Texas Instruments

TI-86 GRAPHING CALCULATOR

GUIDEBOOK

Important

Texas Instruments makes no warranty, either expressed or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an “as-is” basis.

In no event shall Texas Instruments be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of Texas Instruments, regardless of the form of action, shall not exceed the purchase price of this equipment. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

US FCC Information Concerning Radio Frequency Interference

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 with 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, you can 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 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/television technician for help.

Table of Contents

Chapter 9: Parametric Graphing 123

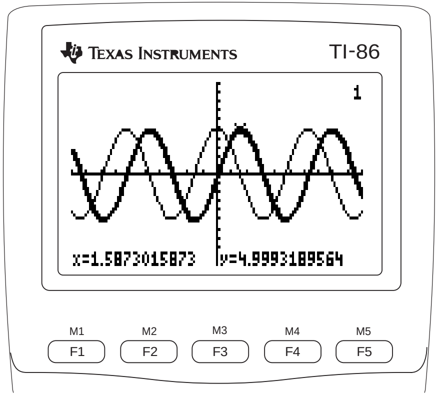
Chapter 10: Differential Equation Graphing 131

Chapter 11: Lists 151

Chapter 12: Vectors 167

Chapter 13: Matrices 177

TI-86 Quick Start



Preparing to Use Your New TI-86

The brief examples in the TI-86 Quick Start demonstrate some common TI-86 features. Before you begin, you must install the batteries, turn on the calculator, adjust the contrast, and reset the memory and the defaults. Chapter 1 has more details on these topics.

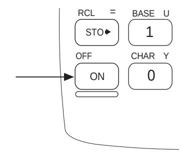
Installing the AAA Batteries

Four AAA batteries are included in the TI-86 retail package. Remove the batteries from the package and install them in the battery compartment on the back of the calculator. Arrange the batteries according to the polarity (+ and −) diagram in the battery compartment.

Turning On and Turning Off the TI-86

After about four minutes of inactivity, the TI-86 turns off automatically.

To turn on the TI-86, press [ON], which is in the bottom-left corner of the keyboard. You should see the entry cursor (■) blinking in the top-left corner of the screen. If you do not see it, adjust the contrast (see below).

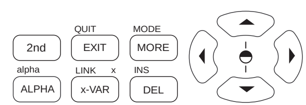


To turn off the calculator, press [2nd], and then the key under OFF, which is [ON]. This guidebook uses brackets ([and]) to express [2nd] and [ALPHA] keystroke combinations. For example, to turn off the TI-86, press [2nd] [OFF].

Adjusting the Contrast

If you release [▲] or [▼] while adjusting the contrast, you must press [2nd] again to continue the adjustment.

1. Press and release the yellow [2nd] key.
2. Press and hold [▲] or [▼] (above or below the half-shaded circle).
   - To darken the screen contrast, press and hold [▲].
   - To lighten the screen contrast, press and hold [▼].



Resetting All Memory and Defaults

To reset all memory and defaults, press [2nd] [MEM] [F3] [F1] [F4]. The messages **Mem cleared** and **Defaults set** are displayed on the home screen, confirming that all memory and defaults are reset. You may need to adjust the contrast after memory and default reset.

Calculating on the Home Screen

To express [2nd] and [ALPHA] keystroke combinations, this guidebook places brackets ([and]) around the word above the key to press.

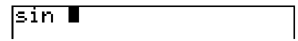
To replicate the screens shown in the Quick Start activities, reset all memory and defaults once before you begin. Before doing an activity, press [CLEAR] to clear the screen (except before the entry retrieval and integer-part examples). Otherwise, the screens your TI-86 shows may differ from the screens pictured next to the activities.

Calculating the Sine of a Number

The TI-86 on-screen division symbol is a forward slash (/), as in a fraction.

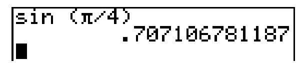
Following evaluation, the entry cursor automatically moves to the next line, ready for a new entry.

1. Enter the sine function. ([CLEAR]) [SIN]



2. Enter a value. You can enter an expression, which is evaluated when you press [ENTER]. [(] [2nd] [π] [÷] **4** [)]

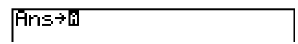
3. Evaluate the problem. The evaluation of the expression sine(π/4) is displayed. [ENTER]



Storing the Last Answer to a Variable

*When the TI-86 evaluates an expression, it automatically stores the answer to the built-in variable **Ans**, replacing any previous value.*

1. Paste the store symbol (→) to the screen. Since a value must precede →, but you did not enter a value, the TI-86 automatically pasted **Ans** before →. ([CLEAR]) [STO▶]



(continued)

When ALPHA-lock is on and you press a key, the letters printed in blue above the keys are pasted to the screen. In the example, press [2] *to enter a* **V**.

❷	Enter the variable name to which you want to store the last answer. ALPHA-lock is on.	[V]	Ans→V
❸	Store the last answer to the variable. The stored value is displayed on the next line.	[ENTER]	Ans→V .70710678187

Using a Variable in an Expression

❶	Enter the variable, and then square it.	([CLEAR]) [ALPHA] [V] [x^2]	V²
❷	Evaluate. The value stored to the variable **V** is squared and displayed.	[ENTER]	V² .5

Editing an Expression

❶	Enter the expression **(25+14)(4−3.2)**.	([CLEAR]) [(] **25** [+] **14** [)] [(] **4** [−] **3** [.] **2** [)]	(25+14)(4−3.2)
❷	Change **3.2** to **2.3**.	[◄] [◄] [◄] [◄] **2** [►] **3**	(25+14)(4−2.3
❸	Move the cursor to the beginning of the expression and insert a value. The insert cursor blinks between **3** and **25**.	[2nd] [◄] [2nd] [INS] **3**	3_25+14)(4−2.3)
❹	Evaluate. The result is displayed.	[ENTER]	3(25+14)(4−2.3) 198.9

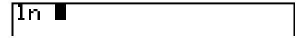
You need not move the cursor to the end of the line to evaluate the expression.

⊖ *negates a value, as in* **-2***.* ⊟ *subtracts, as in* **5-2=3***.*

An ellipsis (...) indicates that the result continues beyond the screen.

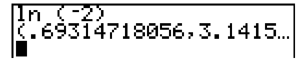
Displaying a Complex Number as a Result

❶ Enter the natural log function. — ([CLEAR]) [LN]



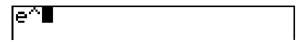
❷ Enter a negative number. — [(] [(-)] **2** [)]

❸ Evaluate. The result is displayed as a complex number. — [ENTER] (press [▶] to display more)



Using a List with a Function

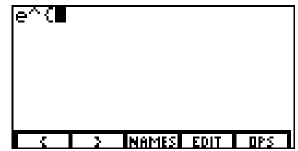
❶ Enter the exponential function. — ([CLEAR]) [2nd] [e^x]



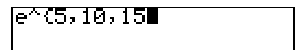
❷ Display the LIST menu, and then select the open brace (**{**) from the LIST menu. — [2nd] [LIST] [F1]

On the TI-86, **{** specifies the beginning of a list.

LIST *menu*

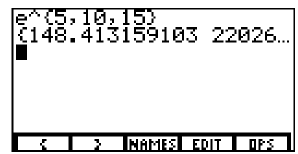


❸ Enter the list elements. Separate each element from the next with a comma. — **5** [,] **10** [,] **15**



❹ Select the close brace (**}**) from the LIST menu to specify the end of the list. — [F2]

❺ Evaluate. The results of the constant **e** raised to the 5th, 10th, and 15th powers are displayed as list elements. — [ENTER] (press [▶] to display more)



Displaying the Integer Part of Real Numbers in a List

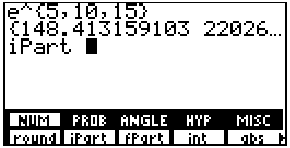
1. Display the MATH menu. (The MATH menu automatically replaces the LIST menu from the last activity.) [2nd] [MATH]

MATH *menu*

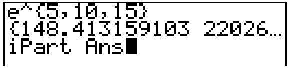
2. Select **NUM** to display the MATH NUM menu. The MATH menu shifts up. [F1]

MATH NUM *menu*

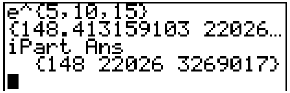
3. Select the **iPart** (integer part) function from the MATH NUM menu. **iPart** is pasted to the screen. (The previous entry was left on the screen to illustrate the effect of **iPart** on the previous answer.) [F2]



4. Paste **Ans** to the cursor location. (The result list from the previous activity is stored to **Ans**.) [2nd] [ANS]



5. Display the integer part of the result list elements from the previous activity. [ENTER]



Removing (Exiting) a Menu

1. In the previous example, the MATH menu and the MATH NUM menu are displayed ([2nd] [MATH] [F1]).

2. Remove the MATH NUM menu from the screen. [EXIT]

3. Remove the MATH menu from the screen. [EXIT]

Finding the Square Root

❶ Paste the square root function to the screen. (CLEAR) 2nd [√]

❷ Enter a value for which you want to find the square root. **144**

❸ Evaluate the expression. The square root of **144** is displayed. ENTER

Calculating Derivatives

❶ Display the CALC menu, and then select **der1**. (CLEAR) 2nd [CALC] F3

CALC *menu*

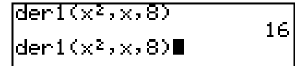
❷ Enter an expression (**x²**) with respect to a variable (**x**) at a given point (**8**). x-VAR x^2 , x-VAR , **8**)

❸ Evaluate. The first derivative of **x²** with respect to **x** at **8** is displayed. ENTER

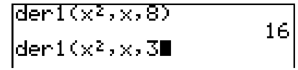
When you press [ENTER]*, the TI-86 stores the expression or instruction you entered to the built-in memory storage area called* ENTRY*.*

Retrieving, Editing, and Re-evaluating the Previous Entry

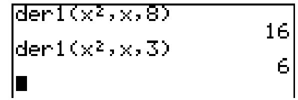
1. Retrieve the last entry from the previous example. (The last activity was not cleared.) — [2nd] [ENTRY]



2. Edit the retrieved entry. — [◄] [◄] **3**



3. Evaluate. The first derivative of $\mathbf{x^2}$ with respect to **x** at **3** is displayed. — [ENTER]



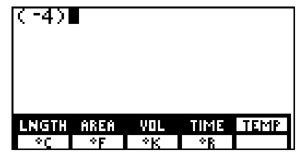
When expressing a measurement for a conversion, you do not enter a unit symbol manually. For example, you need not enter ° to designate degrees.

Converting Degrees Fahrenheit to Degrees Celsius

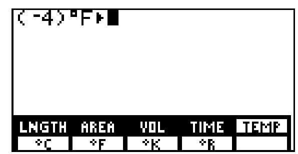
1. Display the CONV menu. — ([CLEAR]) [2nd] [CONV]

2. Display the CONV TEMP menu. The CONV menu shifts up and **TEMP** is highlighted. — [F5]

3. Enter the known measurement. If the measurement is negative, use parentheses. In this example, if you omit parentheses, the TI-86 converts 4°F to about ⁻15.5°C, which it then negates (changes the sign of), returning a positive 15.5°C. — [(] [(-)] **4** [)]

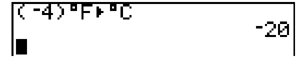


4. Select °**F** to designate Fahrenheit as the known measurement unit. °**F** and the conversion symbol (►) are displayed after the measurement. *(continued)* — [F2]



5. Select °**C** to designate Celsius as the unit to which you want to convert. — [F1]

6. Convert. The °C equivalent of ⁻4°F is displayed. — [ENTER]



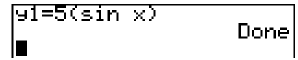
Storing an Unevaluated Expression to an Equation Variable

When storing to an equation variable using **=**, *enter the equation variable first, then* **=**, *and then the unevaluated expression. This is the opposite from the order for storing to most other variables on the TI-86.*

1. Enter the built-in equation variable **y1**. — ([CLEAR]) [2nd] [alpha] [Y] **1**

2. Enter the equals sign (=). — [ALPHA] [=]

3. Enter an expression in terms of **x**. — **5** [(] [SIN] [x-VAR] [)]
4. Store the expression. — [ENTER]



The next section shows how to graph the functions **y1=5(sin x)** and **y2=5(cos x)**.

Plotting Functions on the Graph Screen

The TI-86 plots four types of functions on the graph screen. To plot a graph, you must store an unevaluated expression to a built-in equation variable.

Each activity in this section builds upon the activity that precedes it. You must start here and perform the activities in the sequence in which they are presented. The first activity in this section assumes you are continuing from the last activity in the previous section.

Displaying and Entering Functions in the Equation Editor

1. Display the GRAPH menu. — [GRAPH]

(*continued*)

In the equation editor, you must express each equation in terms of the independent variable **x** *(in* **Func** *graphing mode only; Chapter 5).*

❷ Select **y(x)=** from the GRAPH menu to display the equation editor. **5(sin x)** is the unevaluated expression stored to **y1** in the previous activity. The equation editor menu is displayed as the lower menu.

[F1]

equation editor menu

❸ Move the cursor down. The **y2=** prompt is displayed.

[▼]

❹ Enter the expression **5(cos x)** at the **y2=** prompt. Notice that the equals sign (**=**) of **y2** is highlighted after you enter **5**. Also, the equals sign of **y1** is highlighted. This indicates that both equations are selected to be graphed (Chapter 5).

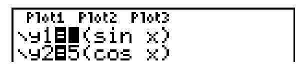
5 [(] [COS] [x-VAR] [)]

Changing the Graph Style of a Function

In the equation editor, the icon to the left of each equation specifies the style in which the graph of that equation appears when you plot it on the graph screen.

❶ Move the cursor to **y1**.

[▲]



❷ Display the next menu group of the equation editor menu. (▸ at the end of a menu group indicates that the menu has more items.)

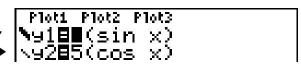
[MORE]

To display up to seven graph styles, depending on the graphing mode, repeat [F3]*.*

❸ Select **STYLE** from the equation editor menu to set (thick) graph style for **y1**.

[F3]

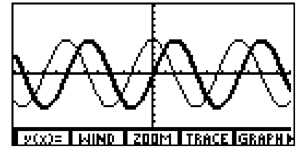
graph style icons



Plotting a Function on the Graph Screen

❶ Select **GRAPH** from the GRAPH menu to plot the graph on the graph screen. The x- and y-axes and GRAPH menu are displayed. Then each selected graph is plotted in the order in which it is listed in the equation editor.

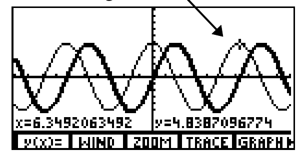
[2nd] [M5]



❷ When the graph is plotted, you can move the free-moving cursor (+) around the graph screen. The cursor coordinates are displayed at the bottom of the graph.

[▶] [▼] [◀] [▲]

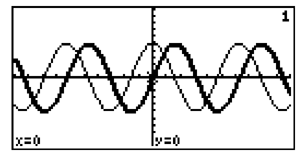
free-moving cursor



Tracing a Function

❶ Select **TRACE** from the GRAPH menu to activate the trace cursor, with which you can trace along the graph of any selected function. The number of the current function (the **1** in **y1**) is displayed in the top-right corner.

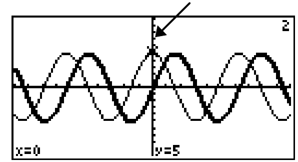
[F4]



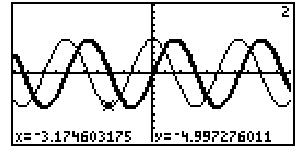
❷ Move the trace cursor from the function **y1** to the function **y2**. The **1** in the top-right corner changes to **2**; the **y** value changes to the value of **y2** at **x=0**. *(continued)*

[▲]

trace cursor

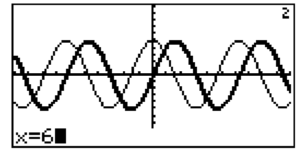


3. Trace the function **y2**. As you trace, the displayed **y** value is the solution for **5(cos x)** at the current **x** value, which also is displayed on the screen. — ▶ and ◀

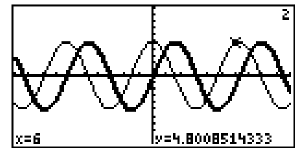


Evaluating y for a Specific x Value (During a Trace)

1. Enter a real number (or an expression that resolves to a real number) that is within the dimensions of the current graph screen. When you enter the first character, the **x=** prompt is displayed. — **6**



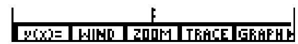
2. Evaluate **y2** at **x=6**. The trace cursor moves directly to the solution. The **y** value, or solution of the equation at **x**, is displayed on the screen. — [ENTER]



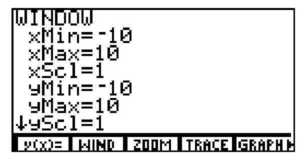
Changing a Window Variable Value

The window variables values determine the dimensions of the graph screen.

1. Display the GRAPH menu. — [GRAPH]



2. Select **WIND** from the GRAPH menu to display the window editor. — [F2]



(*continued*)

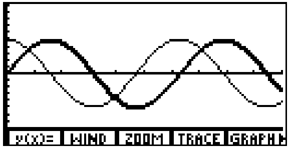
3. Change the value stored in the **xMin** window variable to **0**.

   0

4. Plot the graph on the redefined graph screen. Since **xMin=0**, only the first and fourth quadrants of the graph plane are displayed.

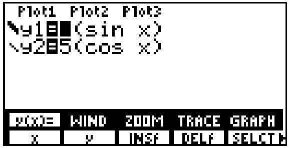
   [F5]

Deselecting a Function

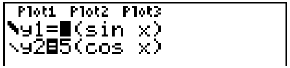
1. Select **y(x)=** from the GRAPH menu to display the equation editor and equation editor menu. The GRAPH menu shifts up and **y(x)=** is highlighted.

   [F1]

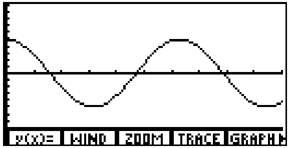
2. Select **SELCT** from the equation editor menu to deselect the function **y1=**. The equals sign is no longer highlighted.

   [F5]

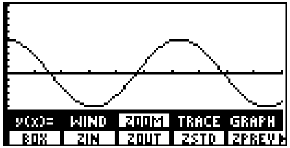
3. Plot the graph on the graph screen. Since you deselected **y1**, the TI-86 only plots **y2**. To select a function in the equation editor, repeat these steps. (**SELCT** both selects and deselects functions.)

   [2nd] [M5]

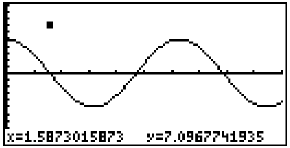
Zooming In on a Portion of the Graph Screen

❶ Select **ZOOM** to display the GRAPH ZOOM menu. The GRAPH menu shifts up and **ZOOM** is highlighted. [F3]

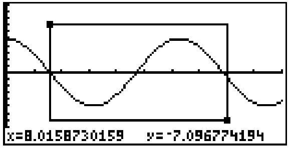


❷ Select **BOX** from the GRAPH ZOOM menu to activate the zoom-box cursor. [F1]

❸ Move the zoom-box cursor to a point that is to be a corner of the redefined graph screen, and then mark the point with a small square. [▶] [▼] [◀] [▲] [ENTER]

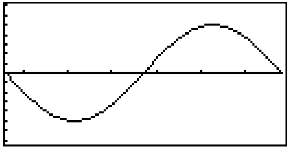


❹ Move the cursor away from the small square to a point that is to be the opposite corner of the redefined graph screen. As you move the cursor, a rectangle is drawn on the graph. [▶] [▼] [◀] [▲]

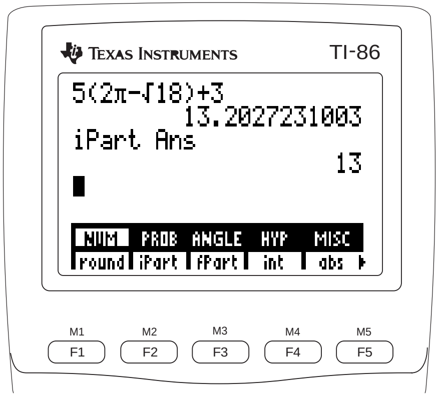


❺ Zoom in on the graph. The window variables change automatically to the specifications of the zoom box. [ENTER]

❻ Clear the menus from the graph screen. [CLEAR]



1 Operating the TI-86



Installing or Replacing Batteries

Your new TI-86 includes four AAA alkaline batteries. You must install them before you can turn on the calculator. A lithium backup battery is installed in the calculator already.

To express 2nd *and* ALPHA *keystroke combinations, this guidebook places brackets* ([and]) *around the word above the key to press.*

Do not remove the lithium backup battery unless four fresh AAA batteries are in place.
Properly dispose of the old batteries.

1. If the calculator is on, turn it off (press 2nd [OFF]) to avoid loss of information stored in memory.
2. Slide the protective cover over the keyboard.
3. Holding the calculator upright, push down on the battery cover latch, and then remove the cover.
4. Remove all four old batteries.
5. Install four new AAA alkaline batteries, arranged according to the polarity (+ and −) diagram inside the battery compartment.
6. Replace the battery cover by inserting the two prongs into the two slots at the bottom of the battery compartment, and then push the cover until the latch snaps closed.

When to Replace Batteries

If you do not use your TI-86 frequently, the AAA batteries could last more than two weeks after the first low-battery message.

When the AAA batteries are low, a low-battery message is displayed as you turn on the calculator. Generally, the calculator will continue to operate for one or two weeks after the low-battery message is first displayed. Eventually, the TI-86 will turn off automatically and will not operate until you replace the AAA batteries.

```
Your batteries
are low.

Recommend
change of
batteries.
```

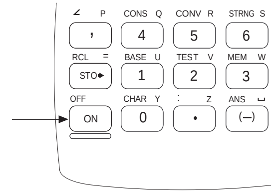
The lithium backup battery is inside the battery compartment, above the AAA batteries. It retains all memory when the AAA batteries are low or have been removed. To avoid loss of data, do not remove the lithium battery unless four fresh AAA batteries are installed. Replace the lithium backup battery about every three or four years.

Properly dispose of the old battery.

To replace the lithium backup battery, remove the battery cover and unscrew the tiny screw holding the BACK UP BATTERY cover in place. Install a new CR1616 or CR1620 battery according to the polarity (+ and −) diagram on the cover. Replace the cover and screw.

Turning On and Turning Off the TI-86

To turn on the TI-86, press [ON].



- If you previously had turned off the calculator by pressing [2nd] [OFF], the TI-86 clears any errors and displays the home screen as it was last displayed.
- If Automatic Power Down™ (APD™) previously had turned off the calculator, the TI-86 will return as you left it, including the display, cursor, and any error.

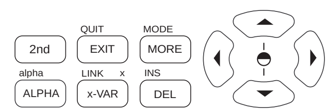
To turn off the TI-86 manually, press [2nd] [OFF]. All settings and memory contents are retained by the Constant Memory™ feature. Any error condition is cleared.

APD turns off the TI-86 automatically after about four minutes of non-use to extend battery life.

Adjusting the Display Contrast

If you release [▲] or [▼] while adjusting the contrast, you must press [2nd] again to continue the adjustment.

1. Press and release the yellow [2nd] key.
2. Press and hold [▲] or [▼] (above or below the half-shaded circle).
   - To darken the screen contrast, press and hold [▲].
   - To lighten the screen contrast, press and hold [▼].



The TI-86 has 40 contrast settings, so each number **0** *through* **9** *represents four settings.*

You can adjust the display contrast anytime to suit your viewing angle and lighting conditions. As you adjust, a number from **0** (lightest) to **9** (darkest) in the top-right corner indicates the current contrast setting. The number is not visible when the contrast is extremely light or dark.

As the batteries weaken over time, the actual contrast level of each number shifts. For example, say you set the contrast to **3** with fresh batteries. As the batteries weaken, you will need to set the contrast to **4**, then **5**, then **6**, and so on, to retain the original contrast level. However, you need not replace the batteries until the low-battery message is displayed.

The Home Screen

When you first turn on your TI-86, the home screen is displayed. Initially, the home screen is a blank screen, except for the entry cursor (■) in the top-left corner. If you do not see the cursor, press [2nd], and then press and hold [▼] or [▲] to adjust the contrast (page 17).

On the home screen, you can enter and evaluate expressions, and view the results. You also can execute instructions, store and recall variable values, and set up graphs and editors.

To return to the home screen from any other screen, press [2nd] [QUIT].

Displaying Entries and Answers

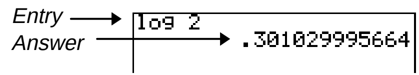
You need not clear the home screen to begin a new entry.

The home screen displays up to eight lines with a maximum of 21 characters per line. If an expression or series of instructions exceeds 21 characters and spaces, it automatically continues on the next line.

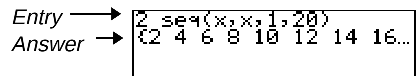
After all eight lines are full, text scrolls off the top of the display. You can press [▲] to scroll up the home screen, only as far as the first character in the current entry. To retrieve, edit, and re-execute previous entries, use [2nd] [ENTRY] (page 28).

The mode settings control the way the TI-86 interprets expressions and displays answers (page 34).

When an entry is executed on the home screen, the answer is displayed on the right side of the next line. When you execute an instruction, **Done** is typically displayed on the right side of the next line.



If an answer is too long to display on the screen, an ellipsis (...) is displayed, initially to the right. To view more of the answer, press [▶]. When you do, an ellipsis is displayed to the left. To scroll back, press [◀].



Entering Numbers

The TI-86 on-screen division symbol is a forward slash (/), as in a fraction.

A symbol or abbreviation of each key's primary function is printed in white on the key. For example, when you press [+], a plus sign is pasted to the cursor location. This guidebook describes number-entry keystrokes as **1**, **2**, **3**, and so on, instead of [1] [2] [3].

Entering Negative Numbers

To enter a negative number, press [(-)] (the negate key), and then press the appropriate number keys. For example, to enter **⁻5**, press [(-)] **5**. Do not attempt to express a negative number using [−] (the subtract key). [(-)] and [−] are two different keys with different uses.

Always use parentheses to clarify negation when you use conversion instructions (Chapter 4).

The order in which the TI-86 evaluates negation and other functions within an expression is governed by the Equation Operating System™ (Appendix). For example, the result of **⁻4²** is **⁻16**, while the result of **(⁻4)²** is **16**. If you are unsure about the order of evaluation, use [(] and [)] to clarify the intended use of the negation symbol.

Using Scientific or Engineering Notation

❶	Enter the mantissa (part of the number that precedes the exponent). This value can be an expression.	[(] **19** [÷] **2** [)]	`(19/2)▮`
❷	Paste **E** to the cursor location.	[EE]	`(19/2)E▮`
❸	If the exponent is negative, paste ⁻ to the cursor location. Then enter a one-, two-, or three-digit exponent.	[(-)] **2**	`(19/2)E-2▮`
❹	Evaluate the expression.	[ENTER]	`(19/2)E-2` `.095`

In scientific notation only, one digit precedes the decimal.

In engineering notation, one, two, or three digits precede the decimal and the power of 10 exponent is a multiple of 3.

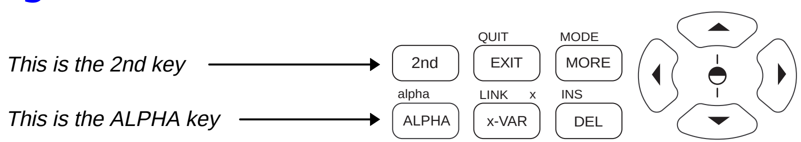
When you include scientific- or engineering-notation numbers in an expression, the TI-86 does not necessarily display answers in scientific or engineering notation. The mode settings (page 34) and the size of the number determine the notation of displayed answers.

Entering Complex Numbers

On the TI-86, the complex number a+bi is entered as **(**a,b**)** in rectangular complex-number form or as **(**$r\angle\theta$**)** in polar complex-number form. For more information about complex numbers, read Chapter 4.

```
(1,2)+(-3,1)
                (-2,3)
(1∠2)*3
(-1.24844050964,2.72…
▮
```

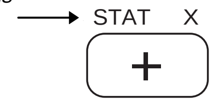
Entering Other Characters



The 2nd Key

The [2nd] key is yellow. When you press [2nd], the cursor becomes ⇧ (the 2nd cursor). When you press the next key, the yellow character, abbreviation, or word printed above that key is activated, instead of the key's primary function.

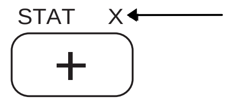
[2nd] [STAT] *returns the* STAT *menu*



The ALPHA Key

To enter a space within text, press [ALPHA] [␣]. *Spaces are not valid within variable names.*

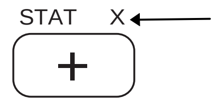
The [ALPHA] key is blue. When you press [ALPHA], the cursor becomes Ⓐ (the uppercase ALPHA cursor). When you press the next key, the blue uppercase character printed above that key is pasted to the cursor location.



[ALPHA] [X] *returns an* **X**

For convenience, you can press [x-VAR] *instead of* [2nd] [alpha] [x] *to enter the commonly used* **x** *variable.*

When you press [2nd] [alpha], the cursor becomes ⓐ (the lowercase alpha cursor). When you press the next key, the lowercase version of the blue character is pasted to the cursor location.



[2nd] [alpha] [X] *returns an* **x**

ALPHA-lock and alpha-lock

The **Name=** *prompt and store symbol* (**→**) *set ALPHA-lock automatically.*

To enter more than one uppercase or lowercase alpha character consecutively, set ALPHA-lock (for uppercase letters) or alpha-lock (for lowercase letters).

To set ALPHA-lock when the entry cursor is displayed, press [ALPHA] [ALPHA].

- To cancel ALPHA-lock, press [ALPHA].
- To switch from ALPHA-lock to alpha-lock, press [2nd] [alpha].

To set alpha-lock when the entry cursor is displayed, press [2nd] [alpha] [ALPHA].

- To cancel alpha-lock, press [ALPHA] [ALPHA].

In most cases, the appearance of the cursor indicates what will happen when you press the next key.

Graphs and editors sometimes use additional cursors, which are described in other chapters.

♦ To switch from alpha-lock to ALPHA-lock, press [ALPHA].

You can use [2nd] when ALPHA-lock or alpha-lock is on. Also, if you press a key that has no blue character above it, such as [GRAPH], [DEL], or [◄], the key's primary function still applies.

Common Cursors

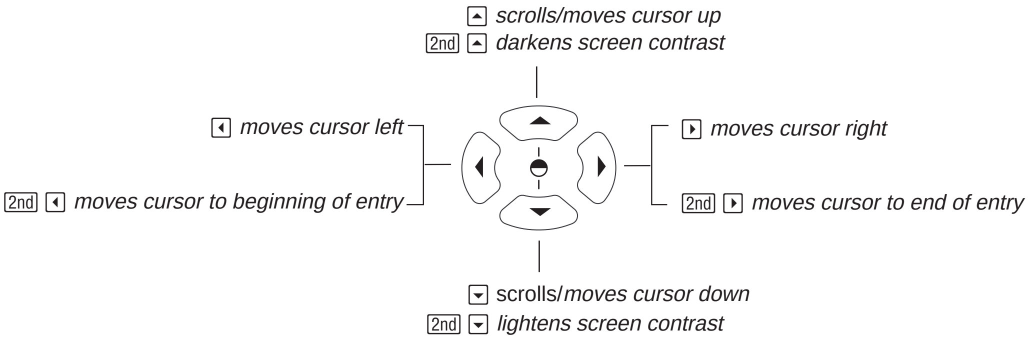
Entry	■	Enters a character at the cursor, overwriting any existing character
Insert	__	Inserts a character at the cursor location and shifts remaining characters right
Second	⬆	Enters a 2nd character or executes a 2nd operation (yellow on the keyboard)
ALPHA	Ⓐ	Enters an uppercase ALPHA character (blue on the keyboard)
alpha	ⓐ	Enters the lowercase version of an ALPHA character (blue on the keyboard)
Full	▦	Accepts no data; maximum characters are entered at a prompt or memory is full

♦ If you press [ALPHA] after [2nd] [INS], the cursor becomes an underlined **A** (A̲).

♦ If you press [2nd] [ALPHA] after [2nd] [INS], the cursor becomes an underlined **a** (a̲).

♦ If you press [2nd] after [2nd] [INS], the insert cursor becomes an underlined ↑ (↑̲).

Cursor Direction Keys



If you hold down [▶], [▼], [◀], or [▲], the cursor continues to move.

Inserting, Deleting, and Clearing Characters

The entry cursor (■) overwrites characters.

[2nd] [INS]	Changes the cursor to the insert cursor (__); inserts characters at the insert cursor and shifts remaining characters right; to cancel insert, press [2nd] [INS] or press [▶], [▼], [◀], or [▲]
[DEL]	Deletes a character at the cursor; to continue deleting to the right, hold down [DEL]
[CLEAR]	Clears the current entry on the home screen; [CLEAR] [CLEAR] clears the entire home screen

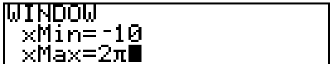
Entering Expressions and Instructions

Entering an Expression

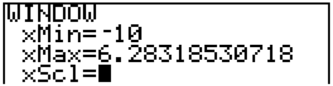
An expression is any combination of numbers and variables that serve as arguments for one or more functions. On the TI-86, you typically enter an expression in the same order as you would write it on paper. For example, πr^2, **5 tan xStat**, and **40(⁻5+3)−(2+3))** are expressions.

You can use an expression on the home screen to calculate an answer.

In most places where a value is required, you can use an expression to enter the value.



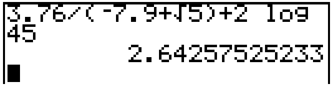
For example, enter an expression as a window variable value (Chapter 5). When you press [▼], [▲], [ENTER], or [EXIT], the TI-86 evaluates the expression and replaces it with the result.



To enter an expression, you enter numbers, variables, and functions from the keyboard and menus (page 31). When you press [ENTER], the expression is evaluated (regardless of the cursor location) according to EOS order-of-evaluation rules (Appendix), and the answer is displayed.

To enter the expression $3.76 \div (\text{-}7.9 + \sqrt{5}) + 2 \log 45$ and then evaluate it, you would press these keys:

3 [.] **76** [÷] [(] [(-)] **7** [.] **9** [+] [2nd] [√] **5** [)] [+] **2** [LOG] **45** [ENTER]



Using Functions in Expressions

A function returns a value. Some examples of functions are **÷** , **‑** , **+** , **√** , and **log**. To use functions, you usually must enter one or more valid arguments.

In this guidebook, optional arguments are shown in brackets ([and]). Do not include these brackets when you enter the arguments.

When this guidebook describes the syntax of a function or instruction, each argument is in italics. For example: **sin** *angle*. Press [SIN] to enter **sin**, and then enter a valid *angle* measurement (or an expression that resolves to *angle*). For functions or instructions with more than one argument, you must separate each argument from the other with a comma.

Some functions require the arguments to be in parentheses. When you are unsure of the evaluation order, use parentheses to clarify a function's place within an expression.

Using an Instruction

The A to Z Reference describes all TI-86 functions and instructions, including their required and optional arguments.

An instruction initiates an action. For example, **ClDrw** is an instruction that, when executed, clears all drawn elements from a graph. You cannot use an instruction in an expression. Generally, the first letter of each instruction name is uppercase on the TI-86. Some instructions take more than one argument, as indicated by an open parenthesis (**(**) at the end of the name. For example, **Circl(** requires three arguments, **Circl(***x*,*y*,*radius***)**.

Entering Functions, Instructions, and Operators

In the CATALOG, *to move to the first item beginning with a letter, press that letter (as in* [L] *in the example).*

You can enter a function, instruction, or operator in any of three ways (**log 45**, for example).

- Paste it to the cursor location from the keyboard or a menu ([LOG] **45**).
- Paste it to the cursor location from the CATALOG ([2nd] [CATLG-VARS] [F1] [L] [F1] [F1] [ENTER] **45**).
- Enter it letter by letter ([2nd] [alpha] [ALPHA] [L] [O] [G] [␣] [ALPHA] [ALPHA] **45**).

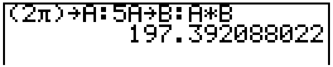
As you can see in the example, using the built-in function or instruction typically is easier.

When you select a function, instruction, or operator, a symbol comprising one or more characters is pasted to the cursor location. Once the symbol is pasted to the cursor location, you can edit individual characters.

For example, assume that you pressed [2nd] [CATLG-VARS] [MORE] [MORE] [F5] [F1] [F1] [ENTER] to paste **yMin** to the cursor location as part of an expression. Then you realized you wanted **xMin**. Instead of pressing nine keys to select **xMin**, you can simply press [◄] [◄] [◄] [◄] [x-VAR].

Entering Consecutive Entries

In the example, the → symbol indicates that the value before it is to be stored to the variable after it (Chapter 2). To paste → to the screen, press [STO►].



To enter two or more expressions or instructions consecutively, separate each from the next with a colon ([2nd] [:]). When you press [ENTER], the TI-86 executes each entry from left to right and displays the result of the last expression or instruction. The entire group entry is stored in last entry (page 28).

The Busy Indicator

When the TI-86 is calculating or graphing, a moving vertical line is displayed as the busy indicator in the top-right corner of the screen. When you pause a graph or a program, the busy indicator is replaced by the pause indicator, a moving vertical dotted line.

Interrupting a Calculation or Graph

To interrupt a calculation or graph in progress, press [ON]. When you interrupt a calculation, the **ERROR 06 BREAK** message and menu are displayed.

- To return to the home screen, select **QUIT** (press [F5]).
- To go to the beginning of the expression, select **GOTO** (press [F1]). Press [ENTER] to recalculate the expression.

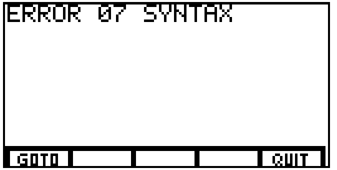
Chapter 5: Function Graphing introduces graphing.

When you interrupt a graph, a partial graph and the GRAPH menu are displayed.

- ♦ To return to the home screen, press [CLEAR] [CLEAR] or any non-graphing key.
- ♦ To restart graphing, select an instruction that displays the graph.

Diagnosing an Error

If a syntax error occurs within a stored equation during program execution, select **GOTO** *to return to the equation editor, not to the program (Chapter 5).*



When the TI-86 detects an error, it returns an error message, such as **ERROR 04 DOMAIN** or **ERROR 07 SYNTAX**. The Appendix describes each error type and possible reasons for the error.

- ♦ If you select **QUIT** (or press [2nd] [QUIT] or [CLEAR] [CLEAR]), the home screen is displayed.
- ♦ If you select **GOTO**, the previous screen is displayed with the cursor on or near the error.

Correcting an Error

1. Note the error type (**ERROR** ## *errorType*).
2. Select **GOTO**, if available. The previous screen is displayed with the cursor on or near the error.
3. Determine the cause for the error. If you cannot, refer to the Appendix for possible causes.
4. Correct the error and continue.

Reusing Previous Entries and the Last Answer

Retrieving the Last Entry

When you press [ENTER] on the home screen to evaluate an expression or to execute an instruction, the entire expression or instruction is placed in a storage area called ENTRY (last entry). When you turn off the TI-86, ENTRY is retained in memory.

To retrieve the last entry, press [2nd] [ENTRY]. The current line is cleared and the entry is pasted to the line.

```
5(2π-√18)+3
            13.2027231003
5(2π-√18)+3█
```

Retrieving and Editing the Last Entry

❶	On the home screen, retrieve the previous entry.	[2nd] [ENTRY]	`5(2π-√18)+3█`
❷	Edit the retrieved entry.	[◄] [◄] [◄] [◄] [◄] **32**	`5(2π-√32█+3`
❸	Re-execute the edited entry.	[ENTER]	`5(2π-√32)+3` `6.13165528844`

Retrieving Previous Entries

The TI-86 retains as many previous entries as possible in ENTRY, up to a capacity of 128 bytes. To scroll from the newest to the older previous entries stored to ENTRY, repeat [2nd] [ENTRY]. If you press [2nd] [ENTRY] after displaying the oldest stored entry, the newest stored entry is displayed again; continuing to press [2nd] [ENTRY] repeats the order.

Retrieving Multiple Entries

Consecutively entered entries separated by colons (page 26) are stored as one entry.

To store two or more expressions or instructions together to ENTRY, enter them on one line, separating each from the other with a colon, and then press [ENTER]. Upon execution, the entire group is stored in ENTRY. The example below shows one of many ways you can manipulate this feature to avoid tedious manual re-entry.

The formula for finding the area of a circle is A=πr^2.

The equation solver (Chapter 15) is another tool with which you can perform this task.

❶ Use trial and error to find the radius of a circle with an area of 200 square centimeters. Store **8** to **r** as your first guess, then execute π**r²**.

8 [STO▸] [2nd] [alpha] [R] [2nd] [:] [2nd] [π] [R] [ALPHA] [ALPHA] [x^2] [ENTER]

```
8→r:πr²
          201.06192983
```

❷ Retrieve **8→r:πr²** and insert **7.958** as a new guess. Continue guessing to approach the answer of **200**.

[2nd] [ENTRY] [2nd] [◂] **7** [2nd] [INS] [.] **958** [ENTER]

```
8→r:πr²
          201.06192983
7.958→r:πr²
          198.956321336
```

Clearing the ENTRY Storage Area

To clear all data from the ENTRY storage area, begin on a blank line on the home screen, select **ClrEnt** from the MEM menu (press [2nd] [MEM] [F5]), and then press [ENTER].

Retrieving the Last Answer

When an expression is evaluated successfully on the home screen or in a program, the TI-86 stores the answer to a built-in variable called **Ans** (last answer). **Ans** may be a real or complex number, list, vector, matrix, or string. When you turn off the TI-86, the value in **Ans** is retained in memory.

To copy the variable name **Ans** to the cursor location, press [2nd] [ANS]. You can use the variable **Ans** anywhere that the value stored to it is valid. When the expression is evaluated, the TI-86 calculates the result using the value stored in **Ans**.

❶	Calculate the area of a garden plot **1.7** meters by **4.2** meters.	**1** [.] **7** [×] **4** [.] **2** [ENTER]
❷	Calculate the yield per square meter if the plot produces a total of **147** tomatoes.	**147** [÷] [2nd] [ANS] [ENTER]

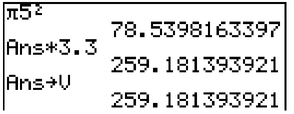
Using Ans Preceding a Function

Previous answers are stored to **Ans**. If you begin an expression by entering a function that requires a preceding argument, the TI-86 automatically enters **Ans** as the argument.

❶	Enter and execute an expression.	**5** [÷] **2** [ENTER]
❷	Enter a function without an argument. **Ans** is pasted to the screen, followed by the function.	[×] **9** [.] **9** [ENTER]

Storing Results to a Variable

❶	Calculate the area of a circle with radius **5** meters.	[2nd] [π] **5** [x^2] [ENTER]
❷	Calculate the volume of a cylinder of radius **5** meters and height **3.3** meters.	[×] **3** [.] **3** [ENTER]
❸	Store the result to the variable **V**.	[STO▶] [V] [ENTER]



Using TI-86 Menus

The symbols for many TI-86 features are found in menus instead of on the TI-86 keyboard.

Displaying a Menu

The way to display a particular menu depends on the menu's location on the TI-86.

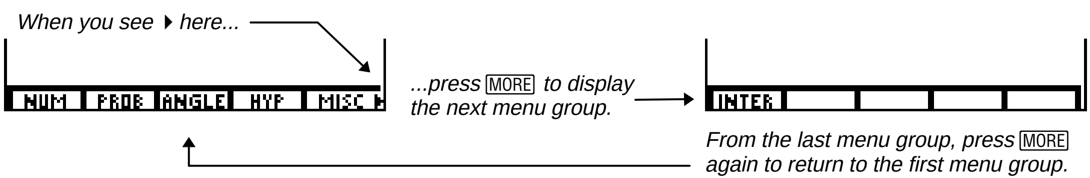
Menu-Displaying Method	Example
Press a key that has a menu name on it	[GRAPH] displays the GRAPH menu
Press [2nd] and then a 2nd-key menu name	[2nd] [MATH] displays the MATH menu
Select a menu name from another menu	[2nd] [MATH] [F1] displays the MATH NUM menu
Select an editor or selection screen	[2nd] [LIST] [F4] displays the list editor menu with the list editor
Accidentally commit an error	**1** [STO▸] [ENTER] displays the error menu

Some TI-86 menus have as many as 25 items.

When you display a menu, a menu group of one to five items is displayed on the bottom of the screen. If the more symbol (▸) is displayed after the fifth item in a menu group, the menu continues for at least one more menu group. To view the next menu group, press [MORE]. The last menu group of one to five items does not have a ▸ symbol.

For example, press [2nd] [MATH] to display the MATH menu.

▸, ▾, ◂, and ▴ do not work on menus.



The Menu Keys

[2nd] *upper menu keys* → M1 M2 M3 M4 M5

lower menu keys → [F1] [F2] [F3] [F4] [F5]

[2nd] [QUIT] *clears all menus* → QUIT

[2nd] [M1] *through* [M5] *selects upper menu items* → [2nd] [EXIT] [MORE]

[MORE] *scrolls lower menu groups*

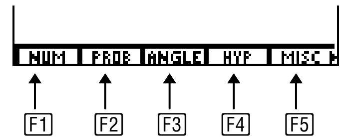
[EXIT] *removes the lower menu*

Selecting a Menu Item

The Appendix Menu Map shows every TI-86 menu.

Typically, a TI-86 menu item is five characters long or less.

When you display a menu, one to five items are displayed. To select a menu item, press the menu selection key directly below the item. For example, in the MATH menu to the right, press [F1] to select **NUM**, press [F2] to select **PROB**, and so on.

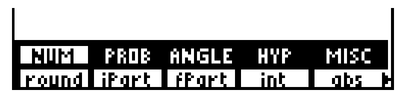


When you select a menu item that displays another menu, the first menu moves up one line on the screen to make room for the new menu. All items on the original menu are displayed in reverse type, except the item you selected.

[MORE] only scrolls the lower menu; it does not scroll the upper menu.

When you select **NUM***...*

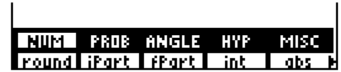
...the MATH *menu moves up and the* MATH NUM *menu is displayed.*



To remove the MATH NUM *menu and move the* MATH *menu down, press* [EXIT].

To select an item from the upper menu, press [2nd] and the appropriate key [M1] through [M5].

To select **PROB** *from the upper menu, press* [2nd] [M2].
To select **iPart** *from the lower menu, press* [F2].



When an editor menu is displayed as the upper menu, and you select an item from the lower menu that displays yet another menu, the editor menu remains as the upper menu.

When you select **NUM** *from the lower menu...*

Upper: equation editor menu
Lower: MATH *menu*

...the equation editor menu remains and the MATH NUM *menu is displayed.*

The MATH *menu disappears.*

Upper: equation editor menu
Lower: MATH NUM *menu*

Exiting (Removing) a Menu

To remove a menu from the bottom of a graph screen, press [CLEAR] *after plotting the graph (Chapter 5).*

To remove the lower menu from the screen, press [EXIT].

When you press [EXIT]*...*

Upper: MATH *menu*
Lower: MATH NUM *menu*

...the MATH NUM *menu disappears and the* MATH *menu moves down.*

Lower: MATH *menu*

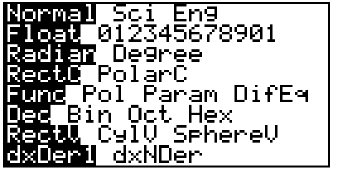
Press [EXIT] *again, and the* MATH *menu disappears.*

No menu

Viewing and Changing Modes

In the screen to the right, the default mode settings are highlighted along the left side of the screen.

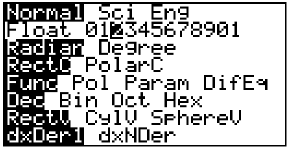
To display the mode settings, press [2nd] [MODE]. The current settings are highlighted. Mode settings control how the TI-86 displays and interprets numbers and graphs. The Constant Memory feature retains current mode settings when the TI-86 is turned off. All numbers, including elements of matrices and lists, are displayed according to the mode settings.



Changing a Mode Setting

This example changes the decimal mode setting to 2, as in U.S. dollars and cents.

1. Move the cursor to the line of the setting that you want to change (decimal setting in the example). [▼]
2. Move the cursor to the setting you want (2 decimal places). [▶] [▶] [▶]
3. Execute the change. [ENTER]



Notation Modes

*In **Normal** notation, if the answer is more than 12 digits or the absolute value of the answer < .001, it is displayed in scientific notation.*

Notation modes do not affect how you enter numbers.

Normal Displays results with digits to the left and right of the decimal (as in **123456.7**)

Sci (scientific) Displays results in two parts: significant digits (with one digit to the left of the decimal) are displayed to the left of **E** and the appropriate power of 10 is displayed to the right of **E** (as in **1.234567E5**)

Eng (engineering) Displays results in two parts: significant digits (with one, two, or three digits to the left of the decimal) are displayed to the left of **E** and the appropriate power of 10 (which is always a multiple of 3) is displayed to the right of **E** (as in **123.4567E3**)

Decimal Modes

Float (floating) Displays results up to 12 digits, plus any sign and the floating decimal point

(fixed) (**012345678901**; each number is a setting) Displays results with the specified number of digits to the right of the decimal point (rounds answers to the specified decimal place); the second **0** sets 10; the second **1** sets 11

Angle Modes

Radian Interprets angle values as radians; displays answers in radians

Degree Interprets angle values as degrees; displays answers in degrees

Complex Number Modes

RectC (rectangular complex mode) Displays complex-number results as **(***real*,*imaginary***)**

PolarC (polar complex mode) Displays complex-number results as **(***magnitude*∠*angle***)**

Graphing Modes

Func (function graphing) Plots functions where **y** is a function of **x**

Pol (polar graphing) Plots functions where **r** is a function of θ

Param (parametric graphing) Plots relations where **x** and **y** are functions of **t**

DifEq (differential equation graphing) Plots differential equations in terms of **t**

Number Base Modes

Non-decimal modes are valid only on the home screen or in the program editor.

Dec (decimal number base) Interprets and displays numbers as decimal (base 10)

Bin (binary number base) Interprets numbers as binary (base 2); displays b suffix with answers

Oct (octal number base) Interprets numbers as octal (base 8); displays o suffix with answers

Hex (hexadecimal number base) Interprets numbers as hexadecimal (base 16); displays h suffix with answers

Vector Coordinate Modes

Vector modes do not affect how you enter vectors.

RectV (rectangular vector coordinates) Displays answers in the form $[x\ y]$ for two-element vectors and $[x\ y\ z]$ for three-element vectors

CylV (cylindrical vector coordinates) Displays results in the form $[r\ \angle\theta]$ for two-element vectors and $[r\ \angle\theta\ z]$ for three-element vectors

SphereV (spherical vector coordinates) Displays results in the form $[\mathrm{r}\ \angle\ \theta]$ for two-element vectors and $[r\ \angle\theta\ \angle\phi]$ for three-element vectors

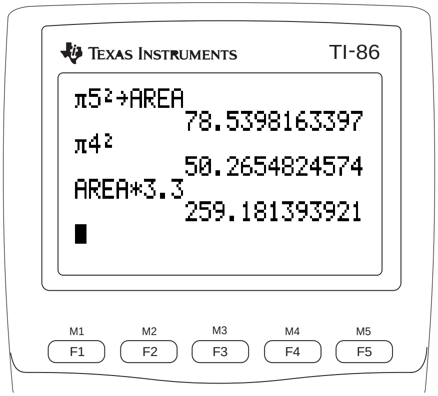
Differentiation Modes

dxDer1 (exact differentiation) Uses **der1** (Chapter 3) to differentiate exactly and calculate the value for each function in an expression (**dxDer1** is more accurate than **dxNDer**, but it restricts the kinds of functions that are valid in the expression)

The value stored to δ affects **dxNDer** *(Appendix).*

dxNDer (numeric differentiation) Uses **nDer** to differentiate numerically and calculate the value for an expression (**dxNDer** is less accurate than **dxDer1**, but more kinds of functions are valid in the expression)

2 The CATALOG, Variables, and Characters



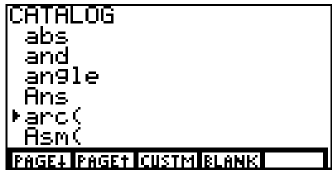
The CATALOG [2nd] [CATLG-VARS] [F1]

The CATALOG *is the first item on the* CATLG-VARS *menu.*

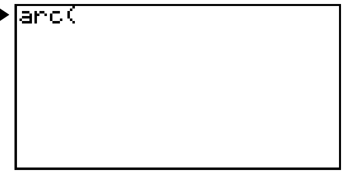
The CATALOG displays all TI-86 functions and instructions in alphabetical order. Items that do not begin with a letter (such as **+** or **▸Bin**) are at the end of the CATALOG.

The selection cursor (▸) indicates the current item. To select an item from the CATALOG, move the selection cursor to the item and press [ENTER]. The CATALOG disappears and the name is pasted to the previous cursor location.

Use [▼] or [▲] to move ▸ to an item...



...and press [ENTER]. The item is pasted to the cursor location.



To jump...	Do this:
To the first item beginning with a particular letter	Press the letter; ALPHA-lock is on
To special characters at the end of the CATALOG	Press [▲] from the first CATALOG item
Down one whole screen	Select **PAGE↓** from the CATALOG menu ([F1])
Up one whole screen	Select **PAGE↑** from the CATALOG menu ([F2])

The menu items **CUSTM** and **BLANK** are on the CATALOG menu and each VARS screen menu. With them, you can create and edit your own CUSTOM menu of up to 15 CATALOG items and variables, including program names. For details about the CUSTOM menu, read page 44.

This chapter describes the first two data storage methods listed here. The other methods are described in the appropriate chapters.

Storing Data to Variables

On the TI-86, data can be stored to variables in several ways. You can:

- Use [STO▸] to store a value to a variable.
- Use **=** to store an unevaluated expression to an equation variable.
- Use an editor's **Name=** prompt to store several types of data to a variable.
- Change TI-86 settings or reset defaults and memory to the factory settings.
- Execute functions that cause the TI-86 to store data automatically to built-in variables.

The TI-86 has built-in variable names with specific purposes, such as equation variables, list names, statistical result variables, window variables, and **Ans**. You can store values to some of them. They are introduced in the appropriate chapters of this guidebook.

Creating a Variable Name

You can create your own variable name when you use [STO▸], **=**, or a **Name=** prompt to store data. When you create a user-created variable name, follow these guidelines.

- The user-created variable name can be from one to eight characters long.
- The first character must be a letter, which includes all CHAR GREEK menu items, as well as Ñ, ñ, Ç, and ç from the CHAR MISC menu.
- A user-created variable name cannot replicate a TI-86 feature symbol or built-in variable. For example, you cannot create **abs**, because **abs** is the absolute value function symbol. You cannot create **Ans**, because it is already a built-in variable name.
- The TI-86 distinguishes between uppercase and lowercase characters in variable names. For example, **ANS**, **Ans**, and **ans** are three different variable names. Therefore, only **Ans** is a built-in variable name; **ANS** and **ans** can be user-created variable names.

Storing a Value to a Variable Name

❶	Enter a value, which can be an expression.	[2nd] [π] 5 [x^2]	
❷	Enter → (the store symbol) next to the value.	[STO▸]	
❸	Create a variable name one to eight characters long, starting with a letter. ALPHA-lock is on.	[A] [R] [E] [A]	
❹	Store the value to the variable. The value stored to the variable is displayed as a result.	[ENTER]	

Storing an Unevaluated Expression

When you store an expression to memory using [STO▸] (with the → sign), the expression is evaluated and the result is stored to a variable.

When you store an unevaluated expression using [ALPHA] [=], or the equation editor (Chapter 5), or the equation solver (Chapter 15), the unevaluated expression is stored to an equation variable.

When you use =, variable is first, then =, then expression. In contrast, when you use →, value is first, then →, then variable.

To store an unevaluated expression on the home screen or in a program, the syntax is:

variable=*expression*

where *variable* always precedes the equals sign and *expression* always follows the equals sign.

You can use **=** to store a mathematical expression to an equation variable. For example, **F=M∗A**.

Storing an Answer

To store an answer to a variable before you evaluate another expression, use [STO▶] and **Ans**.

In the example, the TI-86 multiplies the value stored to **AREA** *times* **3.3**.

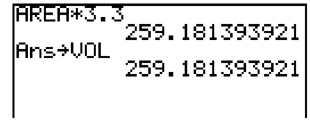
To paste **AREA** *to the cursor location, you can press* [2nd] [CATLG-VARS] [F3], *move the selection cursor (▶) to* **AREA**, *and press* [ENTER].

❶ Enter and evaluate an expression.

[ALPHA] [ALPHA] [A] [R] [E] [A] [ALPHA] [×] **3** [.] **3** [ENTER]

❷ Store the answer to a user-created variable or to a valid built-in variable. The value stored to the variable is displayed as a result.

[STO▶] [V] [O] [L] [ENTER]



Copying a Variable Value

To paste **→** *to the cursor location, press* [STO▶].

To copy the contents of *variableA* into *variableB*, the syntax is:

*variableA***→***variableB*

For example, **RegEq→y1** stores the regression equation (Chapter 14) to the variable **y1**.

Displaying a Variable Value

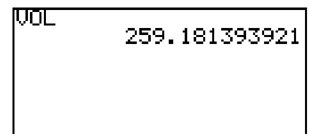
To paste a variable name, you can select it from a VARS *menu (page 42).*

❶ With the cursor on a blank line on the home screen, paste the variable name to the cursor location, as described above.

[2nd] [CATLG-VARS] [F3] [▼] (location may vary) [ENTER]

❷ Display the contents of the variable.

[ENTER]



You also can display variables containing some data types by displaying them in the appropriate editor (such as the list editor or window variable editor) or graph. These methods are detailed in subsequent chapters of this guidebook.

Recalling a Variable Value

To cancel RCL, *press* [CLEAR].

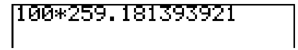
Editing a recalled value does not change the value stored to the variable.

1. Move the cursor to where you want to insert the recalled variable value. — **100** [×]

2. Display the **Rcl** prompt at the bottom of the screen. ALPHA-lock is on. — [2nd] [RCL]

3. Enter the variable name you want to recall. — [V] [O] [L]

4. Recall the variable contents to the cursor location. The **Rcl** prompt disappears and the entry cursor returns. — [ENTER]



Classifying Variables as Data Types

When you store data in an editor, the TI-86 recognizes the data type according to the editor. For example, only vectors are stored using the vector editor.

