Fraction–display mode.
1 · 5 ENTER	1 1/2	Enters 1.5; shown as a fraction.
1 · 3 · 4 ENTER	1 3/4	Enters 1 $^3/_4$.
↱ FDISP	1.7500	Displays *x* as a decimal number.
↱ FDISP	1 3/4	Displays *x* as a fraction.

If you didn't get the same results as the example, you may have accidentally changed how fractions are displayed. (See "Changing the Fraction Display" later in this chapter.)

The next topic includes more examples of valid and invalid input fractions.

Fractions in the Display

In Fraction–display mode, numbers are evaluated internally as decimal numbers, then they're displayed using the most precise fractions allowed. In addition, accuracy annunciators show the direction of any inaccuracy of the fraction compared to its 12–digit decimal value. (Most statistics registers are exceptions — they're always shown as decimal numbers.)

Display Rules

The fraction you see may differ from the one you enter. In its default condition, the calculator displays a fractional number according to the following rules. (To change the rules, see "Changing the Fraction Display" later in this chapter.)

- The number has an integer part and, if necessary, a proper fraction (the numerator is less than the denominator).
- The denominator is no greater than 4095.
- The fraction is reduced as far as possible.

Examples:

These are examples of entered values and the displayed results. For comparison, the internal 12–digit values are also shown. The ▲ and ▼ annunciators in the last column are explained below.

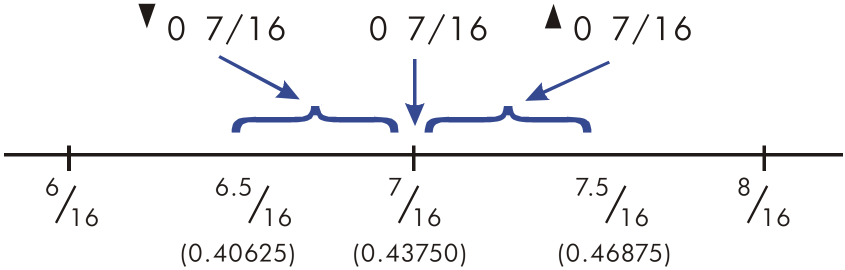
Entered Value	Internal Value	Displayed Fraction
$2\ ^{3}/_{8}$	2.37500000000	2 3/8
$14\ ^{15}/_{32}$	14.4687500000	14 15/32
$^{54}/_{12}$	4.50000000000	4 1/2
$6\ ^{18}/_{5}$	9.60000000000	9 3/5
$^{34}/_{12}$	2.83333333333	2 5/6 ▼
$^{15}/_{8192}$	0.00183105469	0 7/3823 ▲
$12345678\ ^{12345}/_{3}$	12349793.0000	12349793
$16\ ^{3}/_{16384}$	16.0001831055	16 1/4095

Accuracy Indicators

The accuracy of a displayed fraction is indicated by the ▲ and ▼ annunciators at the right of the display. The calculator compares the value of the fractional part of the internal 12–digit number with the value of the displayed fraction:

- If no indicator is lit, the fractional part of the internal 12–digit value exactly matches the value of the displayed fraction.
- If ▼ is lit, the fractional part of the internal 12–digit value is slightly less than the displayed fraction — the *exact* numerator is no more than 0.5 *below* the displayed numerator.
- If ▲ is lit, the fractional part of the internal 12–digit value is slightly greater than the displayed fraction — the *exact* numerator is no more than 0.5 *above* the displayed numerator.

This diagram shows how the displayed fraction relates to nearby values — ▲ means the exact numerator is "a little above" the displayed numerator, and ▼ means the exact numerator is "a little below".



This is especially important if you change the rules about how fractions are displayed. (See "Changing the Fraction Display" later.) For example, if you force all fractions to have 5 as the denominator, then $^{2}/_{3}$ is displayed as 0 3/5▲ because the exact fraction is approximately $^{3.3333}/_{5}$, "a little above" $^{3}/_{5}$. Similarly, –$^{2}/_{3}$ is displayed as -0 3/5▲ because the true numerator is "a little above" 3.

Sometimes an annunciator is lit when you wouldn't expect it to be. For example, if you enter 2 $^{2}/_{3}$, you see 2 2/3▲, even though that's the exact number you entered. The calculator always compares the fractional part of the internal value and the 12–digit value of just the fraction. If the internal value has an integer part, its fractional part contains less than 12 digits — and it can't exactly match a fraction that uses all 12 digits.

Changing the Fraction Display

In its default condition, the calculator displays a fractional number according to certain rules. However, you can change the rules according to how you want fractions displayed:

- You can set the maximum denominator that's used.
- You can select one of three fraction formats.

The next few topics show how to change the fraction display.

Setting the Maximum Denominator

For any fraction, the denominator is selected based on a value stored in the calculator. If you think of fractions as *a b/c*, then */c* corresponds to the value that controls the denominator.

The */c* value defines only the *maximum* denominator used in Fraction–display mode — the specific denominator that's used is determined by the fraction format (discussed in the next topic).

- To set the maximum denominator value, enter the value and then press [↰] [/c]. Fraction-display mode will be automatically enabled. The value you enter cannot exceed 4095.
- To recall the */c* value to the X–register, press [1] [↰] [/c].
- To restore the default value to 4095, press [0] [↰] [/c] or enter any value greater than 4095 as the maximum denominator. Again, Fraction-display mode will be automatically enabled.

The */c* function uses the absolute value of the integer part of the number in the X–register. It doesn't change the value in the LAST X register.

If the displayed fraction is too long to fit in the display, the ➡ annunciator will appear; you can then use [↱] [<] and [↱] [>] to scroll page by page to see the rest of the fraction. To see the number's decimal representation, press [↰] and then hold [SHOW].

Example:

This example illustrates the steps required to set the maximum denominator to 3125 and then show a fraction that is too long for the display.

Keys:	Display:	Description:
[3] [1] [2] [5] [↰] [/c]		Set the maximum denominator to 3125.
[1] [4] [↱] [e^x]	0 1202604 888/31	Note the missing digits in the denominator.
[>]	0 25	Scroll right to see the rest of the denominator.

Notes:

1. In ALG mode, you can enter an expression in line 1 and then press [↰] [/c]. In this case, the expression is evaluated and the result is used to determine the maximum denominator.

2. In ALG mode, you can use the result of a calculation as the argument for the /c function. With the value in line 2, simply press ↰ /c. The value in line 2 is displayed in Fraction format and the integer part is used to determine the maximum denominator.
3. You may not use either a complex number or a vector as the argument for the / c command. The error message "INVALID DATA" will be displayed.

Choosing a Fraction Format

The calculator has three fraction formats. The displayed fractions are always the most accurate fractions within the rules for the selected format.

- **Most precise fractions.** Fractions have any denominator up to the */c* value, and they're reduced as much as possible. For example, if you're studying math concepts with fractions, you might want *any* denominator to be possible (*/c* value is 4095). This is the default fraction format.
- **Factors of denominator.** Fractions have only denominators that are factors of the */c* value, and they're reduced as much as possible. For example, if you're calculating stock prices, you might want to see 53 1/4 and 37 7/8 (*/c* value is 8). Or if the */c* value is 12, possible denominators are 2, 3, 4, 6, and 12.
- **Fixed denominator**. Fractions always use the */c* value as the denominator — they're not reduced. For example, if you're working with time measurements, you might want to see 1 25/60 (*/c* value is 60).

There are three flags that control the fraction format. These flags are numbered 7, 8, and 9. Each flag is either clear or set. Their purposes are as follows:

- Flag 7 toggles fraction-display mode on or off; clear=off and set=on.
- Flag 8 toggles between using any value less than or equal to the /c value or using only factors of the /c value; clear = use any value and set = use only factors of the /c value.
- Flag 9 operates only if Flag 8 is set and toggles between reducing or not reducing the fractions; clear = reduce and set = do not reduce.

With Flags 8 and 9 appropriately cleared or set, you can get the three fraction formats as shown in the table below:

To Get This Fraction Format:	Change These Flags: 8	9
Most precise	Clear	—
Factors of denominator	Set	Clear
Fixed denominator	Set	Set

You can change flags 8 and 9 to set the fraction format using the steps listed here. (Because flags are especially useful in programs, their use is covered in detail in chapter 14.)

1. Press ↰ FLAGS to get the flag menu.
2. To set a flag, press 1(1SF) and type the flag number, such as 8.

   To clear a flag, press 2(2CF) and type the flag number.

   To see if a flag is set, press 3(3FS?) and type the flag number. Press C or ← to clear the YES or NO response.)

Example:

This example illustrates the display of fractions in the three formats using the number π. This example assumes fraction-display format is active and that Flag 8 is in its default state (cleared).

Keys:	Display:	Description:
4 0 9 5 ↰ /c		Sets the maximum /c value back to the default.
↰ π	0	Most precise format
	3 16/113	Flag 8 = clear.
↰ FLAGS 1(1SF)	0	Flag 8 = set;
8	3 116/819	Factors of denominator format; 819*5=4095
↰ FLAGS 1(1SF)	0 0/4095	Flag 9 = set;
9	3 580/4095	Fixed denominator format
↰ FLAGS 2(2CF)	0	Return to default format (most precise)
8 ↰ FLAGS 2(2CF) 9	3 16/113	

Examples of Fraction Displays

The following table shows how the number 2.77 is displayed in the three fraction formats for two */c* values.

Fraction Format	How 2.77 Is Displayed	
	***/c* = 4095**	***/c* = 16**
Most Precise	2 77/100 (2.7700)	2 10/13▲ (2.7692)
Factors of Denominator	2 1051/1365▲ (2.7699)	2 3/4▲ (2.7500)
Fixed Denominator	2 3153/4095▲ (2.7699)	2 12/16▲ (2.7500)

The following table shows how different numbers are displayed in the three fraction formats for a */c* value of 16.

Fraction Format ∗	Number Entered and Fraction Displayed				
	2	2.5	$2\,^{2}/_{3}$	2.9999	$2\,^{16}/_{25}$
Most precise	2	2 1/2	2 2/3▲	3▼	2 9/14▼
Factors of denominator	2	2 1/2	2 11/16▼	3▼	2 5/8▲
Fixed denominator	2 0/16	2 8/16	2 11/16▼	3 0/16▼	2 10/16▲

∗ For a */c* value of 16.

Rounding Fractions

If Fraction–display mode is active, the RND function converts the number in the X–register to the closest decimal representation of the fraction. The rounding is done according to the current */c* value and the states of flags 8 and 9. The accuracy indicator turns off if the fraction matches the decimal representation exactly. Otherwise, the accuracy indicatior stays on, (See "Accuracy Indicators" earlier in this chapter.)

In an equation or program, the RND function does fractional rounding if Fraction–display mode is active.

Example:

Suppose you have a 56 $^{3}/_{4}$–inch space that you want to divide into six equal sections. How wide is each section, assuming you can conveniently measure $^{1}/_{16}$–inch increments? What's the cumulative roundoff error?

Keys:	Display:	Description:
↰ FLAGS ENTER 8		Sets Flag 8
1 6 ↰ /c		Sets up fraction format for $^{1}/_{16}$–inch increments. (Flags 8 and 9 should be the same as for the previous example.)
5 6 · 3 · 4 ↱ STO D	56 3/4	Stores the distance in *D.*
6 ÷	9 7/16▲	The sections are a bit wider than 9 $^{7}/_{16}$ inches.
↱ RND	9 7/16	Rounds the width to this value.
6 ×	56 5/8	Width of six sections.
RCL D −	-0 1/8	The cumulative round off error.
↰ FLAGS 2(2CF) 8	-0 1/8	Clears flag 8.
↱ FDISP	-0.1250	Turns off Fraction–display mode.

Fractions in Equations

You can use a fraction in an equation. When an equation is displayed, all numerical values in the equation are shown in their entered form. Also, fraction-display mode is available for operations involving equations.

When you're evaluating an equation and you're prompted for variable values, you may enter fractions — values are displayed using the current display format.

See chapter 6 for information about working with equations.

Fractions in Programs

You can use a fraction in a program just as you can in an equation; numerical values are shown in their entered form.

When you're running a program, displayed values are shown using Fraction–display mode if it's active. If you're prompted for values by INPUT instructions, you may enter fractions. The program’s result is displayed using the current display format.

A program can control the fraction display using the */c* function and by setting and clearing flags 7, 8, and 9. See "Flags" in chapter 14.

See chapters 13 and 14 for information about working with programs.

6

Entering and Evaluating Equations

How You Can Use Equations

You can use equations on the HP 35s in several ways:

- For specifying an equation to evaluate (this chapter).
- For specifying an equation to solve for unknown values (chapter 7).
- For specifying a function to integrate (chapter 8).

Example: Calculating with an Equation.

Suppose you frequently need to determine the volume of a straight section of pipe. The equation is

$$V = .25\ \pi\ d^2\ l$$

where *d* is the inside diameter of the pipe, and *l* is its length.

✓ You could key in the calculation over and over; for example, . 2 5 ENTER ↰ π × 2 . 5 ↱ x^2 × 1 6 × calculates the volume of 16 inches of 2 $^1/_2$–inch diameter pipe (78.5398 cubic inches). However, by storing the *equation*, you get the HP 35s to "remember" the relationship between diameter, length, and volume — so you can use it many times.

Put the calculator in Equation mode and type in the equation using the following keystrokes:

Keys:	Display:	Description:
[EQN]	`EQN LIST TOP` or the current equation in line 2	Selects Equation mode, shown by the **EQN** annunciator.
[RCL]		Begins a new equation, [RCL] turns on the **A..Z** annunciator so you can enter a variable name.
[V] [↰] [=]	`V=_`	[RCL] [V] types `V`
[·] [2] [5]	`V=      0.25_`	Digit entry uses the "_" entry cursor.
[×] [↰] [π] [×]	`V=0.25×π×_`	[×] ends the number.
[RCL] [D] [y^x] [2]	`V=0.25×π×D^ 2_`	[y^x] types ^.
[×] [RCL] [L]	`V=0.25×π×D^2×L_`	
[ENTER]	`V=0.25×π×D^2×L`	Terminates and displays the equation.

[↰] [SHOW]	`CK=49CA`
	`LN=14`
		Shows the checksum and length for the equation, so you can check your keystrokes.

By comparing the checksum and length of your equation with those in the example, you can verify that you've entered the equation properly. (See "Verifying Equations" at the end of this chapter for more information.)

Evaluate the equation (to calculate *V*):

Keys:	Display:	Description:
[ENTER]	`D?` *value*	Prompts for variables on the right–hand side of the equation. Prompts for *D* first; value is the current value of *D*.

[2] [·] [1] [·] [2]	`D?`
	`2 1/2_`
		Enters 2 $^1/_2$ inches as a fraction.

[R/S]	`L?` *value*	Stores *D*, prompts for *L*; value is current value of *L*.

[1] [6] [R/S]	`V=`
	`78.5398`
		Stores *L*; calculates *V* in cubic inches and stores the result in *V*.

Summary of Equation Operations

All equations you create are saved in the *equation list.* This list is visible whenever you activate Equation mode.

You use certain keys to perform operations involving equations. They're described in more detail later.

When displaying equations in the equation list, two equations are displayed at a time. The currently active equation is shown on line 2.

Key	Operation
EQN	Enters and leaves Equation mode.
ENTER	Evaluates the displayed equation. If the equation is an *assignment*, evaluates the right–hand side and stores the result in the variable on the left–hand side. If the equation is an *equality* or *expression*, calculates its value like XEQ. (See "Types of Equations" later in this chapter.)
XEQ	Evaluates the displayed equation. Calculates its value, replacing "=" with "–" if an "=" is present.
SOLVE	Solves the displayed equation for the unknown variable you specify. (See chapter 7.)
↰ ∫	Integrates the displayed equation with respect to the variable you specify. (See chapter 8.)
←	Deletes the current equation or deletes the element to the left of the cursor.
‹ or ›	Begins editing the displayed equation, only moving the cursor and not deleting any content.
↱ ‹ or ↱ ›	Scroll the current equation display screen.
⌃ or ⌄	Steps up or down through the equation list.
↱ ⌃ or ↱ ⌄	Jumps to the top or bottom of the equation list.
↰ SHOW	Shows the displayed equation's checksum (verification value) and length (bytes of memory).
↰ UNDO	Recovers the most recently deleted element or equation.
C	Leaves Equation mode.

You can also use equations in programs — this is discussed in chapter 13.

Entering Equations into the Equation List

The *equation list* is a collection of equations you enter. The list is saved in the calculator's memory. Each equation you enter is automatically saved in the equation list.

To enter an equation:

You can make an equation as long as you want – it is limited only by the amount of available memory.

1. Make sure the calculator is in its normal operating mode, usually with a number in the display. For example, you can't be viewing the catalog of variables or programs.
2. Press EQN. The **EQN** annunciator shows that Equation mode is active, and an entry from the equation list is displayed.
3. Start typing the equation. The previous display is replaced by the equation you're entering — the previous equation isn't affected. If you make a mistake, press ← or ⬑ UNDO as required.
4. Press ENTER to terminate the equation and see it in the display. The equation is automatically saved in the equation list — right after the entry that was displayed when you started typing. (If you press C instead, the equation is saved, but Equation mode is turned off.)

Equations can contain variables, numbers, vectors, functions, and parentheses — they're described in the following topics. The example that follows illustrates these elements.

Variables in Equations

You can use any of the calculator's variables in an equation: *A* through *Z*,**(I)** and **(**J**)**. You can use each variable as many times as you want.(For information about **(I) and (**J**)**, see "Indirectly Addressing Variables and Labels" in chapter 14.)

To enter a variable in an equation, press RCL *variable*. When you press RCL, the **A..Z** annunciator shows that you can press a variable key to enter its name in the equation.

Numbers in Equations

You can enter any valid number in an equation, including base 2, 8 and 16, real, complex, and fractional numbers. Numbers are always shown using ALL display format, which displays up to 12 characters.

To enter a number in an equation, you can use the standard number–entry keys, including [.], [+/−], and [E]. Do not use [+/−] for subtraction.

Functions in Equations

You can enter many HP 35s functions in an equation. A complete list is given under “Equation Functions” later in this chapter. Appendix G, "Operation Index," also gives this information.

When you enter an equation, you enter functions in about the same way you put them in ordinary algebraic equations:

- In an equation, certain functions are normally shown *between their* arguments, such as "+" and "÷". For such *infix* operators, enter them in an equation in the same order.
- Other functions normally have one or more arguments *after* the function name, such as "COS" and "LN". For such *prefix* functions, enter them in an equation where the function occurs — the key you press puts a left parenthesis after the function name so you can enter its arguments.
- If the function has two or more arguments, press [↰] [0] to separate them.

Parentheses in Equations

You can include parentheses in equations to control the order in which operations are performed. Press ( ) to insert parentheses. (For more information, see "Operator Precedence" later in this chapter.)

Example: Entering an Equation.

Enter the equation $r = 2 \times c \times (t - a) + 25$

Keys:	Display:	Description:
EQN	V=0.25×π×D^2×L	Shows the last equation used in the equation list.
RCL R ↰ =	R=_	Starts a new equation with variable *R*.
2	R= 2_	Enters a number
× RCL C ×	R=2×C×_	Enters infix operators.
( )	R=2×C×(_)	Enters a prefix function with a left parenthesis.
RCL T − RCL A > + 2 5	=2×C×(T-A)+ 25 _	Enters the argument and right parenthesis.
ENTER	R=2×C×(T-A)+25	Terminates the equation and displays it.

↰ SHOW	CK=9E5F
	LN=14
		Shows its checksum and length.

C		Leaves Equation mode.

Displaying and Selecting Equations

The equation list contains two built-in equations, 2*2 lin. solve and 3*3 lin. Solve, and the equations you've entered. You can display the equations and select one to work with.

To display equations:

1. Press EQN. This activates Equation mode and turns on the **EQN** annunciator. The display shows an entry from the equation list:
   - `EQN LIST TOP` if the equation pointer is at the top of the list.
   - The current equation (the last equation you viewed).
2. Press ⌃ or ⌄ to step through the equation list and view each equation. The list "wraps around" at the top and bottom. `EQN LIST TOP` marks the "top" of the list.

To view a long equation:

1. Display the equation in the equation list, as described above. If it's more than 14 characters long, only 14 characters are shown. The ➡ annunciator indicates more characters to the right.
2. Press > to begin editing the equation at the beginning, or press < to begin editing the equation at the end. Then press < or > repeatedly to move the cursor through the equation one character at a time. ⬅ and ➡ display when there are more characters to the left or right.
3. Press ↱ < or ↱ > to scroll the long equations in line 2 by a screen.

To select an equation:

Display the equation in the equation list, as described above. The displayed equation in line 2 is the one that's used for all equation operations.

Example: Viewing an Equation.

View the last equation you entered.

Keys:	Display:	Description:
EQN	`R=2×C×(T−A)+25`	Displays the current equation in the equation list.
>	`R=2×C×(T−A)+25`	Activates cursor to the left of the equation
ENTER <	`=2×C×(T−A)+25_`	Activates cursor to the right of the equation
C		Leaves Equation mode.

Editing and Clearing Equations

You can edit or clear an equation that you're typing. You can also edit or clear equations saved in the equation list. However, you cannot edit or clear the two built-in equations 2*2 lin. solve and 3*3 lin. solve. If you attempt to insert a equation between the two built-in equations, the new equation will be inserted after 3*3 lin. solve.

To edit an equation you're typing:

1. Press < or > to move the cursor allowing you to insert characters before the cursor.
2. Move the cursor and press ← repeatedly to delete the unwanted number or function. Pressing ← when the equation editing line is empty has no effect, but pressing ENTER on an empty equation line causes the empty equation line to be deleted. The display then shows the previous entry in the equation list.
3. Press ENTER (or C) to save the equation in the equation list.

To edit a saved equation:

1. Display the desired equation, press > to activate the cursor at the beginning of the equation or press < to activate the cursor at the end of the equation.(See "Displaying and Selecting Equations" above.)
2. When the cursor is active in the equation, you can edit the equation just like you would when entering a new equation.
3. Press ENTER (or C) to save the edited equation in the equation list, replacing the previous version.

Using menus while editing an equation:

1. When editing an equation, selecting a setting menu (such as MODE, ↰ DISPLAY, or ↱ CLEAR), will end the equation edit status.
2. When editing an equation, selecting an insert or view menu (such as L.R., ↰ $\bar{x},\bar{y}$, ↱ S,σ, ↱ SUMS, ↱ BASE, ↰ LOGIC, R↓, ↰ MEM and ↰ CONST), the equation will still be in edit mode after inserting the item.
3. The menus x?y, FLAGS, ↱ x?0 are disabled in equation mode.

To clear a saved equation:

Scroll the equation list up or down until the desired equation is in line 2 of the display, and then press ←.

To clear all saved equations:

In EQN mode, press ↱ CLEAR. Select 3 (3EQN). The CLR EQN? Y N menu is displayed. Select < (Y) ENTER.

Example: Editing an Equation.

Remove 25 in the equation from the previous example.

Keys:	Display:	Description:
EQN	R=2xCx(T-A)+25	Shows the current equation in the equation list.
<	=2xCx(T-A)+25_	Activates cursor at the end of the equation
← ← ←	=2xCxCOS(T-A)_	Deletes the number 25.
ENTER	R=2xCx(T-A)	Shows the end of edited equation in the equation list.
C		Leaves Equation mode.

Types of Equations

The HP 35s works with three types of equations:

- **Equalities.** The equation contains an "=", and the left side contains more than just a single variable. For example, $x^2 + y^2 = r^2$ is an *equality.*
- **Assignments.** The equation contains an "=", and the left side contains just a single variable. For example, $A = 0.5 \times b \times h$ is an *assignment.*

- **Expressions.** The equation does *not* contain an "=". For example, $x^3 + 1$ is an *expression.*

When you're calculating *with an* equation, you might use any type of equation — although the type can affect how it's evaluated. When you're solving a problem for an unknown variable, you'll probably use an equality or assignment. When you're integrating a function, you'll probably use an expression.

Evaluating Equations

One of the most useful characteristics of equations is their ability to be *evaluated* — to generate numeric values. This is what enables you to calculate a result from an equation. (It also enables you to solve and integrate equations, as described in chapters 7 and 8).

Because many equations have two sides separated by "=", the basic value of an equation is the *difference* between the values of the two sides. For this calculation, "=" in an equation is essentially treated as "–". The value is a measure of how well the equation balances.

The HP 35s has two keys for evaluating equations: ENTER and XEQ. Their actions differ only in how they evaluate *assignment* equations:

- XEQ returns the value of the equation, regardless of the type of equation.
- ENTER returns the value of the equation — *unless* it's an *assignment*–type equation. For an assignment equation, ENTER returns the value of the right side only, and also "enters" that value into the variable on the left side — it stores the value in the variable.

The following table shows the two ways to evaluate equations.

<table>
<tr><th>Type of Equation</th><th>Result for ENTER</th><th>Result for XEQ</th></tr>
<tr><td>Equality: $g(x) = f(x)$
Example: $x^2 + y^2 = r^2$</td><td colspan="2">$g(x) – f(x)$
$x^2 + y^2 – r^2$</td></tr>
<tr><td>Assignment: $y = f(x)$
Example: $A = 0.5 \times b \times h$</td><td>$f(x)$ ∗
$0.5 \times b \times h$ ∗</td><td>$y – f(x)$
$A – 0.5 \times b \times h$</td></tr>
<tr><td>Expression: $f(x)$
Example: $x^3 + 1$</td><td colspan="2">$f(x)$
$x^3 + 1$</td></tr>
<tr><td colspan="3">∗ Also stores the result in the left–hand variable, A for example.</td></tr>
</table>

To evaluate an equation:

1. Display the desired equation. (See "Displaying and Selecting Equations" above.)
2. Press ENTER or XEQ. The equation prompts for a value for each variable needed. (If the base of a number in the equation is different from the current base, the calculator automatically changes the result to the current base.)
3. For each prompt, enter the desired value:
   - If the displayed value is good, press R/S.
   - If you want a different value, type the value and press R/S. (Also see "Responding to Equation Prompts" later in this chapter.)

To halt a calculation, press C or R/S. The message `INTERRUPTED` is shown in line 2.

The evaluation of an equation takes no values from the stack — it uses only numbers in the equation and variable values. The value of the equation is returned to the X–register.

Using ENTER for Evaluation

If an equation is displayed in the equation list, you can press ENTER to evaluate the equation. (If you're in the process of *typing* the equation, pressing ENTER only *ends* the equation — it doesn't evaluate it.)

- If the equation is an *assignment,* only the right–hand side is evaluated. The result is returned to the X–register and stored in the left–hand variable, then the variable is viewed in the display. Essentially, ENTER finds the value of the left–hand variable.

- If the equation is an *equality* or *expression,* the entire equation is evaluated — just as it is for XEQ. The result is returned to the X–register.

Example: Evaluating an Equation with ENTER.

Use the equation from the beginning of this chapter to find the volume of a 35–mm diameter pipe that's 20 meters long.

Keys:	Display:	Description:
EQN (⌃ as required)	`V=0.25×π×D^2×L`	Displays the desired equation.

ENTER	`D?`
	`2.5`
		Starts evaluating the assignment equation so the value will be stored in *V*. Prompts for variables on the right–hand side of the equation. The current value for *D* is 2.5.

3 5 R/S	`L?`
	`16`
		Stores *D*, prompts for *L*, whose current value is 16.

2 0 × 1 0 0
0 ENTER
R/S
	`V=`
	`19,242,255.0033`
		Stores *L* in millimeters; calculates *V* in cubic millimeters, stores the result in *V*, and displays *V*.

÷ 1 E 6 ENTER	`19.2423`	Changes cubic millimelers to liters (but doesn't change *V*.

Using XEQ for Evaluation

If an equation is displayed in the equation list, you can press XEQ to evaluate the equation. The entire equation is evaluated, regardless of the type of equation. The result is returned to the X–register.

Example: Evaluating an Equation with XEQ.

Use the results from the previous example to find out how much the volume of the pipe changes if the diameter is changed to 35.5 millimeters.

Keys:	Display:	Description:
EQN	`V=0.25×P×D^2×L`	Displays the desired equation.

XEQ	`V?`
	`19,242,255.0033`
		Starts evaluating the equation to find its value. Prompts for *all* variables.

R/S	`D?`
	`35`
		Keeps the same *V*, prompts for *D*.

3 5 · 5
R/S
	`L?`
	`20,000`
		Stores new *D*, Prompts for *L*.

R/S	`-553,705.7051`	Keeps the same *L*; calculates the value of the equation — the imbalance between the left and right sides.
÷ 1 E 6 ENTER	`-0.5537`	Changes cubic millimeters to liters.

The value of the equation is the old volume (from *V*) *minus* the new volume (calculated using the new *D* value) — so the old volume is smaller by the amount shown.

Responding to Equation Prompts

When you evaluate an equation, you're prompted for a value for each variable that's needed. The prompt gives the variable name and its current value, such as `X?2.5000`. If the unnamed indirect variable (I) or (J) is in an equation, you will not be prompted to for its value, as the current value stored in the unnamed indirect variable will be used automatically. (See chapter 14)

- **To leave the number unchanged,** just press R/S.

- **To change the number,** type the new number and press R/S. This new number writes over the old value in the X–register. You can enter a number as a fraction if you want. If you need to calculate a number, use normal keyboard calculations, then press R/S. For example, you can press 2 ENTER 5 y^x R/S in RPN mode, or press 2 y^x 5 ENTER R/S in ALG mode. Before pressing ENTER, the expression will display in line 2, and after pressing ENTER, the result of the expression will display in line 2.
- **To cancel the prompt,** press C. The current value for the variable remains in the X–register and displays in right-side of the line two. If you press C during digit entry, it clears the number to zero. Press C again to cancel the prompt.
- **To display digits hidden by the prompt,** press ↰ SHOW.

In RPN mode,each prompt puts the variable value in the X–register and disables stack lift. If you type a number at the prompt, it replaces the value in the X–register. When you press R/S, stack lift is enabled, so the value is saved on the stack.

The Syntax of Equations

Equations follow certain conventions that determine how they're evaluated:

- How operators interact.
- What functions are valid in equations.
- How equations are checked for syntax errors.

Operator Precedence

Operators in an equation are processed in a certain order that makes the evaluation logical and predictable:

Order	Operation	Example
1	Parentheses	(X+1)
2	Functions	SIN(X+1)
3	Power (y^x)	X^3
4	Unary Minus (+/–)	-A
5	Multiply and Divide	X×Y, A÷B
6	Add and Subtract	P+Q, A-B
7	Equality	B=C

So, for example, all operations *inside* parentheses are performed *before* operations *outside* the parentheses.

Examples:

Equations	Meaning
A×B^3=C	$a \times (b^3) = c$
(A×B)^3=C	$(a \times b)^3 = c$
A+B÷C=12	$a + (b/c) = 12$
(A+B)÷C=12	$(a + b) / c = 12$
%CHG(T+12,A-6)^2	$[\%\text{CHG}\ ((t + 12), (a - 6))\ ]^2$

Equation Functions

The following table lists the functions that are valid in equations. Appendix G, "Operation Index" also gives this information.

LN	LOG	EXP	ALOG	SQ	SQRT
INV	IP	FP	RND	ABS	!
SGN	INTG	IDIV	RMDR		
SIN	COS	TAN	ASIN	ACOS	ATAN
SINH	COSH	TANH	ASINH	ACOSH	ATANH
→DEG	→RAD	HMS→	→HMS	%CHG	XROOT
→L	→GAL	→MILE	→KM	nCr	nPr
→KG	→LB	→°C	→°F	→CM	→IN
SEED	ARG	RAND	π		
+	–	×	÷	^	
sx	sy	σx	σy	$\bar{x}$	$\bar{y}$
$\bar{x}_w$	$\hat{x}$	$\hat{y}$	r	m	b
n	Σx	Σy	Σx^2	Σy^2	Σxy

For convenience, prefix–type functions, which require one or two arguments, display a left parenthesis when you enter them.

The prefix functions that require two arguments are %CHG, XROOT, IDIV, RMDR, nCr and nPr. Separate the two arguments with a comma.

✓ In an equation, the XROOT function takes its arguments in the opposite order from RPN usage. For example, –8 ENTER 3 $\sqrt[x]{y}$ to is equivalent to `XROOT(3,-8)`.

✓ All other two argument functions take their arguments in the Y, X order used for RPN. For example, 28 ENTER 4 ⬑ nCr is equivalent to `nCr(28,4)`.

For two argument functions, be careful if the second argument is negative. These are valid equations:

```
%CHG(-X,-2)

%CHG(X,(-Y))
```

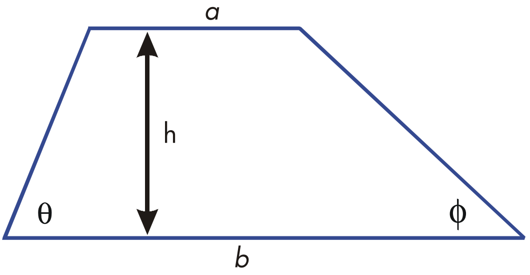
Eight of the equation functions have names that differ from their equivalent operations:

RPN Operation	Equation function
x^2	SQ
$\sqrt{x}$	SQRT
e^x	EXP
10^x	ALOG
$1/x$	INV
$\sqrt[x]{y}$	XROOT
y^x	^
INT÷	IDIV

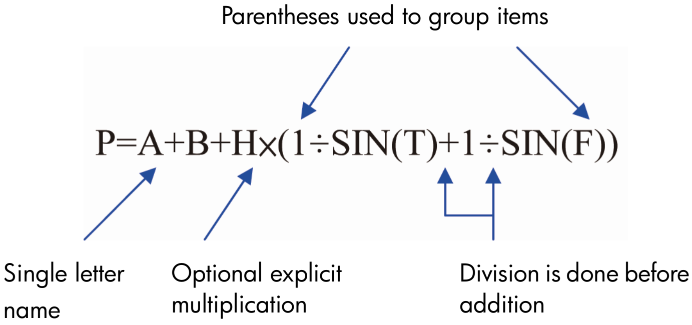
Example: Perimeter of a Trapezoid.

The following equation calculates the perimeter of a trapezoid. This is how the equation might appear in a book:

$$Perimeter = a + b + h\left(\frac{1}{\sin\theta} + \frac{1}{\sin\phi}\right)$$



The following equation obeys the syntax rules for HP 35s equations:


Parentheses used to group items
P=A+B+H×(1÷SIN(T)+1÷SIN(F))
Single letter name
Optional explicit multiplication
Division is done before addition

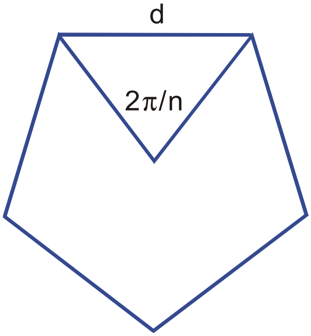
The next equation also obeys the syntax rules. This equation uses the inverse function, `INV(SIN(T))`, instead of the fractional form, `1÷SIN(T)`. Notice that the SIN function is "nested" inside the INV function. (INV is typed by 1/x.)

```
P=A+B+H×(INV(SIN(T))+INV(SIN(F)))
```

Example: Area of a Polygon.

The equation for area of a regular polygon with *n* sides of length *d* is:

$$Area = \frac{1}{4} n\, d^2 \frac{\cos(\pi / n)}{\sin(\pi / n)}$$



You can specify this equation as

```
A=0.25×N×D^2×COS(π÷N)÷SIN(π÷N)
```

Notice how the operators and functions combine to give the desired equation.

You can enter the equation into the equation list using the following keystrokes:

EQN RCL A ↰ = · 2 5 × RCL N × RCL D y^x 2 ×
COS ↰ π ÷ RCL N > ÷ SIN ↰ π ÷ RCL N ENTER

Syntax Errors

The calculator doesn't check the syntax of an equation until you evaluate the equation. If an error is detected, `SYNTAX ERROR` is displayed and the cursor is displayed at the first error location. You have to edit the equation to correct the error. (See "Editing and Clearing Equations" earlier in this chapter.)
By not checking equation syntax until evaluation, the HP 35s lets you create "equations" that might actually be messages. This is especially useful in programs, as described in chapter 13.

Verifying Equations

When you're viewing an equation — not while you're typing an equation — you can press ↰ SHOW to show you two things about the equation: the equation's checksum and its length. Hold the SHOW key to keep the values in the display.

The checksum is a four–digit hexadecimal value that uniquely identifies this equation. If you enter the equation incorrectly, it will not have this checksum. The length is the number of bytes of calculator memory used by the equation.

The checksum and length allow you to verify that equations you type are correct. The checksum and length of the equation you type in an example should match the values shown in this manual.

Example: Checksum and Length of an Equation.

Find the checksum and length for the pipe–volume equation at the beginning of this chapter.

Keys:	Display:	Description:
EQN	V=0.25×π×D^2×L	Displays the desired equation.
(^ as required)		

↰ SHOW (hold)	CK=49CA
	LN=14
		Display equation's checksum and length.

(release)	V=0.25×π×D^2×L	Redisplays the equation.
C		Leaves Equation mode.

7

Solving Equations

In chapter 6 you saw how you can use ENTER to find the value of the left–hand variable in an *assignment*–type equation. Well, you can use SOLVE to find the value of *any* variable in *any* type of equation.

For example, consider the equation

$$x^2 - 3y = 10$$

If you know the value of *y* in this equation, then SOLVE can solve for the unknown *x*. If you know the value of *x*, then SOLVE can solve for the unknown *y*. This works for "word problems" just as well:

$$Markup \times Cost = Price$$

If you know any two of these variables, then SOLVE can calculate the value of the third.

When the equation has only one variable, or when known values are supplied for all variables except one, then to solve for *x* is to find a *root* of the equation. A root of an equation occurs where an *equality* or *assignment* equation balances exactly, or where an *expression* equation equals zero.

Solving an Equation

To solve an equation (excluding built-in equations) for an unknown variable:

1. Press EQN and display the desired equation. If necessary, type the equation as explained in chapter 6 under "Entering Equations into the Equation List."

2. Press ⮫ SOLVE then press the key for the unknown variable. For example, press ⮫ SOLVE X to solve for *x*. The equation then prompts for a value for every other variable in the equation.
3. For each prompt, enter the desired value:

   - If the displayed value is the one you want, press R/S.
   - If you want a different value, type or calculate the value and press R/S. (For details, see "Responding to Equation Prompts" in chapter 6.)

You can halt a running calculation by pressing C or R/S.

When the root is found, it's stored in the relation variable, and the variable value is viewed in the display. In addition, the X–register contains the root, the Y–register contains the previous estimate value or Zero, and the Z–register contains the value of the root D-value(which should be zero).

For some complicated mathematical conditions, a definitive solution cannot be found — and the calculator displays NO ROOT FOUND. See "Verifying the Result" later in this chapter, and "Interpreting Results" and "When SOLVE Cannot Find a Root" in appendix D.

For certain equations it helps to provide one or *two initial guesses* for the unknown variable before solving the equation. This can speed up the calculation, direct the answer toward a realistic solution, and find more than one solution, if appropriate. See "Choosing Initial Guesses for Solve" later in this chapter.

Example: Solving the Equation of Linear Motion.

The equation of motion for a free–falling object is:

$$d = v_0\, t + {}^{1}/_{2}\, g\, t^{2}$$

where *d* is the distance, v_0 is the initial velocity, *t* is the time, and *g* is the acceleration due to gravity.

Type in the equation:

Keys:	Display:	Description:
↱ CLEAR 3 (3ALL)		Clears memory.
< (Y) ENTER		
EQN	3*3 lin. solve EQN LIST TOP	Selects Equation mode.
RCL D ↰ = RCL		Starts the equation.
V × RCL T +	D=VxT+_	
· 5 × RCL G ×	←=VxT+0.5xGxT^ 2_	
RCL T y^x 2		
ENTER	D=VxT+0.5xGxT^2	Terminates the equation and displays the left end.

↰ SHOW	CK=FB3C
	LN=15
		Checksum and length.

g (acceleration due to gravity) is included as a variable so you can change it for different units (9.8 m/s^2 or 32.2 ft/s^2).

Calculate how many meters an object falls in 5 seconds, starting from rest. Since Equation mode is turned on and the desired equation is already in the display, you can start solving for *D*:

Keys:	Display:	Description:
↱ SOLVE	SOLVE_	Prompts for unknown variable.
D	V? *value*	Selects *D*; prompts for *V*.
0 R/S	T? *value*	Stores 0 in *V*; prompts for T.
5 R/S	G? *value*	Stores 5 in *T*; prompts for *G*.
9 · 8 R/S	SOLVING D= 122.5000	Stores 9.8 in *G*; solves for *D*.

Try another calculation using the same equation: how long does it take an object to fall 500 meters from rest?

Keys:	Display:	Description:
EQN	D=V×T+0.5×G×T^2	Displays the equation.
↱ SOLVE T	D? 122.5	Solves for *T*; prompts for *D*.
5 0 0 R/S	V? 0	Stores 500 in *D*; prompts for *V*.
R/S	G? 9.8	Retains 0 in *V*; prompts for *G*.
R/S	SOLVING T= 10.1015	Retains 9.8 in *G*; solves for *T*.

Example: Solving the Ideal Gas Law Equation.

The Ideal Gas Law describes the relationship between pressure, volume, temperature, and the amount (moles) of an ideal gas:

$$P \times V = N \times R \times T$$

where *P* is pressure (in atmospheres or N/m^2), *V* is volume (in liters), *N* is the number of moles of gas, *R* is the universal gas constant (0.0821 liter–atm/mole–K or 8.314 J/mole–K), and T is temperature (Kelvins: K=°C + 273.1).

Enter the equation:

Keys:	Display:	Description:
EQN RCL P ×	P×_	Selects Equation mode and starts the equation.
RCL V ↰ =		
RCL N ×		
RCL R × RCL T	P×V=N×R×T_	
ENTER	P×V=N×R×T	Terminates and displays the equation.