The TI-86 classifies variables according to data type and places each variable on a data-type selection screen. You can display each screen by selecting the appropriate data type from the CATLG-VARS menu, as described on page 43. Here are some examples.

If data...	The TI-86 classifies the data type as...	For example:
Begins with **{** and ends with **}**	A list (VARS LIST screen)	**{1,2,3}**
Begins with [and ends with]	A vector (VARS VECTR screen)	[**1,2,3**]
Begins with [[and ends with]]	A matrix (VARS MATRX screen)	[[**1,2,3**][**4,5,6**][**7,8,9**]]

To display additional menu groups, press [MORE].

The list names **fStat**, **xStat**, *and* **yStat** *are statistical result variables on the* VARS STAT *screen.*

The CATLG-VARS (CATALOG-Variables) Menu

[2nd] [CATLG-VARS]

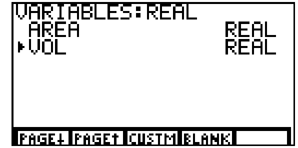
CATLG	ALL	REAL	CPLX	LIST	▸	VECTR	MATRX	STRNG	EQU	CONS
					▸	PRGM	GDB	PIC	STAT	WIND

CATLG	Displays the CATALOG
ALL	Displays a selection screen with all variables and names of all data types
REAL	Displays a selection screen with all real number variables
CPLX	Displays a selection screen with all complex number variables
LIST	Displays a selection screen with all list names
VECTR	Displays a selection screen with all vector names
MATRX	Displays a selection screen with all matrix names
STRNG	Displays a selection screen with all string variables
EQU	Displays a selection screen with all equation variables
CONS	Displays a selection screen with all user-defined constants
PRGM	Displays a selection screen with all program names
GDB	Displays a selection screen with all graph database names
PIC	Displays a selection screen with all picture names
STAT	Displays a selection screen with all statistical result variables
WIND	Displays a selection screen with all window variables

Selecting a Variable Name

The example assumes that the real-number variables **AREA** *and* **VOL** *from the example on page 41 have not been deleted from memory.*

1. Select the appropriate data-type selection screen from the CATLG-VARS menu. [2nd] [CATLG-VARS] [F3]
2. Move the cursor to the variable you want to select. [▼]
3. Select the variable you want. [ENTER]



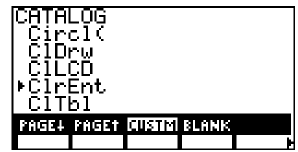
The CUSTOM Menu [2nd] [CATLG-VARS] [F1] [F3]

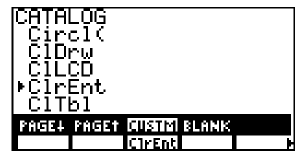
You can select up to 15 items from the CATALOG and VARS screens – program names, functions, instructions, and other items – to create your own CUSTOM menu. To display your CUSTOM menu, press [CUSTOM]. Use [F1] through [F5] and [MORE] to select items like any other menu.

Entering CUSTOM Menu Items

When copying items into the CUSTOM *menu, you can skip menu cells and menu groups.*

1. Select **CUSTM** from the CATALOG. The CUSTOM menu is displayed. ALPHA-lock is on. [2nd] [CATLG-VARS] [F1] [F3]
2. Move the selection cursor (▸) to the item you want to copy to the CUSTOM menu. [C] [▼] [▼] [▼]
3. Copy the item to the CUSTOM menu cell you select, replacing any previous item. [F3]
4. To enter more items, repeat steps 2 and 3 using different items and cells.
5. Display the CUSTOM menu. [2nd] [QUIT] [CUSTOM]





Clearing CUSTOM Menu Items

To clear an item from the second or third menu group, press [MORE] until the item is displayed, and then select it.

1. Select **BLANK** from the CATALOG menu. The CUSTOM BLANK menu is displayed. [2nd] [CATLG-VARS] [F1] [F4]

2. Clear the menu item. [F3]

3. To clear more items, repeat steps 2 and 3.

Deleting a Variable from Memory

You cannot delete a TI-86 built-in variable.

You cannot delete a program variable using **DelVar(** .

From the home screen or in a program, to delete from memory one user-created variable name (except a program name) and its contents, the syntax is:

DelVar(*variable***)**

To delete user-created variable names and their contents (including program names), display the MEM DELET menu ([2nd] [MEM] [F2]), select the data type, select the variable, and then press [ENTER] (Chapter 16). Deleting a variable does not remove it from the CUSTOM menu (page 44).

The CHAR (Character) Menu [2nd] [CHAR]

MISC	GREEK	INTL		
miscellaneous characters menu	Greek characters menu	international characters menu		

Ñ, ñ, Ç, and ç are valid as any character of a variable name, including the first letter.

%, ' , and ! can be functions.

All CHAR GREEK menu items are valid variable-name characters, including the first letter. π ([2nd] [π]) is not valid as a character; π is a constant on the TI-86.

The CHAR MISC (Miscellaneous) Menu [2nd] [CHAR] [F1]

MISC	GREEK	INTL		
?	#	&	%	'

▶

!	@	$	~	|

▶

¿	Ñ	ñ	Ç	ç

The CHAR GREEK Menu [2nd] [CHAR] [F2]

MISC	GREEK	INTL		
α	β	γ	Δ	δ

▶

ε	θ	λ	μ	ρ

▶

Σ	σ	τ	ϕ	Ω

The CHAR INTL (International) Menu [2nd] [CHAR] [F3]

MISC	GREEK	INTL		
´	`	^	¨	

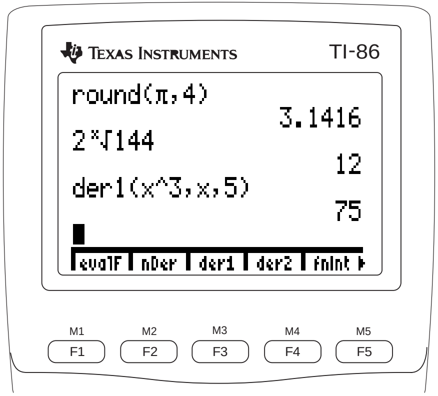
You can combine modifiers on the CHAR INTL menu with uppercase or lowercase vowels to create vowels used in some languages. You can use these vowels in variable names and text.

Adding a Modifier to a Vowel

❶ Select the modifier from the CHAR INTL menu. ALPHA-lock is on. If necessary, switch to alpha-lock. [2nd] [CHAR] [F3] [F4] [2nd] [alpha]

❷ Enter the uppercase or lowercase vowel over which you want the modifier. [O]

3 Math, Calculus, and Test Operations



Keyboard Mathematical Functions

The A to Z Reference details which data types are valid arguments for each function.

You can use these mathematical functions in expressions with real or complex values. You can use some of them with lists, vectors, matrices, or strings.

When you use lists, vectors, or matrices, the valid functions return a list of results calculated on an element-by-element basis. If you use two lists, vectors, or matrices in the same expression, they must be equal in dimension.

The most common mathematical functions are on the TI-86 keyboard. For syntax, details, and examples of these functions, refer to the A to Z Reference.

***x*$^{-1}$** *(the multiplicative inverse) is equivalent to the reciprocal, 1/x.*

Key	Function	Key	Function
[+]	**+** (add)	[SIN]	**sin** (sine)
[−]	**−** (subtract)	[COS]	**cos** (cosine)
[×]	***** (multiply)	[TAN]	**tan** (tangent)
[÷]	**/** (divide)	[2nd] [SIN⁻¹]	**sin^{-1}** (arcsine; inverse of sine)
[(−)]	**-** (negate)	[2nd] [COS⁻¹]	**cos^{-1}** (arccosine; inverse of cosine)
[x^2]	**^{2}** (square)	[2nd] [TAN⁻¹]	**tan^{-1}** (arctangent; inverse of tangent)
[2nd] [√]	√ (square root)	[LOG]	**log** (logarithm)
[2nd] [x^{-1}]	**$^{-1}$** (inverse)	[LN]	**ln** (natural log)
[^]	**^** (raise to a specified power)	[2nd] [e^x]	**e**^{x} (constant e raised to a power)
[2nd] [10^x]	**10^** (10 to a specified power)	[2nd] [π]	π (constant pi; 3.1415926535898)
[EE]	E (exponent)		

The MATH Menu 2nd [MATH]

NUM	PROB	ANGLE	HYP	MISC	▸	INTER				

- NUM: number menu
- PROB: probability menu
- ANGLE: angle menu
- HYP: hyperbolic menu
- MISC: miscellaneous math functions menu
- INTER: interpolate editor

The MATH NUM (Number) Menu 2nd [MATH] F1

NUM	PROB	ANGLE	HYP	MISC						
round	iPart	fPart	int	abs	▸	sign	min	max	mod	

value can sometimes be an expression, list, vector, or matrix. For details about specific syntax options and examples, refer to the A to Z Reference.

round(*value*[**,***#ofDecimals*]**)**	Rounds *value* to 12 decimal places or to *#ofDecimals*
iPart *value*	Returns the integer part or parts of *value*
fPart *value*	Returns the fractional part or parts of *value*
int *value*	Returns the largest integer less than or equal to *value*
abs *value*	Returns the absolute value or magnitude of *value*
sign *value*	Returns **1** if *value* is positive; **0** if *value* is **0**; **⁻1** if *value* is negative
min(*valueA***,***valueB***)**	Returns the smaller of *valueA* and *valueB*
min(*list***)**	Returns the smallest element of *list*
max(*valueA***,***valueB***)**	Returns the larger of *valueA* and *valueB*
max(*list***)**	Returns the largest element of *list*
mod(*numberA***,***numberB***)**	Returns *numberA* modulo *numberB*

The MATH PROB (Probability) Menu 2nd [MATH] F2

NUM	PROB	ANGLE	HYP	MISC
!	nPr	nCr	rand	randIn

▸

randN	randBi			

! *(factorial) is valid for non-integers.*

randInt, **randNorm**, *and* **randBin** *are abbreviated in the* MATH PROB *menu.*

*value***!**	Returns the factorial of a real *value*
items **nPr** *number*	Returns the number of permutations of *items* (**n**) taken *number* (**r**) at a time
items **nCr** *number*	Returns the number of combinations of *items* (**n**) taken *number* (**r**) at a time
rand	Returns a random number > 0 and < 1; to control a random number sequence, first store an integer seed value to **rand** (such as **0→rand**)
randInt(*lower***,***upper* [**,***#ofTrials*]**)**	(random integer) Returns a random integer bound by the specified integers, *lower* ≤ integer ≤ *upper*; to return a list of random integers, specify an integer > 1 for *#ofTrials*
randNorm(*mean***,** *stdDeviation* [**,***#ofTrials*]**)**	(random normal) Returns a random real number from a normal distribution specified by *mean* and *stdDeviation*; to return a list of random numbers, specify an integer > 1 for *#ofTrials*
randBin(*#ofTrials***,** *probabilityOfSuccess* [**,***#ofSimulations*]**)**	(random binomial) Returns a random real number from a binomial distribution, where *#ofTrials* ≥ 1 and 0 ≤ *probabilityOfSuccess* ≤ 1; to return a list of random numbers, specify an integer > 1 for *#ofSimulations*

The MATH ANGLE Menu 2nd [MATH] F3

NUM	PROB	ANGLE	HYP	MISC
o	r	'	▸DMS	

angle can be a list for ° , ʳ , and ▸**DMS**.

In a calculation, the result of a degrees'minutes'seconds' entry is treated as degrees in **Degree** *angle mode only. It is treated as radians in* **Radian** *angle mode.*

angle°	Overrides current angle mode setting to express *angle* in degrees
*angle*ʳ	Overrides current angle mode setting to express *angle* in radians
degrees'*minutes*'*seconds*'	Designates an angle as *degrees*, *minutes*, and *seconds*
angle▸**DMS**	Displays *angle* in degrees°minutes'seconds" format, even though you use *degrees*'*minutes*'*seconds*' to enter a DMS angle

The MATH HYP (Hyperbolic) Menu 2nd [MATH] F4

NUM	PROB	ANGLE	HYP	MISC
sinh	cosh	tanh	$\sinh^{-1}$	$\cosh^{-1}$

▸

$\tanh^{-1}$				

value can sometimes be an expression, list, vector, or matrix. For details about specific syntax options and examples, refer to the A to Z Reference.

sinh *value*	Returns the hyperbolic sine of *value*
cosh *value*	Returns the hyperbolic cosine of *value*
tanh *value*	Returns the hyperbolic tangent of *value*
sinh^{-1} *value*	Returns the hyperbolic arcsine of *value*
cosh^{-1} *value*	Returns the hyperbolic arccosine of *value*
tanh^{-1} *value*	Returns the hyperbolic arctangent of *value*

The MATH MISC (Miscellaneous) Menu [2nd] [MATH] [F5]

NUM	PROB	ANGLE	HYP	MISC						
sum	prod	seq	lcm	gcd	▸	▸Frac	%	pEval	$^x\sqrt{}$	eval

value can sometimes be an expression, list, vector, or matrix. For details about specific syntax options, refer to the A to Z Reference.

sum *list*	Returns the sum of the elements of *list*
prod *list*	Returns the product of the elements of *list*
seq(*expression*,*variable*, *begin*,*end*[,*step*]**)**	Returns a list in which each element is the value of *expression* evaluated for *variable* from *begin* to *end* by *step*
lcm(*valueA*,*valueB***)**	Returns the least common multiple of *valueA* and *valueB*
gcd(*valueA*,*valueB***)**	Returns the greatest common divisor of *valueA* and *valueB*
value▸**Frac**	Displays *value* as a fraction
*value***%**	Returns *value* divided by 100 (multiplied by .01)
*percent***%***number*	Returns *percent* of *number*
pEval(*coefficientList*,*xValue***)**	Returns the value of a polynomial (whose coefficients are given in *coefficientList*) at *xValue*
$x^{th}root{}^{x}\sqrt{}value$	Returns the $x^{th}root$ of *value*
eval *value*	Returns a list of the values of all selected functions in the current graphing mode for the real *value* of the independent variable

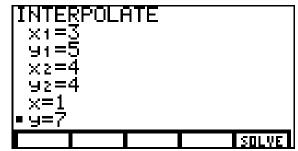
The Interpolate/Extrapolate Editor [2nd] [MATH] [MORE] [F1]

Using the interpolate/extrapolate editor, you can interpolate or extrapolate a value linearly, given two known pairs and the x-value or y-value of the unknown pair.

To interpolate y from the home screen, select **inter(** *from the* CATALOG*, and then enter* **inter(**$x1$,$y1$,$x2$,$y2$,x**)**.

To interpolate x from the home screen, enter **inter(**$y1$,$x1$,$y2$,$x2$,y**)**.

❶	Display the interpolate/extrapolate editor.	[2nd] [MATH] [MORE] [F1]
❷	Enter real values for the first known pair ($x1$,$y1$). The values can be expressions.	**3** [ENTER] **5** [ENTER]
❸	Enter values for the second known pair ($x2$,$y2$).	**4** [ENTER] **4** [ENTER]
❹	Enter a value for either the **x** value or the **y** value of the unknown pair.	**1** [ENTER]
❺	If necessary, move the cursor to the value for which you want to solve (**x** or **y**).	[▲] or [▼]
❻	Select **SOLVE**.	[F5]



You can store individual values with the [STO▶] *key (Chapter 2).*

The result is interpolated or extrapolated and displayed; the variables **x** and **y** are not changed. A solid square in the first column indicates the interpolated or extrapolated value.

After solving for a value, you can continue to use the interpolate/extrapolate editor.

You must set **Dec** *mode to use the calculus functions.*

The CALC (Calculus) Menu 2nd [CALC]

evalF	nDer	der1	der2	fnInt	▸	fMin	fMax	arc		

The calculus functions return values with respect to any user-created variable, to built-in variables **eqn** and **exp**, and to graphing variables such as **x**, **t**, and θ.

For **evalF**, **nDer**, **der1**, *and* **der2**, *variable can be a real or complex number or list.*

You can use **der1** *and* **der2** *in expression. You can use* **nDer** *once in expression.*

For **fnInt**, **fMin**, *and* **fMax**, *lower < upper must be true.*

evalF(*expression***,***variable***,***value***)**	Returns the value of *expression* with respect to *variable* for a given variable *value*
nDer(*expression***,***variable* [**,***value*]**)**	Returns an approximate numerical derivative of *expression* with respect to *variable* for the current variable value or specified variable *value*
der1(*expression***,***variable*[**,***value*]**)**	Returns the value of the first derivative of *expression* with respect to *variable* for the current variable value or specified variable *value*
der2(*expression***,***variable*[**,***value*]**)**	Returns the value of the second derivative of *expression* with respect to *variable* for the current variable value or specified variable *value*
fnInt(*expression***,***variable***,** *lower***,***upper***)**	Returns the numerical integral of *expression* with respect to *variable* between *lower* and *upper* boundaries
fMin(*expression***,***variable***,** *lower***,***upper***)**	Returns the minimum value of *expression* with respect to *variable* between *lower* and *upper* boundaries
fMax(*expression***,***variable***,** *lower***,***upper***)**	Returns the maximum value of *expression* with respect to *variable* between *lower* and *upper* boundaries
arc(*expression***,***variable***,** *start***,***end***)**	Returns the length of a segment of a curve defined by *expression* with respect to *variable* between *start* and *end*

The built-in variable δ defines the step size in calculating **nDer(** (in **dxNDer** differentiation mode only) and **arc(**. The built-in variable **tol** defines the tolerance in calculating **fnInt(**, **fMin(**, **fMax(**, and **arc(**. The value of each must be >0. These factors affect the accuracy of the calculations. As δ becomes smaller, the approximation typically is more accurate. For example, **nDer(A^3,A,5)** returns **75.0001** if δ**=.01**, but returns **75** if δ**=.0001** (Appendix).

The function integral error value is stored to the variable **fnIntErr** (Appendix).

For **arc(** and **fnInt(** while **dxDer1** mode is set, these functions are not valid in *expression*: **evalF(**, **der1(**, **der2(**, **fMin(**, **fMax(**, **nDer(**, **seq(**, and any equation variable, such as **y1**.

You can approximate the fourth derivative at the current value of x with this formula: **nDer(nDer(der2(x^4,x),x),x)**.

The TEST (Relational) Menu 2nd [TEST]

==	<	>	≤	≥	▸	≠				

Relational functions are valid for two lists of the same length. When valueA and valueB are lists, a list of results calculated element by element is returned.

valueA==*valueB*	(equal to) Returns **1** if *valueA* is equal to *valueB*; returns **0** if not equal; *valueA* and *valueB* can be real or complex numbers, lists, vectors, matrices, or strings
valueA<*valueB*	(less than) Returns **1** if *valueA* is less than *valueB*; returns **0** if *valueA* is not less than *valueB*; *valueA* and *valueB* must be real numbers or lists
valueA>*valueB*	(greater than) Returns **1** if *valueA* is greater than *valueB*; returns **0** if *valueA* is not greater than *valueB*; *valueA* and *valueB* must be real numbers or lists
valueA ≤ *valueB*	(less than or equal to) Returns **1** if *valueA* is less than or equal to *valueB*; returns **0** if *valueA* is not less than or equal to *valueB*; *valueA* and *valueB* must be real numbers or lists

valueA≥*valueB*	(greater than or equal to) Returns **1** if *valueA* is greater than or equal to *valueB*; returns **0** if *valueA* is not greater than or equal to *valueB*; *valueA* and *valueB* must be real numbers or lists
valueA≠*valueB*	(not equal to) Returns **1** if *valueA* is not equal to *valueB*; returns **0** if *valueA* is equal to *valueB*; *valueA* and *valueB* can be real or complex numbers, lists, vectors, matrices, or strings

Using Tests in Expressions and Instructions

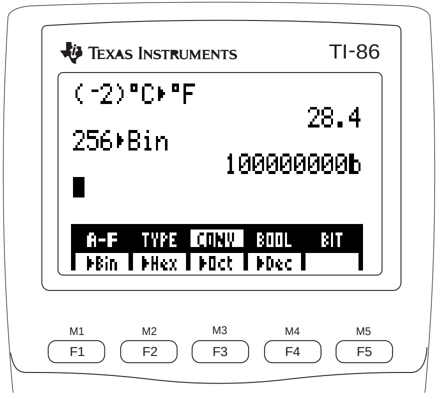
You can use relational functions to control program flow (Chapter 16).

The TI-86 Evaluation Operating System (Appendix) performs all operations except Boolean operators before it performs relational functions. For example:

♦ The expression **2+2==2+3** evaluates to **0**. The TI-86 performs the addition first, and then compares 4 to 5.

♦ The expression **2+(2==2)+3** evaluates to **6**. The TI-86 performs the test in parentheses first, and then adds 2, 1, and 3.

4 Constants, Conversions, Bases, and Complex Numbers



Using Built-In and User-Created Constants

A constant is a variable with a specific value stored to it. The CONS BLTIN menu items are common constants built into the TI-86. You cannot edit the value of a built-in constant.

You can create your own constants and add them to the user-created constant menu for easy access. To enter a user-created constant, you must use the user-created constant editor (page 60); you cannot use [STO▸] or **=** to create a constant.

The CONS (Constants) Menu [2nd] [CONS]

BLTIN	EDIT	USER		
built-in constants menu	user-created constants editor	user-created constants menu		

The CONS BLTIN (Built-In Constants) Menu [2nd] [CONS] [F1]

You can select built-in constants from the CONS BLTIN *menu or enter them using the keyboard and the* CHAR GREEK *menu.*

BLTIN	EDIT	USER		
Na	k	Cc	ec	Rc

▸

Gc	g	Me	Mp	Mn

▸

μ0	ε0	h	c	u

Built-In Constant	Constant Name	Constant Value
Na	Avogadro's number	6.0221367**E**23 $mole^{-1}$
k	Boltzman's constant	1.380658**E**-23 J/K
Cc	Coulomb constant	8.9875517873682**E**9 N m^2/C^2
ec	Electron charge	1.60217733**E**-19 C
Rc	Gas constant	8.31451 J/mole K
Gc	Gravitational constant	6.67259**E**-11 N m^2/kg^2
g	Earth acceleration due to gravity	9.80665 m/sec^2
Me	Mass of an electron	9.1093897**E**-31 kg
Mp	Mass of a proton	1.6726231**E**-27 kg
Mn	Mass of a neutron	1.6749286**E**-27 kg
μ**0**	Permeability of a vacuum	1.2566370614359**E**-6 N/A^2
ε**0**	Permittivity of a vacuum	8.8541878176204**E**-12 F/m
h	Planck's constant	6.6260755**E**-34 J sec
c	Speed of light in a vacuum	299,792,458 m/sec
u	Atomic mass unit	1.6605402**E**-27 kg
π	Pi	3.1415926535898
e	Base of natural log	2.718281828459

To use π*, press* [2nd] [π] *or select it from the* CATALOG*.*
To use **e^***, press* [2nd] [e^x]*.*
To use **e***, press* [2nd] [alpha] [E]*.*

Creating or Redefining a User-Created Constant

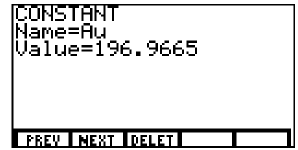
CONS USER menu items are the names of all stored user-created constants, arranged alphanumerically.

1. Display the CONS menu. — [2nd] [CONS]

2. Display the constant editor. The **Name=** prompt, **Value=** prompt, and CONS USER menu are displayed. ALPHA-lock is on. — [F2]

196.9665 is the atomic weight of gold (Au).

3. Enter a constant name. Either enter a new name one to eight characters long, starting with a letter, or select a name from the CONS USER menu. The cursor moves to the **Value=** prompt and the CONS EDIT menu is displayed (see below). — [A] [2nd] [alpha] [U] [ENTER]



You can enter a value later.

4. Enter the real or complex constant value, which can be an expression. The value is stored to the constant as you enter it. The user-created constant becomes a CONS USER menu item. — **196** [.] **9665**

If you select **PREV** *when the first constant name is displayed, or* **NEXT** *when the last constant name is displayed, the* CONS USER *menu replaces the* CONS EDIT *menu.*

You also can delete a constant from the MEM DELET CONS *screen.*

The Constant Editor Menu [2nd] [CONS] [F2] **name** [ENTER] **or** [▼]

PREV	NEXT	DELET		

PREV Displays the name and value (if any) of the previous constant on the CONS USER menu

NEXT Displays the name and value (if any) of the next constant on the CONS USER menu

DELET Deletes the name and value of the constant currently displayed in the constant editor

Entering a Constant Name in an Expression

You can enter a constant in an expression in any of three ways.

- Select the constant name from the CONS BLTIN menu or the CONS USER menu.
- Select a user-created constant name from the VARS CONS screen.
- Use the ALPHA keys, alpha keys, and other character keys to enter a constant name.

Converting Units of Measure

You can enter a conversion expression anywhere that an expression is valid.

With the TI-86, you can convert a value measured in one unit into its equivalent value in another unit of measure. For example, you can convert inches to yards, quarts to liters, or degrees Fahrenheit to degrees Celsius.

The units of measure from which and to which you convert must be compatible. For example, you cannot convert inches to degrees Fahrenheit, or yards to calories. Each menu item on the CONV menu (page 62) represents a unit-of-measure group, such as length (**LNGTH**), volume (**VOL**), and pressure (**PRESS**). Within each menu, all units are compatible.

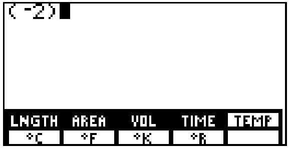
Converting a Unit of Measure

To use any conversion instruction, the syntax is:

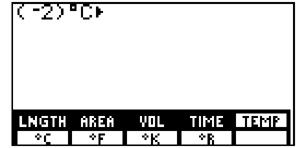
(*value***)***currentUnit*▶*newUnit*

*In the example, ⁻**2** degrees Celsius is converted to degrees Fahrenheit. Always use parentheses when value is negative.*

1. Enter the real *value* to be converted. [(] [(-)] **2** [)]
2. Display the CONV menu. [2nd] [CONV]
3. Select the **TEMP** conversion group. [F5]

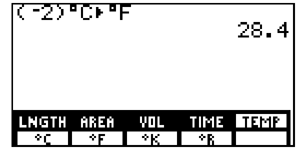


❹ Select the current unit of measure (°**C**) from the conversion group menu. The unit abbreviation and conversion symbol (▸) are pasted to the cursor location. [F1]

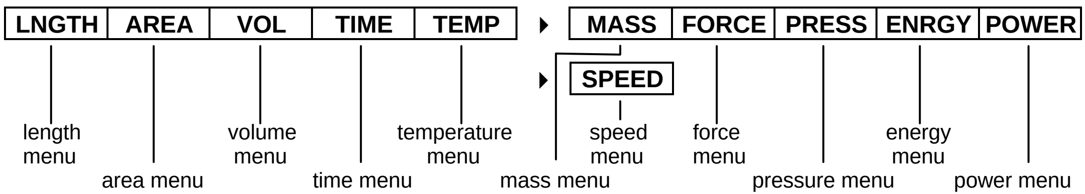


❺ Select the new unit of measure (°**F**) from the conversion group menu. The unit abbreviation is pasted to the cursor location. [F2]

❻ Convert the measurement. [ENTER]



The CONV (Conversions) Menu [2nd] [CONV]



Important: When you convert a negative value, you must enclose in parentheses the value and its negation sign, as in **(⁻4)**. Otherwise, the TI-86 order of evaluation will perform the conversion first, and then apply the negation to the converted value.

If you enter...	...The TI-86 converts it to...
(⁻4)°C▸°F	**24.8** degrees Fahrenheit (⁻4° Celsius converted to degrees Fahrenheit)
⁻4°C▸°F	**⁻39.2** degrees Fahrenheit (4° Celsius converted to degrees Fahrenheit, then negated)

The CONV LNGTH (Length) Menu [2nd] [CONV] [F1]

mm	millimeters	**yd**	yards	**mil**	mils
cm	centimeters	**km**	kilometers	**Ang**	Angstroms
m	meters	**mile**	miles	**fermi**	fermis
in	inches	**nmile**	nautical miles	**rod**	rods
ft	feet	**lt-yr**	light-years	**fath**	fathoms

The CONV AREA Menu [2nd] [CONV] [F2]

ft²	square feet	**km²**	square kilometers	**cm²**	square centimeters
m²	square meters	**acre**	acres	**yd²**	square yards
mi²	square miles	**in²**	square inches	**ha**	hectares

The CONV VOL (Volume) Menu [2nd] [CONV] [F3]

liter	liters	**cm³**	cubic centimeters	**tsp**	teaspoons
gal	gallons	**in³**	cubic inches	**tbsp**	tablespoons
qt	quarts	**ft³**	cubic feet	**ml**	milliliters
pt	pints	**m³**	cubic meters	**galUK**	UK gallons
oz	ounces	**cup**	cups	**ozUk**	UK ounces

The CONV TIME Menu [2nd] [CONV] [F4]

sec	seconds	**day**	days	**ms**	milliseconds
mn	minutes	**yr**	years	**μs**	microseconds
hr	hours	**week**	weeks	**ns**	nanoseconds

The CONV TEMP (Temperature) Menu [2nd] [CONV] [F5]

°C	degrees Celsius	**°F**	degrees Fahrenheit	**°K**	degrees Kelvin	**°R**	degrees Rankin

The CONV MASS Menu [2nd] [CONV] [MORE] [F1]

gm	grams	**amu**	atomic mass units	**ton**	tons
kg	kilograms	**slug**	slugs	**mton**	metric tons
lb	pounds				

The CONV FORCE Menu [2nd] [CONV] [MORE] [F2]

N	Newtons	**tonf**	ton force	**lbf**	pound force
dyne	dynes	**kgf**	kilogram force		

The CONV PRESS (Pressure) Menu [2nd] [CONV] [MORE] [F3]

atm	atmospheres	**lb/in^2**	pounds per square inch	**inHg**	inches of mercury
bar	bars	**mmHg**	millimeters of mercury	**inH$_2$O**	inches of water
N/m^2	Newtons per square meter	**mmH$_2$**	millimeters of water		

The CONV ENRGY (Energy) Menu [2nd] [CONV] [MORE] [F4]

J	Joules	**ft-lb**	foot-pounds	**erg**	ergs
cal	calories	**kw-hr**	kilowatt hours	**l-atm**	liter-atmospheres
Btu	British thermal units	**eV**	electron Volts		

The CONV POWER Menu [2nd] [CONV] [MORE] [F5]

hp	horsepower	**ftlb/s**	foot-pounds per second	**Btu/m**	British thermal units per minute
W	Watts	**cal/s**	calories per second		

The CONV SPEED Menu [2nd] [CONV] [MORE] [MORE] [F1]

ft/s	feet per second	**mi/hr**	miles per hour	**knot**	knots
m/s	meters per second	**km/hr**	kilometers per hour		

To enter a forward slash (⁄), you can use the ÷ key or paste it from the CATALOG*.*

Converting a Value Expressed as a Rate

To convert a value expressed as a rate on the home screen, you can use parentheses and the division operator (⁄). For example, if a car travels 325 miles in 4 hours, and you want to know the rate of speed in kilometers per hour, enter this expression:

(325⁄4)mi⁄hr▸km⁄hr This expression returns **131** km⁄hr (rounded up).

You also can return this result using only a forward slash, as in: **325mile▸km⁄4hr▸hr**

Number Bases

The number base mode setting (Chapter 1) controls how the TI-86 interprets an entered number and displays results on the home screen. However, you can enter numbers in any number base using number base designators b, o, d, and h. Then you can display the result on the home screen in any number base using number base conversions.

All numbers are stored internally as decimal. If you perform an operation in a mode setting other than **Dec**, the TI-86 performs integer mathematics, truncating to an integer after every calculation and expression.

For example, in **Hex** mode, **1⁄3+7** returns **7h** (1 divided by 3, truncated to 0, and then added to 7).

Number Base Ranges

Binary, octal, and hexadecimal numbers on the TI-86 are defined in these ranges.

Type	Low Value/High Value	Decimal Equivalent
Binary	1000 0000 0000 0001**b** 0111 1111 1111 1111**b**	-32,767 32,767
Octal	5120 6357 4134 0001**o** 2657 1420 3643 7777**o**	-99,999,999,999,999 99,999,999,999,999
Hexadecimal	FFFF A50C EF85 C001**h** 0000 5AF3 107A 3FFF**h**	-99,999,999,999,999 99,999,999,999,999

One's and Two's Complements

To obtain the one's complement of a binary number, enter the **not** function (page 68) before the number. For example, **not 111100001111** in **Bin** mode returns **1111000011110000**b.

To obtain the two's complement of a binary number, press [(-)] before entering the number. For example, -**111100001111** in **Bin** mode returns **1111000011110001**b.

The (Number) BASE Menu [2nd] [BASE]

A-F	TYPE	CONV	BOOL	BIT
hexadecimal characters menu	base type menu	base conversion menu	Boolean operator menu	rotate/shift menu

BASE A-F menu items and BASE TYPE menu items are not the same as regular alphabetical characters.

The BASE A-F (Hexadecimal Characters) Menu [2nd] [BASE] [F1]

This is the BASE A-F menu displayed on the home screen. To use A, press [2nd] [M1].

A	TYPE	CONV	BOOL	BIT
B	C	D	E	F

In the example, the upper menu is the list editor menu ([2nd] [LIST] in **Dec** *number base mode).*

When an editor menu is the upper menu, A and B are combined in one cell. If you press [F1] or [MORE]...

{	}	NAMES	"	OPS
A-B	C	D	E	F

...A and B move to two separate cells, and E and F are combined. To switch back, press [F5] or [MORE].

{	}	NAMES	"	OPS
A	B	C	D	E-F

Entering Hexadecimal Digits

If **Hex** *number base mode is not set, you must enter the* **h** *designator, even if the number contains a special hexadecimal character.*

To enter a hexadecimal number, use the number keys as you would for a decimal number. Select the hexadecimal characters A through **F** from the menu as needed.

The BASE TYPE Menu [2nd] [BASE] [F2]

A-F	TYPE	CONV	BOOL	BIT
b	h	o	d	

In an expression, you can designate a number in any number base, regardless of mode. After you enter the number, select the appropriate base type symbol from the BASE TYPE menu. The base type symbol is pasted to the cursor location. Here are some examples.

In **Dec** mode (default):	**10b+10** [ENTER]	**12**	In **Oct** mode:	**10b+10** [ENTER]	**12o**
	10h+10 [ENTER]	**26**		**10d+10** [ENTER]	**22o**
In **Bin** mode:	**10h+10** [ENTER]	**10010b**	In **Hex** mode:	**10b+10** [ENTER]	**12h**
	10d+10 [ENTER]	**1100b**		**10d+10** [ENTER]	**1Ah**

The BASE CONV (Conversion) Menu [2nd] [BASE] [F3]

A-F	TYPE	CONV	BOOL	BIT
▸Bin	▸Hex	▸Oct	▸Dec	

value can be an expression, list, vector, or matrix. For detailed syntax descriptions, refer to the A to Z Reference.

value▸**Bin**	Displays *value* as binary	*value*▸**Oct**	Displays *value* as octal
value▸**Hex**	Displays *value* as hexadecimal	*value*▸**Dec**	Displays *value* as decimal

Converting Number Bases

❶	In **Dec** mode, solve **10b + Fh + 10o + 10**.	**10b+Fh+10o+10** [ENTER]	**35**
❷	Add **1** to the result and convert it to **Bin** number base display.	**Ans+1▸Bin** [ENTER]	**100100b**
❸	Add **1** to the result and convert it to **Hex** number base display.	**Ans+1▸Hex** [ENTER]	**25h**
❹	Add **1** to the result and convert it to **Oct** number base display.	**Ans+1▸Oct** [ENTER]	**46o**
❺	Add **1** to the result and convert it to **Dec** number base display.	**Ans+1** [ENTER]	**39**

The BASE BOOL (Boolean) Menu [2nd] [BASE] [F4]

A-F	TYPE	CONV	BOOL	BIT
and	or	xor	not	

valueA **and** *valueB*   *valueA* **or** *valueB*   *valueA* **xor** *valueB*   **not** *value*

Both the argument and the result must be within defined number ranges (page 66).

Results of Boolean Operations

When a Boolean expression is evaluated, the arguments are converted to hexadecimal integers and the corresponding bits of the arguments are compared, as this table shows.

		Results			
If *valueA* is...	...and *valueB* is...	**and**	**or**	**xor**	**not** (*valueA*)
1	**1**	**1**	**1**	**0**	**0**
1	**0**	**0**	**1**	**1**	**0**
0	**1**	**0**	**1**	**1**	**1**
0	**0**	**0**	**0**	**0**	**1**

The result is displayed according to the current mode setting. For example:

- In **Bin** mode, **101 and 110** returns **100**b.
- In **Hex** mode, **5 and 6** returns **4**h.

The BASE BIT Menu [2nd] [BASE] [F5]

Rotate and shift operate on 16 base digits. To minimize an overflow error, enter the argument in binary form.

A-F	TYPE	CONV	BOOL	BIT
rotR	rotL	shftR	shftL	

rotR *value*	Rotates *value* right
rotL *value*	Rotates *value* left
shftR *value*	Shifts *value* right
shftL *value*	Shifts *value* left

Using Complex Numbers

Variable names with complex numbers stored to them are listed on the VARS CPLX *screen (Chapter 2).*

Lists, matrices, and vectors can have complex elements.

A complex number has two components: real (a) and imaginary (+b*i*). On the TI-86, you enter the complex number a+b*i* as:

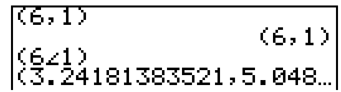
- **(***real*,*imaginary***)** in rectangular form
- **(**magnitude∠angle**)** in polar form

You can enter a complex number in rectangular or polar form, regardless of the current complex number mode setting. The separator (**,** or ∠) determines the form.

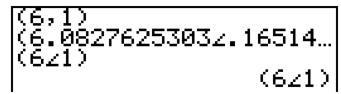
- To enter rectangular form, separate *real* and *imaginary* with a comma ([,]).
- To enter polar form, separate *magnitude* and *angle* with an angle symbol ([2nd] [∠]).

Each component (*real*, *imaginary*, *magnitude*, or *angle*) can be a real number or an expression that evaluates to a real number; expressions are evaluated when you press [ENTER].

When **RectC** complex number mode is set, complex numbers are displayed in rectangular form, regardless of the form in which you enter them (as shown to the right).



When **PolarC** complex number mode is set, complex numbers are displayed in polar form, regardless of the form in which you enter them (as shown to the right).



Complex Results

The graph format settings **RectGC** *and* **PolarGC** *(Chapter 5) determine the complex number form of graph screen coordinates.*

Complex numbers in results, including list, matrix, and vector elements, are displayed in the form (rectangular or polar) specified by the mode setting (Chapter 1) or by a display conversion instruction (page 61).

- When **Radian** angle mode is set, results are displayed as **(***magnitude*∠*angle***)**.
- When **Degree** angle mode is set, results are displayed as **(***real*,*imaginary***)**.

For example, when **PolarC** and **Degree** modes are set, **(2,1)**−**(1∠45)** returns **(1.32565429614∠12.7643896828)**.

Using a Complex Number in an Expression

- Enter the complex number directly.
- Use the ALPHA keys, alpha keys, and other character keys to enter a complex variable.
- Select a complex variable from the VARS CPLX screen.

The CPLX (Complex Number) Menu [2nd] [CPLX]

conj	real	imag	abs	angle	▸	▸Rec	▸Pol			

You can enter the name or a complex list, vector, or matrix as an argument for any CPLX *menu item.*

conj (*real*,*imaginary***)**	Returns the complex conjugate of a complex value, list, vector or matrix; the result is **(***real*,-*imaginary***)**
conj (*magnitude*∠*angle***)**	Returns **(***magnitude*∠-*angle***)**
real (*real*,*imaginary***)**	Returns the real portion of a complex number, list, vector, or matrix as a real number; the result is *real*
real (*magnitude*∠*angle***)**	Returns *magnitude* ***** cosine(*angle*)
imag (*real*,*imaginary***)**	Returns the imaginary (non-real) portion of a complex number, list, vector, or matrix as a real number; the result is *imaginary*
imag (*magnitude*∠*angle***)**	Returns *magnitude* ***** sine(*angle*)
abs (*real*,*imaginary***)**	(Absolute value) Returns the magnitude (modulus) of a complex number, list, vector, or matrix of complex numbers; the result is $\sqrt{(real^2+imaginary^2)}$
abs (*magnitude*∠*angle***)**	Returns *magnitude*

angle (*real*,*imaginary***)**	Returns the polar angle of a complex number, list, vector, or matrix calculated as **$\tan^{-1}$ (***imaginary/real***)** (adjusted by π in the second quadrant or ⁻π in the third quadrant); the result is $\tan^{-1}$(*imaginary/real*)
angle (*magnitude∠angle***)**	Returns *angle* (where ⁻π<*angle*≤π)
complexValue▸**Rec**	Displays *complexValue* in rectangular format **(***real*,*imaginary***)**, regardless of complex mode setting; valid only at the end of a command and only when *complexValue* is indeed complex
complexValue ▸**Pol**	Displays *complexValue* in polar format (*magnitude∠angle*), regardless of complex mode setting; valid only at the end of a command and only when *complexValue* is indeed complex

Select **{** *and* **}** *from the* LIST *menu.*

You must enter commas to separate list elements.

You can enter a complex list, vector, or matrix directly. The syntax below is for lists. To enter a complex vector or matrix, substitute brackets for braces below and use the correct form for either data type (Chapters 12 and 13).

In rectangular form, to use lists of complex numbers with **conj**, **real**, **imag**, **abs**, and **angle**, the syntax is:

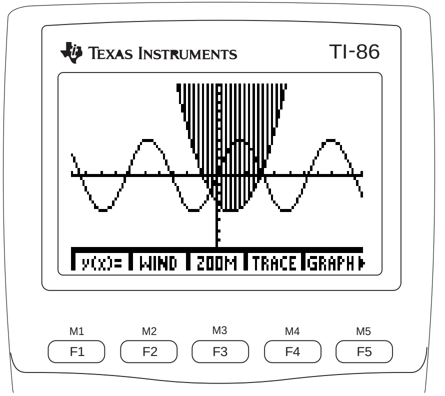
conj{(*realA*,*imaginaryA***),(***realB*,*imaginaryB***),(***realC*,*imaginaryC***),...}**

In polar form, to use lists of complex numbers with **conj**, **real**, **imag**, **abs**, and **angle**, the syntax is:

real{(*magnitudeA∠angleA***),(***magnitudeB∠angleB***),(***magnitudeC∠angleC***),...}**

When you use a list the TI-86 calculates the result element by element and returns a list, in which each element is expressed according to the complex mode setting.

5 Function Graphing



Defining a Graph

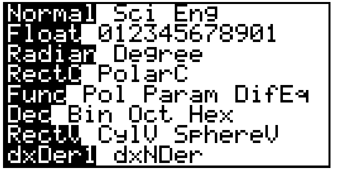
This chapter describes the process for graphing functions in **Func** graphing mode, but the process is similar for each TI-86 graphing mode. Chapters 8, 9, and 10 describe the unique aspects of polar, parametric, and differential equation graphing modes. Chapter 6 describes various graphing tools, many of which you can use in all graphing modes.

Some of these steps are not necessary every time you define a graph.

1. Set the graphing mode (page 74).
2. Define, edit, or select one or more functions in the equation editor (pages 76 and 77).
3. Select the graph style for each function (page 79).
4. Deselect stat plots, if necessary (page 81).
5. Set the viewing window variables (page 81).
6. Select the graph format settings (page 83).

Setting the Graph Mode

To display the mode screen, press 2nd [MODE]. All default mode settings, including **Func** graphing mode, are highlighted in the picture to the right. The graphing modes are on the fifth line.



- **Func** (function graphing)
- **Pol** (polar graphing; Chapter 8)
- **Param** (parametric graphing; Chapter 9)
- **DifEq** (differential equation graphing; Chapter 10)

Each graphing mode has a unique equation editor. You must select the graphing mode and **Dec** number base mode before you enter the functions. The TI-86 retains in memory all equations stored to the **Func**, **Pol**, **Param**, and **DifEq** equation editors. Each mode also has unique graph format settings and window variables.

Stat plot on/off status, zoom factors, mode settings, and tolerance apply to all graphing modes; changing the graphing mode does not affect them.

Chapter 1 describes all mode settings in detail.

These mode settings affect graphing results.

- **Radian** or **Degree** angle mode affects the interpretation of some functions.
- **dxDer1** or **dxNDer** differentiation mode affects plotting of selected functions.

The GRAPH Menu GRAPH

y(x)=	WIND	ZOOM	TRACE	GRAPH	▸	MATH	DRAW	FORMT	STGDB	RCGDB
					▸	EVAL	STPIC	RCPIC		

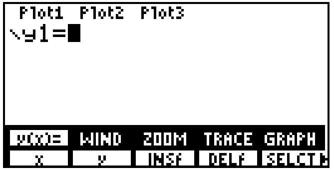
Chapter 6 describes these GRAPH *menu items:* **ZOOM**, **TRACE**, **MATH**, **DRAW**, **STGDB**, **RCGDB**, **EVAL**, **STPIC**, *and* **RCPIC**.

y(x)=	Displays the equation editor; use this screen to enter functions to be graphed
WIND	Displays the window editor; use this editor to change graph screen dimensions
ZOOM	Displays the GRAPH ZOOM menu; use these items to change the graph screen dimensions
TRACE	Activates the trace cursor; use this cursor to trace along the graph of a specific function
GRAPH	Displays the graph screen; graphs all selected functions and turned on stat plots
MATH	Displays the GRAPH MATH menu; use this menu to explore graphs mathematically
DRAW	Displays the GRAPH DRAW menu; use this menu to draw on graphs or test pixels

FORMT	Displays the graph format screen; use this screen to select graph format settings
STGDB	Displays the **Name=** prompt and STGDB menu; use this prompt to enter a **GDB** variable
RCGDB	Displays the **Name=** prompt and RCGDB menu; use this menu to recall a graph database
EVAL	Displays the **Eval x=** prompt; enter an **x** for which you want to solve the current function
STPIC	Displays the **Name=** prompt and STPIC menu; use this prompt to enter a **PIC** variable
RCPIC	Displays the **Name=** prompt and RCPIC menu; use this menu to recall a picture

Using the Equation Editor

To display the equation editor in **Func** graphing mode, select **y(x)=** from the GRAPH menu ([GRAPH] [F1]). The GRAPH menu shifts up and the equation editor menu is displayed as the lower menu. You can store up to 99 functions in the equation editor, if sufficient memory is available.



If a function is selected, its equals sign (=) is highlighted in the equation editor. If the function is deselected, its equals sign is not highlighted. Only selected functions are plotted when the TI-86 plots a graph.

The Equation Editor (GRAPH y(x)=) Menu [GRAPH] [F1]

y(x)=	WIND	ZOOM	TRACE	GRAPH
x	y	INSf	DELf	SELCT

▸

ALL+	ALL−	STYLE		

x	Pastes the variable **x** to the current cursor location (same as [x-VAR] or [2nd] [alpha] [X])
y	Pastes the variable **y** to the current cursor location (same as [2nd] [alpha] [Y])
INSf	Inserts a deleted equation variable (function) name above the current cursor location (only the variable name is inserted)
DELf	Deletes the function that the cursor is on
SELCT	Changes the selection status of the function that the cursor is on (selects or deselects)
ALL+	Selects all defined functions in the equation editor
ALL−	Deselects all defined functions in the equation editor
STYLE	Assigns the next of seven available graph styles to the function that the cursor is on

Defining a Function in the Equation Editor

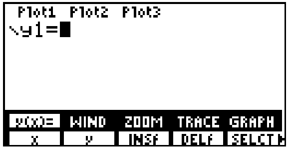
To move from the first equation editor function to the last, press [▲].

To move to the beginning or end of an equation, press [2nd] [◄] or [2nd] [►].

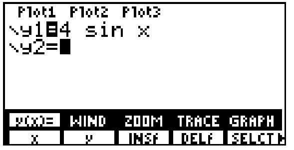
An ellipsis indicates that an equation continues beyond the screen.

❶ Display the equation editor. — [GRAPH] [F1]

❷ If functions are stored in the equation editor, move the cursor down until a blank function is displayed. — ([▼] or [ENTER])



❸ Enter an equation in terms of **x** to define the function. When you enter the first character, the function is selected automatically. (The function's equals sign is highlighted.) — **4** [SIN] [x-VAR]



❹ Move the cursor to the next function. — [ENTER] or [▼]

Notes about Defining Function Equations

You can edit expressions you inserted using **Rcl**.

- You can include functions, variables, constants, matrices, matrix elements, vectors, vector elements, lists, list elements, complex values, or other equations in the equation.
- If you include matrices, vectors, or complex values, the equation must evaluate to a real number at each point.
- You can include another defined function in an equation. For example, given **y1=sin x** and **y2=4+y1**, the function **y2** would equal 4 plus the sine of x.
- To enter a function name, select **y** from the equation editor menu, and then enter the appropriate number.
- To insert the contents of an equation variable, use RCL (Chapter 1). To enter the equation variable at the **Rcl** prompt, use the ALPHA keys, alpha keys, and other character keys.
- To select all functions from the home screen or in the program editor, select **FnOn** from the CATALOG (or enter the individual characters) and press [ENTER].
- To select specific functions from the home screen or in the program editor, select **FnOn** from the CATALOG (or enter the individual characters), enter the number of each function, and press [ENTER]. For example, to select **y1**, **y3**, and **y5**, enter **FnOn 1,3,5**.
- To deselect functions from the home screen or in the program editor, use **FnOff** the same way you use **FnOn** to select functions.
- When a function evaluates to a non-real number, the value is not plotted on the graph; no error is returned.

Selecting Graph Styles

The TI-86 graphs all selected functions on the same graph screen.

Depending on which graphing mode is set, the TI-86 offers up to seven distinct graph styles. You can assign these styles to specific functions to visually differentiate each from the others.

For example, you can show **y1** as a connected line (⋱**y1=** in the equation editor) and **y2** as a dotted line (⋰**y2=**), and shade the area above **y3** (◥**y3=**).

Also, you can manipulate the styles to illustrate actual phenomena graphically, such as a ball flying through the air (using ⁝) or the circular movement of a chair on a Ferris wheel (using ⁝).

◥ (shade above) and ◣ (shade below) are available only in **Func** *graphing mode.*

⋰ (dot) is available in all graphing modes except **DifEq** *graphing mode.*

Icon	Style	Characteristics of the Plotted Function
⋱	Line	A solid line connects each plotted point; this is the default in **Connected** mode
⋱	Thick	A thick solid line connects each plotted point
◥	Above	Shades the area above the function
◣	Below	Shades the area below the function
⁝	Path	A circle cursor traces the leading edge of the function and draws a path as it plots
⁝	Animate	A circle cursor traces the leading edge of the function as it plots; does not draw a path
⋰	Dot	A small dot represents each plotted point; this is the default in **Dot** mode

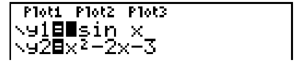
To set the graph style from a program, select **GrStl(** from the CATALOG (A to Z Reference).

Setting the Graph Style in the Equation Editor

In the example, (shade above) is selected for **y2**. *All window variables are set to the default values (page 82).*

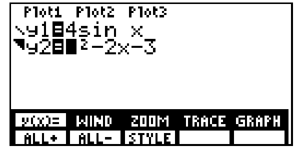
❶ Display the equation editor. [GRAPH] [F1]

❷ Move the cursor to the function or functions for which you want to set the graph style. [▼]



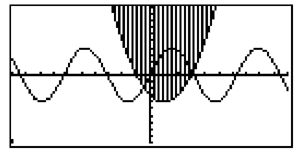
❸ Display the equation editor menu item **STYLE**. [MORE]

❹ Select **STYLE** repeatedly to scroll the graph style icons to the left of the equation name. [F3] [F3]



❺ View the graph with the new graph style. [2nd] [F5]

❻ Clear the GRAPH menu to view the graph only. [CLEAR]



Using Shading Patterns to Differentiate Functions

When you select (shade above) or (shade below) for more than one function, the TI-86 rotates through a series of four shading patterns.

If you assign or to a function that graphs a family of curves (page 86), the same pattern rotation applies to the members of the family of curves.

- First shaded function: vertical lines
- Second shaded function: horizontal lines
- Third shaded function: negatively sloping diagonal lines
- Fourth shaded function: positively sloping diagonal lines

The rotation returns to vertical lines for the fifth shaded function and repeats the order.

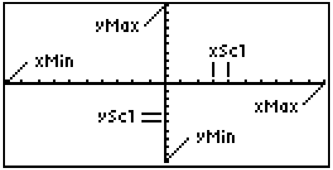
Viewing and Changing On/Off Status of Stat Plots

Plot1 Plot2 Plot3 on the top line of the equation editor displays the on/off status of each stat plot (Chapter 14). When a plot name is highlighted on this line, the plot is on.

To change the on/off status of a stat plot from the equation editor, press [▲], [▶], and [◀] to place the cursor on **Plot1**, **Plot2**, or **Plot3**, and then press [ENTER].

Setting the Window Variables

The graph screen window represents the portion of the coordinate plane displayed on the graph screen. By setting window variables, you can define the graph screen window boundaries and other attributes.



xMin, **xMax**, **yMin**, and **yMax** are the graph screen boundaries.

To remove tick marks from both axes, set **xScl=0** *and* **yScl=0**.

xScl (x scale) is the number of units represented by the distance from one tick mark to the next tick mark on the x-axis.

yScl (y scale) is the number of units represented by the distance from one tick mark to the next tick mark on the y-axis.

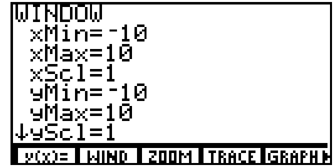
Small **xRes** *values improve graph resolution but may cause the TI-86 to plot graphs more slowly.*

xRes sets pixel resolution for function graphs only, using integers **1** through **8**.

- At **xRes=1** (the default), functions are evaluated and graphed at each pixel on the x-axis.
- At **xRes=8**, functions are evaluated and graphed at every eighth pixel along the x-axis.

Displaying the Window Editor

To display the window editor, select **WIND** from the GRAPH menu ([GRAPH] [F2]). Each graphing mode has a unique window editor. The window editor to the right shows the default values in **Func** graphing mode. ↓ indicates that **xRes=1** (x resolution) is below **yScl** on the window editor.

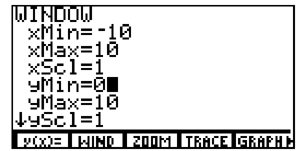


Changing a Window Variable Value

xMin<**xMax** *and* **yMin**<**yMax** *both must be true to graph successfully.*

In the example, **yMin** *is changed to* **0**.

1. Display the window editor. — [GRAPH] [F2]
2. Move the cursor to the window variable you want to change. — [▼] [▼] [▼]
3. Edit the value, which can be an expression. — **0**
4. Evaluate any expressions and store the value. — [ENTER] or [▼]



To change a window variable value from the home screen or in the program editor, enter the value, and then press [STO▶]. Either select the window variable from the VARS WIND screen ([2nd] [CATLG-VARS] [MORE] [MORE] **WIND**) or enter individual characters. Press [ENTER].

Setting Graphing Accuracy with Δx and Δy

The window variables Δ**x** and Δ**y** define the distance from the center of one pixel to the center of any adjacent pixel. When you display a graph, the values of Δ**x** and Δ**y** are calculated from **xMin**, **xMax**, **yMin**, and **yMax** using these formulas:

$\Delta x=(xMin+xMax)/126$ $\Delta y=(yMin+yMax)/62$

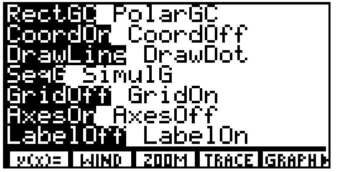
Δ**x** and Δ**y** are not on the window editor. To change them, you must follow the steps above for changing a window variable value from the home screen or in the program editor. When you change the values stored to Δ**x** and Δ**y**, the TI-86 automatically recalculates **xMax** and **yMax** from Δ**x**, **xMin**, Δ**y**, and **yMin**, and the new values are stored.

Setting the Graph Format

The TI-86 retains independent format settings for each graphing mode.

In **DifEq** *graphing mode, the graph format screen key sequence is* [GRAPH] [MORE] [F1] *(Chapter 10).*

To display the graph format screen, select **FORMT** from the GRAPH menu ([GRAPH] [MORE] [F3]). The graph format settings define various characteristics of the displayed graph. The current settings are highlighted.



To change a setting, move the cursor onto the new setting, and then press [ENTER], the same as on the mode screen.

__DifEq__ graphing mode has a unique set of graph format settings (Chapter 10).

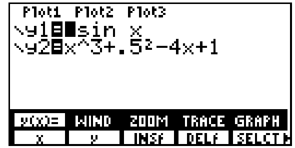
Grid points cover the graph screen in rows that correspond to the tick marks on each axis.

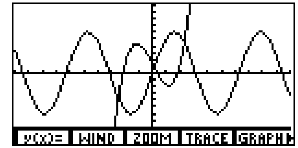
RectGC	Displays the cursor location as rectangular graph coordinates **x** and **y**; when **RectGC** is set, plotting the graph, moving the free-moving cursor, and tracing update **x** and **y**; if **CoordOn** format also is selected, **x** and **y** are displayed
PolarGC	Displays the cursor location as polar graph coordinates **R** and θ; when **PolarGC** is set, plotting the graph, moving the free-moving cursor, and tracing update **x**, **y**, **R** and θ; if **CoordOn** format also is selected, **R** and θ are displayed
CoordOn	Displays the cursor coordinates at the bottom of the graph
CoordOff	Does not display the cursor coordinates at the bottom of the graph
DrawLine	Draws a line between the points calculated for the functions in the equation editor
DrawDot	Plots only the calculated points for the functions in the equation editor
SeqG	(sequential graphing) Evaluates and plots one function completely before evaluating and plotting the next function
SimulG	(simultaneous graphing) Evaluates and plots all selected functions for a single value of **x** and then evaluates and plots them for the next value of **x**
GridOff	Omits the grid points from the display
GridOn	Displays grid points
AxesOn	Displays the axes
AxesOff	Omits the axes from the display; **AxesOff** overrides the **LabelOff**/**LabelOn** format setting
LabelOff	Omits the axis labels from the display
LabelOn	Labels the axes, if **AxesOn** is also selected; **x** and **y** for **Func**, **Pol**, and **Param** modes; various labels in **DifEq** mode

Displaying a Graph

In the example graph to the right, all default settings related to graphing are set.

To display a graph, select **GRAPH** from the GRAPH menu. The graph screen is displayed. If the graph is newly defined, the busy indicator is displayed at the top-right corner as the TI-86 draws the graph.





To view the graph without the GRAPH *menu on the bottom line, press* [CLEAR] *after plotting the graph.*

- In **SeqG** format, the TI-86 draws each selected function one by one, in function-name order (for example, **y1** is graphed first, **y2** is graphed second, and so on).
- In **SimulG** format, the TI-86 draws all selected graphs simultaneously.

You can display and explore a graph from a program (Chapter 16). To use graphing commands on the home screen, select them from the CATALOG or entering the individual characters.

Pausing or Stopping a Graph in Progress

When you pause, the busy indicator in the top-right corner becomes a dotted line.

- To pause graph plotting, press [ENTER]. To resume plotting, press [ENTER] again.
- To stop graph plotting, press [ON]. To replot, select **GRAPH** from the GRAPH menu.

Modifying a Drawn Graph

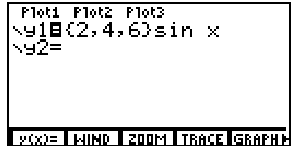
To remove these items from the graph screen:	Press (or select):
Cursor, coordinate values, or menus (To restore menus, press [EXIT] or [GRAPH])	[CLEAR]
Free-moving cursor and coordinate values but not the menus	[ENTER]
Cursor and coordinate values but not the menus	[GRAPH] or **GRAPH**

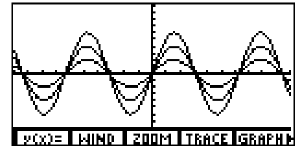
Graphing a Family of Curves

If you enter a list as an element in an equation, the TI-86 plots the function for each value in the list, graphing a family of curves. In **SimulG** graphing order mode, the TI-86 graphs all functions sequentially for the first element in each list, then for the second element, and so on.

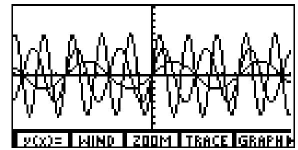
When you use more than one list in an expression, all lists must have the same dimension.

For example, **{2,4,6} sin x** graphs three functions:
2 sin x, **4 sin x**, and **6 sin x**.





The equation **{2,4,6} sin ({1,2,3} x)** also graphs three functions:
2 sin x, **4 sin (2x)**, and **6 sin (3x)**.

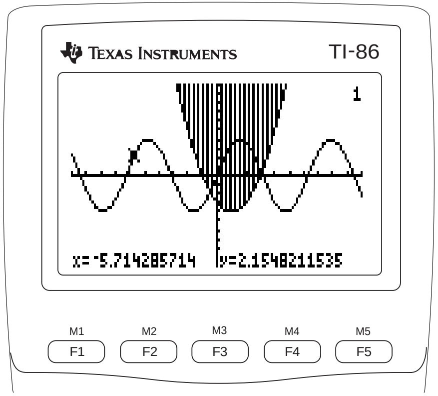


Smart Graph

Smart Graph displays the previously displayed graph when you press GRAPH, as long as all factors that would cause replotting are unchanged since the graph was last displayed. Smart Graph replots if you performed any of these actions since the graph was last displayed.

- Changed a mode setting that affects graphs
- Changed a function or stat plot that was plotted on the last graph screen
- Selected or deselected a function or stat plot
- Changed the value of a variable in a selected function
- Changed the value of a window variable setting
- Changed a graph format setting

6 Graph Tools



Graph Tools on the TI-86

Chapter 5 describes how to use the GRAPH menu items **y(x)=**, **WIND**, **GRAPH**, and **FORMT** to define and display the graph of a function in **Func** graphing mode. This chapter describes how to use the other GRAPH menu items to apply preset graph screen dimensions, explore the graph and trace specific functions, perform mathematical analyses, draw on graphs, and store and recall graphs and drawings. You can use most graph tools in all four graphing modes.

The GRAPH Menu [GRAPH]

This is the GRAPH *menu in* **Func** *graphing mode. The* GRAPH *menu differs slightly from graphing mode to graphing mode.*

y(x)=	WIND	ZOOM	TRACE	GRAPH	▸	MATH	DRAW	FORMT	STGDB	RCGDB
					▸	EVAL	STPIC	RCPIC		

ZOOM Displays the GRAPH ZOOM menu; use these items to apply preset graph screen dimensions

TRACE Activates the trace cursor; use this cursor to trace along graphs of specific functions

MATH Displays the GRAPH MATH menu; use this menu to explore graphs mathematically

DRAW Displays the GRAPH DRAW menu; use this menu to draw on graphs

STGDB Displays the **Name=** prompt and GDB menu; use this prompt to enter a **GDB** variable

RCGDB Displays the **Name=** prompt and GDB menu; use this menu to recall a **GDB** variable

EVAL Displays the **Eval x=** prompt; use this prompt to enter an **x** value for which you want to solve the current function

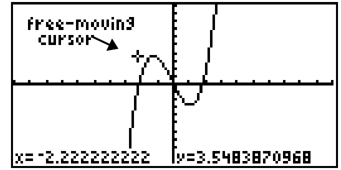
STPIC Displays the **Name=** prompt and PIC menu; use this prompt to enter a **PIC** variable

RCPIC Displays the **Name=** prompt and PIC menu; use this menu to recall **PIC** variable

Using the Free-Moving Cursor

In the example, the function **y(x)=x^3+.3x^2-4x** *is graphed.*

When you select **GRAPH** from the GRAPH menu, the graph screen is displayed with the free-moving cursor at the center of the screen.



The cursor appears as a plus sign with a flashing center pixel. To move the cursor, press ▶, ▼, ◀, or ▲; it moves in the direction of the cursor key you press.

- In **RectGC** format, each cursor movement updates the variables **x** and **y**. In **PolarGC** format, each cursor movement updates **x**, **y**, **R**, and θ.
- In **CoordOn** format, the **x** and **y** cursor coordinates are displayed at the bottom of the graph screen as you move the cursor.

The numeric display mode settings do not affect coordinate display.

Graphing Accuracy

The coordinate values displayed as you move the cursor approximate actual mathematical coordinates, accurate to within the width and height of the pixel. As the difference between **xMin** and **xMax** and between **yMin** and **yMax** becomes smaller (for example, when you zoom in on a graph), graphing is more accurate and coordinate values approximate the actual mathematical coordinates more closely.

The free-moving cursor coordinates represent the cursor location on the graph screen. Moving the free-moving cursor precisely from one plotted point to the next along a function is very difficult. To move along a function easily, use the trace cursor (page 90).

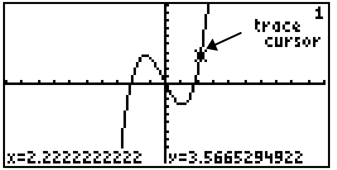
Tracing a Graph

To display the graph and begin a trace, select **TRACE** from the GRAPH menu.

In the example, the function **y(x)=x^3+.3x²-4x** *is graphed.*

The trace cursor appears as a small square with a flashing diagonal line at each corner. Initially, the trace cursor appears on the first selected function, at the **x** value closest to the middle of the screen.

If **CoordOn** format is selected, the cursor coordinates are displayed at the bottom of the screen.



When you enter the first character of an independent variable value, an **x=** *prompt is displayed (or* θ**=** *or* **t=***). The value can be an expression.*

If the function is undefined at an **x** *value, then the* **y** *value is blank.*

To move the trace cursor...	Press these keys:
To the next larger or next smaller plotted point in a function	[▶] or [◀]
To any valid independent-variable value (**x**, θ, or **t**) on the current equation	*value* [ENTER]
From one function to another function at **x**, in the order or reverse order of the selected functions in the equation editor	[▼] or [▲]
From one member to another member of a family of curves (Chapter 5)	[▼] or [▲]

As you move the trace cursor along a function, the **y** value is calculated from the **x** value. That is, **y=y***n***(x)**. When you trace beyond the top or bottom of the graph screen, the coordinates displayed on the screen continue to change as if the cursor were still on the screen.

Panning: To view function coordinates to the left or right of the current graph screen, press and hold [◀] or [▶] while tracing. When you pan beyond the left or right side of the screen during a trace, the TI-86 automatically changes the values of **xMin** and **xMax**.

Quick Zoom: While tracing, you can press [ENTER] to adjust the graph screen so that the trace cursor location becomes the center of a new graph screen, even if you have moved the cursor beyond the top or bottom of the display. In effect, this is vertical panning.

Stopping and Resuming a Trace

To stop tracing and restore the free-moving cursor, press [CLEAR] or [GRAPH].

To resume tracing, select **TRACE** from the GRAPH menu. If Smart Graph has not replotted the graph (Chapter 5), the trace cursor is at the point where you stopped tracing.

Resizing the Graph Screen with ZOOM Operations

To view the current window variable values, select **WIND** *from the* GRAPH *menu.*

The standard TI-86 graph screen displays the portion of the xy plane defined by the values stored to the window variables. With the GRAPH ZOOM menu items, you can change some or all of the window variable values and redisplay the graph, usually with one keystroke. As a result, a smaller or larger portion of the xy plane is displayed.

The GRAPH ZOOM Menu [GRAPH] [F3]

y(x)=	WIND	ZOOM	TRACE	GRAPH
BOX	ZIN	ZOUT	ZSTD	ZPREV

ZFIT	ZSQR	ZTRIG	ZDECM	ZDATA
ZRCL	ZFACT	ZOOMX	ZOOMY	ZINT
ZSTO				

To cancel the effect of any ZOOM *menu item and return to the default window variable values, select* **ZSTD**.

If you graph a circle but it appears elliptical, you can use **ZSQR** *to reset the window variable values so that the circle graph appears circular.*

BOX	Draws a box to define the graph screen
ZIN	(zoom in) Magnifies the graph around the cursor by factors of **xFact** and **yFact**
ZOUT	(zoom out) Displays more of the graph around the cursor by factors of **xFact** and **yFact**
ZSTD	Displays the graph in standard dimensions; resets the default window variable values
ZPREV	Reverses the last zoom operation; window variables revert to previous values
ZFIT	Recalculates **yMin** and **yMax** to include the minimum and maximum **y** values of the selected functions between the current **xMin** and **xMax**
ZSQR	Sets equal-size pixels on the x-axis and y-axis; adjusts window variable values in one direction so that Δ**x=**Δ**y**, while **xScl** and **yScl** remain unchanged; the midpoint of the current graph (not the axes intersection) becomes the midpoint of the new graph
ZTRIG	Sets built-in window variables appropriate for trigonometric functions in **Radian** mode: **xMin=⁻8.24668071567** **xScl=1.5707963267949** ($\pi/2$) **yMax=4** **xMax=8.24668071567** **yMin=⁻4** **yScl=1**
ZDECM	Sets Δ**x=.1**, Δ**y=.1**, **xMin=⁻6.3**, **xMax=6.3**, **xScl=1**, **yMin=⁻3.1**, **yMax=3.1**, and **yScl=1**
ZDATA	Sets window variable values to display all statistical data points; adjusts **xMin** and **xMax** only; applies to histograms, scatter plots, and stat plots only (Chapter 14)
ZRCL	Uses window variable values stored in the user-defined zoom-window variables (**ZSTO**)
ZFACT	Displays the ZOOM FACTORS screen
ZOOMX	Zooms out by a factor of **xFact** only; ignores **yFact** (page 93)
ZOOMY	Zooms out by a factor of **yFact** only; ignores **xFact**
ZINT	Sets integer values on the axes; sets Δ**x=1**, Δ**y=1**, **xScl=10**, and **yScl=10**; the current cursor becomes the center of the new graph screen after you press [ENTER]
ZSTO	Stores current window variable values to user-defined zoom-window variables (**ZRCL**)

Defining a Custom Zoom In

Using **BOX**, you can zoom in on any rectangular area within the current graph screen.

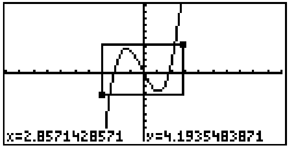
Before you begin these steps, enter a function in the equation editor. In the example, the function **y(x)=x^3+.3x²−4x** *is graphed.*

To cancel **BOX** *without redefining the graph screen, press* [CLEAR].

When you replot the graph, the TI-86 updates the window variable values.

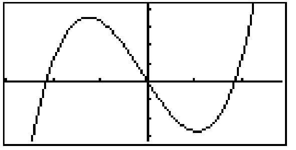
❶ Select **BOX** from the GRAPH ZOOM menu. The zoom cursor is displayed at center screen. [GRAPH] [F3] [F1]

❷ Move the cursor to any spot you want to define as a corner of the zoom box; mark the corner with a small square. [▶] [▼] [◀] [▲] [ENTER]



❸ Move the cursor away from the first corner, creating an adjustable box whose diagonal corners are the small square and the cursor. [▶] [▼] [◀] [▲]

❹ When you have defined the box, replot all selected functions in the new graph screen. [ENTER]



❺ Clear the menus from the screen. [CLEAR]

Setting Zoom Factors

To store to **xFact** *or* **yFact** *from the home screen or in the program editor, you can select it from the* VARS ALL *screen or enter it using* ALPHA *and alpha keys.*

Zoom factors define the magnification or reduction factor by which **ZIN**, **ZOUT**, **ZOOMX**, and **ZOOMY** zoom in or zoom out around a point. To display the zoom factors editor, select **ZFACT** from the GRAPH ZOOM menu (press [GRAPH] [F3] [MORE] [MORE] [F2]). **xFact** and **yFact** must be ≥ 1. The default value for both factors is **4** in all graphing modes.

Zooming In and Zooming Out on a Graph

ZIN magnifies the part of the graph surrounding the cursor location. **ZOUT** displays a greater portion of the graph, centered on the cursor location. **xFact** and **yFact** determine the extent. The steps below describe how to use **ZIN**. To use **ZOUT**, select it instead of **ZIN** in step 2.

In the example, the function **y(x)=x^3+.3x²−4x** *is graphed.*

❶ Check **xFact** and **yFact**; change as needed.

[GRAPH] [F3] [MORE] [MORE] [F2]

When you select a ZOOM *feature, Smart Graph displays the current graph.*

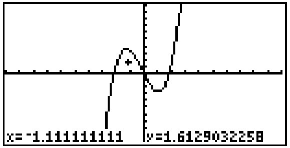
To cancel a zoom before you complete it, press [CLEAR].

❷ Select **ZIN** from the GRAPH ZOOM menu to display the zoom cursor.

[F3] [F2]

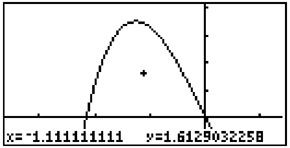
❸ Move the zoom cursor to the intended new center point of the graph screen.

[▶] [▼] [◀] [▲]



❹ Zoom in. The TI-86 adjusts the graph screen by **xFact** and **yFact**, updates window variable values, and replots the selected functions centered on the cursor location.

[ENTER]



You can continue to zoom in (or zoom out) on the current graph, unless you press a key other than [ENTER], [▶], [▼], [◀], or [▲].

- To zoom in (or zoom out) again at the same point, press [ENTER].
- To zoom in (or zoom out) at a new center point, move the cursor and press [ENTER].

To zoom out only on the horizontal axis by a factor of **xFact**, select **ZOOMX** instead of **ZIN** in step 2 above. **ZOOMX** plots the selected functions centered on the cursor location and updates some window variable values; **yMin** and **yMax** are unchanged.

To zoom out only on the vertical axis by a factor of **yFact**, select **ZOOMY** instead of **ZIN** in step 2 above. **ZOOMY** plots the selected functions centered on the cursor location and updates some window variable values; **xMin** and **xMax** are unchanged.

You can select all zoom-window variables from the VARS WIND *screen in any graph mode.*

You also can enter the variable characters individually.

The zoom-window variables resume their standard default values when you reset defaults.

Storing and Recalling Zoom-Window Variable Values

- To store all current zoom-window variable values simultaneously as a user-defined custom zoom feature, select **ZSTO** from the GRAPH ZOOM menu.
- To execute a user-defined custom zoom, which resets the graph screen to the stored zoom-window variables, select **ZRCL** from the GRAPH ZOOM menu.

Using **ZSTO** in these graphing modes:	Stores to these zoom-window variables:
Func, **Pol**, **Param**, and **DifEq** graphing modes	**zxMin**, **zxMax**, **zxScl**, **zyMin**, **zyMax**, and **zyScl**
Pol graphing mode only	**zθMin**, **zθMax**, and **zθStep**
Param graphing mode only	**ztMin**, **ztMax**, and **ztStep**
DifEq graphing mode only	**ztMin**, **ztMax**, **ztStep**, and **ztPlot**

Using Interactive Math Functions

When you select a GRAPH MATH operation, Smart Graph displays the current graph with the trace cursor. To perform the GRAPH MATH operation, press [▼] and [▲] to move to the function.

When a GRAPH MATH menu operation prompts you to specify left bound, right bound, and guess, the accuracy of the values you specify will affect the length of time the TI-86 spends calculating the answer; the better the guess, the shorter the calculation time.

The GRAPH MATH Menu [GRAPH] [MORE] [F1]

MATH	DRAW	FORMT	STGDB	RCGDB
ROOT	dy/dx	∫f(x)	FMIN	FMAX

▸

INFLC	YICPT	ISECT	DIST	ARC

▸

TANLN				

The GRAPH MATH *menu differs slightly for* **Pol** *and* **Param** *graphing modes (Chapters 8 and 9).*

DifEq *graphing mode has no* GRAPH MATH *menu.*

ROOT	Finds the root of a function using a specified left bound, right bound, and guess
dy/dx	Finds a numeric derivative (slope) of a function at the trace cursor location
∫f(x)	Finds a function's numerical integral using a specified left bounds and right bound
FMIN	Finds a function's minimum using a specified left bound, right bound, and guess
FMAX	Finds a function's maximum using a specified left bound, right bound, and guess
INFLC	Finds a function's inflection point using a specified left bound, right bound, and guess
YICPT	Finds a function's y-intercept (**y** at **x=0**)
ISECT	Finds the intersection of two functions using a specified left bound, right bound, and guess
DIST	Finds the straight-line distance between a specified left bound and right bound
ARC	Finds the distance along a function between two specified points on the function
TANLN	Draws the tangent line at a specified point

Settings That Affect GRAPH MATH Operations

- The tolerance variable **tol** (Appendix) affects the accuracy of **∫f(x)**, **FMIN**, **FMAX**, and **ARC**. Accuracy increases as the tolerance value becomes smaller.
- The step-size variable δ (Appendix) affects the accuracy of **dy/dx**, **INFLC** in **dxNDer** differentiation mode (Chapter 1), **ARC**, and **TANLN**. Accuracy increases as the step-size value becomes smaller.
- The differentiation mode setting affects **dy/dx**, **INFLC**, **ARC**, and **TANLN**; **dxDer1** (exact) mode is more accurate than **dxNDer** (numeric) mode (Chapter 1).

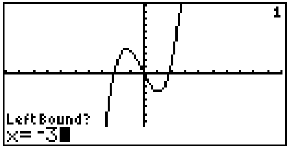
Using ROOT, FMIN, FMAX, or INFLC

The steps for **ROOT**, **FMIN**, **FMAX**, and **INFLC** are the same, except for the menu selection in step 1.

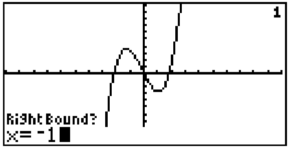
In the example, the function **y(x)=x^3+.3x²−4x** *is selected. Step 2 is not necessary here because only one function is selected.*

When you enter a value directly for the left bound, right bound, or guess, an **x=** *prompt is displayed on the bottom of the graph screen.*

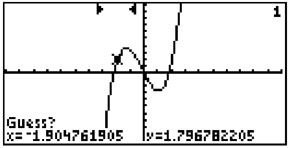
1. Select **ROOT** from the GRAPH MATH menu. A **Left Bound?** prompt is displayed. — [GRAPH] [MORE] [F1] [F1]
2. Move the cursor onto the function for which you want to find a root. — [▼] [▲]



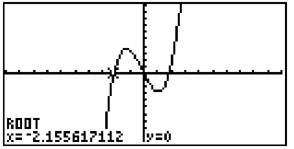
3. Specify the left bound for **x**. Either move the trace cursor to the left bound or enter a value directly. **Right Bound?** is displayed. — [(−)] **3** [ENTER] (or [◄] [►] [ENTER])



4. Specify the right bound for **x** as in step 3. **Guess?** is displayed. — [(−)] **1** [ENTER] (or [◄] [►] [ENTER])
5. Guess an **x** value near the root between the left bound and the right bound. Either move the cursor or enter a value. — [◄] [►] (or [(−)] **2**)



6. Solve for **x**. The result cursor is displayed at the solution point, the cursor coordinate values are displayed, and the **x** value is stored in **Ans**. — [ENTER]



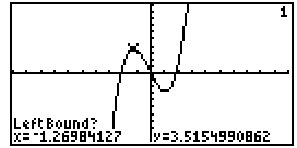
Using ∫f(x), DIST, or ARC

The steps for using ∫**f(x)**, **DIST**, and **ARC** are the same, except for the menu selection in step 1.

In the example, the function **y(x)=x^3+.3x**^{2}**−4x** *is selected. Steps 2 and 4 are not necessary here because only one function is selected.*

❶ Select **DIST** from the GRAPH MATH menu. The current graph is displayed with a **Left Bound?** prompt.

[GRAPH] [MORE] [F1] [MORE] [F4]



❷ Move the cursor onto the function on which the left bound is a point.

[▼] [▲]

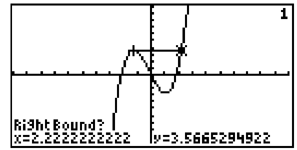
❸ Select the left bound for **x**. Either move the cursor to the left bound or enter the **x** value. **Right Bound?** is displayed.

[◄] [►] [ENTER] or *value* [ENTER]

For **DIST**, *when you are specifying the right bound, a line is drawn from the left bound to the right bound.*

❹ (**DIST** only) If you want the right bound to be a point on another function, move the cursor to the other function.

[▼] [▲]



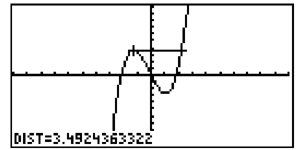
❺ Select the right bound. Either move the cursor to the right bound or enter its **x** value.

[◄] [►] or *value*

❻ Solve.

[ENTER]

- For **DIST**, the solution **DIST=** is displayed and stored in **Ans**.
- For **ARC**, the solution **ARC=** is displayed and stored in **Ans**.
- For ∫**f(x)**, the solution ∫**f(x)=** is displayed, shaded, and stored in **Ans**. The function integral error value is stored to the variable **fnIntErr** (Appendix). To remove the shading, select **CLDRW** from the GRAPH DRAW menu (page 103).



Using dy/dx or TANLN

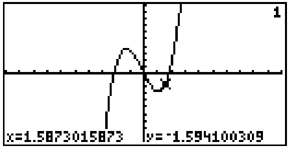
The steps for using **dy/dx** and **TANLN** are the same, except for the menu selection in step 1.

In the example, the function **y(x)=x^3+.3x²−4x** *is selected.*

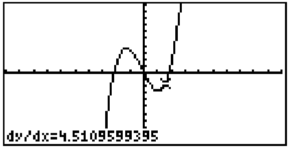
TANLN (GRAPH MATH menu) and TanLn (GRAPH DRAW menu) both draw a tangent line on the graph; only TANLN displays the solution, dy/dx.

1. Select **dy/dx** from the GRAPH MATH menu. The current graph is displayed.
   [GRAPH] [MORE] [F1] [F2]

2. Move the cursor to the function with the point for which you want to find the derivative, or slope.
   [▼] [▲]



3. Move the cursor to the point (or enter the **x** value).
   [◄] [►]
   [ENTER]



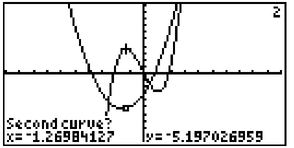
4. Solve.
   - For **dy/dx**, the solution **dy/dx=** is displayed and stored in **Ans**.
   - For **TANLN**, a tangent line also is displayed.

   To remove the tangent line and **dy/dx=** prompt, select **CLDRW** from the GRAPH DRAW menu.

In the example, the functions **y(x)=x^3+.3x²−4x** *and* **y(x)=x²+3x−3** *are selected.*

Using ISECT

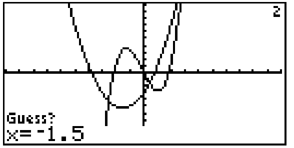
1. Select **ISECT** from the GRAPH MATH menu. The current graph is displayed with **First Curve?** at the bottom of the graph screen. — [GRAPH] [MORE] [F1] [MORE] [F3]



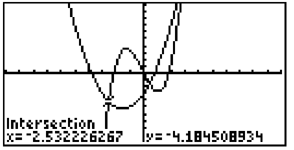
2. Select the first function (curve). The cursor moves to the next function and **Second Curve?** is displayed. — [▼] [▲] [ENTER]

3. Select the second function (curve). **Guess?** is displayed. — [▼] [▲] [ENTER]

4. Guess the intersection. Either move the cursor to a point near an intersection or enter an **x** value. — [(−)] **1** [.] **5** (or [◄] [►])



5. Solve. The result cursor is displayed at the intersection , the cursor coordinates are the result, and the **x** value is stored to **Ans**. — [ENTER]



Using YICPT

To use **YICPT**, select **YICPT** from the GRAPH MATH menu ([GRAPH] [MORE] [F1] [MORE] [F2]). Press [▼] and [▲] to select a function, and then press [ENTER]. The result cursor is displayed at the y-intercept, the cursor coordinates values are displayed, and **y** is stored in **Ans**.

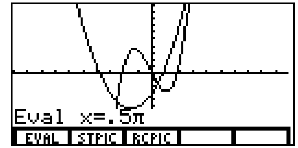
Evaluating a Function for a Specified x

To clear entered numbers from the **Eval x=** *prompt, press* [CLEAR].

To cancel **EVAL**, *press* [CLEAR] *after clearing the* **Eval x=** *prompt.*

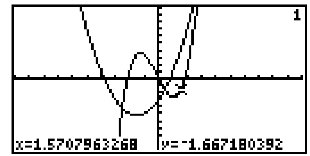
Expressions are valid for **x**.

1. Select **EVAL** from the GRAPH menu. The graph is displayed with the **Eval x=** prompt in the bottom-left corner. — [GRAPH] [MORE] [MORE] [F1]



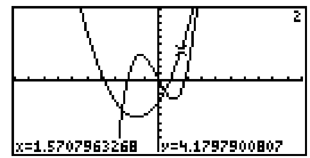
2. Enter a real **x** value between window variables **xMin** and **xMax**. — [.] **5** [2nd] [π]

3. Evaluate. The result cursor is on the first selected function at the entered **x** value. The coordinate values are displayed. The number in the top-right corner indicates which function is evaluated. — [ENTER]



You may continue to enter valid **x** *values for which to evaluate the selected functions.*

4. Move the result cursor to the next or previous selected function. The result cursor is on the next or previous function at entered **x** value, the coordinate values are displayed, and the function number changes. — [▲] [▼]



Drawing on a Graph

You can use the drawing tools (except **DrInv**) to draw points, lines, circles, shaded areas, and text on the current graph in any graphing mode. The drawing tools use the display's x- and y-coordinate values.

Before Drawing on a Graph

All drawings are temporary; they are not stored in a graph database. Any action that causes Smart Graph to replot the graph erases all drawings. Therefore, before you use any drawing tool, consider whether you want to perform any of these graphing activities first.

- Change a mode setting that affects graphs
- Select, deselect, or edit a current function or stat plot
- Change the value of a variable used in a selected function
- Change a window variable value
- Change a graph format setting or graph style
- Clear current drawings with **CLDRW**

Saving and Recalling Drawn Pictures

Graph database (GDB) and picture (PIC) variable names can be from one to eight characters long. The first character must be a letter.

To store the elements that define the current graph to a graph database (**GDB**) variable, select **STGDB** from the GRAPH menu. These information types are stored to a **GDB** variable:

- Equation editor functions
- Graph style settings
- Window variable values
- Format settings

To recall the stored **GDB** later, select **RCGDB** from the GRAPH menu, and then select the **GDB** variable from the GRAPH RCGDB menu. When you recall a **GDB**, the information stored in the **GDB** replaces any current information of these types.

The next section describes how to draw lines, points, curves, and text onto a graph; you then can store the drawings to a **PIC** *variable.*

To store the current graph display, including drawings, to a picture (**PIC**) variable, select **STPIC** from the GRAPH menu. Only the graph picture is stored to the specified **PIC** variable.

To superimpose one or more stored graph pictures onto a graph later, select **RCPIC** from the GRAPH menu, and then select the **PIC** variable from the GRAPH RCPIC menu.

Clearing Drawn Pictures

To clear drawn pictures while the graph is displayed, select **CLDRW** from the GRAPH DRAW menu. The graph is replotted and displayed with no drawn elements.

To clear drawn pictures from the home screen, select **ClDrw** from the CATALOG. **ClDrw** is pasted to the cursor location. Press [ENTER]. **Done** is displayed; when you display the graph again, no drawings are displayed.

The GRAPH DRAW Menu [GRAPH] [MORE] [F2]

*DrInv is not available in **Pol**, **Param**, or **DifEq** graphing modes.*

MATH	DRAW	FORMT	STGDB	RCGDB
Shade	LINE	VERT	HORIZ	CIRCL

▸

DrawF	PEN	PTON	PTOFF	PTCHG

▸

CLDRW	PxOn	PxOff	PxChg	PxTest

▸

TEXT	TanLn	DrInv		

You can use these GRAPH DRAW menu items only on the home screen or in the program editor.

*For **PxOn**, **PxOff**, **PxChg**, and **PxTest**, row and column are integers, where 0≤row≤62 and 0≤column≤126.*

*For **DrawF**, **TanLn**, and **DrInv**, expression is in terms of **x**. Also, you cannot include a list in expression to draw a family of curves.*

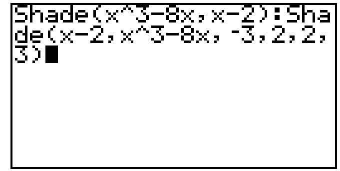
Shade(	Shades a specified area of a graph (See page 104)
DrawF *expression*	Draws *expression* as a function
PxOn(*row*,*column***)**	Turns on the pixel at **(***row*,*column***)**
PxOff(*row*,*column***)**	Turns off the pixel at **(***row*,*column***)**
PxChg(*row*,*column***)**	Changes the on/off status of the pixel at **(***row*,*column***)**
PxTest(*row*,*column***)**	Returns **1** if the pixel at **(***row*,*column***)** is on, or **0** if the pixel is off
TanLn(*expression*,*x***)**	Draws *expression* as a function and a tangent line of *expression* at *x*
DrInv *expression*	Draws the inverse of *expression*

Shading Areas of a Graph

To shade an area of a graph, the syntax is:

Shade(*lowerFunc*,*upperFunc*[,*xLeft*,*xRight*,*pattern*,*patternRes*]**)**

To replicate the example without additional graphs, turn off all equations and stat plots before entering the instructions as shown.

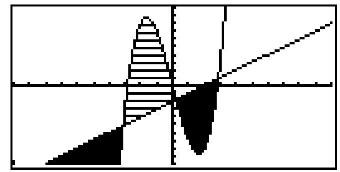


pattern specifies one of four shading patterns.

1	vertical (default)
2	horizontal
3	negative slope(45°)
4	positive slope (45°)

patternRes specifies one of eight shading resolutions.

1	every pixel (default)
2	every second pixel
3	every third pixel
4	every fourth pixel
5	every fifth pixel
6	every sixth pixel
7	every seventh pixel
8	every eighth pixel



- The area that is specifically above *lowerFunc* and below *upperFunc* is shaded.
- *xLeft* > **xMin** and *xRight* < **xMax** must be true.
- *xLeft* and *xRight* specify left and right bounds for shading. (**xMin** and **xMax** are defaults.)

These GRAPH DRAW menu items are interactive. Also, you can use all of them, except **PEN**, on the home screen or in a program (A to Z Reference).

LINE Draws a line segment from one point to another point you specify with the cursor

VERT Draws a vertical line, which you can move to any displayed **x** value

HORIZ Draws a horizontal line, which you can move to any displayed **y** value

CIRCL Draws a circle with a center point and radius you specify with the cursor

PEN Draws the path of the cursor as you move it on the graph screen

PTON Turns on the point at the cursor location

PTOFF Turns off the point at the cursor location

PTCHG Changes the on/off status of a point at the cursor location

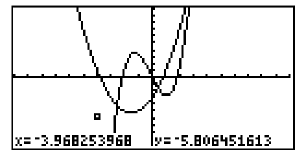
CLDRW Clears all drawings from the graph screen; replots the graph

TEXT Draws characters on the graph at the cursor location

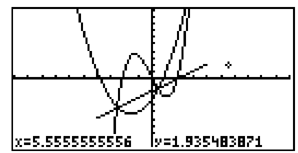
Drawing a Line Segment

In the example, the functions **y(x)=x^3+.3x2−4x** *and* **y(x)=x2+3x−3** *are selected.*

1. Select **LINE** from the GRAPH DRAW menu. The graph is displayed. [GRAPH] [MORE] [F2] [F2]
2. Define one segment endpoint with the cursor. [▶] [▼] [◀] [▲] [ENTER]



3. Define the other endpoint of the segment. As you move the cursor, a line anchored at the first defined endpoint extends to the cursor. [▶] [▼] [◀] [▲]
4. Draw the line. [ENTER]

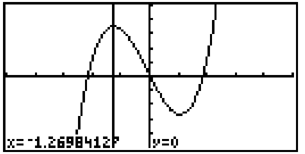


To draw more line segments, repeat steps 2 and 3; to cancel **LINE**, press [CLEAR].

Drawing a Vertical or Horizontal Line

In the example, the function **y(x)=x^3+.3x²−4x** *is selected. Also,* **ZIN** *was executed once with the zoom cursor at (0,0),* **xFact=2**, *and* **yFact=2**.

1. Select **VERT** (or **HORIZ**) from the GRAPH DRAW menu. The graph is displayed and a vertical or horizontal line is drawn at the cursor. — [GRAPH] [MORE] [F2] [F3] (or [F4])
2. Move the line to the **x** value (or to the **y** value, if horizontal) through which you want the line to pass. — [◀] [▶] (or [▲] [▼])
3. Draw the line on the graph. — [ENTER]



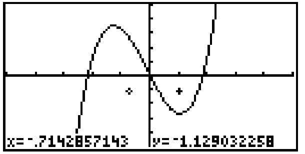
To draw more lines, repeat steps 2 and 3; to cancel **VERT** or **HORIZ**, press [CLEAR].

Drawing a Circle

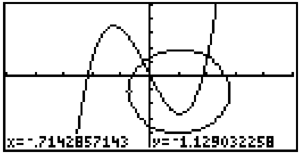
In the example, the function **y(x)=x^3+.3x²−4x** *is selected. Also,* **ZIN** *was executed once with the zoom cursor at (0,0),* **xFact=2**, *and* **yFact=2**.

Here the circle appears as a circle, regardless of window variable values. When you use **Circl(** *from the* CATALOG *to draw a circle, the current window variable values may distort the shape.*

1. Select **CIRCL** from the GRAPH DRAW menu. The graph is displayed. — [GRAPH] [MORE] [F2] [F5]
2. Define the center point of the circle with the cursor. — [▶] [▼] [◀] [▲] [ENTER]



3. Move the cursor to any point on the intended circumference. — [▶] [▼] [◀] [▲]
4. Draw the circle. — [ENTER]



To draw more circles, repeat steps 2 through 4; to cancel **CIRCL**, press [CLEAR].

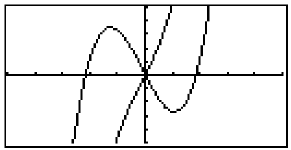
Drawing a Function, Tangent Line, or Inverse Function

For **DrawF**, **TanLn**, *and* **DrInv**, *you can use as expression any variable to which a valid expression is stored (including deselected equation variables).*

For **DrawF**, **TanLn**, and **DrInv**, *expression* is in terms of **x**. When you select **DrawF**, **TanLn**, or **DrInv** from the GRAPH DRAW menu, it is pasted to the home screen or program editor. Upon execution, the drawing is returned. **DrInv** draws the inverse of *expression* by plotting its **x** values on the y-axis and its **y** values on the x-axis. **DrInv** is available only in **Func** graphing mode.

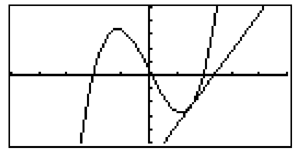
In the illustrations, **y1=x^3+.3x^2−4x** *is selected.*

DrawF *expression*



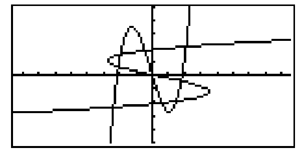
DrawF x^3+.3x^2+4x

TanLn(*expression*,*x***)**



TanLn(y1,1.5)

DrInv *expression*



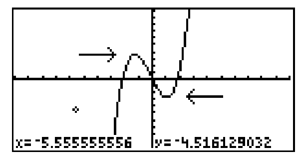
DrInv y1

Drawing Freehand Points, Lines, and Curves

In the example, the function **y(x)=x^3+.3x^2−4x** *is selected. Also,* **ZSTD** *was executed.*

To draw a diagonal line or curve, turn on the pen, press [ENTER] [ENTER], *press* [◄] [▲] *(or* [▼] [►], *and so on), and repeat.*

1. Select **PEN** from the GRAPH DRAW menu. — [GRAPH] [MORE] [F2] [MORE] [F2]
2. Move the cursor to where you want to begin drawing. — [►] [▼] [◄] [▲]
3. Turn on the pen. — [ENTER]
4. Draw whatever you want. — [►] [▼] [◄] [▲]
5. Turn off the pen. — [ENTER]



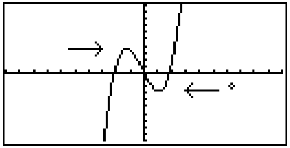
To draw more points, lines, or curves, repeat steps 2 through 5. To cancel, press [CLEAR].

Placing Text on a Graph

This example adds to the **PEN** *example drawing. Before you start, you may want to store the arrows to a picture variable (page 102).*

❶ Select **TEXT** from the GRAPH DRAW menu. The text cursor is displayed. — GRAPH MORE F2 MORE MORE MORE F1

❷ Move the cursor to where you want to enter text. Text is entered below the text cursor. — ▶ ▼ ◀ ▲

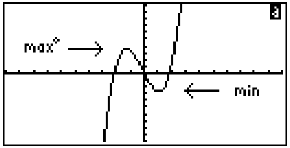


To erase a character when using **TEXT**, *move the* **TEXT** *cursor above it and then press* ALPHA [␣] *or* 2nd [alpha] [␣] *to overwrite it.*

❸ Set alpha-lock and enter **min**. (The alpha cursor (a) is displayed in the top-right corner.) — 2nd [alpha] ALPHA [M] [I] [N]

❹ Move the cursor to another location. — ▶ ▼ ◀ ▲

❺ Enter **max** (alpha-lock remains on). — [M] [A] [X]



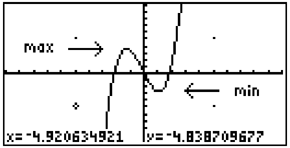
Turning On or Turning Off Points

In the example, the function **y(x)=x^3+.3x²−4x** *is selected. Also,* **ZSTD** *was executed. Points are turned on at (⁻5,5), (5,5), (5,⁻5), and (⁻5,⁻5).*

❶ Select **PTON** (or **PTOFF**) from the GRAPH DRAW menu. — GRAPH MORE F2 MORE F3

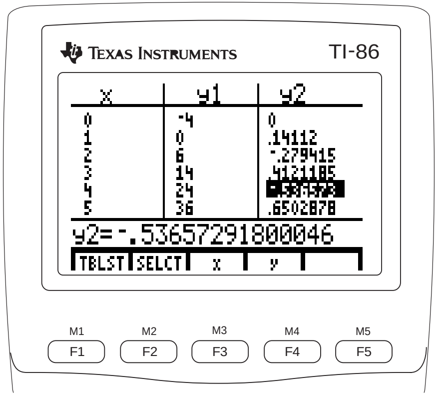
❷ Move the cursor to where you want to draw (or erase) a point. — ▶ ▼ ◀ ▲

❸ Turn on (or turn off) the point. — ENTER



To continue drawing points, repeat steps 2 and 3. To cancel **PTON**, press CLEAR.

7 Tables



Texas Instruments
TI-86
x
y1
y2
0
1
2
3
4
5
-4
0
6
14
24
36
0
.14112
-.279415
.4121185
.6502878
y2=-.53657291800046
TBLST
SELCT
x
y
M1
M2
M3
M4
M5
F1
F2
F3
F4
F5

Displaying the Table

To display the equation editor, press [GRAPH] [F1] (Chapter 5).

The table displays the independent values and corresponding dependent values for up to 99 selected functions in the equation editor. Each dependent variable in the table represents a selected function stored in the equation editor for the current graphing mode.

TABLE Menu [TABLE]

TABLE	TBLST			

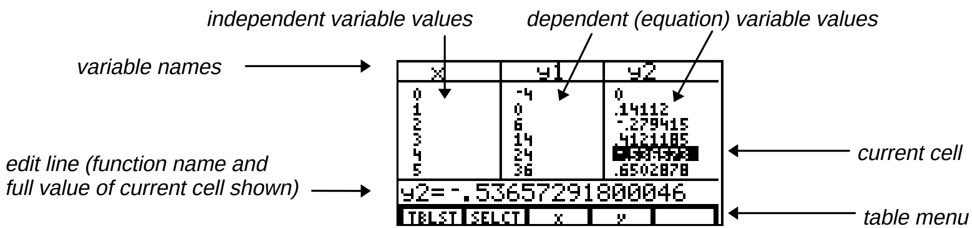
table screen

table setup editor

The Table [TABLE] [F1]

*In the example, **y1=x²+3x-4** and **y2=sin (3x)** are selected and all defaults set.*

The table abbreviates values in the columns, if necessary.



To edit an equation, press [▲] in the equation's table column until the cursor highlights the equation variable on the top line, and then press [ENTER]. The expression stored to the current equation variable is displayed in the edit line.

*In **DifEq** mode, if an equation has an initial conditions list, the table uses the first list element to evaluate the equation (Chapter 10).*

Independent and Dependent Variables in the Table

Graphing Mode	Independent Variable	Dependent (Equation) Variables
Func (function)	**x**	**y1** through **y99**
Pol (polar)	θ	**r1** through **r99**
Param (parametric)	**t**	**xt1**/**yt1** through **xt99**/**yt99**
DifEq (differential equation)	**t**	**Q1** through **Q9**

Navigating the Table

To...	Do this:
Display more dependent variables in the table	Press [▶] or [◀]
Display greater values in any column	Press [▼] (only when **Indpnt: Auto** is set; page 112)
Set **TblStart** to a lower value	Press [▲] in the independent variable column until the cursor moves past the current **TblStart** (page 112)
Display the equation in the edit line, where you can edit or deselect it	Press [◀] or [▶] to move the cursor to an equation variable column, hold [▲] until the cursor highlights the equation name, and then press [ENTER]; the equation is displayed in the edit line

The Table Menus [TABLE] [F1]

The table has a unique menu for each graphing mode, as shown below.

In Function Graphing Mode

TBLST	**SELCT**	**x**	**y**	

In Parametric Graphing Mode

TBLST	**SELCT**	**t**	**xt**	**yt**

In Polar Graphing Mode

TBLST	**SELCT**	θ	**r**	

In Differential Equation Graphing Mode

TBLST	**SELCT**	**t**	**Q**	

TBLST	Displays the table setup editor
SELCT	On the edit line, deselects or cancels deselection of the equation
x and **y**; θ and **r**; **t**, **xt**, and **yt**; or **t** and **Q**	On the edit line, pastes the variable to the cursor location; the variables change according to graphing mode

- To add an equation to the table, select it in the equation editor (Chapter 5). **SELCT** only removes equations from the table.
- To remove an equation from a column in the table, select **SELCT** from the table menu. Remaining equations that follow the removed equation shift left one column.
- To deselect an equation with **SELCT**, the equation and cursor must be displayed in the edit line. If the equation is in the edit line but the cursor is not, press [ENTER].
- To compare two dependent variables not defined consecutively in the equation editor, use **SELCT** from the table screen menu to deselect the dependent variables in between.

Setting Up the Table

To display the table using the current table setup settings, select **TABLE** *from the* TABLE *menu.*

To display the table setup editor, select **TBLST** from the TABLE menu. The screen to the right shows the default table setup settings.

TblStart specifies the first independent variable value (**x**, θ, or **t**) in the table (only when **Indpnt: Auto** is selected).

TblStart *and* **ΔTbl** *must be real numbers; you can enter an expression.*

ΔTbl (table step) specifies the increment or decrement from one independent variable value to the next independent variable value in the table.

- If **ΔTbl** is positive, then the values of **x**, θ, or **t** increase as you scroll down the table.
- If **ΔTbl** is negative, then the values of **x**, θ, or **t** decrease as you scroll down the table.

In **DifEq** *graphing mode, it is a good practice to set* **TblStart** = **tMin** *and* **ΔTbl** = **tStep**.

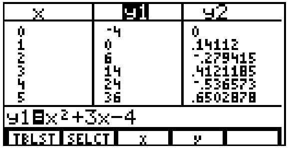
Indpnt: Auto displays independent variable values automatically in the first column of the table, starting at **TblStart**.

Indpnt: Ask displays an empty table. As you enter **x** values in the **x=** prompt (**x=***value* [ENTER]), each value is added to the independent variable column and the corresponding dependent variable values are calculated and displayed. When **Ask** is set, you cannot scroll beyond the six independent variable values that are currently displayed in the table.

Viewing and Editing Dependent Variable Equations

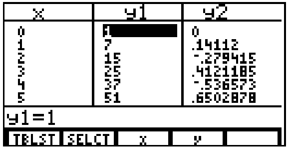
In the example, **y1=x²+3x-4** *and* **y2=sin (3x)** *are selected and all defaults set.*

1. Display the table. — [TABLE] [F1]
2. Move the cursor into the column of the dependent variable you want to edit, and then move up the column until the name is highlighted. — [▶] [▲]



When you display the equation in the edit line, the column equation name is highlighted.

3. Display the equation in the edit line. — [ENTER]
4. Edit the equation. — [▶] [▶] [▶] **5** [▶] [+] **1**
5. Enter the edited equation. — [ENTER]
   - ♦ The dependent variable values are recalculated.
   - ♦ The cursor returns to the edited dependent variable's first value.
   - ♦ The equation editor is updated.

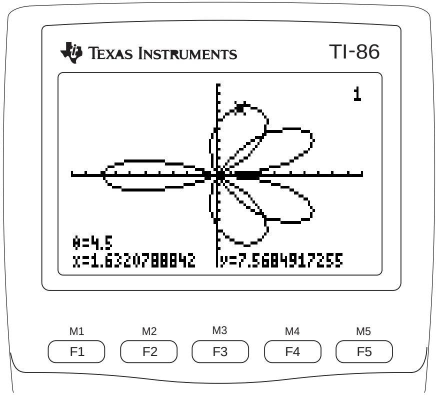


Clearing the Table

When you use **ClTbl** *in a program, the table is cleared upon program execution (Chapter 16).*

To clear the table when **Indpnt: Ask** is set, select **ClTbl** from the CATALOG, and then press [ENTER]. All independent and dependent variable columns are cleared. **ClTbl** does nothing when **Indpnt: Auto** is set.

8 Polar Graphing

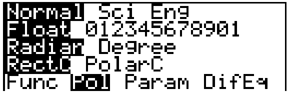


Preview: Polar Graphing

The graph of the polar equation A sin (Bθ) forms the shape of a flower. Graph the flower for A=8 and B=2.5. Then explore the appearance of the flower for other values of A and B.

❶ Select **Pol** mode from the mode screen.

[2nd] [MODE] [▼] [▼] [▼] [▼] [▶] [ENTER]

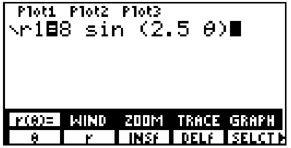


❷ Display the equation editor and polar equation editor menu.

[GRAPH] [F1]

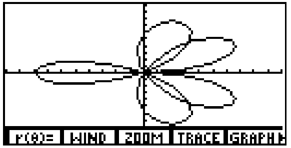
❸ (Deselect or delete all equations if any.) Store **r1(θ)=8sin(2.5θ)**.

([MORE] [F2] [MORE])
8 [SIN] [(] **2** [.] **5** [F1] [)]



❹ Select **ZSTD** from the GRAPH ZOOM menu. **r1** is plotted on the graph screen.

[2nd] [M3] [F4]



❺ Display the window editor, and then change θ**Max** to **4**π.

[F2]
[▼] **4** [2nd] [π]

❻ Select **ZSQR** from the GRAPH ZOOM menu. **xMin** and **xMax** are changed to display the graph in correct proportion.

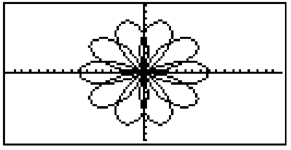
[F3] [MORE] [F2]

To remove the GRAPH *menu from the graph screen, as shown, press* [CLEAR].

To redisplay the GRAPH *menu, press* [GRAPH].

❼ Change the values of A and B and redisplay the graph.

[F1] (enter other A and B values)



Defining a Polar Graph

The steps for defining a polar graph are similar to the steps for defining a function graph. This chapter assumes that you are familiar with Chapter 5: Function Graphing and Chapter 6: Graph Tools. Chapter 8 details aspects of polar graphing that differ from function graphing.

Setting Polar Graphing Mode

To display the mode screen, press [2nd] [MODE]. To graph polar equations, you must select **Pol** graphing mode before you enter equations, set the format, or edit window variable values. The TI-86 retains separate equation, format, and window data for each graphing mode.

The GRAPH Menu [GRAPH]

Chapter 5 describes these GRAPH *menu items:* **GRAPH** *and* **FORMT***.*

Chapter 6 describes these GRAPH *menu items:* **ZOOM**, **TRACE**, **DRAW**, **STGDB**, **RCGDB**, **EVAL**, **STPIC**, *and* **RCPIC***.*

r(θ)=	WIND	ZOOM	TRACE	GRAPH

▸

MATH	DRAW	FORMT	STGDB	RCGDB

▸

EVAL	STPIC	RCPIC		

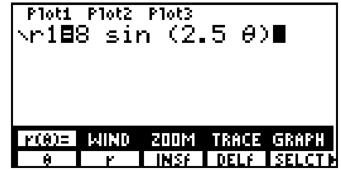
r(θ)=: polar equation editor

WIND: polar window editor

MATH: polar graph math menu

Displaying the Polar Equation Editor

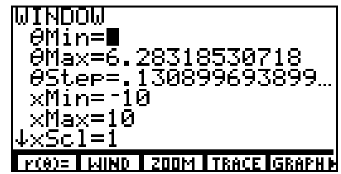
To display the polar equation editor, select **r(θ)=** from the GRAPH menu in **Pol** graphing mode ([GRAPH] [F1]). The polar equation editor menu displayed on the bottom line is the same as the **Func** mode equation editor menu, except that θ and **r** replace **x** and **y**.



In this editor, you can enter and display up to 99 polar equations, **r1** through **r99**, if sufficient memory is available. Equations are defined in terms of the independent variable θ.

The default graph style is ⋱ (line) in **Pol** graphing mode. ◥ (shade above) and ◣ (shade below) graph styles are not available in **Pol** graphing mode.

Setting the Graph Screen Window Variables



To display the polar window editor, select **WIND** from the GRAPH menu ([GRAPH] [F2]). **Pol** graphing mode has the same window variables as **Func** graphing mode, except:

- **xRes** is not available in **Pol** graphing mode.
- θ**Min**, θ**Max**, and θ**Step** are available in **Pol** graphing mode.

The values shown in the picture to the right are the defaults in **Radian** mode. ↓ indicates that **yMin=-10**, **yMax=10**, and **yScl=1** are beyond the screen.

θ**Max** *default is 2π.*

θ**Step** *default is π/24.*

θMin=0	Specifies the first θ value to evaluate within the graph screen
θMax=6.28318530718	Specifies the last θ value to evaluate within the graph screen
θStep=.13089969389957	Specifies the increment from one θ value to the next θ value

DrawLine *graph format typically displays a more meaningful polar graph than* **DrawDot** *graph format.*

Setting the Graph Format

To display the format screen in **Pol** graphing mode, select **FORMT** from the GRAPH menu ([GRAPH] [MORE] [F3]). Chapter 5 describes the format settings. Although the same settings are available for **Func**, **Pol**, and **Param** graphing modes, the TI-86 retains in memory separate format settings for each mode. In **Pol** graphing mode, **PolarGC** shows the cursor coordinates in terms of **r** and θ, the variables that define the equations.

Displaying the Graph

To plot the selected polar equations, you can select **GRAPH**, **TRACE**, **EVAL**, **RCGDB**, or a **ZOOM**, **MATH**, **DRAW**, or **RCPIC** operation, from the GRAPH menu. The TI-86 evaluates **r** for each value of θ (from θ**Min** to θ**Max** in intervals of θ**Step**) and then plots each point. As the graph is plotted, the variables θ, **r**, **x**, and **y** are updated.

Using Graph Tools in Pol Graphing Mode

The Free-Moving Cursor

The free-moving cursor in **Pol** graphing works the same as in **Func** graphing.

- In **RectGC** format, moving the cursor updates the values of **x** and **y**; if **CoordOn** format is selected, **x** and **y** are displayed.
- In **PolarGC** format, moving the cursor updates **x**, **y**, **r**, and θ; if **CoordOn** format is selected, **r** and θ are displayed.

Tracing a Polar Equation

To begin a trace, select **TRACE** from the GRAPH menu (press [GRAPH] [F4]). The trace cursor appears on the first selected equation at θ**Min**.

- In **RectGC** format, moving the trace cursor updates the values of θ, **x**, and **y**; if **CoordOn** format is selected, θ, **x**, and **y** are displayed.
- In **PolarGC** format, moving the trace cursor updates **x**, **y**, **r**, and θ; if **CoordOn** format is selected, **r** and θ are displayed.

To move the trace cursor...	Press:
Along the graph of the equation by increments or decrements of θ**Step**	[▶] or [◀]
From one equation to another	[▼] or [▲]

QuickZoom is available in **Pol** *graphing; panning is not (Chapter 6).*

If you move the trace cursor beyond the top or bottom of the graph screen, the coordinate values at the bottom of the screen continue to change appropriately.

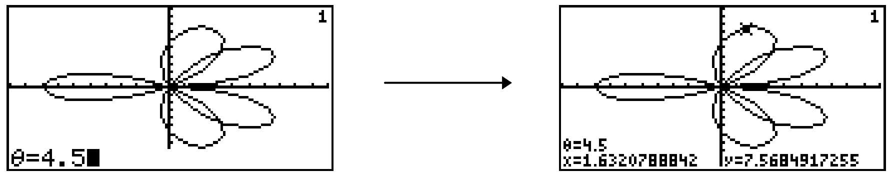
If you have graphed a family of curves, [▼] and [▲] move through each curve before moving to the next polar equation.

Moving the Trace Cursor to a θ Value

To move the trace cursor to any valid θ value on the current equation, enter the number. When you enter the first digit, a **θ=** prompt is displayed in the bottom-left corner. The value you enter must be valid for the current graph screen. When you have completed the entry, press [ENTER] to reactivate the trace cursor.

In the example, **r1=8sin(2.5θ)** *is graphed.*

Values for θ, **x**, *and* **y** *are displayed on the graph to the right because* **RectGC** *graph format is selected.*



Using Zoom Operations

The GRAPH ZOOM menu items, except **ZFIT**, work the same in **Pol** graphing as in **Func** graphing. In **Pol** graphing mode, **ZFIT** adjusts the graph screen in both the x and y directions.

The zoom operations affect only the **x** window variables (**xMin**, **xMax**, and **Xscl**) and the **y** window variables (**yMin**, **yMax**, and **yScl**), except **ZSTO** and **ZRCL**, which also affect the θ window variables (θ**Min**, θ**Max**, and θ**Step**).

The GRAPH MATH Menu [GRAPH] [MORE] [F1]

MATH	DRAW	FORMT	STGDB	RCGDB
DIST	dy/dx	dr/dθ	ARC	TANLN

The other GRAPH MATH *menu items are the same as described in Chapter 6.*

dr/dθ Finds the numerical derivative (slope) of a function at a point

The distances calculated by **DIST** and **ARC** are distances in the rectangular coordinate plane. **dy/dx** and **dr/dθ** are independent of the **RectGC** or **PolarGC** format.

At a point where the derivative is undefined, **TANLN** will draw the line, but no result is displayed or stored in **Ans**.

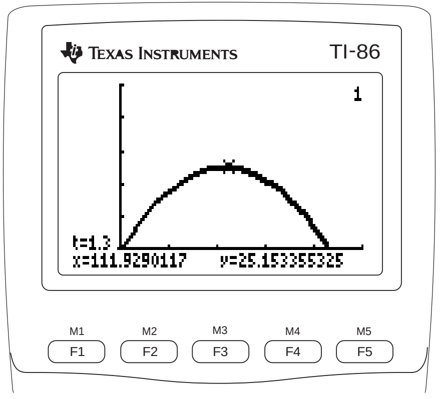
Evaluating an Equation for a Specified θ

When the trace cursor is not active, the GRAPH menu item **EVAL** evaluates selected polar equations directly on the graph for a given value of θ. **eval** in a program or from the home screen returns a list of **r** values.

Drawing on a Polar Graph

The GRAPH DRAW menu items work the same in **Pol** graphing as in **Func** graphing. DRAW instruction coordinates in **Pol** graphing mode are the x- and y-coordinates of the graph screen. **DrInv** is not available in **Pol** graphing mode.

9 Parametric Graphing



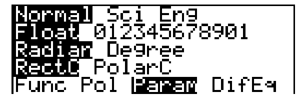
Preview: Parametric Graphing

Graph the parametric equation that describes the path of a ball kicked at an initial speed of 30 meters per second, at an initial angle of 25 degrees with the horizontal (from ground level). How far does the ball travel? When does it hit the ground? How high does it go?

In the example, ignore all forces except gravity. For initial velocity v_0 and angle θ, the position of the ball as a function of time has horizontal and vertical components.

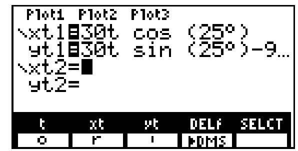
❶ Select **Param** mode from the mode screen.

[2nd] [MODE] [▼] [▼] [▼] [▼] [▶] [▶] [ENTER]



❷ Display the equation editor and parametric equation editor menu. Deselect all equations and plots (if any are defined).