↰ SHOW	CK=EDC8
	LN=9
		Checksum and length.

A 2–liter bottle contains 0.005 moles of carbon dioxide gas at 24°C. Assuming that the gas behaves as an ideal gas, calculate its pressure. Since Equation mode is turned on and the desired equation is already in the display, you can start solving for *P*:

Keys:	Display:	Description:
↱ SOLVE P	V? *value*	Solves for *P*; prompts for *V*.
2 R/S	N? *value*	Stores 2 in *V*; prompts for *N*.
· 0 0 5 R/S	R? *value*	Stores .005 in *N*; prompts for *R*.
· 0 8 2 1 R/S	T? *value*	Stores .0821 in *R*; prompts for *T*.
2 4 + 2 7 3 · 1 ENTER	T? 297.1000	Calculates *T* (Kelvins).
R/S	SOLVING P= 0.0610	Stores 297.1 in *T*; solves for *P* in atmospheres.

A 5–liter flask contains nitrogen gas. The pressure is 0.05 atmospheres when the temperature is 18°C. Calculate the density of the gas (*N* × 28/*V*, where 28 is the molecular weight of nitrogen).

Keys:	Display:	Description:
EQN	PxV=NxRxT	Displays the equation.
↱ SOLVE N	P? 0.0610	Solves for *N*; prompts for P.
· 0 5 R/S	V? 2.0000	Stores .05 in *P*; prompts for *V*.
5 R/S	R? 0.0821	Stores 5 in *V*; prompts for *R*.
R/S	T? 297.1000	Retains previous *R*; prompts for *T*.
✓ 1 8 ENTER 2 7 3 · 1 +	T? 291.1000	Calculates *T* (Kelvins).

R/S	SOLVING N= 0.0105	Stores 291.1 in *T*; solves for *N*.
✓ 2 8 ×	0.2929	Calculates mass in grams, *N* × 28.
✓ RCL V ÷	0.0586	Calculates density in grams per liter.

Solving built-in Equation

The built-in equations are: “2*2 lin. solve” (Ax+By=C, Dx+Ey=F) and “3*3 lin. Solve”(Ax+By+Cz=D, Ex+Fy+Gz=H, Ix+Jy+Kz=L). If you select one of them, the XEQ, ENTER and ∫ key will have no effect. Pressing the ↱ SOLVE will request 6 variables (A to F) for the 2*2 case or 12 variables (A to L) for the 3*3 case, and use them to find x, y for a 2*2 linear equation system or x, y and z for a 3*3 linear equation system. The result will be saved in variables x, y, and z. The calculator can detect cases with infinitely many solutions or no solutions.

Example: solve the x,y in simultaneous equations $\begin{cases} x+2y=5 \\ 3x+4y=11 \end{cases}$

Keys:	Display:	Description:
EQN	3*3 lin. solve EQN LIST TOP	Enters equation mode.
⌄	EQN LIST TOP 2*2 lin. solve	Displays the built-in equation
↱ SOLVE	A? value	Prompts for *A*.
1 R/S	B? value	Stores 1 in *A*; prompts for *B*.
2 R/S	C? value	Stores 2 in *B*; prompts for *C*.
5 R/S	D? value	Stores 5 in *C*; prompts for *D*.
3 R/S	E? value	Stores 3 in *D*; prompts for *E*.

4 R/S	F? value		Stores 4 in *E* ;prompts for *F*.
1 1 R/S	X= 1.0000	⬆ ⬇	Stores 11 in *F* and calculates x and y.
⌄	y= 2.0000	⬆ ⬇	value of y

Understanding and Controlling SOLVE

SOLVE first attempts to solve the equation directly for the unknown variable. If the attempt fails, SOLVE changes to an iterative (repetitive) procedure. The procedure starts by evaluating the equation using two initial guesses for the unknown variable. Based on the results with those two guesses, SOLVE generates another, better guess. Through successive iterations, SOLVE finds a value for the unknown that makes the value of the equation equal to zero.

When SOLVE evaluates an equation, it does it the same way XEQ does — any "=" in the equation is treated as a " – ". For example, the Ideal Gas Law equation is evaluated as $P \times V - (N \times R \times T)$. This ensures that an *equality* or *assignment* equation balances at the root, and that an *expression* equation equals zero at the root.

Some equations are more difficult to solve than others. In some cases, you need to enter initial guesses in order to find a solution. (See "Choosing Initial Guesses for SOLVE," below.) If SOLVE is unable to find a solution, the calculator displays NO ROOT FND.

See appendix D for more information about how SOLVE works.

Verifying the Result

After the SOLVE calculation ends, you can verify that the result is indeed a solution of the equation by reviewing the values left in the stack:

- The X–register (press C to clear the viewed variable) contains the solution (root) for the unknown; that is, the value that makes the evaluation of the equation equal to zero.

- ✓ The Y–register (press R↓) contains the previous estimate for the root or equals to zero. This number should be the same as the value in the X–register. If it is not, then the root returned was only an *approximation*, and the values in the X– and Y–registers bracket the root. These bracketing numbers should be close together.
- ✓ The Z– register (press R↓ again) contains D-value of the equation at the root. For an exact root, this should be zero. If it is not zero, the root given was only an *approximation*; this number should be close to zero.

If a calculation ends with the `NO ROOT FND`, the calculator could not converge on a root. (You can see the value in the X–register — the final estimate of the root — by pressing C or ← to clear the message.) The values in the X– and Y–registers bracket the interval that was last searched to find the root. The Z–register contains the value of the equation at the final estimate of the root.

- If the X– and Y–register values aren't close together, or the Z–register value isn't close to zero, the estimate from the X–register probably isn't a root.
- If the X– and Y–register values *are* close together, and the Z–register value *is* close to zero, the estimate from the X–register may be an approximation to a root.

Interrupting a SOLVE Calculation

To halt a calculation, press C or R/S, and the message "`INTERRUPTED`" will be shown. The current best estimate of the root is in the unknown variable; use ↰ VIEW to view it without disturbing the stack, but solving cannot be resumed.

Choosing Initial Guesses for SOLVE

The two initial guesses come from:

- The number currently stored in the unknown variable.
- The number in the X–register (the display).

These sources are used for guesses *whether you enter guesses or not*. If you enter only one guess and store it in the variable, the second guess will be the same value since the display also holds the number you just stored in the variable. (If such is the case, the calculator changes one guess slightly so that it has two different guesses.)

Entering your own guesses has the following advantages:

- By narrowing the range of search, guesses can reduce the time to find a solution.
- If there is more than one mathematical solution, guesses can direct the SOLVE procedure to the desired answer or range of answers. For example, the equation of linear motion

  $$d = v_0 t + {}^1/_2\, gt^2$$

  can have two solutions for *t*. You can direct the answer to the required solution by entering appropriate guesses.

  The example using this equation earlier in this chapter didn't require you to enter guesses before solving for *T* because in the first part of that example you stored a value for *T* and solved for *D*. The value that was left in *T* was a good (realistic) one, so it was used as a guess when solving for *T*.
- If an equation does not allow certain values for the unknown, guesses can prevent these values from occurring. For example,

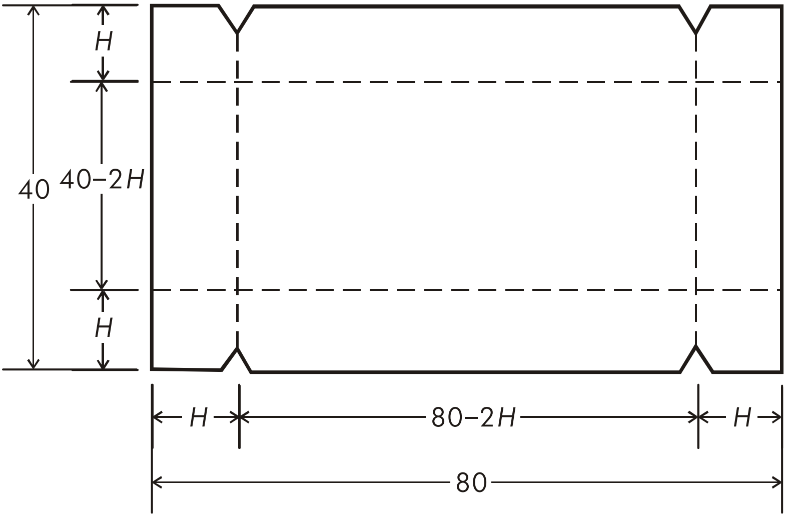
  $$y = t + \log x$$

  results in an error if $x \leq 0$ (message `NO ROOT FND`).

In the following example, the equation has more than one root, but guesses help find the desired root.

Example: Using Guesses to Find a Root.

Using a rectangular piece of sheet metal 40 cm by 80 cm, form an open–top box having a volume of 7500 cm^3. You need to find the height of the box (that is, the amount to be folded up along each of the four sides) that gives the specified volume. A *taller* box is preferred to a *shorter* one.



If H is the height, then the length of the box is $(80 - 2H)$ and the width is $(40 - 2H)$. The volume V is:

$$V = (80 - 2H) \times (40 - 2H) \times H$$

which you can simplify and enter as

$$V = (40 - H) \times (20 - H) \times 4 \times H$$

Type in the equation:

Keys:	Display:	Description:
EQN RCL V ↰ =	`V=_`	Selects Equation mode and starts the equation
() 4 0 − RCL H >	`V=(40-H)_`	

× ( ) 2 0 −		
RCL H >	(40-H)x(20-H)_	
× 4 × RCL H	H)x(20-H)x4xH_	
ENTER	V=(40-H)x(20-H	Terminates and displays the equation.

↰ SHOW	CK=49A4
	LN=19
		Checksum and length.

It seems reasonable that either a tall, narrow box or a short, flat box could be formed having the desired volume. Because the taller box is preferred, larger initial estimates of the height are reasonable. However, heights greater than 20 cm are not physically possible because the metal sheet is only 40 cm wide. Initial estimates of 10 and 20 cm are therefore appropriate.

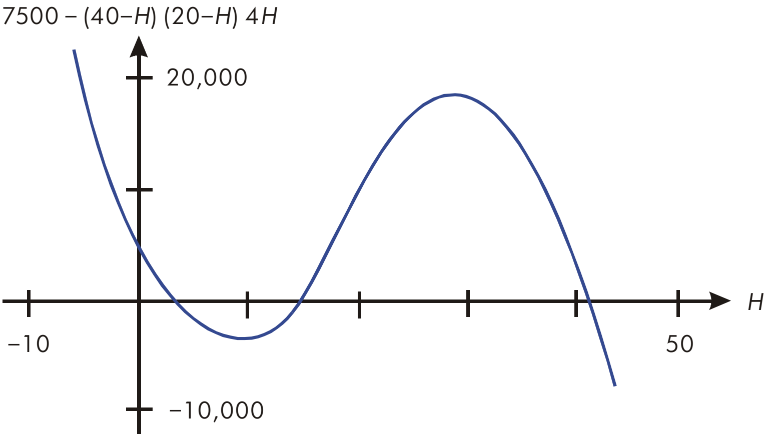
Keys:	Display:	Description:
C		Leaves Equation mode.
1 0 ↱ STO H		Stores lower and upper limit guesses.
ENTER 2 0	20_	
EQN	V=(40-H)x(20-H	Displays current equation.
↱ SOLVE H	V? *value*	Solves for *H*; prompts for *V*.
7 5 0 0 R/S	H= 15.0000	Stores 7500 in *V*; solves for *H*.

Now check the quality of this solution — that is, whether it returned an exact root — by looking at the value of the previous estimate of the root (in the Y–register) and the value of the equation at the root (in the Z–register).

Keys:	Display:	Description:
✓ R↓	15.0000	This value from the Y–register is the estimate made just prior to the final result. Since it is the same as the solution, the solution is an exact root.
✓ R↓	0.0000	This value from the Z–register shows the equation equals zero at the root.

The dimensions of the desired box are 50 × 10 × 15 cm. If you ignored the upper limit on the height (20 cm) and used initial estimates of 30 and 40 cm, you would obtain a height of 42.0256 cm — a root that is physically meaningless. If you used small initial estimates such as 0 and 10 cm, you would obtain a height of 2.9774 cm — producing an undesirably short, flat box.

If you don't know what guesses to use, you can use a graph to help understand the behavior of the equation. Evaluate your equation for several values of the unknown. For each point on the graph, display the equation and press XEQ — at the prompt for *x* enter the *x–coordinate,* and then obtain the corresponding value of the equation, the *y–coordinate.* For the problem above, you would always set *V* = 7500 and vary the value of *H* to produce different values for the equation. Remember that the value for this equation is the *difference* between the left and right sides of the equation. The plot of the value of this equation looks like this.



For More Information

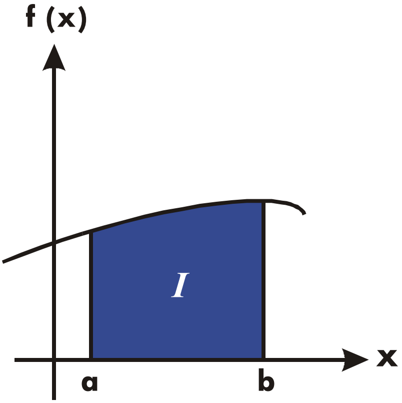
This chapter gives you instructions for solving for unknowns or roots over a wide range of applications. Appendix D contains more detailed information about how the algorithm for SOLVE works, how to interpret results, what happens when no solution is found, and conditions that can cause incorrect results.

8

Integrating Equations

Many problems in mathematics, science, and engineering require calculating the definite integral of a function. If the function is denoted by *f(x)* and the interval of integration is *a* to *b*, then the integral can be expressed mathematically as

$$I = \int_a^b f(x)\,dx$$



The quantity *I* can be interpreted geometrically as the area of a region bounded by the graph of the function *f(x)*, the *x*–axis, and the limits *x* = *a* and *x* = *b* (provided that *f(x)* is nonnegative throughout the interval of integration).

The ∫ operation (∫ FN) integrates the current equation with respect to a specified variable (∫ FN d_). The function may have more than one variable.

Integrating Equations (∫ FN)

To integrate an equation:

1. If the equation that defines the integrand's function isn't stored in the equation list, key it in (see "Entering Equations into the Equation List" in chapter 6) and leave Equation mode. The equation usually contains just an expression.
2. ✓ Enter the limits of integration: key in the *lower* limit and press ENTER, then key in the upper limit.
3. Display the equation: Press EQN and, if necessary, scroll through the equation list (press ⌃ or ⌄) to display the desired equation.
4. Select the variable of integration: Press ⬑ ∫ *variable.* This starts the calculation.

∫ uses far more memory than any other operation in the calculator. If executing ∫ causes a MEMORY FULL message, refer to appendix B.

You can halt a running integration calculation by pressing C or R/S,and the message "INTERRUPTED" will be shown in line 2, but the integration cannot be resumed. However, no information about the integration is available until the calculation finishes normally.

The display format setting affects the level of accuracy assumed for your function and used for the result. The integration is more precise but takes *much* longer in the ALL and higher FIX, SCI, and ENG settings. The *uncertainty* of the result ends up in the Y–register, pushing the limits of integration up into the T– and Z–registers. For more information, see "Accuracy of Integration" later in this chapter.

To integrate the same equation with different information:

✓ If you use the same limits of integration, press R↓ R↓ move them into the X– and Y–registers. Then start at step 3 in the above list. If you want to use different limits, begin at step 2.

To work another problem using a different equation, start over from step 1 with an equation that defines the integrand.

Example: Bessel Function.

The Bessel function of the first kind of order 0 can be expressed as

$$J_0(x) = \frac{1}{\pi}\int_0^{\pi} \cos(x \sin t)dt$$

Find the Bessel function for *x*–values of 2 and 3.

Enter the expression that defines the integrand's function:

$$\cos\ (x \sin\ t\ )$$

Keys:	Display:	Description:
↱ CLEAR 3		Clears memory.
(3ALL) < (Y) ENTER		
EQN	3*3 lin. solve EQN LIST TOP	Selects Equation mode.
COS RCL X	COS(X)	Types the equation.
× SIN	COS(XxSIN()	
RCL T	COS(XxSIN(T))	
> >	COS(XxSIN(T))_	
ENTER	COS(XxSIN(T))	Terminates the expression and displays its left end.

↰ SHOW	CK=E1EC
	LN=13
		Checksum and length.

C		Leaves Equation mode.

Now integrate this function with respect to t from zero to π ; *x* = 2.

Keys:	Display:	Description:
MODE 2 (2RAD)		Selects Radians mode.
✓ 0 ENTER ↰ π	3.1416	Enters the limits of integration (lower limit first).
EQN	COS(XxSIN(T))	Displays the function.
↰ ∫	∫FN d_	Prompts for the variable of integration.

T	X? *value*	Prompts for value of *X*.
2 R/S	INTEGRATING ∫ = 0.7034	x = 2. Starts integrating; calculates result for $\int_0^{\pi} f(t)$
✓ ↰ π ÷	0.2239	The final result for J_0 (2).

Now calculate J_0(3) with the same limits of integration. You must re-specify the limits of integration (0, π) since they were pushed off the stack by the subsequent division by π.

Keys:	Display:	Description:
✓ 0 ENTER ↰ π	3.1416	Enters the limits of integration (lower limit first).
EQN	COS(X×SIN(T))	Displays the current equation.
↰ ∫	∫FN d_	Prompts for the variable of integration.
T	X? 2.0000	Prompts for value of *X*.
3 R/S	INTEGRATING ∫ = -0.8170	x = 3. Starts integrating and calculates the result for $\int_0^{\pi} f(t)$.
✓ ↰ π ÷	-0.2601	The final result for J_0(3).

✓ **Example: Sine Integral.**

Certain problems in communications theory (for example, pulse transmission through idealized networks) require calculating an integral (sometimes called the *sine* integral) of the form

$$S_i(t) = \int_0^t (\frac{\sin x}{x})dx$$

Find *Si* (2).

Enter the expression that defines the integrand's function:

$$\frac{\sin x}{x}$$

If the calculator attempted to evaluate this function at $x = 0$, the lower limit of integration, an error (DIVIDE BY 0) would result. However, the integration algorithm normally does *not* evaluate functions at either limit of integration, unless the endpoints of the interval of integration are extremely close together or the number of sample points is extremely large.

Keys:	Display:	Description:
EQN	3*3 lin. solve EQN LIST TOP	Selects Equation mode.
SIN RCL X	SIN(X)	Starts the equation.
⟩	SIN(X)_	The closing right parenthesis is required in this case.
÷ RCL X	SIN(X)÷X_	
ENTER	SIN(X)÷X	Terminates the equation.

⬑ SHOW	CK=0EE0
	LN=8
		Checksum and length.

C		Leaves Equation mode.

Now integrate this function with respect to *x* (that is, *X*) from zero to 2 ($t = 2$).

Keys:	Display:	Description:
MODE 2 (2RAD)		Selects Radians mode.
0 STO X ENTER 2	2_	Enters limits of integration (lower first).
EQN	SIN(X)÷X	Displays the current equation.
⬑ ∫ X	INTEGRATING ∫ = 1.6054	Calculates the result for *Si*(2).

Accuracy of Integration

Since the calculator cannot compute the value of an integral exactly, it *approximates* it. The accuracy of this approximation depends on the accuracy of the integrand's function itself, as calculated by your equation. This is affected by round–off error in the calculator and the accuracy of the empirical constants.

Integrals of functions with certain characteristics such as spikes or very rapid oscillations *might* be calculated inaccurately, but the likelihood is very small. The general characteristics of functions that can cause problems, as well as techniques for dealing with them, are discussed in appendix E.

Specifying Accuracy

The display format's setting (FIX, SCI, ENG, or ALL) determines the *precision* of the integration calculation: the greater the number of digits displayed, the greater the precision of the calculated integral (and the greater the time required to calculate it). The fewer the number of digits displayed, the faster the calculation, but the calculator will presume that the function is accurate to the only number of digits specified.

To specify the *accuracy* of the integration, set the display format so that the display shows *no more than* the number of digits that you consider accurate *in the integrand's values*. This same level of accuracy and precision will be reflected in the result of integration.

If Fraction–display mode is on (flag 7 set), the accuracy is specified by the previous display format.

Interpreting Accuracy

After calculating the integral, the calculator places the estimated *uncertainty* of that integral's result in the Y–register. Press x◂▸y to view the value of the uncertainty.

For example, if the integral *Si(2)* is 1.6054 ± 0.0002, then 0.0002 is its uncertainty.

Example: Specifying Accuracy.

With the display format set to SCI 2, calculate the integral in the expression for *Si(2)* (from the previous example).

Keys:	Display:	Description:
↰ DISPLAY 2 (2SCI) 2	1.61E0	Sets scientific notation with two decimal places, specifying that the function is accurate to two decimal places.
✓ R↓ R↓	0.00E0 2.00E0	Rolls down the limits of integration from the Z–and T–registers into the X–and Y–registers.
EQN	SIN(X)÷X	Displays the current Equation.
↰ ∫ X	INTEGRATING ∫ = 1.61E0	The integral approximated to two decimal places.
x↔y	1.61E-2	The uncertainty of the approximation of the integral.

The integral is 1.61±0.0161. Since the uncertainty would not affect the approximation until its third decimal place, you can consider all the displayed digits in this approximation to be accurate.

If the uncertainty of an approximation is larger than what you choose to tolerate, you can increase the number of digits in the display format and repeat the integration (provided that *f(x)* is still calculated accurately to the number of digits shown in the display), In general, the uncertainty of an integration calculation decreases by a factor of ten for each additional digit, specified in the display format.

Example: Changing the Accuracy.

For the integral of *Si(2)* just calculated, specify that the result be accurate to four decimal places instead of only two.

Keys:	Display:	Description:
↰ DISPLAY 2 (2SCI) 4	1.6079E−2	Specifies accuracy to four decimal places. The uncertainty from the last example is still in the display.
✓ R↓ R↓	0.0000E0 2.0000E0	Rolls down the limits of integration from the Z– and T–registers into the X– and Y–registers.
EQN	SIN(X)÷X	Displays the current equation.
↰ ∫ X	INTEGRATING ∫ = 1.6054E0	Calculates the result.
x↔y	1.6056E−4	Note that the uncertainty is about 1/100 as large as the uncertainty of the SCI 2 result calculated previously.
DISPLAY 1 (2SCI) 4	0.0002	Restores FIX 4 format.
MODE 1 (1DEG)	0.0002	Restores Degrees mode.

This uncertainty indicates that the result *might* be correct to only three decimal places. In reality, this result is accurate to *seven* decimal places when compared with the actual value of this integral. Since the uncertainty of a result is calculated conservatively, *the calculator's approximation in most cases is more accurate than its uncertainty indicates.*

For More Information

This chapter gives you instructions for using integration in the HP 35s over a wide range of applications. Appendix E contains more detailed information about how the algorithm for integration works, conditions that could cause incorrect results and conditions that prolong calculation time, and obtaining the current approximation to an integral.

9

Operations with Complex Numbers

The HP 35s can use complex numbers in the form

x·i·y x+y·i rθa

It has operations for complex arithmetic (+, –, ×, ÷), complex trigonometry (sin, cos, tan), and the mathematics functions –*z*, 1/*z*, z_1^{z2} , ln *z*, and *e* ^{z}. (where z $_1$ and z $_2$ are complex numbers).

The form, x+yi, is only available in ALG mode.

To enter a complex number:

Form: x·i·y

1. Type the real part.
2. Press [i].
3. Type the *imaginary* part.

Form: x+y·i

1. Type the real part.
2. Press[+]
3. Type the *imaginary* part.
4. Press [i].

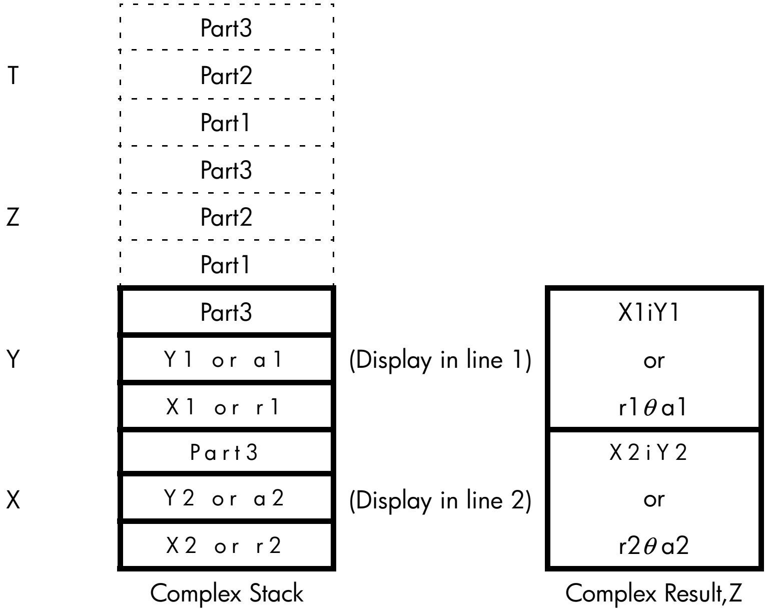
Form: rθa

1. Type the value of r.
2. Press[↱] [θ].
3. Type the *value of θ*.

The examples in this chapter all utilize RPN mode unless otherwise noted.

The Complex Stack

A complex number occupies part 1 and part 2 of a stack level. In RPN mode, the complex number occupying part 1 and part 2 of the X-register is displayed in line 2, while the complex number occupying part 1 and part 2 of the Y-register is displayed in line 1.



Complex Operations

Use the complex operations as you do real operations in ALG and RPN mode.

To do an operation with one complex number:

1. Enter the complex number *z* as described before.
2. Select the complex function.

Functions for One Complex Number, z

To Calculate:	Press:
Change sign, $-z$	+/_
Inverse, $1/z$	1/x
Natural log, ln z	↱ LN
Natural antilog, e^z	↱ e^x
Sin z	SIN
Cos z	COS
Tan z	TAN
Absolute value, ABS(z)	↱ ABS
Argument value, ARG(z)	↰ ARG

✓ To do an arithmetic operation with two complex numbers:

1. Enter the first complex number, z_1 as described before.
2. Enter the second complex number z_2 as described before.
3. Select the arithmetic operation:

Arithmetic With Two Complex Numbers, z_1 and z_2

To Calculate:	Press:
Addition, $z_1 + z_2$	+
Subtraction, $z_1 - z_2$	−
Multiplication, $z_1 \times z_2$	×
Division, $z_1 \div z_2$	÷
Power function, $z_1^{z_2}$	y^x

Examples:

Here are some examples of trigonometry and arithmetic with complex numbers:

Evaluate sin (2i3)

Keys:	Display:	Description:
↰ DISPLAY 9 (9×iy)		Sets display format.
2 i 3 SIN	9.1545i-4.1689	Result is 9.1545 *i* –4.1689.

Evaluate the expression

$$z_1 \div (z_2 + z_3),$$

where z_1 = 23 *i* 13, z_2 = –2i1 z_3 = 4 *i*– 3

Perform the calculation as

Keys:	Display:	Description:
↰ DISPLAY 9 (9×iy)		Sets display format
2 3 i 1 3 ENTER	23.0000i13.0000	ENTER z1
	23.0000i13.0000	
2 +/- i 1 ENTER	-2.0000i1.0000	ENTER z2
	-2.0000i1.0000	
4 i 3 +/- +	23.0000i13.0000	($z_2 + z_3$). Result is 2 *i* -2.
	2.0000i-2.0000	
÷	2.5000i.9000	$z_1 \div (z_2 + z_3)$. Result is 2.5 *i* 9.

Evaluate (4 *i* –2/5) × (3 *i* –2/3).

Keys:	Display:	Description:
↰ DISPLAY 9 (9×iy)		Sets display format
4 i · 2 · 5 +/-	4.0000i-0.4000	Enters 4i-2/5
ENTER	4.0000i-0.4000	

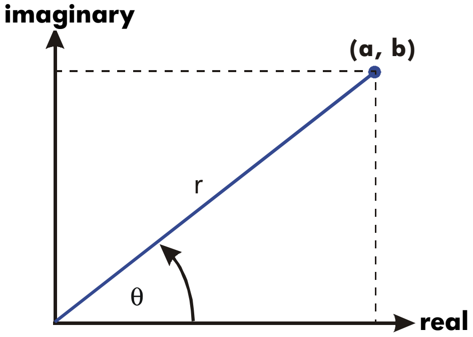
3 i · 2 · 3 +/-	4.0000i-0.4000 3i-0 2/3	Enters 3i-2/3
×	11.7333i-3.8667	Result is 11.7333i-3.8667

Evaluate $e^{z^{-2}}$, where z = (1*i* 1).

Keys:	Display:	Description:
1 i 1 ENTER	1.0000i1.0000 1.0000i1.0000	ENTER 1i1Intermediate result of
2 +/- y^x	0.0000i-5.0000	Z−2,result is 0i-5
↱ e^x	0.8776i-0.4794	Final result is 0.8776 *i*– 0.4794.

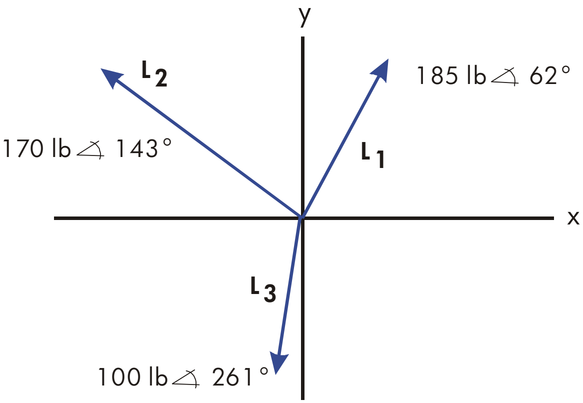
Using Complex Numbers in Polar Notation

Many applications use real numbers in *polar* form or *polar* notation. These forms use pairs of numbers, as do complex numbers, so you can do arithmetic with these numbers by using the complex operations.



Example: Vector Addition.

Add the following three loads.



Keys:	Display:	Description:
MODE 1 (1DEG)		Sets Degrees mode.
↰ DISPLAY · 0 (10rθa)		Sets complex mode
1 8 5 ↱ θ	185.0000θ62.0000	Enters L_1
6 2 ENTER	185.0000θ62.0000	
1 7 0 ↱ θ	170.0000θ143.00...	Enters L_2.
1 4 3 ENTER	170.0000θ143.000➡	
1 0 0 ↱ θ	185.0000θ62.0000	Enters L_3 and adds $L_2 + L_3$
2 6 1 +	151.4529θ178.660➡	
+	178.9372θ111.148➡	Adds $L_1 + L_2 + L_3$.
↱ >	⬅9	Scrolls the screen to see the rest of the answer

You can do a complex operation with numbers whose complex forms are different; however, the result form is dependent on the setting in DISPLAY menu.

Evaluate 1i1+3θ10+5θ30

Keys:	Display:	Description:
MODE 1 (1DEG)		Sets Degrees mode.
↰ DISPLAY · 0 (10rθa)		Sets complex mode
1 i 1 ENTER	1.4142θ45.0000	Enters 1i1
	1.4142θ45.0000	
3 ↱ θ 1 0	3.0000θ10.0000	Enters 3θ10
ENTER	3.0000θ10.0000	
5 ↱ θ 3 0	1.4142θ45.0000	Enters 5θ30 and adds 3θ 10
+	7.8861θ22.5241	
+	9.2088θ25.8898	Adds 1i1,result is 9.2088θ 25.8898

Complex Numbers in Equations

You can type complex numbers in equations. When an equation is displayed, all numeric forms are shown as they were entered, like xiy, or rθ a

When you evaluate an equation and are prompted for variable values, you may enter complex numbers. The values and format of the result are controlled by the display setting. This is the same as calculating in ALG mode.

Equations that contain complex numbers can be solved and integrated.

Complex Number in Program

In a program, you can type a complex number. For example, 1i2+3θ 10+5

θ 30 in program is:

Program lines: (ALG mode)	Description
F001 LBL F	Begins the program
F002 1i2+3θ10+5θ30	
F003 RTN	

When you are running a program and are prompted for values by INPUT instructions, you can enter complex numbers. The values and format of the result are controlled by the display setting.

The program that contains the complex number can also be solved and integrated.

10

Vector Arithmetic

From a mathematical point of view, a vector is an array of 2 or more elements arranged into a row or a column.

Physical vectors that have two or three components and can be used to represent physical quantities such as position, velocity, acceleration, forces, moments, linear and angular momentum, angular velocity and acceleration, etc.

To enter a vector:

1. Press [↱] [[]]
2. Enter the first number for the vector.
3. Press [↰] [,] and enter a second number for a 2-D or 3-D vector.
4. Press [↰] [,] and enter a third number for a 3-D vector.

The HP 35s cannot handle vectors with more than 3 dimensions.

Vector operations

✓ **Addition and subtraction:**

The addition and subtraction of vectors require that two vector operands have the same length. Attempting to add or subtract vectors of different length produces the error message “`INVALID DATA`”.

1. Enter the first vector
2. Enter the second vector
3. Press [+] or [−]

✓ Calculate [1.5,-2.2]+[-1.5,2.2]

Keys:	Display:	Description:
MODE 5 (5RPN)		Switches to RPN mode(if necessary)
↱ [] 1 · 5 ↰	[1.5000,-2.2000]	Enters [1.5,-2.2]
, +/- 2 · 2	[1.5000,-2.2000]	
ENTER		
↱ [] +/- 1 · 5	[1.5000,-2.2000]	Enters [-1.5,2.2]
↰ , 2 · 2	[-1.5,2.2]	
+	0.0000 [0.0000,0.0000]	Adds two vectors

Calculate [-3.4,4.5]-[2.3,1.4]

Keys:	Display:	Description:
MODE 4 (4ALG)		Switches to ALG mode
↱ [] +/- 3 · 4	[-3.4,4.5]_	Enters [-3.4,4.5]
↰ , 4 · 5 >		
− ↱ [] 2 · 3	◀3.4,4.5]-[2.3,1.4]	Enters [2.3,1.4]
↰ , 1 · 4		
ENTER	[-3.4,4.5]-[2.3,... [-5.7000,3.1000]	Subtracts two vectors

✓ **Multiplication and divisions by a scalar:**

1. Enter a vector
2. Enter a scalar
3. Press × for multiplication or ÷ for division

✓ Calculate [3,4]x5

Keys:	Display:	Description:
MODE 5 (5RPN)		Switches to RPN mode
↱ [] 3 ↰ , 4	[3.0000,4.0000]	Enters [3,4]
ENTER	[3.0000,4.0000]	
5	[3.0000,4.0000] 5_	Enters 5 as a scalar
×	0.0000 [15.0000,20.0000]	Performs multiplication

Calculate [-2,4]÷2

Keys:	Display:	Description:
MODE 4 (4ALG)		Switches to ALG mode
↱ [] +/- 2 ↰	[-2,4]_	Enters [-2,4]
, 4 >		
÷ 2	[-2,4]÷2	Enters 5 as a scalar
ENTER	[-2,4]÷2 [-1.0000,2.0000]	Performs division

Absolute value of the vector

The absolute value function “ABS”, when applied to a vector, produces the magnitude of the vector. For a vector A=(A1, A2, …An), the magnitude is defined as $\left|A\right| = \sqrt{A_1{}^2 + A_2{}^2 + \cdots + A_n{}^2}$.

1. Press ↱ ABS
2. Enter a vector
3. Press ENTER

For example: Absolute value of vector [5,12]:

↱ ABS ↱ [] 5 ↰ , 1 2 ENTER .The answer is 13. In RPN mode: MODE 5 (5RPN) ↱ [] 5 ↰ , 1 2 ↱ ABS .

Dot product

Function DOT is used to calculate the dot product of two vectors with the same length. Attempting to calculate the dot product of two vectors of different length causes an error message “INVALID DATA”.

For 2-D vectors: [A, B], [C, D], dot product is defined as [A, B]▪[C, D]= A x C +B x D.

For 3-D vectors: [A, B, X], [C, D,Y], dot product is defined as [A, B, X]▪[C, D, Y]= A x C +B x D+X x Y

1. Enter the first Vector
2. Press ×
3. Enter the second vector
4. Press ENTER

Note: The sign, × ,here means ”dot product” instead of “cross product”. For cross product, see chapter 17.

Calculate the dot product of two vectors, [1,2] and [3,4]

Keys:	Display:	Description:
MODE 4 (4ALG)		Switches to ALG mode
↱ [] 1 ↰ , 2 >	[1,2]_	Enters the first vector [1,2]
× ↱ [] 3 ↰ , 4	[1,2]×[3,4]	Executes × for dot product, and enters the second vector
ENTER	11.0000	The dot product of two vectors is 11

✓ Calculate the dot product of two vectors, [9,5] and [2.2]

Keys:	Display:	Description:
MODE 5 (5RPN)		Switches to RPN mode
↱ [] 9 ↰ , 5	[9.0000,5.0000]	Enters the first vector [9,5]
ENTER	[9.0000,5.0000]	
↱ [] 2 ↰ , 2	[9.0000,5.0000] [2,2]	and enters the second vector [2,2]

×	28.0000	Presses × for dot product ,and the dot product of two vectors is 28

Angle between vectors

The angle between two vectors, A and B, can be found as $\theta =$ ACOS(A▪B/$|A||B|$)

Find the angle between two vectors: A=[1,0],B=[0,1]

Keys:	Display:	Description:
MODE 4(4ALG)		Switches to ALG mode
MODE 1 (1DEG)		Sets Degrees mode
↱ ACOS	ACOS()	Arc cosine function
↱ [] 1 ↰ , 0 >	ACOS([1,0])	Enters vector A [1,0]
× ↱ [] 0 ↰ , 1 >	ACOS([1,0]×[0,1])	Enters vector B [0,1] for dot product of A and B
÷ ↱ ABS ↱ [] 1 ↰ , 0 >	⬅,1]÷ABS([1,0])➡	Finds the magnitude of vector A [1,0]
÷ ↱ ABS ↱ [] 0 ↰ , 1	⬅1,0]÷ABS([0,1])➡	Finds the magnitude of vector B [0,1]
ENTER	ACOS([1,0]×[0... 90.0000	The angle between two vectors is 90

✓ Find the angle between two vectors: A=[3,4],B=[0,5]

Keys:	Display:	Description:
MODE 5(5RPN)		Switches to RPN mode
MODE 1(1DEG)		Sets Degrees mode
↱ [] 3 ↰ , 4	90	Finds the dot product of two vectors
ENTER ↱ [] 0 ↰ , 5 ×	20.0000	
↱ [] 3 ↰ , 4	20.0000	Finds the magnitude of vector [3,4]
↱ ABS	5.0000	

↱ [ ] 0 ↰ , 5	5.0000	Finds the magnitude of vector [0,5]
↱ ABS	5.0000	
×	20.0000 25.0000	Multiplies two vectors
÷	90 0.8000	Divides two values
↱ ACOS	90 36.8699	The angle between two vectors is 36.8699

Vectors in Equations

Vectors can be used in equations and in equation variables exactly like real numbers. A vector can be entered when prompted for a variable.

Equations containing vectors can be solved, however the solver has limited ability if the unknown is a vector.

Equations containing vectors can be integrated, however the result of the equation must be a real or a 1-D vector or a vector with 0 as the 2nd and 3rd elements.

Vectors in Programs

Vectors can be used in program in the same way as real and complex numbers

For example, [5, 6] +2 x [7, 8] x [9, 10] in a program is:

Program lines:	Description:
G0001 LBL G	Begins the program
G0002 [5,6] + 2 x [7,8] x[9,10]	[5,6]
G0003 RTN	

A vector can be entered when prompted for a value for a variable. Programs that contain vectors can be used for solving and integrating.

Creating Vectors from Variables or Registers

It is possible to create vectors containing the contents of memory variables, stack registers, or values from the indirect registers, in run or program modes.

In ALG mode, begin entering the vector by pressing ↱ []. RPN mode works similarly to ALG mode, except that the EQN key must be pressed first, followed by pressing ↱ [] .

To enter an element containing the value stored in a lettered variable, press RCL and the *variable* letter.

To enter an element from a stack register, press the R↓ key and use the > or < keys to move the underline symbol so that it is under the stack register to be used and press ENTER

To enter an element indirectly indicated by the value in the I or J register, press RCL and either (I) or (J).

For example, to construct the vector [C, REGZ, (J)] in RPN mode, press EQN ↱ [], then RCL C ↰ , R↓ > ENTER ↰ , RCL (J) ENTER.

11

Base Conversions and Arithmetic and Logic

The BASE menu (⮳ BASE) allows you to enter numbers and force the display of numbers in decimal, binary, octal and hexadecimal base.

The LOGIC menu(⮳ LOGIC) provides access to logic functions.

BASE Menu

Menu label	Description
DEC	*Decimal mode.* This is the normal calculator mode
HEX	*Hexadecimal mode.* The **HEX** annunciator is displayed when this mode is active. Numbers are displayed in hexadecimal format. In RPN mode, the keys SIN, COS, TAN, $\sqrt{x}$, y^x and $1/x$ act as shortcut to enter the digits A to F. In ALG mode, press RCL A, B, C, D, E or F to enter the digits A to F.
OCT	*Octal mode.* The **OCT** annunciator is displayed when this mode is active. Numbers are displayed in Octal format.
BIN	*Binary mode.* The **BIN** annunciator is displayed when this mode is active. Numbers are displayed in Binary format. If a number has more than 12 digits, the ⮳ › and ⮳ ‹ keys allow to view the full number (See "Windows for Long Binary Numbers" later in this chapter.)
d	placed at the end of a number means that this number is a decimal number
h	placed at the end of a number means that this number is an hexadecimal number. To enter an hexadecimal number, type the number followed by “h”

o	placed at the end of a number means that this number is an octal number. To enter an octal number, type the number followed by "o"
b	placed at the end of a number means that this number is a binary number. To enter a binary number, type the number followed by "b"

Examples: Converting the Base of a Number.

The following keystrokes do various base conversions.

Convert 125.99_{10} to hexadecimal, octal, and binary numbers.