[GRAPH] [F1] ([MORE] [F2] [MORE])

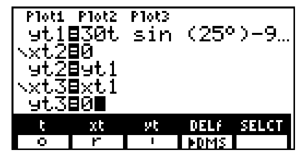


❸ Define the path of the ball as **xt1** and **yt1** in terms of **t**.
Horizontal: xt1=t$v_0$cos(θ)
Vertical: yt1=t$v_0$sin(θ)−1/2(gt^2)
Gravity constant: g=9.8 m/sec^2

30 [F1] [COS] [(] **25** [2nd] [MATH] [F3] [F1] [)] [▼] **30** [2nd] [M1] [SIN] [(] **25** [F1] [)] [−] **9** [.] **8** [÷] **2** [2nd] [M1] [x^2] [▼]

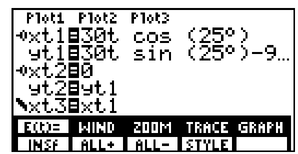
❹ Define the vertical component vector as **xt2** and **yt2** and define the horizontal component vector as **xt3** and **yt3**.

0 [▼] [2nd] [M3] **1** [▼] [2nd] [M2] **1** [▼] **0**



❺ Change the graph style of **xt3**/**yt3** to ◥ (thick). Change the graph style of **xt2**/**yt2** and **xt1/yt1** to -0 (path).

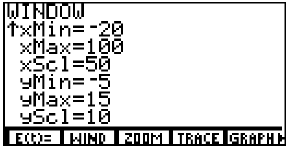
[EXIT] [MORE] [F4] [▲] [▲] [F4] [F4] [▲] [▲] [▲] [F4] [F4]



6. Enter these window variable values.

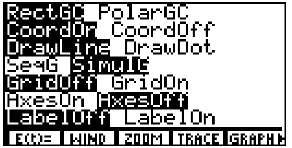
tMin=0	**xMin=⁻20**	**yMin=⁻5**	
tMax=5	**xMax=100**	**yMax=15**	
tStep=.1	**xScl=50**	**yScl=10**	

[2nd] [M2] **0** [▼] **5** [▼] [.] **1** [▼] [(-)] **20** [▼] **100** [▼] **50** [▼] [(-)] **5** [▼] **15** [▼] **10**



7. Set **SimulG** and **AxesOff** graphing formats, so the path of the ball and the vectors will be plotted simultaneously on a clear graph screen.

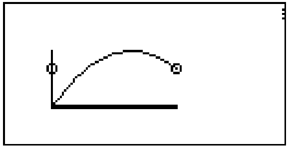
[MORE] [F3] [▼] [▼] [▼] [▶] [ENTER] [▼] [▼] [▶] [ENTER]



To simulate the ball in flight, change the graph style of **xt1/yt1** *to (animate).*

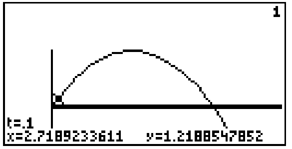
8. Plot the graph. The plotting action simultaneously shows the ball in flight and the vertical and horizontal component vectors of the motion.

[F5]



9. Trace the graph to obtain numerical results. Tracing begins at **tMin** and traces the path of the ball over time. The value displayed for **x** is distance; **y** is height; **t** is time.

[F4] [▶]



Defining a Parametric Graph

The steps for defining a parametric graph are similar to the steps for defining a function graph. This chapter assumes that you are familiar with Chapter 5: Function Graphing and Chapter 6: Graph Tools. This chapter details those aspects of parametric graphing that differ from function graphing.

Setting Parametric Graphing Mode

To display the mode screen, press [2nd] [MODE]. To graph parametric equations, you must select **Param** graphing mode before you enter equations, set the format, or edit window variable values. The TI-86 retains in memory separate equation, format, and window data for each graphing mode.

The GRAPH Menu [GRAPH]

Chapter 5 describes these GRAPH *menu items:* **GRAPH** *and* **FORMT**.

Chapter 6 describes these GRAPH *menu items:* **ZOOM**, **TRACE**, **DRAW**, **STGDB**, **RCGDB**, **EVAL**, **STPIC**, *and* **RCPIC**.

E(t)=	WIND	ZOOM	TRACE	GRAPH	▸	MATH	DRAW	FORMT	STGDB	RCGDB
					▸	EVAL	STPIC	RCPIC		

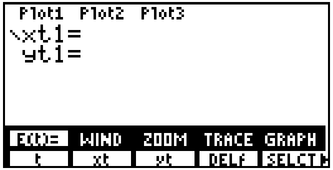
E(t)=: parametric equation editor
WIND: parametric window editor
MATH: parametric graph math menu

Displaying the Parametric Equation Editor

To display the parametric equation editor, select **E(t)=** from the GRAPH menu in **Param** graphing mode ([GRAPH] [F1]). The equation editor menu displayed on the bottom line is the same as the **Func**-mode equation editor menu, except that **t** and **xt** replace **x** and **y**, and **yt** displaces **INSf**.

A common application of parametric graphs is graphing equations over time.

In this editor, you can enter and display both the x and y components of up to 99 parametric equations, **xt1** and **yt1** through **xt99** and **yt99**, if sufficient memory is available. Each is defined in terms of the independent variable **t**.



Two components, **x** and **y**, define a single parametric equation. You must define both **xt** and **yt** for each equation. The default graph style is (line) in **Param** mode. (shade above) and (shade below) graph styles are not available in **Param** mode.

Selecting and Deselecting a Parametric Equation

When a parametric equation is selected, the equals signs (**=**) of both **xt** and **yt** are highlighted. To change the selection status of a parametric equation, move the cursor onto either **xt** or **yt**, and then select **SELCT** from the equation editor menu. The status is changed for **xt** and **yt**.

Deleting a Parametric Equation

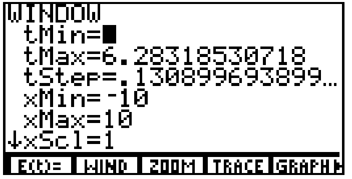
To delete a parametric equation using **DELf**, move the cursor to either **xt** or **yt**, and then select **DELf** from the equation editor menu. Both components are deleted.

To delete a parametric equation using the MEM DELET menu (Chapter 17), you must select the **xt** component. If you select the **yt** component, the equation is retained in memory.

Setting the Graph Screen Window Variables

To display the parametric window editor, select **WIND** from the GRAPH menu ([GRAPH] [F2]). **Param** graphing mode has the same window variables as **Func** graphing mode, except:

- **xRes** is not available in **Param** mode.
- **tMin**, **tMax**, and **tStep** are available in **Param** mode.



The values shown in the picture to the right are the defaults in **Radian** mode. ↓ indicates that **yMin=⁻10**, **yMax=10**, and **yScl=1** are beyond the screen.

tMax *default is 2π.*

tStep *default is π/24.*

tMin=0	Specifies the starting **t** value
tMax=6.28318530718	Specifies the ending **t** value
tStep=.13089969389957	Specifies the increment from one **t** value to the next

DrawLine *graph format typically displays a more meaningful parametric graph than* **DrawDot** *graphing format.*

Setting the Graph Format

To display the format screen in **Param** graphing mode, select **FORMT** from the GRAPH menu ([GRAPH] [MORE] [F3]). Chapter 5 describes the format settings. The TI-86 retains in memory separate format settings for **Func**, **Pol**, **Param** , and **DifEq** graphing modes.

Displaying the Graph

To plot the selected parametric equations, you can select **GRAPH**, **TRACE**, **EVAL**, **RCGDB**, or a **ZOOM**, **MATH**, **DRAW**, or **RCPIC** operation. The TI-86 evaluates **x** and **y** for each value of **t** (from **tMin** to **tMax** in intervals of **tStep**) and then plots each point defined by **x** and **y**. As the graph is plotted, the variables **x**, **y**, and **t** are updated.

Using Graph Tools in Param Graphing Mode

The Free-Moving Cursor

The free-moving cursor in **Param** graphing works the same as in **Func** graphing.

♦ In **RectGC** format, moving the cursor updates the values of **x** and **y**.; if **CoordOn** format is selected, **x** and **y** are displayed.

♦ In **PolarGC** format, moving the cursor updates **x**, **y**, **r**, and θ; if **CoordOn** format is selected, **r** and θ are displayed.

Tracing a Parametric Function

To begin a trace, select **TRACE** from the GRAPH menu ([GRAPH] [F4]). When you begin a trace, the trace cursor is on the first selected function at **tMin**.

♦ In **RectGC** format, moving the trace cursor updates the values of **x**, **y**, and **t**; if **CoordOn** format is selected, **t**, **x**, and **y** and are displayed.

- In **PolarGC** format, moving the trace cursor updates **x**, **y**, **r**, θ, and **t**; if **CoordOn** format is selected, **r**, θ, and **t** are displayed. The **x** and **y** (or **r** and θ) values are calculated from **t**.

To move the trace cursor...	Press:
Along the graph of the equation by increments or decrements of **tStep**	▸ or ◂
From one equation to another	▾ or ▴

QuickZoom is available in **Param** *graphing; panning is not (Chapter 6).*

If you move the trace cursor beyond the top or bottom of the graph screen, the coordinate values at the bottom of the screen continue to change appropriately. If you have graphed a family of curves, ▾ and ▴ move through each curve before moving to the next parametric function.

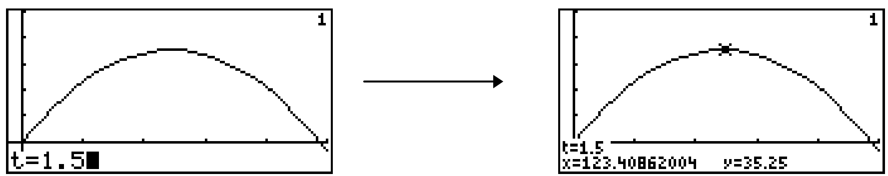
Moving the Trace Cursor to a t Value

You can enter an expression at the **t=** *prompt.*

To move the trace cursor to any valid **t** value on the current equation, enter the number. When you enter the first digit, a **t=** prompt is displayed in the bottom-left corner. The value you enter must be valid for the current graph screen. When you have completed the entry, press [ENTER] to reactivate the trace cursor.

In the example, the parametric equation is:
xt1=95t cos 30°
yt1=95t sin 30°−16t²
Also, **AxesOn** *graph format is set.*

(The example on page 124 is similar to this example.)



Using Zoom Operations

The GRAPH ZOOM menu items, except **ZFIT**, work the same in **Param** graphing as in **Func** graphing. In **Param** mode, **ZFIT** adjusts the graph screen in both the x and y directions.

The GRAPH ZOOM menu items affect only the **x** window variables (**xMin**, **xMax**, and **xScl**) and the **y** window variables (**yMin**, **yMax**, and **yScl**), except **ZSTO** and **ZRCL**, which also affect the **t** window variables (**tMin**, **tMax**, and **tStep**).

The GRAPH MATH Menu [GRAPH] [MORE] [F1]

MATH	DRAW	FORMT	STGDB	RCGDB
DIST	dy/dx	dy/dt	dx/dt	ARC

▸

TANLN				

The other GRAPH MATH *menu items are the same as described in Chapter 5.*

dy/dx Returns the derivative of **yt** divided by the derivative of **xt**

dy/dt Returns the derivative of the **yt** equation at a point with respect to **t**

dx/dt Returns the derivative of the **xt** equation at a point with respect to **t**

The distances calculated by **DIST** and **ARC** are distances in the rectangular coordinate plane.

At a point where the derivative is undefined, **TANLN** will draw the line, but no result is displayed or stored in **Ans**.

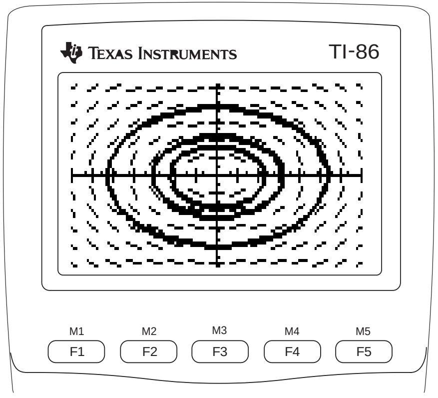
Evaluating an Equation for a Specified t

When the trace cursor is not active, the GRAPH menu item **EVAL** evaluates selected polar equations directly on the graph for a given value of **t**. **eval** in a program or from the home screen returns a list of x and y values in this form: **{***xt1(t) yt1(t) xt2(t) xt2(t)* ...**}**.

Drawing on a Parametric Graph

The DRAW menu items work in **Param** graphing the same as in **Func** graphing. DRAW instruction coordinates in **Param** graphing are the **x**- and **y**-coordinate values of the graph screen.

10 Differential Equation Graphing



Defining a Differential Equation Graph

Chapters 8 and 9 each begin with an example; Chapter 10 has several differential equation examples throughout the chapter.

Most steps for defining a differential equation graph are similar to the steps for defining a function graph. This chapter assumes that you are familiar with Chapter 5: Function Graphing and Chapter 6: Graph Tools. This chapter details aspects of differential equation graphing that differ from function graphing.

Generally, **DifEq** graphing mode differs from other graphing modes in these ways.

- ♦ You must select the field format or accept the default before defining the equations (page 133).
- ♦ If an equation is higher than first order, you must convert it to an equivalent system of first-order differential equations, and then store the system in the equation editor (page 140 and page 142).
- ♦ When **FldOff** field format is selected, you must set initial conditions for each equation in the system (page 136).
- ♦ After you have selected the field format setting, you must select **AXES** from the GRAPH menu and enter axes information or accept the defaults (page 137).

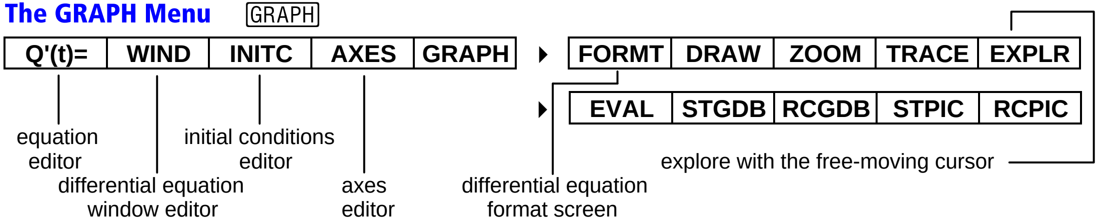
Setting Differential Equation Graphing Mode

To display the mode screen, press [2nd] [MODE]. To graph differential equations, you must select **DifEq** graphing mode before you set the format, enter equations, or edit window variable values. The TI-86 retains in memory separate format, equation, and window data for each graphing mode.

Chapter 5 describes the GRAPH *menu item* **GRAPH**.

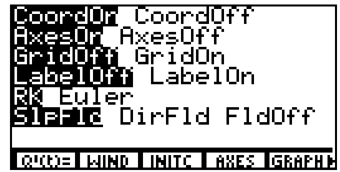
Chapter 6 describes these GRAPH *menu items:* **DRAW**, **ZOOM**, **TRACE**, **EVAL**, **STGDB**, **RCGDB**, **STPIC**, *and* **RCPIC**.

The GRAPH Menu [GRAPH]



Setting the Graph Format

To display the format screen in **DifEq** graphing mode, select **FORMT** from the GRAPH menu ([GRAPH] [MORE] [F1]).



- The **RK** **Euler** and **SlpFld** **DirFld** **FldOff** format settings are available only in **DifEq** mode.
- The **RectGC** **PolarGC**, **DrawLine** **DrawDot**, and **SeqG** **SimulG** format settings are not available in **DifEq** graphing mode.
- All other format settings are the same as described in Chapter 5.

The TI-86 retains independent format settings for each graphing mode.

Solution Method Format

RK Uses the Runge-Kutta method to solve differential equations more accurately than the **Euler** solution method format, but not as fast

Euler Uses the Euler method to solve differential equations; requires a number of iterations between **tStep** values, so **EStep=** prompt replaces **difTol=** prompt on the window editor

Field Format

SlpFld (slope field) Adds the slope field to the graph of only one first-order equation with **t** on the x-axis and a specified **Q***n* equation on the y-axis

DirFld (direction field) Adds the direction field to the graph of only one second-order equation with **Q***x*# on the x-axis and **Q***y*# on the y-axis

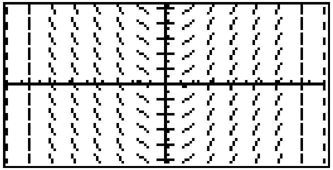
FldOff (field off) Graphs all selected differential equations with **t** or **Q1** on the x-axis, **Q1** or **Q2** on the y-axis, and no field; initial conditions must be defined for all equations (page 136)

The examples below show the basic slope and direction fields; all unspecified settings and values are defaults. To replicate these examples, reset defaults, enter the specified information in **DifEq** graphing mode, and then press [GRAPH] [F5].

Axes information is stored to **GDB** *and* **PIC** *variables.*

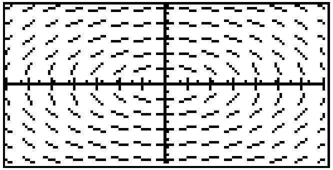
To remove menus from a graph, as shown in the examples, press [CLEAR].

SlpFld *field format*



Q'1=t *(y'=x)*

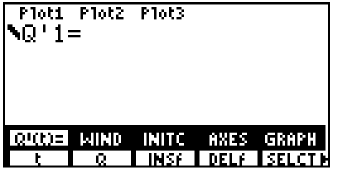
DirFld *field format*



Q'1=Q2 *and* **Q'2=-Q1** *(y''=-y)*

Displaying the Differential Equation Editor

To display the differential equation editor, select **Q'(t)=** from the GRAPH menu in **DifEq** graphing mode ([GRAPH] [F1]). The **DifEq** equation editor menu on the bottom line is the same as the **Func** mode equation editor menu, except that **t** and **Q** replace **x** and **y**.



In this editor, you can enter and display a system of up to nine first-order differential equations, **Q'1** through **Q'9**, if sufficient memory is available. Equations are defined in terms of the independent variable **t** and/or **Q'**.

You can refer to another differential equation variable in a **DifEq** equation, as in **Q'2=Q1**. However, you cannot enter a list in a **DifEq** equation.

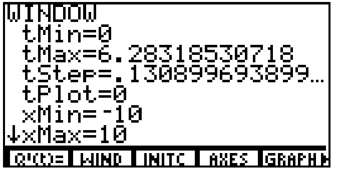
When the TI-86 calculates a differential equation system, it references all equations in the equation editor, regardless of selection status, starting at **Q'1**. You must define **Q'***n* equation variables consecutively, starting at **Q'1**. For example, if **Q'1** and **Q'2** are not defined, but you attempt to solve an equation defined in **Q'3**, the calculator returns an error.

The TI-86 allows you to analyze each equation independently. For example, you can enter **Q'1=t** and **Q'2=t²** and analyze each equation independently.

The TI-86 graphs only those selected equations that are appropriate for the specified axes.

- ♦ The default graph style is (thick) in **DifEq** mode.
- ♦ (shade above), (shade below), and (dot) are not available in **DifEq** graphing mode.

Setting the Graph Screen Window Variables



To display the differential equation window editor, select **WIND** from the GRAPH menu ([GRAPH] [F2]). **DifEq** has the same window variables as **Func** graphing mode, except:

- ♦ **xRes** is not available in **DifEq** mode.
- ♦ **tMin**, **tMax**, **tStep**, and **tPlot** are available in **DifEq** mode.
- ♦ **difTol** (**RK**) and **EStep** (**Euler**) are available in **DifEq** mode.

The values shown in the picture on page 135 are defaults in **Radian** mode. **x** and **y** settings correspond to the axes variables (page 137). ↓ indicates that **xScl=1**, **yMin=⁻10**, **yMax=10**, **yScl=1**, and **difTol=.001** (in **RK** format) or **EStep=1** (in **Euler** format) are beyond the screen.

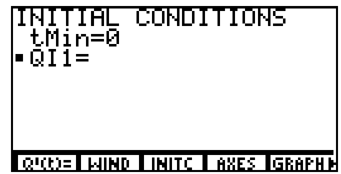
tMax *default is 2π.*

tStep *default is π/24.*

tMin=0	Specifies the **t** value at which to begin evaluating within a graph screen
tMax=6.28318530718	Specifies the last **t** value to evaluate within a graph screen
tStep=.1308969389958	Specifies the increment from one **t** value to the next **t** value
tPlot=0	Specifies the point at which plotting begins (ignored when **t** is an axis)
difTol=.001 (in **RK** format)	Specifies tolerance to help select step size for solving; must be ≥ 1**E**⁻12
EStep=1 (in **Euler** format)	Specifies Euler iterations between **tStep** values; must be an integer >0 and ≤ 25

Setting the Initial Conditions

Initial conditions information is stored to **GDB** *and* **PIC** *variables.*



To display the initial conditions editor, select **INITC** from the GRAPH menu ([GRAPH] [F3]). On this editor, you can set the initial value at **t=tMin** for each first-order equation in the equation editor.

tMin is the first **t** value to evaluate. **QI1** is the initial value of **Q***n*. A small square next to an initial condition variable indicates that a value is required for a defined differential equation.

You can enter an expression, list, or list name for initial conditions **tMin** and **QI***n*. When you enter a list name, the elements are displayed when you press [ENTER], [▼] or [▲].

- If **SlpFld** or **DirFld** format is set, you need not specify initial conditions. The TI-86 returns the appropriate field with no specific solutions.
- If **FldOff** format is set, you **must** specify initial conditions.

Setting the Axes

To display the axes editor, select **AXES** from the GRAPH menu in **DifEq** mode ([GRAPH] [F4]).

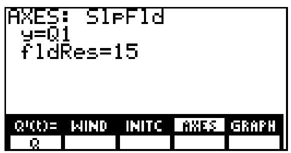
x= assigns a variable to the x-axis
y= assigns a variable to the y-axis
dTime= specifies a point in time (real number)
fldRes= (resolution) sets number of rows (1 through 25)

At the **x=** and **y=** prompts, you can enter the independent variable **t**, as well as **Q**, **Q'** , **Q**n, or **Q'**n, where n is an integer ≥ 1 and ≤ 9. If you assign **t** to one axis and **Q**n or **Q'**n to the other axis, only the equation stored to **Q**n or **Q'**n is plotted; other differential equations in the equation editor are not plotted; their selection status is ignored. **dTime** is only valid for second-order equations with **t** in either equation.

The axes editor and defaults for each field format are shown below. When **SlpFld** field format is set, the x-axis is always **t**, so the AXES: SlpFld editor does not display **x=t**.

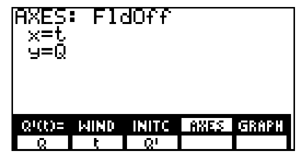
Axes information is stored to **GDB** *and* **PIC** *variables.*

When **SlpFld** format is set:



When **DirFld** format is set:

When **FldOff** format is set:



Differential Equation Graphing Tips

- Since the TI-86 plots slope fields and direction fields before it plots equations, you can press [ENTER] to pause the graph and view the fields with no solutions plotted.
- If you do not specify initial conditions for the equations assigned to the axes, the TI-86 simply draws the field and stops. This gives you access to both the field format options and the interactive initial conditions simultaneously.

The Built-In Variable fldPic

Stat plot and screen drawings are not stored to **fldPic**.

As the TI-86 plots a field, it stores the field and any displayed label, axes, or cursor coordinate information to the built-in variable **fldPic**.

These actions do not update **fldPic**.

- Switching the solving method format from **RK** to **Euler** or from **Euler** to **RK**
- Entering or editing any initial condition variable value (**QI1** through **QI9**)
- Editing a value for **difTol**, **EStep**, **tMin**, **tMax**, **tStep**, or **tPlot**
- Changing a graph style

These actions update **fldPic**.

- Editing an equation in the equation editor
- Re-assigning an axis, editing a **dTime** value, or editing a **fldRes** value
- Using a GRAPH ZOOM menu item
- Changing a format setting other than solving method format
- Editing a value for **xMin**, **xMax**, **xScl**, **yMin**, **yMax**, or **yScl**

Displaying the Graph

To plot the differential equations, you can select **GRAPH**, **TRACE**, **EVAL**, or **STGDB**, or a **DRAW**, **ZOOM**, or **STPIC** operation, from the GRAPH menu. The TI-86 solves each equation from **tMin** to **tMax**. If **t** is not an axis, it plots each point beginning at **tPlot**; otherwise, it begins at **tMin**. As a graph is plotted, the variables **x**, **y**, **t**, and **Q**n are updated.

tStep affects trace resolution and graph appearance, but not the accuracy of the trace values. **tStep** does not determine the step size for solving; using the **RK** algorithm (Runge-Kutta 2-3) determines the step size. If the x-axis is **t**, setting **tStep**<(**tMax** − **tMin**)/126 increases plotting time without increasing accuracy.

Entering and Solving Differential Equations

In **Func** graphing mode, **x** is the independent variable and **y** is the equation variable. To avoid conflict between **Func** equations and **DifEq** equations on the TI-86, **t** is the independent variable and **Q'**n is the equation variable in **DifEq** graphing mode. Therefore, when you enter an equation in the differential equation editor, you must express it in terms of **t** and **Q'**n.

For example, to express the first-order differential equation $y'=x^2$, you would substitute **t²** for x^2 and **Q'1** for y', and then enter **Q'1=t²** in the equation editor.

Graphing in SlpFld Format

In the example, the default window variable values are set initially.

❶ Display the mode screen and set **DifEq** graphing mode.

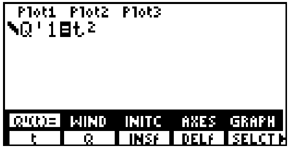
[2nd] [MODE] [▼] [▼] [▼] [▼] [▶] [▶] [▶] [ENTER]

❷ Display the format screen and set **SlpFld** field format.

[GRAPH] [MORE] [F1] [▼] [▼] [▼] [▼] [▼] [ENTER]

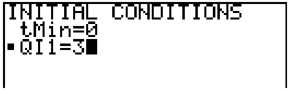
❸ Display the equation editor and store the differential equation $y'=x^2$, substituting **Q'1** for y' and **t** for x. Clear any other equations.

[F1] [F1] [x^2]



❹ Display the initial conditions editor and enter the initial conditions. A small square indicates that an initial condition is required.

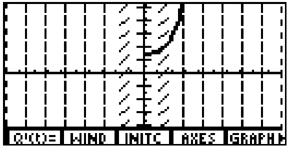
[2nd] [M3] **3**



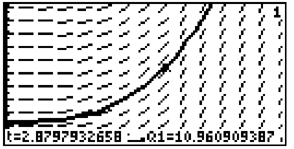
In **SlpFld** *field format,* **x=t** *is always true;* **y=Q1** *and* **fldRes=15** *are the default axes settings.*

5. Display the axes editor and enter the equation variable for which you want to solve. (Do not set **y=Q**.) — [F4] [F1] **1**
6. Accept or change **fldRes** (resolution).

7. Display the graph. With the default window variable values set, the slope fields for this graph are not very illustrative. — [2nd] [M5]



8. Change the window variables **xMin**, **xMax**, **yMin**, and **yMax**. — [F2] [▼] [▼] [▼] [▼] **0** [▼] **5** [▼] [▼] **0** [▼] **20**
9. Select **TRACE** from the GRAPH menu to re-plot the graph and activate the trace cursor. — [MORE] [F4]
   Trace the solution. The trace cursor coordinates for **t** and **Q1** are displayed. — [▶] and [◀]



Transforming an Equation into a First-Order System

On the TI-86, to enter a second-order or higher (up to ninth-order) differential equation, you must transform it to a system of first-order differential equations. For example, to enter the second-order differential equation y''= ⁻y, you must transform it to two first-order differential equations, as shown in the chart below.

Differentiate...	Define the variables as...	And then substitute:
Q'1=y'	**Q1**=y	**Q'1=Q2** (since **Q'1**=y'=**Q2**)
Q'2=y''	**Q2**=y'	**Q'2=⁻Q1**

Graphing in DirFld Format

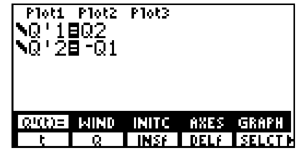
In **DifEq** *graphing mode,* **t** *is the independent variable and* **Q'**n *is the dependent variable, where* $n \geq 1$ *and* ≤ 9.

In the example, the default window variable values are set initially.

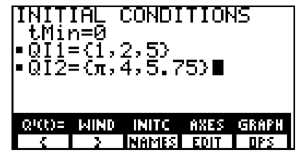
1. Display the mode screen and set **DifEq** graphing mode.
   [2nd] [MODE] [▼] [▼] [▼] [▼] [►] [►] [►] [ENTER]

2. Display the format screen and set **DirFld** graphing format.
   [GRAPH] [MORE] [F1] [▼] [▼] [▼] [▼] [▼] [►] [ENTER]

3. Display the equation editor and store the transformed system of differential equations for y''=⁻y to the equation editor, substituting **Q1** for y and **Q2** for y'.
   [F1] [F2] **2** [▼] [(-)] [F2] **1**



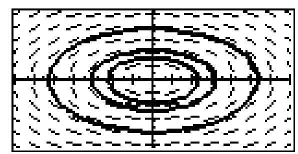
4. Display the initial conditions editor and enter the initial conditions if you want a specific solution. To enter a list of initial conditions, use **{** and **}** from the LIST menu.
   [2nd] [M3] [2nd] [LIST] [F1] **1** [,] **2** [,] **5** [F2] [▼] [F1] [2nd] [π] [,] **4** [,] **5** [.] **75** [F2]



When **DirFld** *field format is selected,* **x=Q1**, **y=Q2**, **dTime=0**, *and* **fldRes=15** *are the default axes settings. Since* **t** *is not part of the equation,* **dTime** *is ignored.*

5. Display the axes editor and enter the two equation variables for which you want to solve. You must omit the prime mark (**'**).
   [2nd] [M4] [F1] **1** [▼] [F1] **2**
6. Accept or change **fldRes** (resolution).

7. Select **ZSTD** from the GRAPH ZOOM menu to set the standard window variable values and display the graph.
   [EXIT] [MORE] [F3] [F4]
8. Clear the GRAPH menu from the screen.
   [CLEAR]



Graphing a System of Equations in FldOff Format

For this example, you must transform the fourth-order differential equation **$y^{(4)}-y=e^{-x}$** into an equivalent system of first-order differential equations, as shown in the chart below.

Differentiate...	Define the variables as...	And then substitute:
	t=x	
Q'1=y'	**Q1**=y	**Q'1**=**Q2** (since **Q'1**=y'=**Q2**)
Q'2=y''	**Q2**=y'	**Q'2**=**Q3**
Q'3=y'''	**Q3**=y''	**Q'3**=**Q4**
Q'4=$y^{(4)}$	**Q4**=y'''	**Q'4**=e^{-t}+**Q1** (since **Q'4**=$y^{(4)}=e^{-x}+y=e^{-t}$+**Q1**)

*In **DifEq** graphing mode, **t** is the independent variable and **Q'***n* is the equation variable, where $n \geq 1$ and ≤ 9.*

1. Display the mode screen and set **DifEq** graphing mode.

   [2nd] [MODE] [▼] [▼] [▼] [▼] [►] [►] [►] [ENTER]

2. Display the format screen and set **FldOff** field format.

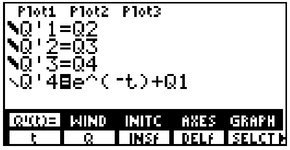
   [GRAPH] [MORE] [F1] [▼] [▼] [▼] [▼] [▼] [►] [►] [ENTER]

3. Display the equation editor and store the transformed system of differential equations for $y^{(4)}=e^{-x}+y$, substituting as shown in the chart.

   [F1] [F2] **2** [▼] [F2] **3** [▼] [F2] **4** [▼] [2nd] [e^x] [(] [(-)] [F1] [)] [+] [F2] **1**

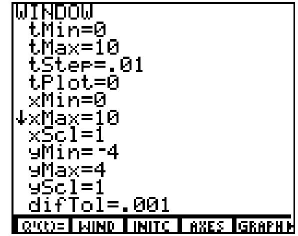
4. Deselect **Q'3**, **Q'2**, and **Q'1** to plot **Q'4=e^(-t)+Q1** only.

   [▲] [F5] [▲] [F5] [▲] [F5]



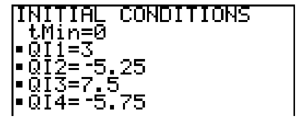
❺ Display the window editor and set the window variable values.

[2nd] [M2] [▼] **10** [▼] [.] **01** [▼] [▼] **0** [▼] [▼] [▼] [(–)] **4** [▼] **4**



❻ Display the initial conditions editor and enter the initial conditions. A small square indicates that an initial condition is required.

[F3] **3** [▼] [(–)] **5** [.] **25** [▼] **7** [.] **5** [▼] [(–)] **5** [.] **75**



When **FldOff** *field format is selected,* **x=t** *and* **y=Q** *are the default axes settings.*

❼ Display the axes editor. Enter the equation variables for which you want to solve.

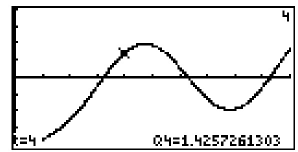
[F4]

❽ Display the graph. Explore the equation with the trace cursor.

[EXIT] [MORE] [F4] [▶] and [◀]

❾ Enter a **t** value to move the trace cursor to the solution for that **t** value. The **t** and **Q4** coordinates are displayed.

4 [ENTER]

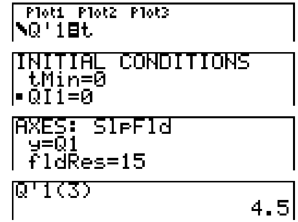


To paste **'** *to the home screen, you can select it from the* CHAR MISC *menu or from the* CATALOG.

Due to TI-86 system requirements, you must express **Q1(3)** *as* **Q'1(3)** *on the calculator.*

Solving a Differential Equation for a Specified Value

On the home screen in **DifEq** graphing mode, you can solve a differential equation stored to a specified independent variable value or expression. The syntax is: **Q'***n***(***value***)**.



- The equation must be stored to a **DifEq** equation variable (**Q'1** through **Q'9**).
- The initial conditions must be defined.
- The result sometimes varies, depending on the axes settings.

Using Graph Tools in DifEq Graphing Mode

The Free-Moving Cursor

The free-moving cursor works in **DifEq** mode as it does in **Func** graphing. The cursor coordinate values for **x** and **y** are displayed, and the variables are updated.

Tracing a Differential Equation

To begin a trace, select **TRACE** from the GRAPH menu ([GRAPH] [MORE] [F4]). The trace cursor appears on the first equation at or near **tPlot** (or **tMin**, if **t** is an axis).

The trace coordinates displayed at the bottom of the screen reflect the axes settings. For example, if **x=t** and **y=Q1**, then **t** and **Q1** are displayed. If **t** is not an axis, three trace values are displayed. If **t** is an axis, only **t** and the variable designated as the y-axis are displayed.

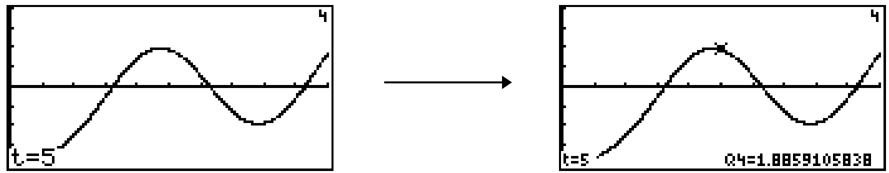
QuickZoom is available in **DifEq** *graphing; panning is not (Chapter 6).*

The trace cursor moves in increments or decrements of **tStep**. As you trace an equation, the coordinates are updated and displayed. If the cursor moves off the screen, the coordinate values displayed at the bottom of the screen continue to change appropriately.

Moving the Trace Cursor to a t Value

To move the trace cursor to any valid **t** value on the current equation, enter the number. When you enter the first digit, a **t=** prompt is displayed in the bottom-left corner. The value you enter must be valid for the current graph screen. When you have completed the entry, press [ENTER] to reactivate the trace cursor.

Values for **t** *and* **Q** *are displayed on the graph to the right because* **x=t** *and* **y=Q** *graph axes are selected.*



Drawing on a Differential Equation Graph

The GRAPH DRAW menu items work the same in **DifEq** graphing mode as in **Func** graphing. DRAW instruction coordinates are the x- and y-coordinates of the graph screen.

DrEqu is available only in **DifEq** mode. **DrInv** is not available in **DifEq** graphing mode.

Drawing an Equation and Storing Solutions to Lists

To draw a solution on the current graph screen and store the results to specified list names, the syntax is:

DrEqu(*xAxisVariable*,*yAxisVariable*[,*xList*,*yList*,*tList*]**)**

xAxisVariable and *yAxisVariable* specify the axes on which the drawing is based; they may differ from the current graph screen's axes settings.

xList, *yList*, and *tList* are optional list names to which you can store the solutions **x**, **y**, and **t**. You then can display the lists on the home screen or in the list editor (Chapter 11).

__DrEqu(__ does not store values to __x__, __y__, or __t__.

Use the free-moving cursor to select initial conditions.

You cannot trace the drawing. However, you can plot *xList*, *yList*, or *tList* as a stat plot after you draw the equation, and then trace them (Chapter 14). Also, you can fit statistical regression models to the lists (Chapter 14).

In the example, the default window variable values are set. If necessary, select **ZSTD** *from the* GRAPH ZOOM *menu.*

If you select **FldOff** *field format, you must enter initial conditions before you use* **DrEqu(**.

❶ Display the mode screen and set **DifEq** graphing mode.

[2nd] [MODE] ▼ ▼ ▼ ▼ ▶ ▶ ▶ [ENTER]

❷ Display the format screen and set **DirFld** field format.

[GRAPH] [MORE] [F1] ▼ ▼ ▼ ▼ ▼ ▶ [ENTER]

❸ Display the equation editor and store the equations **Q'1=Q2** and **Q'2=⁻Q1**. (Delete all other equations.)

[F1] [F2] **2** ▼ [(-)] [F2] **1**

❹ Remove the format screen, and then select **DrEqu** from the GRAPH DRAW menu. **DrEqu(** is pasted to the home screen.

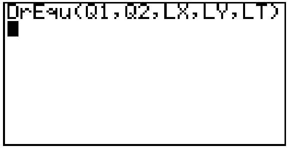
[EXIT] [EXIT] [GRAPH] [MORE] [F2] [F1]

❺ Assign variables to the x- and y-axes.

[ALPHA] [Q] **1** [,] [ALPHA] [Q] **2** [,]

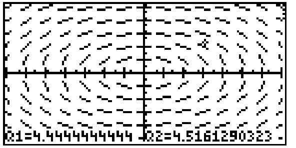
❻ Specify list names to which to store the solution lists for **x**, **y**, and **t**.

[ALPHA] [L] [ALPHA] [X] [,] [ALPHA] [L] [ALPHA] [Y] [,] [ALPHA] [L] [ALPHA] [T] [)]



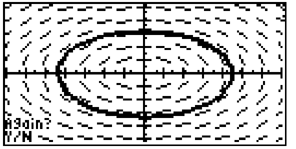
In the example, since no initial conditions were set, the equation in **Q'1** *is not plotted.*

❼ Display the graph screen and plot the direction field. [ENTER]



❽ Move the free-moving cursor to the initial condition coordinates you want. [▸] [▾] [◂] [▴]

❾ Draw the solution. The solution lists for **x**, **y**, and **t** are stored to **LX**, **LY**, and **LT**. The **Again?** prompt is displayed and ALPHA-lock is on for [Y] and [N] only. [ENTER]



- ♦ To use **DrEqu(** again with new initial conditions, press [Y], [▸], [▾], [◂], or [▴].
- ♦ To leave **DrEqu(** and display the GRAPH menu, press [N] or [EXIT].

Using ZOOM Operations

The GRAPH ZOOM menu items, except **ZFIT**, work the same in **DifEq** graphing mode as in **Func** graphing mode. In **DifEq** graphing mode, **ZFIT** adjusts the graph screen in both the x direction and y direction.

The ZOOM menu items affect only the **x** (**xMin**, **xMax**, and **xScl**) and **y** (**yMin**, **yMax**, and **yScl**) window variables. The **t** window variables (**tMin**, **tMax**, **tStep**, and **tPlot**) are not affected, except with **ZSTD** and **ZRCL**. You may want to edit the **t** window variables to ensure that sufficient points are plotted. **ZSTD** sets **difTol=.001** and **t** and **Q** as the axes.

Drawing Solutions Interactively with EXPLR

❶ Display the mode screen and set **DifEq** graphing mode.

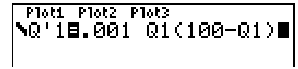
[2nd] [MODE] [▼] [▼] [▼] [▼] [▶] [▶] [▶] [ENTER]

❷ Display the format screen and set **FldOff** field format.

[GRAPH] [MORE] [F1] [▼] [▼] [▼] [▼] [▼] [▶] [▶] [ENTER]

❸ Display the equation editor and store the equation **Q'1=.001Q1(100−Q1)**. (Delete all other equations.)

[F1] [.] **001** [F2] **1** [(] **100** [−] [F2] **1** [)]

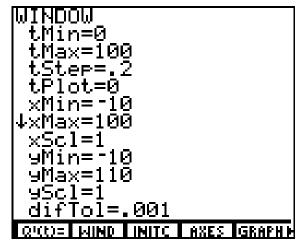


❹ Set the axes to **x=t** and **y=Q1**.

[2nd] [M4] [▼] [▶] **1**

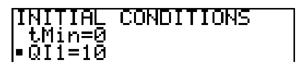
❺ Display the window editor and set the window variable values.

[2nd] [M2] [▼] **100** [▼] [.] **2** [▼] [▼] [▼] **100** [▼] [▼] [▼] **110**

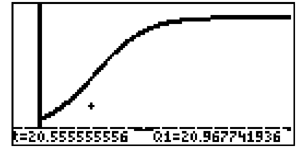


❻ Display the initial conditions editor and enter the initial condition.

[F3] **10**

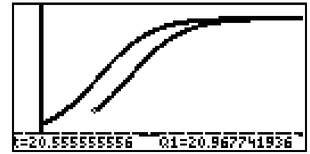


❼ Select **EXPLR** from the GRAPH menu. [MORE] [F5]



❽ Move the free-moving cursor to the initial condition for which you want to solve. ▶ ▼ ◀ ▲

❾ Draw the solution to **Q1**, using the cursor coordinates (x,y) as initial condition (**t**,**Q'1(t)**). [ENTER]



To continue drawing more solutions, move the free-moving cursor and then press [ENTER].

To stop using **EXPLR**, press [EXIT].

If **SlpFld** or **DirFld** is set, the axes are set to specific solutions automatically.

♦ For **SlpFld**, **x=t** and **y=Q1** are set.
♦ For **DirFld**, **x=Q1** and **y=Q2** are set.

If the axes are set to a specific solution **t**, **Q**n, or **Q'**n, that solution is drawn.

If the axes are not set to a specific solution and **t** is one variable and **Q** is the other, **Q1** is drawn.

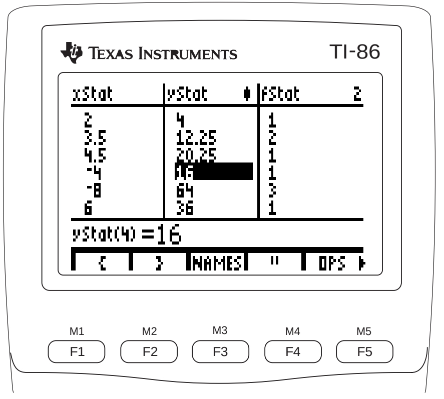
If both axes are set to a **Q** variable, executing **EXPLR** results in an error.

Evaluating Differential Equations for a Specified t

When the trace cursor is not active, the GRAPH menu item **EVAL** evaluates currently selected differential equations **Q**n for a specified value of **t**, **tMin≤t≤tMax**. You can use it directly on the graph. In a program or from the home screen, **eval** returns a list of **Q** values.

When **DirFld** or **SlpFld** field format is set, you must specify initial conditions before using **EVAL**.

11 Lists



Texas Instruments
TI-86
xStat
yStat
fStat
yStat(4) =16
NAMES
OPS
M1
M2
M3
M4
M5
F1
F2
F3
F4
F5

Lists on the TI-86

The length and number of lists you can store in the TI-86 is limited only by memory capacity.

A list is a set of real or complex elements, as in **{5,⁻20,13,9}**. On the TI-86, you can:

- ♦ Enter a list directly in an expression (page 153)
- ♦ Enter a list and store it to a list name (variable) (page 154)
- ♦ Enter a list name in the list editor (page 156), and then enter elements directly or use an attached formula to generate them automatically (page 161)
- ♦ Collect data with the Calculator-Based Laboratory™ (CBL 2™/CBL™) or Calculator-Based Ranger™ (CBR) and store it to a list name on the TI-86 (Chapter 18)
- ♦ Create lists dynamically using the LIST OPS menu item **seq** (page 159)

If you enter more than one list in an equation or expression, all lists must have the same number of elements.

On the TI-86, you can use a list:

- ♦ As a set of values for an argument in a function to return a list of answers (Chapter 1)
- ♦ As part of an equation to graph a family of curves (Chapter 5)
- ♦ As a set of statistical data to analyze with statistical functions and plot on the graph screen (Chapter 14)

The LIST Menu [2nd] [LIST]

{	}	NAMES	EDIT	OPS
open brace	close brace	list names menu	list editor	list operations menu

When you enter a list, **{** (open brace) specifies the beginning and **}** (close brace) specifies the end. To paste **{** or **}** to the cursor location, select either from the LIST menu.

The LIST NAMES Menu [2nd] [LIST] [F3]

The LIST NAMES *menu shown here has no user-created list names.*

{	}	NAMES	EDIT	OPS
fStat	xStat	yStat		

Chapter 14 describes **fStat**, **xStat**, *and* **yStat**.

Each user-created list name is added to the LIST NAMES menu and VARS LIST screen. List names, including **fStat**, **xStat**, and **yStat**, are sorted in alphanumeric order in both places.

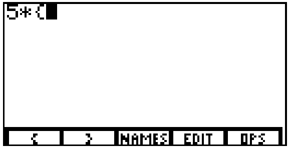
Creating, Storing, and Displaying Lists

Entering a List Directly in an Expression

To enter a list directly, the syntax is: **{***element1*,*element2*,...,*element n***}**

❶ Enter any part of the expression that precedes the list. **5** [×]

❷ Select **{** from the LIST menu to begin the list. [2nd] [LIST] [F1]



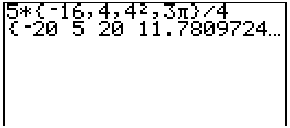
❸ Enter each list element, separating each from the other with a comma. Each list element can be an expression. [(-)] **16** [,] **4** [,] **4** [x^2] [,] **3** [2nd] [π]

❹ Select **}** from the LIST menu to end the list. [F2]

An ellipsis (...) indicates that a list continues beyond the screen. Use [▸] *and* [◂] *to scroll the list.*

❺ Enter any part of the expression that follows the list. [÷] **4**

❻ Evaluate the expression. Any elements that are expressions are evaluated first. [ENTER]

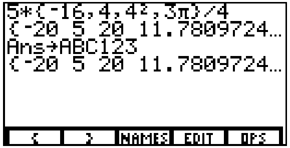


Creating a List Name by Storing a List

To store a list, the syntax is: **{***element1*,*element2*, … ,*element n***}**→*listName*

*You need not enter the close brace (**}**) when you use* STO▸ *to store a list name.*

❶	Enter a list directly. (To store a result expressed as a list and currently stored in **Ans**, as shown in the example, begin these steps at step 2.)	(steps 2 through 5 above)
❷	Paste → to the cursor location. ALPHA-lock is on.	STO▸
❸	Enter the list name. Either select a name from the LIST NAMES menu or directly enter a name one to eight characters long, starting with a letter.	[A] [B] [C] ALPHA **1 2 3**
❹	Store the list to the list name.	ENTER

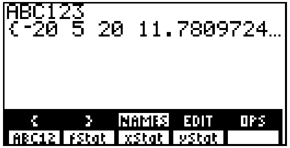


To delete a list name from memory, use the MEM DELETE:LIST *screen (Chapter 17).*

Displaying List Elements Stored to a List Name

The TI-86 distinguishes between uppercase and lowercase letters in list names. For example, **ABC123**, **Abc123**, *and* **abc123** *are three different list names.*

❶	Enter the list name on the home screen; either select it from the LIST NAMES menu or enter the characters.	2nd [LIST] F3 F1
❷	Display the list elements.	ENTER



Displaying or Using a Single List Element

To display or use a single list element, the syntax is: *listName***(***element#***)**

*listName***(***element#***)** *is valid as part of an expression.*

element# is ≥ 1 and ≤ the dimension of the list.

1. Enter the list name; either select it from the LIST NAMES menu or enter the characters. — [2nd] [LIST] [F3] [F1]
2. In parentheses, enter the element's place number in the list. — [(] **4** [)]
3. Display the list element. — [ENTER]

Storing a New Value to a List Element

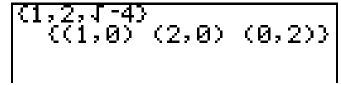
To store a value to a current element or one element beyond the end of a list, the syntax is:
value→*listName***(***element#***)**

value can be an expression.

1. Enter the value to be stored in a current list element or one element beyond the end. — [2nd] [√] **18**
2. Paste → to the cursor location. — [STO▶]
3. Enter the list name; either select it from the LIST NAMES menu or enter the characters. — [F1]
4. Enter the element's place number in parentheses. (In the example, **5** is one beyond the current dimension of **ABC123**). — [ALPHA] [(] **5** [)]
5. Enter the new value to the element number. (√**18** is evaluated and added as the fifth element.) — [ENTER]

Complex List Elements

A complex number can be a list element. If at least one list element is a complex number, all elements in the list are displayed as complex. (√-4 results in a complex number.)

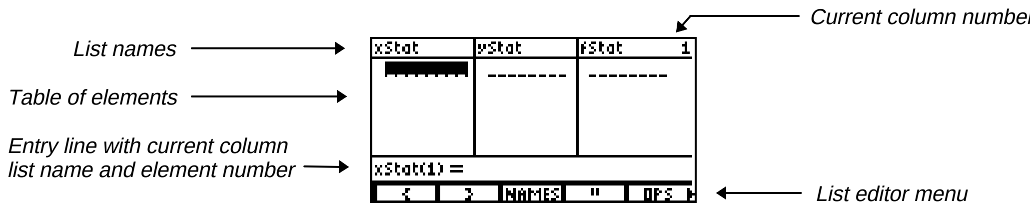


The List Editor [2nd] [LIST] [F4]

The list editor is a table where you can store, edit, and view up to 20 lists that are in memory. Also, you can create list names and attach formulas to lists in the list editor.

You also can press [2nd] [STAT] [F2] to display the list editor.

The list editor abbreviates list names and element values when necessary. The entry line displays entire list names and element values.



The List Editor Menu [2nd] [LIST] [F4]

{	}	NAMES	"	OPS	►	►REAL				

*The list editor menu items **{**, **}**, **NAMES**, and **OPS** are identical to the LIST menu items (page 152).*

"	Designates the beginning and end of a formula to be attached to a list name
►REAL	Converts the current list to a list of real numbers

To use LIST OPS menu items (or any other functions or instructions) in the list editor, the cursor location must be appropriate for the result. For example, you can use the LIST OPS menu item **sortA** when a list name is highlighted but not when an element is highlighted.

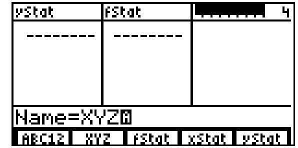
Creating a List Name in the Unnamed Column

After memory is reset, **xStat**, **yStat**, *and* **fStat** *are stored to columns* **1**, **2**, *and* **3**. *Resetting defaults does not affect the list editor.*

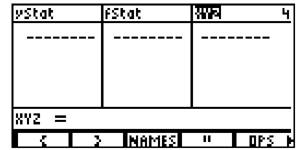
To move from the list name in column 1 to the unnamed column, press [◀] [▶].

❶ Display the list editor. [2nd] [LIST] [F4]

❷ Move the cursor to the unnamed column (column 4). The **Name=** prompt is displayed in the entry line. ALPHA-lock is on. [▲] [▶] [▶] [▶]



❸ Enter the list name. The list name is displayed at the top of the current column. In the entry line, a list name prompt is displayed. The name becomes a LIST NAMES menu item and a VARS LIST screen item. [X] [Y] [Z] [ENTER]



Inserting a List Name into the List Editor

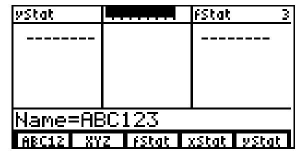
If all 20 columns have list names, you must remove a list name to make room for the unnamed column.

To cancel the list name insertion, press [CLEAR].

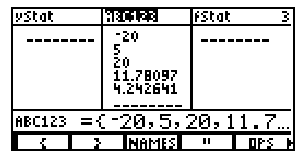
If a formula were attached to **ABC123**, *the formula would be displayed in the entry line instead of the list shown in step 3 (page 162.)*

❶ Move the cursor to column **3**. [◀]

❷ Insert a new, unnamed column. List names shift right, clearing column **3**. The **Name=** prompt and LIST NAMES menu are displayed. [2nd] [INS]



❸ Select **ABC12** from the LIST NAMES menu to insert the list name **ABC123** into column **3**. Elements stored to **ABC123** fill the column **3** table of elements. The full value of all **ABC123** elements is displayed in the entry line. [F1] [ENTER]

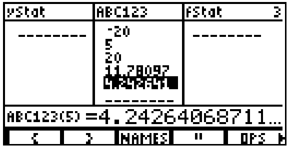


Displaying and Editing a List Element

To cancel any editing and restore the original element at the cursor, press [CLEAR] [ENTER].

❶ Move the cursor onto the fifth element of **ABC123**. In the entry line, the list name, the element number in parentheses, and the element's full value are displayed.

[▼] [▼] [▼] [▼] [▼]



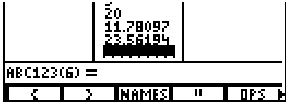
You can enter an expression as an element.

❷ Switch to edit-element context and edit the element in the entry line.

5 [×] [(] **6** [2nd] [π] [)] [÷] **4**

❸ Enter the edited element. Any expression is evaluated and the value is stored to the current element.

[ENTER] (or [▼] or [▲])



Deleting Elements from a List

To delete a single element from a list, move the cursor onto the element and press [DEL]. The element is deleted. You can clear all elements from a list in any of three ways.

- In the list editor, press [▲] to move the cursor onto a list name and press [CLEAR] [ENTER].
- In the list editor, move the cursor onto each element, and then press [DEL] one by one.
- On the home screen or in the program editor, enter **0→dimL** *listName* to set the dimension of *listName* to **0** (A to Z Reference).

Removing a List from the List Editor

To remove a list from the list editor, move the cursor onto the list name and then press [DEL]. The list is not deleted from memory; it is only removed from the list editor.

You can remove all user-created lists from the list editor and restore list names **xStat**, **yStat**, and **fStat** to columns **1**, **2**, and **3** in either of two ways.

- Use **SetLEdit** with no arguments (page 161).
- Reset all memory (Chapter 17). Resetting defaults does not affect the list editor.

Using List Operations

The LIST OPS (Operations) Menu [2nd] [LIST] [F5]

{	}	NAMES	EDIT	OPS
dimL	sortA	sortD	min	max

▸

sum	prod	seq	li▸vc	vc▸li
Fill	aug	cSum	Deltal	Sortx
Sorty	Select	SetLE	Form	

For all LIST OPS *menu items except* **Fill(** *and sometimes* **dimL**, *a directly entered list* (**{***element1*,*element2*,...**}**) *is valid for the list argument.*

dimL *list*	Returns the dimension of (or number of elements in) *list*
#ofElements →**dimL** *listName*	Creates *listName* as a list that is *#ofElements* in length; each element is a **0**
#ofElements→**dimL** *listName*	Redimensions an existing *listName*; previously entered elements within the new dimension remain; each new list element is a **0**; each element in the old list that is outside the new dimension is deleted

SortA *and* **SortD** *sort complex lists based on magnitude (modulus).*

sortA *list*	Sorts *list* elements in ascending order, from low to high values
sortD *list*	Sorts *list* elements in descending order, from high to low values

For a complex list, **min** *or* **max** *returns the smallest or largest magnitude (modulus).*	**min(***list***)**	Returns the smallest element of a real or complex *list*
	max(*list***)**	Returns the largest element of a real or complex *list*
	sum *list*	Returns the sum of all the elements of a real or complex *list*, adding from the last element to the first
	prod *list*	Returns the product of all the elements of a real or complex *list*
	seq(*expression***,***variable***,** *begin***,***end*[**,***step*]**)**	Returns a list in which each element is the result of the evaluation of *expression* with regard to *variable* for the values ranging from *begin* to *end* in intervals of *step* (*step* can be negative)
	li▸vc *list* **li▸vc {***element1***,***element2***,...}**	Converts a real or complex *list* to a vector
	vc▸li *vector* **vc▸li** [*element1***,***element2***,...**]	Converts a real or complex *vector* to a list
	Fill(*number***,***listName***)**	Stores a real or complex *number* to every element of *listName*
	aug(*listA***,***listB***)**	(augment) Concatenates the real or complex elements of *listA* and *listB*
	cSum(*list***)**	Returns a list of the cumulative sums of real or complex *list* elements, starting with the first element and proceeding to the last
Selecting **Deltal** *from the menu pastes* **Deltalst(** *to the cursor location.*	**Deltalst(***list***)**	Returns a list containing the differences between consecutive elements for all elements in a real or complex *list*
	Sortx [*ListName***,***ListName***,** *frequencyListName*]	In ascending order of **x** elements, sorts *xListName* , sorts **x** and **y** data pairs, and optionally, their frequencies, in *xListName*, *yListName* , and *frequencyListName*; **xStat** and **yStat** are defaults
For **Sortx** *and* **Sorty***, both lists must have the same number of elements.*	**Sorty** [*xListName***,***ListName***,** *frequencyListName*]	In ascending order of **y** elements, sorts *xListName* , sorts **x** and **y** data pairs, and optionally, their frequencies, in *xListName*, *yListName* , and *frequencyListName*; **xStat** and **yStat** are defaults

Select(*xListName***,** *yListName***)**	Selects one or more specific data points from a scatter plot or xyLine plot (only), then stores the selected data points to *xListName* and *yListName* (Chapter 14)
SetLEdit [*column1ListName***,** *column2ListName***,...,** *column20ListName*]	Sets up the list editor; **SetLEdit** with one to 20 *ListNames* loads them in the specified order; **SetLEdit** with no arguments removes all current list names from the list editor and enters the default lists **xStat**, **yStat**, and **fStat** to columns 1, 2, and 3
Form("*formula***",***listName***)**	Attaches *formula* to *listName*; *formula* resolves to a list, which is dynamically stored and updated in *listName* (page 162)

Selecting **SetLE** *from the menu pastes* **SetLEdit** *to the cursor location.*

You can create new list names as **SetLEdit** *arguments.*

Using Mathematical Functions with Lists

You can use a list as a single argument for many TI-86 functions; the result is a list. The function must be valid for every element in the list; however, when graphing, undefined points do not result in an error.