Keys:	Display:	Description:
1 2 5 ↱ BASE	7Dh	Converts the decimal number to base 16.
2 (2HEX)		
↱ BASE 3 (3OCT)	175o	Base 8.
↱ BASE 4 (4BIN)	1111101b	Base 2.
↱ BASE 1 (1DEC)	125.0000	

Note: When non decimal bases are use, only the integer part of numbers are used for display. The fractional parts are kept (unless operations are performed that erase them) and will be displayed if the decimal base is selected.

Convert $24FF_{16}$ to binary base. The binary number will be more than 14 digits (the maximum display) long.

Keys:	Display:	Description:
✓ ↱ BASE 2 (2HEX)	24FFh	Use the 1/x key to type "F".
2 4 1/x 1/x ↱		
BASE 6 (6h)		

[↱] [BASE] [4] (4BIN)	10010011111111➡	The entire binary number does not fit. The ➡ annunciator indicates that the number continues to the right.
[↱] [>]	⬅b	Displays the rest of the number. The full number is 10010011111111$_b$.
[↱] [<]	10010011111111➡	Displays the first 14 digits again.
[↱] [BASE] [1](1DEC)	9,471.0000	Restores base 10.

you can use [BASE] menu to enter base-n sign b/o/d/h following the operand to represent 2/8/10/16 base number in any base mode. A number without a base sign is a decimal number

Note:

In ALG mode:

1. The result's base mode is determined by the current base mode setting.
2. If there is no active command line (there is no blinking cursor on line 1), changing the base will update line 2 to be in the new base.
3. After pressing [ENTER] or changing the base mode, calculator will automatically add a current base sign b/o/h following the result to represent base 2/8/16 number in line 2.
4. To edit expression again, press [<] or [>]

In RPN mode:

When you enter a number in line 2, press [ENTER], and then change the base mode, the calculator will convert the base of the numbers in line 1 and line 2, and the sign b/o/h will be added following the number to represent base 2/8/16.

To view the next screen's content in line 2, press [↱][<] or [↱][>] to change the screen.

LOGIC Menu

Menu label	Description
AND	Logical bit-by-bit "AND" of two arguments. For example: AND(1100b,1010b)=1000b
XOR	Logical bit-by-bit "XOR" of two arguments. For example: XOR(1101b,1011b)=110b
OR	Logical bit-by-bit "OR" of two arguments. For example: OR(1100b,1010b)=1110b
NOT	Returns the one's complement of the argument. Each bit in the result is the complement of the corresponding bit in the argument. For example: NOT(1011b)= 111111111111111111111111111111110100b
NAND	Logical bit-by-bit "NAND" of two arguments. For example: NAND(1100b,1010b)=11111111111111111111111 1111111110111b
NOR	Logical bit-by-bit "NOR" of two arguments. For example: NOR(1100b,1010b)= 111111111111111111111111111111110001b

The “AND”, “OR”, “XOR”, “NOT”, “NAND”, “NOR” can be used as logic functions. Fraction, complex, vector arguments will be seen as an "INVALID DATA" in logic function.

Arithmetic in Bases 2, 8, and 16

You can perform arithmetic operations using +, −, ×, and ÷ in any base. The only function keys that are actually deactivated in HEX mode are √x, e^x, LN, y^x, 1/x, and Σ+. However, you should realize that most operations other than arithmetic will not produce meaningful results since the fractional parts of numbers are truncated.

Arithmetic in bases 2, 8, and 16 is in 2's complement form and uses integers only:

- If a number has a fractional part, only the integer part is used for an arithmetic calculation.

- The result of an operation is always an integer (any fractional portion is truncated).

Whereas conversions change only the display of the number but not the actual number in the X–register, *arithmetic does* alter the number in the X–register.

If the result of an operation cannot be represented in valid bits, the display shows OVERFLOW and then shows the largest positive or negative number possible.

Example:

Here are some examples of arithmetic in Hexadecimal, Octal, and Binary modes:

$$12F_{16} + E9A_{16} = ?$$

	Keys:	Display:	Description:
	↱ BASE 2 (2HEX)		Sets base 16; **HEX** annunciator on.
✓	1 2 1/x ↱ BASE 6 (6h) ENTER y^x 9 SIN ↱ BASE 6 (6h) +	FC9h	Result.

$$7760_8 - 4326_8 = ?$$

	Keys:	Display:	Description:
	↱ BASE 3 (3OCT)	7711o	Sets base 8; **OCT** annunciator on. Converts displayed number to octal.
✓	7 7 6 0 ↱ BASE 7 (7o) ENTER 4 3 2 6 ↱ BASE 7 (7o) −	3432o	Result.

$$100_8 \div 5_8 = ?$$

	Keys:	Display:	Description:
✓	1 0 0 ↱ BASE 7 (7o) ENTER 5 ↱ BASE 7 (7o) ÷	14o	Integer part of result.

$$5A0_{16} + 1001100_2 = ?$$

	Keys:	Display:	Description:
✓	↱ BASE 2 (2HEX) 5 SIN 0 ↱ BASE 6 (6h) ENTER	5A0h	Sets base 16; **HEX** annunciator on.

↱ BASE 4 (4BIN) 1 0 0 1 1 0 0 ↱ 8 (8b)	1001100b	Changes to base 2; **BIN** annunciator on. This terminates digit entry, so no ENTER is needed between the numbers.
✓ +	10111101100b	Result in binary base.
↱ BASE 2 (2HEX)	5ECh	Result in hexadecimal base.
↱ BASE 1 (1DEC)	1,516.0000	Restores decimal base.

The Representation of Numbers

Although the *display* of a number is converted when the base is changed, its stored form is not modified, so decimal numbers are not truncated — until they are used in arithmetic calculations.

When a number appears in hexadecimal, octal, or binary base, it is shown 36 bits (12 octal digits or 9 hexadecimal digits). Leading zeros are not displayed, but they are important because they indicate a positive number. For example, the binary representation of 125_{10} is displayed as:

1111101b

which is the same as these 36 digits:

000000000000000000000000000001111101b

Negative Numbers

The leftmost (most significant or "highest") bit of a number's binary representation is the sign bit; it is set (1) for negative numbers. If there are (undisplayed) leading zeros, then the sign bit is 0 (positive). A negative number is the 2's complement of its positive binary number.

Keys:	Display:	Description:
5 4 6 ↱ BASE 2 (2HEX)	222h	Enters a positive, decimal number; then converts it to hexadecimal.

+/- 5 4 6 ENTER	FFFFFFDDEh	2's complement (sign changed).
↱ BASE 4 (4BIN)	11111111111111➡	Binary version; ➡ indicates more digits. The number is negative since the highest bit is 1.
↱ >	⬅11111111111101➡	Displays the rest of the number by scrolling one screen
↱ >	⬅11011110b	Displays the rightmost window;
↱ BASE 1 (1DEC)	-546.0000	Negative decimal number.

Range of Numbers

The 36-bit binary number size determines the range of numbers that can be represented in hexadecimal (9 digits), octal (12 digits), and binary bases (36 digits), and the range of decimal numbers (11 digits) that can be converted to these other bases.

Range of Numbers for Base Conversions

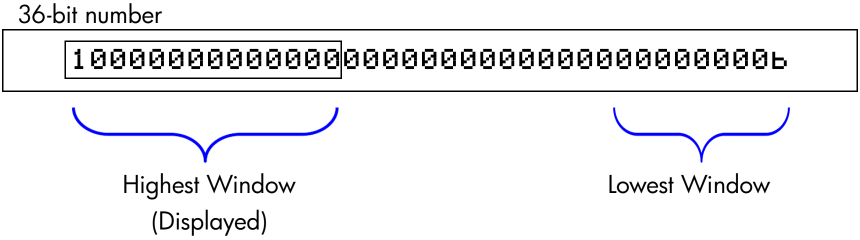
Base	Positive Integer of Largest Magnitude	Negative Integer of Largest Magnitude
Hexadecimal	7FFFFFFFFh	800000000h
Octal	377777777777o	400000000000o
Binary	0111111111111111111111 11111111111111b	1000000000000000000000 00000000000000b
Decimal	34,359,738,367	–34,359,738,368

Numbers outside of this range can not be entered when a non decimal base is selected.

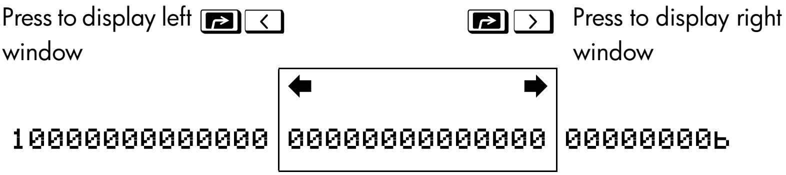
In BIN/OCT/HEX, If a number entered in decimal base is outside the range given above, then it produces the message TOO BIG. Any operation using TOO BIG causes an overflow condition, which substitutes the largest positive or negative number possible for the too-big number.

Windows for Long Binary Numbers

The longest binary number can have 36 digits. Each 14–digit display of a long number is called a *window*.



When a binary number is larger than the 14 digits, the ⬅ or ➡ annunciator (or both) appears, indicating in which direction the additional digits lie. Press the indicated key (⮫ < or ⮫ >) to view the obscured window.


Press to display left window ⮫ <
⮫ > Press to display right window
⬅
➡
10000000000000
00000000000000
00000000b

Using base in program and equations

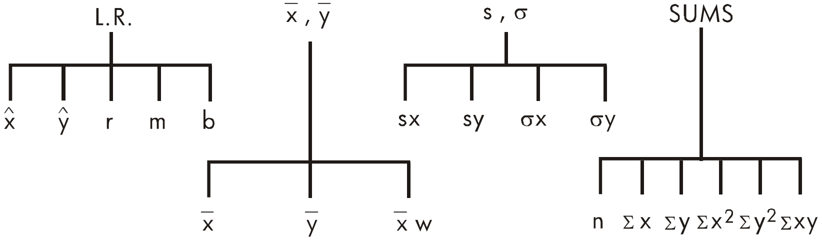
Equations and program are affected by the base setting and binary, octal and hexadecimal numbers can be entered in equation and in program as well as when the calculator prompts for a variable. Results will be displayed according to the current base.

12

Statistical Operations

The statistics menus in the HP 35s provide functions to statistically analyze a set of one– or two–variable data(real numbers):

- Mean, sample and population standard deviations.
- Linear regression and linear estimation ($\hat{x}$ and $\hat{y}$).
- Weighted mean (*x* weighted by *y*).
- Summation statistics: *n*, Σx, Σy, Σx^2, Σy^2, and Σxy.



Entering Statistical Data

One– and two–variable statistical data are entered (or deleted) in similar fashion using the [Σ+] (or [↰] [Σ–]) key. Data values are accumulated as summation statistics in six *statistics registers* (-27 through -32), whose names are displayed in the SUMS menu. (Press [↱] [SUMS] and see n Σx Σy Σx^2 Σy^2 Σxy).

Note

Always clear the statistics registers before entering a new set of statistical data (press [↱] [CLEAR] [4] (4Σ)).

Entering One–Variable Data

1. Press ↱ CLEAR 4 (4Σ)to clear existing statistical data.
2. Key in each *x*–value and press Σ+.
3. The display shows *n*, the number of statistical data values now accumulated.

Pressing Σ+ actually enters two variables into the statistics registers because the value already in the Y–register is accumulated as the *y*–value. For this reason, the calculator will perform linear regression and show you values based on *y* even when you have entered only *x*–data — or even if you have entered an unequal number of *x*–and *y*–values. No error occurs, but the results are obviously not meaningful.

To recall a value to the display *immediately after it has been entered,* press ↱ LAST*x*.

✓ Entering Two–Variable Data

If the data is a pair of variables, enter first the dependent variable (the 2nd variable of the pair) and press ENTER, and then enter the independent variable (the first variable of the pair) and press Σ+.

1. Press ↱ CLEAR 4 (4Σ) to clear existing statistical data.
2. Key in the *y*–value *first* and press ENTER.
3. Key in the corresponding *x*–value and press Σ+.
4. The display shows *n*, the number of statistical data pairs you have accumulated.
5. Continue entering *x*, *y*–pairs. *n* is updated with each entry.

To recall an *x*–value to the display immediately after it has been entered, press ↱ LAST*x*.

Correcting Errors in Data Entry

If you make a mistake when entering statistical data, delete the incorrect data and add the correct data. Even if only one value of an *x*, *y*–pair is incorrect, you must delete and reenter *both* values.

To correct statistical data:

1. Reenter the incorrect data, but instead of pressing Σ+, press ↰ Σ–. This deletes the value(s) and decrements *n*.
2. Enter the correct value(s) using Σ+.

✓ If the incorrect values were the ones just entered, press ↰ LAST*x* to retrieve them, then press ↰ Σ– to delete them. (The incorrect *y*–value was still in the Y–register, and its x–value was saved in the LAST X register.) After deleting the incorrect statistical data, calculator will display the value of Y-register in line 1 and value of n in line 2.

Example:

Key in the *x*, *y*–values on the left, then make the corrections shown on the right:

Initial x, y	Corrected x, y
20, 4	20, 5
400, 6	40, 6

Keys:	Display:	Description:
↱ CLEAR 4 (4Σ)		Clears existing statistical data.
✓ 4 ENTER 2 0 Σ+	4.0000 1.0000	Enters the first new data pair.
✓ 6 ENTER 4 0 0 Σ+	6.0000 2.0000	Display shows *n*, the number of data pairs you entered.
↱ LAST*x*	6.0000 400.0000	Brings back last *x*–value. Last *y* is still in Y–register.
↰ Σ–	6.0000 1.0000	Deletes the last data pair.
✓ 6 ENTER 4 0 Σ+	6.0000 2.0000	Reenters the last data pair.
✓ 4 ENTER 2 0 ↰ Σ–	4.0000 1.0000	Deletes the first data pair.

✓ 5 ENTER 2 0 Σ+	5.0000 2.0000	Reenters the first data pair. There is still a total of two data pairs in the statistics registers.

Statistical Calculations

Once you have entered your data, you can use the functions in the statistics menus.

Statistics Menus

Menu	Key	Description
L.R.	↰ L.R.	The linear–regression menu: linear estimation $\hat{x}$ $\hat{y}$ and curve–fitting r m b. See ''Linear Regression'' later in this chapter.
$\bar{x}$, $\bar{y}$	↰ $\bar{x},\bar{y}$	The mean menu: $\bar{x}$ $\bar{y}$ $\bar{x}$w . See "Mean" below.
s,σ	↱ S,σ	The standard–deviation menu: sx sy σx σy. See "Sample Standard Deviation" and "Population Standard Deviation" later in this chapter.
SUMS	↱ SUMS	The summation menu: n Σx Σy Σx2 Σy2 Σxy. See "Summation Statistics" later in this chapter.

Mean

Mean is the arithmetic average of a group of numbers.

- Press ↰ $\bar{x},\bar{y}$ ($\bar{x}$) for the mean of the *x*–values.
- Press ↰ $\bar{x},\bar{y}$ ▷ ($\bar{y}$) for the mean of the *y*–values.
- Press ↰ $\bar{x},\bar{y}$ ▷ ▷ ($\bar{x}$w) for the *weighted* mean of the *x*–values using the *y*–values as weights or frequencies. The weights can be integers or non–integers.

Example: Mean (One Variable).

Production supervisor May Kitt wants to determine the average time that a certain process takes. She randomly picks six people, observes each one as he or she carries out the process, and records the time required (in minutes):

15.5	9.25	10.0
12.5	12.0	8.5

Calculate the mean of the times. (Treat all data as *x*–values.)

Keys:	Display:	Description:
↱ CLEAR 4 (4Σ)		Clears the statistics registers.
1 5 · 5 Σ+	1.0000	Enters the first time.
9 · 2 5 Σ+ 1 0 Σ+ 1 2 · 5 Σ+ 1 2 Σ+ 8 · 5 Σ+	6.0000	Enters the remaining data; six data points accumulated.
↰ $\bar{x},\bar{y}$ ($\bar{x}$)	$\bar{x}$ $\bar{y}$ $\bar{x}$w 11.2917	Calculates the mean time to complete the process.

Example: Weighted Mean (Two Variables).

A manufacturing company purchases a certain part four times a year. Last year's purchases were:

Price per Part (x)	$4.25	$4.60	$4.70	$4.10
Number of Parts (y)	250	800	900	1000

Find the average price (weighted for the purchase quantity) for this part. Remember to enter *y*, the weight (frequency), before *x*, the price.

Keys:	Display:	Description:
↱ CLEAR 4 (4Σ)		Clears the statistics registers.
✓ 2 5 0 ENTER 4 · 2 5 Σ+		Enters data; displays *n*.
✓ 8 0 0 ENTER 4 · 6 Σ+		
✓ 9 0 0 ENTER 4 · 7 Σ+	900.0000 3.0000	

✓ 1 0 0 0 ENTER 4 1 Σ+	1,000.0000 4.0000	Four data pairs accumulated.
↰ x̄,ȳ > > (x̄w)	x̄ ȳ x̄w 4.4314	Calculates the mean price weighted for the quantity purchased.

Sample Standard Deviation

Sample standard deviation is a measure of how dispersed the data values are about the mean sample standard deviation assumes the data is a sampling of a larger, complete set of data, and is calculated using *n* – 1 as a divisor.

- Press ↱ S,σ (sx) for the standard deviation of *x*–values.
- Press ↱ S,σ > (sy) for the standard deviation of *y*–values.

The (σx) and (σy) items in this menu are described in the next section, "Population Standard Deviation."

Example: Sample Standard Deviation.

Using the same process–times as in the above "mean" example, May Kitt now wants to determine the standard deviation time (s_x) of the process:

15.5	9.25	10.0
12.5	12.0	8.5

Calculate the standard deviation of the times. (Treat all the data as *x*–values.)

Keys:	Display:	Description:
↱ CLEAR 4 (4Σ)		Clears the statistics registers.
1 5 · 5 Σ+	1.0000	Enters the first time.
9 · 2 5 Σ+ 1 0 Σ+ 1 2 · 5 Σ+ 1 2 Σ+ 8 · 5 Σ+	6.0000	Enters the remaining data; six data points entered.
↱ S,σ (sx)	sx sy σx σy 2.5808	Calculates the standard deviation time.

Population Standard Deviation

Population standard deviation is a measure of how dispersed the data values are about the mean. Population standard deviation assumes the data constitutes the *complete* set of data, and is calculated using n as a divisor.

- Press ↱ S,σ > > (σx) for the population standard deviation of the *x*–values.
- Press ↱ S,σ > > > (σy) for the population standard deviation of the *y*–values.

Example: Population Standard Deviation.

Grandma Hinkle has four grown sons with heights of 170, 173, 174, and 180 cm. Find the population standard deviation of their heights.

Keys:	Display:	Description:
↱ CLEAR 4 (4Σ)		Clears the statistics registers.
1 7 0 Σ+ 1 7 3		Enters data. Four data points
Σ+ 1 7 4 Σ+ 1 8		accumulated.
0 Σ+	4.0000	
↱ S,σ > > (σx)	sx sy σx	Calculates the population
	σy	standard deviation.
	3.6315	

Linear Regression

Linear regression, L.R. (also called *linear estimation)* is a statistical method for finding a straight line that best fits a set of *x*,*y*–data.

Note To avoid a STAT ERROR message, enter your data *before* executing any of the functions in the L.R. menu.

L.R. (Linear Regression) Menu

Menu Key	Description
x̂	Estimates (predicts) *x* for a given hypothetical value of *y*, based on the line calculated to fit the data.
ŷ	Estimates (predicts) *y* for a given hypothetical value of *x*, based on the line calculated to fit the data.
r	Correlation coefficient for the (*x*, *y*) data. The correlation coefficient is a number in the range –1 through +1 that measures how closely the calculated line fits the data.
m	Slope of the calculated line.
b	*y*–intercept of the calculated line.

- ✓ To find an estimated value for *x* (or *y*), key in a given hypothetical value for *y* (or *x*), then press ⬑ L.R. (x̂) (or ⬑ L.R. > (ŷ).
- To find the values that define the line that best fits your data, press ⬑ L.R. followed by r, m, or b.

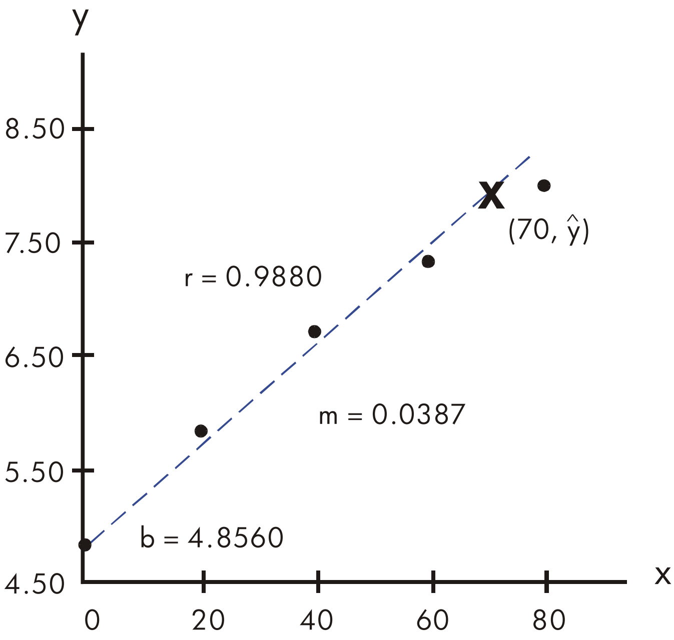
Example: Curve Fitting.

The yield of a new variety of rice depends on its rate of fertilization with nitrogen. For the following data, determine the linear relationship: the correlation coefficient, the slope, and the *y*–intercept.

X, Nitrogen Applied (kg per hectare)	0.00	20.00	40.00	60.00	80.00
Y, Grain Yield (metric tons per hectare)	4.63	5.78	6.61	7.21	7.78

Keys:	Display:	Description:
⬏ CLEAR 4 (4Σ)		Clears all previous statistical data.

✓ 4 · 6 3 ENTER 0 Σ+		Enters data; displays *n*.
✓ 5 · 7 8 ENTER 2 0 Σ+	7.2100 4.0000	
✓ 6 · 6 1 ENTER 4 0 Σ+		
✓ 7 · 2 1 ENTER 6 0 Σ+		
✓ 7 · 7 8 ENTER 8 0 Σ+	7.7800 5.0000	Five data pairs entered.
↰ L.R. › › (r)	x̂ ŷ r m b 0.9880	Displays linear–regression menu. Correlation coefficient; data closely approximate a straight line.
›	x̂ ŷ r m b 0.0387	Slope of the line.
›	x̂ ŷ r m b 4.8560	*y*–intercept.



What if 70 kg of nitrogen fertilizer were applied to the rice field? Predict the grain yield based on the above statistics.

Keys:	Display:	Description:
✓ C 7 0	7.7800 70_	Enters hypothetical *x*–value.
↰ L.R. > ($\hat{y}$)	$\hat{x}$ $\underline{\hat{y}}$ r m b 7.5615	The predicted yield in tons per hectare.

Limitations on Precision of Data

Since the calculator uses finite precision, it follows that there are limitations to calculations due to rounding. Here are two examples:

Normalizing Close, Large Numbers

The calculator might be unable to correctly calculate the standard deviation and linear regression for a variable whose data values differ by a relatively small amount. To avoid this, normalize the data by entering each value as the difference from one central value (such as the mean). For normalized *x*–values, this difference must then be added back to the calculation of $\bar{x}$ and $\hat{x}$, and $\hat{y}$ and *b* must also be adjusted. For example, if your *x*–values were 7776999, 7777000, and 7777001, you should enter the data as –1, 0, and 1; then add 7777000 back to $\bar{x}$ and $\hat{x}$. For b, add back 7777000 × *m.* To calculate $\hat{y}$, be sure to supply an *x*–value that is less 7777000.

Similar inaccuracies can result if your *x* and *y* values have greatly different magnitudes. Again, scaling the data can avoid this problem.

Effect of Deleted Data

Executing ↰ Σ- does not delete any rounding errors that might have been generated in the statistics registers by the original data values. This difference is not serious unless the incorrect data have a magnitude that is enormous compared with the correct data; in such a case, it would be wise to clear and reenter all the data.

Summation Values and the Statistics Registers

The statistics registers are six unique locations in memory that store the accumulation of the six summation values.

Summation Statistics

Pressing [↱] [SUMS] gives you access to the contents of the statistics registers:

- (n) to recall the number of accumulated data sets.
- Press [>] (Σx) to recall the sum of the *x*–values.
- Press [>] [>] (Σy) to recall the sum of the *y*–values.
- Press [>] [>] [>] (Σx²), [>] [>] [>] [>] (Σy²), and [>] [>] [>] [>] [>] (Σxy) to recall the sums of the squares and the sum of the products of the *x* and *y* — values that are of interest when performing other statistical calculations in addition to those provided by the calculator.

If you've entered statistical data, you can see the contents of the statistics registers. Press [↰] [MEM] [1](1VAR) [ENTER], then use [⌃] and [⌄] to view the statistics registers.

Example: Viewing the Statistics Registers.

Use [Σ+] to store data pairs (1,2) and (3,4) in the statistics registers. Then view the stored statistical values.

	Keys:	Display:	Description:
	[↱] [CLEAR] [4] (4Σ)		Clears the statistics registers.
✓	[2] [ENTER] [1] [Σ+]	2.0000 1.0000	Stores the first data pair (1,2).
✓	[4] [ENTER] [3] [Σ+]	4.0000 2.0000	Stores the second data pair (3,4).
	[↰]	n=	⬆ Displays VAR catalog and views n
	[MEM] [1](1VAR)	2.0000	⬇ register.
	[⌃]	Σxy=	⬆ views Σxy register.
		14.0000	⬇

^	Σy^2= 20.0000	⬆ Views Σy^2 register. ⬇
^	Σx^2= 10.0000	⬆ Views Σx^2 register. ⬇
^	Σy= 6.0000	⬆ Views Σy register. ⬇
^	Σx= 4.0000	⬆ Views Σx register. ⬇
^	n= 2.0000	⬆ Views *n* register. ⬇
C	4.0000 2.0000	Leaves VAR catalog.

Access to the Statistics Registers

The statistics register assignments in the HP 35s are shown in the following table. Summation registers should be referred to by names and not by numbers in expression, equations and programs.

Statistics Registers

Register	Number	Description
n	-27	Number of accumulated data pairs.
Σx	-28	Sum of accumulated *x*–values.
Σy	-29	Sum of accumulated *y*–values.
Σx^2	-30	Sum of squares of accumulated *x*–values.
Σy^2	-31	Sum of squares of accumulated *y*–values.
Σxy	-32	Sum of products of accumulated *x*– and *y*–values.

You can load a statistics register with a summation by storing the number (-27 through -32) of the register you want in *I or J* and then storing the summation (*value* STO (I) *or* (J)). Similarly, you can press ↰ VIEW (I) *or* (J) (or RCL (I) *or* (J)) to view (or recall)a register value — the display is labeled with the register name. The SUMS menu contains functions for recalling the register values. See "Indirectly Addressing Variables and Labels" in chapter 14 for more information.

Part 2

Programming

13

Simple Programming

Part 1 of this manual introduced you to functions and operations that you can use *manually*, that is, by pressing a key for each individual operation. And you saw how you can use equations to repeat calculations without doing all of the keystrokes each time.

In part 2, you'll learn how you can use *programs* for repetitive calculations — calculations that may involve more input or output control or more intricate logic. A program lets you repeat operations and calculations in the precise manner you want.

In this chapter you will learn how to program a series of operations. In the next chapter, "Programming Techniques," you will learn about subroutines and conditional instructions.

Example: A Simple Program.

To find the area of a circle with a radius of 5, you would use the

formula $A = \pi r^2$ and press

RPN mode: 5 x^2 ↰ π ×

ALG mode: 5 y^x 2 × ↰ π ENTER

to get the result for this circle, 78.5398.

But what if you wanted to find the *area* of many different circles?

Rather than repeat the given keystrokes each time (varying only the "5" for the different radii), you can put the repeatable keystrokes into a program:

RPN mode	ALG mode
0001 x²	0001 SQ(x)xπ
0002 π	
0003 x	

This very simple program assumes that the value for the radius is in the X– register (the display) when the program starts to run. It computes the area and leaves it in the X–register.

In RPN mode, to enter this program into program memory, do the following:

Keys: **(In RPN mode)**	**Display:**	**Description:**
↱ CLEAR 3		Clears memory.
(3ALL) < (Y) ENTER		
↱ PRGM		Activates Program–entry mode (**PRGM** annunciator on).
GTO · ·	PRGM TOP	Resets program pointer to PRGM TOP.
↱ x^2	0001 x²	(Radius)2
↰ π	0002 π	
×	0003 x	*Area* = *πx*2
↱ PRGM		Exits Program–entry mode.

Try running this program to find the area of a circle with a radius of 5:

Keys: **(In RPN mode)**	**Display:**	**Description:**
GTO · ·		This sets the program to its beginning.
5 R/S	78.5398	The answer!

In ALG mode, to enter this program into program memory, do the following:

Keys: **(In ALG mode)**	**Display:**	**Description:**
↱ CLEAR 3		Clears memory.
(3ALL) < (Y) ENTER		
↱ PRGM		Activates Program–entry mode (**PRGM** annunciator on).

GTO · ·	PRGM TOP	Resets program pointer to PRGM TOP.
↱ x² RCL X > × ↰ π	0001 SQ(X)×π	*Area* = π*x*2
↱ PRGM		Exits Program–entry mode.

Try running this program to find the area of a circle with a radius of 5:

Keys: (In ALG mode)	**Display:**	**Description:**
GTO · ·		This sets the program to its beginning.
5 STO X ENTER	5▸X 5.0000	Stores 5 into X
R/S	78.9358	The answer!

We will continue using the above program for the area of a circle to illustrate programming concepts and methods.

Designing a Program

The following topics show what instructions you can put in a program. What you put in a program affects how it appears when you view it and how it works when you run it.

Selecting a Mode

Programs created and saved in RPN mode should be edited and executed in RPN mode, and programs or steps created and saved in ALG mode should be edited and executed in ALG mode. If not, the result may be incorrect.

Program Boundaries (LBL and RTN)

If you want more than one program stored in program memory, then a program needs a *label* to mark its beginning (such as A001 LBL A) and a *return* to mark its end (such as A005 RTN).
Notice that the line numbers acquire an A to match their label.

Program Labels

Programs and segments of programs (called *routines*) should start with a label. To record a label, press:

↱ LBL *letter–key*

The label is a single letter from A through Z. The letter keys are used as they are for variables (as discussed in chapter 3). You cannot assign the same label more than once (this causes the message DUPLICAT.LBL), but a label can use the same letter that a variable uses.

It is possible to have one program (the top one) in memory without any label. However, adjacent programs need a label between them to keep them distinct.

Programs can not have more than 999 lines.

Program Returns

Programs and subroutines should end with a return instruction. The keystrokes are:

↰ RTN

When a program finishes running, the last RTN instruction returns the program pointer to PRGM TOP, the top of program memory.

Using RPN, ALG and Equations in Programs

You can calculate in programs the same ways you calculate on the keyboard:

- Using RPN operations (which work with the stack, as explained in chapter 2).
- Using ALG operations (as explained in appendix C).
- Using equations (as explained in chapter 6).

The previous example used a series of *RPN operations* to calculate the area of the circle. Instead, you could have used an *equation* in the program. (An example follows later in this chapter.) Many programs are a combination of RPN *and* equations, using the strengths of both.

Strengths of RPN Operations	**Strengths of Equations and ALG Operations**
Use less memory.	Easier to write and read.
Execute faster.	*Can* automatically prompt.

When a program executes a line containing an equation, the equation is evaluated in the same way that XEQ evaluates an equation in the equation list. For program evaluation, "=" in an equation is essentially treated as "–". (There's no programmable equivalent to ENTER for an assignment equation — other than writing the equation as an expression, then using STO to store the value in a variable.)

For both types of calculations, you can include RPN instructions to control input, output, and program flow.

Data Input and Output

For programs that need more than one input or return more than one output, you can decide how you want the program to enter and return information.

For input, you can prompt for a variable with the INPUT instruction, you can get an equation to prompt for its variables, or you can take values entered in advance onto the stack.

For output, you can display a variable with the VIEW instruction, you can display a message derived from an equation, you can display process in line 1, you can display the program result in line 2, or you can leave unmarked values on the stack.

These are covered later in this chapter under "Entering and Displaying Data."

Entering a Program

Pressing ↱ PRGM toggles the calculator into and out of Program–entry mode — turns the **PRGM** annunciator on and off. Keystrokes in Program–entry mode are stored as program lines in memory. Each instruction (command) or expression occupies one program line. In ALG mode, you can enter an expression directly in a program

To enter a program into memory:

1. Press ↱ PRGM to activate Program–entry mode.
2. Press GTO · · to display PRGM TOP. This sets the *program pointer* to a known spot, before any other programs. As you enter program lines, they are inserted *before* all other program lines.

   If you don't need any other programs that might be in memory, clear program memory by pressing ↱ CLEAR 3 (3PGM). To confirm that you want *all* programs deleted, press < (Y) ENTER after the message CLR PGMS? Y_N.
3. Give the program a *label* — a single letter, A through Z. Press ↱ LBL *letter.* Choose a letter that will remind you of the program, such as "A" for "area."

   If the message DUPLICAT.LBL is displayed, use a different letter. You can clear the existing program instead — press ↰ MEM 2(2PGM), use ^ or v to find the label, and press ↱ CLEAR and C.
4. To record calculator operations as program instructions, press the same keys you would to do an operation manually. Remember that many functions don't appear on the keyboard but must be accessed using menus.
   To enter an equation in a program line, see the instructions below.

5. End the program with a *return* instruction, which sets the program pointer back to PRGM TOP after the program runs. Press ↰ RTN.
6. Press C (or ↱ PRGM) to cancel program entry.

Numbers in program lines are stored precisely as you entered them, and they're displayed using ALL or SCI format. (If a long number is shortened in the display, press ↰ SHOW to view all digits.)

To enter an equation in a program line:

1. Press EQN to activate Equation–entry mode. The **EQN** annunciator turns on.
2. Enter the equation as you would in the equation list. See chapter 6 for details. Use ← to correct errors as you type.
3. Press ENTER to terminate the equation and display its left end. (The equation does *not* become part of the equation list.)

After you've entered an equation, you can press ↰ SHOW to see its checksum and length. Hold the SHOW key to keep the values in the display.

For a long equation, the ➡ and ⬅ annunciators show that scrolling is active for this program line. You can use ↱ < and ↱ > to scroll the display.

Clear functions and backspace key

Note these special conditions during program entry:

- C always cancels program entry. It never clears a number to zero.
- In program line view status, ← deletes the current program line and < / > begins the edit status. In program line edit status, ← deletes a character before the cursor.
- To *program* a function to clear the X–register, use ↱ CLEAR 1 (1x).

When you insert or erase a line in a program, GTO and XEQ statements are automatically updated if needed.

For example:

```
A001 LBL A
A002 2+3
A003 1+2
A004
GTO A003
```

Now, erase line A002, and line A004 changes to “A003 GTO A002”

Function Names in Programs

The name of a function that is used in a program line is *not* necessarily the same as the function's name on its key, in its menu, or in an equation. The name that is used in a program is usually a fuller abbreviation than that which can fit on a key or in a menu.

Example: Entering a Labeled Program.

The following keystrokes delete the previous program for the area of a circle and enter a new one that includes a label and a return instruction. If you make a mistake during entry, press ← to delete the current program line, then reenter the line correctly.

Keys: (In RPN mode)	Display:	Description:
↱ PRGM		Activates Program–entry mode (**PRGM** on).
↱ CLEAR 3 (3PGM) ◁ (Y) ENTER	PRGM TOP	Clears all of program memory.
↱ LBL A	A001 LBL A	Labels this program routine A (for "area").
↱ x^2	A002 x^2	Enters the three program lines.
↰ π	A003 π	
×	A004 x	
↰ RTN	A005 RTN	Ends the program.
↰ MEM 2 (2PGM)	LBL A LN=15	Displays label A and the length of the program in bytes.

↰ SHOW	CK=DAF1
	LN=15
		Checksum and length of program.

C C		Cancels program entry (**PRGM** annunciator off).

A different checksum means the program was not entered exactly as given here.

Example: Entering a Program with an Equation.

The following program calculates the area of a circle using an equation, rather than using RPN operations like the previous program.

Keys: (In RPN mode)	Display:	Description:
↱ PRGM GTO · ·	`PRGM TOP`	Activates Program–entry mode; sets pointer to top of memory.
↱ LBL E	`E001 LBL E`	Labels this program routine E (for "equation").
↱ STO R	`E002 STO R`	Stores radius in variable R
EQN ↰ π × RCL R y^x 2 ENTER	`E003 π×R^2`	Selects Equation–entry mode; enters the equation; returns to Program–entry mode.

↰ SHOW	`CK=7E5B`
	`LN=5`	

↰ RTN	`E004 RTN`	Ends the program.

↰ MEM 2 (2`PGM`)	`LBL E`
	`LN=17`
		Displays label E and the length of the program in bytes.

↰ SHOW	`CK=2073`
	`LN=17`
		Checksum and length of program.

C C		Cancels program entry.

Running a Program

To run or *execute* a program, program entry cannot be active (no program–line numbers displayed; **PRGM** off). Pressing C will cancel Program–entry mode.

Executing a Program (XEQ)

Press XEQ *label* to execute the program labeled with that letter:

To execute a program from it's beginning press XEQ label ENTER. For example, press XEQ A ENTER. The display will show "XEQ A001" and execution will start at the top of Label A.

You can also execute a program starting at another position by pressing XEQ label Line number, for example XEQ A 0 0 5.

If there is only one program in memory, you can also execute it after moving pointer to the top of the program line and pressing R/S (run / stop) key. The **PRGM** annunciator displays and the B annunciator turns on while the program is running.

If necessary, enter the data before executing the program.

Example:

Run the programs labeled A and E to find the areas of three different circles with radii of 5, 2.5, and 2π. Remember to enter the radius before executing A or E.

Keys: (In RPN mode)	Display:	Description:
5 XEQ A ENTER	RUNNING 78.5398	Enters the radius, then starts program A. The resulting area is displayed.
2 · 5 XEQ E ENTER	19.6350	Calculates area of the second circle using program E.
2 ↰ π ×		Calculates area of the third circle.
XEQ A ENTER	124.0251	

Testing a Program

If you know there is an error in a program, but are not sure where the error is, then a good way to test the program is by stepwise execution. It is also a good idea to test a long or complicated program before relying on it. By stepping through its execution, one line at a time, you can see the result after each program line is executed, so you can verify the progress of known data whose correct results are also known.

1. As for regular execution, make sure program entry is not active (**PRGM** annunciator off).
2. Set the program pointer to the start of the program (that is, at its LBL instruction). The instruction moves the program pointer without starting execution.
3. Press and hold ⌄. This displays the current program line. When you release ⌄, the line is executed. The result of that execution is then displayed (it is in the X–register).

   To move to the *preceding* line, you can press ⌃. No execution occurs.
4. The program pointer moves to the next line. Repeat step 3 until you find an error (an incorrect result occurs) or reach the end of the program.

If Program–entry mode is active, then ⌄ or ⌃ simply changes the program pointer, without executing lines. Holding down a cursor key during program entry makes the lines roll by automatically.

Example: Testing a Program.

Step through the execution of the program labeled A. Use a radius of 5 for the test data. Check that Program–entry mode is *not* active before you start:

Keys: **(In RPN mode)**	**Display:**	**Description:**
5 GTO A	5.0000	Moves program counter to label A.
ENTER		
⌄ (hold) (release)	A001 LBL A 5.0000	
⌄ (hold) (release)	A002 x^2 25.0000	Squares input.

⌄ (hold) (release)	A003 π 3.1416	Value of *π*.
⌄ (hold) (release)	A004 × 78.5398	25*π*.
⌄ (hold) (release)	A005 RTN 78.5398	End of program. Result is correct.

Entering and Displaying Data

The calculator's *variables* are used to store data input, intermediate results, and final results. (Variables, as explained in chapter 3, are identified by a letter from *A* through *Z*, but the variable names have nothing to do with program labels.)

In a program, you can get data in these ways:

- From an INPUT instruction, which prompts for the value of a variable. (This is the most handy technique.)
- From the stack. (You can use STO to store the value in a variable for later use.)
- From variables that already have values stored.
- From automatic equation prompting (if enabled by flag 11 set). (This is also handy if you're using equations.)

In a program, you can display information in these ways:

- With a VIEW instruction, which shows the name and value of a variable. (This is the most handy technique.)
- On the stack - only the values in the X and Y registers are visible. (You can use PSE for a 1-second look at the X and Y registers.)
- In a displayed equation (if enabled by flag 10 set). (The "equation" is usually a message, not a true equation.)

Some of these input and output techniques are described in the following topics.

Using INPUT for Entering Data

The INPUT instruction (↰ INPUT *Variable*) stops a running program and displays a prompt for the given variable. This display includes the existing value for the variable, such as

```
R?
0.0000
```

where

"R" is the variable's name,
"?" is the prompt for information, and
0.0000 is the current value stored in the variable.

Press R/S *(run/stop)* to resume the program. The value you keyed in then writes over the contents of the X–register *and* is stored in the given variable. If you have not changed the displayed value, then that value is retained in the X–register.

The area–of–a–circle program with an INPUT instruction looks like this:

RPN mode	ALG mode
A001 LBL A	A001 LBL A
A002 INPUT R	A002 INPUT R
A003 x^2	A003 SQ(R)xπ
A004 π	A004 RTN
A005 x	
A006 RTN	

To use the INPUT function in a program:

1. Decide which data values you will need, and assign them names.
   (In the area–of–a–circle example, the only input needed is the radius, which we can assign to *R*.)

2. In the beginning of the program, insert an INPUT instruction for each variable whose value you will need. Later in the program, when you write the part of the calculation that needs a given value, insert a RCL *variable* instruction to bring that value back into the stack.

Since the INPUT instruction also leaves the value you just entered in the Xñregister, you don't *have* to recall the variable at a later time ó you could INPUT it and use it when you need it. You might be able to save some memory space this way. However, in a long program it is simpler to just input all your data up front, and then recall individual variables as you need them.

Remember also that the user of the program can do calculations while the program is stopped, waiting for input. This can alter the contents of the stack, which might affect the next calculation to be done by the program. Thus the program should not assume that the X–, Y–, and Zñregisters' contents will be the same before and after the INPUT instruction. If you collect all the data in the beginning and then recall them when needed for calculation, then this prevents the stack's contents from being altered just before a calculation.

To respond to a prompt:

When you run the program, it will stop at each INPUT and prompt you for that variable, such as `R?0.0000`. The value displayed (and the contents of the X–register) will be the current contents of R.

- **To leave the number unchanged,** just press R/S.
- **To change the number,** type the new number and press R/S. This new number writes over the old value in the X–register. You can enter a number as a fraction if you want. If you need to calculate a number, use normal keyboard calculations, then press R/S. For example, you can press 2 ENTER 5 y^x R/S in RPN mode, or press 2 y^x 5 ENTER R/S in ALG mode (Before you press ENTER, the expression will be displayed in line 2. After you press ENTER, the result of expression will replace the expression to display in line 2 and be saved in X-register).

- **To cancel the INPUT prompt,** press C. The current value for the variable remains in the X–register. If you press R/S to resume the program, the canceled INPUT prompt is repeated. If you press C during digit entry, it clears the number to zero. Press C again to cancel the INPUT prompt.

Using VIEW for Displaying Data

The programmed VIEW instruction (↰ VIEW *variable*) stops a running program and displays and identifies the contents of the given variable, such as

```
A=
78.5398
```

This is a *display only*, and does not copy the number to the X–register. If Fraction–display mode is active, the value is displayed as a fraction.

- Pressing ENTER copies this number to the X–register.
- If the number is wider than 14 characters, such as binary, complex, vector numbers, pressing ↱ < and ↱ > displays the rest.
- Pressing C (or ←) erases the VIEW display and shows the X–register.
- Pressing ↱ CLEAR clears the contents of the displayed variable.

Press R/S to continue the program.

If you don't want the program to stop, see "Displaying Information without Stopping" below.

For example, see the program for "Normal and Inverse–Normal Distributions" in chapter 16. Lines T015 and T016 at the end of the T routine display the result for X. Note also that this VIEW instruction in this program is preceded by a RCL instruction. The RCL instruction is not necessary, but it is convenient because it brings the VIEWed variable to the X–register, making it available for manual calculations. (Pressing ENTER while viewing a VIEW display would have the same effect.) The other application programs in chapters 16 and 17 also ensure that the VIEWed variable is in the X–register as well.

Using Equations to Display Messages

Equations aren't checked for valid syntax until they're evaluated. This means you can enter almost *any* sequence of characters into a program as an equation — you enter it just as you enter *any* equation. On any program line, press EQN to start the equation. Press number and math keys to get numbers and symbols. Press RCL before each letter. Press ENTER to end the equation.

If flag 10 is set, equations are *displayed* instead of being *evaluated*. This means you can display any message you enter as an equation. (Flags are discussed in detail in chapter 14.)

When the message is displayed, the program stops — press R/S to resume execution. If the displayed message is longer than 14 characters, the ➡ annunciator turns on when the message is displayed. You can then use ↱ > and ↱ < to scroll the display.

If you don't want the program to stop, see "Displaying Information without Stopping" below.

Example: INPUT, VIEW, and Messages in a Program.

Write an equation to find the surface area and volume of a cylinder given its radius and height. Label the program *C* (for *cylinder),* and use the variables *S* (surface area), *V* (volume), *R* (radius), and *H* (height). Use these formulas:

$$V = \pi R^2 H$$

$$S = 2\pi R^2 + 2\pi RH = 2\pi R (R + H)$$

Keys: (In RPN mode)	Display:	Description:
↱ PRGM ↱		Program, entry; clears the program memory.
CLEAR 3 (3PGM)	PRGM TOP	
< (Y) ENTER		
↱ LBL C	C001 LBL C	Labels program.
↰ INPUT R	C002 INPUT R	
↰ INPUT H	C003 INPUT H	Instructions to prompt for radius and height.

Keys: (In RPN mode)	Display:	Description:
EQN ↰ π ×		Calculates the volume.
RCL R y^x 2 ×		
RCL H ENTER		
	C004 π×R^2×H	
↰ SHOW	CK=74FE	Checksum and length of
	LN=7	equation.
STO V	C005 STO V	Store the volume in *V*.
EQN 2 × ↰		Calculates the surface area.
π × RCL R ×		
( ) RCL R +		
RCL H ENTER	C006 2×π×R×(R+➡	
↰ SHOW	CK=19B3	Checksum and length of
	LN=11	equation.
STO S	C007 STO S	Stores the surface area in S.
↰ FLAGS 1		Sets flag 10 to display
(1SF) · 0	C008 SF 10	equations.
EQN RCL V		Displays message in
RCL O RCL L		equations.
↱ SPACE + ↱		
SPACE RCL A		
RCL R RCL E		
RCL A ENTER	C009 VOL + ARE➡	
↰ FLAGS 1		Clears flag 10.
(2CF) · 0	C010 CF 10	
↰ VIEW V	C011 VIEW V	Displays volume.
↰ VIEW S	C012 VIEW S	Displays surface area.
↰ RTN	C013 RTN	Ends program.
↰ MEM 2	LBL C	Displays label C and the
(2PGM)	LN=67	length of the program in
		bytes.
↰ SHOW	CK=97C3	Checksum and length of
	LN=67	program.
C C		Cancels program entry.

Now find the volume and surface area–of a cylinder with a radius of 2 $^{1}/_{2}$ cm and a height of 8 cm.

Keys: **(In RPN mode)**	**Display:**	**Description:**
XEQ C ENTER	`R?` *value*	Starts executing *C*; prompts for *R*. (It displays whatever value happens to be in *R*.)
2 . 1 . 2	`H?`	Enters 2 $^{1}/_{2}$ as a fraction.
R/S	*value*	Prompts for *H*.
8 R/S	`VOL + AREA`	Message displayed.

R/S	`V=`
	`157.0796`
		Volume in cm^{3}.

R/S	`S=`
	`164.9336`
		Surface area in cm^{2}.

Displaying Information without Stopping

Normally, a program stops when it displays a variable with VIEW or displays an equation message. You normally have to press R/S to resume execution.

If you want, you can make the program continue while the information is displayed. If the *next* program line — after a VIEW instruction or a viewed equation — contains a PSE *(pause)* instruction, the information is displayed *and* execution continues after a 1–second pause. In this case, no scrolling or keyboard input is allowed.

The display is cleared by other display operations, and by the RND operation if flag 7 is set (rounding to a fraction).

Press ↱ PSE to enter PSE in a program.

The VIEW and PSE lines — or the equation and PSE lines — are treated as one operation when you execute a program one line at a time.

Stopping or Interrupting a Program

Programming a Stop or Pause (STOP, PSE)

- Pressing R/S (*run/stop*) during program entry inserts a STOP instruction. This will display the contents of the X-register and halt a running program until you resume it by pressing R/S from the keyboard. You can use STOP rather than RTN in order to end a program without returning the program pointer to the top of memory.
- Pressing ↱ PSE during program entry inserts a PSE *(pause)* instruction. This will suspend a running program and display the contents of the X–register for about 1 second — with the following exception. If PSE immediately follows a VIEW instruction or an equation that's displayed (flag 10 set), the variable or equation is displayed instead — and the display remains after the 1–second pause.

Interrupting a Running Program

You can interrupt a running program at any time by pressing C or R/S. The program completes its current instruction before stopping. Press R/S *(run/stop)* to resume the program.

If you interrupt a program and then press XEQ, GTO, or ↰ RTN, you *cannot* resume the program with R/S. Re-execute the program instead (XEQ *label line number*).

Error Stops

If an error occurs in the course of a running program, program execution halts and an error message appears in the display. (There is a list of messages and conditions in appendix F.)

To see the line in the program containing the error–causing instruction, press ↱ PRGM. The program will have stopped at that point. (For instance, it might be a ÷ instruction, which caused an illegal division by zero.)

Editing a Program

You can modify a program in program memory by inserting, deleting, and editing program lines. If a program line contains an equation, you can edit the equation.

To delete a program line:

1. Select the relevant program or routine and press ⌄ or ⌃ to locate the program line that must be changed. Hold the cursor key down to continue scrolling.
2. Delete the line you want to change —press ← directly (Undo function is active). The pointer then moves to the *preceding* line. (If you are deleting more than one consecutive program line, start with the *last* line in the group.)
3. Key in the new instruction, if any. This replaces the one you deleted.
4. Exit program entry (C or ↱ PRGM).

To insert a program line:

1. Locate and display the program line that is *before* the spot where you would like to insert a line.
2. Key in the new instruction; it is inserted *after* the currently displayed line.

For example, if you wanted to insert a new line between lines A004 and A005 of a program, you would first display line A004, then key in the instruction or instructions. Subsequent program lines, starting with the original line A005, are moved down and renumbered accordingly.

To edit operand, expression or equation in a program line:

1. Locate or display the program line that you want to edit.
2. Press > or < to start editing the program line. These turn on the "_" editing cursor, but do not delete anything in the program line

   > key actives the cursor to the left of the program line

   < key actives the cursor to the end of the program line

3. Moving the cursor"_" and press ← repeatedly to delete the unwanted number or function, then retype the rest of the program line. (After pressing ←, Undo function is active)

Notice:

1. When the cursor is active in the program line, ∨ or ∧ key will be disabled.

2. When you are editing a program line (cursor active), and the program line is empty, using ← will have no effect. If you want to erase the program line, press ENTER and the program line will be erased.

3. You can use ↱ > and ↱ < key to review long program lines and without editing it.

4. In ALG mode, ENTER can not be used as a function, it is used to validate a program line.

5. **An e**quation can be editing in any mode no matter which mode it was entered in.

Program Memory

Viewing Program Memory

Pressing ↱ PRGM toggles the calculator into and out of program entry (**PRGM** annunciator on, program lines displayed). When Program–entry mode is active, the contents of program memory are displayed.

Program memory starts at PRGM TOP. The list of program lines is circular, so you can wrap the program pointer from the bottom to the top and reverse. While program entry is active, there are four ways to change the program pointer (the displayed line):

- ↱ ∧ and ↱ ∨ allows you to move from label to label. If no labels are defined, It will move to the top or bottom of the program.
- To move more than one line at a time ("scrolling"), continue to hold the ∨ or ∧ key.

- Press GTO · · to move the program pointer to PRGM TOP.
- Press GTO · *label nnn* to move to a specific line.

If Program–entry mode is not active (if no program lines are displayed), you can also move the program pointer by pressing GTO *label line number.*

Canceling Program–entry mode does *not* change the position of the program pointer.

Memory Usage

If during program entry you encounter the message MEMORY FULL, then there is not enough room in program memory for the line you just tried to enter. You can make more room available by clearing programs or other data. See "Clearing One or More Programs" below, or "Managing Calculator Memory" in appendix B.

The Catalog of Programs (MEM)

The catalog of programs is a list of all program labels with the number of bytes of memory used by each label and the lines associated with it. Press ↰ MEM 2(2PGM) to display the catalog, and press ⌄ or ⌃ to move within the list. You can use this catalog to:

- Review the labels in program memory and the memory cost of each labeled program or routine.
- Execute a labeled program. (Press XEQ or R/S while the label is displayed.)
- Display a labeled program. (Press ↱ PRGM while the label is displayed.)
- Delete specific programs. (Press ↱ CLEAR while the label is displayed.)
- See the checksum associated with a given program segment. (Press ↰ SHOW.)

The catalog shows you how many bytes of memory each labeled program segment uses. The programs are identified by program label:

```
LBL C
LN=67
```

where 67 is the number of bytes used by the program.

Clearing One or More Programs

To clear a specific program from memory

1. Press ↰ MEM 2 (2PGM) ENTER and display (using ∨ and ∧) the label of the program.
2. Press ↱ CLEAR.
3. Press C to cancel the catalog or ← to back out.

To clear all programs from memory:

1. Press ↱ PRGM to display program lines (**PRGM** annunciator on).
2. Press ↱ CLEAR 3 (3PGM) to clear program memory.
3. The message CLR PGMS? Y N prompts you for confirmation. Press < (Y) ENTER.
4. Press ↱ PRGM to cancel program entry.

Clearing all of memory (↱ CLEAR 3 (3ALL)) also clears all programs.

The Checksum

The *checksum* is a unique hexadecimal value given to each program label and its associated lines (until the next label). This number is useful for comparison with a known checksum for an existing program that you have keyed into program memory. If the known checksum and the one shown by your calculator are the same, then you have correctly entered all the lines of the program. To see your checksum:

1. Press ↰ MEM 2 (2PGM) ENTER for the catalog of program labels.
2. Display the appropriate label by using the cursor keys, if necessary.
3. Press and hold ↰ SHOW to display CK=*checksum and* LN=*length*.

For example, to see the checksum for the current program (the "cylinder" program):

Keys: (In RPN mode)	Display:	Description:
↰ MEM 2	LBL C	Displays label C, which takes
(2PGM) ENTER	LN=67	67 bytes.

↰ SHOW (hold)	CK=97C3
	LN=67
		Checksum and length.

If your checksum does *not* match this number, then you have not entered this program correctly.

You will see that all of the application programs provided in chapters 16 and 17 include checksum values with each labeled routine so that you can verify the accuracy of your program entry.

In addition, each equation in a program has a checksum. See "To enter an equation in a program line" earlier in this chapter.

Nonprogrammable Functions

The following functions of the HP 35s are *not* programmable:

↱ CLEAR 3 (3PGM)
↱ CLEAR 3 (3ALL)
←
⌄, ⌃, <, >
↱ PRGM
↱ ⌄, ↱ ⌃
↰ UNDO

GTO . .
GTO . *label line number*
↰ MEM
↰ SHOW
EQN
↱ FDISP
↱ CLEAR 6 (6CLVARx)

Programming with BASE

You can program instructions to change the base mode using ↱ BASE. These settings work in programs just as they do as functions executed from the keyboard.

This allows you to write programs that accept numbers in any of the four bases, do arithmetic in any base, and display results in any base.

When writing programs that use numbers in a base other than 10, set the base mode both as the current setting for the calculator and in the program (as an instruction).

Selecting a Base Mode in a Program

Insert a BIN, OCT, or HEX instruction into the beginning of the program. You should usually include a DEC instruction at the end of the program so that the calculator's setting will revert to Decimal mode when the program is done.

An instruction in a program to change the base mode will determine how input is interpreted and how output looks *during and after program execution,* but it does *not* affect the program lines as you enter them.

Numbers Entered in Program Lines

Before starting program entry, set the base mode. The current setting for the base mode determines the result of program.

An annunciator tells you which base is the current setting. Compare the program lines below in the decimal and non-decimal mode. All decimal and non-decimal numbers are left–justified in the calculator's display.

Decimal mode set:		**Binary mode set:**	
:		:	
:		:	
PRGM		**PRGM BIN**	
`A009 BIN`		`A009 BIN`	
`A010 10`	Decimal number can omit the sign “d”	`A010 10b`	Binary number should add the base sign ″b”
:		:	
:		:	

Polynomial Expressions and Horner's Method

Some expressions, such as polynomials, use the same variable several times for their solution. For example, the expression

$$Ax^4 + Bx^3 + Cx^2 + Dx + E$$

uses the variable x four different times. A program to calculate such an expression using RPN operations could repeatedly recall a stored copy of x from a variable.

Example:

Write a program using RPN operations for $5x^4 + 2x^3$, then evaluate it for $x = 7$.

Keys: (In RPN mode)	Display:	Description:
↱ PRGM GTO		
· ·	PRGM TOP	
↱ LBL A	A001 LBL A	
↰ INPUT X	A002 INPUT X	
5	A003 5	5
RCL X	A004 RCL X	
4	A005 4	
y^x	A006 y^x	x^4
×	A007 ×	$5x^4$
RCL X	A008 RCL X	
3	A009 3	
y^x	A010 y^x	x^3
2	A011 2	
×	A012 ×	$2x^3$
+	A013 +	$5x^4 + 2x^3$
↰ RTN	A014 RTN	
↰ MEM 2	LBL A	Displays label *A*, which
(2PGM)	LN=46	takes 46 bytes.
↰ SHOW	CK=EA18	Checksum and length.
	LN=46	
C C		Cancels program entry.

Now evaluate this polynomial for $x = 7$.

Keys: (In RPN mode)	Display:	Description:
XEQ A ENTER	X?	Prompts for *x*.
	value	
7 R/S	12,691.0000	Result.

A more general form of this program for any equation $Ax^4 + Bx^3 + Cx^2 + Dx + E$ would be:

```
A001  LBL A
A002  INPUT A
A003  INPUT B
A004  INPUT C
A005  INPUT D
A006  INPUT E
A007  INPUT X
A008  RCL X
A009  RCL× A
A010  RCL+ B
A011  RCL× X
A012  RCL+ C
A013  RCL× X
A014  RCL+ D
A015  RCL× X
A016  RCL+ E
A017  RTN
```

Checksum and length: 9E5E 51

14

Programming Techniques

Chapter 13 covered the basics of programming. This chapter explores more sophisticated but useful techniques:

- Using subroutines to simplify programs by separating and labeling portions of the program that are dedicated to particular tasks. The use of subroutines also shortens a program that must perform a series of steps more than once.
- Using conditional instructions (comparisons and flags) to determine which instructions or subroutines should be used.
- Using loops with counters to execute a set of instructions a certain number of times.
- Using indirect addressing to access different variables using the same program instruction.

Routines in Programs

A program is composed of one or more *routines.* A routine is a functional unit that accomplishes something specific. Complicated programs need routines to group and separate tasks. This makes a program easier to write, read, understand, and alter.

A routine typically starts at a label and ends with an instruction that stops program/ routing execution such as RTN or STOP.

Calling Subroutines (XEQ, RTN)

A *subroutine* is a routine that is *called from* (executed by) another routine and *returns to* that same routine when the subroutine is finished.

- If you plan to have only one program in the calculator memory, you can separate the routine in various labels. If you plan to have more than one program in the calculator memory, it is better to have routines part of the main program label, starting at a specific line number.
- A subroutine can itself call other subroutines.

The flow diagrams in this chapter use this notation:

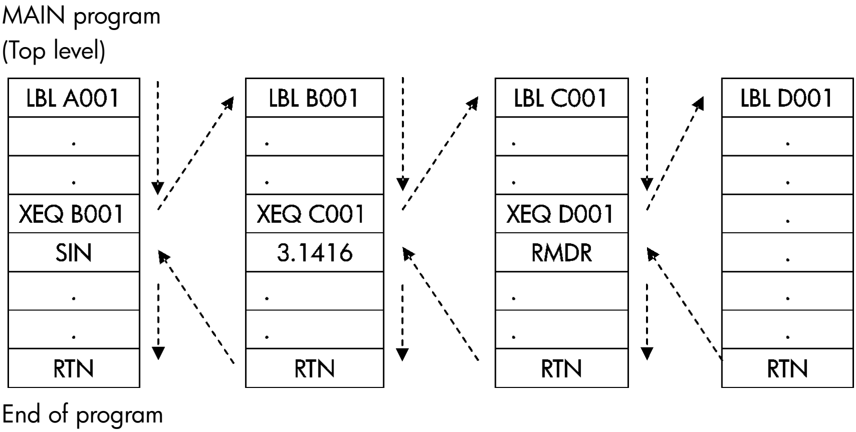
`A005 GTO B001` → ①	Program execution branches from this line to the line number marked ← ① ("from 1").
`B001 LBL B` ← ①	Program execution branches from a line number marked → ① ("to 1") to this line.

The example below show you to call a subroutine to change the sign of the number you input. Subroutine E that is called from routine D by line `D003 XEQ E001` changes sign of the number. Subroutine E ends with a RTN instruction that sends program execution back to routine D (to store and display the result) at line D004. See the flow diagrams below.

```
D001 LBL D                Starts here.
D002 INPUT X
D003 XEQ E001     → ①     Calls subroutine E.
D004 STO X        ← ②     Returns here.
D005 VIEW X
D006 RTN
E001 LBL E        ← ①     Starts subroutine.
E002 +/-                  Change sign of the number
E003 RTN          → ②     Returns to routine D.
```

Nested Subroutines

A subroutine can call another subroutine, and that subroutine can call yet another subroutine. This "nesting" of subroutines — the calling of a subroutine within another subroutine — is limited to a stack of subroutines 20 levels deep (not counting the topmost program level). The operation of nested subroutines is as shown below:



Attempting to execute a subroutine nested more than 20 levels deep causes an XEQ OVERFLOW error.

Example: A Nested Subroutine.

The following subroutine, labeled S, calculates the value of the expression

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

as part of a larger calculation in a larger program. The subroutine calls upon *another* subroutine (a nested subroutine), labeled Q, to do the repetitive squaring and addition. This saves memory by keeping the program shorter than it would be without the subroutine.

In RPN mode,

	`S001 LBL S`		Starts subroutine here.
	`S002 INPUT A`		Enters *A*.
	`S003 INPUT B`		Enters *B*.
	`S004 INPUT C`		Enters *C*.
	`S005 INPUT D`		Enters D.
	`S006 RCL D`		Recalls the data.
	`S007 RCL C`		
	`S008 RCL B`		
	`S009 RCL A`		
	`S010 x²`		A^2.
	`S011 XEQ Q001`	→ ①	$A^2 + B^2$.
② →	`S012 XEQ Q001`	→ ③	$A^2 + B^2 + C^2$
④ →	`S013 XEQ Q001`	→ ⑤	$A^2 + B^2 + C^2 + D^2$
⑥ →	`S014 √x`		$\sqrt{A^2 + B^2 + C^2 + D^2}$
	`S015 RTN`		Returns to main routine.
	`Q001 LBL Q`	← ①③⑤	Nested subroutine
	`Q002 x<>y`		
	`Q003 x²`		
	`Q004 +`		Adds x^2.
②④⑥ ←	`Q005 RTN`		Returns to subroutine S.

Branching (GTO)

As we have seen with subroutines, it is often desirable to transfer execution to a part of the program other than the next line. This is called **branching**.

Unconditional branching uses the GTO (*go to*) instruction to branch to a specific program line (label and line number).

A Programmed GTO Instruction

The GTO *label* instruction (press GTO *label line number*) transfers the execution of a running program to the specified program line. The program continues running from the new location, and *never* automatically returns to its point of origination, so GTO is not used for subroutines.

For example, consider the "Curve Fitting" program in chapter 16. The GTO Z 001 instruction branches execution from any one of three independent initializing routines to LBL Z, the routine that is the common entry point into the heart of the program:

```
S001 LBL S                 Can start here.
   .
   .
   .
S004 GTO Z001    →①        Branches to Z001.

L001 LBL L                 Can start here.
   .
   .
   .
L004 GTO Z001    →①        Branches to Z001.

E001 LBL E                 Can start here.
   .
   .
   .
E004 GTO Z001    →①        Branches to Z001.

Z001 LBL Z       ←①        Branch to here.
   .
   .
   .
```

Using GTO from the Keyboard

You can use GTO to move the program pointer to a specified label line number *without* starting program execution.

- To PRGM TOP: GTO · ·.
- To a specific line number: GTO *label line number* (*line number* < 1000). For example, GTO · A 0 0 5. For example, press GTO A 0 0 5. The display will show "GTO A005" .
- If you want to go to the first line of a label, for example. A001: GTO A ENTER (press and hold), the display will show "GTO A001".

Conditional Instructions

Another way to alter the sequence of program execution is by a *conditional test*, a true/false test that compares two numbers and skips the next program instruction if the proposition is false.

For instance, if a conditional instruction on line A005 is x=0? (that is, *is x equal to zero*?), then the program compares the contents of the X–register with zero. If the X–register *does* contain zero, then the program goes on to the next line. If the X–register does *not* contain zero, then the program *skips* the next line, thereby branching to line A007. This rule is commonly known as "Do if true."

```
                              A001 LBL A
                              .
                              .
                              .
   Do next if true.           A005 x=0?        → ②       Skip next if false.
                         ① ←  A006 GTO B001
                              A007 LN          ← ②
                              A008 STO A
                              .
                              .
                              .
                         ① →  B001 LBL B
                              .
                              .
                              .
```

The above example points out a common technique used with conditional tests: the line immediately after the test (which is only executed in the "true" case) is a *branch* to another label. So the net effect of the test is to branch to a different routine under certain circumstances.

There are three categories of conditional instructions:

- Comparison tests. These compare the X–and Y–registers, or the X–register and zero.
- Flag tests. These check the status of flags, which can be either set or clear.
- Loop counters. These are usually used to loop a specified number of times.

Tests of Comparison (x?y, x?0)

There are 12 comparisons available for programming. Pressing ⬑ x?y or ⬏ x?0 displays a menu for one of the two categories of tests:

- x?y for tests comparing *x* and *y.*
- x?0 for tests comparing x and 0.

Remember that *x* refers to the number in the X–register, and *y* refers to the number in the Y–register. These do *not* compare the *variables* X and Y. You can use x?y and x?0 to compare two numbers, if one of these isn't real number, it will return an error message INVALID DATA.

Select the category of comparison, then press the menu key for the conditional instruction you want.

The Test Menus

x?y	x?0
≠ for *x* ≠ *y*?	≠ for *x*≠0?
≤ for x≤*y*?	≤ for *x*≤0?
< for *x<y?*	< for *x*<0?
> for *x>y?*	> for *x*>0?
≥ for *x*≥*y?*	≥ for *x*≥0?
= for *x=y?*	= for *x*=0?

If you execute a conditional test from the keyboard, the calculator will display YES or NO.

For example, if *x* =2 and *y* =7, test *x<y* .

	Keys:	Display:
In RPN mode	[7] [ENTER] [2] [↰] [$x?y$] [>] [>] (<) [ENTER]	`YES`
In ALG mode	[7] [x◂▸y] [2] [↰] [$x?y$] [>] [>] (<) [ENTER]	`YES`

Example:

The "Normal and Inverse–Normal Distributions" program in chapter 16 uses the *x*<*y*? conditional in routine T:

Program Lines: **(In RPN mode)**	**Description**
. . .	
`T009 ÷`	Calculates the correction for X $_{guess}$.
`T010 STO+ X`	Adds the correction to yield a new X $_{guess}$.
`T011 ABS`	
`T012 0.0001`	
`T013 x<y?`	Tests to see if the correction is significant.
`T014 GTO T001`	Goes back to start of loop if correction is significant. Continues if correction is not significant.
`T015 RCL X`	
`T016 VIEW X`	Displays the calculated value of X.
. . .	

Line T009 calculates the correction for X$_{guess}$. Line T013 compares the absolute value of the calculated correction with 0.0001. If the value is less than 0.0001 ("Do If True"), the program executes line T014; if the value is equal to or greater than 0.0001, the program skips to line T015.

Flags

A flag is an indicator of status. It is either *set* (*true*) or clear (*false*). *Testing a flag* is another conditional test that follows the "Do if true" rule: program execution proceeds directly if the tested flag is set, and skips one line if the flag is clear.

Meanings of Flags

The HP 35s has 12 flags, numbered 0 through 11. All flags can be set, cleared, and tested from the keyboard or by a program instruction. The default state of all 12 flags is *clear*. The three–key memory clearing operation described in appendix B clears all flags. Flags are *not* affected by ↱ CLEAR 3 (3ALL) < (Y) ENTER.

- **Flags 0, 1, 2, 3, and 4** have no pre-assigned meanings. That is, their states will mean whatever you define them to mean in a given program. (See the example below.)

- **Flag 5,** when set, will interrupt a program when an overflow occurs within the program, displaying OVERFLOW and ⚠. An *overflow* occurs when a result exceeds the largest number that the calculator can handle. The largest possible number is substituted for the overflow result. If flag 5 is clear, a program with an overflow is not interrupted, though OVERFLOW is displayed briefly when the program eventually stops.

- **Flag 6** is *automatically* set by the calculator any time an overflow TOO BIG occurs (although you can also set flag 6 yourself). It has no effect, but can be tested. Besides, when using non-decimal bases in programs, flag 6 also gets set for TOO BIG in programs.

  Flags 5 and 6 allow you to control overflow conditions that occur during a program. Setting flag 5 stops a program at the line just after the line that caused the overflow. By testing flag 6 in a program, you can alter the program's flow or change a result anytime an overflow occurs.

- **Flags 7, 8 and 9** control the display of fractions. Flag 7 can also be controlled from the keyboard. When Fraction–display mode is toggled on or off by pressing ↱ FDISP, flag 7 is set or cleared as well.

Flag Status	**Fraction–Control Flags**		
	7	**8**	**9**
Clear (Default)	Fraction display off; display real numbers in the current display format.	Fraction denominators not greater than the /*c* value.	Reduce fractions to smallest form.
Set	Fraction display on; display real numbers as fractions.	Fraction denominators are factors of the /*c* Value.	No reduction of fractions. (Used only if flag 8 is set.)

- **Flag 10** controls program execution of equations:
  When flag 10 is clear (the default state), equations in running programs are evaluated and the result put on the stack.

  When flag 10 is set, equations in running programs are displayed as messages, causing them to behave like a VIEW statement:

  1. Program execution halts.
  2. The program pointer moves to the next program line.
  3. The equation is displayed without affecting the stack. You can clear the display by pressing ← or C. Pressing any other key executes that key's function.
  4. If the next program line is a PSE instruction, execution continues after a 1–second pause.

  The status of flag 10 is controlled only by execution of the SF and CF operations from the keyboard, or by SF and CF statements in programs.

- **Flag 11** controls prompting when executing equations in a program — *it doesn't affect automatic prompting during keyboard execution*:

  When flag 11 is clear (the default state), evaluation, SOLVE, and ∫ FN of equations in programs proceed without interruption. The current value of each variable in the equation is automatically recalled each time the variable is encountered. INPUT prompting is not affected.

  When flag 11 is set, each variable is prompted for when it is first encountered in the equation. A prompt for a variable occurs only once, regardless of the number of times the variable appears in the equation. When solving, no prompt occurs for the unknown; when integrating, no prompt occurs for the variable of integration. Prompts halt execution. Pressing R/S resumes the calculation using the value for the variable you keyed in, or the displayed (current) value of the variable if R/S is your sole response to the prompt.

  Flag 11 is automatically cleared after evaluation, SOLVE, or ∫ FN of an equation in a program. The status of flag 11 is also controlled by execution of the SF and CF operations from the keyboard, or by SF and CF statements in programs.

Annunciators for Set Flags

Flags 0, 1, 2, 3 and 4 have annunciators in the display that turn on when the corresponding flag is set. The presence or absence of **0**, **1**, **2**, **3** or **4** lets you know at any time whether any of these five flags is set or not. However, there is no such indication for the status of flags 5 through 11. The states of these flags can be determined by executing the FS? instruction from the keyboard. (See "Using Flags" below.)

Using Flags

Pressing [↰] [FLAGS] displays the FLAGS menu: SF CF FS?

After selecting the function you want, you will be prompted for the flag number (0–11). For example, press [↰] [FLAGS] [1](1SF) [0] to set flag 0; press [↰] [FLAGS] [1](1SF)[·][0] to set flag 10; press [↰] [FLAGS] [1](1SF) [·][1] to set flag 11.

FLAGS Menu

Menu Key	Description
SF *n*	*Set flag*. Sets flag *n*.
CF *n*	*Clear flag*. Clears flag *n*.
FS? *n*	*Is flag set?* Tests the status of flag *n*.

A flag test is a conditional test that affects program execution just as the comparison tests do. The FS? *n* instruction tests whether the given flag is set. If it is, then the next line in the program is executed. If it is not, then the next line is skipped. This is the "Do if True" rule, illustrated under "Conditional Instructions" earlier in this chapter.

If you test a flag from the keyboard, the calculator will display "YES" or "NO".

It is good practice in a program to make sure that any conditions you will be testing start out in a known state. Current flag settings depend on how they have been left by earlier programs that have been run. You should not *assume* that any given flag is clear, for instance, and that it will be set only if something in the program sets it. You should make *sure* of this by clearing the flag before the condition arises that might set it. See the example below.

Example: Using Flags.

Program Lines: **(In RPN mode)**	**Description:**
`S001 LBL S`	
`S002 CF 0`	Clears flag 0, the indicator for In *X*.
`S003 CF 1`	Clears flag 1, the indicator for In *Y*.
`S004 INPUT X`	Prompts for and stores X
`S005 FS? 0`	If flag 0 is set…
`S006 LN`	… takes the natural log of the X-input
`S007 STO X`	Stores that value in X after flag test
`S008 INPUT Y`	Prompts for and stores Y
`S009 FS?1`	If flag 1 is set…
`S010 LN`	… takes the natural log of the Y-input
`S011 STO Y`	Stores that value in Y after flag test
`S012 VIEW X`	Displays value
`S013 VIEW Y`	Displays value
`S014 RTN`	

Checksum and length: 16B3 42

If you write lines S002 CF0 and S003 CF1(as shown above), the flags 0 and 1 are cleared so lines S006 and S010 do not take the natural logarithms of the X- and Y-inputs.

If you replace lines S002 and S003 by SF 0 and CF 1, then flag 0 is set so line S006 takes the natural log of the X-input.

If you replace lines S002 and S003 by CF 0 and SF1, then flag 1 is set so line S010 takes the natural log of the Y-input.

If you replace lines S002 and S003 by SF0 and SF1, then flags 0 and 1 are set so lines S006 and S010 take the natural logarithms of the X- and Y-inputs.

Use above program to see how to use flags

Keys: (In RPN mode)	Display:	Description:
XEQ S ENTER	X? *value*	Executes label *S*; prompts for X value
1 R/S	Y? *value*	Stores 1 in X; prompts Y value
1 R/S	X= 1.0000	Stores 1 in X ;displays X value after flag test
R/S	Y= 1.0000	Displays Y value after flag test

You can try other three cases. Remember to press ↰ FLAGS 2(2CF) 0 and ↰ FLAGS 2(2CF) 1 to clear flag 1 and 0 after you try them.

Example: Controlling the Fraction Display.

The following program lets you exercise the calculator's fraction–display capability. The program prompts for and uses your inputs for a fractional number and a denominator (the */c* value). The program also contains examples of how the three fraction–display flags (7, 8, and 9) and the "message–display" flag (10) are used.

Messages in this program are listed as MESSAGE and are entered as equations:

1. Set Equation–entry mode by pressing EQN (the **EQN** annunciator turns on).
2. Press RCL *letter* for each alpha character in the message; press ↱ SPACE for each space character.
3. Press ENTER to insert the message in the current program line and end Equation–entry mode.

	Program Lines: (In RPN mode)	Description:
F001	LBL F	Begins the fraction program.
F002	CF 7	Clears three fraction flags.
F003	CF 8	
F004	CF 9	
F005	SF 10	Displays messages.
F006	DEC	Selects decimal base.
F007	INPUT V	Prompts for a number.
F008	INPUT D	Prompts for denominator (2 – 4095).
F009	RCL V	Displays message, then shows the decimal number.
F010	DECIMAL	
F011	PSE	
F012	STOP	
F013	RCL D	
F014	⁄c	Sets ⁄*c* value and sets flag 7.
F015	RCL V	
F016	MOST PRECISE	Displays message, then shows the fraction.
F017	PSE	
F018	STOP	
F019	SF 8	Sets flag 8.
F020	FACTOR DENOM	Displays message, then shows the fraction.
F021	PSE	
F022	STOP	
F023	SF 9	Sets flag 9.
F024	FIXED DENOM	Displays message, then shows the fraction.
F025	PSE	
F026	STOP	
F027	GTO F001	Goes to beginning of program.

Checksum and length: BE54 123

Use the above program to see the different forms of fraction display:

Keys: (In RPN mode)	Display:	Description:
XEQ F ENTER	V? *value*	Executes label *F*; prompts for a fractional number (*V*).
2 . 5 3 R/S	D? *value*	Stores 2.53 in V; prompts for denominator (D).
1 6 R/S	DECIMAL 16.0000 2.5300	Stores 16 as the */c* value. Displays message, then the decimal number.
R/S	MOST PRECISE 2 8/15 ▼ 2 8/15	Message indicates the fraction format (denominator is no greater than 16), then shows the fraction. ▼ indicates that the numerator is "a little below" 8.
R/S	FACTOR DENOM 2 1/2 ▲ 2 1/2	Message indicates the fraction format (denominator is factor of 16), then shows the fraction.
R/S	FIXED DENOM 2 8/16 ▲	Message indicates the fraction format (denominator is 16), then shows the fraction.
R/S C ↰ FLAGS 2(2CF) . 0	2.5300 2.5300	Stops the program and clears flag 10

Loops

Branching backwards — that is, to a label in a previous line — makes it possible to execute part of a program more than once. This is called *looping*.

```
D001 LBL D
D002 INPUT M
D003 INPUT N
D004 INPUT T
D005 GTO D001
```

This routine is an example of an *infinite loop*. It can be used to collect the initial data. After entering the three values, it is up to you to manually interrupt this loop by pressing XEQ label line number to execute other routines.

Conditional Loops (GTO)

When you want to perform an operation until a certain condition is met, but you don't know how many times the loop needs to repeat itself, you can create a loop with a conditional test and a GTO instruction.

For example, the following routine uses a loop to diminish a value A by a constant amount *B* until the resulting *A* is less than or equal to *B*.

Program lines: **(In RPN mode)**	**Description:**
`S001 LBL S`	
`S002 INPUT A`	
`S003 INPUT B`	
`S004 RCL A`	It is easier to recall *A* than to remember where it is in the stack.
`S005 RCL– B`	Calculates *A* – *B*.
`S006 STO A`	Replaces old A with new result.
`S007 RCL B`	Recalls constant for comparison.
`S008 x<y?`	Is B < new *A*?
`S009 GTO S004`	Yes: loops to repeat subtraction.
`S010 VIEW A`	No: displays new *A*.
`S011 RTN`	

Checksum and length: 2737 33

Loops with Counters (DSE, ISG)

When you want to execute a loop a specific number of times, use the ↰ ISG (*increment; skip if greater than*) or ↱ DSE (*decrement; skip if less than or equal to*) conditional function keys. Each time a loop function is executed in a program, it automatically *decrements* or *increments* a counter value stored in a variable. It compares the current counter value to a final counter value, then continues or exits the loop depending on the result.

For a count–down loop, use ↱ DSE *variable*

For a count–up loop, use ↰ ISG *variable*

These functions accomplish the same thing as a FOR–NEXT loop in BASIC:

FOR *variable = initial–value* TO *final–value* STEP *increment*
.
.
.
NEXT *variable*

A DSE instruction is like a FOR–NEXT loop with a negative increment.

After pressing a shifted key for ISG or DSE (↰ ISG or ↱ DSE), you will be prompted for a variable that will contain the *loop–control number* (described below).

The Loop–Control Number

The specified variable should contain a loop–control number *±ccccccc.fffii,* where:

- *±ccccccc* is the current counter value (1 to 12 digits). This value *changes* with loop execution.
- *fff* is the final counter value (must be three digits). This value does *not* change as the loop runs. An unspecified value for fff is assumed to be 000.

- *ii* is the interval for incrementing and decrementing (must be two digits or unspecified). This value does *not* change. An unspecified value for *ii* is assumed to be 01 (increment/decrement by 1).

Given the loop–control number *ccccccc.fffii,* DSE decrements *ccccccc* to *ccccccc — ii*, compares the new *ccccccc* with *fff,* and makes program execution skip the next program line if this *ccccccc* ≤ *fff.*

Given the loop–control number ccccccc.*fffii,* ISG increments *ccccccc* to *ccccccc + ii*, compares the new *ccccccc* with *fff,* and makes program execution skip the next program line if this *ccccccc* > *fff.*

If current value > final value, continue loop.	①→ ①←	W001 LBL W . . . W009 DSE A W010 GTO W001 W011 XEQ X001 . . .	→② ←②	If current value ≤ final value, exit loop.
If current value ≤ final value, continue loop.	①→ ①←	W001 LBL W . . . W009 ISG A W010 GTO W001 W011 XEQ X001 . . .	→② ←②	If current value > final value, exit loop.

For example, the loop–control number 0.050 for ISG means: start counting at zero, count up to 50, and increase the number by 1 each loop.

If the loop-control number is a complex number or vector, it will use the real part or first part to control the loop.

The following program uses ISG to loop 10 times in RPN mode. The loop counter (1.010) is stored in the variable Z. Leading and trailing zeros can be left off.

```
L001 LBL L
L002 1.01
L003 ST0 Z
L004 ISG Z
L005 GTO L004
L006 RTN
```

Press XEQ L ENTER, then press ↰ VIEW Z to see that the loop–control number is now 11.0100.

Indirectly Addressing Variables and Labels

Indirect addressing is a technique used in advanced programming to specify a variable or label *without specifying beforehand exactly which one*. This is determined when the program runs, so it depends on the intermediate results (or input) of the program.

Indirect addressing uses four different keys: I, (I), J , and (J).

These keys are active for many functions that take *A* through *Z* as variables or labels.

- *I and J* are variables whose contents can refer to another variable. It holds a number just like any other variable (*A* through *Z*).
- *(I) and (J)* are programming functions that directs, "Use the number in I or J to determine which variable or label to address."
  This is an *indirect address*. (A through Z are *direct addresses*.)

Both I and (I) are used together to create an indirect address and this applies to both J and (J) as well.

By itself, (I) or (J) is either undefined (no number in (I) or (J)) or uncontrolled (using whatever number happens to be left over in I or J).

The Variables "I" and "J"

You can store, recall, and manipulate the contents of I or J just as you can the contents of other variables. You can even solve for *I,J* and integrate using I or J . The functions listed below can use variable "*i*"(the variable J is the same).