When you use lists for two or more arguments in the same function, all lists must have the same number of elements (equal dimension). Here are some examples of a list as a single argument.

{1,2,3}+10 returns **{11 12 13}**

{5,10,15}*{2,4,6} returns **{10 40 90}**

3+{1,7,(2,1)} returns **{(4,0) (10,0) (5,1)}**

√**{4,16,36,64}** returns **{2 4 6 8}**

sin {7,5} returns **{.656986598719 ⁻.958924274663}**

{1,15,36}<19 returns **{1 1 0}**

Attaching a Formula to a List Name

You cannot edit an element of a list created from an attached formula unless you first detach the formula from the list name.

When you include more than one list name in an attached formula, each list must have the same dimension.

You can attach a formula to a list name so that the formula generates a list that is stored and dynamically updated in the list name.

- When you edit an element of a list that is referenced in the formula, the corresponding element in the list to which the formula is attached is updated.
- When you edit the formula itself, all elements in the list to which the formula is attached are updated.

To attach a formula to a list name on the home screen or in the program editor, the syntax is:

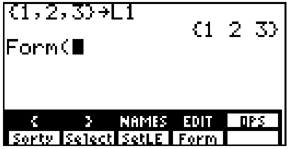
Form("*formula***",***listName***)**

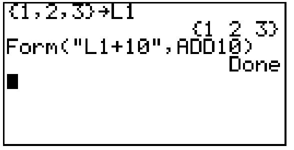
When you enter a new list name as the second argument for **Form(** , the list name is created and stored in the LIST NAMES menu and VARS LIST screen upon execution.

Begin these steps on a blank line on the home screen.

To view a formula attached to a list name, use the list editor (page 157).

❶	Store elements to a list name.	[2nd] [LIST] [F1] **1** [,] **2** [,] **3** [F2] [STO▶] [L] [ALPHA] **1** [ENTER]
❷	Select **Form** from the LIST OPS menu; **Form(** is pasted to the cursor location.	[F5] [MORE] [MORE] [MORE] [F4]
❸	Enter a formula in quotation marks.	[2nd] [STRNG] [F1] [ALPHA] [L] **1** [+] **10** [F1]
❹	Enter a comma and then the list name to which you want to attach the formula.	[,] [ALPHA] [ALPHA] [A] [D] [D] [ALPHA] **10** [)]
❺	Attach the formula to the list name.	[ENTER]





Comparing an Attached List with a Regular List

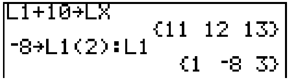
To see the differences between an attached list and a regular list, follow these steps. The example below builds on the example above for attaching a formula to a list. Notice that the formula in step 1 below is not attached to **LX** because it is not set off by quotation marks.

1. Generate a regular list by storing the expression **L1+10** to the list name **LX**.

   [ALPHA] [L] **1** [+] **10** [STO▶] [L] [X] [ENTER]

2. Change the second element in **L1** to **⁻8** and display the edited list.

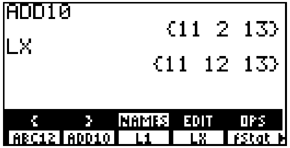
   [(-)] **8** [STO▶] [L] [ALPHA] **1** [(] **2** [)] [2nd] [:] [ALPHA] [L] **1** [ENTER]



If other list names are stored on the LIST NAMES *menu, pressing* [F1] *and* [F3] *may not paste* **ADD10** *and* **LX** *to the home screen as shown.*

3. Compare the elements of the regular list **LX** with **ADD10**, to which the formula **L1+10** is attached. Notice that element 2 of **LX** is unchanged. Meanwhile, element 2 of **ADD10** has been recalculated, since element 2 of **L1** has been edited.

   [2nd] [LIST] [F3] [F2] [ENTER] [F4] [ENTER]



Using the List Editor to Attach a Formula

In the example, only fStat, **xStat**, *and* **yStat** *are on the* LIST NAMES *menu and* **xStat={⁻2,9,6,1,⁻7}**.

The attached formula must be set off by quotation marks.

1. Display the list editor.

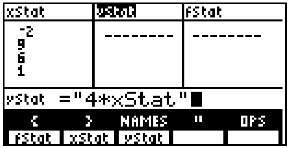
   [2nd] [LIST] [F4]

2. Highlight the list name to which you want to attach the formula.

   [▲] [▶]

3. Enter the formula in quotation marks.

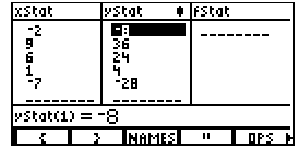
   [F4] **4** [×] [F3] [F2] [2nd] [F4]



The list editor displays a formula-lock symbol next to each list name that has a formula attached to it.

❹ Attach the formula and generate the list. ENTER

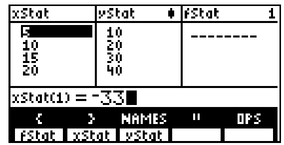
- ♦ The TI-86 calculates each list element.
- ♦ A lock symbol is displayed next to the list name to which the formula is attached.

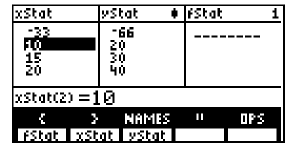


To edit an attached formula, press ENTER in step 3, and then edit the formula.

Using the List Editor With Attached-Formula Lists

When you edit an element of a list referenced in an attached formula, the TI-86 updates the corresponding element in the list to which the formula is attached.





When you edit or enter elements of a displayed list in any of the three current list editor columns while an attached-formula list also is displayed, the TI-86 takes slightly longer to execute the edit or entry. To reduce this effect, move lists with formulas off the current three-column display, either by scrolling columns to the left or right or by rearranging the list editor.

Executing and Displaying Attached Formulas

An attached formula must resolve to a list upon execution. Some examples of formulas that resolve to a list are **"5*xStat"**, **"seq(x,x,1,10)"**, and **"{3,5, ⁻8,4}^{2}/10"**. Execution of the formula occurs when you attempt to display the list to which the formula is attached. Also, the formula is executed whenever a list referenced by the formula is modified — whether on the home screen, in the list editor, or in a program.

You can successfully attach to a list a formula that does not yet resolve to a list. For example, you can attach **"5*xStat"** to the list name **BY5** with no elements stored to **xStat**. However, if you attempt to display **BY5** when **xStat** has no elements, an error occurs.

When you attach such a formula to a list name in the list editor, the formula is successfully attached, but an error occurs. This is because the list editor attempts to execute the formula immediately after attaching it to the list name.

To view the list editor again, you must return to the home screen and either enter something to cause the formula to resolve to a list or remove the attached-formula list from the list editor using the LIST OPS menu item **SetLE** (page 161).

Handling Errors Related to Attached Formulas

All elements of a list referenced by an attached formula must be valid for the attached formula.

On the home screen, you can attach to a list a formula that references another list that has no elements (dimension is **0**; page 161). However, you cannot display the attached-formula list in the list editor or on the home screen until you enter at least one element to the list that the formula references.

Tip: If an error menu is returned when you attempt to display an attached-formula list in the list editor, you can select **GOTO**, write down the formula that is attached to the list name, and then press [CLEAR] [ENTER] to detach (clear) the formula. Then you can use the list editor to find the source of the error. After making the appropriate changes, you can re-attach the formula to the list name.

If you do not want to clear the formula, you can select **QUIT**, display the referenced list on the home screen, and find and edit the source of the error. To edit an element of a list on the home screen, store the new value to *listName***(***element#***)** (page 155).

Detaching a Formula from a List Name

You can detach a formula in any of five ways.

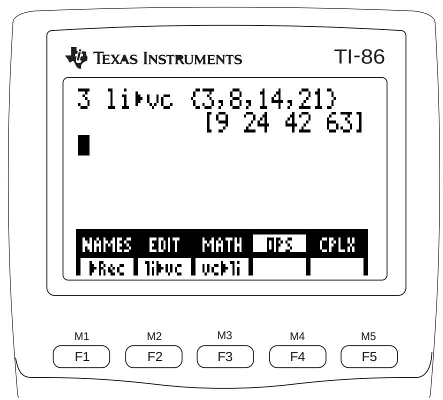
- Use **dimL** to change the dimension of the list (page 159).
- Use *value*→*listName*(*element#*) to store *value* to an attached-formula list element.
- Use ""→*listName*, where *listName* is the attached-formula list.
- In the list editor, move the cursor onto the name of the attached-formula list, and then press [ENTER] [CLEAR] [ENTER]. All list elements remain, but the formula is detached and the lock symbol disappears.
- In the list editor, move the cursor onto an element of the attached-formula list. Press [ENTER], edit the element, and then press [ENTER]. The element changes, the formula is detached, and the lock symbol disappears. All other list elements remain.

Editing an Element of a Attached-Formula List

As described above, one way to detach a formula from a list name is to edit an element of the attached-formula list. The TI-86 protects against inadvertently detaching the formula from the list name when you move the cursor onto one of the elements.

Because of the protection feature, you must press [ENTER] before you can edit an element of an attached-formula list. The protection feature prevents you from deleting an element of an attached-formula list. To delete an element of a attached-formula list, you must first detach the formula in any of the ways described above.

12 Vectors



TEXAS INSTRUMENTS
TI-86
3 li▸vc {3,8,14,21}
[9 24 42 63]
NAMES EDIT MATH OPS CPLX
▸Rec li▸vc vc▸li
M1 M2 M3 M4 M5
F1 F2 F3 F4 F5

Vectors on the TI-86

A vector is a one-dimensional array, arranged in either one row or one column. The vector elements can be real or complex. You can create, display, and edit vectors on the home screen or in the vector editor. When you create a vector, the elements are stored to the vector name.

The TI-86 vector editor displays a vector vertically. On the home screen, a vector is entered and displayed horizontally. When you use a vector in an expression, the TI-86 automatically interprets the vector in the form (row vector or column vector) that is appropriate for the expression. For example, a column vector is appropriate for the expression *matrix***vector*.

On the TI-86, you can store up to 255 elements to a vector in rectangular form. You can use two- or three-element vectors to define magnitude and direction in a two- or three-dimensional space. You can express two- or three-element vectors in different forms, depending on the type of vector.

To express a...	You enter:	And the TI-86 returns:
Two-element rectangular vector	$[x,y]$	$[x\ y]$
Two-element cylindrical vector	$[r\angle\theta]$	$[r\angle\theta]$
Two-element spherical vector	$[r\angle\theta]$	$[r\angle\theta]$
Three-element rectangular vector	$[x,y,z]$	$[x\ y\ z]$
Three-element cylindrical vector	$[r\angle\theta,z]$	$[r\angle\theta\ \ z]$
Three-element spherical vector	$[r\angle\theta\angle\phi]$	$[r\angle\theta\angle\phi]$

Creating, Storing, and Displaying Vectors

The VECTR (Vector) Menu [2nd] [VECTR]

NAMES	EDIT	MATH	OPS	CPLX
vector names menu	vector editor	vector math menu	vector operations menu	complex vector menu

The VECTR NAMES Menu [2nd] [VECTR] [F1]

The VECTR NAMES menu contains all currently stored vector names in alphanumeric order. To paste a vector name to the current cursor location, select it from the menu.

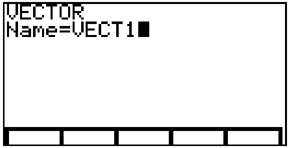
Creating a Vector in the Vector Editor [2nd] [VECTR] [F2]

The TI-86 distinguishes between uppercase and lowercase letters in vector names. For example, **VECT1**, **Vect1**, *and* **vect1** *are three different vector names.*

↓ or ↑ in the first column indicates additional vector elements.

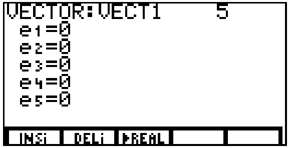
❶ Display the vector **Name=** prompt screen. — [2nd] [VECTR] [F2]

❷ ALPHA-lock is on. The VECTR NAMES menu is displayed. Enter a name from one to eight characters long, starting with a letter. — [V] [E] [C] [T] [ALPHA] **1**



❸ Display the vector editor. The vector editor menu also is displayed. — [ENTER]

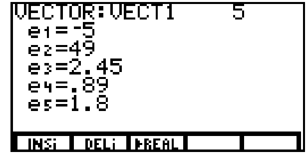
❹ Accept or change the vector *elements* dimension with an integer ≥ 1 and ≤ 255. The vector is displayed; all elements are **0**. — **5** [ENTER]



You can enter an expression at a vector element prompt.

❺ Enter each vector element value at each vector element prompt. You can enter expressions. To move to the next prompt, press [ENTER] or [▾]. The vector elements are stored to **VECT1**, which becomes a VECTR NAMES menu item.

[(–)] **5** [▾] **49** [▾] **2** [.] **45** [▾] [.] **89** [▾] **1** [.] **8**



The Vector Editor Menu

[2nd] [VECTR] [F2] ***vectorName*** [ENTER]

INSi	DELi	▸REAL		

INSi Inserts a blank element (**e***n***=**) prompt at the cursor location; shifts current elements down

DELi Deletes the element from the cursor location and from the vector; shifts elements up

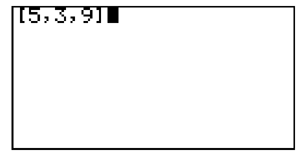
▸REAL Converts the displayed complex number vector to a real number vector

Creating a Vector on the Home Screen

❶ Define the beginning of the vector with [. [2nd] [[ ]

❷ Enter each vector element, separating each from the next with a comma. **5** [,] **3** [,] **9**

❸ Define the end of the vector with]. [2nd] [ ]]



To delete a vector name from memory, use the MEM DELETE:VECTR *screen (Chapter 17).*

❹ Store the vector to a vector name from one to eight characters long, starting with a letter. The vector is displayed horizontally and the vector name becomes a VECTR NAMES menu item. [STO▸] [2nd] [alpha] [V] [E] [C] [T] [ALPHA] [ALPHA] **1** [ENTER]

Creating a Complex Vector

If any element of a vector is complex, all elements of the vector are displayed as complex. For example, when you enter the vector [**1,2,(3,1)**] , the TI-86 displays [**(1,0) (2,0) (3,1)**].

To create a complex vector from two real vectors, the syntax is:

*realVector***+(0,1)***imaginaryVector*→*complexVectorName*

realVector contains the real part of each element and *imaginaryVector* contains the imaginary part.

Displaying a Vector

To display a vector, paste the vector name to the home screen, and then press [ENTER].

To display a specific element of *vectorName* on the home screen or in a program, the syntax is:

*vectorName***(***element#***)**

Real two- and three-element vector results are displayed according to the current vector mode setting: **RectV**, **CylV**, or **SphereV** (Chapter 1). You can select a vector conversion instruction from the VECTR OPS menu to override the mode setting (page 173).

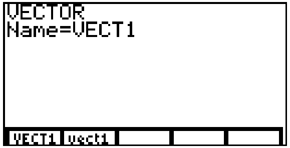
Complex vectors are displayed in rectangular form only.

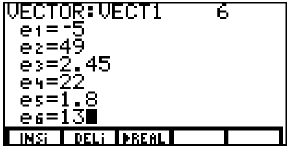
Using a Vector in an Expression

When you execute the expression, the answer is displayed as a vector.

- You can enter the vector directly (for example, **35**−[**5**,**10**,**15**]).
- You can use [ALPHA] and [2nd] [alpha] to enter a vector name's individual characters.
- You can select the vector name from the VECTR NAMES menu ([2nd] [VECTR] [F1]).
- You can select the vector name from the VARS VECTR screen ([2nd] [CATLG-VARS] [MORE] [F1]).

Editing Vector Dimension and Elements





1. Display the vector **Name=** prompt screen. — [2nd] [VECTR] [F2]
2. Enter the vector name. Either select it from the VECTR NAMES menu or enter the characters. — [F1]
3. Display the vector editor. — [ENTER]
4. Change or accept the vector dimension. — **6** [ENTER]
5. Move the cursor to any element and edit it. Continue moving the cursor to other elements. — [▼] [▼] [▼] **22** [▼] [▼] **13**
6. Save the changes and exit the vector editor. — [EXIT]

You can use [CLEAR], [DEL], and [2nd] [INS] to edit matrix elements. You also can overwrite existing characters.

To use [STO►] to change an element value on the home screen, the syntax is:

value→*vectorName*(*element#*)

The VECTR MATH Menu [2nd] [VECTR] [F3]

NAMES	EDIT	MATH	OPS	CPLX
cross	unitV	norm	dot	

cross(*vectorA***,***vectorB***)** Returns the cross product of *vectorA* and *vectorB*, both of which are real or complex two-element or three-element vectors; expressed with variables, **cross([a,b,c],[d,e,f])** returns **[bf−ce cd−af ae−bd]**

unitV *vector* Returns a unit vector where each element of a real or complex *vector* is divided by the vector norm

norm *vector* Returns the Frobenius norm ($\sqrt{\Sigma(real^2+imaginary^2)}$) where the sum is over all elements of a real or complex *vector*

dot(*vectorA***,***vectorB***)** Returns the dot product of *vectorA* and *vectorB*, both of which are real or complex vectors; expressed with variables, **dot([a,b,c],[d,e,f])** returns **ad+be+cf**

The VECTR OPS (Operations) Menu [2nd] [VECTR] [F4]

NAMES	EDIT	MATH	OPS	CPLX
dim	Fill	▸Pol	▸Cyl	▸Sph

▸

▸Rec	li▸vc	vc▸li		

Press [STO▸] *to enter the* → *symbol after* *#ofElements*.

dim *vector* Returns the dimension of (or number of elements in) *vector*

#ofElements→**dim***vectorName* Creates a new *vectorName* of the specified length (*#ofElements*); each element is 0

#ofElements→**dim***vectorName* Redimensions *vectorName* to the specified length (*#ofElements*)

Fill(*number***,***vectorName***)** Stores a real or complex *number* to every element in *vectorName*

For the conversion functions below, the three-element vector conversion equations for cylindrical form [r θ z] are:

$x = r\cos\theta$ $y = r\sin\theta$ $z = z$

The three-element vector conversion equations for spherical form [r θ ϕ] are:

$x = r\cos\theta\sin\phi$ $y = r\sin\theta\sin\phi$ $z = r\cos\phi$

Complex elements are valid only for **li▸vc** *and* **vc▸li**.

vector▸**Pol**	Displays a 2-element *vector* in polar form $[r\angle\theta]$
vector▸**Cyl**	Displays a 2- or 3-element *vector* as a cylindrical vector $[r\angle\theta\ \ \mathbf{0}]$ or $[r\angle\theta\ \ z]$
vector▸**Sph**	Displays a 2- or 3-element *vector* as a spherical vector $[r\angle\theta\ \ \mathbf{0}]$ or $[r\angle\theta\ \ \phi]$
complexVector▸**Rec**	Displays a 2- or 3-element *complexVector* in rectangular form $[x\ y]$ or $[x\ y\ z]$
li▸vc *list*	Converts a real or complex *list* into a vector
vc▸li *vector*	Converts a real or complex *vector* into a list

The VECTR CPLX (Complex) Menu [2nd] [VECTR] [F5]

NAMES	EDIT	MATH	OPS	CPLX
conj	real	imag	abs	angle

conj *complexVector*	Returns a vector in which each element is the complex conjugate of the corresponding element of a *complexVector*
real *complexVector*	Returns a real vector in which each element is the real portion of the corresponding element of a *complexVector*
imag *complexVector*	Returns a real vector in which each element is the imaginary portion of the corresponding element of a *complexVector*
abs *Vector*	Returns a real vector in which each element is either the absolute value of the corresponding element of a real *vector* or the magnitude (modulus) of the corresponding element of a *complexVector*
angle *complexVector*	Returns a real vector in which each element is either **0** if the element of *complexVector* is real or the polar angle if the element of *complexVector* is complex; polar angles are calculated as **tan⁻¹(***complex/real***)** adjusted by $+\pi$ in the second quadrant and by $^-\pi$ in the third quadrant

Using Mathematical Functions with Vectors

To add or subtract two vectors, the dimension of vectorA must equal the dimension of vectorB.

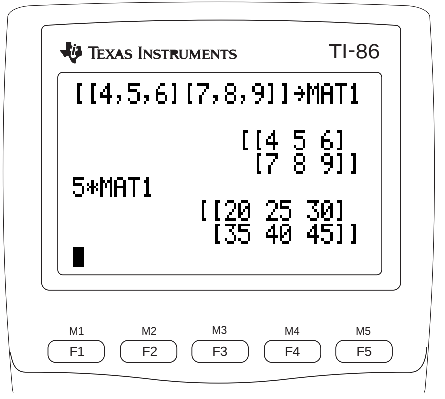
You cannot multiply two vectors or divide one vector by another vector.

== *and* ≠ *are on the* TEST *menu.*

round, **iPart**, **fPart**, *and* **int** *are on the* MATH NUM *menu.*

vectorA+*vectorB*	Adds each *vectorA* element to the corresponding *vectorB* element; returns a vector of the sums
vectorA−*vectorB*	Subtracts each *vectorB* element from the corresponding *vectorA* element; returns a vector of the differences
*vector*****number* or *number*****vector*	Returns a vector that is the product of a real or complex *number* times each element in a real or complex *vector*
*matrix*****vector*	Returns a vector that is the product of each *vector* element times each *matrix* element; *matrix* column dimension and *vector* dimension must be equal
vector/*number*	Returns a vector that is the quotient of each real or complex *vector* element divided by a real or complex *number*
-*vector*	(negation) Changes the sign of each *vector* element
vectorA==*vectorB*	Returns **1** if every corresponding element comparison is true; returns **0** if any is false
vectorA≠*vectorB*	Returns **1** if at least one corresponding element comparison is false
round(*vector*[**,***#ofDecimals*]**)**	Rounds each *vector* element to 12 digits, or rounds to specified *#ofDecimals*
iPart *vector*	Returns the integer part of each real or complex *vector* element
fPart *vector*	Returns the fractional part of each real or complex *vector* element
int *vector*	Returns the greatest integer of each real or complex *vector* element

13 Matrices



TEXAS INSTRUMENTS
TI-86
[[4,5,6][7,8,9]]→MAT1
[[4 5 6]
[7 8 9]]
5*MAT1
[[20 25 30]
[35 40 45]]
M1
M2
M3
M4
M5
F1
F2
F3
F4
F5

Matrices on the TI-86

A matrix is a two-dimensional array, arranged in rows and columns. The matrix elements can be real or complex. You can create, display, and edit matrices on the home screen or in the matrix editor. When you create a matrix, the elements are stored to the matrix name.

Creating, Storing, and Displaying Matrices

The MATRX (Matrix) Menu [2nd] [MATRX]

NAMES	EDIT	MATH	OPS	CPLX
matrix names menu	matrix editor	matrix math menu	matrix operations menu	complex matrix menu

The MATRX NAMES Menu [2nd] [MATRX] [F1]

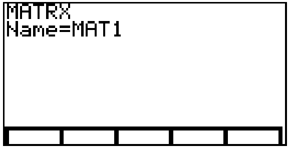
The TI-86 distinguishes between uppercase and lowercase letters in matrix names. For example, **MAT1** *and* **mat1** *are two different vector names.*

The MATRX NAMES menu contains all currently stored matrix names in alphanumeric order. To paste a matrix name to the current cursor location, select it from the menu.

Creating a Matrix in the Matrix Editor [2nd] [MATRX] [F2]

❶ Display the matrix **Name=** prompt screen. [2nd] [MATRX] [F2]

❷ ALPHA-lock is on. The MATRX NAMES menu is displayed. Enter a name from one to eight characters long, starting with a letter. [M] [A] [T] [ALPHA] **1**



An ellipsis (…) at either end of matrix rows indicates additional columns.

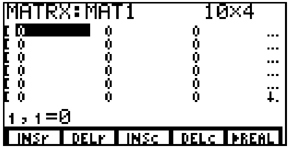
↓ or ↑ in the last column indicates additional rows.

❸ Display the matrix editor and the matrix editor menu.

[ENTER]

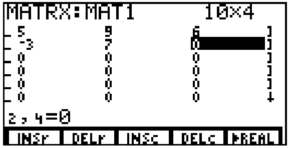
❹ Accept or change the matrix dimensions (*row* × *column*) in the top-right corner of the screen, (1≤*row*≤255 and 1≤*column*≤255); maximum combination is subject to memory availability. The matrix is displayed; all elements are **0**.

10 [ENTER] **4** [ENTER]



❺ Enter each matrix element value at the element prompt (**1,1=** for row 1, column 1). You can enter expressions. To move to the next element, press [ENTER]. To move to the next row, press [▼].

[(-)] **4** [ENTER] **5** [ENTER] **9** [ENTER] **6** [ENTER] **1** [ENTER] [(-)] **3** [ENTER] **7** [ENTER] and so on



The Matrix Editor Menu [2nd] [MATRX] [F2] *matrixName* [ENTER]

INSr	DELr	INSc	DELc	▸REAL

INSr	Inserts a row at the cursor location; shifts subsequent rows down
DELr	Deletes row at the cursor location; shifts subsequent rows up
INSc	Inserts a column at the cursor location; shifts subsequent columns to the right
DELc	Deletes the column at the cursor location; shifts subsequent columns to the left
▸REAL	Converts the displayed complex number matrix to a real number matrix

Creating a Matrix on the Home Screen

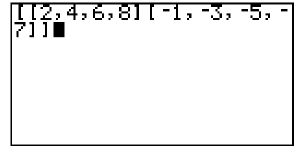
❶ Define the start of the matrix with [, and then define the start of the first row with another [. Enter each element for the row, separating them with commas. Define the end of the first row with].

[2nd] [[ ] [2nd] [[ ]
2 [,] **4** [,] **6** [,]
8 [2nd] [ ]]

The close bracket is not necessary when it precedes [STO▸].

❷ Define the start of each subsequent row with [. Enter the row elements, separating each from the next with a comma. Define the end of each row with]. Then define the end of the matrix with].

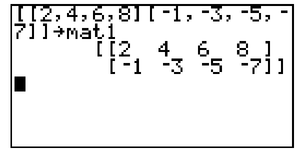
[2nd] [[ ] [(-)] **1** [,]
[(-)] **3** [,] [(-)] **5** [,]
[(-)] **7** [2nd] [ ]]
[2nd] [ ]]



To delete a matrix name from memory, use the MEM DELETE:MATRX *screen (Chapter 17).*

❸ Store the matrix to a matrix name. Either enter a name from one to eight characters long, starting with a letter, or select a name from the MATRX NAMES menu. The matrix is displayed. If newly created, the matrix name becomes a MATRX NAMES menu item.

[STO▸] [2nd] [alpha]
[M] [A] [T]
[ALPHA] [ALPHA] **1**
[ENTER]



Creating a Complex Matrix

If any matrix element is complex, all elements of the matrix are displayed as complex. For example, when you enter the matrix **[[1,2][5,(3,1)]]**, the TI-86 displays **[[(1,0) (2,0)][(5,0) (3,1)]]**.

To create a complex matrix from two real matrices with the same dimensions, the syntax is:

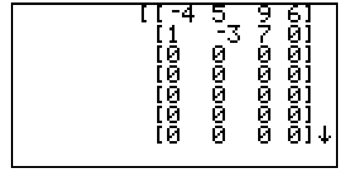
realMatrix+**(0,1)***imaginaryMatrix*→*complexMatrixName*

realMatrix contains the real part of each element and *imaginaryMatrix* contains the imaginary part of each element.

Displaying Matrix Elements, Rows, and Submatrices

To view elements beyond the current screen, use ▶, ▼, ◀, and ▲.

To display an existing matrix on the home screen, enter the matrix name's individual characters or select it from the MATRX NAMES menu, and then press ENTER. The full value of each element is displayed. Elements with very large values may be expressed exponentially.



To display specific elements of *matrixName*, the syntax is:

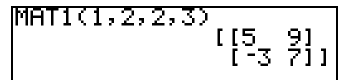
matrixName(*row*,*column*)

To display a row of *matrixName*, the syntax is:

matrixName(*row*)

To display a submatrix of *matrixName*, the syntax is:

matrixName(*beginRow*,*beginColumn*,*endRow*,*endColumn*)



Using a Matrix in an Expression

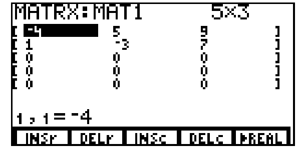
When you execute the expression, the answer is displayed as a matrix.

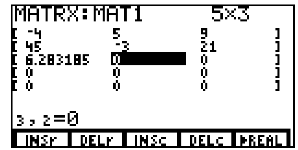
- You can enter the matrix directly (for example, **5*[[2,3][3,5]]**).
- You can use ALPHA and 2nd [alpha] to enter a matrix name's individual characters (for example, **MAT1*3**).
- You can select the matrix name from the MATRX NAMES menu (2nd [MATRX] F1).
- You can select the matrix name from the VARS MATRX screen (2nd [CATLG-VARS] MORE F2).

Editing Matrices in the Matrix Editor

You can use [CLEAR], [DEL], and [2nd] [INS] to edit matrix elements. You also can overwrite existing characters.

1. Display the matrix **Name=** prompt screen. — [2nd] [MATRX] [F2]
2. Enter the matrix name. Either select it from the MATRX NAMES menu or enter the characters. — [M][A][T] [ALPHA] **1**
3. Display the matrix editor. — [ENTER]
4. Edit or accept the row dimension, and then edit or accept the column dimension. — **5** [DEL] [ENTER] **3** [ENTER]
5. Move the cursor to any element and edit it. Continue moving the cursor to other elements. — [▼] **45** [ENTER] [▶] **21** [ENTER] **2** [2nd] [π] [ENTER]
6. Save the changes and leave the matrix editor. — [EXIT]





Editing Matrices on the Home Screen

To change a matrix element value, the syntax is:

value→*matrixName*(*row*,*column*)

To change the values of an entire row of elements, the syntax is:

[*valueA*,*valueB*,...,*value n*]→*matrixName*(*row*)

To change the values of part of a row, beginning at a specified column, the syntax is:

[*valueA*,*valueB*,...,*value n*]→*matrixName*(*row*,*beginColumn*)

To change the values of a submatrix within *matrixName*, the syntax is:

[[*valueA*,...,*value n*] ... [*valueA*,...,*value n*]]→*matrixName*(*beginRow*,*beginColumn*)

The MATRX MATH Menu [2nd] [MATRX] [F3]

NAMES	EDIT	MATH	OPS	CPLX
det	T	norm	eigVl	eigVc

▶

rnorm	cnorm	LU	cond	

det *squareMatrix*	Returns the determinant of *squareMatrix*
matrix$^{\mathsf{T}}$	Returns a transposed matrix; each element's (*row*,*column*) coordinates switch
norm *matrix*	Returns the Frobenius norm ($\sqrt{\Sigma(real^2+imaginary^2)}$) where the sum is over all elements of a real or complex *matrix*
eigVl *squareMatrix*	Returns a list of the normalized eigenvalues of a real or complex *squareMatrix*
eigVc *squareMatrix*	Returns a matrix containing the eigenvectors for a real or complex *squareMatrix*; each column corresponds to an eigenvalue
rnorm *matrix*	(row norm) Returns the largest of the sums of the absolute values of the elements (magnitudes of complex elements) in each row of *matrix*
cnorm *Matrix*	(column norm) Returns the largest of the sums of the absolute values of the elements (magnitudes of complex elements) in each column of *matrix*
LU(*matrix***,** *lMatrixName***,** *uMatrixName***,** *pMatrixName***)**	Calculates the Crout LU (lower-upper) decomposition of a real or complex *matrix*; stores the lower triangular matrix to *lMatrixName*, the upper triangular matrix to *uMatrixName*, and the permutation matrix (which describes the row swaps done during calculation) in *pMatrixName*
cond *squareMatrix*	Calculates **cnorm** *squareMatrix****cnorm** *squareMatrix*$^{-1}$; the closer the product is to 1, the more stable *squareMatrix* can be expected to be in matrix functions

The MATRX OPS (Operations) Menu

[2nd] [MATRX] [F4]

NAMES	EDIT	MATH	OPS	CPLX
dim	Fill	ident	ref	rref

▸

aug	rSwap	rAdd	multR	mRAdd

▸

randM				

Press [STO▸] to enter the → symbol after the close brace.

When you use **aug(**, *the number of rows in matrixA must equal the number of rows in matrixB or the number of elements in vector.*

Elements of matrices created with **randM(** *are integers ≥-9 and ≤9.*

dim *matrix*	Returns the dimensions of *matrix* as a list **{**rows columns**}**
{*rows***,***columns***}→dim** *matrixName*	Creates a new *matrixName* of the specified dimensions; each element is 0
{*rows***,***columns***}→dim** *matrixName*	Redimensions *matrixName* to the specified dimensions
Fill(*number***,***matrixName***)**	Stores a real or complex *number* to each *matrixName* element
ident *dimension*	Returns the square identity matrix of *dimension* × *dimension*
ref *matrix*	Returns the row-echelon form of *matrix*
rref *matrix*	Returns the reduced row-echelon form of *matrix*
aug(*matrixA***,***matrixB***)**	Concatenates *matrixA* and *matrixB*
aug(*matrix***,***vector***)**	Concatenates *matrix* and *vector*
rSwap(*matrix***,***rowA***,***rowB***)**	Returns a matrix after swapping *rowA* and *rowB* of *matrix*
rAdd(*matrix***,***rowA***,***rowB***)**	Returns *matrix* with (*rowA*+*rowB*) of *matrix* stored in *rowB*
multR(*number***,***matrix***,***row***)**	Returns *matrix* with (*row***number*) stored in *row*
mRAdd(*number***,***matrix***,***rowA***,***rowB***)**	Returns *matrix* with ((*rowA***number*)+*rowB*) stored in *rowB*
randM(*rows***,***columns***)**	Creates a matrix of specified dimensions with random elements

The MATRX CPLX (Complex) Menu [2nd] [MATRX] [F5]

NAMES	EDIT	MATH	OPS	CPLX
conj	real	imag	abs	angle

conj *complexMatrix* — Returns a matrix in which each element is the complex conjugate of the corresponding element of a *complexMatrix*

real *complexMatrix* — Returns a real matrix in which each element is the real portion of the corresponding element of a *complexMatrix*

imag *complexMatrix* — Returns a real matrix in which each element is the imaginary portion of the corresponding element of a *complexMatrix*

abs *matrix* — Returns a real matrix in which each element is either the absolute value of the corresponding element of a real *matrix* or the magnitude (modulus) of the corresponding element of a complex *matrix*

angle *complexMatrix* — Returns a real matrix in which each element is either **0** if the element of *complexMatrix* is real or the polar angle if the element of *complexMatrix* is complex; the polar angles are calculated as **tan^{-1}(***imaginary* **/** *real***)** adjusted by +π in the second quadrant and by ⁻π in the third quadrant

Using Mathematical Functions with Matrices

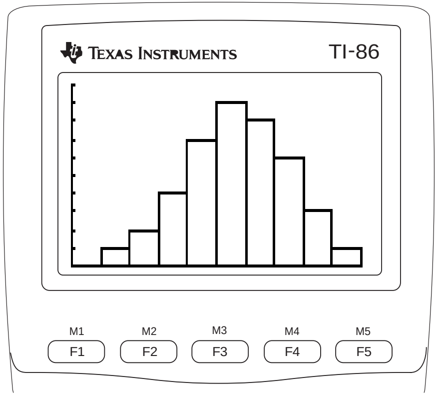
To add or subtract two matrices, the dimensions of matrixA must equal the dimensions of matrixB.

*matrixA***+***matrixB* — Adds each *matrixA* element to the corresponding *matrixB* element; returns a matrix of the sums

matrixA−*matrixB* — Subtracts each *matrixB* element from the corresponding *matrixA* element; returns a matrix of the differences

To multiply two matrices, the column dimension of matrixA must equal the row dimension of matrixB.	*matrixA*******matrixB* or *matrixB*******matrixA*	Multiplies *matrixA* and *matrixB*; returns a square matrix of the products
	*matrix*******number* or *number*******matrix*	Returns a matrix that is the product of a real or complex *number* times each element in a real or complex *matrix*
	*matrix*******vector*	Returns a vector that is the product of each *vector* element times each *matrix* element; the *matrix* column dimension and *vector* dimension must be equal
	-*matrix*	(negation) Changes the sign of each element in *matrix*
To enter **-1**, *press* [2nd] [x^{-1}]. *Do not use* [x-VAR] [^] [(-)] **1**.	*squareMatrix*$^{-1}$	Returns the inverse of *squareMatrix* (not the inverse of each element)
	matrix^2	Squares a square matrix
	*squareMatrix***^***power*	Raises a *squareMatrix* to the designated *power*
e^, **sin**, *and* **cos** *do not return the exponential, sine, or cosine of each matrix element.*	**e^** *squareMatrix*	Returns the square matrix exponential of a real *squareMatrix*
	sin *squareMatrix*	Returns the square matrix sine of a real *squareMatrix*
	cos *squareMatrix*	Returns the square matrix cosine of a real *squareMatrix*
To make relational comparisons, matrixA and matrixB must have equal dimensions.	*matrixA***==***matrixB*	Returns **1** if every corresponding element comparison is true; returns **0** if any is false
	matrixA≠*matrixB*	Returns **1** if at least one corresponding element comparison is false
== *and* ≠ *are on the* TEST *menu.*	**round(***matrix*[**,***#ofDecimals*]**)**	Rounds each *matrix* element to 12 digits or to specified *#of Decimals*
	iPart *matrix*	Returns the integer part of each element of a real or complex *matrix*
round, **iPart**, **fPart**, *and* **int** *are on the* MATH NUM *menu.*	**fPart** *matrix*	Returns the fractional part of each element of a real or complex *matrix*
	int *matrix*	Returns the greatest integer of each element of a real or complex *matrix*

14 Statistics



Statistical Analysis on the TI-86

With the TI-86, you can analyze one-variable and two-variable statistical data, which are stored in lists. One-variable data has one measured variable. Two-variable data has pairs comprising an independent variable and a dependent variable.

When analyzing either kind of data, you can specify a frequency of occurrence for the independent variable values. These specified frequencies must be real numbers ≥ 0.

Setting Up a Statistical Analysis

1. Enter the statistical data into one or more lists (Chapter 11).
2. Calculate the statistical variables or fit a model to the data.
3. Plot the data.
4. Graph the regression equation for the plotted data.

The STAT (Statistics) Menu [2nd] [STAT]

The same list editor is displayed, whether you press [2nd] [STAT] [F2] or [2nd] [LIST] [F4]. For a description of the list editor, see Chapter 11.

CALC	EDIT	PLOT	DRAW	VARS	▸	FCST				
statistical calculations menu	list editor	stat plot menu	statistical drawing tools menu	statistical result variables menu		forecast editor				

Entering Statistical Data

Data for statistical analysis is stored in lists, which you can create and edit in the list editor (Chapter 11), on the home screen (Chapter 11), or in a program (Chapter 16). The TI-86 has three built-in list names for statistics, **xStat** (x-variable list), **yStat** (y-variable list), and **fStat** (frequency list). TI-86 statistical functions use these lists as defaults.

The LIST NAMES Menu [2nd] [STAT] [F2] [F3]

The LIST NAMES *menu shown here has no user-created list names.*

{	}	NAMES	EDIT	OPS
fStat	xStat	yStat		

Editing an element of **xStat** *or* **yStat** *clears any values stored to statistical result variables.*

fStat An automatically updated list of the frequency values used in the last statistical computation requiring a frequency; default is a list where each element is **1**

xStat An automatically updated list of the data from the x-list used in the last statistical analysis

yStat An automatically updated list of the data from the y-list used in the last statistical analysis

The STAT CALC (Calculations) Menu [2nd] [STAT] [F1]

The STAT CALC *functions store the results to statistical result variables (page 193).*

CALC	EDIT	PLOT	DRAW	VARS
OneVa	TwoVa	LinR	LnR	ExpR

▶

PwrR	SinR	LgstR	P2Reg	P3Reg

▶

P4Reg	StReg			

The syntax description for each STAT CALC *menu item follows this section.*

OneVa (one variable) Analyzes data with one measured variable

TwoVa (two variable) Analyzes paired data

For regression analysis, the statistical results are calculated using a least-squares fit.

LinR (linear regression) Fits the model equation y=a+bx to the data; displays values for **a** (slope) and **b** (y-intercept)

LnR (logarithmic regression) Fits the model equation y=a+b ln x to the data using transformed values ln(x) and y; displays values for **a** and **b**

ExpR (exponential regression) Fits the model equation $y=ab^x$ to the data using transformed values x and ln(y); displays values for **a** and **b**; elements in the x-list and y-list elements must be integers

PwrR (power regression) Fits the model equation $y=ax^b$ to the data using transformed values ln(x) and ln(y); displays values for **a** and **b**

***SinR** and **LgstR** are calculated using an iterative least-squares fit.*

SinR (sinusoidal regression) Fits the model equation y=a*sin(bx+c)+d to the data; displays values for **a**, **b**, **c**, and **d**; **SinR** requires at least four data points; it also requires at least two data points per cycle to avoid aliased frequency estimates

LgstR (logistic regression) Fits the model equation $y=a/(1+be^{cx})+d$ to the data; displays **a**, **b**, **c**, and **d**

P2Reg (quadratic regression) Fits the second-degree polynomial $y=ax^2+bx+c$ to the data; displays values for **a**, **b**, and **c**; for three data points, the equation is a polynomial fit; for four or more, it is a polynomial regression; **P2Reg** requires at least three data points

P3Reg (cubic regression) Fits the third-degree polynomial $y=ax^3+bx^2+cx+d$ to the data; displays values for **a**, **b**, **c**, and **d**; for four points, the equation is a polynomial fit; for five or more, it is a polynomial regression; **P3Reg** requires at least four data points

P4Reg (quartic regression) Fits the fourth-degree polynomial $y=ax^4+bx^3+cx^2+dx+e$ to the data; displays values for **a**, **b**, **c**, **d**, and **e**; for five points, the equation is a polynomial fit; for six or more, it is a polynomial regression; **P4Reg** requires at least five data points

StReg (store regression equation) Pastes **StReg(** to the home screen; enter a *variable* and press [ENTER]; the current regression equation is stored to *variable*

When you select **OneVa** *or* **TwoVa**, *the abbreviation* **OneVar** *or* **TwoVar** *is displayed.*

For **OneVa**, the syntax is:

OneVar [*xList*,*frequencyList*]

For **TwoVa**, the syntax is:

TwoVar [*xLlist*,*yList*,*frequencyList*]

For **PwrR** *and* **ExpR**, *the elements of xList and yList must be integers ≥ 1.*

For **LinR**, **LnR**, **ExpR**, **PwrR**, **P2Reg**, **P3Reg**, and **P4Reg**, the syntax is:

TwoVar [*xList*,*yList*,*frequencyList*]

For **SinR**, the syntax is:

SinR [*iterations*,*xList*,*yList*,*period*,*equationVariable*]

Default for iterations is 64.

iterations is the number of iterations to go through; higher values for iterations produce a better fit, but take longer to calculate. *period* is an initial guess at which to begin calculation.

For **LgstR**, the syntax is:

LgstR [*iterations*,*xList*,*yList*,*frequencyList*,*equationVariable*]

To copy the contents **RegEq** to any variable after calculating the regression, the syntax is:

StReg(*variable***)**

Automatic Regression Equation Storage

LinR, **LnR**, **ExpR**, **PwrR**, **SinR**, **LgstR**, **P2Reg**, **P3Reg**, and **P4Reg** are regression models. Each regression model has an optional argument, *equationVariable*, for which you can specify an equation variable, such as **y1**. Upon execution, the regression equation is stored automatically to the specified equation variable, and the function is selected.

Regardless of whether you specify *equationVariable*, the regression equation always is stored to the result variable **RegEq**, which is an item on the STAT VARS menu. The regression equation displays the actual result values.

PRegC is the only statistical result variable calculated for a polynomial regression.

The result for a polynomial regression, sinusoidal regression, or logistic regression is stored in **PRegC** (polynomial/regression coefficients). **PRegC** is a list containing the coefficients for an equation. For example, for **P3Reg**, the result **PRegC={3 5 ⁻2 7}** would represent $y=3x^3+5x^2-2x+7$.

Results of a Statistical Analysis

One- and two-variable statistical functions share the result variables.

When you perform a statistical analysis, the calculated results are stored in the result variables and the data from the lists used in the analysis are stored to **xStat**, **yStat**, and **fStat**. If you edit a list or change the type of analysis, all statistical variables are cleared.

The STAT VARS (Statistical Variables) Menu

[2nd] [STAT] [F5]

The statistical variables are calculated and stored as shown in the table on the next page.

CALC	EDIT	PLOT	DRAW	VARS
$\bar{x}$	σx	Sx	$\bar{y}$	σy

▸	Sy	Σx	Σx²	Σy	Σy²
▸	Σxy	RegEq	corr	a	b
▸	n	minX	maxX	minY	maxY
▸	Med	PRegC	Qrtl1	Qrtl3	tolMe

You can use ALPHA keys, alpha keys, and the CHAR GREEK menu to enter some result variables.

To paste a result variable to the cursor location, either select the variable from the STAT VARS menu or select the variable from the VARS STAT selection screen.

- To use a result variable in an expression, paste it to the appropriate cursor location.
- To display the value of a result variable, paste it to the home screen and press [ENTER].
- To store results to another variable after a calculation, paste the result variable to the home screen, press [STO▸], enter a new variable, and then press [ENTER].

These words are abbreviated in the table:
pop = population
std dev = standard deviation
coeff = coefficient
int = intercept
reg eq = regression equation
pts = points
min = minimum
max = maximum

Result Variables	1-Var Stats	2-Var Stats	Other
mean of x values	$\bar{x}$	$\bar{x}$	
pop std dev of x	σ**x**	σ**x**	
sample std dev of x	**Sx**	**Sx**	
mean of y values		$\bar{y}$	
pop std dev of y		σ**y**	
sample std dev of y		**Sy**	
sum of x values	Σ**x**	Σ**x**	
sum of x^2 values	Σx^2	Σx^2	
sum of y values		Σ**y**	
sum of y^2 values		Σy^2	
sum of x * y		Σ**xy**	
regression equation			**RegEq**
polynomial, **LgstR**, and **SinR** coeff's			**a** (y-int) **b** (slope)

Result Variables	1-Var Stats	2-Var Stats	Other
correlation coeff			**corr**
y-intercept of reg eq			**a**
slope of reg eq			**b**
regression/fit coeff			**a**, **b**
number of data pts	**n**	**n**	
min of x values	**minX**	**minX**	
max of x values	**maxX**	**maxX**	
min of y values		**minY**	
max of y values		**maxY**	
median	**Med**		
1st quartile			**Qrtl1**
3rd quartile			**Qrtl3**
polynomial **LgstR**, and **SinR** reg coeff's			**PRegC**

The first quartile (**Qrtl1**) is the median of the points between **minX** and **Med** (median). The third quartile (**Qrtl3**) is the median of the points between **Med** and **maxX**.

When you calculate a logistic regression, **1** is stored to **tolMet** (**tolMe**) if the TI-86 internal tolerance was met before the calculator arrived at a result; if not met, **0** is stored to **tolMet**.

Plotting Statistical Data

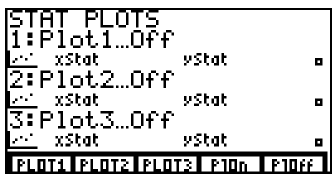
You can plot one, two, or three sets of statistical list data. The five available plot types are scatter plot, xyLine, histogram, modified box plot, and regular box plot.

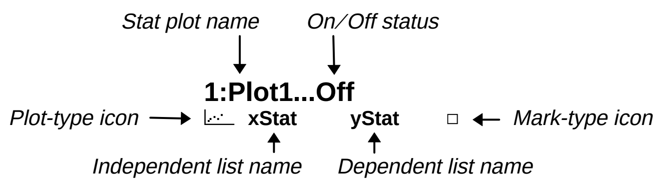
1. Store the statistical data in one or more lists (Chapter 11).
2. Select or deselect functions in the current equation editor as appropriate (Chapter 5).
3. Define the statistical plot.
4. Turn on the plots you want to display.
5. Define the window variables for the graph screen (Chapter 5).
6. Display and explore the plotted graph (Chapter 6).

The STAT PLOT Status Screen [2nd] [STAT] [F3]

The STAT PLOT status screen summarizes the settings for **Plot1**, **Plot2**, and **Plot3**. The illustration below identifies the settings for **Plot1**. This screen is not interactive. To change a setting, select **PLOT1**, **PLOT2**, or **PLOT3** from the STAT PLOT status screen menu.

This screen shows the default stat plot settings. If you select another plot type, some prompts may change.





The STAT PLOT Menu [2nd] [STAT] [F3]

PLOT1	PLOT2	PLOT3	PlOn	PlOff

When you display a stat plot editor, the STAT PLOT *menu remains so that you can easily switch to another stat plot.*

In this guidebook, brackets ([*and*] *) with syntax specify arguments as optional. Do not enter brackets, except with vectors and matrices.*

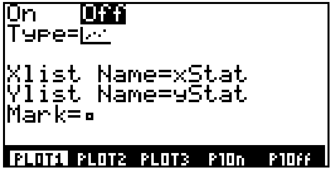
PLOT1	Displays the stat plot editor for **Plot1**
PLOT2	Displays the stat plot editor for **Plot2**
PLOT3	Displays the stat plot editor for **Plot3**
PlOn [**1,2,3**]	Turns on all plots (if you enter no arguments) or turns on specified plots only
PlOff [**1,2,3**]	Turns off all plots (if you enter no arguments) or turns off specified plots only

To turn on or turn off all three stat plots, select **PlOn** or **PlOff** from the STAT PLOT menu. **PlOn** or **PlOff** is pasted to the home screen. Press [ENTER]. All stat plots are now on or off.

Setting Up a Stat Plot

You need not turn on a stat plot to change the settings.

To set up a stat plot, select **PLOT1**, **PLOT2**, or **PLOT3** from the STAT PLOT menu. The stat plot editor for the selected stat plot is displayed. Each stat plot type has a unique stat plot editor. The screen to the right shows the stat plot editor for the default (scatter plot). If you select another plot type, some prompts may change.



Turning On and Turning Off a Stat Plot

You also can use STAT PLOT *menu items* **PlOn** *or* **PlOff** *to turn on or turn off stat plots.*

When you display a stat plot editor, the cursor is on the **On** option.

- ♦ To turn on the stat plot, press [ENTER].
- ♦ To turn off the stat plot, press [▶] [ENTER].

The PLOT TYPE Menu (Selecting a Plot Type)

To display the PLOT TYPE *menu, move the cursor onto the plot type icon at the* **Type=** *prompt.*

When you select a plot type, the appearance of the stat plot editor may change.

PLOT1	PLOT2	PLOT3	PlOn	PlOff
SCAT	xyLINE	MBOX	HIST	BOX

At this prompt...	Enter this information:	Default is:	Displayed menu is:
Xlist Name=	independent-data list name	**xStat**	LIST NAMES menu
Ylist Name=	dependent-data list name	**yStat**	LIST NAMES menu
Freq=	frequency list name (or **1**)	**fStat** (default value: **1**)	LIST NAMES menu
Mark=	plot mark (□ or + or •)	□ (none for **HIST** or **BOX**)	PLOT MARK menu

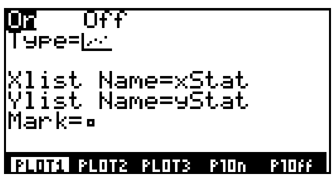
- Any list you enter at the **Xlist Name=** prompt is stored to the list name **xStat**.
- Any list you enter at the **Ylist Name=** prompt is stored to the list name **yStat**.
- Any list you enter at the **Freq=** prompt is stored to **fStat**.

Plot Type Characteristics

Stat plots are displayed on the graph screen (GRAPH F5), as defined by the window variable values (Chapter 5). Some graph tools apply to stat plots.

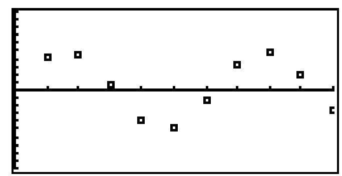
In these stat plot examples, all functions are deselected. Also, menus are cleared from the screen with CLEAR.

SCAT (scatter plot) plots the data points from **Xlist Name** and **Ylist Name** as coordinate pairs, representing each point with a box (□), cross (+), or dot (•) mark type. **Xlist Name** and **Ylist Name** must be the same length. **Xlist Name** and **Ylist Name** can be the same list.

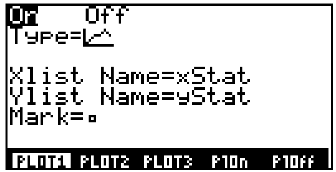


For the example:
xStat={1 2 3 4 5 6 7 8 9 10}
yStat=5 sin(xStat)

Window variable values:
xMin=0 **yMin=⁻10**
xMax=10 **yMax=10**

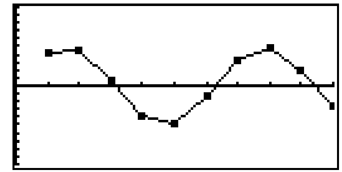


xyLINE is a scatter plot in which the data points are plotted and connected in order of appearance in **Xlist Name** and **Ylist Name**. You may want to use **SortA** or **SortD** from the LIST OPS menu (Chapter 11) to sort the lists before you plot them.

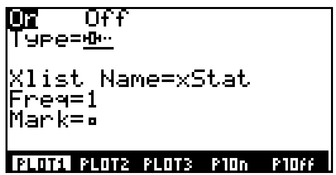


For the example:
xStat={1 2 3 4 5 6 7 8 9 10}
yStat=5 sin(xStat)

Window variable values:
xMin=0 **yMin=⁻10**
xMax=10 **yMax=10**

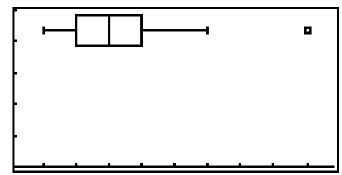


MBOX (modified box plot) plots one-variable data, like the regular box plot, except that the points are 1.5 ∗ Interquartile Range beyond the quartiles. (The Interquartile Range is defined as the difference between the third quartile **Q3** and the first quartile **Q1**.) These points are plotted individually beyond the whisker, using the **Mark** (▫ or + or •) you select.



For the example:
xStat={1 2 2 2.5 3 3.3 4 4 2 6 9}

Window variable values are set by selecting **ZDATA** *from the* GRAPH ZOOM *menu.*

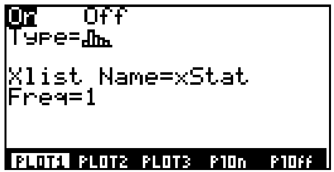


Whiskers are the lines protruding from the sides of the box.

You can trace these points, which are called outliers. When outliers exist, the end of each whisker will display an **x=** prompt. When no outliers exist, **xMin** and **xMax** are the prompts for the end of each whisker. **Q1**, **Med** (median), and **Q3** define the box.

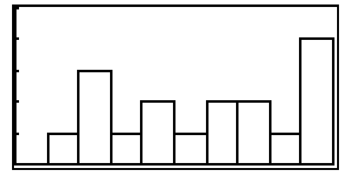
Modified box plots are plotted with respect to **xMin** and **xMax**, but ignore **yMin** and **yMax**. When two modified box plots are plotted, the first one plots at the top of the screen and the second plots in the middle. When three are plotted, the first one plots at the top, the second in the middle, and the third at the bottom.

HIST (histogram) plots one-variable data. The **xScl** window variable value determines the width of each bar, beginning at **xMin**. **ZDATA** (GRAPH ZOOM menu) adjusts **xMin**, **xMax**, **yMin**, and **yMax** to include all values, and also adjusts **xScl**. (**xMax** − **xMin**) / **xScl** ≤ 47 must be true. A value that occurs on the edge of a bar is counted in the bar to the right.



For the example:
xStat={1 2 2 2 3 8 9 5 6 6 7 7 4 4 9 9 9}

Window variable values:
xMin=0 **yMin=0**
xMax=10 **yMax=5**

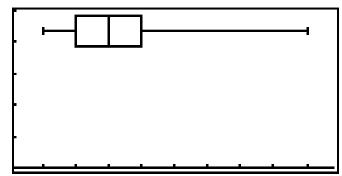


Whiskers are the lines protruding from the sides of the box.

BOX (regular box plot) plots one-variable data. The whiskers on the plot extend from the minimum data point in the set (**xMin**) to the first quartile (**Q1**) and from the third quartile (**Q3**) to the maximum point (**xMax**). The box is defined by **Q1**, **Med** (median), and **Q3**.

For the example:
xStat={1 2 2 2.5 3 3.3 4 4 2 6 9}

Window variable values are set by selecting **ZDATA** *from the* GRAPH ZOOM *menu.*



Box plots are plotted with respect to **xMin** and **xMax**, but ignore **yMin** and **yMax**. When two box plots are plotted, the first one plots at the top of the screen and the second plots in the middle. When three are plotted, the first one plots at the top, the second in the middle, and the third at the bottom.

The STAT DRAW Menu [2nd] [STAT] [F4]

CALC	EDIT	PLOT	DRAW	VARS
HIST	SCAT	xyLINE	BOX	MBOX

▶

DRREG	CLDRW	DrawF	STPIC	RCPIC

When you select any of the first five STAT DRAW *menu items, the TI-86 plots the data stored in the lists* **xStat** *and* **yStat**.

HIST	Draws a histogram of one-variable data
SCAT	Draws a scatter plot of the data points
xyLINE	Draws the data points and a line connecting each point to the next point
BOX	Draws a box plot of the data points
MBOX	Draws a modified box plot of the data points
DRREG	(draw regression equation) Draws the current regression equation
CLDRW	(clear drawings) Displays the current graph with no drawings
DrawF *expression*	(draw function) Plots *expression* as a drawing
STPIC	(store picture) Displays the picture variable **Name=** prompt; enter a valid variable name, starting with a letter, and then press [ENTER] to store the current picture
RCPIC	(recall picture) Displays the picture variable **Name=** prompt and menu; select or enter a valid variable name, and then press [ENTER]; the stored picture is redrawn

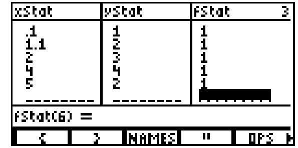
Forecasting a Statistical Data Value

Using the forecast editor, you can forecast an x-value or y-value based on the current regression equation. To use the forecast editor, a regression equation must be stored to **RegEq**.

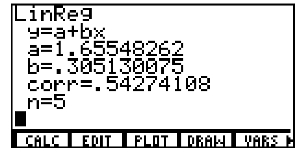
Values entered at forecast editor prompts must be real numbers or expressions that evaluate to real numbers.

*If the most recent calculation was a polynomial regression, you can only forecast the **y** value.*

1. Enter stat data in the list editor. The screen to the right shows all **fStat** elements as **1**, but you need not enter them. **1** is the default for all **fStat** elements. However, if other elements are stored to **fStat**, you must clear them.
   [2nd] [STAT] [F2]
   [.] **1** [▼] **1** [.] **1**
   [▼] **2** [▼] **4** [▼] **5**
   [▼] [▶] **1** [▼] **2**
   [▼] **3** [▼] **4** [▼] **2**



2. Display the home screen.
   [EXIT]
3. Execute a linear regression for **xStat** and **yStat**. The statistical results are displayed.
   [2nd] [STAT] [F1] [F3] [ENTER]
4. Remove the STAT CALC menu to display all results, including **n**.
   [EXIT]

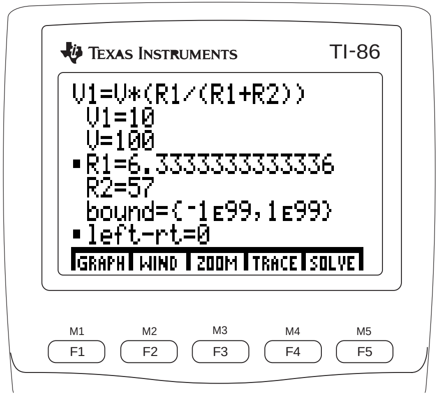


5. Display the forecast editor. The current regression model is displayed on the top line.
   [MORE] [F1]
6. Enter **x=3** , and then move the cursor to the **y=** prompt.
   3 [▼]

7. Select **SOLVE** from the forecast editor menu to solve for **y** at **x=3**. A small square indicates the solution. You can continue to use the forecast editor with other values for **x** or **y**.
   [F5]

When you use **FCST**, the values of **x**, **y**, and **Ans** are not updated. To store the **x** value or **y** value, move the cursor onto the variable to be stored, press [STO▶], enter a valid variable name at the **Sto** prompt, and then press [ENTER].

15 Equation Solving



Preview: The Equation Solver [2nd] [SOLVER]

With the equation solver, you can enter an expression or equation, store values to all but one variable in the expression or equation, and then solve for the unknown variable. These steps introduce the solver. For details, read this chapter.

The VARS EQU *menu is a menu version of the* VARS EQU *screen (Chapter 2).*

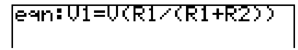
The example uses a formula for a voltage divider.

R1 *and* **R2** *represent resistors.*

V *and* **V1** *represent voltage.*

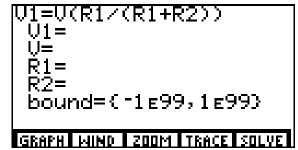
1. Display the equation-entry editor. The VARS EQU menu is displayed on the bottom of the screen.

   [2nd] [SOLVER]



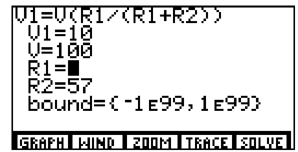
2. Enter an equation. When you press [ENTER], the interactive-solver editor and solver menu are displayed.

   [ALPHA] [V] **1** [ALPHA] [=] [ALPHA] [V] [(] [ALPHA] [R] **1** [÷] [(] [ALPHA] [R] **1** [+] [ALPHA] [R] **2** [)] [)] [ENTER]



3. Enter values for each variable, except the unknown variable **R1**. Some variables may have values stored to them already.

   10 [▼] **100** [▼] [▼] **57**

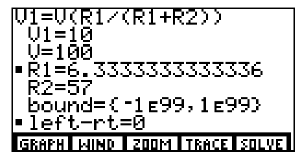


4. Move the cursor to the variable for which you want to solve. You may enter a guess.

   [▲]

5. Solve the equation for the variable. Small squares mark both the solution variable and the equation **left−rt=0** (the left side of the equation minus the right side of the equation). If you edit a value or leave the screen, the squares disappear.

   [F5]

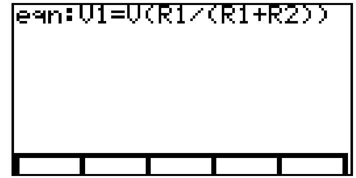


To solve for the unknown variable in an equation on the home screen or in the program editor, select **Solver(** *from the* CATALOG *(A to Z Reference).*

Entering an Equation in the Equation-Entry Editor

The equation solver uses two editors: the equation-entry editor, where you enter and edit the equation you want to solve, and the interactive-solver editor, where you enter known variable values, select the variable for which you want to solve, and display the solution.

To display the equation-entry editor, press 2nd [SOLVER]. In this editor, you can:



The equation can have more than one variable to the left of the equal sign, as in **A+B=C+sin D**.

- Enter an equation directly.
- Enter a defined equation variable's individual characters or select it from the VARS EQU menu.
- Recall the contents of a defined equation variable.

As you enter or edit the equation, the TI-86 automatically stores it to the variable **eqn**.

You can display other menus in the equation-entry editor.

An ellipsis (...) indicates that an entered equation continues beyond the screen. To move directly to the start of the equation, press 2nd ◀; to move directly to the end, press 2nd ▶.

The VARS EQU menu is a menu version of the VARS EQU screen (Chapter 2). The items are all variables to which an equation is stored. This includes all selected and deselected equation variables defined in the equation editors of all four graphing modes (Chapters 5, 8, 9, and 10). The menu items are in alphanumeric order.

- If you select an equation variable from the menu, the variable is pasted to the cursor location, overwriting characters for the length of the variable name.
- If you press 2nd [RCL], select an equation variable from the menu, and then press ENTER, the variable contents are inserted at the cursor location.

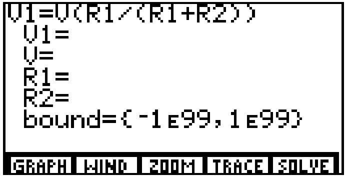
If you enter an equation variable, the TI-86 automatically converts it to the equation **exp=***equationVariable*. If you enter an expression directly, the TI-86 automatically converts the expression to the equation **exp=***expression*.

Setting Up the Interactive-Solver Editor

In the example, the equation **V1=V(R1/(R1+R2))** *was entered in the equation-entry editor.*

If you entered an expression for **eqn**, *then* **exp=** *is the first variable prompt on the interactive-solver editor.*

After you have stored an equation to **eqn** in the equation-entry editor, press [ENTER] to display the interactive-solver editor.



The equation is displayed across the top of the editor. Each variable in the equation is displayed as a prompt. Values already stored to variables are displayed; undefined variables are blank. The solver menu is displayed on the bottom of the editor (page 206).

bound={⁻1E99,1E99} is a list containing the default lower bound (⁻**1E99**) and the default upper bound (**1E99**). You can edit the bounds (below).

Entering Variable Values

To solve for an unknown variable, you must define every other variable in the equation. When you enter or edit a variable value in the interactive-solver editor, the new value is stored to the variable in memory. For any variable, you may enter an expression, which is evaluated when you press [ENTER], [▼], [▲], or [EXIT]. Expressions must resolve to real numbers at each step of the calculation.

Controlling the Solution with Bounds and a Guess

The solver seeks a solution only within the specified bounds. Whenever you display the interactive-solver editor, the default **bound={⁻1E99,1E99}** is displayed. These are the maximum bounds for the TI-86.

The TI-86 solves equations through an iterative process. To control that process, you can enter lower bounds and upper bounds that are close to the solution, and enter a guess within those bounds in the prompt for the unknown variable.

Controlling the process with specific bounds and a guess helps the TI-86 in two ways.

♦ It finds a solution more quickly.

♦ It is more likely to find the solution you want when an equation has multiple solutions.

lowerBound<upperBound must be true.

You can enter a list variable at the **bound=** *prompt if a valid two-element list is stored to it.*

To set more precise bounds at the **bound=** prompt, the syntax is:

bound={*lowerBound*,*upperBound***}**

At the prompt for the unknown variable, you may enter a guess or a list of two guesses. If you do not enter a guess, the TI-86 uses (*lowerBound*+*upperBound*)/2 as a guess.

On the solver graph (page 207), you can guess a solution by moving the free-moving cursor or trace cursor to a point on the graph between *lowerBound* and *upperBound*. To solve for the unknown variable using the new guess, select **SOLVE** from the solver graph menu. The solution is displayed on the interactive-solver editor.

Editing the Equation

If you exit the equation solver, any equation stored to eqn is displayed when you return to the equation solver.

To edit the equation stored to **eqn** when the interactive-solver editor is displayed, press [▲] until the cursor is on the equation. The equation-entry editor is displayed. The TI-86 automatically stores the edited equation to **eqn** as you edit.

If you store an equation to **eqn** by recalling the contents of an equation variable, such as **y1**, and then edit the equation stored to **eqn**, the original equation (in **y1**, for example) is not changed. Likewise, subsequently editing the contents of the equation variable (**y1**, for example) does not change **eqn**.

You can display other menus in the interactive-solver editor

The Solver Menu [2nd] [SOLVER] *equation* [ENTER]

GRAPH	WIND	ZOOM	TRACE	SOLVE

- GRAPH — graphs the equation in **eqn**
- WIND — window editor
- ZOOM — solver zoom menu
- TRACE — graphs **eqn** and activates the trace cursor
- SOLVE — solves for the unknown variable or displays the interactive-solver editor

To display the window editor, select **WIND** from the solver menu.

When you select **GRAPH** or **WIND** from the solver menu, **EDIT** replaces the item you selected on the menu. To return to the interactive-solver editor from the graph or window editor, select **EDIT**.

Solving for the Unknown Variable

After you have stored all known variable values, set the bounds, and entered a guess (optional), move the cursor to the prompt for the unknown variable.

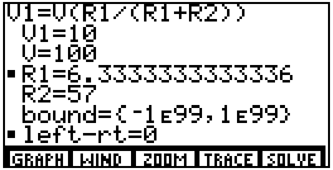
An ellipsis (...) indicates that the variable value continues beyond the screen. To scroll the value, press [▶] and [◀].

The squares disappear when you edit any value.

After solving, you can edit a variable value or edit the equation, and then solve for the same variable or another variable in the equation.

To solve, select **SOLVE** from the solver menu ([F5]).

- ♦ A small square marks the variable for which you solved. The solution value is displayed.
- ♦ A small square also marks the **left-rt=** prompt. The value at this prompt is the value of the left side of the equation minus the value of the right side of the equation, evaluated at the new value of the variable for which you solved. If the solution is precise, **left-rt=0** is displayed.

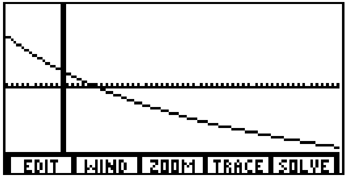


Some equations have more than one solution. To look for additional solutions, you can enter a new guess or set new bounds, and then solve for the same variable.

Graphing the Solution

The graph to the right plots the solution from the example on page 202. The window variable values are: **xMin=⁻10 yMin=⁻50 xMax=50 yMax=50**

When you select **GRAPH** from the solver menu ([F1]), the solver graph is displayed with the free-moving cursor.



- The vertical axis represents the result of the left side of the equation minus the right side of the equation (left−right) at each independent variable value.
- The horizontal axis represents the independent variable for which you solved the equation.

On the graph, solutions exist for the equation where **left−rt=0**, which is where the graph intersects the x-axis. The solver graph:

- Uses the current window and format settings (Chapter 5).
- Does not graph the solution according to the current graphing mode.
- Always graphs a solution as a function graph.
- Does not graph selected functions or turned on stat plots along with the solution.

Solver Graph Tools

You can use the free-moving cursor or trace cursor to select a guess on the graph.

You can explore the graph of a solution with the free-moving cursor, as you would on any other graph. When you do, the coordinate values for the variable (the x-axis) and the value **left−rt** (the y-axis) are updated.

To activate the trace cursor, select **TRACE** from the solver menu. Panning, QuickZoom, and entering a specific value (Chapter 6) are available with the trace cursor on the solver graph.

To return to the solver menu from a trace, press [EXIT].

The Solver ZOOM Menu [2nd] [SOLVER] *equation* [ENTER] [F3]

GRAPH	WIND	ZOOM	TRACE	SOLVE
BOX	ZIN	ZOUT	ZFACT	ZSTD

Chapter 6 and the A to Z Reference describe these features in detail.

BOX Draws a box to redefine the viewing window (Chapter 6)

ZIN Magnifies the graph around the cursor by factors of **xFact** and **yFact** (Chapter 6)

ZOUT Displays more of the graph around the cursor by factors of **xFact** and **yFact** (Chapter 6)

ZFACT Displays the ZOOM FACTORS screen (Chapter 6)

ZSTD Displays the graph in standard dimensions; resets the default window variable values for **Func** graphing mode

The Simultaneous Equation Solver [2nd] [SIMULT]

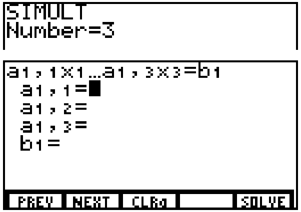
The simultaneous equation solver solves systems of up to 30 linear equations with 30 unknowns.

Entering Equations to Solve Simultaneously

The SIMULT coefficients are not variables.

You can display other menus in the coefficients-entry screen.

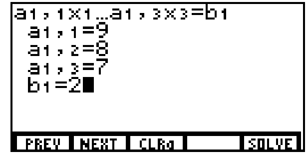
1. Display the SIMULT number screen. [2nd] [SIMULT]
2. Enter an integer ≥ 2 and ≤ 30 for the number of equations. The coefficients-entry editor for the first equation (for a system of n equations and n unknowns) is displayed. The SIMULT ENTRY menu also is displayed. **3** [ENTER]



To move from the coefficients-entry editor for one equation to the editor for another equation, select **PREV** *or* **NEXT**.

❸ Enter a real or complex value (or an expression that resolves to one) for each coefficient in the equation and for **b_1**, which is the solution to that equation.

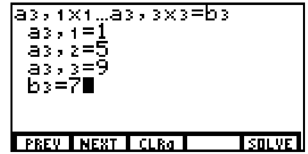
9 [▼] **8** [▼] **7** [▼] **2**



To move among coefficients, press [▼], [▲], *or* [ENTER]. *From the last or first coefficient, these keys move to the next or previous coefficients-entry screen, if possible.*

❹ Display the coefficients-entry screen for the second and third equation, and enter values for them.

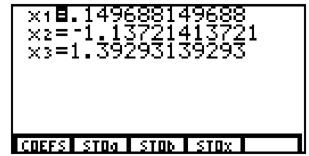
[▼] (or [ENTER] or [F2]) **5** [▼] [(−)] **6** [▼] [(−)] **4** [▼] **2**
[▼] **1** [▼] **5** [▼] **9** [▼] **7**



Ellipses indicate that a value continues beyond the screen. Press [▶] *and* [◀] *to scroll the value.*

❺ Solve the equations. The results of the polynomial are calculated and displayed on the result screen. Results are not stored to variables and cannot be edited. The SIMULT RESULT menu is displayed.

[F5]



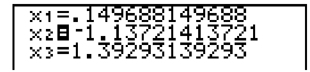
Storing Equation Coefficients and Results to Variables

- To store coefficients $\mathbf{a}_{1,1}$; $\mathbf{a}_{1,2}$;...;$\mathbf{a}_{n,n}$ to an $n \times n$ matrix, select **STOa**.
- To store solutions $\mathbf{b}_1$,$\mathbf{b}_2$,...,$\mathbf{b}_n$ to a vector of dimension n, select **STOb**.
- To store the results $\mathbf{x}_1$, $\mathbf{x}_2$,..., $\mathbf{x}_n$ to a vector of dimension n, select **STOx**.

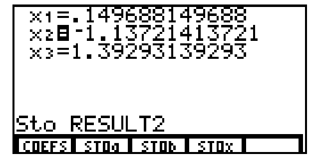
To store a single value on the coefficients-entry screen or result screen, follow these steps.

To switch to the coefficients-entry screen, select **COEFS** *from the* SIMULT RESULT *menu.*

1. Move the cursor to the = sign next to the coefficient or result you want to store. [▼]



2. Display the variable **Name=** prompt. ALPHA-lock is on. [STO▶]



3. Enter the variable to which you want to store the value. [R] [E] [S] [U] [L] [T] [ALPHA] **2**
4. Store the value. The variable name becomes an item on the VARS REAL screen or VARS CPLX screen. [ENTER]

To solve equations simultaneously on the home screen or in a program, select **simult(** *from the* CATALOG.

To return to the coefficients-entry screen, where you can edit coefficients and calculate new solutions, select **COEFS** from the SIMULT RESULT menu.

The Polynomial Root-Finder [2nd] [POLY]

The root finder solves up to 30th-order real or complex polynomials.

Entering and Solving a Polynomial

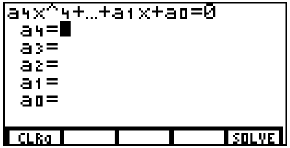
The POLY coefficients are not variables.

You can display other menus in the coefficients-entry editor.

Ellipses indicate that a value continues beyond the screen. Press [▸] and [◂] to scroll the value.

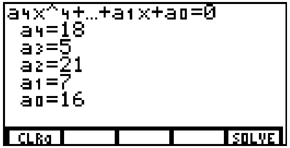
❶ Display the POLY order screen. — [2nd] [POLY]

❷ Enter an integer between 2 and 30. The coefficients-entry editor is displayed with the equation across the top, the coefficient prompts along the left side, and the POLY ENTRY menu on the bottom. — **4** [ENTER]

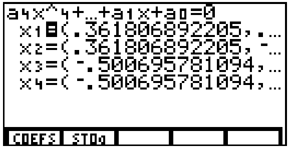


❸ Enter a real or complex value (or an expression that resolves to one) for each coefficient. — **18** [▾] **5** [▾] **21** [▾] **7** [▾] **16**

To clear all coefficients, select **CLRa** from the POLY ENTRY menu.



❹ Solve the equation. The roots of the polynomial are calculated and displayed. Results are not stored to variables and you cannot edit them. Also, the POLY RESULT menu is displayed. Results can be complex numbers. — [F5]

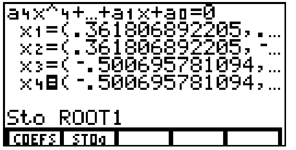


To switch to the coefficients-entry screen, select **COEFS** *from the* POLY RESULT *menu.*

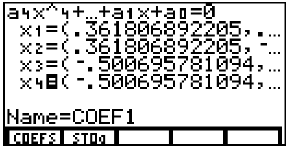
To find roots on the home screen or in a program, select **poly** *from the* CATALOG*.*

Storing a Polynomial Coefficient or Root to a Variable

1. Move the cursor to the = sign next to the coefficient or root value you want to store. [▼] [▼] [▼]
2. Display the **Sto** prompt. ALPHA-lock is on. [STO▶]
3. Enter the variable to which you want to store the value. [R] [O] [O] [T] [ALPHA] **1**
4. Store the value. [ENTER]

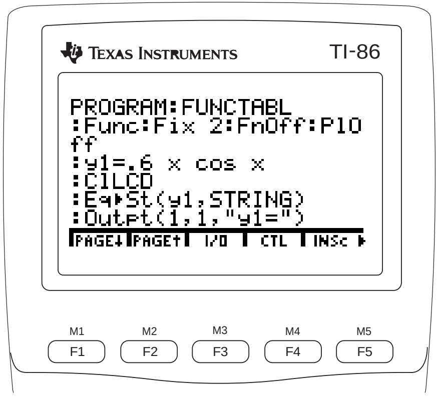


5. Display the **Name=** prompt for the coefficents list name. ALPHA-lock is on. [F2]
6. Enter the list variable name to which you want to store the coefficients. [C] [O] [E] [F] [ALPHA] **1**
7. Store the polynomial coefficient values. [ENTER]



To return to the coefficients-entry screen, where you can edit coefficients and calculate new solutions, select **COEFS** from the POLY RESULT menu.

16 Programming



Writing a Program on the TI-86

A program is a set of expressions, instructions, or both, which you enter or download. Expressions and instructions in the program are executed when you run the program.

You can use most TI-86 features in a program. Programs can retrieve and update all variables stored to memory. Also, the program editor menu has input/output commands, such as **Input** and **Disp**, and program control commands, such as **If**, **Then**, **For**, and **While**.

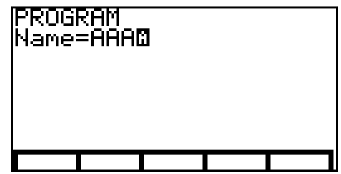
The PRGM Menu [PRGM]

NAMES	EDIT			
program names menu	program editor			

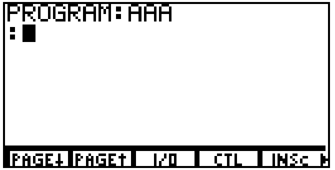
Creating a Program in the Program Editor

The TI-86 distinguishes between uppercase and lowercase letters in program names. For example, **ABC***,* **Abc***, and* **abc** *would be three different program names.*

To begin writing a program, select **EDIT** from the PRGM menu ([PRGM] [F2]). The program **Name=** prompt and PRGM NAMES menu are displayed. ALPHA-lock is on. Enter a program name from one to eight characters long, beginning with a letter. To edit an existing program, you can select the name from the PRGM NAMES menu.



After you enter a program name, press ENTER. The program editor and program editor menu are displayed. The program name is displayed at the top of the screen. The cursor is on the first command line, which begins with a colon. The TI-86 automatically places a colon at the beginning of each command line.



As you write the program, the commands are stored to the program name.

The Program Editor Menu PRGM F2 *programName* ENTER

PAGE↓	PAGE↑	I/O	CTL	INSc	▸	DELc	UNDEL	:		
page down	page up	input/output menu	program control menu	insert a blank command line		delete (cut) a command line	undelete (paste) a deleted command line	paste a colon		

The PRGM I/O (Input/Output) Menu PRGM F2 *programName* ENTER F3

PAGE↓	PAGE↑	I/O	CTL	INSc						
Input	Promp	Disp	DispG	DispT	▸	ClTbl	Get	Send	getKy	ClLCD
					▸	"	Outpt	InpSt		

The PRGM I/O *menu items are instructions. The actions they perform occur as the program runs.*

To see examples that show how to use PRGM I/O menu items in programs, refer to the A to Z Reference.

*If you enter an expression for variable at an **Input** or **Prompt** prompt, it is evaluated and stored.*

*For **Input** and **Prompt**, built-in variables such as **y1** and **r1** are not valid as variable.*

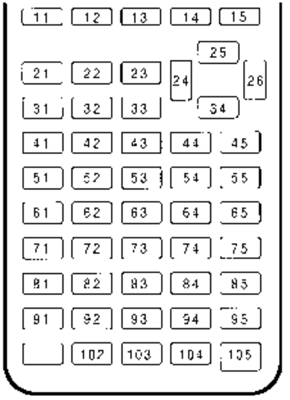
*To halt the program temporarily after **Disp** or **DispG** and examine what the program is displaying, enter **Pause** on the next command line (page 219).*

Input	Displays the current graph and lets you use the free-moving cursor
Input *variable*	Pauses a program, displays **?** as a prompt, and then stores your response to *variable*
Input *promptString*,*variable* **Input "***string***",***variable*	Pauses a program, displays *promptString* or *string* (up to 21 characters) as a prompt, and then stores your response to *variable*
Input "CBLGET",*variable*	Although using **Get(** is preferred on the TI-86, you can use **Input** to receive *variable* from a CBL 2/CBL, CBR, or TI-86 (TI-85 compatible)
Prompt *variableA* [,*variableB*,*variableC*,...]	Displays each *variable* with **?** to prompt you to enter a value for that *variable*
Disp	Displays the home screen
Disp *valueA*,*valueB*,...	Displays each *value*
Disp *variableA*,*variableB*,...	Displays the value stored to each *variable*
Disp "*textA***","***textB***"**,...	Displays each *text* string on the left side of the current display line
DispG	Displays the current graph
DispT	Displays the current table and temporarily halts the program
ClTbl	Clears the current table if **Indpnt: Ask** is set (Chapter 7)
Get(*variable***)**	Gets data from a CBL 2/CBL, CBR, or another TI-86 and stores it to *variable*
Send(*listName***)**	Sends the contents of *listName* to a CBL 2/CBL or CBR
getKy	Returns a number corresponding to the last key pressed, according to the key code diagram (page 217); if no key was pressed, returns **0**
ClLCD	Clears the home screen (LCD stands for liquid crystal display)

"*string*"	Specifies the beginning and end of a *string*
Outpt(*row*,*column*,**"***string***")** **Outpt(***row*,*column*,*stringName***)** **Outpt(***row*,*column*,*value***)** **Outpt(***row*,*column*,*variable***)**	Displays *string*, *stringName*, *value*, or a value stored to *variable* beginning at the specified *row* and *column* on the display
Outpt("CBLSEND",*listName***)**	Although using **Send(** is preferred on the TI-86, you can use **Outpt(** to send *listName* to a CBL 2/CBL or CBR (for TI-85 compatibility)
InpSt *promptString*,*variable* **InpSt** *variable*	Pauses a program, displays *promptString* or **?**, and waits for a response; stores the response to *variable* always as a string; omit quotation marks from your response

The TI-86 Key Code Diagram

When **getKy** is encountered in a program, it returns a number corresponding to the last key pressed, according to the key code diagram to the right. If no key has been pressed, **getKy** returns **0**. Use **getKy** inside loops to transfer control, such as when you create a video game.



This program returns the key code of each key you press.

```
:Float
:0→A
:Lbl TOP
:getKy→A
:If A>0
:Disp A
:Goto TOP
```

To break (interrupt) the program, press [ON] and then press [F5].

The PRGM CTL Menu [PRGM] [F2] *programName* [ENTER] [F4]

PAGE↓	PAGE↑	I/O	CTL	INSc
If	Then	Else	For	End

▸

While	Repea	Menu	Lbl	Goto

▸

IS>	DS<	Pause	Retur	Stop

▸

DelVa	GrStl	LCust		

To see examples that show how to use PRGM CTL menu items in programs, refer to the A to Z Reference.

If, **While**, *and* **Repeat** *instructions can be nested.*

For(*loops can be nested.*

If *condition*	If *condition* is false (evaluates to 0), the next program command is skipped; if *condition* is true (evaluates to a nonzero value), the program continues on to the next command
Then	Following **If**, executes a group of commands if *condition* is true
Else	Following **If** and **Then**, executes a group of commands if *condition* is false
For(*variable***,***begin***,***end* [**,***step*]**)**	Starting at *begin*, repeats a group of commands by an optional real *step* until *variable* > *end*; default *step* is 1
End	Identifies the end of a group of program commands; **For(**, **While**, **Repeat**, and **Else** groups must end with **End**; **Then** groups without an associated **Else** instruction also must end with **End**
While *condition*	Repeats a group of commands while *condition* is true; *condition* is tested when the **While** instruction is encountered; typically, the expression that defines *condition* is a relational test (Chapter 3)
Repeat *condition*	Repeats a group of commands until *condition* is true; *condition* is

	tested when the **End** instruction is encountered
Menu(*item#***,"***title1***",** *label1*[**,***item#***, "***title2***",***label2***,**...]**)**	Sets up branching within a program as selected from menu keys [F1] through [F5]; when encountered, displays the first of up to 3 menu groups (up to 15 *titles*); when you select a *title*, the program branches to the *label* that the *title* represents; *item#* is an integer ≥ 1 and ≤ 15 that specifies *title*'s menu placement; *title* is a text string from one to eight characters long (may be abbreviated in the menu)
Lbl *label*	Assigns a *label* to a program command; label can be one to eight characters long, starting with a letter
Goto *label*	Transfers control to the program branch labeled with *label*
IS>(*variable***,***value***)**	Adds 1 to *variable*; if the answer is > *value*, the next command is skipped; if the answer is ≤ *value*, the next command is executed; *variable* cannot be a built-in variable
DS<(*variable***,***value***)**	Subtracts 1 from *variable*; if the answer is < *value*, the next command is skipped; if the answer is ≥ *value*, the next command is executed; *variable* cannot be a built-in variable
Pause	Halts the program so that you can examine results, including displayed graphs and tables; to resume the program, press [ENTER]
Pause *value*	Displays *value* on the home screen so that you can scroll large values, such as lists, vectors, or matrices; to resume, press [ENTER]
Return	Exits a subroutine (page 224) and returns to the calling program, even if encountered within nested loops; within the main program, stops the program and returns to the home screen (an implied **Return** exits each subroutine upon completion and returns to the calling program)
Stop	Stops a program and returns to the home screen

DelVar(*variable***)**	Deletes from memory *variable* (except program names) and its contents
GrStl(*function#***,***graphStyle#***)**	Specifies the graph style represented by *graphStyle#* for the function represented by *function#*; *function#* is the number part of an equation variable, such as the **5** in **y5**; *graphStyle#* is an integer ≥ 1 and ≤ 7, where **1** = (line), **2** = (thick), **3** = (shade above), **4** = (shade below), **5** = (path), **6** = (animate), and **7** = (dotted)
LCust(item#***,"***title***" [,***item#***,"***title***",...])**	Loads (defines) the TI-86 custom menu, which is displayed when you press [CUSTOM]; item# is an integer ≥ 1 and ≤ 15; *title* is a string with one to eight characters (may be abbreviated in the menu)

Entering a Command Line

A command line that is longer than the screen is wide automatically continues at the beginning of the next line.

You can enter on a command line any instruction or expression that you could execute on the home screen. In the program editor, each new command line begins with a colon. To enter more than one instruction or expression on a single command line, separate each with a colon.

To move the cursor down to the next new command line, press [ENTER]. You cannot move to the next new command line by pressing [▼]. However, you can return to existing command lines to edit them by pressing [▲].

Menus and Screens in the Program Editor

All CATALOG *items are valid in the program editor.*

TI-86 menus and screens may be altered when displayed in the program editor. Menu items that are invalid for a program are omitted from menus. Menus that are not valid in a program, such as the LINK menu or MEM menu, are not displayed at all.

When you select a setting from a screen such as the mode screen or graph format screen, the setting you select is pasted to the cursor location on the command line.

Variables to which you typically store values from an editor, such as the window variables, become items on program-only menus, such as the GRAPH WIND menu. When you select them, they are pasted to the cursor location on the command line.

Running a Program

❶ Paste the program name to the home screen. Either select it from the PRGM NAMES menu ([PRGM] [F1]) or enter individual characters.

To resume the program after a pause, press [ENTER].

❷ Press [ENTER]. The program begins to run.

Each result updates the last-answer variable **Ans** (Chapter 1). The TI-86 reports errors as the program runs. Commands executed during a program do not update the previous-entry storage area ENTRY (Chapter 1).

The example program below is shown as it would appear on a TI-86 screen. The program:

- ♦ Creates a table by evaluating a function, its first derivative, and its second derivative at intervals in the graphing window
- ♦ Displays the graph of the function and its derivatives in three different graph styles, activates the trace cursor, and pauses to allow you to trace the function

`PROGRAM:FUNCTABL`	The name of the program
`:Func:Fix 2:FnOff:PlOff`	Set graphing and decimal modes (mode screen); turn off functions (GRAPH VARS menu) and plots (STAT PLOT menu)
`:y1=.6 x cos x`	Define the function (assignment statement)
`:ClLCD`	Clear the home screen (PRGM I/O menu)
`:Eq▶St(y1,STRING)`	Convert **y1** into the string variable **STRING** (STRNG menu)
`:Outpt(1,1,"y1=")`	Display **y1=** at row 1, column 1 (PRGM I/O menu)
`:Outpt(1,4,STRING)`	Display value stored to **STRING** at row 1, col. 4 (PRGM I/O menu)
`:Outpt(8,1,"PRESS ENTER")`	Display **PRESS ENTER** at line 8, column 1 (PRGM I/O menu)
`:Pause`	Pause the program (PRGM CTL menu)
`:ClLCD`	Clear the home screen (PRGM I/O menu)
`:y2=der1(y1,x,x)`	Define **y2** as the first derivative of **y1** (CALC menu)
`:y3=der2(y1,x,x)`	Define **y3** as the second derivative of **y1** (CALC menu)
`:DispT`	Display the table (PRGM I/O menu)
`:GrStl(1,1):GrStl(2,2):GrStl(3,7)`	Set graph styles for **y1**, **y2**, and **y3** (PRGM CTL menu)
`:2→xRes`	Store **2** to the window variable **xRes** (GRAPH WIND menu)
`:ZTrig`	Set the viewing window variables (GRAPH ZOOM menu)
`:Trace`	Display the graph, activate trace cursor, and pause (GRAPH menu)

Breaking (Interrupting) a Program

To break (interrupt) the program, press [ON]. The ERROR 06 BREAK menu is displayed.

- To display the program editor where the interruption occurred, select **GOTO** ([F1]).
- To return to the home screen, select **QUIT** ([F5]).

Working with Programs

Managing Memory and Deleting a Program

To check whether adequate memory is available for a program you want to enter or download, display the Check RAM screen ([2nd] [MEM] [F1]; Chapter 17). To increase available memory, consider deleting selected items or data types from memory (Chapter 17).

Editing a Program

After you write a program, you can display it in the program editor and edit any command line.

The program editor does not display a ↓ to indicate that command lines continue beyond the screen.

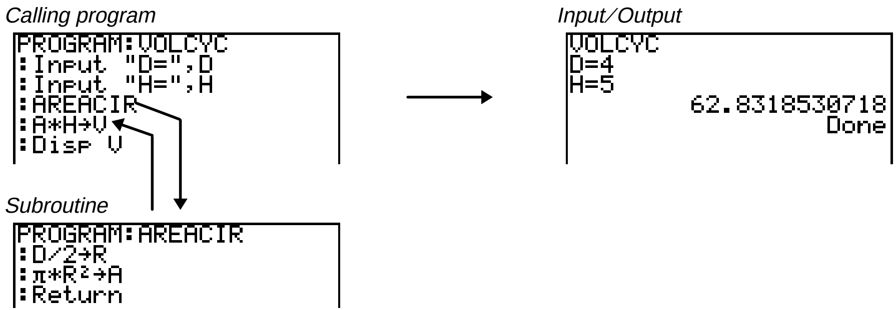
1. Display the program editor ([PRGM] [F2]). The PRGM NAMES menu also is displayed.
2. Enter the name of the program you want to edit. Either select the name from the PRGM NAMES menu or enter the individual characters.
3. Edit the program command lines.
   - Move the cursor to the appropriate location, and then delete, overwrite, or insert characters.
   - Press [CLEAR] to clear the entire command line, except for the leading colon, and then enter a new program command.
   - Select program editor menu items **INSc** ([F5]) and **DELc** ([MORE] [F1]) to insert and delete command lines.

Calling a Program from Another Program

On the TI-86, any stored program can be called from another program as a subroutine. In the program editor, enter the subroutine program name on a command line by itself.

- Press [PRGM] to display the PRGM NAMES menu, and then select the program name.
- Use ALPHA keys and alpha keys to enter the program name's individual characters.

When the program name is encountered as the calling program runs, the next command executed is the first command in the subroutine. It returns to the next command in the calling program when it encounters **Return** (or implied **Return**) at the end of a subroutine.



label used with **Goto** and **Lbl** is local to the program where it is located. *label* in one program is not recognized by another program. You cannot use **Goto** to branch to a *label* in another program.

Copying a Program to Another Program Name

1. Display a new or existing program in the program editor.
2. Move the cursor to the command line on which you want to copy a program.
3. Display the **Rcl** prompt ([2nd] [RCL]).
4. Enter the name of the program you want to copy. Either select the name from the PRGM NAMES menu or enter individual characters.
5. Press [ENTER]. The contents of the recalled program name are inserted into the other program at the cursor location.

Using and Deleting Variables within a Single Program

If you want to use variables within a program but do not need them after the program is run, you can use **DelVar(** within the program to delete the variables from memory.

The program segment to the right uses the variables A and B as counters and then deletes them from memory.

```
:3→B
:For (A,1,100,1)
:B+A→B
:End
:Disp A
:Disp B
:DelVar(A)
:DelVar(B)
```

Running an Assembly Language Program

An assembly language program is a program that runs much faster and has greater control of the calculator than the regular programs described in this chapter. You can download and run TI-created assembly language programs to add features to your TI-86 that are not built in. For example, you can download the TI-83 finance or inferential statistics features to use on your TI-86.

TI assembly language programs and other programs are available on TI's World Wide Web site: **http://www.ti.com/calc**

When you download an assembly language program, it is stored among the other programs as a PRGM NAMES menu item. You can:

- Transmit it using the TI-86 communication link (Chapter 18).
- Delete it using the MEM DELETE:PRGM screen (Chapter 17).
- Call it from another program as a subroutine (page 224).

To run an *assemblyProgramName*, the syntax is: **Asm(***assemblyProgramName***)**

If you write an assembly language program, use the two instructions below from the CATALOG.

AsmComp(*AsciiAssemblyPrgmName***,** *HexAssemblyPrgmName***)**	Compiles an assembly language program written in ASCII and stores the hex version
AsmPrgm	Identifies an assembly language program; must be entered as the first line of an assembly language program

Entering and Storing a String

A string is a sequence of characters that you enclose within quotation marks.

- A string defines characters to be displayed in a program.
- A string accepts input from the keyboard in a program.

You do not use quotation marks to enter a string name. In concatenation, you can substitute stringName for any "string".

To enter a string directly, the syntax is:

"*string***"**

To concatenate (join together) two or more strings, use [+]. The syntax is:

"*stringA***"+"***stringB***"+"***stringC***"+...**

The STRNG (String) Menu [2nd] [STRNG]

"	sub	lngth	Eq▸St	St▸Eq

" also marks the start and end of a formula to be attached to a list; it is also an item on the list editor menu (Chapter 11).

"*string***"**	Marks the start and end of *string*
sub("*string***",***begin***,***length***)** **sub(***stringName***,***begin***,***length***)**	Returns a subset of **"***string***"** or *stringName*, starting at *begin* character place and *length* characters long
lngth "*string***"** or **lngth** *stringName*	Returns the number of characters in **"***string***"** or *stringName*
Eq▸St(*equationVariable***,***stringName***)**	Converts *equationVariable* contents to *stringName*
St▸Eq(*stringName***,***equationVariable***)**	Converts *stringName* to *equationVariable*

Creating a String

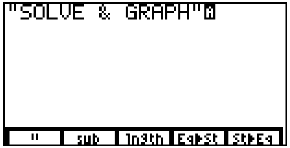
Begin these steps on a blank line on the home screen or in the program editor.

❶ Display the STRNG menu.

[2nd] [STRNG]

❷ Enter the open quotation mark, then the string **SOLVE & GRAPH**, and then the close quotation mark.

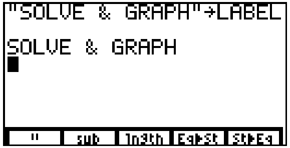
[F1] [ALPHA] [ALPHA] [S] [O] [L] [V] [E] [␣] [2nd] [CHAR] [F1] [F3] [␣] [G] [R] [A] [P] [H] [2nd] [STRNG] [F1]



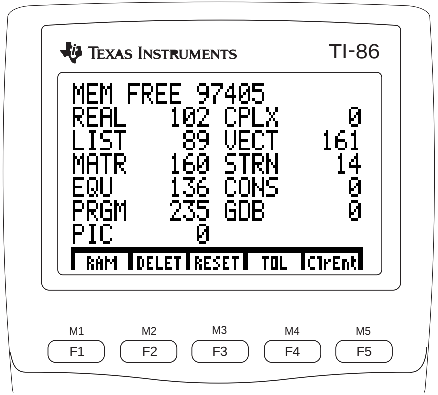
To evaluate the contents of a string, you must use **St▸Eq(** *to convert it to an equation.*

❸ Store the string to the string variable name **LABEL**.

[ALPHA] [STO▸] [L] [A] [B] [E] [L] [ENTER]



17 Memory Management



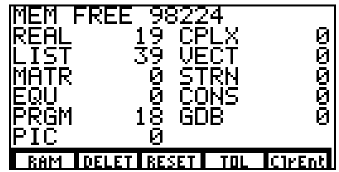
Checking Available Memory

For information on **TOL** *(the tolerance editor), refer to the Appendix.*

The MEM (Memory) Menu [2nd] [MEM]

RAM	DELET	RESET	TOL	ClrEnt
check-RAM screen	memory delete menu	memory/default reset menu	tolerance editor	clears ENTRY storage area

Checking Memory Usage [2nd] [MEM] [F1]



When all memory is cleared and all defaults are set, the standard TI-86 has **98,224** bytes of available random-access memory (RAM). As you store information to RAM, you can monitor memory allocation on the Check RAM screen.

MEM FREE reports the total number of bytes available in RAM. Conversely, all other numbers on the screen report the number of bytes that each data type currently occupies. For example, if you were to store a 50-byte matrix in memory, the **MATR** total would increase to **50** bytes, while the **MEM FREE** total would decrease by 50 to **98174** bytes.

To display the number of bytes that a specific variable occupies, display the DELETE screen for that data type (page 231). Scroll the screen, if necessary.

Deleting Items from Memory

The MEM DELET (Delete) Menu [2nd] [MEM] [F2]

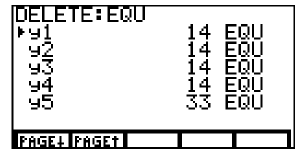
ALL	REAL	CPLX	LIST	VECTR	▶	MATRX	STRNG	EQU	CONS	PRGM
					▶	GDB	PIC			

To delete a parametric equation, delete the **xt** *component.*

Each MEM DELET menu item displays the deletion screen for that data type. For example, when you select **LIST**, the MEM DELETE:LIST screen is displayed. Use the DELETE screens to delete any user-created variable and the information stored to it.

In the example, the equation **y5=x^3−x²+4x−1** *is deleted.*

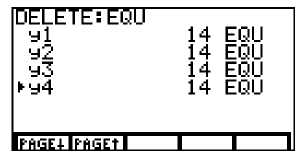
❶ Select **DELET** from the MEM menu to display the MEM DELET menu. [2nd] [MEM] [F2]



❷ Select the data type of the item you want to delete. To scroll down to the next six items or up to the previous six items, select **PAGE↓** or **PAGE↑**. [MORE] [F3]

To move directly to the first item beginning with any letter, enter that letter; ALPHA-lock is on.

❸ Move the selection cursor (▶) to the item you want to delete (**y5**). The uppercase items are in alphanumeric order, followed by the lowercase items in alphanumeric order. [▼] [▼] [▼]



❹ Delete the item. To delete other items on the screen, repeat steps 3 and 4. [ENTER]

Resetting the TI-86

The MEM RESET (Reset) Menu [2nd] [MEM] [F3]

Before resetting all memory, consider deleting selected information to increase memory capacity (page 231).

RAM	DELET	RESET	TOL	ClrEnt
ALL	MEM	DFLTS		

ALL When confirmed, all data is cleared and memory is reset; both messages are displayed

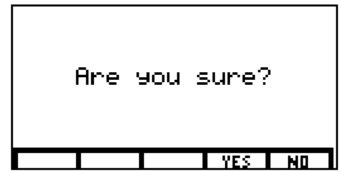
MEM When confirmed, clears all stored data from memory; **Mem Cleared** is displayed

DFLTS When confirmed, resets all defaults; **Defaults Set** is displayed

When you select and confirm **ALL** *or* **DFLTS***, the default contrast is reset; to adjust it, use [2nd] [▲] or [2nd] [▼] (Chapter 1).*

When you select **ALL**, **MEM**, or **DFLTS**, a confirmation menu is displayed.

- ♦ To confirm the selected reset, select **YES** (press [F4]).
- ♦ To cancel the selected reset, select **NO** (press [F5]).

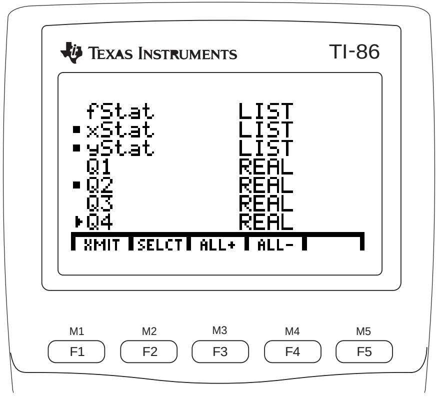


ClrEnt (Clear Entry) [2nd] [MEM] [F5]

The TI-86 retains as many previous entries as possible in ENTRY, up to a capacity of 128 bytes.

To clear the ENTRY storage area of all entries, execute **ClrEnt** on a blank line on the home screen ([2nd] [MEM] [F5] [ENTER]).

18 The TI-86 Communication Link



TI-86 Linking Options

Using the unit-to-unit cable included with the TI-86, you can transmit data between the TI-86 and several other devices.

Linking Two TI-86s

You can link two TI-86 units and select the data types to be transmitted, including programs. You can back up the entire memory of a TI-86 onto another TI-86.

Linking a TI-86 and a TI-85

You can select the data types, including programs, to transfer from a TI-85 to a TI-86. You can send most variables and programs from a TI-86 to a TI-85 using **SND85** (page 239), except lists, vectors, or matrices that exceed TI-85 capacity.

When you run a TI-85 program on a TI-86, the TI-85 **PrtScrn** program instruction is not valid. Also, the EOS implied multiplication on the TI-86 differs from the TI-85 (Appendix). For example, the TI-85 interprets **sin 2x** as **sin (2x)**; the TI-86 interprets **sin 2x** as **(sin 2)x**.

Linking a TI-86 and a CBL 2/CBL or CBR System

The Calculator-Based Laboratory™ (CBL 2™/CBL™) and Calculator-Based Ranger™ (CBR™) systems are optional TI accessories that collect data from physical occurrences, such as science experiments. The CBL 2/CBL and CBR store data to lists, which you can transmit to a TI-86 and analyze. You can transmit list names to a CBL 2/CBL or CBR from a TI-86.

Linking a TI-86 and a PC or Macintosh

TI-86 TI-GRAPH LINK™ is an optional system that links a TI-86 with an IBM®-compatible or Macintosh® computer.

Downloading Programs from the Internet

If you have TI-GRAPH LINK and internet services, you can download programs from TI's World Wide Web site at:

http://www.ti.com/calc

You can download various programs from TI's web site, including assembly language programs that add features such as TI-83 finance and inferential statistics. The site also links to many other TI-86 web sites maintained by user groups, high schools, universities, and individuals.

Connecting the TI-86 to Another Device

Before you begin to transmit data to or from the TI-86, connect it to the other device.

1. Firmly insert one end of the unit-to-unit cable into the port on the bottom edge of the calculator.
2. Firmly insert the other end of the cable into the other device (or PC adapter).

The link menus are not available in the program editor.

The LINK Menu [2nd] [LINK]

SEND	RECV	SND85		
menu of data types to send	receive mode (waiting)	menu of data types to send to a TI-85		

The CBL 2/CBL, CBR, and TI-86 TI-GRAPH LINK *have built-in Silent Link, which eliminates the need for you to set up the devices to send or receive.*

Selecting Data to Send

To list the variables for a specific data type on a selection screen, select the data type from the LINK SEND menu. When you select **BCKUP**, the message **Memory Backup** is displayed.

The LINK SEND Menu [2nd] [LINK] [F1]

BCKUP	PRGM	MATRX	GDB	ALL
memory backup	programs	matrices	graph databases	all real and complex data types

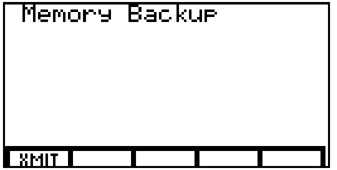
▸

LIST	VECTR	REAL	CPLX	EQU
lists	vectors	real values in all data types	complex values in all data types	equations

▸

CONS	PIC	WIND	STRNG	
user-created constants	pictures	window variable values	strings	

Initiating a Memory Backup

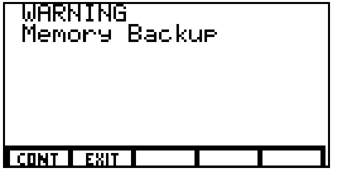


To initiate a memory backup, select **BCKUP** from the LINK SEND menu ([2nd] [LINK] [F1] [F1]). The screen to the right is displayed.

To complete memory backup, prepare the other unit to receive data transmission (page 239), and then select **XMIT** from the memory backup menu ([F1]).

Warning: When you transmit **BCKUP**, the transmitted memory overwrites all memory in the receiving unit; all information in the memory of the receiving unit is lost. To cancel initiation of a memory backup, press [EXIT].

If a transmission error occurs during a backup, the receiving-calculator memory is reset.

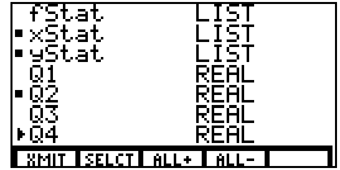


As a safety check to prevent accidental loss of memory, when the receiving calculator is notified of an incoming backup transmission, it displays the warning message and confirmation menu, as shown in the screen to the right.

- ♦ To continue the backup transmission, select **CONT**. The backup transmission continues, replacing all receiving-calculator memory with the backup data.
- ♦ To cancel backup and retain all receiving-calculator memory, select **EXIT**.

Selecting Variables to Send

If no data of the type you select is stored in memory, the message is displayed: **NO VARS OF THIS TYPE***.*



When you select any LINK SEND menu item, except **BCKUP** or **WIND**, each variable of the selected data type is listed in alphanumeric order on a selection screen. The screen to the right is the SEND ALL screen ([2nd] [LINK] [F1] [F5]).

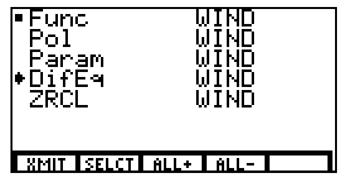
- The data type of each variable is specified.
- Small squares indicate that **xStat**, **yStat**, and **Q2** are selected to be sent.
- The selection cursor is next to **Q4**.

To select a specific variable to be sent, use [▼] and [▲] to move the selection cursor next to the variable, and then select **SELCT** ([F2]) from the selection screen menu.

- To select all variables of this type, select **ALL+** from the selection screen menu ([F3]).
- To deselect all variables of this type, select **ALL-** from the selection screen menu ([F4]).

To complete transmission of the selected variables, prepare the other unit to receive data transmission (page 239), and then select **XMIT** from the selection screen menu ([F1]).

The SEND WIND (Window Variables) Screen



When you select **WIND** from the LINK SEND menu ([2nd] [LINK] [F1] [MORE] [MORE] [F3]), the SEND WIND screen is displayed. Each SEND WIND screen item represents the window variables, format settings, and any other graph-screen data for that TI-86 graphing mode and for **ZRCL** (user-created zoom). The screen to the right shows that the graph screen data for **Func** and **DifEq** graphing modes are selected.

Func Select to send **Func** graphing mode window variable values and format settings

Pol	Select to send **Pol** graphing mode window variable values and format settings
Param	Select to send **Param** graphing mode window variable values and format settings
DifEq	Select to send **DifEq** graphing mode window variable values, **difTol**, axes settings, and format settings
ZRCL	Select to send user-created zoom window variables, and format settings in any mode

To complete transmission of the selected variables, prepare the other unit to receive data transmission (below), and then select **XMIT** from the memory backup menu ([F1]).

Sending Variables to a TI-85

The steps for selecting variables to send to a TI-85 are the same as those for selecting variables to send to a TI-86. However, the LINK SND85 menu has fewer items than the LINK SEND menu.

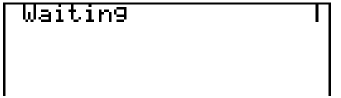
The TI-86 has more capacity for lists, vectors, and matrices than the TI-85. If you send to the TI-85 a list, vector, or matrix that has more elements than the TI-85 allows, the elements that exceed TI-85 capacity are truncated.

The LINK SND85 (Send Data to TI-85) Menu [2nd] [LINK] [F3]

MATRX	LIST	VECTR	REAL	CPLX	▶	CONS	PIC	STRNG		

To prepare a PC to receive data, consult the TI-GRAPH LINK *guidebook.*

Preparing the Receiving Device



To prepare a TI-86 or TI-85 to receive data transmission, select **RECV** from the LINK menu (2nd [LINK] F2). The message **Waiting** and the busy indicator are displayed. The calculator is ready to receive transmitted items.

To cancel receive mode without receiving items, press ON. When the **LINK TRANSMISSION ERROR** message is displayed, select **EXIT** from the menu (F1). The LINK menu is displayed.

Transmitting Data

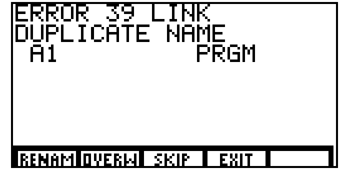
After you select data types on the sending unit and prepare the receiving unit to receive data, you can begin transmitting.

To begin transmitting, select **XMIT** on the selection screen menu of the sending calculator (F1).

To interrupt transmission, press ON on either calculator. When the **LINK TRANSMISSION ERROR** message is displayed, select **EXIT** from the menu (F1). The LINK menu is displayed.

Receiving Transmitted Data

As the TI-86 receives transmitted data, each variable name and data type is displayed line by line. If all selected items are transmitted successfully, the message **Done** is displayed. To scroll the transmitted variables, press ▼ and ▲.



During transmission, if a transmitted variable name is stored already in the memory of the receiving calculator, transmission is interrupted. The duplicated variable name, its data type, and the DUPLICATE NAME menu are displayed, as shown in the screen to the right.

To resume or cancel transmission, you must select an item from the DUPLICATE NAME menu.

RENAM	Displays the **Name=** prompt; enter a unique variable name; press [ENTER] to continue transmission
OVERW	(overwrite) Replaces data stored to the receiving unit's variable with sent variable data
SKIP	Does not overwrite the receiving unit's data; attempts to send the next selected variable
EXIT	Cancels the data transmission

Repeating Transmission to Several Devices

After transmission is complete, the LINK menu is displayed and all selections remain. You can transmit the same selections to a different TI-86 without having to re-select data.

To repeat a transmission with another device, disconnect the unit-to-unit cable from the receiving unit; connect it to another device; prepare the device to receive data; and then select **SEND**, then **ALL**, and then **XMIT**.

Error Conditions

If the cable is connected but a transmission error occurs, push the cable in more firmly to both calculators and try again.

A transmission error occurs after a few seconds if:

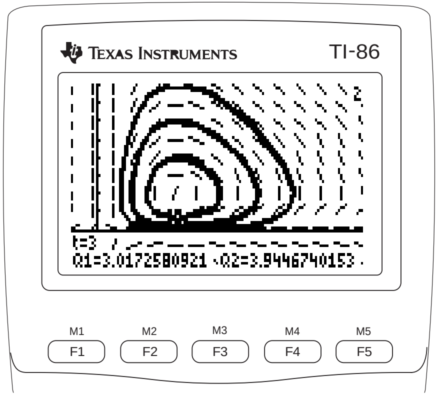
- The cable is not connected to the port of the sending calculator.
- The cable is not connected to the port of the receiving calculator.
- The receiving unit is not set to receive transmission.
- You attempt a backup between a TI-86 and a TI-85.

Insufficient Memory in Receiving Unit

If the receiving unit does not have sufficient memory to receive an item, the receiving unit displays **LINK MEMORY FULL** and the variable name and data type.

- To skip the variable, select **SKIP**. Transmission resumes with the next item.
- To cancel transmission altogether, select **EXIT**.

19 Applications

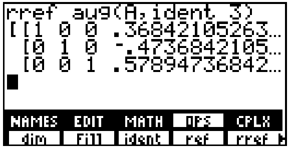


Using Math Operations with Matrices

❶ In the matrix editor, enter matrix **A** as shown.

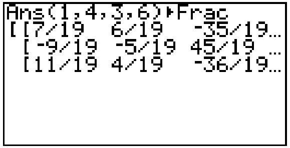
❷ On the home screen, select **rref** from the MATRX OPS menu.

❸ To append a 3×3 identity matrix to matrix **A**, select **aug** from the MATRX OPS menu, enter **A**, select **ident** from the MATRX OPS menu, and then enter **3**. Execute the expression.



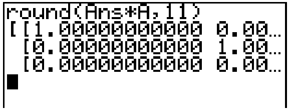
❹ Enter **Ans** (to which the matrix from step 3 is stored). Define a submatrix that contains the solution portion of the result. The submatrix begins at element (1,4) and ends at element (3,6).

❺ Select **▸Frac** from the MATH MISC menu and display the fractional equivalent of the submatrix.



Displaying the result matrix elements to 11 decimal places illustrates accuracy.

❻ Check the result. Set the decimal mode to 11 (the last **1**) Select **round** from the MATH NUM menu for the product of the fractional equivalent of the submatrix times **A**.



Finding the Area between Curves

Find the area of the region bounded by: $f(x)=300\,x/(x^2+625)$
$g(x)=3\cos(.1\,x)$
$x=75$

If necessary, select **ALL-** *from the equation editor menu to deselect all functions. Also, turn off all stat plots.*

1. In **Func** graphing mode, select **y(x)=** from the GRAPH menu to display the equation editor and enter the equations as shown.

   y1=300 x/(x²+625) **y2=3 cos (.1 x)**

2. Select **WIND** from the GRAPH menu and set the window variables as shown.

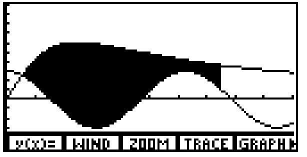
   xMin=0 **xMax=100** **xScl=10** **yMin=-5** **yMax=10** **yScl=1** **xRes=1**

3. Select **GRAPH** from the GRAPH menu to display the graph screen.

4. Select **ISECT** from the GRAPH MATH menu. Move the trace cursor to the intersection of the functions. Press [ENTER] to select **y1**. The cursor moves to **y2**. Press [ENTER]. Then press [ENTER] again to set the current cursor location as the initial guess. The solution uses the solver. The value of **x** at the intersection, which is the lower limit of the integral, is stored to **Ans** and **x**.

5. The area to integrate is between **y1** and **y2**, from **x=5.5689088189** to **x=75**. To see the area on a graph, return to the home screen, select **Shade** from the GRAPH DRAW menu, and execute this expression:

   Shade(y2,y1,Ans,75)



6. Select **TOL** from the MEM menu and set **tol=1E-5**.

7. On the home screen, compute the integral with **fnInt** (CALC menu). The area is 325.839961998.

   fnInt(y1−y2,x,Ans,75)

The Fundamental Theorem of Calculus

If necessary, select **ALL-** *from the equation editor menu to deselect all functions. Also, turn off all stat plots.*

In the example, **nDer(y2,x)** *only approximates* **y3***; you cannot define* **y3** *as* **der1(y2,x)**.