STO I
RCL I
STO +,–, × ,÷ I
RCL +,–, × ,÷ I
INPUT I
VIEW I
∫ FN d I
SOLVE I
DSE I
ISG I
x < > I

The Indirect Address, (I) and (J)

Many functions that use A through Z (as variables or labels) can use (I) or (J) to refer to A through Z (variables or labels) or statistics registers *indirectly*. The function (I) or (J) uses the value in variable *I to J* to determine which variable, label, or register to address. The following table shows how.

If I/J contains:	Then (I)/(J) will address:
-1	variable A or label A
.	.
.	.
.	.
-26	variable Z or label Z
-27	*n* register
-28	Σ*x* register
-29	Σ*y* register
-30	Σx^2 register
-31	Σy^2 register
-32	Σ*xy* register
0	Unnamed Indirect variables start
.	.
.	.
.	.
800	The Max Address is 800
I<-32 or I>800 or variables undefined	error: INVALID (I)
J<-32 or I>800 or variables undefined	error: INVALID (J)

The INPUT**(I)** ,INPUT**(J)** and VIEW**(I)** ,VIEW**(J)**operations label the display with the name of the indirectly–addressed variable or register.

The SUMS menu enables you to recall values from the statistics registers. However, you must use indirect addressing to do other operations, such as STO, VIEW, and INPUT.

The functions listed below can use **(I) or (J)** as an address. For FN=, **(I) or (J)** refers to a label; for all other functions **(I) or (J)** refers to a variable or register.

STO**(I)/(J)**
RCL**(I)/(J)**
STO +, −,× ,÷, **(I)/(J)**
RCL +, −,× ,÷, **(I)/(J)**
X<>**(I)/(J)**
FN=**(I)/(J)**

INPUT**(I)/(J)**
VIEW**(I)/(J)**
DSE**(I)/(J)**
ISG**(I)/(J)**
SOLVE**(I)/(J)**
∫ FN d**(I)/(J)**

You can not solve or integrate for unnamed variables or statistic registers.

Program Control with (I)/(J)

Since the contents of *I* can change each time a program runs — or even in different parts of the same program — a program instruction such as STO (I) or (J) can store value to a different variable at different times. For example, STO(-1) indicates storing the value in Variable A. This maintains flexibility by leaving open (until the program runs) exactly which variable or program label will be needed.

Indirect addressing is very useful for counting and controlling loops. The variable *I or J* serves as an *index*, holding the address of the variable that contains the loop–control number for the functions DSE and ISG.

Equations with (I)/(J)

You can use **(I) or (J)** in an equation to specify a variable indirectly. Notice that (I) or (J) means the variable specified by the number in variable *I or J* (an *indirect* reference), but that *I* or *J and* (I) or (J) (where the user parenthesis are used instead of the (I) or (J) key) means variable I or J.

Unnamed indirect variables

Placing a positive number into variable I or J allows you to access up to 801 indirect variables. The following example indicates how to use them.

Program Lines: (In RPN mode)	Description:
`A001 LBL A`	
`A002 100`	
`A003 STO I`	
`A004 12345`	
`A005 STO (I)`	Defined the storage address range “0-100” and saved “12345” into address 100.
`A006 150`	
`A007 STO I`	
`A008 67890`	
`A009 STO (I)`	Saves “67890” into address 150. The defined indirect storage range is now “0-150”.
`A010 100`	
`A011 STO I`	
`A012 0`	
`A013 STO (I)`	Stores 0 into indirect register 100. The defined range is still “0-150”.
`A014 170`	
`A015 STO I`	
`A016 RCL(I)`	Display “INVALID (I)”, because address “170” is undefined
`A017 RTN`	

Note:

1. If you want to recall the value from an undefined storage address, the error message “`INVALID (I)`”will be shown”. (See A014)
2. The calculator allocates memory for variable 0 to the last non-zero variable. It is important to store 0 in variables after using them in order to release the memory. Each allocated indirect register uses 37 bytes of program memory.
3. There is a maximum of 800 variables.

15

Solving and Integrating Programs

Solving a Program

In chapter 7 you saw how you can enter an equation — it's added to the equation list — and then solve it for any variable. You can also enter a *program* that calculates a function, and then solve *it* for any variable. This is especially useful if the equation you're solving changes for certain conditions or if it requires repeated calculations.

To solve a programmed function:

1. Enter a program that defines the function. (See "To write a program for SOLVE" below.)
2. Select the program to solve: press ↰ FN= *label.* (You can skip this step if you're re–solving the same program.)
3. Solve for the unknown variable: press ↱ SOLVE *variable.*

Notice that FN= is required if you're solving a programmed function, but not if you're solving an equation from the equation list.

To halt a calculation, press C or R/S and the message `INTERRUPTED` will appear in line 2. The current best estimate of the root is in the unknown variable; use ↰ VIEW to view it without disturbing the stack. To resume the calculation, press R/S.

To write a program for SOLVE:

The program can use equations and ALG or RPN operations — in whatever combination is most convenient.

1. Begin the program with a *label*. This label identifies the function that you want SOLVE to evaluate (FN=*label*).
2. Include an INPUT instruction for each variable, including the unknown. INPUT instructions enable you to solve for any variable in a multi–variable function. INPUT for the *unknown* is ignored by the calculator, so you need to write only one program that contains a *separate* INPUT instruction for *every* variable (including the unknown).

   If you include no INPUT instructions, the program uses the values stored in the variables or entered at equation prompts.
3. Enter the instructions to evaluate the function.
   - A function programmed as a multi–line RPN or ALG sequence must be in the form of an expression that goes to zero at the solution. If your equation is *f(x) = g(x)*, your program should calculate *f(x) – g(x)*. "=0" is implied.
   - A function programmed as an equation can be any type of equation — equality, assignment, or expression. The equation is evaluated by the program, and its value goes to zero at the solution. If you want the equation to prompt for variable values instead of including INPUT instructions, make sure flag 11 is set.
4. End the program with a RTN. Program execution should end with the value of the function in the X–register.

Example: Program Using ALG.

Write a program using ALG operations that solves for any unknown in the equation for the "Ideal Gas Law." The equation is:

$$P \times V = N \times R \times T$$

where

P = Pressure (atmospheres or N/m^2).

V = Volume (liters).

N = Number of moles of gas.

R = The universal gas constant
(0.0821 liter–atm/mole–K or 8.314 J/mole–K).

T = Temperature (kelvins; K = °C + 273.1).

To begin, put the calculator in Program mode; if necessary, position the program pointer to the top of program memory.

Keys: (In ALG mode)	Display:	Description:
↱ PRGM		Sets Program mode.
GTO · ·	PRGM TOP	

Type in the program:

Program Lines: (In ALG mode)	Description:
G001 LBL G	Identifies the programmed function.
G002 INPUT P	Stores *P* for pressure
G003 INPUT V	Stores *V* for volume
G004 INPUT N	Stores *N* for number of moles of gas
G005 INPUT R	Stores *R* for gas constant
G006 INPUT T	Stores *T* for temp.
G007 P×V=N×R×T	Press EQN Pressure × volume = Moles × gas constant × temp.
G008 RTN	Ends the program.

Checksum and length: F425 33

Press C to cancel Program–entry mode.

Use program "G" to solve for the pressure of 0.005 moles of carbon dioxide in a 2–liter bottle at 24 °C.

Keys: (In ALG mode)	Display:	Description:
↰ FN= G		Selects "G" — the program. SOLVE evaluates to find the value of the unknown variable.
↱ SOLVE P	V? *value*	Selects *P*; prompts for *V*.
2 R/S	N? *value*	Stores 2 in V; prompts for *N*.

. 0 0 5 R/S	R?	Stores .005 in *N*; prompts for *R*.
	value	
. 0 8 2 1	T?	Stores .0821 in *R*; prompts for *T*.
R/S	*value*	
2 4 + 2 7	T?	Calculates *T*.
3 . 1 ENTER	297.1000	
R/S	SOLVING	Stores 297.1 in *T*; solves for *P*.
	P=	Pressure is 0.0610 atm.
	0.0610	

Example: Program Using Equation.

Write a program that uses an equation to solve the "Ideal Gas Law."

Keys: **(In RPN mode)**	**Display:**	**Description:**
↱ PRGM		Selects Program–entry mode.
GTO . .	PRGM TOP	Moves program pointer to top of the list of programs.
↱ LBL H	H001 LBL H	Labels the program.
↰ FLAGS 1		Enables equation prompting.
(1SF) . 1	H002 SF 11	
EQN		Evaluates the equation, clearing
RCL P ×		flag 11. (Checksum and length:
RCL V ↰ =		EDC8 9).
RCL N ×		
RCL R ×		
RCL T ENTER	H003 PxV=NxRxT	
↰ RTN	H004 RTN	Ends the program.
C	0.0610	Cancels Program–entry mode.

Checksum and length of program: DF52 21

Now calculate the change in pressure of the carbon dioxide if its temperature drops by 10 °C from the previous example.

Keys: (In RPN mode)	Display:	Description:
↱ STO L	0.0610	Stores previous pressure.
↰ FN= H	0.0610	Selects program “H.”
↱ SOLVE P	V? 2.0000	Selects variable *P*; prompts for *V*.
R/S	N? 0.0050	Retains 2 in *V*; prompts for *N*.
R/S	R? 0.0821	Retains .005 in *N*; prompts for *R*.
R/S	T? 297.1000	Retains .0821 in *R*; prompts for *T*.
ENTER 1 0 −	T? 287.1000	Calculates new *T*.
R/S	SOLVING P= 0.0589	Stores 287.1 in *T*; solves for new *P*.
RCL L −	-0.0021	Calculates pressure change of the gas when temperature drops from 297.1 K to 287.1 K (negative result indicates drop in pressure).

Using SOLVE in a Program

You can use the SOLVE operation as part of a program.

If appropriate, include or prompt for initial guesses (into the unknown variable and into the X–register) before executing the SOLVE *variable* instruction. The two instructions for solving an equation for an unknown variable appear in programs as:

`FN=` *label*

`SOLVE` *variable*

The *programmed* SOLVE instruction does not produce a labeled display (*variable* = *value*) since this might not be the significant output for your program (that is, you might want to do further calculations with this number before displaying it). If you *do* want this result displayed, add a VIEW *variable* instruction after the SOLVE instruction.

If no solution is found for the unknown variable, then the next program line is skipped (in accordance with the "Do if True" rule, explained in chapter 14). The program should then handle the case of not finding a root, such as by choosing new initial estimates or changing an input value.

Example: SOLVE in a Program.

The following excerpt is from a program that allows you to solve for *x* or *y* by pressing XEQ X or Y.

Program Lines: (In RPN mode)	Description:
`X001 LBL X`	Setup for *X*.
`X002 24`	Index for *X*.
`X003 GTO L001`	Branches to main routine.
Checksum and length: 62A0 11	
`Y001 LBL Y`	Setup for *Y*.
`Y002 25`	Index for *Y*.
`Y003 GTO L001`	Branches to main routine.
Checksum and length: 221E 11	
`L001 LBL L`	Main routine.
`L002 STO I`	Stores index in *I*
`L003 FN= F`	Defines program to solve.
`L004 SOLVE(I)`	Solves for appropriate variable.
`L005 VIEW(I)`	Displays solution.
`L006 RTN`	Ends program.
Checksum and length: D45B 18	
`F001 LBL F`	Calculates *f* (*x*,*y*). Include INPUT or equation prompting as required.
.	
.	
.	
`F010 RTN`	

Integrating a Program

In chapter 8 you saw how you can enter an equation (or expression) — it's added to the list of equations — and then integrate it with respect to any variable. You can also enter a *program* that calculates a function, and then integrate *it* with respect to any variable. This is especially useful if the function you're integrating changes for certain conditions or if it requires repeated calculations.

To integrate a programmed function:

1. Enter a program that defines the integrand's function. (See "To write a program for ∫ FN" below.)

2. Select the program that defines the function to integrate: press ↰ FN= *label*. (You can skip this step if you're reintegrating the same program.)
3. Enter the limits of integration: key in the *lower limit* and press ENTER, then key in the *upper limit*.
4. Select the variable of integration and start the calculation: press ↰ ∫ *variable*.

Notice that FN= is required if you're integrating a programmed function, but not if you're integrating an equation from the equation list.

You can halt a running integration calculation by pressing C or R/S and the message `INTERRUPTED` will appear line 2. However, the calculation cannot be resumed. No information about the integration is available until the calculation finishes normally.

Pressing XEQ while an integration calculation is running will cancel the ∫ FN= operation. In this case, you should start ∫ FN= again from the beginning.

To write a program for ∫ FN:

The program can use equations, ALG or RPN operations — in whatever combination is most convenient.

1. Begin the program with a *label*. This label identifies the function that you want to integrate (`FN=`*label*).
2. Include an INPUT instruction for each variable, including the variable of integration. INPUT instructions enable you to integrate with respect to any variable in a multi–variable function. INPUT for the variable of integration is ignored by the calculator, so you need to write only one program that contains a *separate* INPUT instruction for *every* variable (including the variable of integration).

   If you include no INPUT instructions, the program uses the values stored in the variables or entered at equation prompts.
3. Enter the instructions to evaluate the function.

   - A function programmed as a multi–line RPN or ALG sequence must calculate the function values you want to integrate.

- A function programmed as an equation is usually included as an expression specifying the integrand — though it can be any type of equation. If you want the equation to prompt for variable values instead of including INPUT instructions, make sure flag 11 is set.

4. End the program with a RTN. Program execution should end with the value of the function in the X–register.

Example: Program Using Equation.

The sine integral function in the example in chapter 8 is

$$Si(t) = \int_0^t \left(\frac{\sin x}{x}\right)dx$$

This function can be evaluated by integrating a program that defines the integrand:

```
S001 LBL S        Defines the function.
S002 SIN(X)÷X     The function as an expression. (Checksum and length:
                  0EE0 8).
S003 RTN          Ends the subroutine
Checksum and length of program: D57E 17
```

Enter this program and integrate the sine integral function with respect to *x* from 0 to 2 (*t* = 2).

Keys: (In RPN mode)	Display:	Description:
MODE 2 (2RAD)		Selects Radians mode.
↰ FN= S		Selects label *S* as the integrand.
0 ENTER 2	2_	Enters lower and upper limits of integration.
↰ ∫ X	INTEGRATING ∫ = 1.6054	Integrates function from 0 to 2; displays result.
MODE 1 (1DEG)	1.6054	Restores Degrees mode.

Using Integration in a Program

Integration can be executed from a program. Remember to include or prompt for the limits of integration before executing the integration, and remember that accuracy and execution time are controlled by the display format at the time the program runs. The two integration instructions appear in the program as:

FN= *label*

∫FN d *variable*

The *programmed* ∫ FN instruction does not produce a labeled display (∫ = *value*) since this might not be the significant output for your program (that is, you might want to do further calculations with this number before displaying it). If you *do* want this result displayed, add a PSE (⮫ PSE) or STOP (R/S) instruction to display the result in the X–register after the ∫ FN instruction.

If the PSE instruction immediately follows an equation that is displayed (Flag 10 set) during each iteration of integrating or solving, the equation will be displayed for 1 second and execution will continue until the end of each iteration. During the display of the equation, no scrolling or keyboard input is allowed.

Example: ∫ FN in a Program.

The "Normal and Inverse–Normal Distributions" program in chapter 16 includes an integration of the equation of the normal density function

$$\frac{1}{S\sqrt{2\pi}}\int_{M}^{D} e^{-(\frac{D-M}{S})^{2}/2}dD.$$

The $e^{((D-M)\div S)^2\div 2}$ function is calculated by the routine labeled F. Other routines prompt for the known values and do the other calculations to find *Q(D)*, the upper–tail area of a normal curve. The integration itself is set up and executed from routine Q:

```
Q001 LBL Q
Q002 RCL M      Recalls lower limit of integration.
Q003 RCL X      Recalls upper limit of integration. (X = D.)
Q004 FN= F      Specifies the function.
Q005 ∫FN d D    Integrates the normal function using the dummy variable D.
```

Restrictions on Solving and Integrating

The SOLVE *variable* and ∫ FN d *variable* instructions cannot call a routine that contains another SOLVE or ∫ FN instruction. That is, neither of these instructions can be used recursively. For example, attempting to calculate a multiple integral will result in an ∫(∫FN) error. Also, SOLVE and ∫ FN cannot call a routine that contains an FN=*label* instruction; if attempted, a SOLVE ACTIVE or ∫FN ACTIVE error will be returned. SOLVE cannot call a routine that contains an ∫ FN instruction (produces a SOLVE(∫FN) error), just as ∫ FN cannot call a routine that contains a SOLVE instruction (produces an ∫(SOLVE) error).

The SOLVE *variable* and ∫ FN d *variable* instructions in a program use one of the 20 pending subroutine returns in the calculator. (Refer to "Nested Subroutines" in chapter 14.)

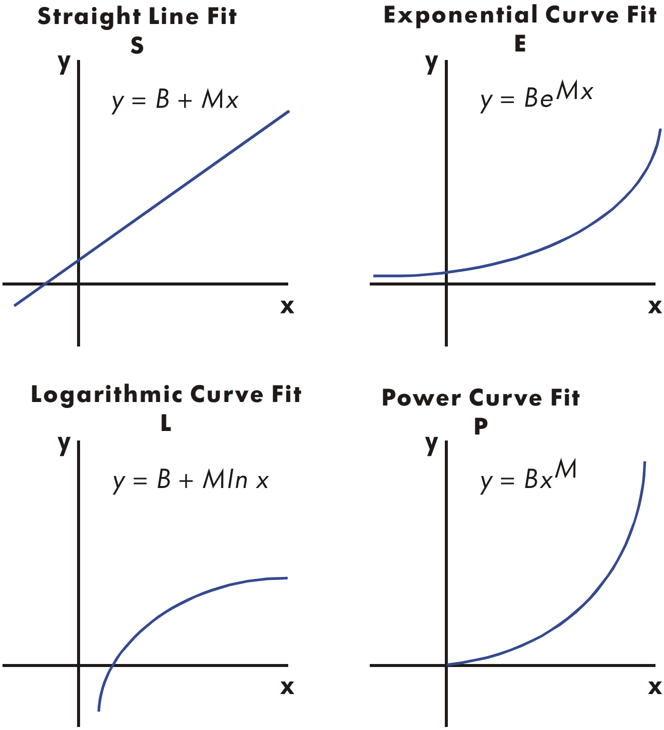
16

Statistics Programs

Curve Fitting

This program can be used to fit one of four models of equations to your data. These models are the straight line, the logarithmic curve, the exponential curve and the power curve. The program accepts two or more (*x*, *y*) data pairs and then calculates the correlation coefficient, *r*, and the two regression coefficients, *m* and *b*. The program includes a routine to calculate the estimates $\hat{x}$ and $\hat{y}$. (For definitions of these values, see "Linear Regression" in chapter 12.)

Samples of the curves and the relevant equations are shown below. The internal regression functions of the HP 35s are used to compute the regression coefficients.



To fit logarithmic curves, values of *x* must be positive. To fit exponential curves, values of *y* must be positive. To fit power curves, both *x* and *y* must be positive. A LOG(NEG) error will occur if a negative number is entered for these cases.

Data values of large magnitude but relatively small differences can incur problems of precision, as can data values of greatly different magnitudes. Refer to "Limitations in Precision of Data" in chapter 12.

Program Listing:

Program Lines: (In RPN mode)	Description
`S001 LBL S`	This routine sets, the status for the straight–line model.
`S002 CF 0`	Clears flag 0, the indicator for ln *X*.
`S003 CF 1`	Clears flag 1, the indicator for In *Y*.
`S004 GTO Z001`	Branches to common entry point *Z*.
Checksum and length: 8E85 12	
`L001 LBL L`	This routine sets the status for the logarithmic model.
`L002 SF 0`	Sets flag 0, the indicator for ln *X*.
`L003 CF 1`	Clears flag 1, the indicator for ln *Y*
`L004 GTO Z001`	Branches to common entry point *Z*.
Checksum and length: AD1B 12	
`E001 LBL E`	This routine sets the status for the exponential model.
`E002 CF 0`	Clears flag 0, the indicator for ln *X*.
`E003 SF 1`	Sets flag 1, the indicator for ln *Y*.
`E004 GTO Z001`	Branches to common entry point *Z*.
Checksum and length: D6F1 12	
`P001 LBL P`	This routine sets the status for the power model.
`P002 SF 0`	Sets flag 0, the indicator for ln *X*.
`P003 SF 1`	Sets flag 1, the indicator for ln *Y*.
Checksum and length: 3800 9	
`Z001 LBL Z`	Defines common entry point for all models.
`Z002 CLΣ`	Clears the statistics registers. Press ↱ CLEAR 4 (4Σ)
`Z003 0`	Sets the loop counter to zero for the first input.
Checksum and length: 8611 10	
`W001 LBL W`	Defines the beginning of the input loop.
`W002 1`	Adjusts the loop counter by one to prompt for input.
`W003 +`	
`W004 STO X`	Stores loop counter in *X* so that it will appear with the prompt for *X*.
`W005 INPUT X`	Displays counter with prompt and stores *X* input.

Program Lines: **(In RPN mode)**	**Description**
W006 FS? 0	If flag 0 is set . . .
W007 LN	. . . takes the natural log of the X–input.
W008 STO B	Stores that value for the correction routine.
W009 INPUT Y	Prompts for and stores *Y*.
W010 FS? 1	If flag 1 is set . . .
W011 LN	. . . takes the natural log of the Y–input.
W012 STO R	
W013 RCL B	
W014 Σ+	Accumulates *B* and *R* as *x,y*–data pair in statistics registers.
W015 GTO W001	Loops for another *X*, *Y* pair.

Checksum and length: 9560 46

U001 LBL U	Defines the beginning of the "undo" routine.
U002 RCL R	Recalls the most recent data pair.
U003 RCL B	
U004 Σ-	Deletes this pair from the statistical accumulation.
U005 GTO W001	Loops for another *X*, *Y* pair.

Checksum and length: A79F 15

R001 LBL R	Defines the start of the output routine
R002 r	Calculates the correlation coefficient.
R003 STO R	Stores it in *R*.
R004 VIEW R	Displays the correlation coefficient.
R005 b	Calculates the coefficient *b*.
R006 FS? 1	If flag 1 is set takes the natural antilog of *b*.
R007 e^X	
R008 STO B	Stores *b* in *B*.
R009 VIEW B	Displays value.
R010 m	Calculates coefficient *m*.
R011 STO M	Stores *m* in *M*.
R012 VIEW M	Displays value.

Checksum and length: 850C 36

Y001 LBL Y	Defines the beginning of the estimation (projection) loop.

Program Lines: (In RPN mode)	Description
`Y002 INPUT X`	Displays, prompts for, and, if changed, stores *x*–value in *X*.
`Y003 FS?0`	If flag 0 is set . . .
`Y004 GTO K001`	Branches to K001
`Y005 GTO M001`	Branches to M001
`Y006 STO Y`	Stores $\hat{y}$–value in *Y*.
`Y007 INPUT Y`	Displays, prompts for, and, if changed, stores *y*–value in *Y*.
`Y008 FS?0`	If flag 0 is set . . .
`Y009 GTO O001`	Branches to O001
`Y010 GTO N001`	Branches to N001
`Y011 STO X`	Stores $\hat{x}$ in *X* for next loop.
`Y012 GTO Y001`	Loops for another estimate.

Checksum and length: C3B7 36

`A001 LBL A`	This subroutine calculates $\hat{y}$ for the straight–line model.
`A002 RCL M`	
`A003 RCL× X`	
`A004 RCL+ B`	Calculates $\hat{y}$ = *MX* + *B*.
`A005 RTN`	Returns to the calling routine.

Checksum and length: 9688 15

`G001 LBL G`	This subroutine calculates $\hat{x}$ for the straight–line model.
`G002 RCL Y`	
`G003 RCL– B`	
`G004 RCL÷ M`	Calculates $\hat{x}$ =(*Y* – *B*) ÷ *M*.
`G005 RTN`	Returns to the calling routine.

Checksum and length: 9C0F 15

`B001 LBL B`	This subroutine calculates $\hat{y}$ for the logarithmic model.
`B002 RCL X`	
`B003 LN`	
`B004 RCL× M`	
`B005 RCL+ B`	Calculates $\hat{y}$ = *M* ln *X* + *B*.
`B006 RTN`	Returns to the calling routine.

Program Lines: **(In RPN mode)**	**Description**
Checksum and length: 889C 18	
H001 LBL H	This subroutine calculates $\hat{x}$ for the logarithmic model.
H002 RCL Y	
H003 RCL– B	
H004 RCL÷ M	
H005 e^X	Calculates $\hat{x} = e^{(Y-B) \div M}$
H006 RTN	Returns to the calling routine.
Checksum and length: 0DBE 18	
C001 LBL C	This subroutine calculates $\hat{y}$ for the exponential model.
C002 RCL M	
C003 RCL× X	
C004 e^X	
C005 RCL× B	Calculates $\hat{y} = Be^{MX}$.
C006 GTO M005	Branches to M005
Checksum and length: 9327 18	
I001 LBL I	This subroutine calculates $\hat{x}$ for the exponential model.
I002 RCL Y	
I003 RCL÷ B	
I004 LN	
I005 RCL÷ M	Calculates $\hat{x} = (\ln (Y \div B)) \div M$.
I006 GTO N005	Goes to N005
Checksum and length: 7219 18	
D001 LBL D	This subroutine calculates $\hat{y}$ for the power model.
D002 RCL X	
D003 RCL M	
D004 y^X	
D005 RCL× B	Calculates $Y = B (X^M)$.
D006 GTO K005	Goes to K005
Checksum and length: 11B3 18	
J001 LBL J	This subroutine calculates $\hat{x}$ for the power model.

Program Lines: (In RPN mode)	Description
J002 RCL Y	
J003 RCL÷ B	
J004 RCL M	
J005 1/x	
J006 y^x	Calculates $\hat{x}= (Y/B)^{1/M}$
J007GTO O005	Goes to O005

Checksum and length: 8524 21

K001 LBL K	Determines if D001 or B001 should be run
K002 FS?1	If flag 1 is set . . .
K003 XEQ D001	Executes D001
K004 XEQ B001	Executes B001
K005 GTO Y006	Goes to Y006

Checksum and length: 4BFA 15

M001 LBL M	Determines if C001 or A001 should be run
M002 FS?1	If flag 1 is set . . .
M003 XEQ C001	Executes C001
M004 XEQ A001	Executes A001
M005 GTO Y006	Goes to Y006

Checksum and length: 1C4D 15

O001 LBL O	Determines if J001 or H001 should be run
O002 FS?1	If flag 1 is set . . .
O003 XEQ J001	Executes J001
O004 XEQ H001	Executes H001
O005 GTO Y011	Goes to Y011

Checksum and length: 0AA5 15

N001 LBL N	Determines if I001 or G001 should be run
N002 FS?1	If flag 1 is set . . .
N003 XEQ I001	Executes I001
N004 XEQ G001	Executes G001
N005 GTO Y011	Goes to Y011

Checksum and length: 666D 15

Flags Used:

Flag 0 is set if a natural log is required of the *X* input. Flag 1 is set if a natural log is required of the *Y* input.

If flag 1 is set in routine N, then I001 is executed. If flag 1 is clear, G001 is executed.

Program instructions:

1. Key in the program routines; press C when done.
2. Press XEQ and select the type of curve you wish to fit by pressing:
   - S ENTER for a straight line;
   - L ENTER for a logarithmic curve;
   - E ENTER for an exponential curve; or
   - P ENTER for a power curve.
3. Key in *x*–value and press R/S.
4. Key in *y*–value and press R/S.
5. Repeat steps 3 and 4 for each data pair. If you discover that you have made an error after you have pressed R/S in step 3 (with the Y?*value* prompt still visible), press R/S again (displaying the X?*value* prompt) and press XEQ U ENTER to *undo* (remove) the last data pair. If you discover that you made an error after step 4, press XEQ U ENTER. In either case, continue at step 3.
6. After all data are keyed in, press XEQ R ENTER to see the correlation coefficient, *R*.
7. Press R/S to see the regression coefficient *B*.
8. Press R/S to see the regression coefficient *M*.
9. Press R/S to see the X?*value* prompt for the $\hat{x}$, $\hat{y}$–estimation routine.
10. If you wish to estimate $\hat{y}$ based on *x*, key in *x* at the X?*value* prompt, then press R/S to see $\hat{y}$ (Y?).
11. If you wish to estimate $\hat{x}$ based on *y*, press R/S until you see the Y?*value* prompt, key in *y*, then press R/S to see $\hat{x}$ (X?).
12. For more estimations, go to step 10 or 11.

13. For a new case, go to step 2.

Variables Used:

B	Regression coefficient (*y*–intercept of a straight line); also used for scratch.
M	Regression coefficient (slope of a straight line).
R	Correlation coefficient; also used for scratch.
X	The *x*–value of a data pair when entering data; the hypothetical *x* when projecting $\hat{y}$; or $\hat{x}$ (*x*–estimate) when given a hypothetical *y*.
Y	The *y*–value of a data pair when entering data; the hypothetical *y* when projecting $\hat{x}$; or $\hat{y}$ (*y*–estimate) when given a hypothetical *x*.
Statistics registers	Statistical accumulation and computation.

Example 1:

Fit a straight line to the data below. Make an intentional error when keying in the third data pair and correct it with the undo routine. Also, estimate *y* for an *x* value of 37. Estimate *x* for a *y* value of 101.

X	40.5	38.6	37.9	36.2	35.1	34.6
Y	104.5	102	100	97.5	95.5	94

Keys: **(In RPN mode)**	**Display:**	**Description:**

XEQ S ENTER	`X?`
	`1.0000`
		Starts straight–line routine.

4 0 . 5 R/S	`Y?` *value*	Enters *x*–value of data pair.

1 0 4 · 5 R/S	X? 2.0000	Enters *y*–value of data pair.
3 8 · 6 R/S	Y? 104.5000	Enters *x*–value of data pair.
1 0 2 R/S	X? 3.0000	Enters *y*–value of data pair.

Now intentionally enter 379 instead of 37.9 so that you can see how to correct incorrect entries.

Keys: **(In RPN mode)**	**Display:**	**Description:**
3 7 9 R/S	Y? 102.0000	Enters wrong *x*–value of data pair.
R/S	X? 4.0000	Retrieves X? prompt.
XEQ U ENTER	X? 3.0000	Deletes the last pair. Now proceed with the correct data entry.
3 7 · 9 R/S	Y? 102.0000	Enters correct *x*–value of data pair.
1 0 0 R/S	X? 4.0000	Enters *y*–value of data pair.
3 6 · 2 R/S	Y? 100.0000	Enters *x*–value of data pair.
9 7 · 5 R/S	X? 5.0000	Enters *y*–value of data pair.
3 5 · 1 R/S	Y? 97.5000	Enters *x*–value of data pair.
9 5 · 5 R/S	X? 6.0000	Enters *y*–value of data pair.
3 4 · 6 R/S	Y? 95.5000	Enters *x*–value of data pair.
9 4 R/S	X? 7.0000	Enters *y*–value of data pair.
XEQ R ENTER	R= 0.9955	Calculates the correlation coefficient.

R/S	B= 33.5271	Calculates regression coefficient *B*.
R/S	M= 1.7601	Calculates regression coefficient *M*.
R/S	X? 7.0000	Prompts for hypothetical *x*–value.
3 7 R/S	Y? 98.6526	Stores 37 in *X* and calculates $\hat{y}$.
1 0 1 R/S	X? 38.3336	Stores 101 in Y and calculates $\hat{x}$.

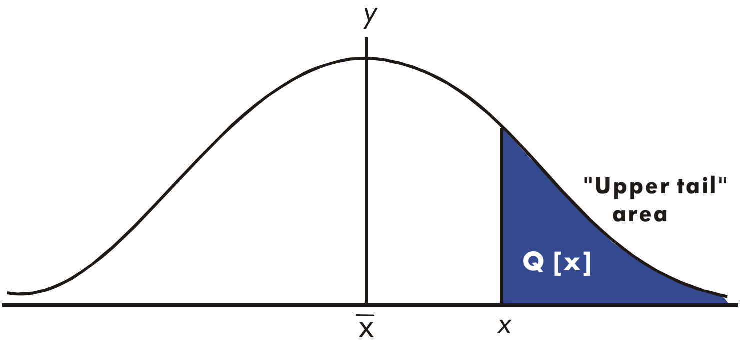
Example 2:

Repeat example 1 (using the same data) for logarithmic, exponential, and power curve fits. The table below gives you the starting execution label and the results (the correlation and regression coefficients and the *x*– and *y*– estimates) for each type of curve. You will need to reenter the data values each time you run the program for a different curve fit.

	Logarithmic	**Exponential**	**Power**
To start:	XEQ L ENTER	XEQ E ENTER	XEQ P ENTER
R	0.9965	0.9945	0.9959
B	–139.0088	51.1312	8.9730
M	65.8446	0.0177	0.6640
Y ($\hat{y}$ when *X*=37)	98.7508	98.5870	98.6845
X ($\hat{x}$ when *Y*=101)	38.2857	38.3628	38.3151

Normal and Inverse–Normal Distributions

Normal distribution is frequently used to model the behavior of random variation about a mean. This model assumes that the sample distribution is symmetric about the mean, *M,* with a standard deviation, *S*, and approximates the shape of the bell–shaped curve shown below. Given a value *x*, this program calculates the probability that a random selection from the sample data will have a higher value. This is known as the upper tail area, *Q(x)*. This program also provides the inverse: given a value *Q(x)*, the program calculates the corresponding value *x*.



$$Q(x) = 0.5 - \frac{1}{\sigma\sqrt{2\pi}} \int_{\bar{x}}^{x} e^{-((x-\bar{x})\div\sigma)^2 \div 2} dx$$

This program uses the built–in integration feature of the HP 35s to integrate the equation of the normal frequency curve. The inverse is obtained using Newton's method to iteratively search for a value of *x* which yields the given probability *Q(x)*.

Program Listing:

Program Lines: (In RPN mode)	Description
S001 LBL S	This routine initializes the normal distribution program.
S002 0	Stores default value for mean.
S003 STO M	
S004 INPUT M	Prompts for and stores mean, *M*.
S005 1	Stores default value for standard deviation.
S006 STO S	
S007 INPUT S	Prompts for and stores standard deviation, *S*.
S008 RTN	Stops displaying value of standard deviation.

Checksum and length: 70BF 26

D001 LBL D	This routine calculates *Q*(*X*) given *X*.
D002 INPUT X	Prompts for and stores *X*.
D003 XEQ Q001	Calculates upper tail area.
D004 STO Q	Stores value in *Q* so VIEW function can display it.
D005 VIEW Q	Displays *Q*(*X*).
D006 GTO D001	Loops to calculate another *Q(X)*.

Checksum and length: 042A 18

I001 LBL I	This routine calculates *X* given *Q*(*X*).
I002 INPUT Q	Prompts for and stores *Q*(*X*).
I003 RCL M	Recalls the mean.
I004 STO X	Stores the mean as the guess for *X*, called X_{guess}.

Checksum and length: A970 12

T001 LBL T	This label defines the start of the iterative loop.
T002 XEQ Q001	Calculates (*Q*(X_{guess})– *Q*(*X*)).
T003 RCL– Q	
T004 RCL X	
T005 STO D	
T006 R↓	
T007 XEQ F001	Calculates the derivative at X_{guess}.
T008 RCL÷ T	
T009 ÷	Calculates the correction for X_{guess}.

Program Lines: (In RPN mode)	Description
T010 STO+ X	Adds the correction to yield a new X_{guess}.
T011 ABS	
T012 0.0001	
T013 x<y?	Tests to see if the correction is significant.
T014 GTO T001	Goes back to start of loop if correction is significant.
	Continues if correction is not significant.
T015 RCL X	
T016 VIEW X	Displays the calculated value of *X*.
T017 GTO I001	Loops to calculate another *X*.

Checksum and length: EDF4 57

Program Lines: (In RPN mode)	Description
Q001 LBL Q	This subroutine calculates the upper–tail area *Q*(*x*).
Q002 RCL M	Recalls the lower limit of integration.
Q003 RCL X	Recalls the upper limit of integration.
Q004 FN= F	Selects the function defined by LBL F for integration.
Q005 ∫FN d D	Integrates the normal function using the dummy variable *D*.
Q006 2	
Q007 π	
Q008 ×	
Q009 √x	
Q010 RCL× S	Calculates $S \times \sqrt{2\pi}$.
Q011 STO T	Stores result temporarily for inverse routine.
Q012 ÷	
Q013 +/-	
Q014 0.5	
Q015 +	Adds half the area under the curve since we integrated using the mean as the lower limit.
Q0016 RTN	Returns to the calling routine.

Checksum and length: 8387 52

Program Lines: (In RPN mode)	Description
F001 LBL F	This subroutine calculates the integrand for the normal function $e^{-((X-M)\div S)^2\div 2}$
F002 RCL D	
F003 RCL- M	

Program Lines: (In RPN mode)	Description
`F004 RCL÷ S`	
`F005 x²`	
`F006 2`	
`F007 ÷`	
`F008 +/-`	
`F009 eˣ`	
`F010 RTN`	Returns to the calling routine.

Checksum and length: B3EB 31

Flags Used:

None.

Remarks:

The accuracy of this program is dependent on the display setting. For inputs in the area between ±3 standard deviations, a display of four or more significant figures is adequate for most applications.

At full precision, the input limit becomes ±5 standard deviations. Computation time is significantly less with a lower number of displayed digits.

In routine Q, the constant 0.5 may be replaced by 2 and 1/x.

You do not need to key in the inverse routine (in routines I and T) if you are not interested in the inverse capability.

Program Instructions:

1. Key in the program routines; press C when done.
2. Press XEQ S ENTER.
3. After the prompt for *M*, key in the population mean and press R/S. (If the mean is zero, just press R/S.)

4. After the prompt for *S*, key in the population standard deviation and press R/S. (If the standard deviation is 1, just press R/S.)
5. To calculate *X* given *Q*(*X*), skip to step 9 of these instructions.
6. To calculate *Q*(*X*) given *X*, XEQ D ENTER.
7. After the prompt, key in the value of *X* and press R/S. The result, *Q*(*X*), is displayed.
8. To calculate *Q*(*X*) for a new *X* with the same mean and standard deviation, press R/S and go to step 7.
9. To calculate *X* given *Q*(*X*), press XEQ I ENTER.
10. After the prompt, key in the value of *Q*(*X*) and press R/S. The result, *X,* is displayed.
11. To calculate *X* for a new *Q*(*X*) with the same mean and standard deviation, press R/S and go to step 10.

Variables Used:

D	Dummy variable of integration.
M	Population mean, default value *zero.*
Q	Probability corresponding to the upper–tail area.
S	Population standard deviation, default value of 1.
T	Variable used temporarily to pass the value $S \times \sqrt{2\pi}$ to the inverse program.
X	Input value that defines the left side of the upper–tail area.

Example 1:

Your good friend informs you that your blind date has "3σ" intelligence. You interpret this to mean that this person is more intelligent than the local population except for people more than three standard deviations above the mean.

Suppose that you intuit that the local population contains 10,000 possible blind dates. How many people fall into the "3σ" band? Since this problem is stated in terms of standard deviations, use the default value of zero for *M* and 1 for *S*.

Keys: (In RPN mode)	Display:	Description:

XEQ S ENTER	`M?`
	`0.0000`
		Starts the initialization routine.

R/S	S? 1.0000	Accepts the default value of zero for *M*.
R/S	1.0000	Accepts the default value of 1 for *S*.
XEQ D ENTER	X? *value*	Starts the distribution program and prompts for *X*.
3 R/S	Q= 0.0013	Enters 3 for *X* and starts computation of *Q*(*X*). Displays the ratio of the population smarter than everyone within three standard deviations of the mean.
1 0 0 0 0 ×	13.4984	Multiplies by the population. Displays the approximate number of blind dates in the local population that meet the criteria.

Since your friend has been known to exaggerate from time to time, you decide to see how rare a "2σ" date might be. Note that the program may be rerun simply by pressing R/S.

Keys: **(In RPN mode)**	**Display:**	**Description:**
R/S	X? 3.0000	Resumes program.
2 R/S	Q= 0.0228	Enters *X*–value of 2 and calculates *Q*(*X*).
1 0 0 0 0 ×	227.5012	Multiplies by the population for the revised estimate.

Example 2:

The mean of a set of test scores is 55. The standard deviation is 15.3. Assuming that the standard normal curve adequately models the distribution, what is the probability that a randomly selected student scored at least 90? What is the score that only 10 percent of the students would be expected to have surpassed? What would be the score that only 20 percent of the students would have failed to achieve?

Keys: **(In RPN mode)**	**Display:**	**Description:**
XEQ S ENTER	M? 0.0000	Starts the initialization routine.
5 5 R/S	S? 1.0000	Stores 55 for the mean.
1 5 · 3 R/S	15.3000	Stores 15.3 for the standard deviation.
XEQ D ENTER	X? *value*	Starts the distribution program and prompts for *X*.
9 0 R/S	Q= 0.0111	Enters 90 for *X* and calculates *Q*(*X*).

Thus, we would expect that only about 1 percent of the students would do better than score 90.

Keys: **(In RPN mode)**	**Display:**	**Description:**
XEQ I ENTER	Q? 0.0111	Starts the inverse routine.
0 · 1 R/S	X= 74.6077	Stores 0.1 (10 percent) in *Q*(*X*) and calculates *X*.
R/S	Q? 0.1000	Resumes the inverse routine.
0 · 8 R/S	X= 42.1232	Stores 0.8 (100 percent minus 20 percent) in *Q*(*X*) and calculates *X*.

Grouped Standard Deviation

The standard deviation of grouped data, S_{xy}, is the standard deviation of data points $x_1, x_2, \ldots, x_n$, occurring at positive integer frequencies $f_1, f_2, \ldots, f_n$.

$$S_{xg} = \sqrt{\frac{\sum x_i^2 f_i - \frac{(\sum x_i f_i)^2}{\sum f_i}}{(\sum f_i) - 1}}$$

This program allows you to input data, correct entries, and calculate the standard deviation and weighted mean of the grouped data.