Consider these three functions:

$F(x)_1 = (\sin x)/x \qquad F(x)_2 = \int_0^x (\sin t)/t \qquad F(x)_3 = \frac{d}{dx} \int_0^x (\sin t)/t \, dt$

1. In **Func** graphing mode, select **y(x)=** from the GRAPH menu, and then enter the functions and set graph styles in the equation editor as shown. (**fnInt** and **nDer** are CALC menu items.)

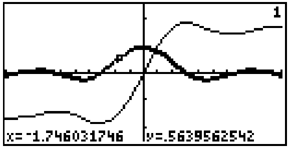
   y1=(sin x)/x **y2=fnInt(y1(t),t,0,x)** **y3=nDer(y2,x)**

2. Select **TOL** from the MEM menu to display the tolerance editor. To improve the rate of the calculations, set **tol=0.1** and δ**=0.001**.

3. Select **WIND** from the GRAPH menu and set the window variable values as shown.

   xMin=⁻10 **xMax=10** **xScl=1** **yMin=⁻2.5** **yMax=2.5** **yScl=1** **xRes=4**

4. Select **TRACE** from the GRAPH menu to display the graph and the trace cursor.



5. Trace **y1** and **y3** to verify that the graph of **y1** and the graph of **y3** are visually indistinguishable.

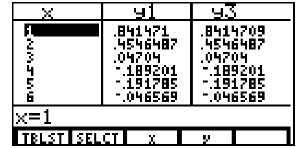
The inability to visually distinguish between the graphs of **y1** and **y3** graphically supports the fact that:

$$\frac{d}{dx} \int_0^x (\sin t)/t \, dt = (\sin x)/x$$

❻ Deselect **y2** in the equation editor.

❼ Select **TBLST** from the TABLE menu. Set **TblStart=1**, **ΔTbl=1**, and **Indpnt: Auto**.

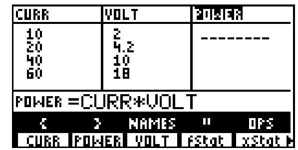
❽ Select **TABLE** from the TABLE menu to display the table. Compare the solution of **y1** with the solution of **y3** to numerically support the formula above.



Electrical Circuits

A measurement device has measured the DC current (C) in milliamperes and voltage (V) in volts on an unknown circuit. From these measurements, you can calculate power (P) in milliwatts using the equation CV=P. What is the average of the measured power?

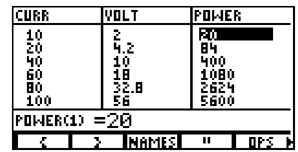
With the TI-86, you can estimate the power in milliwatts at a current of 125 milliamperes using the trace cursor, the interpolate/extrapolate editor, and a regression forecast.



❶ In two consecutive columns of the list editor, store the current measurements shown below to the list name **CURR** and the voltage measurements shown below to the list name **VOLT**.

{10,20,40,60,80,100,120,140,160}→CURR

{2,4.2,10,18,32.8,56,73.2,98,136}→VOLT

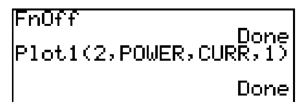


❷ In the next column of the list editor, enter the list name **POWER** .

❸ Enter the formula **CURR ∗VOLT** in the list editor entry line for **POWER**. Press [ENTER] to calculate the values for power and store the answers to the list name **POWER**.

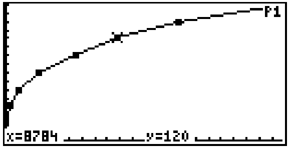
❹ Select **WIND** from the GRAPH menu and set the window variable values as shown.

xMin=0 **xMax=max(POWER)** **xScl=1000** **yMin= 0** **yMax=max(CURR)** **yScl=10** **xRes=4**



❺ From the home screen, select **FnOff** from the CATALOG and press [ENTER] to deselect all functions in the equation editor. Select **Plot1(** from the CATALOG and set up a stat plot with **POWER** on the x-axis and **CURR** on the y-axis.

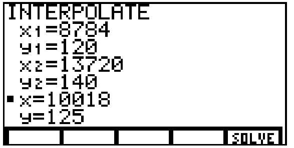
❻ Select **TRACE** from the GRAPH menu to display the stat plot and trace cursor on the graph screen.



❼ Trace the stat plot to approximate the value of **POWER** at **CURR=125**. With this statistical data, the closest to **CURR=125** that you can trace to is **CURR=120** (on the y-axis).

*The **7**s and **8**s in parentheses specify the 7th and 8th elements of **POWER** and **CURR**.*

❽ Select **INTER** from the MATH menu to display the interpolate/extrapolate editor. To interpolate **POWER** at **CURR=125**, enter the nearest pairs:



x1=POWER(7)	**y1=CURR(7)**
x2=POWER(8)	**y2=CURR(8)**

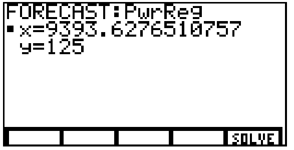
❾ Enter **y=125** and solve for **x**.

❿ On the home screen, select **LinR** from the STAT CALC menu to fit the linear regression model equation to the data stored to **POWER** and **CURR**. Write down the value of the result variable **corr**.

*To enter each regression after **LinR**, press* [2nd] [ENTRY] *and edit as needed.*

⓫ Fit the logarithmic (**LnR**), exponential (**ExpR**), and power (**PwrR**) regressions to the data, writing down the value of **corr** for each regression. Compare the **corr** values of each regression to determine which model fits the data most accurately (the **corr** value closest to **1**).

⓬ Execute the most accurate regression again, and then select **FCST** from the STAT menu. To forecast **POWER** at **CURR=125**, enter **y=125** and solve for **x**.



Compare this answer with the answer returned in step 9.

Program: Taylor Series

When you run this program, you can enter a function and specify the order and center point. Then the program calculates the Taylor Series approximation for the function and plots the function you entered. This example shows how to call a program from another program as a subroutine.

1. Before you enter the program **TAYLOR**, select **EDIT** from the PRGM menu, enter **MOBIUS** at the **Name=** prompt, and then enter this brief program to store the Mobius Series. The program **TAYLOR** calls this program and runs it as a subroutine.

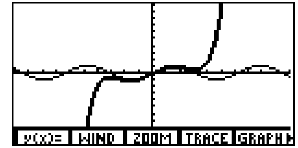
```
PROGRAM:MOBIUS
:{1,-1,-1,0,-1,1,-1,0,0,1,-1,0,-1,1,1,0,-1,0,-1,0}→MSERIES
:Return
```

2. Select **EDIT** from the PRGM menu, enter **TAYLOR** at the **Name=** prompt, and then enter this program to calculate the Taylor Series.

The higher-order derivative values necessary for this program are calculated numerically based on the methods in "Numerical Differentiation of Analytic Functions," J. N. Lyness and C. B. Moler, SIAM Journal of Numerical Analysis 4 (1967): 202-210.

```
PROGRAM:TAYLOR
:Func:FnOff
:y14=pEval(TPOLY,x-center)
:GrStl(14,2)
:1E-9→ε:.1→rr          ε is on the CHAR GREEK menu
:ClLCD
:InpSt "FUNCTION: ",EQ   User enters equation function
:St▶Eq(EQ,y13)
:Input "ORDER: ",order   User enters order
:order+1→dimL TPOLY
:Fill(0,TPOLY)
:Input "CENTER: ",center  User enters center
:evalF(y13,x,center)→f0
:f0→TPOLY(order+1)
```

	:If order≥1
	:der1(y13,x,center)→TPOLY(order)
	:If order≥2
	:der2(y13,x,center)/2→TPOLY(order-1)
	:If order≥3
*Begins **Then** group*	:Then
Calls subroutine	:MOBIUS
*Begins **For** group*	:For(N,3,order,1)
	:abs f0→gmax:gmax→bmi
	:1→m:0→ssum
*Begins **While** group*	:While abs bmi≥ε*gmax
	:While MSERIES(m)==0
*Creates nested **While** group*	:m+1→m
	:End
	:0→bsum
	:For(J,1,m*N,1)
	:rr*e^(2π(J/(m*N))*(0,1))+(center,0)→x
*Creates nested **For** group*	:real y13→gval
	:bsum+gval→bsum
	:max(abs gval,gmax)→gmax
	:End
	:bsum/(m*N)-f0→bmi
	:ssum+MSERIES(m)*bmi→ssum
	:m+1→m
*Ends **While** group*	:End
	:ssum/(rr^N)→TPOLY(order+1-N)
*Ends **For** group*	:End
*Ends **Then** group*	:End
	:ZStd



❸ On the home screen, select **TAYLOR** from the PRGM NAMES menu, and then press [ENTER] to run the program.

❹ When prompted, enter: **FUNCTION: sin x**
ORDER: 5
CENTER: 0

Characteristic Polynomial and Eigenvalues

❶ In the matrix editor or on the home screen, enter matrix **A** as shown.

```
[[-1,2,5][3,-6,9][2,-5,7]]→A
```

The first eigenvalue is real, since the imaginary part is **0**.

❷ On the home screen, select **eigVl** from the MATRX MATH menu to find the complex eigenvalues for the matrix **A** and store them to the list name **EV**.

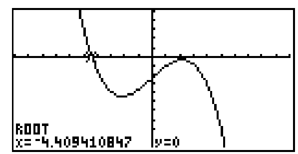
If necessary, select **ALL-** *from the equation editor menu to deselect all functions. Also, turn off all stat plots.*

❸ Graph the characteristic polynomial Cp(x) of matrix **A** without knowing the analytic form of Cp(x) based on the formula Cp(x)=det(A−x∗I). In **Func** graphing mode, select **y(x)=** from the GRAPH menu and enter the function in the equation editor as shown.

\y1=det (A−x∗ident 3)

❹ Select **WIND** from the GRAPH menu and set the window variable values as shown.

xMin=⁻10 xMax=10 xScl=1 yMin=⁻100 yMax=50 yScl=10 xRes=4

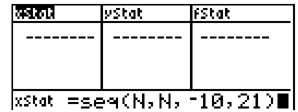


❺ Select **ROOT** from the GRAPH MATH menu and use it to display the real eigenvalue interactively. Use **Left Bound=⁻5**, **Right Bound=⁻4**, and **Guess=⁻4.5**.

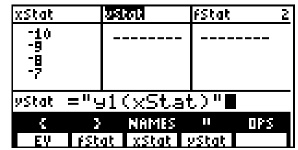
Compare the root (**x** value) you displayed interactively with the first element of the result list in step 2.

Next, use the list editor and a degree-three polynomial regression to find an analytic formula in terms of **x** for the characteristic polynomial **y1=det(A−x*ident 3)**. Create two lists that you can use to find the analytic formula.

❻ In the list editor, create elements for **xStat** by entering the expression **seq(N,N,⁻10,21)** in the **xStat** entry line. **seq** is on the MATH MISC menu.



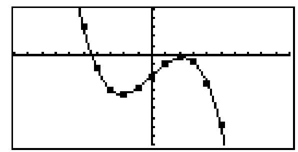
❼ Create elements for **yStat** by attaching the formula **"y1(xStat)"** to **yStat** in the entry line. The expression is evaluated when you press [ENTER] or exit the list editor.



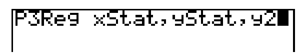
❽ On the home screen, select **Plot1(** from the CATALOG and execute **Plot1(2,xStat,yStat,1)** to turn on **Plot1** as an xyLine plot using the lists **xStat** and **yStat**.

To clear the menus from the graph screen, press [CLEAR].

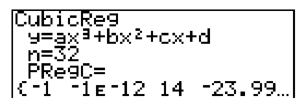
❾ Select **GRAPH** from the GRAPH menu to display **Plot1** and **y1** on the graph screen.



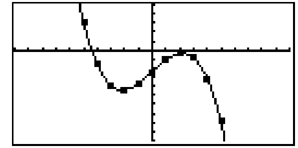
❿ On the home screen, select **P3Reg** from the STAT CALC menu. Execute **P3Reg xStat,yStat,y2** to find the explicit characteristic polynomial in terms of **x** and store it to **y2**.



The cubic regression coefficients stored in the result list **PRegC** suggest that a=⁻1, b=0, c=14, and d=⁻24. So the characteristic polynomial seems to be Cp(x)=⁻x^3+14x−24.



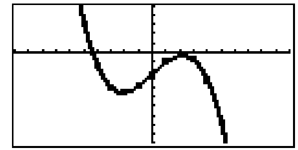
11. Support this conjecture by graphing **y1**, **y2** (to which Cp(x) is stored), and **Plot1** together.



12. In the equation editor, enter the apparent characteristic polynomial of matrix **A** and select (thick) graph style as shown.

    y3=-x^3+14x−24

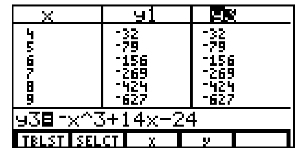
13. Graph **y1**, **y2**, **y3**, and **Plot1**.



14. Deselect **y2** in the equation editor.

15. Select **TABLE** from the TABLE menu to display **y1** and **y3** in the table.

    Compare the values for the characteristic polynomial.



Convergence of the Power Series

A closed-form analytic antiderivative of (sin x)/x does not exist. However, substituting t for x, you can find an infinite series analytic solution by taking the series definition of sin t, dividing each term of the series by t, and then integrating term by term to yield:

$$\sum_{n=1}^{\infty} -1^{n+1} t^{2n-1}/((2n-1)(2n-1)!)$$

Plot finite approximations of this power series solution on the TI-86 with **sum** and **seq**.

1. Select **TOL** from the MEM menu and set **tol=1**.

2. On the mode screen, set **Radian** angle mode and **Param** graphing mode.

If necessary, select **ALL−** *from the equation editor menu to deselect all functions. Also, turn off all stat plots.*

3. In the equation editor, enter the parametric equations for the power series approximation as shown. Select **sum** and **seq** from the LIST OPS menu. Select **!** from the MATH PROB menu.

   xt1=t **yt1=sum seq((-1)^(j+1)t^(2j−1)/((2j−1)(2j−1)!),j,1,10,1)**

4. In the equation editor, enter the parametric equations as shown to plot the antiderivative of (sin x)/x and compare it with the plot of the power series approximation. (Select **fnInt** from the CALC menu.)

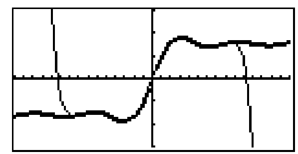
   xt2=t **yt2=fnInt((sin w)/w,w,0,t)**

This example is set up in **Param** *mode, which allows you to control the solution with* **tStep** *and increase plotting speed.*

5. Select **WIND** from the GRAPH menu and set the window variable values as shown.

   tMin=-15 **xMin=-15** **yMin=-3**
   tMax=15 **xMax=15** **yMax=3**
   tStep=0.5 **xScl=1** **yScl=1**

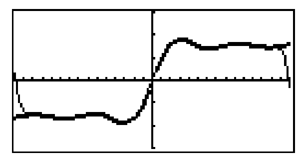
6. Select **FORMT** from the GRAPH menu and set **SimulG** format.



7. Select **GRAPH** from the GRAPH menu to plot the parametric equations on the graph screen.

To clear the menus from the graph screen, press [CLEAR].

8. In the equation editor, modify **yt1** to compute the first 16 terms of the power series by changing **10** to **16**. Plot the equations again.



   In this example, the window variable **tStep** controls the plotting speed. Select **WIND** from the GRAPH menu and set **tStep=1** and observe the difference in plotting speed and curve smoothness.

Reservoir Problem

On the TI-86, you can use parametric graphing animation to solve a problem.

Consider a water reservoir with a height of 2 meters. You must install a small valve on the side of the reservoir such that water spraying from the open valve hits the ground as far away from the reservoir as possible. At what height should you install the valve to maximize the length of the water stream when the valve is wide open?

Assume a full tank at time=0, no acceleration in the x direction, and no initial velocity in the y direction. Also, ignore valve-size and valve-type factors. Integrating the definition of acceleration in both the x and y directions twice yields the equations $x=v_0t$ and $y=h_0-(gt^2)/2$. Solving Bernoulli's equation for v_0 and substituting into v_0t results in this pair of parametric equations:

$xt=t\sqrt{(2g(2-h_0))}$ $yt=h_0-(gt^2)/2$

t = time in seconds
h_0 = height of the valve in meters
g = the built-in acceleration of gravity constant

When you graph these equations on the TI-86, the y-axis (x=0) is the side of the reservoir where the valve is to be installed. The x-axis (y=0) is the ground. Each plotted parametric equation represents the water stream when the valve is at each of several heights.

If necessary, select **ALL–** *from the equation editor menu to deselect all functions. Also, turn off all stat plots.*

❶ In **Param** graphing mode, select **E(t)=** from the GRAPH menu and enter the equations in the equation editor as shown. This pair of equations plots the path of the water stream when the valve is installed at a height of 0.5 meters.

xt1=t√(2g(2−0.5)) **yt1=0.5−(g∗t²)/2**

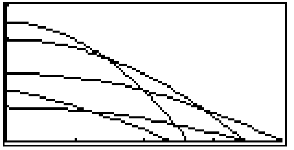
❷ Move the cursor to **xt2=.** Press [2nd] [RCL] [F2] [ALPHA] **1**, and press [ENTER] to recall the contents of **xt1** into **xt2**. For **xt2**, change the valve height (which is **0.5**) to **0.75** meters. Do the same with **yt1** and **yt2**.

❸ Repeat step 3 to create three more pairs of equations. Change the valve height to **1.0** meters for **xt3** and **yt3**, **1.5** meters for **xt4** and **yt4**, and **1.75** meters for **xt5** and **yt5**.

❹ Select **WIND** from the GRAPH menu and set the window variable values as shown.

tMin=0	**xMin=0**	**yMin=0**
tMax=√(4/g)	**xMax=2**	**yMax=2**
tStep=0.01	**xScl=0.5**	**yScl=0.5**

❺ Select **FORMT** from the GRAPH menu and set **SimulG** graph format.



To clear the menus from the graph screen, press [CLEAR].

❻ Select **GRAPH** from the GRAPH menu to plot the trajectory of the water jets from the five specified heights.

Which height seems to create the longest water stream?

Predator-Prey Model

The growth rates of predator and prey populations, such as foxes and rabbits, depend upon the populations of both species. This initial-value problem is a form of the predator-prey model.

F'=⁻F+0.1F∗R R'=3R−F∗R

Q1 = population of foxes (F)
Q2 = population of rabbits (R)
QI1= initial population of foxes (2)
QI2 = initial population of rabbits (5)

Find the population of foxes and rabbits after 3 months (**t=3**).

❶ In **DifEq** graphing mode, select **Q't=** from the GRAPH menu and enter the functions and set graph styles in the equation editor as shown.

Q'1=⁻Q1+0.1Q1∗Q2 **Q'2=3Q2−Q1∗Q2**

❷ Select **FORMT** from the GRAPH menu and set **FldOff** field format.

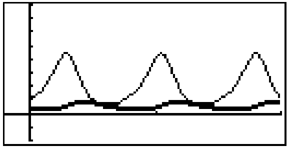
❸ Select **WIND** from the GRAPH menu and set the window variable values as shown.

tMin=0	**xMin=⁻1**	**yMin=⁻10**
tMax=10	**xMax=10**	**yMax=40**
tStep=π/24	**xScl=5**	**yScl=5**
tPlot=0		**difTol=.001**

❹ Select **INITC** from the GRAPH menu and set the initial conditions as shown.

tMin=0 **QI1=2** **QI2=5**

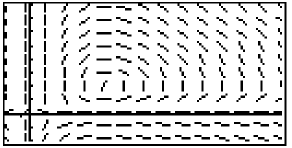
❺ Select **GRAPH** from the GRAPH menu to plot the graph of the two populations over time.



❻ To see the direction field of the phase-plane solution, select **FORMT** from the GRAPH screen, and then set **DirFld** field format.

❼ Select **INITC** from the GRAPH menu and delete the values for **QI1** and **QI2**.

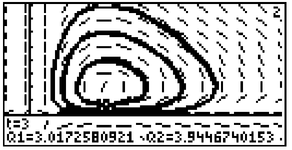
❽ Select **GRAPH** from the GRAPH menu to display the direction field of the phase-plane solution.



❾ To see a family of specific phase-plane solutions on top of the direction field, select **INITC** from the GRAPH menu, and then enter lists for **QI1** and **QI2** as shown.

QI1={2,6,7} **QI2={6,12,18}**

❿ Select **TRACE** from the GRAPH menu to display the graph with the trace cursor.



⓫ Press **3** to see how many foxes and how many rabbits are alive at **t=3**. (Round the values of **Q1** (foxes) and **Q2** (rabbits) to whole numbers.) How many foxes and rabbits are alive at **t=6**? at **t=12**?

On what value of **Q1** and **Q2** do the phase-plane orbits seem to converge? What is the significance of this value?

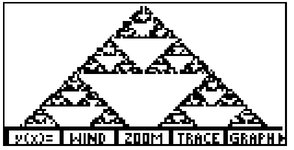
Program: Sierpinski Triangle

This program creates a drawing of a widely known fractal, the Sierpinski Triangle, and stores the drawing to the picture variable **TRI**.

1. Select **EDIT** from the PRGM menu, enter **SIERP** at the **Name=** prompt, and then enter this program.

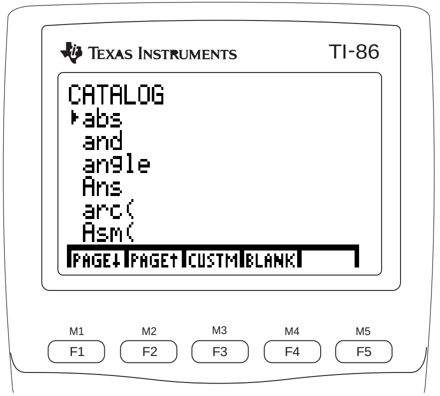
	PROGRAM:SIERP
	:FnOff :ClDrw
	:PlOff
	:AxesOff
Sets viewing window	:0→xMin:1→xMax
	:0→yMin:1→yMax
	:rand→X:rand→Y
*Begins **For** group*	:For(K,1,3000)
	:rand→N
If/Then group	:If N≤(1/3)
	:Then
	:.5X→X
	:.5Y→Y
	:End
If/Then group	:If N>(1/3) and N≤(2/3)
	:Then
	:.5(.5+X)→X
	:.5(1+Y)→Y
	:End
If/Then group	:If N>(2/3)
	:Then
	:.5(1+X)→X
	:.5Y→Y
	:End
Draws point	:PtOn(X,Y)
*End of **For***	:End
Stores picture	:StPic TRI

2. On the home screen, select **SIERP** from the PRGM NAMES menu and press [ENTER] to run the program, which may run for several minutes before completion.



3. After you run the program, you can recall and display the picture by executing **RcPic TRI**.

20 A to Z Function and Instruction Reference



Quick-Find Locator

This section lists the TI-86 functions and instructions in functional groups along with the page numbers where they are described in this chapter.

Graphing

Lists

Mathematics, Algebra, and Calculus

Matrices

Programming

Statistics

Strings

Vectors

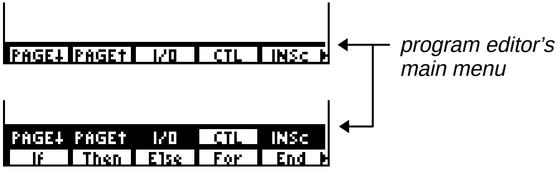
Alphabetical Listing of Operations

All the operations in this section are included in the CATALOG. Non-alphabetic operations (such as **+**, **!**, and **>**) are listed at the end of the CATALOG. In this A to Z Reference, however, these operations are listed under their alphabetic equivalent (such as addition, factorial, and greater than).

You always can use the CATALOG to select an operation and paste it to the home screen or to a command line in the program editor. You also can use the specific keystrokes, menus, or screens listed in this section.

† Indicates menus or screens that paste the operation's name only if you are in the program editor. In most cases, you can use these menus or screens from the home screen to perform the operation interactively, without pasting the name.

‡ Indicates menus or screens that are valid only from the program editor's main menu. From the home screen, you cannot use these menus or screens to select an operation.



The syntax for some operations uses brackets [ ] to indicate optional arguments. If you use an optional argument, do not enter the brackets.

abs

MATH NUM menu
CPLX menu
MATRX CPLX menu
VECTR CPLX menu

abs *realNumber* or **abs (***realExpression***)**

Returns the absolute value of *realNumber* or *realExpression*.

abs (*complexNumber***)**

Returns the magnitude (modulus) of *complexNumber*.

abs (*real*,*imaginary***)** returns $\sqrt{(real^2+imaginary^2)}$.
abs (*magnitude*∠*angle***)** returns *magnitude*.

abs *list*
abs *matrix*
abs *vector*

Returns a list, matrix, or vector in which each element is the absolute value of the corresponding real or complex element in the argument.

```
abs -256.4 [ENTER]                256.4

abs -4*3+13 [ENTER]                  25
abs (-4*3+13) [ENTER]                 1

abs (3,4) [ENTER]                     5
abs (3∠4) [ENTER]                     3

abs {1.25,-5.67} [ENTER]
                            {1.25 5.67}

abs [(3,4),(3∠4)] [ENTER]
                                  [5 3]
```

Addition: +

[+]

numberA **+** *numberB*

Returns the sum of two real or complex numbers.

number **+** *list*

Returns a list in which a real or complex *number* is added to each element of a real or complex *list*.

In **RectC** complex number mode:

```
(2,5)+(5,9) [ENTER]              (7,14)

4+{1,2,3} [ENTER]               {5 6 7}

3+{1,7,(2,1)} [ENTER]
                 {(4,0) (10,0) (5,1)}
```

listA **+** *listB*
matrixA **+** *matrixB*
vectorA **+** *vectorB*

Returns a list, matrix, or vector that is the sum of the corresponding real or complex elements in the arguments. The two arguments must have the same dimension.

For information about adding two strings, refer to **Concatenation** on page 274.

```
{1,2,3}+{4,5,6} [ENTER]        {5 7 9}

[[1,2,3][4,5,6]]+[[4,5,6][7,8,9]]
[ENTER]                    [[5  7  9 ]
                            [11 13 15]]

[1,2,3]+[4,5,6] [ENTER]        [5 7 9]
```

and

BASE BOOL menu

integerA **and** *integerB*

Compares two real integers bit by bit. Internally, both integers are converted to binary. When corresponding bits are compared, the result is 1 if both bits are 1; otherwise, the result is 0. The returned value is the sum of the bit results.

For example, 78 **and** 23 = 6.

78 = 1001110**b**
23 = 0010111**b**
0000110**b** = 6

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.

In **Dec** number base mode:

```
78 and 23 [ENTER]                    6
```

In **Bin** number base mode:

```
1001110 and 10111 [ENTER]
                                  110b
Ans▶Dec [ENTER]                     6d
```

angle

CPLX menu
MATRX CPLX menu
VECTR CPLX menu

angle (*complexNumber***)**

Returns the polar angle of *complexNumber*, adjusted by $+\pi$ in the 2nd quadrant or $^-\pi$ in the 3rd quadrant. The polar angle of a real number is always 0.

angle (*real*,*imaginary***)** returns **tan**$^{-1}$**(***imaginary*/*real***)**.
angle (*magnitude*∠*angle***)** returns *angle*, $^-\pi <$ *angle* $\leq \pi$.

angle *complexList*
angle *complexMatrix*
angle *complexVector*

Returns a list, matrix, or vector in which each element is the polar angle of the corresponding element in the argument.

If *complexVector* has only two real elements, the returned value is a real number, not a vector.

In **Radian** angle mode and **PolarC** complex number mode:

```
angle (3,4) [ENTER]            .927295218002
angle (3∠2) [ENTER]                        2
(6∠π/3)→A [ENTER]       (6∠1.0471975512)
angle A [ENTER]                 1.0471975512
angle {(3,4),(3∠2)} [ENTER]
                          {.927295218002 2}
```

Ans

[2nd] [ANS]

Ans

Returns the last answer.

```
1.7*4.2 [ENTER]                         7.14
147/Ans [ENTER]                20.5882352941
```

arc(

CALC menu

arc (*expression*,*variable*,*start*,*end***)**

Returns the length along *expression* with respect to *variable*, from *variable* = *start* to *variable* = *end*.

```
arc(x²,x,0,1) [ENTER]
                               1.47894285752
arc(cos x,x,0,π) [ENTER]
                               3.82019778904
```

Asm(

CATALOG

Asm(*assemblyProgramName***)**

Executes an assembly language program. For more information, refer to Chapter 16.

AsmComp(

CATALOG

AsmComp(*AsciiAssemblyPrgmName*,*HexAssemblyPrgmName***)**

Compiles an assembly language program written in ASCII and stores the hex version. The compiled hex version, which uses about half the storage space of the ASCII version, cannot be edited.

When you execute the ASCII version, the TI-86 compiles it each time. To speed up execution, use **AsmComp(** to compile the ASCII version once and then execute the hex version each time you want to run the program.

AsmPrgm

CATALOG

AsmPrgm

Must be used as the first line of an assembly language program.

Assignment: =

[ALPHA] [=]

equationVariable **=** *expression*

Stores *expression* to *equationVariable*, without evaluating *expression*. (If you use [STO▶] to store an expression to a variable, the expression is evaluated and then the result is stored.)

```
y1=2 x²+6 x−5 [ENTER]              Done
```

The built-in equation variables used for graphing are case-sensitive. Use **y1**, not **Y1**.

aug(

LIST OPS menu

MATRX OPS menu

aug(*listA*,*listB***)**

Returns a list consisting of *listB* appended (concatenated) to the end of *listA*. The lists can be real or complex.

```
aug({1,-3,2},{5,4}) [ENTER]
                        {1 -3 2 5 4}
```

aug(*matrixA***,***matrixB***)**

Returns a matrix consisting of *matrixB* appended as new columns to the end of *matrixA*. The matrices can be real or complex. Both must have the same number of rows.

```
[[1,2,3][4,5,6]]→MATA [ENTER]
                          [[1 2 3]
                           [4 5 6]]
[[7,8][9,10]]→MATB [ENTER]
                           [[7 8 ]
                            [9 10]]
aug(MATA,MATB) [ENTER]
                      [[1 2 3 7 8 ]
                       [4 5 6 9 10]]
```

aug(*matrix***,***vector***)**

Returns a matrix consisting of *vector* appended as a new column to the end of *matrix*. The arguments can be real or complex. The number of rows in *matrix* must equal the number of elements in *vector*.

Axes(

† GRAPH VARS menu

Axes(*xAxisVariable***,***yAxisVariable***)**

Specifies the variables plotted for the axes in **DifEq** graphing mode. The *xAxisVariable* or *yAxisVariable* can be **t**, **Q1** through **Q9**, or **Q′1** through **Q′9**.

```
Axes(Q1,Q2) [ENTER]                  Done
```

AxesOff

† graph format screen

AxesOff

Turns off the graph axes.

AxesOn

† graph format screen

AxesOn

Turns on the graph axes.

---

b

BASE TYPE menu

*integer***b**

Designates a real *integer* as binary, regardless of the number base mode setting.

In **Dec** number base mode:

```
10b [ENTER]                             2
10b+10 [ENTER]                         12
```

Bin

† mode screen

Bin

Sets binary number base mode. Results are displayed with the b suffix. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the b, d, h, or o designator, respectively, from the BASE TYPE menu.

In **Bin** number base mode:

```
10+Fh+10o+10d [ENTER]          100011b
```

▸Bin

BASE CONV menu

number▸**Bin**
list▸**Bin**
matrix▸**Bin**
vector▸**Bin**

Returns the binary equivalent of the real or complex argument.

In **Dec** number base mode:

```
2*8 [ENTER]                    16
Ans▸Bin [ENTER]                10000b

{1,2,3,4}▸Bin [ENTER]
                 {1b 10b 11b 100b}
```

Box

† STAT DRAW menu

Box *xList*,*frequencyList*

Draws a box plot on the current graph, using the real data in *xList* and the frequencies in *frequencyList*.

Box *xList*

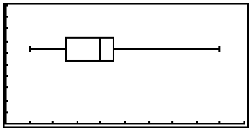
Uses frequencies of 1.

Box

Uses the data in built-in variables **xStat** and **fStat**. These variables must contain valid data of the same dimension; otherwise, an error occurs.

Starting with a **ZStd** graph screen:

```
{1,2,3,4,5,9}→XL [ENTER]
                    {1 2 3 4 5 9}
{1,1,1,4,1,1}→FL [ENTER]
                    {1 1 1 4 1 1}
0→xMin:0→yMin [ENTER]           0
Box XL,FL [ENTER]
```



Circl(

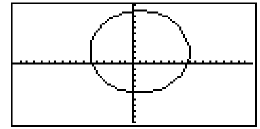
† GRAPH DRAW menu

Circl(x,y,*radius***)**

Draws a circle with center (x,y) and *radius* on the current graph.

Starting with a **ZStd** graph screen:

```
ZSqr:Circl(1,2,7) [ENTER]
```



ClDrw

† GRAPH DRAW menu

† STAT DRAW menu

ClDrw

Clears all drawn elements from the current graph.

ClLCD

‡ program editor I/O menu

ClLCD

Clears the home screen (LCD).

ClrEnt

MEM menu

ClrEnt

Clears the contents of the Last Entry storage area.

ClTbl

‡ program editor I/O menu

ClTbl

Clears all values from the current table if **Indpnt: Ask** (**IAsk**, page 304) is set.

cnorm

MATRX MATH menu

cnorm *matrix*

Returns the column norm of a real or complex *matrix*. For each column, **cnorm** sums the absolute values (magnitudes of complex elements) of the elements in that column and returns the largest of those column sums.

```
[[1,-2,3][4,5,-6]]→MAT [ENTER]
                         [[1 -2 3 ]
                          [4 5  -6]]
cnorm MAT [ENTER]                 9
```

cnorm *vector*

Returns the sum of the absolute values of the real or complex elements in *vector*.

```
[-1,2,-3]→VEC [ENTER]        [-1 2 -3]
cnorm VEC [ENTER]                    6
```

Concatenation: +

[+]

stringA **+** *stringB*

Returns a string consisting of *stringB* appended (concatenated) to the end of *stringA*.

```
"your name:"→STR [ENTER]
                    your name:
"Enter "+STR [ENTER]
                    Enter your name:
```

---

cond

MATRX MATH menu

cond *squareMatrix*

Returns the condition number of a real or complex *squareMatrix*, which is calculated as:

cnorm *squareMatrix* ***** **cnorm** *squareMatrix*$^{-1}$

The condition number indicates how well-behaved *squareMatrix* is expected to be for certain matrix functions, particularly inverse. For a well-behaved matrix, the condition number is close to 1.

log(cond *squareMatrix***)** indicates the number of digits that may be lost due to round-off errors in computing the inverse.

For a matrix with no inverse, **cond** returns an error.

```
[[1,0,0][0,1,0][0,0,1]]→MAT1
[ENTER]                       [[1 0 0]
                               [0 1 0]
                               [0 0 1]]

cond MAT1 [ENTER]                    1
log (Ans) [ENTER]                    0

[[1,2,3][4,5,6][7,8,9]]→MAT2
[ENTER]                       [[1 2 3]
                               [4 5 6]
                               [7 8 9]]

cond MAT2 [ENTER]               1.8E14
log (Ans) [ENTER]         14.2552725051
```

conj

CPLX menu

MATRX CPLX menu

VECTR CPLX menu

conj (_complexNumber_**)**

Returns the complex conjugate of _complexNumber_.

In **RectC** mode, **conj (**_real_,_imaginary_**)** returns **(**_real_,⁻_imaginary_**)**.

In **PolarC** mode, **conj (**_magnitude_∠_angle_**)** returns **(**_magnitude_∠⁻_angle_**)**, $^-\pi < angle \le \pi$.

conj _complexList_
conj _complexMatrix_
conj _complexVector_

Returns a complex list, matrix, or vector in which each element is the complex conjugate of the original.

In **RectC** complex number mode:

```
conj (3,4) [ENTER]                (3,-4)
conj (3∠2) [ENTER]
                 (⁻1.24844050964,⁻2.7…
```

In **PolarC** complex number mode:

```
conj (3∠2) [ENTER]
                               (3∠⁻2)
conj (3,4) [ENTER]
                   (5∠⁻.927295218002)
conj {√⁻2,(3,4)} [ENTER]
                 {(1.41421356237∠⁻1.5…
```

CoordOff

† graph format screen

CoordOff

Turns off cursor coordinates so they are not displayed at the bottom of a graph.

CoordOn

† graph format screen

CoordOn

Displays cursor coordinates at the bottom of a graph.

cos

[COS]

cos *angle* or **cos (***expression***)**

Returns the cosine of *angle* or *expression*, which can be real or complex.

An angle is interpreted as degrees or radians according to the current angle mode. In any angle mode, you can designate an angle as degrees or radians by using the ° or ^{r} designator, respectively, from the MATH ANGLE menu.

In **Radian** angle mode:

```
cos π/2 [ENTER]              ⁻.5
cos (π/2) [ENTER]              0
cos 45° [ENTER]     .707106781187
```

In **Degree** angle mode:

```
cos 45 [ENTER]      .707106781187
cos (π/2)ʳ [ENTER]             0
```

cos *list*

Returns a list in which each element is the cosine of the corresponding element in *list*.

In **Radian** angle mode:

```
cos {0,π/2,π} [ENTER]    {1 0 ⁻1}
```

In **Degree** angle mode:

```
cos {0,60,90} [ENTER]    {1 .5 0}
```

cos *squareMatrix*

The squareMatrix cannot have repeated eigenvalues.

Returns a square matrix that is the matrix cosine of *squareMatrix*. The matrix cosine corresponds to the result calculated using power series or Cayley-Hamilton Theorem techniques. This is *not* the same as simply calculating the cosine of each element.

cos⁻¹

[2nd] [COS⁻¹]

cos⁻¹ *number* or **cos⁻¹ (***expression***)**

Returns the arccosine of *number* or *expression*, which can be real or complex.

In **Radian** angle mode:

```
cos⁻¹ .5 [ENTER]     1.0471975512
```

In **Degree** angle mode:

```
cos⁻¹ 1 [ENTER]                0
```

cos⁻¹ *list*

Returns a list in which each element is the arccosine of the corresponding element in *list*.

In **Radian** angle mode:

```
cos⁻¹ {0,.5} [ENTER]
          {1.57079632679,1.047…
```

cosh

MATH HYP menu

cosh *number* or **cosh (***expression***)**

Returns the hyperbolic cosine of *number* or *expression*, which can be real or complex.

```
cosh 1.2 [ENTER]        1.81065556732
```

cosh *list*

Returns a list in which each element is the hyperbolic cosine of the corresponding element in *list*.

```
cosh {0,1.2} [ENTER]
                   {1 1.81065556732}
```

cosh^{-1}

MATH HYP menu

cosh^{-1} *number* or **cos^{-1} (***expression***)**

Returns the inverse hyperbolic cosine of *number* or *expression*, which can be real or complex.

```
cosh⁻¹ 1 [ENTER]                     0
```

cosh^{-1} *list*

Returns a list in which each element is the inverse hyperbolic cosine of the corresponding element in *list*.

```
cosh⁻¹ {1,2.1,3} [ENTER]
              {0 1.37285914424 1.7…
```

cross(

VECTR MATH menu

cross(*vectorA*,*vectorB***)**

Returns the cross product of two real or complex vectors, where:

cross([a,b,c],[d,e,f]) = [bf−ce cd−af ae−bd]

Both vectors must have the same dimension (either 2 or 3 elements). A 2-D vector is treated as a 3-D vector with 0 as the third element.

```
cross([1,2,3],[4,5,6]) [ENTER]
                        [⁻3 6 ⁻3]
cross([1,2],[3,4]) [ENTER]
                         [0 0 ⁻2]
```

cSum(

LIST OPS menu

cSum(*list***)**

Returns a list of the cumulative sums of the real or complex elements in *list*, starting with the first element.

```
cSum({1,2,3,4}) [ENTER]          {1 3 6 10}

{10,20,30}→L1 [ENTER]            {10 20 30}
cSum(L1) [ENTER]                 {10 30 60}
```

▸Cyl

VECTR OPS menu

vector▸**Cyl**

Displays a 2- or 3-element real *vector* result in cylindrical form, [r∠θ z], even if the display mode is not set for cylindrical (**CylV**).

```
[⁻2,0]▸Cyl [ENTER]
                   [2∠3.14159265359 0]

[⁻2,0,1]▸Cyl [ENTER]
                   [2∠3.14159265359 1]
```

CylV

† mode screen

CylV

Sets cylindrical vector coordinate mode ([r∠θ z]).

In **CylV** vector coordinate mode and **Radian** angle mode:

```
[3,4,5] [ENTER]    [5∠.927295218002 5]
```

d

BASE TYPE menu

number **d**

Designates a real *number* as decimal, regardless of the number base mode setting.

In **Bin** number base mode:

```
10d [ENTER]                      1010b
10d+10 [ENTER]                   1100b
```

Dec

† mode screen

Dec

Sets decimal number base mode. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the **b**, **d**, **h**, or **o** designator, respectively, from the BASE TYPE menu.

In **Dec** number base mode:

```
10+10b+Fh+10o [ENTER]               35
```

▶Dec

BASE CONV menu

number▶**Dec**
list▶**Dec**
matrix▶**Dec**
vector▶**Dec**

Returns the decimal equivalent of the real or complex argument.

In **Hex** number base mode:

```
2*F [ENTER]                 1Eh
Ans▶Dec [ENTER]             30d

{A,B,C,D,E}▶Dec [ENTER]
          {10d 11d 12d 13d 14d}
```

Degree

† mode screen

Degree

Sets degree angle mode.

In **Degree** angle mode:

```
sin 90 [ENTER]                     1
sin (π/2) [ENTER]     .027412133592
```

Degree entry: °

MATH ANGLE menu

number° or (*expression*)°

Designates a real *number* or *expression* as degrees, regardless of the angle mode setting.

list°

Designates each element in *list* as degrees.

In **Radian** angle mode:

```
cos 90 [ENTER]       ⁻.448073616129
cos 90° [ENTER]                   0

cos {45,90,180}° [ENTER]
           {.707106781187 0 ⁻1}
```

Deltalst(

LIST OPS menu
(Deltal shows on menu)

Deltalst(*list***)**

Returns a list containing the differences between consecutive real or complex elements in *list*. This subtracts the first element in *list* from the second element, the second from the third, and so on. The resulting list is always one element shorter than *list*.

```
Deltalst({20,30,45,70}) [ENTER]
                   {10 15 25}
```

DelVar(

‡ program editor CTL menu (DelVa shows on menu)

DelVar(*variable***)**

Deletes the specified user-created *variable* from memory.

You cannot use **DelVar(** to delete a program variable or built-in variable.

```
2→A [ENTER]                          2
(A+2)² [ENTER]                      16
DelVar(A) [ENTER]                 Done
(A+2)² [ENTER]      ERROR 14 UNDEFINED
```

der1(

CALC menu

der1(*expression*,*variable*,*value***)**

Returns the first derivative of *expression* with respect to *variable* at the real or complex *value*.

```
der1(x^3,x,5) [ENTER]               75
```

der1(*expression*,*variable***)**

Uses the current value of *variable*.

```
3→x [ENTER]                          3
der1(x^3,x) [ENTER]                 27
```

der1(*expression*,*variable*,*list***)**

Returns a list containing the first derivatives at the values specified by the elements in *list*.

```
der1(x^3,x,{5,3}) [ENTER]      {75 27}
```

der2(

CALC menu

der2(*expression*,*variable*,*value***)**

Returns the second derivative of *expression* with respect to *variable* at the real or complex *value*.

```
der2(x^3,x,5) [ENTER]               30
```

der2(*expression*,*variable***)**

Uses the current value of *variable*.

```
3→x [ENTER]                          3
der2(x^3,x) [ENTER]                 18
```

der2(*expression*,*variable*,*list***)**

Returns a list containing the second derivatives at the values specified by the elements in *list*.

```
der2(x^3,x,{5,3}) [ENTER]      {30 18}
```

det

MATRX MATH menu

det *squareMatrix*

Returns the determinant of *squareMatrix*. The result is real for a real matrix, complex for a complex matrix.

```
[[1,2][3,4]]→MAT [ENTER]        [[1 2]
                                 [3 4]]
det MAT [ENTER]                     -2
```

DifEq

† mode screen

DifEq

Sets differential equation graphing mode.

dim

MATRX OPS menu

VECTR OPS menu

dim *matrix*

Returns a list containing the dimensions (number of rows and columns) of a real or complex *matrix*.

dim *vector*

Returns the length (number of elements) of a real or complex *vector*.

```
[[2,7,1][-8,0,1]]→MAT [ENTER]
                              [[2  7 1]
                               [-8 0 1]]
dim MAT [ENTER]                   {2 3}
dim [-8,0,1] [ENTER]                  3
```

→dim

[STO▸], then MATRX OPS menu

[STO▸], then VECTR OPS menu

{*rows*,*columns***}→dim** *matrixName*

If *matrixName* does not exist, creates a new matrix with the specified dimensions and fills it with zeros.

If *matrixName* exists, redimensions that matrix to the specified dimensions. Existing elements within the new dimensions are not changed; elements outside the new dimensions are deleted. If additional elements are created, they are filled with zeros.

```
[[2,7][-8,0]]→MAT [ENTER]
                               [[2  7]
                                [-8 0]]
{3,3}→dim MAT [ENTER]            {3 3}
MAT [ENTER]                   [[2  7 0]
                               [-8 0 0]
                               [0  0 0]]
```

#ofElements→**dim** *vectorName*

If *vectorName* does not exist, creates a new vector with the specified *#ofElements* and fills it with zeros.

If *vectorName* exists, redimensions that vector to the specified *#ofElements*. Existing elements within the new dimension are not changed; elements outside the new dimension are deleted. If additional elements are created, they are filled with zeros.

```
DelVar(VEC) [ENTER]              Done
4→dim VEC [ENTER]                   4
VEC [ENTER]                 [0 0 0 0]

[1,2,3,4]→VEC [ENTER]       [1 2 3 4]
2→dim VEC [ENTER]                   2
VEC [ENTER]                     [1 2]
3→dim VEC [ENTER]                   3
VEC [ENTER]                   [1 2 0]
```

dimL

LIST OPS menu

dimL *list*

Returns the length (number of elements) of a real or complex *list*.

```
dimL {2,7,-8,0} [ENTER]             4

1/dimL {2,7,-8,0} [ENTER]         .25
```

→dimL

[STO▸], then LIST OPS menu

#ofElements→**dimL** *listName*

If *listName* does not exist, creates a new list with the specified *#ofElements* and fills it with zeros.

If *listName* exists, redimensions that list to the specified *#ofElements*. Existing elements within the new dimension are not changed; elements outside the new dimension are deleted. If additional elements are created, they are filled with zeros.

```
3→dimL NEWLIST [ENTER]              3
NEWLIST [ENTER]               {0 0 0}

{2,7,-8,1}→L1 [ENTER]      {2 7 -8 1}
5→dimL L1 [ENTER]                   5
L1 [ENTER]               {2 7 -8 1 0}
2→dimL L1 [ENTER]                   2
L1 [ENTER]                      {2 7}
```

DirFld

† graph format screen (scroll down to second screen)

DirFld

In **DifEq** graphing mode, turns on direction fields. To turn off direction and slope fields, use **FldOff**.

Disp

‡ program editor I/O menu

Disp *valueA*,*valueB*,*valueC*, ...

Displays each value. The values can include strings and variable names.

Disp

Displays the home screen.

```
10→x [ENTER]                     10
Disp x^3+3 x−6 [ENTER]         1024
                               Done
"Hello"→STR [ENTER]
                         Hello
Disp STR+", Jan" [ENTER]
                         Hello, Jan
                               Done
```

DispG

† GRAPH menu

‡ program editor I/O menu

DispG

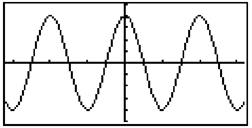
Displays the current graph.

Program segment in **Func** graphing mode:

```
  :
:y1=4cos x
:⁻10→xMin:10→xMax
:⁻5→yMin:5→yMax
:DispG
  :
```

Function names are case-sensitive. Use ***y1****, not* ***Y1****.*

To select from a list of window variable names, press [2nd] [CATLG-VARS] [MORE] [MORE] [F5].



DispT

‡ program editor
I/O menu

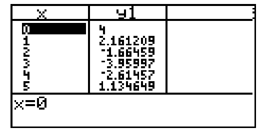
DispT

Displays the table.

Program segment in **Func** graphing mode:

```
   :
:y1=4cos x
:DispT
   :
```

Function names are case-sensitive. Use ***y1****, not* ***Y1****.*



Division: /

[÷]

numberA **/** *numberB* or **(***expressionA***)/(***expressionB***)**

Returns one argument divided by another. The arguments can be real or complex.

```
-98/4 [ENTER]                -24.5
-98/(4*3) [ENTER]     -8.16666666667
```

number **/** *list* or **(***expression***)/** *list*

Returns a list in which each element is *number* or *expression* divided by the corresponding element in *list*.

```
100/{10,25,2} [ENTER]      {10 4 50}
```

list **/** *number* or *list* **/(***expression***)**
vector **/** *number* or *vector* **/(***expression***)**

Returns a list or vector in which each element of *list* or *vector* is divided by *number* or *expression*.

```
{120,92,8}/4 [ENTER]       {30 23 2}
```

In **RectC** complex number mode:

```
[8,1,(5,2)]/2 [ENTER]
               [(4,0) (.5,0) (2.5,1…
```

listA **/** *listB*

Returns a list in which each element of *listA* is divided by the corresponding element of *listB*. The lists must have the same dimension.

```
{1,2,3}/{4,5,6} [ENTER]
                      {.25 .4 .5}
```

DMS entry: '

MATH ANGLE menu

*In a trig calculation, the result of a DMS entry is treated as degrees in the **Degree** angle mode only. It is treated as radians in **Radian** angle mode.*

degrees'*minutes*'*seconds*'

Designates the entered angle is in DMS format. *degrees* (≤ 999,999), *minutes* (< 60), and *seconds* (< 60, may have decimal places) must be entered as real numbers, not as variable names or expressions.

Do not use ° and " symbols to specify *degrees* and *seconds*. For example, 5°59' is interpreted as implied multiplication of 5° ∗ 59' according to the current angle mode setting.

`54'32'30'` [ENTER] `54.5416666667`

In **Degree** angle mode:

`cos 54'32'30'` [ENTER] `.580110760699`

In **Radian** angle mode:

`cos 54'32'30'` [ENTER] `⁻.422502666138`

Do not use the following notation; in **Degree** angle mode:

`5°59'` [ENTER] `295`

▸DMS

MATH ANGLE menu

angle▸**DMS**

Displays *angle* in DMS format. The result is shown in *degrees*°*minutes*'*seconds*" format, even though you use *degrees*'*minutes*'*seconds*' to enter a DMS angle.

In **Degree** angle mode:

`45.371▸DMS` [ENTER] `45°22'15.6"`

`54'32'30'∗2` [ENTER] `109.083333333`
`Ans▸DMS` [ENTER] `109°5'0"`

dot(

VECTR MATH menu

dot(*vectorA***,***vectorB***)**

Returns the dot product of two real or complex vectors.

dot([a,b,c],[d,e,f]) returns **a∗d**+**b∗e**+**c∗f**.

`dot([1,2,3],[4,5,6])` [ENTER] `32`

DrawDot

† graph format screen

DrawDot

Sets dot graphing format.

DrawF

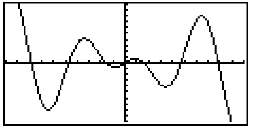
GRAPH DRAW menu

DrawF *expression*

Draws *expression* (in terms of **x**) on the current graph.

In **Func** graphing mode:

`ZStd:DrawF 1.25 x cos x` [ENTER]



DrawLine

† graph format screen

DrawLine

Sets connected line graphing format.

DrEqu(

† GRAPH menu

To enter the ' character for the Q' variables, use the CHAR MISC *menu.*

DrEqu(*xAxisVariable***,***yAxisVariable***,***xList***,***yList***,***tList***)**

In **DifEq** graphing mode, draws the solution to a set of differential equations stored in the **Q'** variables specified by *xAxisVariable* and *yAxisVariable*. If direction fields are off (**FldOff** is selected), the initial values must be stored also.

After the solution is drawn, **DrEqu(** waits for you to move the cursor to a new initial value and press [ENTER] to draw the new solution.

You then are prompted to press **Y** (to specify another initial value) or **N** (to stop).

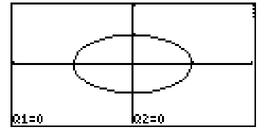
For the last-drawn solution, the **x**, **y**, and **t** values (beginning at their initial values) are stored to *xList*, *yList*, and *tList*, respectively.

DrEqu(*xAxisVariable***,***yAxisVariable***)**

Does not store **x**, **y**, and **t** values for the solution.

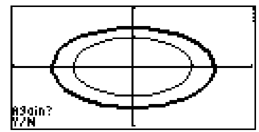
In **DifEq** graphing mode, starting with a **ZStd** graph screen:

```
Q'1=Q2:Q'2=⁻Q1 [ENTER]          Done
0→tMin:1→QI1:0→QI2 [ENTER]         0
DrEqu(Q1,Q2,XL,YL,TL) [ENTER]
```



Move the cursor to a new initial value.

[ENTER]



Press **N** to stop graphing. You can then examine **XL**, **YL**, and **TL**.

DrInv

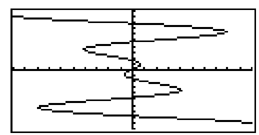
GRAPH DRAW menu

DrInv *expression*

Draws the inverse of *expression* by plotting **x** values on the y-axis and **y** values on the x-axis.

In **Func** graphing mode:

```
ZStd:DrInv 1.25 x cos x [ENTER]
```



DS<(

‡ program editor CTL menu

:DS<(*variable*,*value***)**
:*command-if-variable≥value*
:*commands*

Decrements *variable* by 1. If the result is < *value*, skips *command-if-variable≥value*.

If the result is ≥ *value*, then *command-if-variable≥value* is executed.

variable cannot be a built-in variable.

Program segment:

```
  :
:9→A
:Lbl Start
:Disp A
:DS<(A,5)
:Goto Start
:Disp "A is now <5"
  :
```

dxDer1

† mode screen

dxDer1

Sets **der1** as the current differentiation type. **der1** differentiates exactly and calculates the value for each function in an expression. It is more accurate than **dxNDer**, but more restrictive in that only certain functions are valid in the expression.

The current differentiation type is used by the **arc(** and **TanLn(** functions, as well as interactive graphing operations dy/dx, dr/dθ, dy/dt, dx/dt, ARC, TanLn, and INFLC.

dxNDer

† mode screen

dxNDer

Sets **nDer** as the current differentiation type. **nDer** differentiates numerically and calculates the value for an expression. It is less accurate than **dxDer1**, but less restrictive in the functions that are valid in the expression.

The current differentiation type is used by the **arc(** and **TanLn(** functions, as well as interactive graphing operations dy/dx, dr/dθ, dy/dt, dx/dt, ARC, TanLn, and INFLC.

e^

[2nd] [e^x]

e^*power* or **e^(***expression***)**

Returns **e** raised to *power* or *expression*. The argument can be real or complex.

`e^0` [ENTER] 1

e^*list*

Returns a list in which each element is **e** raised to the power specified by the corresponding element in *list*.

```
e^{1,0,.5} [ENTER]
                {2.71828182846 1 1.6…
```

e^*squareMatrix*

The squareMatrix cannot have repeated eigenvalues.

Returns a square matrix that is the matrix exponential of *squareMatrix*. The matrix exponential corresponds to the result calculated using power series or Cayley-Hamilton Theorem techniques. This is *not* the same as simply calculating the exponential of each element.

---

eigVc

MATRX MATH menu

The squareMatrix cannot have repeated eigenvalues.

eigVc *squareMatrix*

Returns a matrix containing the eigenvectors for a real or complex *squareMatrix*, where each column in the result corresponds to an eigenvalue. The eigenvectors of a real matrix may be complex. Note that an eigenvector is not unique; it may be scaled by any constant factor. TI-86 eigenvectors are normalized.

In **RectC** complex number mode:

```
[[-1,2,5][3,-6,9][2,-5,7]]→MAT
[ENTER]                  [[-1 2  5]
                          [3  -6 9]
                          [2  -5 7]]
eigVc MAT [ENTER]
               [[(.800906446592,0) …
                [(-.484028886343,0)…
                [(-.352512270699,0)…
```

---

eigVl

MATRX MATH menu

eigVl *squareMatrix*

Returns a list of the eigenvalues of a real or complex *squareMatrix*. The eigenvalues of a real matrix may be complex.

In **RectC** complex number mode:

```
[[-1,2,5][3,-6,9][2,-5,7]]→MAT
[ENTER]                  [[-1 2  5]
                          [3  -6 9]
                          [2  -5 7]]
eigVl MAT [ENTER]
               {(-4.40941084667,0) …
```

Else

‡ program editor
CTL menu

Refer to syntax information for **If**, beginning on page 305. See the **If:Then:Else:End** syntax.

End

‡ program editor
CTL menu

End

Identifies the end of a **While**, **For**, **Repeat**, or **If-Then-Else** loop.

Eng

† mode screen

Eng

Sets engineering notation mode, in which the power-of-10 exponent is a multiple of 3.

In **Eng** notation mode:

```
123456789 [ENTER]          123.456789E6
```

In **Normal** notation mode:

```
123456789 [ENTER]             123456789
```

Eq▶St(

STRNG menu

Eq▶St(*equationVariable*,*stringVariable***)**

Converts the contents of *equationVariable* to a string and stores it to *stringVariable*. Be sure to specify an equation variable, not an equation.

To create an equation variable, use an equal sign (=) to define the variable. For example, enter **A=B∗C**, not **B∗C→A**.

```
A=B∗C [ENTER]                 Done
5→B [ENTER]                      5
2→C [ENTER]                      2
A [ENTER]                       10
Eq▶St(A,STR) [ENTER]          Done
STR [ENTER]             B∗C
```

Equal: =

[ALPHA] [=]

Refer to syntax information for **Assignment** on page 270.

If you use **=** in an expression in which the first argument is not a variable name at the beginning of a line, the **=** is treated as −(.

Example of = treated as −(, where 4=6+1 is evaluated as 4−(6+1):

```
4=6+1 [ENTER]                   -3
```

For true/false comparison, use == instead:

```
4==6+1 [ENTER]                   0
```

Equal to: ==

TEST menu

The == operator is used to compare arguments, while = is used to assign a value or expression to a variable.

numberA **==** *numberB*
matrixA **==** *matrixB*
vectorA **==** *vectorB*
stringA **==** *stringB*

Tests whether the condition *argumentA* == *argumentB* is true or false. Numbers, matrices, and vectors can be real or complex. If complex, the magnitude (modulus) of each element is compared. Strings are case-sensitive.