Program Listing:

Program Lines: (In ALG mode)	Description
`S001 LBL S`	Start grouped standard deviation program.
`S002 CLΣ`	Clears statistics registers (-27 through -32).
`S003 0`	
`S004 STO N`	Clears the count *N*.

Checksum and length: E5BC 13

`I001 LBL I`	Input statistical data points.
`I002 INPUT X`	Stores data point in *X*.
`I003 INPUT F`	Stores data–point frequency in *F*.
`I004 1`	Enters increment for *N*.
`I005 STO B`	
`I006 RCL F`	Recalls data–point frequency f_i.

Checksum and length: 3387 19

`F001 LBL F`	Accumulate summations.
`F002 -27`	
`F003 STO I`	Stores index for register -27.
`F004 RCL F`	
`F005 STO+(I)`	Updates $\sum f_i$ in register -27.
`F006 RCL× X`	$x_i f_i$
`F007 STO Z`	
`F008 -28`	
`F009 STO I`	Stores index for register -28.
`F010 RCL Z`	
`F011 STO+(I)`	Updates $\sum x_i f_i$ in register -28.
`F012 RCL× X`	$x_i^2 f_i$
`F013 STO Z`	Stores index for register -30.
`F014 -30`	
`F015 STO I`	
`F016 RCL Z`	

Program Lines: (In ALG mode)	Description
F017 STO+(I)	Updates $\sum x_i^2 f_i$ in register -30.
F018 RCL B	
F019 STO+ N	Increments (or decrements) *N*.
F020 RCL N	
F021 RCL F	
F022 ABS	
F023 STO F	
F024 VIEW N	Displays current number of data pairs.
F025 GTO I001	Goes to label line number*I* for next data input.

Checksum and length: F6CB 84

G001 LBL G	Calculates statistics for grouped data.
G002 sx	Grouped standard deviation.
G003 STO S	
G004 VIEW S	Displays grouped standard deviation.
G005 $\bar{x}$	Weighted mean.
G006 STO M	
G007 VIEW M	Displays weighted mean.
G008 GTO I001	Goes back for more points.

Checksum and length: DAF2 24

U001 LBL U	Undo data–entry error.
U002 -1	Enters decrement for *N*.
U003 STO B	
U004 RCL F	Recalls last data frequency input.
U005 +/-	Changes sign of f_i.
U006 STO F	
U007 GTO F001	Adjusts count and summations.

Checksum and length: 03F4 23

Flags Used:

None.

Program Instructions:

1. Key in the program routines; press C when done.
2. Press XEQ S ENTER to start entering new data.
3. Key in x_i –value (data point) and press R/S.
4. Key in f_i –value (frequency) and press R/S.
5. Press R/S after VIEWing the number of points entered.
6. Repeat steps 3 through 5 for each data point.

   If you discover that you have made a data-entry error (x_i or f_i) after you have pressed R/S in step 4, press XEQ U ENTER and then press R/S again. Then go back to step 3 to enter the correct data.
7. When the last data pair has been input, press XEQ G ENTER to calculate and display the grouped standard deviation.
8. Press R/S to display the weighted mean of the grouped data.
9. To add additional data points, press R/S and continue at step 3.
   To start a new problem, start at step 2.Variables Used:

X	Data point.
F	Data–point frequency.
N	Data–pair counter.
S	Grouped standard deviation.
M	Weighted mean.
i	Index variable used to indirectly address the correct statistics register.
Register -27	Summation Σf_i.
Register -28	Summation $\Sigma x_i f_i$.
Register -30	Summation $\Sigma x_i 2 f_i$.

Example:

Enter the following data and calculate the grouped standard deviation.

Group	**1**	**2**	**3**	**4**	**5**	**6**
$\mathbf{x_i}$	5	8	13	15	22	37
$\mathbf{f_i}$	17	26	37	43	73	115

Keys: **(In ALG mode)**	**Display:**	**Description:**
XEQ S ENTER	`X?` *value*	Prompts for the first x_i.
5 R/S	`F?` *value*	Stores 5 in *X*; prompts for first f_i.

1 7 R/S	`N=`
	`1.0000`
		Stores 17 in *F*; displays the counter.

R/S	`X?`
	`5.0000`
		Prompts for the second x_i.

8 R/S	`F?`
	`17.0000`
		Prompts for second f_i.

2 6 R/S	`N=`
	`2.0000`
		Displays the counter.

R/S	`X?`
	`8.0000`
		Prompts for the third x_i.

1 4 R/S	`F?`
	`26.0000`
		Prompts for the third f_i.

3 7 R/S	`N=`
	`3.0000`
		Displays the counter.

You erred by entering 14 instead of 13 for x_3. Undo your error by executing routine U:

XEQ U ENTER	`N=`
	`2.0000`
		Removes the erroneous data; displays the revised counter.

R/S	`X?`
	`14.0000`
		Prompts for new third x_i.

1 3 R/S	`F?`
	`37.0000`
		Prompts for the new third f_i.

R/S	`N=`
	`3.0000`
		Displays the counter.

R/S	`X?`
	`13.0000`
		Prompts for the fourth x_i.

1 5 R/S	`F?`
	`37.0000`
		Prompts for the fourth f_i.

4 3 R/S	N= 4.0000	Displays the counter.
R/S	X? 15.0000	Prompts for the fifth x_i.
2 2 R/S	F? 43.0000	Prompts for the fifth f_i.
7 3 R/S	N= 5.0000	Displays the counter.
R/S	X? 22.0000	Prompts for the sixth x_i.
3 7 R/S	F? 73.0000	Prompts for the sixth f_i.
1 1 5 R/S	N= 6.0000	Displays the counter.
XEQ G ENTER	S= 11.4118	Calculates and displays the grouped standard deviation (*sx*) of the six data points.
R/S	M= 23.4084	Calculates and displays weighted mean ($\bar{x}$).
C	23.4084	Clears VIEW.

17

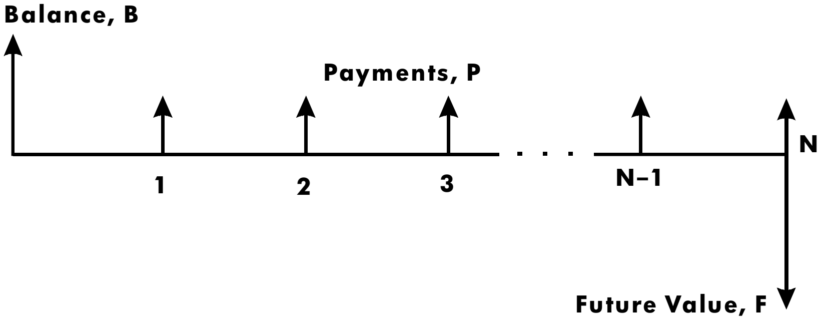
Miscellaneous Programs and Equations

Time Value of Money

Given any four of the five values in the "Time–Value–of–Money equation" (TVM), you can solve for the fifth value. This equation is useful in a wide variety of financial applications such as consumer and home loans and savings accounts.

The TVM equation is:

$$P\left[\frac{1-(1+I/100)^{-N}}{I/100}\right]+F(1+(I/100))^{-N}+B=0$$



The signs of the cash values (balance, *B*; payment, *P*; and future balance, *F*) correspond to the direction of the cash flow. Money that you receive has a positive sign while money that you pay has a negative sign. Note that any problem can be viewed from two perspectives. The lender and the borrower view the same problem with reversed signs.

Equation Entry:

Key in this equation:

```
Px100x(1-(1+I÷100)^-N)÷I+Fx(1+I÷100)^-N+B
```

Keys: (In RPN mode)	Display:	Description:
EQN	EQN LIST TOP or current equation	Selects Equation mode.
RCL P × 1 0 0	Px 100_	Starts entering equation.
× ( ) 1 −	Px100x(1-)	
( ) 1 +	Px100x(1-(1+))	
RCL I ÷ 1 0 0	⬅0x(1-(1+I÷ 100) ➡	
> y^x	⬅(1-(1+I÷100)^) ➡	
+/− RCL N >	⬅(1+I÷100)^-N)_	
÷ RCL I + RCL F	⬅100)^-N)÷I+Fx_	
×		
( ) 1 + RCL I	⬅^-N)÷I+Fx(1+I)	
÷ 1 0 0 >	⬅I+Fx(1+I÷100)_	
y^x +/− RCL N	⬅x(1+I÷100)^-N_	
+ RCL B	⬅1+I÷100)^-N+B_	
ENTER	Px100x(1-(1+I÷	➡Terminates the equation.

↰ SHOW (hold)	CK=CEFA
	LN=41
		Checksum and length.

Remarks:

The TVM equation requires that *I* must be non–zero to avoid a DIVIDE BY 0 error. If you're solving for *I* and aren't sure of its current value, press 1 ↱ STO I before you begin the SOLVE calculation (↱ SOLVE I).

The order in which you're prompted for values depends upon the variable you're solving for.

SOLVE instructions:

1. If your *first* TVM calculation is to solve for interest rate, I, press 1 ↱ STO I.
2. Press EQN. If necessary, press ⌃ or ⌄ to scroll through the equation list until you come to the TVM equation.
3. Do one of the following five operations:
   - **a.** Press ↱ SOLVE N to calculate the number of compounding periods.
   - **b.** Press ↱ SOLVE I to calculate periodic interest.

     For monthly payments, the result returned for *I* is the *monthly* interest rate, *i*; press 12 × to see the annual interest rate.
   - **c.** Press ↱ SOLVE B to calculate initial balance of a loan or savings account.
   - **d.** Press ↱ SOLVE P to calculate periodic payment.
   - **e.** Press ↱ SOLVE F to calculate future value or balance of a **loan**.
4. Key in the values of the four known variables as they are prompted for; press R/S after each value.
5. When you press the last R/S, the value of the unknown variable is calculated and displayed.
6. To calculate a new variable, or recalculate the same variable using different data, go back to step 2.

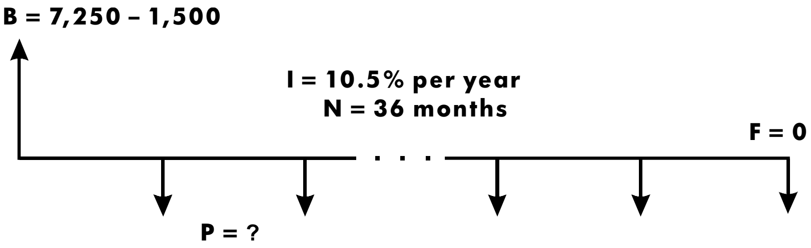
SOLVE works effectively in this application without initial guesses.

Variables Used:

N	The number of compounding periods.
I	The *periodic* interest rate as a percentage. (For example, if the *annual* interest rate is 15% and there are 12 payments per year, the *periodic* interest rate, *i*, is 15÷12=1.25%.)
B	The initial balance of loan or savings account.
P	The periodic payment.
F	The future value of a savings account or balance of a loan.

Example:

Part 1. You are financing the purchase of a car with a 3–year (36–month) loan at 10.5% annual interest compounded monthly. The purchase price of the car is $7,250. Your down payment is $1,500.



Keys:	Display:	Description:
(In RPN mode)		
DISPLAY 1 (iFIX) 2		Selects FIX 2 display format.
EQN (⌄ as needed)	Px100x(1-(1+I÷➡	Displays the leftmost part of the TVM equation.
⮫ SOLVE P	I? *value*	Selects P; prompts for *I*.
1 0 · 5 ENTER 1 2 ÷	I? 0.88	Converts your annual interest rate input to the equivalent monthly rate.
R/S	N? *value*	Stores 0.88 in *I*; prompts for *N*.
3 6 R/S	F? *value*	Stores 36 in *N*; prompts for *F*.

0 R/S	B? *value*	Stores 0 in *F*; prompts for *B*.
7 2 5 0 ENTER 1 5 0 0 −	B? 5,750.00	Calculates *B*, the beginning loan balance.
R/S	SOLVING P= -186.89	Stores 5750 in *B*; calculates monthly payment, *P*.

The answer is negative since the loan has been viewed from the borrower's perspective. Money received by the borrower (the beginning balance) is positive, while money paid out is negative.

Part 2. What interest rate would reduce the monthly payment by $10?

Keys: (In RPN mode)	Display:	Description:
EQN	`Px100x(1-(1+I÷`➡	Displays the leftmost hart of the TVM equation.

↱ SOLVE I	`P?`
	`-186.89`
		Selects *I*; prompts for *P*.

↱ RND	`P?`
	`-186.89`
		Rounds the payment to two decimal places.

1 0 +	`P?`
	`-176.89`
		Calculates new payment.

R/S	`N?`
	`36.00`
		Stores –176.89 in *P*; prompts for N.

R/S	`F?`
	`0.00`
		Retains 36 in *N*; prompts for *F*.

R/S	`B?`
	`5,750.00`
		Retains 0 in *F*; prompts for *B*.

R/S	`SOLVING`
	`I=`
	`0.56`
		Retains 5750 in *B;* calculates monthly interest rate.

1 2 ×	`6.75`	Calculates annual interest rate.

Part 3. Using the calculated interest rate (6.75%), assume that you sell the car after 2 years. What balance will you still owe? In other words, what is the future balance in 2 years?

Note that the interest rate, *I*, from part 2 is *not* zero, so you won't get a `DIVIDE BY 0` error when you calculate the new *I*.

Keys: (In RPN mode)	Display:	Description:
EQN	`Px100x(1-(1+I÷`➡	Displays leftmost part of the TVM equation.

↱ SOLVE F	`P?`
	`-176.89`
		Selects *F*; prompts for *P*.

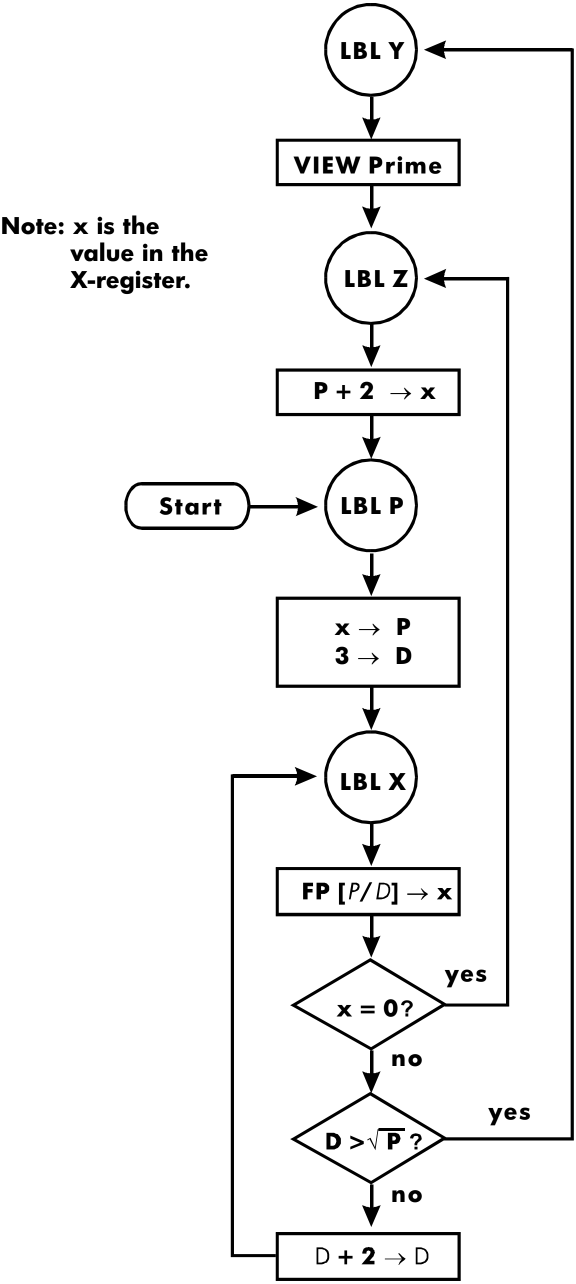
R/S	`I?`
	`0.56`
		Retains *P*; prompts for *I*.

[R/S]	N? 36.00	Retains 0.56 in *I*; prompts for *N*.
[2] [4] [R/S]	B? 5,750.00	Stores 24 in *N*; prompts for *B*.
[R/S]	SOLVING F= -2,047.05	Retains 5750 in *B*; calculates *F*, the future balance. Again, the sign is negative, indicating that you must, pay out this money.
[↰] [DISPLAY] [1] (1FIX) [4]		Sets FIX 4 display format.

Prime Number Generator

This program accepts any positive integer greater than 3. If the number is a prime number (not evenly divisible by integers other than itself and 1), then the program returns the input value. If the input is not a prime number, then the program returns the first prime number larger than the input.

The program identifies non–prime numbers by exhaustively trying all possible factors. If a number is not prime, the program adds 2 (assuring that the value is still odd) and tests to see if it has found a prime. This process continues until a prime number is found.


LBL Y
VIEW Prime
Note: x is the value in the X-register.
LBL Z
P + 2 → x
Start
LBL P
x → P
3 → D
LBL X
FP [P/D] → x
x = 0?
yes
no
D > √P ?
yes
no
D + 2 → D

Program Listing:

Program Lines: (In ALG mode)	Description
`Y001 LBL Y`	This routine displays prime number *P*.
`Y002 VIEW P`	
Checksum and length:	2CC5 6
`Z001 LBL Z`	This routine adds 2 to *P*.
`Z002 2+ P`	
Checksum and length:	EFB2 9
`P001 LBL P`	This routine stores the input value for *P*.
`P002 LASTx▶ P`	
`P003 FP(P÷2)`	
`P004 x<>y`	
`P005 0`	
`P006 x=y?`	Tests for *even* input
`P007 1+P▶P`	Increments *P* if input an even number.
`P008 3▶D`	Stores 3 in test divisor, *D*
Checksum and length:	EA89 47
`X001 LBL X`	This routine tests *P* to see if it is prime.
`X002 FP(P÷D)`	Finds the fractional part of *P* ÷ *D*.
`X003 x=0?`	Tests for a remainder of zero (*not* prime).
`X004 GTO Z001`	If the number is not prime, tries next possibility.
`X005 SQRT(P)`	
`X006 x<>y`	
`X007 D`	
`X008 x>y?`	Tests to see whether all possible factors have been tried.
`X009 GTO Y001`	If all factors have been tried, branches to the display routine.
`X010 2+D▶D`	
`X011 GTO X001`	Branches to test potential prime with new factor.
Checksum and length:	C6B5 53

Flags Used:

None.

Program Instructions:

1. Key in the program routines; press C when done.
2. Key in a positive integer greater than 3.
3. Press XEQ P ENTER to run program. Prime number, *P* will be displayed.
4. To see the next prime number, press R/S.

Variables Used:

P	Prime value and potential prime values.
D	Divisor used to test the current value of *P*.

Remarks:

No test is made to ensure that the input is greater than 3.

Example:

What is the first prime number after 789? What is the next prime number?

Keys: (In ALG mode)	Display:	Description:

7 8 9 XEQ
P ENTER
	`P=`
	`797.0000`
		Calculates next prime number after 789.

R/S	`P=`
	`809.0000`
		Calculates next prime number after 797.

Cross Product in Vectors

Here is an example showing how to use the program function to calculate the cross product.

Cross product:

$$\mathbf{v}_1 \times \mathbf{v}_2 = (YW - ZV)\mathbf{i} + (ZU - XW)\mathbf{j} + (XV - YU)\mathbf{k}$$

where

$\mathbf{v}_1 = X\,\mathbf{i} + Y\,\mathbf{j} + Z\,\mathbf{k}$

and

$\mathbf{v}_2 = U\,\mathbf{i} + V\,\mathbf{j} + W\,\mathbf{k}$

Program Lines: (In RPN mode)	Description
`R001 LBL R`	Defines the beginning of the rectangular input/display routine.
`R002 INPUT X`	Displays or accepts input of *X*.
`R003 INPUT Y`	Displays or accepts input of *Y*.
`R004 INPUT Z`	Displays or accepts input of *Z*.
`R005 GTO R001`	Goes to R001 to input vectors
Checksum and length: D82E 15	
`E001 LBL E`	Defines the beginning of the vector–enter routine.
`E002 RCL X`	Copies values in *X*, *Y* and *Z* to *U*, *V* and *W* respectively.
`E003 STO U`	
`E004 RCL Y`	
`E005 STO V`	
`E006 RCL Z`	
`E007 STO W`	
`E008 GTO R001`	Goes to R001 to input vectors
Checksum and length: B6AF 24	

Program Lines: (In RPN mode)	Description
C001 LBL C	Defines the beginning of the cross–product routine.
C002 RCL Y	
C003 RCL× W	
C004 RCL Z	
C005 RCL× V	
C006 –	Calculates (*YW – ZV*), which is the *X* component.
C007 STO A	
C008 RCL Z	
C009 RCL× U	
C010 RCL X	
C011 RCL× W	
C012 –	Calculates *(ZU – WX*), which is the *Y* component.
C013 STO B	
C014 RCL X	
C015 RCL× V	
C016 RCL Y	
C017 RCL× U	
C018 –	
C019 STO Z	Stores (*XV – YU),* which is the *Z* component.
C020 RCL A	
C021 STO X	Stores *X* component.
C022 RCL B	
C023 STO Y	Stores *Y* component.
C024 GTO R001	Goes to R001 to input vectors

Checksum and length: 838D 72

Example:

Calculate the cross product of two vectors, v1=2i+5j+4k and v2=i-2j+3k

Keys:	Display:	Description:
XEQ R ENTER	X? value	Run R routine to input vector value
1 R/S	y? value	Input v2 of x-component
2 +/- R/S	z? value	Input v2 of y-component
3 R/S	X? 1	Input v2 of z-component
XEQ E ENTER	X? 1	Run E routine to exchange v2 in U, V, and W variables
2 R/S	y? -2	Input v1 of x-component
5 R/S	z? 3	Input v1 of y-component
4 R/S	X? 2	Input v1 of z-component
XEQ C ENTER	X? 23	Run C routine to calculate x-component of cross product
R/S	y? -2	Calculate y-component of cross product
R/S	z? -9	Calculate z-component of cross product

Part 3

Appendixes and Reference

A

Support, Batteries, and Service

Calculator Support

You can obtain answers to questions about using your calculator from our Calculator Support Department. Our experience shows that many customers have similar questions about our products, so we have provided the following section, "Answers to Common Questions." If you don't find an answer to your question, contact the Calculator Support Department listed on page A–8.

Answers to Common Questions

Q: How can I determine if the calculator is operating properly?

A: Refer to page A–5, which describes the diagnostic self–test.

Q: My numbers contain commas instead of periods as decimal points. How do I restore the periods?

A: Use the ↰ DISPLAY 5 (5 ·) function (page 1–23).

Q: How do I change the number of decimal places in the display?

A: Use the ↰ DISPLAY menu (page 1–21).

Q: How do I clear all or portions of memory?

A: ↱ CLEAR displays the CLEAR menu, which allows you to clear x (the number in the X-register), all direct variables, all of memory, all statistical data, all stack levels and all indirect variables

Q: What does an "E" in a number (for example, 2.51E-13) mean?

A: *Exponent* of ten; that is, 2.51×10^{-13}.

Q: The calculator has displayed the message MEMORY FULL. What should I do?

A: You must clear a portion of memory before proceeding. (See appendix B.)

Q: Why does calculating the sine (or tangent) of π radians display a very small number instead of 0?

A: π cannot be represented *exactly* with the 12–digit precision of the calculator.

Q: Why do I get incorrect answers when I use the trigonometric functions?

A: You must make sure the calculator is using the correct angular mode (MODE 1DEG, 2RAD, or 3GRD).

Q: What does an *annunciator* in the display mean?

A: It indicates something about the status of the calculator. See "Annunciators" in chapter 1.

Q: Numbers show up as fractions. How do I get decimal numbers?

A: Press ↱ FDISP.

Environmental Limits

To maintain product reliability, observe the following temperature and humidity limits:

- Operating temperature: 0 to 45 °C (32 to 113 °F).
- Storage temperature: –20 to 65 °C (–4 to 149 °F).
- Operating and storage humidity: 90% relative humidity at 40 °C (104 °F) maximum.

Changing the Batteries

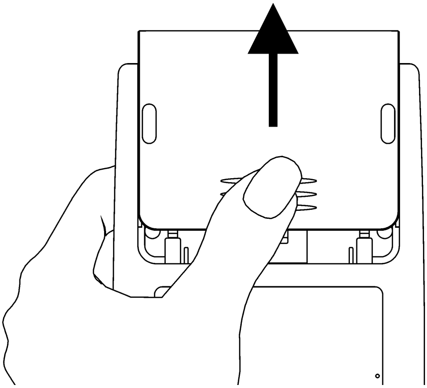
The calculator is powered by two 3-volt lithium coin batteries, CR2032.

Replace the batteries as soon as possible when the low battery annunciator (▭) appears. If the battery annunciator is on, and the display dims, you may lose data. If data is lost, the `MEMORY CLEAR` message is displayed.

Once you've removed the batteries, replace them within 2 minutes to avoid losing stored information. (Have the new batteries readily at hand before you open the battery compartment.)

To install batteries:

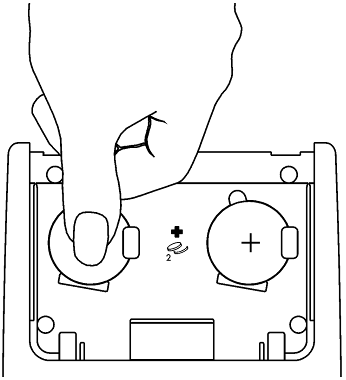
1. Have two fresh button–cell batteries at hand. Avoid touching the battery terminals — handle batteries only by their edges.
2. Make sure the calculator is OFF. **Do not press ON (C) again until the entire battery–changing procedure is completed. If the calculator is ON when the batteries are removed, the contents of Continuous Memory will be erased.** 用 **BK+B**
3. Turn the calculator over and slide off the battery cover.



4. To prevent memory loss, never remove two old batteries at the same time. Be sure to remove and replace the batteries one at a time.

Warning **Do not mutilate, puncture, or dispose of batteries in fire. The batteries can burst or explode, releasing hazardous chemicals.**

5. Insert a new CR2032 lithium battery, making sure that the positive sign (+) is facing outward.



6. Remove and insert the other battery as in steps 4 through 5. Make sure that the positive sign (+) on each battery is facing outward.

7. Replace the battery compartment cover.

8. Press C.

Testing Calculator Operation

Use the following guidelines to determine if the calculator is working properly. Test the calculator after every step to see if its operation has been restored. If your calculator requires service, refer to page A–8.

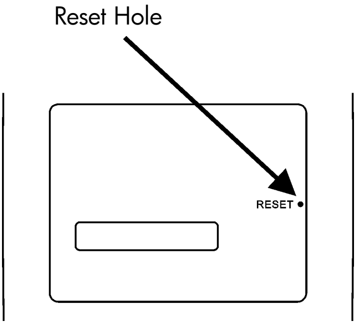
■ **The calculator won't turn on (steps 1–4) or doesn't respond when you press the keys (steps 1–3):**

1. Reset the calculator. Hold down the C key and press GTO. It may be necessary to repeat these reset keystrokes several times.

2. Erase memory. Press and hold down C, then press and hold down both R/S and i. Memory is cleared and the MEMORY CLEAR message is displayed when you release all three keys.

3. Remove the batteries (see "Changing the Batteries") and lightly press a coin against both battery contacts in the calculator. Replace the batteries and turn on the calculator. It should display `MEMORY CLEAR`.

4. If the calculator still does not respond to keystrokes, use a thin, pointed object to press the RESET hole. Stored data usually remain intact.



If these steps fail to restore calculator operation, it requires service.

- **If the calculator responds to keystrokes but you suspect that it is malfunctioning:**

1. Do the self–test described in the next section. If the calculator fails the self test, it requires service.

2. If the calculator passes the self–test, you may have made a mistake operating the calculator. Reread portions of the manual and check "Answers to Common Questions" (page A–1).

3. Contact the Calculator Support Department listed on page A–8.

The Self–Test

If the display can be turned on, but the calculator does not seem to be operating properly, do the following diagnostic self–test.

1. Hold down the C key, then press XEQ at the same time.

2. Press any key eight times and watch the various patterns displayed. After you've pressed the key eight times, the calculator displays the copyright message © `2007 HP DEV CO. L. P.` and then the message `KBD 01`.

3. Press the keys in the following sequence:

R/S → GTO → XEQ → MODE → ⌃ → < → > → RCL → R↓ →
x↔y → i → ⌄ → SIN → COS → TAN → $\sqrt{x}$ → y^x → 1/x →
ENTER → +/− → E → ( ) → ⬅ → EQN → 7 → 8 → 9 → ÷ → ↰ →
4 → 5 → 6 → × → ↱ → 1 → 2 → 3 → − → C → 0 → · →
Σ+ → +

- If you press the keys in the proper order and they are functioning properly, the calculator displays KBD followed by two–digit numbers. (The calculator is counting the keys using hexadecimal base.)
- If you press a key out of order, or if a key isn't functioning properly, the next keystroke displays a fail message (see step 4).

4. The self–test produces one of these two results:

- The calculator displays 35S-OK if it passed the self–test. Go to step 5.
- The calculator displays 35S-FAIL followed by a one–digit number, if it failed the self–test. If you received the message because you pressed a key out of order, reset the calculator (hold down C, press GTO) and do the self test again. If you pressed the keys in order, but got this message, repeat the self–test to verify the results. If the calculator fails again, it requires service (see page A–8). Include a copy of the fail message with the calculator when you ship it for service.

5. To exit the self–test, reset the calculator (hold down C and press GTO).

Pressing C and MODE starts a continuous self–test that is used at the factory. You can halt this factory test by pressing any key.

Warranty

HP 35s Scientific Calculator; Warranty period: 12 months

1. HP warrants to you, the end-user customer, that HP hardware, accessories and supplies will be free from defects in materials and workmanship after the date of purchase, for the period specified above. If HP receives notice of such defects during the warranty period, HP will, at its option, either repair or replace products which prove to be defective. Replacement products may be either new or like-new.
2. HP warrants to you that HP software will not fail to execute its programming instructions after the date of purchase, for the period specified above, due to defects in material and workmanship when properly installed and used. If HP receives notice of such defects during the warranty period, HP will replace software media which does not execute its programming instructions due to such defects.
3. HP does not warrant that the operation of HP products will be uninterrupted or error free. If HP is unable, within a reasonable time, to repair or replace any product to a condition as warranted, you will be entitled to a refund of the purchase price upon prompt return of the product with proof of purchase.
4. HP products may contain remanufactured parts equivalent to new in performance or may have been subject to incidental use.
5. Warranty does not apply to defects resulting from (a) improper or inadequate maintenance or calibration, (b) software, interfacing, parts or supplies not supplied by HP, (c) unauthorized modification or misuse, (d) operation outside of the published environmental specifications for the product, or (e) improper site preparation or maintenance.

6. HP MAKES NO OTHER EXPRESS WARRANTY OR CONDITION WHETHER WRITTEN OR ORAL. TO THE EXTENT ALLOWED BY LOCAL LAW, ANY IMPLIED WARRANTY OR CONDITION OF MERCHANTABILITY, SATISFACTORY QUALITY, OR FITNESS FOR A PARTICULAR PURPOSE IS LIMITED TO THE DURATION OF THE EXPRESS WARRANTY SET FORTH ABOVE. Some countries, states or provinces do not allow limitations on the duration of an implied warranty, so the above limitation or exclusion might not apply to you. This warranty gives you specific legal rights and you might also have other rights that vary from country to country, state to state, or province to province.
7. TO THE EXTENT ALLOWED BY LOCAL LAW, THE REMEDIES IN THIS WARRANTY STATEMENT ARE YOUR SOLE AND EXCLUSIVE REMEDIES. EXCEPT AS INDICATED ABOVE, IN NO EVENT WILL HP OR ITS SUPPLIERS BE LIABLE FOR LOSS OF DATA OR FOR DIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL (INCLUDING LOST PROFIT OR DATA), OR OTHER DAMAGE, WHETHER BASED IN CONTRACT, TORT, OR OTHERWISE. Some countries, States or provinces do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.
8. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. HP shall not be liable for technical or editorial errors or omissions contained herein.

FOR CONSUMER TRANSACTIONS IN AUSTRALIA AND NEW ZEALAND: THE WARRANTY TERMS CONTAINED IN THIS STATEMENT, EXCEPT TO THE EXTENT LAWFULLY PERMITTED, DO NOT EXCLUDE, RESTRICT OR MODIFY AND ARE IN ADDITION TO THE MANDATORY STATUTORY RIGHTS APPLICABLE TO THE SALE OF THIS PRODUCT TO YOU.

Customer Support

AP

Country :	Telephone numbers
Australia	1300-551-664 or 03-9841-5211

China	010-68002397
Hong Kong	2805-2563
Indonesia	+65 6100 6682
Japan	+852 2805-2563
Malaysia	+65 6100 6682
New Zealand	09-574-2700
Philippines	+65 6100 6682
Singapore	6100 6682
South Korea	2-561-2700
Taiwan	+852 2805-2563
Thailand	+65 6100 6682
Vietnam	+65 6100 6682

EMEA

Country :	Telephone numbers
Austria	01 360 277 1203
Belgium	02 620 00 86
Belgium	02 620 00 85
Czech Republic	296 335 612
Denmark	82 33 28 44
Finland	09 8171 0281
France	01 4993 9006
Germany	069 9530 7103
Greece	210 969 6421
Netherlands	020 654 5301
Ireland	01 605 0356
Italy	02 754 19 782
Luxembourg	2730 2146
Norway	23500027
Portugal	021 318 0093
Russia	495 228 3050
South Africa	0800980410
Spain	913753382
Sweden	08 5199 2065
Switzerland (French)	022 827 8780

Switzerland (German)	01 439 5358
Switzerland (Italian)	022 567 5308
United Kingdom	0207 458 0161

LA

Country :	Telephone numbers
Anguila	1-800-711-2884
Antigua	1-800-711-2884
Argentina	0-800- 555-5000
Aruba	800-8000 ♦ 800-711-2884
Bahamas	1-800-711-2884
Barbados	1-800-711-2884
Bermuda	1-800-711-2884
Bolivia	800-100-193
Brazil	0-800-709-7751
British Virgin Islands	1-800-711-2884
Cayman Island	1-800-711-2884
Curacao	001-800-872-2881 + 800-711-2884
Chile	800-360-999
Colombia	01-8000-51-4746-8368 (01-8000-51- HP INVENT)
Costa Rica	0-800-011-0524
Dominica	1-800-711-2884
Dominican Republic	1-800-711-2884
Ecuador	1-999-119 ♦ 800-711-2884 (Andinatel) 1-800-225-528 ♦ 800-711-2884 (Pacifitel)
El Salvador	800-6160
French Antilles	0-800-990-011♦ 800-711-2884
French Guiana	0-800-990-011♦ 800-711-2884
Grenada	1-800-711-2884
Guadelupe	0-800-990-011♦ 800-711-2884
Guatemala	1-800-999-5105
Guyana	159 ♦ 800-711-2884

Haiti	183 ♦ 800-711-2884
Honduras	800-0-123 ♦ 800-711-2884
Jamaica	1-800-711-2884
Martinica	0-800-990-011 ♦ 877-219-8671
Mexico	01-800-474-68368 (800 HP INVENT)
Montserrat	1-800-711-2884
Netherland Antilles	001-800-872-2881 ♦ 800-711-2884
Nicaragua	1-800-0164 ♦ 800-711-2884
Panama	001-800-711-2884
Paraguay	(009) 800-541-0006
Peru	0-800-10111
Puerto Rico	1-877 232 0589
St. Lucia	1-800-478-4602
St Vincent	01-800-711-2884
St. Kitts & Nevis	1-800-711-2884
St. Marteen	1-800-711-2884
Suriname	156 ♦ 800-711-2884
Trinidad & Tobago	1-800-711-2884
Turks & Caicos	01-800-711-2884
US Virgin Islands	1-800-711-2884
Uruguay	0004-054-177
Venezuela	0-800-474-68368 (0-800 HP INVENT)

NA

Country :	**Telephone numbers**
Canada	800-HP-INVENT
USA	800-HP INVENT

Please logon to http://www.hp.com for the latest service and support information.

Regulatory information

Federal Communications Commission Notice

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and the receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio or television technician for help.

Modifications
The FCC requires the user to be notified that any changes or modifications made to this device that are not expressly approved by Hewlett-Packard Company may void the user's authority to operate the equipment.

Declaration of Conformity for Products Marked with FCC Logo, United States Only
This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.
If you have questions about the product that are not related to this declaration, write to
Hewlett-Packard Company
P. O. Box 692000, Mail Stop 530113
Houston, TX 77269-2000
For questions regarding this FCC declaration, write to
Hewlett-Packard Company
P. O. Box 692000, Mail Stop 510101

Houston, TX 77269-2000
or call HP at 281-514-3333
To identify your product, refer to the part, series, or model number located on the product.

Canadian Notice

This Class B digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Avis Canadien

Cet appareil numérique de la classe B respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

European Union Regulatory Notice

This product complies with the following EU Directives:

- Low Voltage Directive 2006/95/EC
- EMC Directive 2004/108/EC

Compliance with these directives implies conformity to applicable harmonized European standards (European Norms) which are listed on the EU Declaration of Conformity issued by Hewlett-Packard for this product or product family.
This compliance is indicated by the following conformity marking placed on the product:

CE This marking is valid for non-Telecom products and EU harmonized Telecom products (e.g. Bluetooth).	CE XXXX*(!) This marking is valid for EU non-harmonized Telecom products. *Notified body number (used only if applicable - refer to the product label)

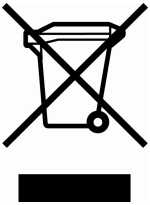
Hewlett-Packard GmbH, HQ-TRE, Herrenberger Strasse 140, 71034 Boeblingen, Germany

Japanese Notice

この装置は、情報処理装置等電波障害自主規制協議会（VCCI）の基準に基づくクラスB情報技術装置です。この装置は、家庭環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。

取扱説明書に従って正しい取り扱いをしてください。

Disposal of Waste Equipment by Users in Private Household in the European Union



This symbol on the product or on its packaging indicates that this product must not be disposed of with your other household waste. Instead, it is your responsibility to dispose of your waste equipment by handing it over to a designated collection point for the recycling of waste electrical and electronic equipment. The separate collection and recycling of your waste equipment at the time of disposal will help to conserve natural resources and ensure that it is recycled in a manner that protects human health and the environment. For more information about where you can drop off your waste equipment for recycling, please contact your local city office, your household waste disposal service or the shop where you purchased the product.

Perchlorate Material - special handling may apply

This calculator's Memory Backup battery may contain perchlorate and may require special handling when recycled or disposed in California.

B

User Memory and the Stack

This appendix covers

- The allocation and requirements of user memory,
- How to reset the calculator without affecting memory,
- How to clear (purge) all of user memory and reset the system defaults, and
- Which operations affect stack lift.

Managing Calculator Memory

The HP 35s has 30KB of user memory available to you for any combination of stored data (variables, equations, or program lines). SOLVE, ∫ FN, and statistical calculations also require user memory. (The ∫ FN operation is particularly "expensive" to run.)

All of your stored data is preserved until you explicitly clear it. The message `MEMORY FULL` means that there is currently not enough memory available for the operation you just attempted. You need to clear some (or all) of user memory. For instance, you can:

- Clear any or all equations (see "Editing and Clearing Equations" in chapter 6).
- Clear any or all programs (see "Clearing One or More Programs" in chapter 13).
- Clear all of user memory (press ↱ CLEAR 3 (`3ALL`)).

To see how much memory is available, press ↰ MEM. The display shows the number of bytes available.

To see the memory requirements of specific equations in the equation list:

1. Press EQN to activate Equation mode. (EQN LIST TOP or the left end of the current equation will be displayed.)
2. If necessary, scroll through the equation list (press ⌃ or ⌄) until you see the desired equation.
3. Press ↰ SHOW to see the checksum (hexadecimal) and length (in bytes) of the equation. For example, CK=382E LN=41.

To see the total memory requirements of specific programs:

1. Press ↰ MEM 2 (2PGM) to display the first label in the program list.
2. Scroll through the program list (press ⌃ or ⌄ until you see the desired program label and size). For example, LBL F LN=57.
3. Optional: Press ↰ SHOW to see the checksum (hexadecimal) and length (in bytes) of the program. For example, CK=9CC9 LN=57.

To see the memory requirements of an equation in a program:

1. Display the program line containing the equation.
2. Press ↰ SHOW to see the checksum and length. For example, CK=AB71 LN=15.

Resetting the Calculator

If the calculator doesn't respond to keystrokes or if it is otherwise behaving unusually, attempt to reset it. Resetting the calculator halts the current calculation and cancels program entry, digit entry, a running program, a SOLVE calculation, an ∫ FN calculation, a VIEW display, or an INPUT display. Stored data usually remain intact.

To reset the calculator, hold down the C key and press GTO. If you are unable to reset the calculator, try installing fresh batteries. If the calculator cannot be reset, or if it still fails to operate properly, you should attempt to clear memory using the special procedure described in the next section.

If the calculator still does not respond to keystrokes, use a thin, pointed object to press the RESET hole.
The calculator can reset itself if it is dropped or if power is interrupted.

Clearing Memory

The usual way to clear user memory is to press ↱ CLEAR 3 (3ALL). However, there is also a more powerful clearing procedure that resets additional information and is useful if the keyboard is not functioning properly.

If the calculator fails to respond to keystrokes, and you are unable to restore operation by resetting it or changing the batteries, try the following MEMORY CLEAR procedure. These keystrokes clear all of memory, reset the calculator, *and* restore all format and modes to their original, *default* settings (shown below):

1. Press and hold down the C key.
2. Press and hold down R/S.
3. Press i . (You will be pressing three keys simultaneously). When you release all three keys, the display shows MEMORY CLEAR if the operation is successful.

Category	CLEAR ALL	MEMORY CLEAR (Default)
Angular mode	Unchanged	Degrees
Base mode	Unchanged	Decimal
Contrast setting	Unchanged	Medium
Decimal point	Unchanged	“.“
Thousand separator	Unchanged	“1,000”
Denominator (/*c* value)	Unchanged	4095
Display format	Unchanged	FIX 4
Flags	Unchanged	Cleared
Complex mode	Unchanged	xiy
Fraction–display mode	Unchanged	Off
Random–number seed	Unchanged	Zero
Equation pointer	EQN LIST TOP	EQN LIST TOP
Equation list	Cleared	Cleared
FN = label	Null	Null
Program pointer	PRGM TOP	PRGM TOP
Program memory	Cleared	Cleared
Stack lift	Enabled	Enabled
Stack registers	Cleared to zero	Cleared to zero
Variables	Cleared to zero	Cleared to zero
Indirect Variables	Not defined	Not defined
Logic	Unchanged	RPN

Memory may inadvertently be cleared if the calculator is dropped or if power is interrupted.

The Status of Stack Lift

The four stack registers are always present, and the stack always has a *stack–lift status*. That is to say, the stack lift is always *enabled* or *disabled* regarding its behavior when the next number is placed in the X–register. (Refer to chapter 2, "The Automatic Memory Stack.")

All functions except those in the following two lists will enable stack lift.

Disabling Operations

The five operations ENTER, Σ+, Σ–, ↱ CLEAR 1 (1X) and ↱ CLEAR 5 (5STK) disable stack lift. A number keyed in after one of these disabling operations writes over the number currently in the X–register. The Y–, Z– and T–registers remain unchanged.

In addition, when C and ← act like CL*x*, they also disable stack lift.

The INPUT function *disables* stack lift as it halts a program for prompting (so any number then entered writes over the X-register), but it *enables* stack lift when the program resumes.

Neutral Operations

The following operations do not affect the status of stack lift:

DEG, RAD, GRAD	FIX, SCI, ENG, ALL	DEC, HEX, OCT, BIN	CLVARS
PSE	SHOW	RADIX . RADIX ,	CLΣ
OFF RCL +	R/S and STOP	⌃ and ⌄	C* and ←*
MEM 1 (1VAR)**	MEM 2 (2PGM)**	GTO . .	GTO . label nnn
EQN	FDISP	Errors	PRGM and program entry
Switching binary windows	Digit entry	xiy rθa	UNDO

∗ Except when used like CL*x*.

∗∗ Including all operations performed while the catalog is displayed except {VAR} ENTER and {PGM} XEQ, which enable stack lift.

The Status of the LAST X Register

The following operations save *x* in the LAST X register in RPN mode:

+, −, × , ÷	$\sqrt{x}$, x^2,	e^x, 10^x
LN, LOG	y^x, $\sqrt[x]{y}$	l/x, INT÷, Rmdr
SIN, COS, TAN	ASIN, ACOS, ATAN	$\hat{x}$ $\hat{y}$
SINH, COSH, TANH	ASINH, ACOSH, ATANH	IP, FP, SGN, INTG, RND, ABS
%, %CHG	Σ+, Σ–	RCL+, −, ×, ÷
	HMS→, →HMS	→DEG, →RAD
nCr nPr	!	ARG
CMPLX +, −, × ,÷	CMPLX e^x, LN, y^x, 1/x	CMPLX SIN, COS, TAN
→kg, →lb	→°C, →°F	→cm, →in
→l, →gal	→KM →MILE	

Notice that */c* does not affect the LAST X register.

The recall-arithmetic sequence X RCL + *variable* stores x in LASTx and X RCL *variable* + stores the recalled number in LASTx.

In ALG mode, the LAST X register is a companion to the stack: it holds the number that is the result of last expression. It supports using the previous expression result in ALG mode.

Accessing Stack Register Contents

The values held in the four stack registers, X, Y, Z and T, are accessible in RPN mode in an equation or program using the REGX, REGY, REGZ and REGT commands.

To use these instructions, press EQN first. Then, pressing R↓ produces a menu in the display showing the X–, Y–, Z–, T–registers. Pressing > or < will move the underline symbol, indicating which register is presently selected. Pressing ENTER will place an instruction into a program or equation that recalls the value of the chosen stack register for further use. These are displayed as REGX, REGY, REGZ, and REGT.

For example, a program line entered by first pressing EQN and then entering the instructions REGX x REGY x REGZ x REGT will compute the product of the values in the 4 stack registers and place the result into the X-register. It will leave the previous values of X, Y, and Z in the stack registers Y, Z, and T.

Many such efficient uses of values in the stack are possible in this manner that would not otherwise be available on the HP35s.

C

ALG: Summary

About ALG

This appendix summarizes some features unique to ALG mode, including,

- Two argument arithmetic
- Exponential and Logarithmic functions (↰ 10^x, ↰ LOG, ↱ e^x, ↱ LN)
- Trigonometric functions
- Parts of numbers
- Reviewing the stack
- Operations with complex numbers
- Integrating an equation
- Arithmetic in bases 2, 8, and 16
- Entering statistical two–variable data

Press MODE 4(4ALG) to set the calculator to ALG mode. When the calculator is in ALG mode, the ALG annunciator is on.

In ALG mode, operations are performed in the following order.

1. Expression in parenthesis.
2. Factorial (!) function requires inputting values before you press !.
3. Functions that require inputting values after pressing the function key, for example, COS, SIN, TAN, ACOS, ASIN, ATAN, LOG, LN, x^2, 1/x, $\sqrt{x}$, π, $\sqrt[3]{x}$, %, RND, RAND, IP, FP, INTG, SGN, nPr, nCr, %CHG, INT÷, Rmdr, ABS, e^x, 10^x, unit conversion.
4. $\sqrt[x]{y}$ and y^x.

5. Unary Minus +/-

6. ×, ÷

7. +, –

8. =

Doing Two argument Arithmetic in ALG

This discussion of arithmetic using ALG replaces the following parts that are affected by ALG mode. Two argument arithmetic operations are affected by ALG mode:

- Simple arithmetic
- Power functions (y^x, $\sqrt[x]{y}$)
- Percentage calculations (% or ↱ %CHG)
- Permutations and Combinations (↰ nCr, ↱ nPr)
- Quotient and Remainder of Division (↰ INTG 2 (2INTG÷), ↰ INTG 3 (3Rmdr))

Simple Arithmetic

Here are some examples of simple arithmetic.

In ALG mode, you enter the first number, press the operator (+, –, ×, ÷), enter the second number, and finally press the ENTER key.

To Calculate:	Press:	Display:
12 + 3	1 2 + 3 ENTER	12+3 15.0000
12 – 3	1 2 – 3 ENTER	12-3 9.0000
12 × 3	1 2 × 3 ENTER	12×3 36.0000
12 ÷ 3	1 2 ÷ 3 ENTER	12÷3 4.0000

Power Functions

In ALG mode, to calculate a number *y* raised to a power *x*, key in *y* y^x *x*, then press ENTER.

To Calculate:	Press:	Display:
12^3	1 2 y^x 3 ENTER	12^3 1,728.0000
$64^{1/3}$ (cube root)	↰ $\sqrt[x]{y}$ 3 > 6 4 ENTER	XROOT(3,64) 4.0000

Percentage Calculations

The Percent Function. The % key divides a number by 100.

To Calculate:	Press:	Display:
27% of 200	↱ % 2 0 0 > 2 7 ENTER	%(200,27) 54.0000
200 less 27%	2 0 0 − ↱ % 2 0 0 > 2 7 ENTER	200-%(200,27) 146.0000
25 plus 12%	2 5 + ↱ % 2 5 > 1 2 ENTER	25+%(25,12) 28.0000

To Calculate:	Press:
x% of *y*	↱ % *y* > *x* ENTER
Percentage change from *y* to *x*. (*y*≠ 0)	↰ %CHG *y* > *x* ENTER

Example:

Suppose that the $15.76 item cost $16.12 last year. What is the percentage change from last year's price to this year's?

Keys:	Display:	Description:
↰ %CHG 1 6 · 1 2 > 1 5 · 7 6 ENTER	 %CHG(16.12,15.7 -2.2333	This year's price dropped about 2.2% from last year's price.

Permutations and Combinations

Example: Combinations of People.

A company employing 14 women and 10 men is forming a six–person safety committee. How many different combinations of people are possible?

Keys:	Display:	Description:
↰ nCr 2 4 > 6 ENTER	nCr(24,6) 134,596.0000	Total number of combinations possible.

Quotient and Remainder Of Division

You can use ↰ INTG 2 (2INTG÷) and ↰ INTG 3 (3Rmdr) to produce either the quotient or remainder of division operations involving two integers.

↰ INTG 2 (2INTG÷) *Integer 1* > *Integer 2*. ENTER

↰ INTG 3 (3Rmdr) *Integer 1* > *Integer 2*. ENTER

Example:

To display the quotient and remainder produced by 58 ÷ 9

Keys:	Display:	Description:
↰ INTG 2 (2INTG÷)	IDIV(58,9)	Displays the quotient.
5 8 > 9 ENTER	6.0000	
↰ INTG 3 (3Rmdr)	RMDR(58,9)	Displays the remainder.
5 8 > 9 ENTER	4.0000	

Parentheses Calculations

Use parentheses when you want to postpone calculating an intermediate result until you entered more numbers. For example, suppose you want to calculate:

$$\frac{30}{85-12}\times 9$$

If you were to key in 3 0 ÷ 8 5 − 1 2 × 9, the calculator would calculate the result, -107.6471. However, that's not what you want. To delay the division until you've subtracted 12 from 85, use parentheses:

Keys:	Display:	Description:
3 0 ÷ ( ) 8 5 −	`30÷(85-)`	No calculation is done.
1 2)	`30÷(85-12)_`	Calculates 85 – 12.
× 9	`30÷(85-12)x9_`	Calculates 30/73.

ENTER	`30÷(85-12)x9`
	`3.6986`
		Calculates 30/(85 – 12) × 9.

You can omit the multiplication sign (×) before a left parenthesis. Implied multiplication is not available in Equation mode. For example, the expression 2 × (5 – 4) can be entered as 2 ( ) 5 − 4, without the × key inserted between 2 and the left parenthesis.

Exponential and Logarithmic Functions

To Calculate:	Press:	Display:

Natural logarithm (base *e*)	↱ LN 1 ENTER	`LN(1)`
		`0.0000`

Common logarithm (base 10)	↰ LOG 1 0 ENTER	`LOG(10)`
		`1.0000`

Natural exponential	↱ e^x 2 ENTER	`EXP(2)`
		`7.3891`

Common exponential (antilogarithm)	↰ 10^x 2 ENTER	`ALOG(2)`
		`100.0000`

Trigonometric Functions

Assume the unit of the angle is MODE 1 (1DEG)

To Calculate:	Press:	Display:
Sine of *x*.	SIN 3 0 ENTER	SIN(30) 0.5000
Cosine of *x*.	COS 6 0 ENTER	COS(60) 0.5000
Tangent of *x*.	TAN 4 5 ENTER	TAN(45) 1.0000
Arc sine of *x*.	↱ ASIN 1 ENTER	ASIN(1) 90.0000
Arc cosine of *x*.	↱ ACOS 0 ENTER	ACOS(0) 90.0000
Arc tangent of *x*.	↱ ATAN 0 ENTER	ATAN(0) 0.0000

Hyperbolic functions

To Calculate:	Press:
Hyperbolic sine of *x* (SINH).	↰ HYP SIN,key in a number, press ENTER
Hyperbolic cosine of *x* (COSH).	↰ HYP COS, key in a number, press ENTER
Hyperbolic tangent of *x* (TANH).	↰ HYP TAN, key in a number, press ENTER
Hyperbolic arc sine of *x* (ASINH).	↰ HYP ↱ ASIN, key in a number, press ENTER
Hyperbolic arc cosine of *x* (ACOSH).	↰ HYP ↱ ACOS, key in a number, press ENTER
Hyperbolic arc tangent of *x* (ATANH).	↰ HYP ↱ ATAN, key in a number, press ENTER

Parts of numbers

To calculate:	Press:	Display:
The integer part of 2.47	↰ INTG 6 (6IP) 2 · 4 7 ENTER	IP(2.47) 2.0000
The fractional part of 2.47	↰ INTG 5 (5FP) 2 · 4 7 ENTER	FP(2.47) 0.4700
The absolute value of –7	↱ ABS +/– 7 ENTER	ABS(-7) 7.0000
The sign value of 9	↰ INTG 1 (1SGN) 9 ENTER	SGN(9) 1.0000
The greatest integer equal to or less than –5.3	↰ INTG 4 (4INTG) +/– 5 · 3 ENTER	INTG(-5.3) -6.0000

Reviewing the Stack

The R↓ or ↱ R↑ key produces a menu in the display— X–, Y–, Z–, T–registers, to let you review the entire contents of the stack. The difference between the R↓ and the ↱ R↑ key is the location of the underline in the display. Pressing the ↱ R↑ displays the underline on the T register; pressing the R↓ displays the underline on the Y register.

Pressing R↓ displays the following menu:

X Y Z T

value

Pressing ↱ R↑ displays the following menu:

X Y Z T

value —

You can press R↓ and ↱ R↑ (along with > or <) to review the entire contents of the stack and recall them. They will appear as REGX, REGY, REGZ or REGT depending on which part of the stacked was recalled and may be used in further calculations.

The value of X-, Y-, Z-, T-register in ALG mode is the same in RPN mode. After normal calculation, solving, programming, or integrating, the value of the four registers will be the same as in RPN or ALG mode and retained when you switch between ALG and RPN logic modes.

Integrating an Equation

1. Key in an equation. (see "Entering Equations into the Equation List" in chapter 6) and leave Equation mode.
2. Enter the limits of integration: key in the *lower* limit and press x↔y, then key in the upper limit.
3. Display the equation: Press EQN and, if necessary, scroll through the equation list (press ^ or v) to display the desired equation.
4. Select the variable of integration: Press ↰ ∫ *variable.* This starts the calculation.

Operations with Complex Numbers

To enter a complex number:

Form: xiy

1. Type the real part.
2. Press i.
3. Type the imaginary part.

Form: x+yi

1. Type the real part.
2. Press +.
3. Type the imaginary part.
4. Press i.

Form: rθa

1. Type the value of r.
2. Press ↱ θ.
3. Type the *value of θ*.

To do an operation with one complex number:

1. Select the function.
2. Enter the complex number *z*.
3. Press ENTER to calculate.
4. The calculated result will be displayed in Line 2 and the displayed form will be the one that you have set in MODE.

To do an arithmetic operation with two complex numbers:

1. Enter the first complex number, z_1.
2. Select the arithmetic operation.
3. Enter the second complex number, z_2.
4. Press ENTER to calculate.
5. The calculated result will be displayed in Line 2 and the displayed form will be the one that you have set in MODE.

Here are some examples with complex numbers:

Examples:

Evaluate sin (2 + 3*i*)

Keys:	Display:	Description:
↰ DISPLAY 9 (9×·i·y)		Sets display form
SIN 2 + 3 i	SIN(2+3i)	
ENTER	SIN(2+3i) 9.1545i-4.1689	Result is 9.1545 *i*–4.1689

Examples:

Evaluate the expression

$$z_1 \div (z_2 + z_3),$$

where $z_1 = 23 + 13\,i$, $z_2 = -2 + i$ $z_3 = 4 - 3\,i$

Keys:	Display:	Description:
↰ DISPLAY · 1 (10x+y·i)		Sets display form
( ) 2 3 + 1 3 i ▷ ÷ ( ) +/- 2 + i + 4 − 3 i	⬅i÷(-2+i+4-3i)	
ENTER	(23+13i)÷(-2+... 2.5000+9.0000i	Result is 2.5000 + 9.0000 *i*

Examples:

Evaluate (4 - 2/5 *i*) × (3 - 2/3 *i*)

Keys:	Display:	Description:
( ) 4 − · 2 · 5 i ▷ × ( ) 3 − · 2 · 3 i	⬅5i)×(3-0 2/3i)	
ENTER	(4-0 2/5i)×(3... 11.7333i-3.8667	Result is 11.7333 *i*–3.8667

Arithmetic in Bases 2, 8, and 16

Here are some examples of arithmetic in Hexadecimal, Octal, and Binary modes:

Example:

$$12F_{16} + E9A_{16} = ?$$

Keys:	Display:	Description:
↱ BASE 2 (2HEX)		Sets base 16; **HEX** annunciator on.

1 2 RCL F ↱ BASE 6(6h) + RCL E 9 RCL A ↱ BASE 6(6h) ENTER	12Fh+E9Ah FC9h	Result.

$7760_8 - 4326_8$=?

↱ BASE 3(3OCT)	12Fh+E9Ah 7711o	Sets base 8: OCT annunciator on.
7 7 6 0 ↱ BASE 7(7o) − 4 3 2 6 ↱ BASE 7(7o) ENTER	7760o−4326o 3432o	Converts displayed number to octal.

$100_8 \div 5_8$=?

1 0 0 ↱ BASE 7 (7o) ÷ 5 ↱ BASE 7(7o) ENTER	100o÷5o 14o	Integer part of result.

$5A0_{16} + 10011000_2$ =?

↱ BASE 2(2HEX) 5 RCL A 0 ↱ BASE 6(6h) +	5A0h+	Set base 16; **HEX** annunciator on.
1 0 0 1 1 0 0	⬅A0h+10011000b	
0 ↱ BASE 8(8b) ENTER	5A0h+10011000b 638h	Result in hexadecimal base.
↱ BASE 1(1DEC)	5A0h+b10011000b 1,592.0000	Restores decimal base.

Entering Statistical Two–Variable Data

In ALG mode, remember to enter an (*x*, *y*) pair in *reverse order* (*y* x↔y *x or y* ENTER *x*) so that *y* ends up in the Y–register and X in the X–register.

1. Press ↱ CLEAR 4 (4Σ) to clear existing statistical data.

2. Key in the *y*–value *first* and press x↔y.

3. Key in the corresponding *x*–value and press Σ+.

4. The display shows *n* the number of statistical data pairs you have accumulated.
5. Continue entering *x*, *y*–pairs. *n* is updated with each entry.

If you wish to delete the incorrect values that were just entered, press ↰ Σ–. After deleting the incorrect statistical data, the calculator will display the last statistical data entered in line 1 (top line of the display) and value of n in line 2. If there are no statistical data, the calculator will display n=0 in line 2.

Example:

After keying in the *x*, *y*–values on the left, make the corrections shown on the right:

Initial x, y	Corrected x, y
20, 4 400, 6	20, 5 40, 6

Keys:	Display:	Description:
↱ CLEAR 4 (4Σ)		Clears existing statistical data.
4 x↔y 2 0 Σ+	20 Σ+ 1.0000	Enters the first new data pair.
6 x↔y 4 0 0 Σ+	400 Σ+ 2.0000	Display shows *n*, the number of data pairs you entered.
↱ LAST*x*	LASTx 400.0000	Brings back last *x*–value. Last *y* is still in Y–register.
↰ Σ–	400 Σ– 1.0000	Deletes the last data pair.
6 x↔y 4 0 Σ+	40 Σ+ 2.0000	Reenters the last data pair.
4 x↔y 2 0 ↰ Σ–	20 Σ– 1.0000	Deletes the first data pair.
5 x↔y 2 0 Σ+	20 Σ+ 2.0000	Reenters the first data pair. There is still a total of two data pairs in the statistics registers.

Linear Regression

Linear regression, or L.R. (also called *linear estimation)*, is a statistical method for finding a straight line that best fits a set of *x*,*y*–data.

- To find an estimated value for *x* (or *y*), key in a given hypothetical value for *y* (or *x*) ,press ENTER, then press ↰ L.R. ($\hat{x}$) (or ↰ L.R. > ($\hat{y}$)).
- To find the values that define the line that best fits your data, press ↰ L.R. followed by (r), (m), or (b).

D

More about Solving

This appendix provides information about the SOLVE operation beyond that given in chapter 7.

How SOLVE Finds a Root

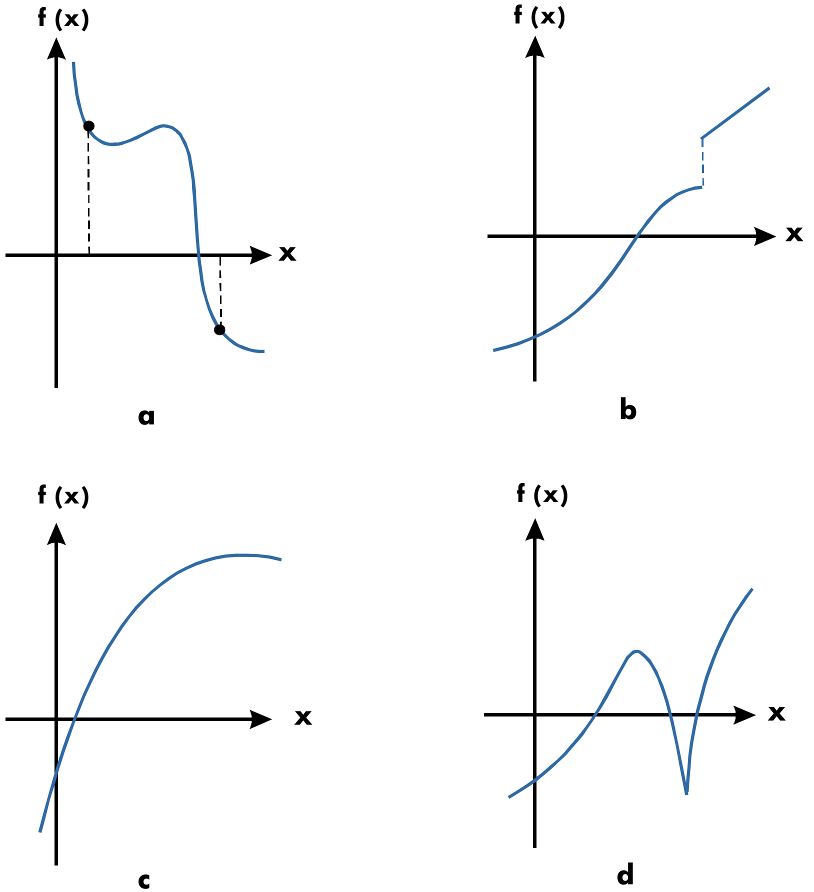
SOLVE first attempts to solve the equation directly for the unknown variable. If the attempt fails, SOLVE changes to an iterative(repetitive) procedure. The *iterative* operation is to execute repetitively the specified equation. The value returned by the equation is a function *f(x)* of the unknown variable *x*. (*f(x)* is mathematical shorthand for a function defined in terms of the unknown variable *x*.) SOLVE starts with an estimate for the unknown variable, *x*, and refines that estimate with each successive execution of the function, *f(x)*.

If any two successive estimates of the function *f(x)* have opposite signs, then SOLVE presumes that the function *f(x)* crosses the *x*–axis in at least one place between the two estimates. This interval is systematically narrowed until a root is found.

For SOLVE to find a root, the root has to exist within the range of numbers of the calculator, and the function must be mathematically defined where the iterative search occurs. SOLVE always finds a root, provided one exists (within the overflow bounds), if one or more of these conditions are met:

- Two estimates yield *f(x)* values with opposite signs, and the function's graph crosses the *x*–axis in at least one place between those estimates (figure a, below).
- *f(x)* always increases or always decreases as *x* increases (figure b, below).
- The graph of *f(x)* is either concave everywhere or convex everywhere (figure c, below).

- If *f(x)* has one or more local minima or minima, each occurs singly between adjacent roots of *f(x)* (figure d, below).



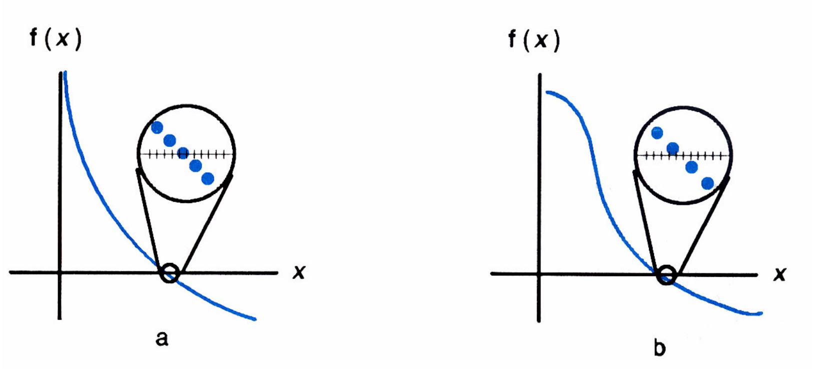
Function Whose Roots Can Be Found

In most situations, the calculated root is an accurate estimate of the theoretical, infinitely precise root of the equation. An "ideal" solution is one for which *f(x)* = 0. However, a very small non–zero value for *f(x)* is often acceptable because it might result from approximating numbers with limited (12–digit) precision.

Interpreting Results

The SOLVE operation will produce a solution under either of the following conditions:

- If it finds an estimate for which *f(x)* equals zero. (See figure a, below.)
- If it finds an estimate where *f(x)* is not equal to zero, but the calculated root is a 12–digit number adjacent to the place where the function's graph crosses the *x*–axis (see figure b, below). This occurs when the two final estimates are neighbors (that is, they differ by 1 in the 12th digit), and the function's value is positive for one estimate and negative for the other. Or they are $(0, 10^{-499})$ or $(0, -10^{-499})$. In *most cases*, *f(x)* will be relatively close to zero.



✓ To obtain additional information about the result, press R↓ see the previous estimate of the root (*x*), which was left in the Y–register. Press R↓ again to see the value of *f(x)*, which was left in the Z–register. If *f(x)* equals zero or is relatively small, it is very likely that a solution has been found. However, if *f(x)* is relatively large, you must use caution in interpreting the results.

Example: An Equation With One Root.

Find the root of the equation:

$$-2x^3 + 4x^2 - 6x + 8 = 0$$

Enter the equation as an expression:

Keys:	Display:	Description:
EQN		Select Equation mode.
+/- 2 ×		Enters the equation.
RCL X y^x 3		
+ 4 ×		
RCL X y^x 2		
− 6 × RCL X		
+ 8 ENTER	-2xX^3+4xX^2-6➡	

↰ SHOW	CK=B9AD
	LN=18
		Checksum and length.

C		Cancels Equation mode.

Now, solve the equation to find the root:

Keys:	Display:	Description:
0 ↱ STO X	10_	Initial guesses for the root.
ENTER 1 0		
EQN	-2xX^3+4xX^2-6➡	Selects Equation mode; displays the left end of the equation.
↱ SOLVE X	SOLVING X= 1.6506	Solves for *X*; displays the result.
✓ R↓	1.6506	Final two estimates are the same to four decimal places.
✓ R↓	-4.0000E-11	*f(x)* is *very* small, so the approximation is a good root.

Example: An Equation with Two Roots.

Find the two roots of the parabolic equation:

$$x^2 + x - 6 = 0.$$

Enter the equation as an expression:

Keys:	Display:	Description:
EQN		Selects Equation mode.
RCL X y^x 2 +		Enters the equation.
RCL X − 6	`X^2+X-6`	
ENTER		

↰ SHOW	`CK=3971`
	`LN=7`
		Checksum and length.

C		Cancels Equation mode.

Now, solve the equation to find its positive and negative roots:

Keys:	Display:	Description:
0 ↱ STO X ENTER 1 0	`10_`	Your initial guesses for the positive root.
EQN	`X^2+X-6`	Selects Equation mode; displays the equation.

↱ SOLVE X	`SOLVING`
	`X=`
	`2.0000`
		Calculates the positive root using guesses 0 and 10.

✓ R↓	`2.0000`	Final two estimates are the same.
✓ R↓ ↰ SHOW	`0.00000000000`	*f(x)* = 0.
0 ↱ STO X ENTER 1 0 +/−	`-10_`	Your initial guesses for the negative root.
EQN	`X^2+X-6`	Redisplays the equation.

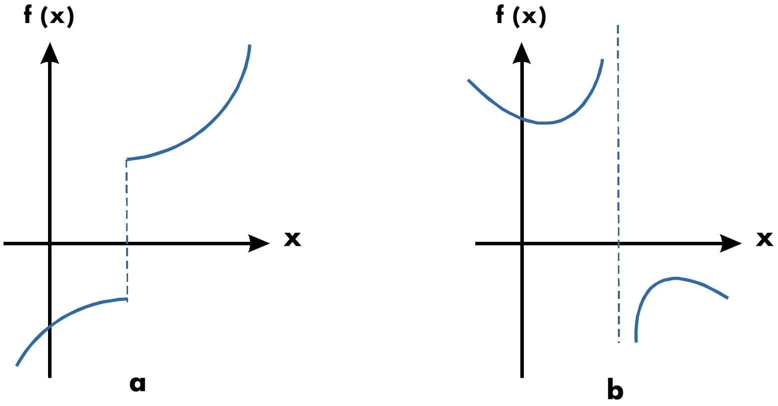
↱ SOLVE X	`SOLVING`
	`X=`
	`-3.0000`
		Calculates negative root using guesses 0 and –10.

✓ R↓ R↓ ↰ SHOW	`0.00000000000`	*f(x)* = 0.

Certain cases require special consideration:

- If the function's graph has a discontinuity that crosses the *x*–axis, then the SOLVE operation returns a value adjacent to the discontinuity (see figure a, below). In this case, *f(x)* may be relatively large.

- Values of *f(x)* may be approaching infinity at the location where the graph changes sign (see figure b, below). This situation is called a *pole*. Since the SOLVE operation determines that there is a sign change between two neighboring values of *x*, it returns the possible root. However, the value for *f(x)* will be relatively large. If the pole occurs at a value of *x* that is exactly represented with 12 digits, then that value would cause the calculation to halt with an error message.



Special Case: A Discontinuity and a Pole

Example: A Discontinuous Function.

Find the root of the equation:

$$IP(x) = 1.5$$

Enter the equation:

Keys:	Display:	Description:
EQN		Selects Equation mode.
⬑ INTG 6 (6IP)		Enter the equation.
RCL X > ⬑ =		
1 · 5 ENTER	IP(X)=1.5	

⬑ SHOW	CK=D2C1
	LN=9
		Checksum and length.

C		Cancels Equation mode.

Now, solve to find the root:

	Keys:	Display:	Description:
	0 ↱ STO X ENTER 5	`5_`	Your initial guesses for the root.
	EQN	`IP(X)=1.5`	Selects Equation mode; displays the equation.

	↱ SOLVE X	`SOLVING`
		`X=`
		`2.0000`
			Finds a root with guesses 0 and 5.

	↰ SHOW	`1.99999999999`	Shows root, to 11 decimal places.
✓	R↓ ↰ SHOW	`2.00000000000`	The previous estimate is slightly bigger.
✓	R↓	`-0.5000`	*f(x)* is relatively large.

Note the difference between the last two estimates, as well as the relatively large value for *f(x)*. The problem is that there is no value of *x* for which *f(x)* equals zero. However, at *x* = 1.99999999999, there is a neighboring value of *x* that yields an opposite sign for *f(x)*.

Example:

Find the root of the equation

$$\frac{x}{x^2-6}-1=0$$

As *x* approaches $\sqrt{6}$, *f(x)* becomes a very large positive or negative number.

Enter the equation as an expression.

Keys:	Display:	Description:
EQN		Selects Equation mode.
RCL X ÷ ( ) RCL X y^x 2 − 6 > − 1 ENTER		Enters the equation.
	`X÷(X^2-6)-1`	

↰ SHOW	CK=7358
	LN=11
		Checksum and length.

C		Cancels Equation mode.

Now, solve to find the root.

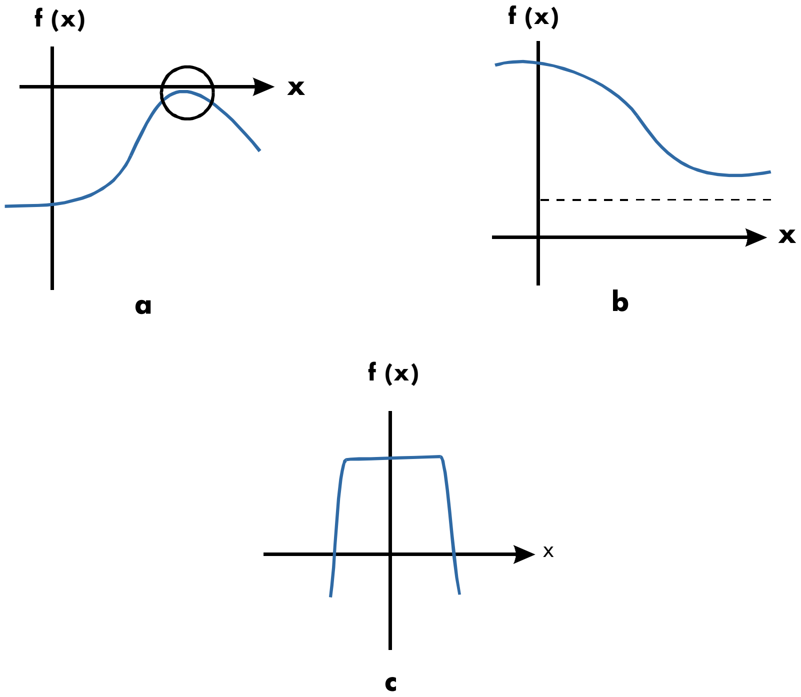
Keys:	Display:	Description:
2 · 3 ↱		Your initial guesses for the root.
STO X ENTER 2 · 7	2.7_	
EQN	X÷(X^2-6)-1	Selects Equation mode; displays the equation.
↱ SOLVE X	NO ROOT FND	No root found for *f(x)*.

When SOLVE Cannot Find a Root

Sometimes SOLVE fails to find a root. The following conditions cause the message NO ROOT FND:

- The search terminates near a local minimum or maximum (see figure a, below).
- The search halts because SOLVE is working on a horizontal asymptote—an area where *f(x)* is essentially constant for a wide range of *x* (see figure b, below).
- The search is concentrated in a local "flat" region of the function (see figure c, below).

In these cases, the values in the stack will be same as the values before executing SOLVE.



Case Where No Root Is Found

Example: A Relative Minimum.

Calculate the root of this parabolic equation:

$$x^2 - 6x + 13 = 0.$$

It has a minimum at $x = 3$.

Enter the equation as an expression:

Keys:	Display:	Description:
EQN		Selects Equation mode.
RCL X y^x 2		Enters the equation.
− 6 × RCL X		
+ 1 3 ENTER	X^2-6xX+13	

↰ SHOW	CK=EC74
	LN=10
		Checksum and length.

C		Cancels Equation mode.

Now, solve to find the root:

Keys:	Display:	Description:
0 ↱ STO X		Your initial guesses for the root.
ENTER 1 0	10_	
EQN	X^2-6xX+13	Selects Equation mode; displays the equation.
↱ SOLVE X	NO ROOT FND	Search fails with guesses 0 and 10

Example: **An Asymptote.**

Find the root of the equation

$$10-\frac{1}{X}=0$$

Enter the equation as an expression.

Keys:	Display:	Description:
EQN		Selects Equation mode.
1 0 − 1/x		Enters the equation.
RCL X ENTER	10-INV(X)	

↰ SHOW	CK=6EAB
	LN=9
		Checksum and length.

C		Cancels Equation mode.
· 0 0 5 ↱		Your positive guesses for the root.
STO X ENTER 5	5_	
EQN	10-INV(X)	Selects Equation mode; displays the equation.
↱ SOLVE X	X= 0.1000	Solves for *x* using guesses 0.005 and 5.
✓ R↓	0.1000	Previous estimate is the same.
✓ R↓ ↰ SHOW	0.00000000000	*f* (x) = 0

Watch what happens when you use negative values for guesses:

Keys:	Display:	Description:
+/- 1 ↱ STO X ENTER	-1.0000	Your negative guesses for the root.
+/- 2 EQN	10-INV(X)	Selects Equation mode; displays the equation.
↱ SOLVE X	X= 0.1000	Solves for *X*; displays the result.

Example: Find the root of the equation.

$$\sqrt{[x \div (x+0.3)]} - 0.5 = 0$$

Enter the equation as an expression:

Keys:	Display:	Description:
EQN		Selects Equation mode.
√x RCL X ÷ ( ) RCL X + · 3 > > − · 5		Enters the equation.
ENTER	SQRT(X÷(X+0.3))➡	

↰ SHOW	CK=9F3B
	LN=19
		Checksum and length.

C		Cancels Equation mode.

First attempt to find a positive root:

Keys:	Display:	Description:
0 ↱ STO X		Your positive guesses for the root.
ENTER 1 0	10_	
EQN	SQRT(X÷(X+0.3))➡	Selects Equation mode; displays the left end of the equation.
↱ SOLVE X	X= 0.1000	Calculates the root using guesses 0 and 10.

Now attempt to find a negative root by entering guesses 0 and –10. Notice that the function is undefined for values of *x* between 0 and –0.3 since those values produce a positive denominator but a negative numerator, causing a negative square root.

Keys:	Display:	Description:
0 ↱ STO X		
ENTER +/– 1 0	`-10_`	
EQN	`SQRT(X÷(X+0.3))`➡	Selects Equation mode; displays the left end of the equation.
↱ SOLVE X	`NO ROOT FND`	No root found for *f(x)*.

Example: A Local "Flat" Region.

Find the root of the function

f(x) = *x* + 2 if *x* < –1,

f(x) = 1 for –1 ≤ *x* ≤ 1 (a local flat region),

f(x) = –*x* + 2 if *x* >1.

In RPN mode, enter the function as the program:

```
J001 LBL J
J002 1
J003 2
J004 RCL+ X
J005 x<y?
J006 RTN
J007 4
J008 -
J009 +/-
J010 x>y?
J011 R↓
J012 RTN
```

Checksum and length: 9412 39

Solve for X using initial guesses of 10^{-8} and -10^{-8}.

Keys: (In RPN mode)	Display:	Description:
1 E 8		Enters guesses.
+/- ⮫ STO X 1	-1E-8_	
+/- E 8 +/-		
⮪ FN= J	-1.0000E-8	Selects program "J" as the function.
⮫ SOLVE X	X= -2.0000	Solves for *X*; displays the result.

Round–Off Error

The limited (12–digit) precision of the calculator can cause errors due to rounding off, which adversely affect the iterative solutions of SOLVE and integration. For example,

$$[(|x|+1)+10^{15}]^2 - 10^{30} = 0$$

has no roots because *f(x)* is always greater than zero. However, given initial guesses of 1 and 2, SOLVE returns the answer 1.0000 due to round–off error.

Round–off error can also cause SOLVE to fail to find a root. The equation

$$|x^2 - 7| = 0$$

has a root at $\sqrt{7}$. However, no 12–digit number *exactly* equals $\sqrt{7}$, so the calculator can never make the function equal to zero. Furthermore, the function never changes sign SOLVE returns the message NO ROOT FND.

E

More about Integration

This appendix provides information about integration beyond that given in chapter 8.

How the Integral Is Evaluated

The algorithm used by the integration operation, ∫FN dx, calculates the integral of a function *f(x)* by computing a weighted average of the function's values at many values of *x* (known as sample points) within the interval of integration. The accuracy of the result of any such sampling process depends on the number of sample points considered: generally, the more sample points, the greater the accuracy. If *f(x)* could be evaluated at an infinite number of sample points, the algorithm could — neglecting the limitation imposed by the inaccuracy in the calculated function *f(x)* — always provide an exact answer.

Evaluating the function at an infinite number of sample points would take forever. However, this is not necessary since the maximum accuracy of the calculated integral is limited by the accuracy of the calculated function values. Using only a finite number of sample points, the algorithm can calculate an integral that is as accurate as is justified considering the inherent uncertainty in *f(x)*.

The integration algorithm at first considers only a few sample points, yielding relatively inaccurate approximations. If these approximations are not yet as accurate as the accuracy of *f(x)* would permit, the algorithm is iterated (repeated) with a larger number of sample points. These iterations continue, using about twice as many sample points each time, until the resulting approximation is as accurate as is justified considering the inherent uncertainty in *f(x)*.

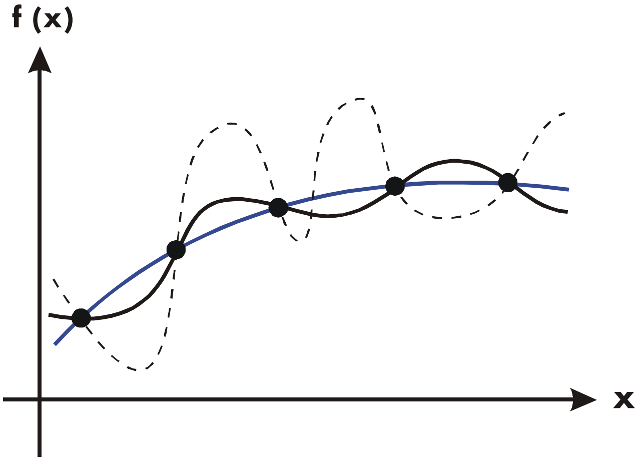
As explained in chapter 8, the uncertainty of the final approximation is a number derived from the display format, which specifies the uncertainty for the function. At the end of each iteration, the algorithm compares the approximation calculated during that iteration with the approximations calculated during two previous iterations. If the difference between any of these three approximations and the other two is less than the uncertainty tolerable in the final approximation, the calculation ends, leaving the current approximation in the X–register and its uncertainty in the Y–register.

It is extremely unlikely that the errors in each of three successive approximations — that is, the differences between the actual integral and the approximations — would all be larger than the disparity among the approximations themselves. Consequently, the error in the final approximation will be less than its uncertainty (provided that *f(x)* does not vary rapidly). Although we can't know the error in the final approximation, the error is extremely unlikely to exceed the displayed uncertainty of the approximation. In other words, the uncertainty estimate in the Y–register is an almost certain "upper bound" on the difference between the approximation and the actual integral.

Conditions That Could Cause Incorrect Results

Although the integration algorithm in the HP 35s is one of the best available, in certain situations it — like all other algorithms for numerical integration — might give you an incorrect answer. *The possibility of this occurring is extremely remote*. The algorithm has been designed to give accurate results with almost any *smooth* function. Only for functions that exhibit *extremely* erratic behavior is there any substantial risk of obtaining an inaccurate answer. Such functions rarely occur in problems related to actual physical situations; when they do, they usually can be recognized and dealt with in a straightforward manner.

Unfortunately, since all that the algorithm knows about *f(x)* are its values at the sample points, it cannot distinguish between *f(x)* and any other function that agrees with *f(x)* at all the sample points. This situation is depicted below, showing (over a portion of the interval of integration) three functions *whose* graphs include the many sample points in common.



With this number of sample points, the algorithm will calculate the same approximation for the integral of any of the functions shown. The actual integrals of the functions shown with solid blue and black lines are about the same, so the approximation will be fairly accurate if *f(x)* is one of these functions. However, the actual integral of the function shown with a dashed line is quite different from those of the others, so the current approximation will be rather inaccurate if *f(x)* is this function.

The algorithm comes to know the general behavior of the function by sampling the function at more and more points. If a fluctuation of the function in one region is not unlike the behavior over the rest of the interval of integration, at some iteration the algorithm will likely detect the fluctuation. When this happens, the number of sample points is increased until successive iterations yield approximations that take into account the presence of the most rapid, *but characteristic,* fluctuations.

For example, consider the approximation of

$$\int_0^\infty xe^{-x}dx.$$

Since you're evaluating this integral numerically, you might think that you should represent the upper limit of integration as 10^{499}, which is virtually the largest number you can key into the calculator.

Try it and see what happens. Enter the function *f(x)* = xe^{-x}.

Keys:	Display:	Description:
EQN		Select equation mode.
RCL X × ↱ e^x	XxEXP()	Enter the equation.
+/- RCL X ENTER	XxEXP(-X)	End of the equation.

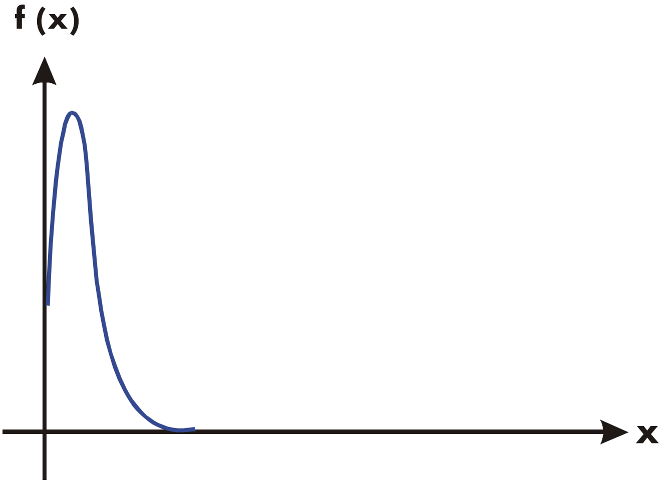
↰ SHOW	CK=2FE6
	LN=9
		Checksum and length.

C		Cancels Equation mode.

Set the display format to SCI 3, specify the lower and upper limits of integration as zero and 10^{499}, than start the integration.

Keys:	Display:	Description:
✓ ↰ DISPLAY 2 (2SCI)		Specifies accuracy level and limits of integration.
3 ENTER 1 E 4 9 9	1E499_	
EQN	XxEXP(-X)	Selects Equation mode; displays the equation.
↰ ∫ X	INTEGRATING ∫ = 0.000E0	Approximation of the integral.

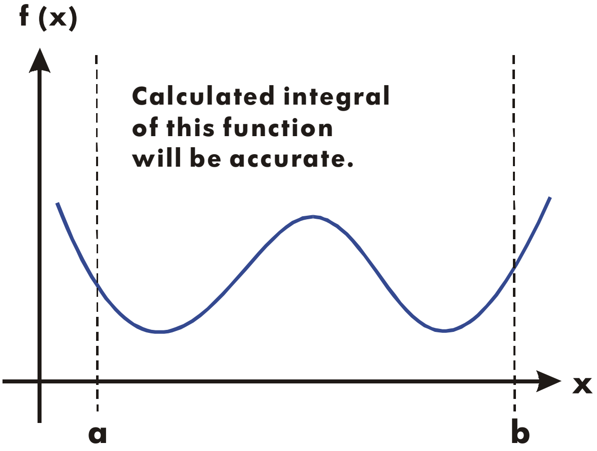
The answer returned by the calculator is clearly incorrect, since the actual integral of *f(x)* = xe^{-x} from zero to ∞ is exactly 1. But the problem is *not* that ∞ was represented by 10^{499}, since the actual integral of this function from zero to 10^{499} is very close to 1. The reason for the incorrect answer becomes apparent from the graph of *f(x)* over the interval of integration.

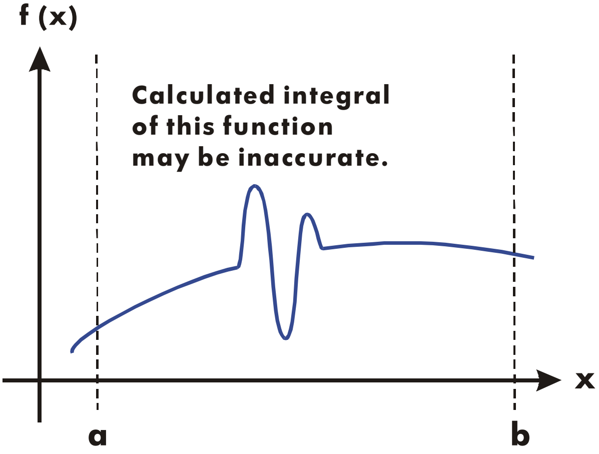


The graph is a spike very close to the origin. Because no sample point happened to discover the spike, the algorithm assumed that *f(x)* was identically equal to zero throughout the interval of integration. Even if you increased the number of sample points by calculating the integral in SCI 11 or ALL format, none of the additional sample points would discover the spike when this particular function is integrated over this particular interval. (For better approaches to problems such as this, see the next topic, "Conditions That Prolong Calculation Time.")

Fortunately, functions exhibiting such aberrations (a fluctuation that is uncharacteristic of the behavior of the function elsewhere) are unusual enough that you are unlikely to have to integrate one unknowingly. A function that could lead to incorrect results can be identified in simple terms by how rapidly it and its low–order derivatives vary across the interval of integration. Basically, the more rapid the variation in the function or its derivatives, and the lower the order of such rapidly varying derivatives, the less quickly will the calculation finish, and the less reliable will be the resulting approximation.

Note that the rapidity of variation in the function (or its low–order derivatives) must be determined with respect to the width of the interval of integration. With a given number of sample points, a function *f(x)* that has three fluctuations can be better characterized by its samples when these variations are spread out over most of the interval of integration than if they are confined to only a small fraction of the interval. (These two situations are shown in the following two illustrations.) Considering the variations or fluctuation as a type of oscillation in the function, the criterion of interest is the ratio of the period of the oscillations to the width of the interval of integration: the larger this ratio, the more quickly the calculation will finish, and the more reliable will be the resulting approximation.





In many cases you will be familiar enough with the function you want to integrate that you will know whether the function has any quick wiggles relative to the interval of integration. If you're not familiar with the function, and you suspect that it may cause problems, you can quickly plot a few points by evaluating the function using the equation or program you wrote for that purpose.

If, for any reason, after obtaining an approximation to an integral, you suspect its validity, there's a simple procedure to verify it: subdivide the interval of integration into two or more adjacent subintervals, integrate the function over each subinterval, then add the resulting approximations. This causes the function to be sampled at a brand new set of sample points, thereby more likely revealing any previously hidden spikes. If the initial approximation was valid, it will equal the sum of the approximations over the subintervals.

Conditions That Prolong Calculation Time

In the preceding example, the algorithm gave an incorrect answer because it never detected the spike in the function. This happened because the variation in the function was too quick relative to the width of the interval of integration. If the width of the interval were smaller, you would get the correct answer; but it would take a very long time if the interval were still too wide.

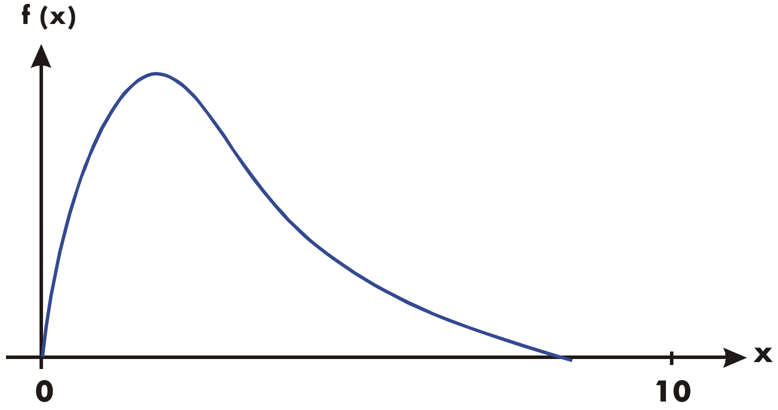
Consider an integral where the interval of integration is wide enough to require excessive calculation time, but not so wide that it would be calculated incorrectly. Note that because $f(x) = xe^{-x}$ approaches zero very quickly as x approaches ∞, the contribution to the integral of the function at large values of x is negligible. Therefore, you can evaluate the integral by replacing ∞, the upper limit of integration, by a number not so large as 10^{499} — say 10^{3}.

Rerun the previous integration problem with this new limit of integration:

Keys:	Display:	Description:
0 ENTER 1 E 3	`1E3_`	New upper limit.
EQN	`X×EXP(-X)`	Selects Equation mode; displays the equation.

↰ ∫ X	INTEGRATING ∫ = 1.000E0	Integral. (The calculation takes a minute or two.)
x◂▸y	1.000E-3	Uncertainty of approximation.

This is the correct answer, but it took a very long time. To understand why, compare the graph of the function between *x* = 0 and *x* = 10^3, which looks about the same as that shown in the previous example, with the graph of the function between *x* = 0 and *x* = 10:



You can see that this function is "interesting" only at small values of *x*. At greater values of *x*, the function is not interesting, since it decreases smoothly and gradually in a predictable manner.

The algorithm samples the function with higher densities of sample points until the disparity between successive approximations becomes sufficiently small. For a narrow interval in an area where the function is interesting, it takes less time to reach this critical density.

To achieve the same density of sample points, the total number of sample points required over the larger interval is much greater than the number required over the smaller interval. Consequently, several more iterations are required over the larger interval to achieve an approximation with the same accuracy, and therefore calculating the integral requires considerably more time.

Because the calculation time depends on how soon a certain density of sample points is achieved in the region where the function is interesting, the calculation of the integral of any function will be prolonged if the interval of integration includes mostly regions where the function is not interesting. Fortunately, if you must calculate such an integral, you can modify the problem so that the calculation time is considerably reduced. Two such techniques are subdividing the interval of integration and transformation of variables. These methods enable you to change the function or the limits of integration so that the integrand is better behaved over the interval(s) of integration.

F

Messages

The calculator responds to certain conditions or keystrokes by displaying a message. The ⚠ symbol comes on to call your attention to the message. For significant conditions, the message remains until you clear it. Pressing C or ← clears the message and the previous display content will be shown. Pressing any other key clears the message but the function of the key will not be executed.

∫FN ACTIVE	A running program attempted to select a program label (FN=*label*) while an integration calculation was running.
∫(∫FN)	A running program attempted to integrate a program (∫FN d *variable*) while another integration calculation was running.
∫(SOLVE)	A running program attempted to solve a program while an integration calculation was running.
ALL VARS=0	The catalog of variables (↰ MEM 1 (1VAR)) indicates no values stored.
BAD GUESS	You set a wrong guess number (like a complex number or vector) when SOLVING equation for a variable.
CALCULATING	The calculator is executing a function that might take a while.
CLR ALL? Y N	Allow you to verify clearing everything in memory.
CLR EQN? Y N	Allows you to verily clearing the equation you are editing. (Occurs only in Equation–entry mode.)
CLR PGMS? Y N	Allows you to verify clearing *all programs* in memory. (Occurs only in Program–entry mode.)
DIVIDE BY 0	Attempted to divide by zero. (Includes ↰ %CHG if Y–register contains zero.)
DUPLICAT.LBL	Attempted to enter a program label that already exists for another program routine.

EQN LIST TOP
Indicates the "top" of equation memory. The memory scheme is circular, so EQN LIST TOP is also the "equation" after the last equation in equation memory.

INTEGRATING
The calculator is calculating the integral of an equation or program. *This might take a while*.

INTERRUPTED
A running CALCULATE,SOLVE or ∫ FN operation was interrupted by pressing C or R/S in ALG, RPN, EQN, or PGM mode.

INVALID DATA
Data error:

- Attempted to save or calculate error data.
- Attempted to calculate combinations or permutations with $r > n$, with non–integer r or n, or with $n \geq 10^{16}$.
- Attempted to save a complex number or vector in the statistical data.
- Attempted to save a base-n number that contains digits greater than the largest base-n number digit allowed.
- Attempted to save an invalid data in the statistical register using $x \leftrightarrow y$ operation.
- Attempt to compare complex numbers or vectors.
- Attempted to use a trigonometric or hyperbolic function with an illegal argument:
  - TAN with x an odd multiple of 90°.
  - ↱ ACOS or ↱ ASIN with $x < -1$ or $x > 1$.
  - ↰ HYP ↱ ATAN with $x \leq -1$; or $x \geq 1$.
  - ↰ HYP ↱ ACOS with $x < 1$.

INVALID VAR
Attempted to enter an invalid variable name when solving an equation.

INVALID x!
Attempted a factorial or gamma operation with x as a negative integer.

`INVALID y`^{x}

Exponentiation error:

- Attempted to raise 0 to the 0th power or to a negative power.
- Attempted to raise a negative number to a non–integer power.
- Attempted to raise complex number (0 + *i* 0) to a number with a negative real part.

`INVALID (I)`

Attempted an operation with an invalid indirect value ((I) is not defined).

`INVALID (J)`

Attempted an operation with an invalid indirect value ((J) is not defined).

`LOG(0)`

Attempted to take a logarithm of zero or (0 + *i*0).

`LOG(NEG)`

Attempted to take a logarithm of a negative number.

`MEMORY CLEAR`

All of user memory has been erased (see page).

`MEMORY FULL`

The calculator has insufficient memory available to do the operation (See appendix B).

`NO`

The condition checked by a test instruction is not true. (Occurs only when executed from the keyboard.)

`NONEXISTENT`

Attempted to refer to a nonexistent program label (or line number) with GTO, XEQ, or `FN`. Note that the error `NONEXISTENT` can mean

- you explicitly (from the keyboard) called a program label that does not exist; or
- the program that you called referred to *another* label, which does not exist.

The result of integration does not exist.

`NO LABELS`

The catalog of programs (↰ MEM 2 (`2PGM`)) indicates no program labels stored.

`NO SOLUTION`

No solution could be found for this system of linear equations.

`MULT SOLUTION`

Multiple solutions have been found for this system of linear equations.

NO ROOT FND

SOLVE (include EQN and PGM mode)cannot find the root of the equation using the current initial guesses (see page). These conditions include: bad guess, solution not found, point of interest, left unequal to right. A SOLVE operation executed in a program does not produce this error; the same condition causes it instead to skip the next program line (the line following the instruction SOLVE *variable*).

OVERFLOW

Warning (displayed momentarily); the magnitude of a result is too large for the calculator to handle. The calculator returns ±9.99999999999E499 in the current display format. (See "Range of Numbers and Overflow" on page .) This condition sets flag 6. If flag 5 is set, overflow has the added effect of halting a running program and leaving the message in the display until you press a key.

PRGM TOP

Indicates the "top" of program memory. The memory scheme is circular, so PRGM TOP is also the "line" after the last line in program memory.

RUNNING B

The calculator is running an equation or program (other than a SOLVE or ∫FN routine).

SELECT FN

Attempted to execute SOLVE *variable* or ∫FN d *variable* without a selected program label. This can happen the first time that you use SOLVE or ∫ FN after the message MEMORY CLEAR, or it can happen if the current label no longer exists.

SOLVE ACTIVE

A running program attempted to select a program label (FN=*label*) while a SOLVE operation was running.

SOLVE(SOLVE)

A running program attempted to solve a program while a SOLVE operation was running.

SOLVE(∫FN)

A running program attempted to integrate a program while a SOLVE operation was running.

SOLVING

The calculator is solving an equation or program for its root. This might take a while.

SQRT(NEG)

Attempted to calculate the square root of a negative number.

STAT ERROR

Statistics error:

- Attempted to do a statistics calculation with $n = 0$.
- Attempted to calculate s_x s_y, $\hat{x}$, $\hat{y}$, m, r, or b with n = 1.
- Attempted to calculate r, $\hat{x}$ or $\bar{x}w$ with x–data only (all y–values equal to zero).
- Attempted to calculate $\hat{x}$, $\hat{y}$, r, m, or b with all x–values equal.

SYNTAX ERROR

A syntax error was detected during evaluation of an expression, equation, SOLVE, or ∫. Pressing ← or C clears the error message and allows you to correct the error.

TOO BIG

The magnitude of the number is too large to be converted to HEX, OCT, or BIN base; the number must be in the range
–34,359,738,368 ≤ n ≤34,359,738,367.

XEQ OVERFLOW

A running program attempted an 21st nested XEQ *label*. (Up to 20 subroutines can be nested.) Since SOLVE and ∫ FN each uses a level, they can also generate this error.

YES

The condition checked by a test instruction is true. (Occurs only when executed from the keyboard.)

Self–Test Messages:

35S-OK

The self–test and the keyboard test passed.

35S-FAIL *n*

The self–test or the keyboard test failed, and the calculator requires service.

© 2007 HP DEV CO. L. P.

Copyright message displayed after successfully completing the self–test.

Operation Index

This section is a quick reference for all functions and operations and their formulas, where appropriate. The listing is in alphabetical order by the function's name. This name is the one used in program lines. For example, the function named FIX *n* is executed as ⬑ DISPLAY 1 (1FIX) *n*.

Nonprogrammable functions have their names in key boxes. For example, ←.

Non–letter and Greek characters are alphabetized before all the letters; function names preceded by arrows (for example, →DEG) are alphabetized as if the arrow were not there.

The last column, marked ∗, refers to notes at the end of the table.

Name	Keys and Description	Page	∗
+/–	+/– Changes the sign of a number.	1–15	1
+	+ *Addition*. Returns *y* + *x*.	1–19	1
–	– *Subtraction*. Returns *y* – *x*.	1–19	1
×	× *Multiplication*. Returns *y* × *x*.	1–19	1
÷	÷ *Division*. Returns *y* ÷ *x*.	1–19	1
^	y^x *Power*. Indicates an exponent.	6–16	1
←	Deletes the last digit keyed in; clears *x*; clears a menu; erases last function keyed in an equation; deletes an equation; deletes a program step.	1–4 1–8 6–3 13–7	
^	Displays previous entry in catalog; moves to previous equation in equation list; moves program pointer to previous step.	1–28 6–3 13–11 13–20	

Name	Keys and Description	Page	*
⌄	Displays next entry in catalog; moves to next equation in equation list; moves program pointer to next line (during program entry); executes the current program line (not during program entry).	1–28 6–3 13–11 13–20	
‹ or ›	Moves the cursor and does not delete any content.	1–14	
↱ ‹ or ↱ ›	Scrolls the display to show more digits to the left and right; displays the rest of an equation or binary number; goes the next menu page in the CONST and SUMS menus.	1–11 6–4 11–8	
↱ ⌃	Goes to the top line of the equation or the first line of the last label in program mode.	6–3	
↱ ⌄	Goes to the last line of the equation or the first line of the next label in program mode.	6–3	
,	↰ , Separates the two or three arguments of a function.	6–5	1
1/x	1/x *Reciprocal.*	1–18	1
10^x	↱ 10^x *Common exponential.* Returns 10 raised to the × power.	4–2	1
%	↱ % *Percent.* Returns (*y* × *x*) ÷ 100.	4–6	1
%CHG	↰ %CHG *Percent change.* Returns *(x – y)*(100 ÷ *y).*	4–6	1
π	↰ π Returns the approximation 3.14159265359 (12 digits).	4–3	1
Σ+	Σ+ Accumulates (*y*, *x*) into statistics registers.	12–2	
Σ–	↰ Σ– Removes (*y*, *x*) from statistics registers.	12–2	
Σ*x*	↱ SUMS › (Σx) Returns the sum of *x*–values.	12–11	1

Name	Keys and Description	Page	*
Σx^2	↱ SUMS > > > (Σx^2) Returns the sum of squares of *x*–values.	12–11	1
Σxy	↱ SUMS > > > > > (Σxy) Returns the sum of products of *x*–and *y*–values.	12–11	1
Σy	↱ SUMS > > (Σy) Returns the sum of *y*–values.	12–11	1
Σy^2	↱ SUMS > > > > (Σy^2) Returns the sum of squares of *y*–values.	12–11	1
σx	↱ S,σ > > (σx) Returns population standard deviation of *x*–values: $\sqrt{\sum (x_i - \overline{x})^2 \div n}$	12–7	1
σy	↱ S,σ > > > (σy) Returns population standard deviation of *y*–values: $\sqrt{\sum (y_i - \overline{y})^2 \div n}$	12–7	1
∫ FN d *variable*	↰ ∫ (∫ FN d _) *variable* Integrates the displayed equation or the program selected by FN=, using lower limit of the variable of integration in the Y–register and upper limit of the variable of integration in the X–register.	8–2 15–7	
( )	( ) *parenthesis*. press > to leave the parenthesis for further calculation.	6–6	1
[ ]	↱ [ ]: A vector symbol for performing vector operations	10–1	1
θ	↱ θ: A complex number symbol for performing complex number operations	9–1	1

Name	Keys and Description	Page	*
A through Z	RCL *variable* Value of named variable.	6–4	1
ABS	↱ ABS *Absolute value.* Returns $\|x\|$.	4–17	1
ACOS	↱ ACOS *Arc cosine.* Returns cos $^{-1}$x.	4–4	1
ACOSH	↰ HYP ↱ ACOS *Hyperbolic arc cosine.* Returns cosh $^{-1}$ x.	4–6	1
MODE 4 (4ALG)	Activates Algebraic mode.	1–9	
ALOG	↱ 10^x *Common exponential.* Returns 10 raised to the specified power (antilogarithm).	6–16	1
ALL	↰ DISPLAY 4 (4ALL) Displays all significant digits. May have to scroll right (↱ ›) to see all of the digits.	1–23	
AND	↰ LOGIC 1 (1AND) Logic operator	11–4	1
ARG	↰ ARG Replaces a complex number with its Argument "θ"	4–17	1
ASIN	↱ ASIN *Arc sine* Returns sin $^{-1}$ x.	4–4	1
ASINH	↰ HYP ↱ ASIN *Hyperbolic arc sine.* Returns sinh $^{-1}$ x.	4–6	1
ATAN	↱ ATAN *Arc tangent.* Returns tan $^{-1}$ x.	4–4	1
ATANH	↰ HYP ↱ ATAN *Hyperbolic arc tangent.* Returns tanh $^{-1}$ x.	4–6	1
b	↰ L.R. › › › › (b) Returns the *y–intercept* of the regression line: $\bar{y} - m\bar{x}$.	12–11	1

Name	Keys and Description	Page	∗
b	↱ BASE 8 (8b) Indicates a binary number	11–2	1
↱ BASE	Displays the base–conversion menu.	11–1	
BIN	↱ BASE 4 (4BIN) Selects Binary (base 2) mode.	11–1	
C	Turns on calculator; clears *x*; clears messages and prompts; cancels menus; cancels catalogs; cancels equation entry; cancels program entry; halts execution of an equation; halts a running program.	1–1 1–4 1–8 1–29 6–3 13–7 13–19	
/*c*	↰ /c *Denominator.* Sets denominator limit for displayed fractions to *x*. If *x* = 1, displays current /*c* value.	5–4	
→°C	↱ →°C Converts ° F to ° C.	4–14	1
CF n	↰ FLAGS 2 (2CF) *n* Clears flag n (n = 0 through 11).	14–12	
↱ CLEAR	Displays menu to clear numbers or parts of memory; clears indicated variable or program from a MEM catalog; clears displayed equation;	1–5 1–28	
↱ CLEAR 3 (3ALL)	Clears all stored data, equations, and programs.	1–29	
↱ CLEAR 3 (3PGM)	Clears all programs (calculator in Program mode).	13–23	
↱ CLEAR 3 (3EQN)	Clears the displayed equation (calculator in Equation mode).	13–7	
CLΣ	↱ CLEAR 4 (4Σ) Clears statistics registers.	12–1	
CLVARS	↱ CLEAR 2 (2VARS) Clears all variables to zero.	3–6	
CL*x*	↱ CLEAR 1 (1X) Clears *x* (the X–register) to zero.	2–3 2–7 13–7	

Name	Keys and Description	Page	*
CLVARx	↱ CLEAR 6 (6CLVARx) Clears indirect variables whose address is greater than the x address to zero.	1–4	
CLSTK	↱ CLEAR 5 (5STK) Clears *all stack* levels to zero.	2–7	
→CM	↱ →cm Converts inches to centimeters.	4–14	1
nCr	↰ nCr *Combinations* of *n* items taken *r* at a time. Returns n! ÷ (r! (n – r)!).	4–15	1
COS	COS *Cosine.* Returns cos *x*.	4–3	1
COSH	↰ HYP COS *Hyperbolic cosine*. Returns cosh *x*.	4–6	1
↰ CONST	Accesses the 41 physics constants.	4–8	
d	↱ BASE 5 (5d) indicates a decimal number	11–1	1
DEC	↱ BASE 1 (1DEC) Selects Decimal mode.	11–1	
DEG	MODE 1 (1DEG) Selects Degrees angular mode.	4–4	
→DEG	↱ →DEG *Radians to degrees.* Returns (360/2π) *x*.	4–13	1
↰ DISPLAY	Displays menu to set the display format, radix (. or ,), thousand separator, and display format of complex number.	1–21	
DSE *variable*	↱ DSE *variable* *Decrement, Skip if Equal* or less. For control number *ccccccc.fffii* stored in a variable, subtracts *ii* (increment value) from *ccccccc* (counter value) and, if the result ≤*fff* (final value), skips the next program line.	14–18	
E	Begins entry of exponents and adds "E" to the number being entered. Indicates that a power of 10 follows.	1–15	1

Name	Keys and Description	Page	*
ENG n	↰ DISPLAY 3(3ENG)*n* Selects Engineering display with *n* digits following the first digit (*n* = 0 through 11).	1–22	
←ENG and ENG→	Causes the exponent display for the number being displayed to change in multiple of 3.	1–22	
ENTER	Separates two numbers keyed in sequentially; completes equation entry; evaluates the displayed equation (and stores result if appropriate).	1–19 6–4 6–11	
ENTER	ENTER Copies *x* into the Y–register, lifts *y* into the Z–register, lifts *z* into the T–register, and loses t.	2–6	
EQN	Activates or cancels (toggles) Equation–entry mode.	6–3 13–7	
e^x	↱ e^x *Natural exponential.* Returns e raised to the *x* power.	4–1	1
EXP	↱ e^x *Natural exponential.* Returns *e* raised to the specified power.	6–16	1
→°F	↰ →°F Converts °C to °F.	4–14	1
↱ FDISP	Turns on and off Fraction–display mode.	5–1	
FIX n	DISPLAY 1(1FIX) *n* Selects Fixed display with n decimal places: 0 ≤ n ≤ 11.	1–21	
↰ FLAGS	Displays the menu to set, clear, and test flags.	14–12	
FN = *label*	↰ FN= *label* Selects *labeled* program as the current function (used by SOLVE and ∫ FN).	15–1 15–7	
FP	↰ INTG 5(5FP) *Fractional part of x.*	4–17	1

Name	Keys and Description	Page	*
FS? *n*	↰ FLAGS 3 (3FS?) *n* If flag *n* (n = 0 through 11) is set, executes the next program line; if flag *n* is clear, skips the next program line.	14–12	
→GAL	↰ →gal Converts liters to gallons.	4–14	1
GRAD	MODE 3 (3GRD)Sets Grads angular mode.	4–4	
GTO · *label nnn*	Sets program pointer to line *nnn* of program *label.*	13–21	
GTO · ·	Sets program pointer to PRGM TOP.	13–21	
h	↱ BASE 6 (6h) Indicates a hexadecimal number	11–1	1
HEX	↱ BASE 2 (2HEX) Selects Hexadecimal (base 16) mode.	11–1	
↰ HYP	Displays the HYP_ prefix for hyperbolic functions.	4–6	
→HMS	↱ →HMS *Hours to hours, minutes, seconds.* Converts *x* from a decimal fraction to hours–minutes–seconds format.	4–13	1
HMS→	↰ HMS→ *Hours, minutes, seconds to hours.* Converts *x* from hours–minutes–seconds format to a decimal fraction.	4–13	1
i	Used for entering complex numbers	9–2	1
(I)/(J)	RCL (I) / (J), STO (I) / (J). Value of variable whose letter corresponds to the numeric value stored in variable I/J.	6–4 14–21	1
→IN	↰ →in Converts centimeters to inches.	4–14	1
IDIV	↰ INTG 2(2INT÷) Produces the quotient of a division operation involving two integers.	6–16	1

Name	Keys and Description	Page	*
INT÷	↰ INTG 2 (2INT÷) Produces the quotient of a division operation involving two integers.	4–2	1
INTG	↰ INTG 4 (4INTG) Obtains the greatest integer equal to or less than given number.	4–18	1
INPUT *variable*	↰ INPUT *variable* Recalls the *variable* to the X–register, displays the variable's name and value, and halts program execution. Pressing R/S (to resume program execution) or ⌄ (to execute the current program line) stores your input in the variable. (Used only in programs.)	13–13	
INV	1/x Reciprocal of argument.	6–16	1
IP	↰ INTG 6 (6IP) *Integer part* of *x*.	4–17	1
ISG *variable*	↰ ISG *variable* *Increment, Skip if Greater.* For control number *ccccccc.fffii* stored in variable, adds *ii* (increment value) to *ccccccc* (counter value) and, if the result > *fff* (final value), skips the next program line.	14–18	
→KG	↱ →kg Converts pounds to kilograms.	4–14	1
→KM	↱ →KM Converts miles to. kilometers	4–14	1
→L	↱ →l Converts gallons to liters.	4–14	1
LAST*x*	↱ LAST*x* Returns number stored in the LAST X register.	2–8	
→LB	↰ →lb Converts kilograms to pounds.	4–14	1

Name	Keys and Description	Page	*
LBL *label*	↱ LBL *label* Labels a program with a single letter for reference by the XEQ, GTO, or FN= operations. (Used only in programs.)	13–3	
LN	↱ LN *Natural logarithm.* Returns log $_e$ *x*.	4–1	1
LOG	↰ LOG *Common logarithm.* Returns log$_{10}$ *x*.	4–1	1
↰ L.R.	Displays menu for linear regression.	12–4	
m	↰ L.R. > > > (m) Returns the slope of the regression line: $[\Sigma(x_i-\bar{x})(y_i-\bar{y})]\div\Sigma(x_i-\bar{x})^2$	12–7	1
→MILE	↰ →MILE Converts kilometers to miles.	4–14	1
↰ MEM	Displays the amount of available memory and the catalog menu.	1–28	
↰ MEM 2 (2PGM)	Begins catalog of programs.	13–22	
↰ MEM 1 (1VAR)	Begins catalog of variables.	3–4	
MODE	Displays menu to set ALG or RPN mode or angular modes.	1–7 4–4	
n	↱ SUMS (n) Returns the number of sets of data points.	12–11	1
NAND	↰ LOGIC 5 (5NAND) Logic operator	11–4	1
NOR	↰ LOGIC 6 (6XOR) Logic operator	11–4	1
NOT	↰ LOGIC 4 (4NOT) Logic operator	11–4	1
o	↱ BASE 7 (7o) Indicates an octal number	11–2	1
OCT	↱ BASE 3 (3OCT) Selects Octal (base 8) mode.	11–1	

Name	Keys and Description	Page	*
OR	↰ LOGIC 3 (3OR) Logic operator	11–4	1
↱ OFF	Turns the calculator off.	1–1	
nPr	↱ nPr *Permutations* of *n* items taken *r* at a time. Returns *n*!÷(*n* – *r*)!.	4–15	1
↱ PRGM	Activates or cancels (toggles) Program–entry mode.	13–6	
PSE	↱ PSE *Pause.* Halts program execution briefly to display *x*, variable, or equation, then resumes. (Used only in programs.)	13–18 13–19	
r	↰ L.R. > > (r) Returns the correlation coefficient between the *x*–and *y*–values: $\dfrac{\sum(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum(x_i-\overline{x})^2\times(y_i-\overline{y})^2}}$	12–7	1
r*θ*a	↰ DISPLAY 1 0 (10rθa) Changes the display of complex numbers.	1–25	
RAD	MODE 1 (1RAD) Selects Radians angular mode.	4–4	
→RAD	↰ →RAD *Degrees to radians.* Returns (2π/360) *x*.	4–13	1
RADIX ,	↰ DISPLAY 6 (6,) Selects the comma as the radix mark (decimal point).	1–23	
RADIX .	↰ DISPLAY 5 (5.) Selects the period as the radix mark (decimal point).	1–23	
RANDOM	↱ RAND Executes the RANDOM function. Returns a random number in the range 0 through 1.	4–15	1
RCL *variable*	RCL *variable* *Recall.* Copies *variable* into the X–register.	3–7	

Name	Keys and Description	Page	∗
RCL+ *variable*	RCL + *variable* Returns *x* + variable.	3–7	
RCL– *variable*	RCL − *variable.* Returns *x* – variable.	3–7	
RCLx *variable*	RCL × *variable.* Returns *x* × variable.	3–7	
RCL÷ *variable*	RCL ÷ *variable.* Returns *x* ÷ variable.	3–7	
RMDR	↰ INTG 3 (3Rmdr) Produces the remainder of a division operation involving two integers.	6–16	1
RND	↰ RND *Round.* Rounds *x* to n decimal places in FIX *n* display mode; to *n* + 1 significant digits in SCI *n* or ENG *n* display mode; or to decimal number closest to displayed fraction in Fraction–display mode.	4–18 5–8	1
RPN	MODE 5 (5RPN)Activates Reverse Polish notation.	1–9	
RTN	↰ RTN *Return.* Marks the end of a program; the program pointer returns to the top or to the calling routine.	13–4 14–1	
R↓	R↓ *Roll down.* Moves *t* to the Z–register, *z* to the Y–register, *y* to the X–register, and *x* to the T–register in RPN mode. Displays the X,Y,Z,T menu to review the stack in ALG mode.	2–3 C–7	
R↑	↱ R↑ *Roll up.* Moves *t* to the X–register, *z* to the T–register, *y* to the Z–register, and *x* to the Y–register in RPN mode. Displays the X,Y,Z,T menu to review the stack in ALG mode.	2–3 C–7	
↱ S,σ	Displays the standard–deviation Menu.	12–4	

Name	Keys and Description	Page	*
SCI n	↰ DISPLAY 2 (2SCI) *n* Selects Scientific display with *n* decimal places. (*n* = 0 through 11.)	1–22	
SEED	↰ SEED Restarts the random–number sequence with the seed $\lvert x \rvert$.	4–15	
SF *n*	↰ FLAGS 1 (1SF) *n* Sets flag *n* (*n* = 0 through 11).	14–12	
SGN	↰ INTG 1 (1SGN) *Indicates the sign of x.*	4–17	1
↰ SHOW	Shows the full mantissa (all 12 digits) of *x* (or the number in the current program line); displays hex checksum and decimal byte length for equations and programs.	6–19 13–23	
SIN	SIN *Sine.* Returns sin *x*.	4–3	1
SINH	↰ HYP SIN *Hyperbolic sine.* Returns sinh *x*.	4–6	1
↱ SOLVE *variable*	↱ SOLVE *variable* Solves the displayed equation or the program selected by FN=, using initial estimates in *variable* and *x*.	7–1 15–1	
↱ SPACE	Inserts a blank space character during equation entry.	14–14	1
SQ	↱ x^2 *Square* of argument.	6–16	1
SQRT	$\sqrt{x}$ *Square root* of *x*.	6–16	1
STO *variable*	↱ STO *variable* Store. Copies *x* into variable.	3–2	
STO + *variable*	↱ STO + *variable* Stores variable + *x* into variable.	3–6	
STO – *variable*	↱ STO − *variable* Stores variable – *x* into variable.	3–6	
STO × *variable*	↱ STO × *variable* Stores variable × *x* into variable.	3–6	
STO ÷ *variable*	↱ STO ÷ *variable* Stores variable ÷ *x* into variable.	3–6	

Name	Keys and Description	Page	*
STOP	R/S *Run/*stop. Begins program execution at the current program line; stops a running program and displays the X–register.	13–19	
↱ SUMS	Displays the summation menu.	12–4	
s*x*	↱ S,σ (sx) Returns sample standard deviation of *x*–values: $\sqrt{\sum (x_i - \bar{x})^2 \div (n-1)}$	12–6	1
s*y*	↱ S,σ > (sy) Returns sample standard deviation of *y*–values: $\sqrt{\sum (y_i - \bar{y})^2 \div (n-1)}$	12–6	1
TAN	TAN *Tangent*. Returns tan *x*.	4–3	1
TANH	↰ HYP TAN *Hyperbolic tangent.* Returns tanh *x*.	4–6	1
VIEW *variable*	↰ VIEW *variable* Displays the labeled contents of *variable* without recalling the value to the stack.	3–4 13–15	
XEQ	Evaluates the displayed equation.	6–12	
XEQ *label*	XEQ *label* Executes the program identified by *label.*	14–1	
x^2	↱ x^2 *Square* of *x*.	4–2	1
$\sqrt{\ }$ x	$\sqrt{x}$ *Square root* of *x*.	4–2	1
$\sqrt[x]{y}$	↰ $\sqrt[x]{y}$ *The x^{th} root of y.*	4–2	1
$\bar{x}$	↰ $\bar{x},\bar{y}$ ($\bar{x}$) Returns the mean of *x* values: $\Sigma x_i \div n$.	12–4	1

Name	Keys and Description	Page	*
$\hat{x}$	↰ L.R. ($\hat{x}$) Given a *y*–value in the X–register, returns the *x–estimate* based on the regression line: $\hat{x}$ = (*y* – b) ÷ m.	12–11	1
!	↱ ! Factorial (or gamma). Returns (*x*)(*x* – 1) ... (2)(1), or Γ (*x* + 1).	4–15	1
XROOT	↰ $\sqrt[x]{y}$ The *argument*$_1$ root of *argument*$_2$.	6–16	1
$\bar{x}$ w	↰ $\bar{x},\bar{y}$ > > ($\bar{x}$ w)Returns weighted mean of *x* values: ($\Sigma y_i x_i$) ÷ Σy_i.	12–4	1
↰ $\bar{x},\bar{y}$	Displays the mean (arithmetic average) menu.	12–4	
x<> variable	↰ *x*⇆ *x exchange.* Exchanges *x* with a variable.	3–8	
x<>y	*x*◂▸*y* *x exchange y.* Moves *x* to the Y–register and *y* to the X–register.	2–4	
↰ *x?y*	Displays the "*x?y*" comparison tests menu.	14–7	
x≠y	↰ *x?y* (≠) If *x≠y*, executes next program line; if *x=y*, skips next program line.	14–7	
x≤y?	↰ *x?y* > (≤) If *x≤y*, executes next program line; if *x>y*, skips next program line.	14–7	
x<y?	↰ *x?y* > >(<) If *x<y*, executes next program line; if *x≥y*, skips next program line.	14–7	
x>y?	↰ *x?y* > > > (>) If *x>y*, executes next program line; if *x≤y*, skips next program line.	14–7	
x≥y?	↰ *x?y* > > > > (≥) If *x≥y*, executes next program line; if *x<y*, skips next program line.	14–7	

Name	Keys and Description	Page	*
x=*y*?	↰ x?y ▷ ▷ ▷ ▷ ▷ (=) If *x*=*y*, executes next program line; if *x*≠*y*, skips next program line.	14–7	
↱ x?0	Displays the "*x*?0" comparison tests menu.	14–7	
x≠0?	↱ x?0 (≠) If *x*≠0, *executes* next program line; if *x*=0, skips the next program line.	14–7	
x≤0?	↱ x?0 ▷ (≤) If *x*≤0, executes next program line; if *x*>0, skips next program line.	14–7	
x<0?	↱ x?0 ▷ ▷ (<) If *x*<0, executes next program line; if *x*≥0, skips next program line.	14–7	
x>0?	↱ x?0 ▷ ▷ ▷ (>) If *x*>0, executes next program line; if *x*≤0, skips next program line.	14–7	
x≥0?	↱ x?0 ▷ ▷ ▷ ▷ (≥) If *x*≥0, executes next program line; if *x*<0, skips next program line.	14–7	
x=0?	↱ x?0 ▷ ▷ ▷ ▷ ▷ (=) If *x*=0, executes next program line; if *x*≠0, skips next program line:	14–7	
XOR	↰ LOGIC 2 (2XOR) Logic operator	11–4	1
xiy	↰ DISPLAY 9 (9xiy) Changes display of complex numbers.	4–11	
x+yi	↰ DISPLAY · 1 (11x+yi) Changes display of complex numbers. ALG mode only.	1–25	
$\bar{y}$	↰ x̄,ȳ ▷ (ȳ) Returns the mean of *y* values. $\Sigma y_i \div n$.	12–4	1

Name	Keys and Description	Page	*
$\hat{y}$	↰ L.R. > ($\hat{y}$) Given an x–value in the X–register, returns the *y*–estimate based on the regression line: $\hat{y} = m\,x + b$.	12–11	1
y^x	y^x *Power.* Returns *y* raised to the xth power.	4–2	1

Notes:

1. Function can be used in equations.

Index