- If true (*argumentA* = *argumentB*), returns **1**.
- If false (*argumentA* ≠ *argumentB*), returns **0**.

```
2+2==2+2 [ENTER]                1
2+(2==2)+2 [ENTER]              5
[1,2]==[3-2,⁻1+3] [ENTER]       1
"A"=="a" [ENTER]                0
```

listA **==** *listB*

Returns a list of **1**s and/or **0**s to indicate if each element in *listA* is = the corresponding element in *listB*.

```
{1,5,9}=={1,⁻6,9} [ENTER]   {1 0 1}
```

Euler

† graph format screen (scroll down to second screen)

Euler

In **DifEq** graphing mode, uses an algorithm based on the Euler method to solve differential equations. Typically, **Euler** is less accurate than **RK** but finds the solutions much quicker.

eval

MATH MISC menu

eval *xValue*

Returns a list containing the **y** values of all defined and selected functions evaluated at a real *xValue*.

Remember that built-in equation variables **y1** and **y2** are case-sensitive:

```
y1=x^3+x+5 [ENTER]          Done
y2=2 x [ENTER]              Done
eval 5 [ENTER]          {135 10}
```

evalF(

CALC menu

evalF(*expression*,*variable*,*value***)**

Returns the value of *expression* evaluated with respect to *variable* at a real or complex *value*.

`evalF(x^3+x+5,x,5)` [ENTER] → `135`

evalF(*expression*,*variable*,*list***)**

Returns a list containing the values of *expression* evaluated with respect to *variable* at each element in *list*.

`evalF(x^3+x+5,x,{3,5})` [ENTER] → `{35 135}`

Exponent: E

[EE]

number **E** *power* or **(***expressionA***) E (***expressionB***)**

Returns a real or complex *number* raised to the *power* of 10, where *power* is a real integer such that ⁻999 < *power* < 999. Any *expressions* must evaluate to appropriate values.

`12.3456789E5` [ENTER] → `1234567.89`

`(1.78/2.34)E2` [ENTER] → `76.0683760684`

list **E** *power* or *list* **E (***expression***)**

Returns a list in which each element is the corresponding element in *list* raised to the *power* of 10.

`{6.34,854.6}E3` [ENTER] → `{6340 854600}`

ExpR

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

ExpR *xList*,*yList*,*frequencyList*,*equationVariable*

Fits an exponential regression model ($y=ab^x$) to real data pairs in *xList* and *yList* (**y** values must be > 0) and frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

ExpR *xList*,*yList*,*equationVariable*

Uses frequencies of 1.

ExpR *xList*,*yList*,*frequencyList*

Stores the regression equation to **RegEq** only.

ExpR *xList*,*yList*

Uses frequencies of 1, and stores the regression equation to **RegEq** only.

ExpR *equationVariable*

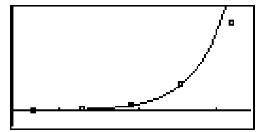
Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

In **Func** graphing mode:

```
{1,2,3,4,5}→L1 [ENTER]
                                  {1 2 3 4 5}
{1,20,55,230,742}→L2 [ENTER]
                             {1 20 55 230 742}
ExpR L1,L2,y1 [ENTER]
```

```
ExpReg
 y=a*b^x
 a=.411389488
 b=4.78796057
 corr=.97681282
 n=5
```

```
Plot1(1,L1,L2) [ENTER]                    Done
ZData [ENTER]
```



ExpR

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

Factorial: !

MATH PROB menu

number! or (*expression*)!

Returns the factorial of a real integer or non-integer, where 0 ≤ integer ≤ 449 and 0 ≤ non-integer ≤ 449.9. For a non-integer, the Gamma function is used to find the factorial. An *expression* must evaluate to an appropriate value.

`6!` [ENTER]	`720`
`12.5!` [ENTER]	`1710542068.32`

list!

Returns a list in which each element is the factorial of the corresponding element in *list*.

`{6,7,8}!` [ENTER]	`{720 5040 40320}`

---

fcstx

† STAT menu

fcstx *yValue*

Based on the current regression equation (**ReqEq**), returns the forecasted **x** at a real *yValue*.

---

fcsty

† STAT menu

fcsty *xValue*

Based on the current regression equation (**ReqEq**), returns the forecasted **y** at a real *xValue*.

Fill(

LIST OPS menu

MATRX OPS menu

VECTR OPS menu

Fill(*number*,*listName***)**
Fill(*number*,*matrixName***)**
Fill(*number*,*vectorName***)**

Replaces each element in an existing *listName*, *matrixName*, or *vectorName* with a real or complex *number*.

```
{3,4,5}→L1 [ENTER]                 {3 4 5}
Fill(8,L1) [ENTER]                    Done
L1 [ENTER]                         {8 8 8}

Fill((3,4),L1) [ENTER]                Done
L1 [ENTER]           {(3,4) (3,4) (3,4)}
```

Fix

† mode screen

Fix *integer* or **Fix (***expression***)**

Sets fixed decimal mode for *integer* number of decimal places, where $0 \le$ *integer* ≤ 11. An *expression* must evaluate to an appropriate integer.

```
Fix 3 [ENTER]                         Done
π/2 [ENTER]                          1.571
Float [ENTER]                         Done
π/2 [ENTER]                  1.57079632679
```

FldOff

† graph format screen (scroll down to second screen)

FldOff

In **DifEq** graphing mode, turns off the slope and direction fields. To turn on slope fields, use **SlpFld**. To turn on direction fields, use **DirFld**.

Float

† mode screen

Float

Sets floating decimal mode.

In **Radian** angle mode:

```
Fix 11 [ENTER]                        Done
sin (π/6) [ENTER]             .50000000000
Float [ENTER]                         Done
sin (π/6) [ENTER]                       .5
```

fMax(

CALC menu

fMax(*expression*,*variable*,*lower*,*upper***)**

Returns the value at which a local maximum of *expression* with respect to *variable* occurs, between real *lower* and *upper* values for *variable*.

The tolerance is controlled by the built-in variable **tol**, whose default is 1E⁻5. To view or set **tol,** press [2nd] [MEM] [F4] to display the tolerance editor.

```
fMax(sin x,x,⁻π,π) [ENTER]
                      1.57079632598
```

fMin(

CALC menu

fMin(*expression*,*variable*,*lower*,*upper***)**

Returns the value at which a local minimum of *expression* with respect to *variable* occurs, between real *lower* and *upper* bounds for *variable*.

The tolerance is controlled by the built-in variable **tol**, whose default is 1E⁻5. To view or set **tol,** press [2nd] [MEM] [F4] to display the tolerance editor.

```
fMin(sin x,x,⁻π,π) [ENTER]
                     ⁻1.57079632691
```

fnInt(

CALC menu

fnInt(*expression*,*variable*,*lower*,*upper***)**

Returns the numerical function integral of *expression* with respect to *variable*, between real *lower* and *upper* bounds for *variable*.

The tolerance is controlled by the built-in variable **tol**, whose default is 1E⁻5. To view or set **tol,** press [2nd] [MEM] [F4] to display the tolerance editor.

```
fnInt(x²,x,0,1) [ENTER]
                      .333333333333
```

FnOff

† GRAPH VARS menu

FnOff *function#*,*function#*, ...

Deselects the specified equation function numbers.

```
FnOff 1,3 [ENTER]              Done
```

FnOff

Deselects all equation function numbers.

`FnOff [ENTER]` Done

FnOn

† GRAPH VARS menu

FnOn *function#*,*function#*, ...

Selects the specified equation function numbers, in addition to any others already selected.

`FnOn 1,3 [ENTER]` Done

FnOn

Selects all equation function numbers.

`FnOn [ENTER]` Done

For(

‡ program editor CTL menu

:For(*variable*,*begin*,*end*,*step***)** or **:For(***variable*,*begin*,*end***)**
:*loop* :*loop*
:End **:End**
:*commands* :*commands*

Executes the commands in *loop* iteratively, where the number of repetitions is controlled by *variable*. The first time through the loop, *variable* = *begin*. At the **End** of the loop, *variable* is incremented by *step*. The loop is repeated until *variable* > *end*. If you do not specify *step*, the default is 1.

You can specify values such that *begin* > *end*. If so, be sure to specify a negative *step*.

Program segment:

```
 :
For(A,0,8,2)
Disp A²
End
 :
```

Displays 0, 4, 16, 36, and 64.

```
 :
For(A,0,8)
Disp A²
End
 :
```

Displays 0, 1, 4, 9, 16, 25, 36, 49, and 64.

Form(

LIST OPS menu

Form("*formula***",***listName***)**

Generates the contents of *listName* automatically, based on the attached *formula*. If you express *formula* in terms of a list, you can generate one list based on the contents of another.

The contents of *listName* are updated automatically if you edit *formula* or edit a list referenced in *formula*.

```
{1,2,3,4}→L1 [ENTER]              {1 2 3 4}
Form("10*L1",L2) [ENTER]               Done
L2 [ENTER]                    {10 20 30 40}

{5,10,15,20}→L1 [ENTER]
                              {5 10 15 20}
L2 [ENTER]                 {50 100 150 200}

Form("L1/5",L2) [ENTER]                Done
L2 [ENTER]                        {1 2 3 4}
```

fPart

MATH NUM menu

fPart *number* or **fPart (***expression***)**

Returns the fractional part of a real or complex *number* or *expression*.

fPart *list*
fPart *matrix*
fPart *vector*

Returns a list, matrix, or vector in which each element is the fractional part of the corresponding element in the specified argument.

```
fPart 23.45 [ENTER]                     .45

fPart (-17.26*8) [ENTER]               -.08

[[1,-23.45][-99.5,47.15]]→MAT
[ENTER]                    [[1     -23.45]
                            [-99.5 47.15 ]]

fPart MAT [ENTER]             [[0   -.45]
                               [-.5 .15 ]]
```

►Frac

MATH MISC menu

number►**Frac**

Displays a real or complex *number* as its rational equivalent, a fraction reduced to its simplest terms.

If *number* cannot be simplified or if the denominator is more than four digits, the decimal equivalent is returned.

```
1/3+2/7 [ENTER]              .619047619048
Ans►Frac [ENTER]                      13/21
```

list▶**Frac**
matrix▶**Frac**
vector▶**Frac**

Returns a list, matrix, or vector in which each element is the rational equivalent of the corresponding element in the argument.

```
{1/2+1/3,1/6-3/8}→L1 [ENTER]
            {.833333333333 ⁻.208…
Ans▶Frac [ENTER]        {5/6 ⁻5/24}
```

Func

† mode screen

Func

Sets function graphing mode.

gcd(

MATH MISC menu

gcd(*integerA*,*integerB***)**

Returns the greatest common divisor of two nonnegative integers.

```
gcd(18,33) [ENTER]                3
```

gcd(*listA*,*listB***)**

Returns a list in which each element is the gcd of the two corresponding elements in *listA* and *listB*.

```
gcd({12,14,16},{9,7,5}) [ENTER]
                          {3 7 1}
```

Get(

‡ program editor
I/O menu

Get(*variable***)**

Gets data from a CBL or CBR System or another TI-86 and stores it to *variable*.

getKy

‡ program editor
I/O menu

getKy

Returns the key code for the last key pressed. If no key has been pressed, **getKy** returns **0**. Refer to the TI-86 key code diagram in Chapter 16.

Program:

```
PROGRAM:CODES
:Lbl TOP
:getKy→KEY
:While KEY==0
: getKy→KEY
:End
:Disp KEY
:Goto TOP
```

To break the program, press [ON] and then [F5].

Goto

‡ program editor
CTL menu

Goto *label*

Transfers (branches) program control to the *label* specified by an existing **Lbl** instruction.

Program segment:

```
 :
:0→TEMP:1→J
:Lbl TOP
:TEMP+J→TEMP
:If J<10
:Then
: J+1→J
: Goto TOP
:End
:Disp TEMP
 :
```

Greater than: >

TEST menu

numberA **>** *numberB* or **(***expressionA***)** **>** **(***expressionB***)**

Tests whether the condition is true or false. The arguments must be real numbers.

- If true (*numberA* > *numberB*), returns **1**.
- If false (*numberA* ≤ *numberB*), returns **0**.

2>0 [ENTER]	1
88>123 [ENTER]	0
-5>-5 [ENTER]	0
(20*5/2)>(18*2) [ENTER]	1

number **>** *list*

Returns a list of **1**s and/or **0**s to indicate if *number* is > the corresponding element in *list*.

`1>{1,-6,10} [ENTER]` → `{0 1 0}`

listA **>** *listB*

Returns a list of **1**s and/or **0**s to indicate if each element in *listA* is > the corresponding element in *listB*.

`{1,5,9}>{1,-6,10} [ENTER]` → `{0 1 0}`

Greater than or equal to: ≥

TEST menu

numberA ≥ *numberB* or **(***expressionA***)** ≥ **(***expressionB***)**

Tests whether the condition is true or false. The arguments must be real numbers.

- If true (*numberA* ≥ *numberB*), returns **1**.
- If false (*numberA* < *numberB*), returns **0**.

`2≥0 [ENTER]` → `1`

`88≥123 [ENTER]` → `0`

`-5≥-5 [ENTER]` → `1`

`(20*5/2)≥(18*2) [ENTER]` → `1`

number ≥ *list*

Returns a list of **1**s and/or **0**s to indicate if *number* is ≥ the corresponding element in *list*.

`1≥{1,-6,10} [ENTER]` → `{1 1 0}`

listA ≥ *listB*

Returns a list of **1**s and/or **0**s to indicate if each element in *listA* is ≥ the corresponding element in *listB*.

`{1,5,9}≥{1,-6,10} [ENTER]` → `{1 1 0}`

GridOff

† graph format screen

GridOff

Turns off grid format so that grid points are not displayed.

GridOn

† graph format screen

GridOn

Turns on grid format so that grid points are displayed in rows and columns corresponding to the tick marks on each axis.

GrStl(

CATALOG

GrStl(*function#*,*graphStyle#***)**

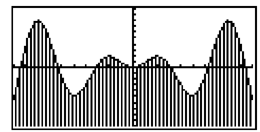
Sets the graph style for *function#*. For *graphStyle#*, specify an integer from **1** through **7**:

1 = (line)	**4** = (below)	**7** = (dot)
2 = (thick)	**5** = (path)	
3 = (above)	**6** = (animate)	

Depending on the graphing mode, some graph styles may not be available.

In **Func** graphing mode:

```
y1=x sin x [ENTER]          Done
GrStl(1,4) [ENTER]          Done
ZStd [ENTER]
```



h

BASE TYPE menu

*integer***h**

Designates a real *integer* as hexadecimal, regardless of the number base mode setting.

In **Dec** number base mode:

```
10h [ENTER]          16
10h+10 [ENTER]       26
```

Hex

† mode screen

Hex

Sets hexadecimal number base mode. Results are displayed with the **h** suffix. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the **b**, **d**, **h**, or **o** designator, respectively, from the BASE TYPE menu.

To enter hexadecimal numbers **A** through **F**, use the BASE A-F menu. Do not use [ALPHA] to type a letter.

In **Hex** number base mode:

```
F+10b+10o+10d [ENTER]          23h
```

▶Hex

BASE CONV menu

number▶**Hex**
list▶**Hex**
matrix▶**Hex**
vector▶**Hex**

Returns the hexadecimal equivalent of the real or complex argument.

In **Bin** number base mode:

```
1010*1110 [ENTER]        10001100b
Ans▶Hex [ENTER]                8Ch

{100,101,110}▶Hex [ENTER]
                        {4h 5h 6h}
```

Hist

† STAT DRAW menu

Hist *xList,frequencyList*

Draws a histogram on the current graph, using the real data in *xList* and the frequencies in *frequencyList*.

Hist *xList*

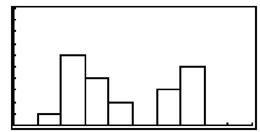
Uses frequencies of 1.

Hist

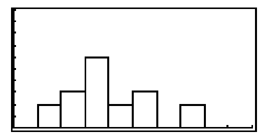
Uses the data in built-in variables **xStat** and **fStat**. These variables must contain valid data of the same dimension; otherwise, an error occurs.

Starting with a **ZStd** graph screen:

```
{1,2,3,4,6,7}→XL [ENTER]
                     {1 2 3 4 6 7}
{1,6,4,2,3,5}→FL [ENTER]
                     {1 6 4 2 3 5}
0→xMin:0→yMin [ENTER]             0
Hist XL,FL [ENTER]
```



```
{1,1,2,2,2,3,3,3,3,3,3,4,4,5,5,5,
7,7}→XL [ENTER]
              {1 1 2 2 2 3 3 3 3 3 …
ClDrw:Hist XL [ENTER]
```



Horiz

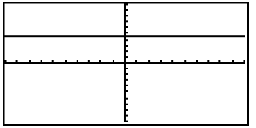
† GRAPH DRAW menu

Horiz *yValue*

Draws a horizontal line on the current graph at *yValue*.

In a **ZStd** graph screen:

```
Horiz 4.5 [ENTER]
```



IAsk

CATALOG

IAsk

Sets the table so that the user can enter individual values for the independent variable.

IAuto

CATALOG

IAuto

Sets the table so that the TI−86 generates the independent-variable values automatically, based on values entered for **TblStart** and **∆Tbl**.

ident

MATRX OPS menu

ident *dimension*

Returns the identity matrix of *dimension* rows × *dimension* columns.

```
ident 4 [ENTER]
                [[1 0 0 0]
                 [0 1 0 0]
                 [0 0 1 0]
                 [0 0 0 1]]
```

If

‡ program editor
CTL menu

:If *condition*
:*command-if-true*
:*commands*

If *condition* is true, executes *command-if-true*. Otherwise, skips *command-if-true*. The *condition* is true if it evaluates to any nonzero number, or false if it evaluates to zero.

To execute multiple commands if *condition* is true, use **If:Then:End** instead.

Program segment:

```
  :
:If x<0
:Disp "x is negative"
  :
```

:If *condition*
:Then
:*commands-if-true*
:End
:*commands*

If *condition* is true (nonzero), executes *commands-if-true* from **Then** to **End**. Otherwise, skips *commands-if-true* and continues with the next command following **End**.

Program segment:

```
  :
:If x<0
:Then
: Disp "x is negative"
: abs(x)→x
:End
  :
```

:If *condition*
:Then
:*commands-if-true*
:Else
:*commands-if-false*
:End
:*commands*

If *condition* is true (nonzero), executes *commands-if-true* from **Then** to **Else** and then continues with the next command following **End**.

If *condition* is false (zero), executes *commands-if-false* from **Else** to **End** and then continues with the next command following **End**.

Program segment:

```
  :
:If x<0
:Then
: Disp "x is negative"
:Else
: Disp "x is positive or zero"
:End
  :
```

imag

CPLX menu

imag (*complexNumber***)**

Returns the imaginary (nonreal) part of *complexNumber*. The imaginary part of a real number is always 0.

imag (*real*,*imaginary***)** returns *imaginary*.
imag (*magnitude*∠*angle***)** returns *magnitude* **sin** *angle*.

```
imag (3,4) [ENTER]                4
imag (3∠4) [ENTER]   -2.27040748592
```

imag *complexList*
imag *complexMatrix*
imag *complexVector*

Returns a list, matrix, or vector in which each element is the imaginary part of the original argument.

```
imag {-2,(3,4),(3∠4)} [ENTER]
             {0 4 -2.27040748592}
```

InpSt

‡ program editor
I/O menu

InpSt *promptString*,*variable*

Pauses a program, displays *promptString*, and waits for the user to enter a response. The response is stored to *variable* always as a string. When entering the response, the user should not enter quotation marks.

To prompt for a number or expression instead of a string, use **Input**.

InpSt *variable*

Displays **?** as the prompt.

Program segment:

```
     :
:InpSt "Enter your name:",STR
     :
```

Input

‡ program editor
I/O menu

Input *promptString*,*variable*

Pauses a program, displays *promptString*, and waits for the user to enter a response. The response is stored to *variable* in the form in which the user enters it.

- A number or expression is stored as a number or expression.
- A list, vector, or matrix is stored as a list, vector, or matrix.
- An entry enclosed in " marks is stored as a string.

Input *variable*

Displays **?** as the prompt.

Program segment:

```
     :
:Input "Enter test score:",SCR
     :
```

Input

Pauses a program, displays the graph screen, and lets the user update **x** and **y** (or **r** and θ in **PolarGC** graph format) by moving the free-moving cursor. To resume the program, press [ENTER].

Program segment in **RectGC** graph format:

```
  :
:Input
:Disp x,y
  :
```

Input "CBLGET",*variable*

Receives list data sent from a CBL or CBR System and stores it to *variable* on the TI-86. Use this **"CBLGET"** syntax for both CBL and CBR.

You can receive data also by using **Get(** as described on page 299.

```
Input "CBLGET",L1 [ENTER]          Done
```

---

int

MATH NUM menu

int *number* or **int (***expression***)**

Returns the largest integer ≤ *number* or *expression*. The argument can be real or complex.

For a negative non-integer, **int** returns the integer that is one less than the integer part of the number. To return the exact integer part, use **iPart** instead.

```
int 23.45 [ENTER]                  23
int -23.45 [ENTER]                -24
```

int *list*
int *matrix*
int *vector*

Returns a list, matrix, or vector in which each element is the largest integer less than or equal to the corresponding element in the specified argument.

```
[[1.25,-23.45][-99,47.15]]→MAT
[ENTER]                 [[1.25 -23.45]
                         [-99  47.15 ]]
int MAT [ENTER]              [[1   -24]
                              [-99 47 ]]
```

inter(

† MATH menu

inter(*x1*,*y1*,*x2*,*y2*,*xValue***)**

Calculates the line through points (*x1*,*y1*) and (*x2*,*y2*) and then interpolates or extrapolates a **y** value for the specified *xValue*.

Using points (3,5) and (4,4), find the y value at x=1:

```
inter(3,5,4,4,1) [ENTER]          7
```

inter(*y1*,*x1*,*y2*,*x2*,*yValue***)**

Interpolates or extrapolates an **x** value for the specified *yValue*. Notice that points (*x1*,*y1*) and (*x2*,*y2*) must be entered as (*y1*,*x1*) and (*y2*,*x2*).

Using points (-4,-7) and (2,6), find the x value at y=10:

```
inter(-7,-4,6,2,10) [ENTER]
                 3.84615384615
```

Inverse: $^{-1}$

[2nd] [x^{-1}]

number^{-1} or **(***expression***)**$^{-1}$

Returns 1 divided by a real or complex *number*, where *number* ≠ 0.

```
5⁻¹ [ENTER]                      .2
(10*6)⁻¹ [ENTER]      .016666666667
```

list^{-1}

Returns a list in which each element is 1 divided by the corresponding element in *list*.

```
{-.5,10,2/8}⁻¹ [ENTER]   {-2 .1 4}
```

squareMatrix^{-1}

Returns an inverted *squareMatrix*, where det ≠ 0.

```
[[1,2][3,4]]⁻¹ [ENTER]   [[-2  1  ]
                          [1.5 -.5]]
```

iPart

MATH NUM menu

iPart *number* or **iPart (***expression***)**

Returns the integer part of *number* or *expression*. The argument can be real or complex.

```
iPart 23.45 [ENTER]         23
iPart -23.45 [ENTER]       -23
```

iPart *list*
iPart *matrix*
iPart *vector*

Returns a list, matrix, or vector in which each element is the integer part of the corresponding element in the specified argument.

```
[[1.25,-23.45][-99.5,47.15]]→MAT
[ENTER]                 [[1.25  -23.45]
                         [-99.5 47.15 ]]

iPart MAT [ENTER]           [[1   -23]
                             [-99 47 ]]
```

IS>(

‡ program editor CTL menu

:IS>(*variable*,*value***)**
:*command-if-variable≤value*
:*commands*

Increments *variable* by 1. If the result is > *value*, skips *command-if-variable≤value*.

If the result is ≤ *value*, then *command-if-variable≤value* is executed.

variable cannot be a built-in variable.

Program segment:

```
 :
:0→A
:Lbl Start
:Disp A
:IS>(A,5)
:Goto Start
:Disp "A is now >5"
 :
```

LabelOff

† graph format screen

LabelOff

Turns off axes labels.

LabelOn

† graph format screen

LabelOn

Turns on axes labels.

Lbl

‡ program editor
CTL menu

Lbl *label*

Creates a *label* of up to eight characters. A program can use a **Goto** instruction to transfer control (branch) to a specified label.

Program segment, assuming a correct password has already been stored to the **password** variable:

```
  :
:Lbl Start
:InpSt "Enter password:",PSW
:If PSW≠password
:Goto Start
:Disp "Welcome"
  :
```

***InpSt** stores input as a string, so be sure to store a string to the **password** variable.*

lcm(

MATH MISC menu

lcm(*integerA*,*integerB***)**

Returns the least common multiple of two nonnegative integers.

```
lcm(5,2) [ENTER]          10
lcm(6,9) [ENTER]          18
lcm(18,33) [ENTER]       198
```

LCust(

‡ program editor
CTL menu

LCust(*item#*,**"***title***"** [,*item#*,**"***title***"**, ...]**)**

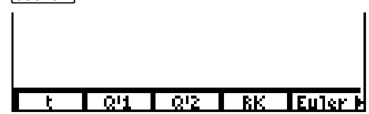
Loads (defines) the TI-86's custom menu, which is displayed when the user presses [CUSTOM]. The menu can have up to 15 items, shown in three groups of five items. For each *item#*/*title* pair:

- *item#* — integer from 1 through 15 that identifies the item's position in the menu. The item numbers must be specified in order, but you can skip numbers.
- **"***title***"** — string with up to 8 characters (not counting the quotes) that will be pasted to the current cursor location when the item is selected. This can be a variable name, expression, function name, program name, or any text string.

Program segment:

```
  :
:LCust(1,"t",2,"Q'1",3,"Q'2",4,"R
K",5,"Euler",6,"QI1",7,"QI2",8,"t
Min")
  :
```

After executed and when the user presses [CUSTOM]:



Less than: <

TEST menu

numberA **<** *numberB* or **(***expressionA***)** **<** **(***expressionB***)**

Tests whether the condition is true or false. The arguments must be real numbers.

- If true (*numberA* < *numberB*), returns **1**.
- If false (*numberA* ≥ *numberB*), returns **0**.

`2<0` [ENTER]	`0`
`88<123` [ENTER]	`1`
`-5<-5` [ENTER]	`0`
`(20*5/2)<(18*3)` [ENTER]	`1`

number **<** *list*

Returns a list of **1**s and/or **0**s to indicate if *number* is < the corresponding element in *list*.

`1<{1,-6,10}` [ENTER]	`{0 0 1}`

listA **<** *listB*

Returns a list of **1**s and/or **0**s to indicate if each element in *listA* is < the corresponding element in *listB*.

`{1,5,9}<{1,-6,10}` [ENTER]	`{0 0 1}`

Less than or equal to: ≤

TEST menu

numberA ≤ *numberB* or **(***expressionA***)** ≤ **(***expressionB***)**

Tests whether the condition is true or false. The arguments must be real numbers.

- If true (*numberA* ≤ *numberB*), returns **1**.
- If false (*numberA* > *numberB*), returns **0**.

`2≤0` [ENTER]	`0`
`88≤123` [ENTER]	`1`
`-5≤-5` [ENTER]	`1`
`(20*5/2)≤(18*3)` [ENTER]	`1`

number ≤ *list*

Returns a list of **1**s and/or **0**s to indicate if *number* is ≤ the corresponding element in *list*.

`1≤{1,-6,10}` [ENTER]	`{1 0 1}`

listA ≤ *listB*

Returns a list of **1**s and/or **0**s to indicate if each element in *listA* is ≤ the corresponding element in *listB*.

`{1,5,9}≤{1,-6,10}` [ENTER]	`{1 0 1}`

LgstR

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

LgstR *returns a* **tolMet** *value that indicates if the result meets the TI-86's internal tolerance.*

- *If* **tolMet**=**1**, *the result is within the internal tolerance.*
- *If* **tolmet**=**0**, *the result is outside the internal tolerance, although it may be useful for general purposes.*

LgstR [*iterations*,]*xList*,*yList*,*frequencyList*,*equationVariable*

Fits a logistic regression model ($y=a/(1+be^{cx})+d$) to real data pairs in *xList* and *yList* and frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**. The equation's coefficients always are stored as a list to built-in variable **PRegC**.

The number of *iterations* is optional. If omitted, 64 is the default. A large number of *iterations* may produce more accurate results but may require longer calculation times. A smaller number may produce less accurate results but with shorter calculation times.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

LgstR [*iterations*,]*xList*,*yList*,*equationVariable*

Uses frequencies of 1.

LgstR [*iterations*,]*xList*,*yList*,*frequencyList*

Stores the regression equation to **RegEq** only.

LgstR [*iterations*,]*xList*,*yList*

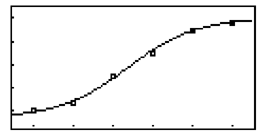
Uses frequencies of 1, and stores the regression equation to **RegEq** only.

In **Func** graphing mode:

```
{1,2,3,4,5,6}→L1 [ENTER]
                        {1 2 3 4 5 6}
{1,1.3,2.5,3.5,4.5,4.8}→L2 [ENTER]
                {1 1.3 2.5 3.5 4.5 4…
LgstR L1,L2,y1 [ENTER]
```

```
LogisticReg
 y=a/(1+be^(cx))+d
 n=6
 tolMet=1
 PRegC=
{4.31285605279 51.75…
```

```
Plot1(1,L1,L2) [ENTER]            Done
ZData [ENTER]
```



LgstR [*iterations*,]*equationVariable*

Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

LgstR [*iterations*]

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

Line(

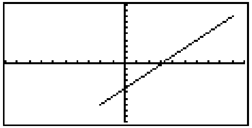
† GRAPH DRAW menu

Line($x1$,$y1$,$x2$,$y2$**)**

Draws a line from point ($x1$,$y1$) to ($x2$,$y2$).

In **Func** graphing mode and a **ZStd** graph screen:

`Line(-2,-7,9,8)` [ENTER]



LinR

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

LinR *xList*,*yList*,*frequencyList*,*equationVariable*

Fits a linear regression model (y=a+bx) to real data pairs in *xList* and *yList* and frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

LinR *xList*,*yList*,*equationVariable*

Uses frequencies of 1.

LinR *xList*,*yList*,*frequencyList*

Stores the regression equation to **RegEq** only.

LinR *xList*,*yList*

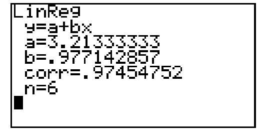
Uses frequencies of 1, and stores the regression equation to **RegEq** only.

LinR *equationVariable*

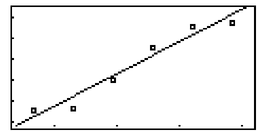
Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

In **Func** graphing mode:

```
{1,2,3,4,5,6}→L1 [ENTER]
                          {1 2 3 4 5 6}
{4.5,4.6,6,7.5,8.5,8.7}→L2 [ENTER]
                    {4.5 4.6 6 7.5 8.5 8.7}
LinR L1,L2,y1 [ENTER]
```




```
Plot1(1,L1,L2) [ENTER]                 Done
ZData [ENTER]
```



LinR

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

List entry: { }

LIST menu

{*element1*,*element2*, ...**}**

Defines a list in which each element is a real or complex number or variable.

```
{1,2,3}→L1 [ENTER]                    {1 2 3}
```

In **RectC** complex number mode:

```
{3,(2,4),8*2}→L2 [ENTER]
                    {(3,0) (2,4) (16,0)}
```

---

li▶vc

LIST OPS menu

VECTR OPS menu

li▶vc *list*

Returns a vector converted from a real or complex *list*.

```
li▶vc {2,7,-8,0} [ENTER]
                              [2 7 -8 0]
```

---

ln

[LN]

ln *number* or **ln (***expression***)**

Returns the natural logarithm of a real or complex *number* or *expression*.

ln *list*

Returns a list in which each element is the natural logarithm of the corresponding element in *list*.

```
ln 2 [ENTER]                 .69314718056
ln (36.4/3) [ENTER]         2.49595648597
```

In **RectC** complex number mode:

```
ln -3 [ENTER]      (1.09861228867,3.141…
ln {2,3} [ENTER]
                   {.69314718056 1.0986…
```

---

lngth

STRNG menu

lngth *string*

Returns the length (number of characters) of *string*. The character count includes spaces but not quotation marks.

```
lngth "The answer is:" [ENTER]         14
"The answer is:"→STR [ENTER]
                         The answer is:
lngth STR [ENTER]                      14
```

LnR

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

LnR *xList*,*yList*,*frequencyList*,*equationVariable*

Fits a logarithmic regression model (y=a+b ln x) to the real data pairs in *xList* and *yList* (**x** values must be > 0) and frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

LnR *xList*,*yList*,*equationVariable*

Uses frequencies of 1.

LnR *xList*,*yList*,*frequencyList*

Stores the regression equation to **RegEq** only.

LnR *xList*,*yList*

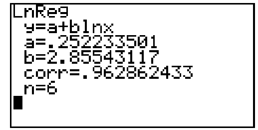
Uses frequencies of 1, and stores the regression equation to **RegEq** only.

LnR *equationVariable*

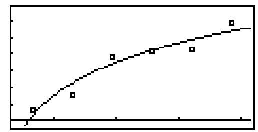
Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

In **Func** graphing mode:

```
{1,2,3,4,5,6}→L1 [ENTER]
                         {1 2 3 4 5 6}
{.6,1.5,3.8,4.2,4.3,5.9}→L2 [ENTER]
              {.6 1.5 3.8 4.2 4.3 5.9}
LnR L1,L2,y1 [ENTER]
```




```
Plot1(1,L1,L2) [ENTER]              Done
ZData [ENTER]
```



LnR

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

log

[LOG]

log *number* or **log (***expression***)**

Returns the logarithm of a real or complex *number* or *expression*, where:

$10^{logarithm} = number$

```
log 2 [ENTER]                .301029995664
log (36.4/3) [ENTER]         1.08398012893
```

In **RectC** complex number mode:

```
log (3,4) [ENTER]
              (.698970004336,.4027…
```

log *list*

Returns a list in which each element is the logarithm of the corresponding element in *list*.

In **RectC** complex number mode:

```
log {⁻3,2} [ENTER]
              {(.47712125472,1.364…
```

LU(

MATRX MATH menu

LU(*matrix*,*lMatrixName*, *uMatrixName*, *pMatrixName***)**

Calculates the Crout LU (lower-upper) decomposition of a real or complex *matrix*. The lower triangular matrix is stored in *lMatrixName*, the upper triangular matrix in *uMatrixName*, and the permutation matrix (which describes the row swaps done during the calculation) in *pMatrixName*.

lMatrixName **∗** *uMatrixName* = *pMatrixName* **∗** *matrix*

```
[[6,12,18][5,14,31][3,8,18]]
→MAT [ENTER]                 [[6 12 18]
                              [5 14 31]
                              [3 8  18]]
LU(MAT,L,U,P) [ENTER]              Done
L [ENTER]                      [[6 0 0]
                                [5 4 0]
                                [3 2 1]]
U [ENTER]                      [[1 2 3]
                                [0 1 4]
                                [0 0 1]]
P [ENTER]                      [[1 0 0]
                                [0 1 0]
                                [0 0 1]]
```

Matrix entry: [ ]

[2nd] [[] and [2nd] []]

[[*row1***][***row2***] ...]**

Defines a matrix entered row-by-row in which each element is a real or complex number or variable.

Enter each **[***row***]** as **[***element*,*element*, ... **]**.

```
[[1,2,3][4,5,6]]→MAT [ENTER]
                        [[1 2 3]
                         [4 5 6]]
```

max(

MATH NUM menu

max(*numberA*,*numberB***)**

Returns the larger of two real or complex numbers.

```
max(2.3,1.4) [ENTER]                2.3
```

max(*list***)**

Returns the largest element in *list*.

```
max({1,9,π/2,e^2}) [ENTER]            9
```

max(*listA*,*listB***)**

Returns a list in which each element is the larger of the corresponding elements in *listA* and *listB*.

```
max({1,10},{2,9}) [ENTER]        {2 10}
```

MBox

† STAT DRAW menu

MBox *xList*,*frequencyList*

Draws a modified box plot on the current graph, using the real data in *xList* and the frequencies in *frequencyList*.

MBox *xList*

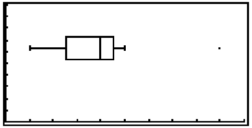
Uses frequencies of 1.

MBox

Uses the data in built-in variables **xStat** and **fStat**. These variables must contain valid data of the same dimension; otherwise, an error occurs.

Starting with a **ZStd** graph screen:

```
{1,2,3,4,5,9}→XL [ENTER]
                  {1 2 3 4 5 9}
{1,1,1,4,1,1}→FL [ENTER]
                  {1 1 1 4 1 1}
0→xMin:0→yMin [ENTER]             0
MBox XL,FL [ENTER]
```



Menu(

‡ program editor
CTL menu

Menu(*item#***,"***title1***",***label1*[**,** … **,***item#***,"***title15***",***label15*]**)**

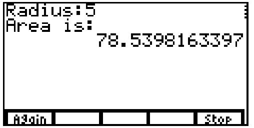
Generates a menu of up to 15 items during program execution. Menus are displayed as three groups of five items. For each item:

- *item#* — integer from 1 through 15 that identifies this item's position in the menu.
- **"***title***"** — text string that will be displayed for this item on the menu. Typically, use from 1 through 5 characters; additional characters may not be seen on the menu.
- *label* — valid label to which program execution will branch when the user selects this item.

Program segment:

```
 :
:Lbl A
:Input "Radius:",RADIUS
:Disp "Area is:",π*RADIUS²
:Menu(1,"Again",A,5,"Stop",B)
:Lbl B
:Disp "The End"
```

Example when executed:



min(

MATH NUM menu

min(*numberA***,***numberB***)**

Returns the smaller of two real or complex numbers.

```
min(3,-5) [ENTER]                 -5
min(-5.2,-5.3) [ENTER]          -5.3
min(5,2+2) [ENTER]                 4
```

min(*list***)**

Returns the smallest element in *list*.

```
min({1,3,-5}) [ENTER]             -5
```

min(*listA***,***listB***)**

Returns a list in which each element is the smaller of the corresponding elements in *listA* and *listB*.

```
min({1,2,3},{3,2,1}) [ENTER]
                             {1 2 1}
```

mod(

MATH NUM menu

mod(*numberA***,***numberB***)**

Returns *numberA* modulo *numberB*. The arguments must be real.

```
mod(7,0) [ENTER]                   7
mod(7,3) [ENTER]                   1
mod(-7,3) [ENTER]                  2
mod(7,-3) [ENTER]                 -2
mod(-7,-3) [ENTER]                -1
```

mRAdd(

MATRX OPS menu

mRAdd(*number*,*matrix*,*rowA*,*rowB***)**

Returns the result of a "multiply and add row" matrix operation, where:

a. *rowA* of a real or complex *matrix* is multiplied by a real or complex *number*.

b. The results are added to (and then stored in) *rowB*.

```
[[5,3,1][2,0,4][3,-1,2]]→MAT
[ENTER]                   [[5 3  1]
                           [2 0  4]
                           [3 -1 2]]

mRAdd(5,MAT,2,3) [ENTER]
                        [[5  3  1 ]
                         [2  0  4 ]
                         [13 -1 22]]
```

Multiplication: *

[×]

numberA * *numberB*

Returns the product of two real or complex numbers.

```
2*5 [ENTER]                     10
```

number * *list* or *list* * *number*
number * *matrix* or *matrix* * *number*
number * *vector* or *vector* * *number*

Returns a list, matrix, or vector in which each element is *number* multiplied by the corresponding element in *list*, *matrix*, or *vector*.

```
4*{10,9,8} [ENTER]     {40 36 32}
```

In **RectC** complex number mode:

```
[8,1,(5,2)]*3 [ENTER]
           [(24,0) (3,0) (15,6)]
```

listA * *listB*

Returns a list in which each element of *listA* is multiplied by the corresponding element of *listB*. The lists must have the same dimension.

```
{1,2,3}*{4,5,6} [ENTER]    {4 10 18}
```

matrix * *vector*

Returns a vector in which *matrix* is multiplied by *vector*. The number of columns in *matrix* must equal the number of elements in *vector*.

```
[[1,2,3][4,5,6]]→MAT [ENTER]
                       [[1 2 3]
                        [4 5 6]]
MAT*[7,8,9] [ENTER]
                       [50 122]
```

matrixA ✱ *matrixB*

Returns a matrix in which *matrixA* is multiplied by *matrixB*. The number of columns in *matrixA* must equal the number of rows in *matrixB*.

```
[[2,2][3,4]]→MATA [ENTER]        [[2 2]
                                  [3 4]]
[[1,2,3][4,5,6]]→MATB [ENTER]
                               [[1 2 3]
                                [4 5 6]]
MATA*MATB [ENTER]          [[10 14 18]
                            [19 26 33]]
```

multR(

MATRX OPS menu

multR(*number*,*matrix*,*row***)**

Returns the result of a "row multiplication" matrix operation, where:

a. The specified *row* of a real or complex *matrix* is multiplied by a real or complex *number*.

b. The results are stored in the same *row*.

```
[[5,3,1][2,0,4][3,-1,2]]→MAT
[ENTER]                         [[5 3  1]
                                 [2 0  4]
                                 [3 -1 2]]
multR(5,MAT,2) [ENTER]
                            [[5   3  1 ]
                             [10  0  20]
                             [3  -1  2 ]]
```

nCr

MATH PROB menu

items **nCr** *number*

Returns the number of combinations of *items* (**n**) taken *number* (**r**) at a time. Both arguments must be real nonnegative integers.

```
5 nCr 2 [ENTER]                      10
```

nDer(

CALC menu

To view or set the value for δ, press [2nd] [MEM] [F4] to display the tolerance screen.

nDer(*expression*,*variable*,*value***)**

Returns an approximate numerical derivative of *expression* with respect to *variable* evaluated at a real or complex *value*. The approximate numerical derivative is the slope of the secant line through the points:

(*value*−δ,f(*value*−δ)) and (*value*+δ,f(*value*+δ))

As the step value δ gets smaller, the approximation usually gets more accurate.

For δ=.001:

```
nDer(x^3,x,5) [ENTER]          75.000001
```

For δ=1E-4:

```
nDer(x^3,x,5) [ENTER]          75
```

nDer(*expression*,*variable***)**

Uses the current value of *variable*.

```
5→x [ENTER]                    5
nDer(x^3,x) [ENTER]            75
```

Negation: ‑

[(-)]

‑*number* or ‑**(***expression***)**
‑*list*
‑*matrix*
‑*vector*

Returns the negative of the real or complex argument.

```
‑2+5 [ENTER]                   3
‑(2+5) [ENTER]                 ‑7
‑{0,‑5,5} [ENTER]              {0 5 ‑5}
```

norm

MATRX MATH menu

VECTR MATH menu

norm *matrix*

Returns the Frobenius norm of a real or complex *matrix*, calculated as:

$$\sqrt{\Sigma(real^2+imaginary^2)}$$

where the sum is over all elements.

```
[[1,‑2][‑3,4]]→MAT [ENTER]
                               [[1 ‑2]
                                [‑3 4 ]]
norm MAT [ENTER]               5.47722557505
```

norm *vector*

Returns the length of a real or complex *vector*, where:

norm [a,b,c] returns $\sqrt{a^2+b^2+c^2}$.

```
norm [3,4,5] [ENTER]
                    7.07106781187
```

norm *number* or **norm (***expression***)**
norm *list*

Returns the absolute value of a real or complex *number* or *expression*, or of each element in *list*.

```
norm ⁻25 [ENTER]                 25
```

In **Radian** angle mode:

```
norm {-25,cos ⁻(π/3)} [ENTER]
                          {25 .5}
```

Normal

† mode screen

Normal

Sets normal notation mode.

In **Eng** notation mode:

```
123456789 [ENTER]      123.456789E6
```

In **Sci** notation mode:

```
123456789 [ENTER]      1.23456789E8
```

In **Normal** notation mode:

```
123456789 [ENTER]         123456789
```

not

BASE BOOL menu

not *integer*

Returns the one's complement of a real *integer*. Internally, *integer* is represented as a 16-bit binary number. The value of each bit is flipped (0 becomes 1, and vice versa) for the one's complement.

For example, **not** 78:

78 = 0000000001001110**b**
1111111110110001**b** (one's complement)

Sign bit; 1 indicates a negative number

To find the magnitude of a negative binary number, determine its two's complement (take the one's complement and then add 1). For example:

1111111110110001**b** = one's complement of 78
0000000001001110**b** (one's complement)
\+ 0000000000000001**b**
0000000001001111**b** = 79 (two's complement)

Therefore, **not** 78 = ⁻79.

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.

In **Dec** number base mode:

```
not 78 [ENTER]                    ⁻79
```

In **Bin** number base mode:

```
not 1001110 [ENTER]
                     1111111110110001b
Ans▶Dec [ENTER]                  ⁻79d
```

Not equal to: ≠

TEST menu

numberA ≠ *numberB*
matrixA ≠ *matrixB*
vectorA ≠ *vectorB*
stringA ≠ *stringB*

Tests whether the condition *argumentA* ≠ *argumentB* is true or false. Numbers, matrices, and vectors can be real or complex. If complex, the magnitude (modulus) of each element is compared. Strings are case-sensitive.

- If true (*argumentA* ≠ *argumentB*), returns **1**.
- If false (*argumentA* = *argumentB*), returns **0**.

```
2+2≠3+2 [ENTER]                1
2+(2≠3)+2 [ENTER]              5
[1,2]≠[3-2,-1+3] [ENTER]       0
"A"≠"a" [ENTER]                1
```

listA ≠ *listB*

Returns a list of **1**s and/or **0**s to indicate if each element in *listA* is ≠ the corresponding element in *listB*.

```
{1,5,9}≠{1,-6,9} [ENTER]       {0 1 0}
```

nPr

MATH PROB menu

items **nPr** *number*

Returns the number of permutations of *items* (**n**) taken *number* (**r**) at a time. Both arguments must be real nonnegative integers.

```
5 nPr 2 [ENTER]                20
```

o

BASE TYPE menu

integer **o**

Designates a real *integer* as octal, regardless of the number base mode setting.

In **Dec** number base mode:

```
10o [ENTER]                    8
10o+10 [ENTER]                 18
```

Oct

† mode screen

Oct

Sets octal number base mode. Results are displayed with the o suffix. In any number base mode, you can designate an appropriate value as binary, decimal, hexadecimal, or octal by using the b, d, h, or o designator, respectively, from the BASE TYPE menu.

In **Oct** number base mode:

`10+10b+Fh+10d` [ENTER] `43o`

▸Oct

BASE CONV menu

number▸**Oct**
list▸**Oct**
matrix▸**Oct**
vector▸**Oct**

Returns the octal equivalent of the real or complex argument.

In **Dec** number base mode:

`2*8` [ENTER] `16`
`Ans▸Oct` [ENTER] `20o`

`{7,8,9,10}▸Oct` [ENTER]
`{7o 10o 11o 12o}`

OneVar

STAT CALC menu
(OneVa shows on menu)

OneVar *xList*,*frequencyList*

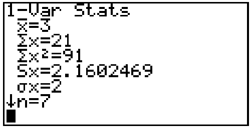
Performs one-variable statistical analysis using real data points in *xList* and frequencies in *frequencyList*.

The values used for *xList* and *frequencyList* are stored automatically to built-in variables **xStat** and **fStat**, respectively.

OneVar *xList*

Uses frequencies of 1.

`{0,1,2,3,4,5,6}→XL` [ENTER]
`{0 1 2 3 4 5 6}`
`OneVar XL` [ENTER]



Scroll down to see more results.

OneVar

Uses **xStat** and **fStat** for *xList* and *frequencyList*. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

---

or

BASE BOOL menu

integerA **or** *integerB*

Compares two real integers bit by bit. Internally, both integers are converted to binary. When corresponding bits are compared, the result is 1 if either bit is 1; the result is 0 only if both bits are 0. The returned value is the sum of the bit results.

For example, 78 **or** 23 = 95.

78 = 1001110**b**
23 = 0010111**b**
1011111**b** = 95

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.

In **Dec** number base mode:

`78 or 23` [ENTER] 95

In **Bin** number base mode:

`1001110 or 10111` [ENTER] 1011111**b**

`Ans▶Dec` [ENTER] 95**d**

Outpt(

‡ program editor
I/O menu

Outpt(*row*,*column*,*string***)**

Displays *string* beginning at *row* and *column*, where $1 \le row \le 8$ and $1 \le column \le 21$.

Outpt(*row*,*column*,*value***)**

Displays *value* beginning at the specified *row* and *column*.

Outpt("CBLSEND",*listName***)**

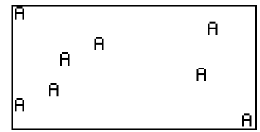
Sends the contents of *listName* to the CBL or CBR System.

You can send data also by using **Send(** as described on page 350.

Program segment:

```
  :
:ClLCD
:For(i,1,8)
: Outpt(i,randInt(1,21),"A")
:End
  :
```

Example result after execution:



P2Reg

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

P2Reg *xList*,*yList*,*frequencyList*,*equationVariable*

Performs a second order polynomial regression using real data pairs in *xList* and *yList* and frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**. The equation's coefficients always are stored as a list to built-in variable **PRegC**.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

P2Reg *xList*,*yList*,*equationVariable*

Uses frequencies of 1.

P2Reg *xList*,*yList*,*frequencyList*

Stores the regression equation to **RegEq** only.

P2Reg *xList*,*yList*

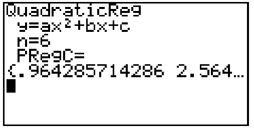
Uses frequencies of 1, and stores the regression equation to **RegEq** only.

P2Reg *equationVariable*

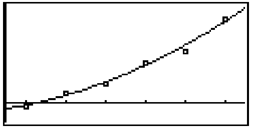
Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

In **Func** graphing mode:

```
{1,2,3,4,5,6}→L1 [ENTER]
                        {1 2 3 4 5 6}
{-2,6,11,23,29,47}→L2 [ENTER]
                   {-2 6 11 23 29 47}
P2Reg L1,L2,y1 [ENTER]
```




```
Plot1(1,L1,L2) [ENTER]           Done
ZData [ENTER]
```



P2Reg

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

P3Reg

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

P3Reg *xList*,*yList*,*frequencyList*,*equationVariable*

Performs a third order polynomial regression using real data pairs in *xList* and *yList* and frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**. The equation's coefficients always are stored as a list to built-in variable **PRegC**.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

P3Reg *xList*,*yList*,*equationVariable*

Uses frequencies of 1.

P3Reg *xList*,*yList*,*frequencyList*

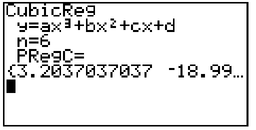
Stores the regression equation to **RegEq** only.

P3Reg *xList*,*yList*

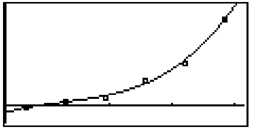
Uses frequencies of 1, and stores the regression equation to **RegEq** only.

In **Func** graphing mode:

```
{1,2,3,4,5,6}→L1 [ENTER]
                          {1 2 3 4 5 6}
{⁻6,15,27,88,145,294}→L2 [ENTER]
                  {⁻6 15 27 88 145 294}
P3Reg L1,L2,y1 [ENTER]
```




```
Plot1(1,L1,L2) [ENTER]                Done
ZData [ENTER]
```



P3Reg *equationVariable*

Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

P3Reg

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

P4Reg

STAT CALC menu

Built-in equation variables such as ***y1****,* ***r1****, and* ***xt1*** *are case-sensitive. Do not use* ***Y1****,* ***R1****, and* ***XT1****.*

P4Reg *xList*,*yList*,*frequencyList*,*equationVariable*

Performs a fourth order polynomial regression using real data pairs in *xList* and *yList* and frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**. The equation's coefficients always are stored as a list to built-in variable **PRegC**.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

P4Reg *xList*,*yList*,*equationVariable*

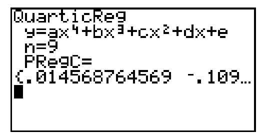
Uses frequencies of 1.

P4Reg *xList*,*yList*,*frequencyList*

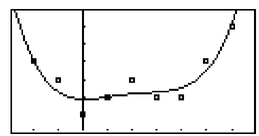
Stores the regression equation to **RegEq** only.

In **Func** graphing mode:

```
{-2,-1,0,1,2,3,4,5,6}→L1 [ENTER]
                  {-2 -1 0 1 2 3 4 5 6}
{4,3,1,2,3,2,2,4,6}→L2 [ENTER]
                    {4 3 1 2 3 2 2 4 6}
P4Reg L1,L2,y1 [ENTER]
```




```
Plot1(1,L1,L2) [ENTER]                Done
ZData [ENTER]
```



P4Reg *xList*,*yList*

Uses frequencies of 1, and stores the regression equation to **RegEq** only.

P4Reg *equationVariable*

Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

P4Reg

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

Param

† mode screen

Param

Sets parametric graphing mode.

Pause

‡ program editor
CTL menu

Pause *string*
Pause *value*
Pause *list*
Pause *matrix*
Pause *vector*

Displays the specified argument and then suspends program execution until the user presses [ENTER].

Program segment:

```
  :
:Input "Enter x:",x
:y1=x²-6
:Disp "y1 is:",y1
:Pause "Press ENTER to graph"
:ZStd
  :
```

Pause

Suspends program execution until the user presses [ENTER].

Percent: %

MATH MISC menu

number% or (*expression*)%

Returns a real *number* or *expression* divided by 100.

```
5% [ENTER]                .05
5%*200 [ENTER]             10
(10+5)%*200 [ENTER]        30
```

---

pEval(

MATH MISC menu

pEval(*coefficientList*,*xValue***)**

Returns the value of a polynomial (whose coefficients are given in *coefficientList*) at *xValue*.

Evaluate $y=2x^2+2x+3$ at $x=5$:

```
pEval({2,2,3},5) [ENTER]     63
```

PlOff

STAT PLOT menu

PlOff [1,2,3]

Deselects the specified stat plot numbers.

```
PlOff 1,3 [ENTER]          Done
```

PlOff

Deselects all stat plot numbers.

```
PlOff [ENTER]              Done
```

PlOn

STAT PLOT menu

PlOn [1,2,3]

Selects the specified stat plot numbers, in addition to any plot numbers that are already selected.

```
PlOn 2,3 [ENTER]           Done
```

PlOn

Selects all stat plot numbers.

```
PlOn [ENTER]               Done
```

Plot1(
Plot2(
Plot3(

† STAT PLOT menu

The syntax and descriptions to the right refer to **Plot1(**, *but they apply as well to* **Plot2(** *and* **Plot3(**.

Scatter plot

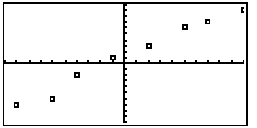
Plot1(1,*xListName***,***yListName***,***mark***)**
Plot1(1,*xListName***,***yListName***)**

Defines and selects the plot using real data pairs in *xListName* and *yListName*.

The optional *mark* specifies the character used to plot the points. If you omit *mark*, a box is used.

mark: **1** = box (□) **2** = cross (+) **3** = dot (•)

```
{-9,-6,-4,-1,2,5,7,10}→L1 [ENTER]
                  {-9 -6 -4 -1 2 5 7 1…
{-7,-6,-2,1,3,6,7,9}→L2 [ENTER]
                   {-7 -6 -2 1 3 6 7 9}
Plot1(1,L1,L2) [ENTER]              Done
ZStd [ENTER]
```



xyLine plot

Plot1(2,*xListName***,***yListName***,***mark***)**
Plot1(2,*xListName***,***yListName***)**

Modified box plot

Plot1(3,*xListName***,1** *or frequencyListName***,***mark***)**
Plot1(3,*xListName***,1** *or frequencyListName***)**
Plot1(3,*xListName***)**

Defines and selects the plot using real data points in *xListName* with the specified frequencies. If you omit **1** *or frequencyListName*, frequencies of 1 are used.

Histogram

Plot1(4,*xListName***,1** *or frequencyListName***)**
Plot1(4,*xListName***)**

Box plot

Plot1(5,*xListName***,1** *or frequencyListName***)**
Plot1(5,*xListName***)**

Pol

† mode screen

Pol

Sets polar graphing mode.

▶Pol

CPLX menu

complexNumber▶**Pol**

Displays *complexNumber* in polar form (*magnitude∠angle*), regardless of the complex number mode.

list▶**Pol**
matrix▶**Pol**
vector▶**Pol**

Returns a list, matrix, or vector in which each element of the argument is displayed in polar form.

In **RectC** complex number mode:

```
√-2 [ENTER]          (0,1.41421356237)
Ans▶Pol [ENTER]
               (1.41421356237∠1.570…
{1,√-2} [ENTER]
               {(1,0) (0,1.141421356…
Ans▶Pol [ENTER]
               {(1∠0) (1.4142135623…
```

PolarC

† mode screen

PolarC

Sets polar complex number mode (*magnitude∠angle*).

In **PolarC** complex number mode:

```
√-2 [ENTER]     (1.41421356237∠1.570…
```

Polar complex: ∠

[2nd] [∠]

magnitude∠angle

Used to enter complex numbers in polar form. The *angle* is interpreted according to the current angle mode.

In **Radian** angle mode and **PolarC** complex number mode:

```
(1,2)+(3∠π/4) [ENTER]
               (5.16990542093∠.9226…
```

PolarGC

† graph format screen

PolarGC

Displays graph coordinates in polar form.

poly

† [2nd] [POLY]

poly *coefficientList*

Returns a list containing the real and complex roots of a polynomial whose coefficients are given in *coefficientList*.

$$a_n x^n + \ldots + a_2 x^2 + a_1 x^1 + a_0 x^0 = 0$$

Find the roots of $2x^3-8x^2-14x+20=0$:

```
poly {2,-8,-14,20} [ENTER]
                        {5 -2 1}
```

Power: ^

[^]

number^*power* or **(***expression***)^(***expression***)**

Returns *number* raised to *power*. The arguments can be real or complex.

```
4^2 [ENTER]                   16
2^-5 [ENTER]              .03125
```

listA^*listB*

Returns a list in which each element of *listA* is raised to the power specified by the corresponding element in *listB*.

```
{2,3,4}^{3,4,5} [ENTER]
                   {8 81 1024}
```

squareMatrix^*power*

Returns a matrix equivalent to *squareMatrix* multiplied by itself *power* number of times, where 0 ≤ *power* ≤ 255. This is not the same as simply raising each element to *power*.

```
[[2,3][4,5]]^3 [ENTER]
                   [[116 153]
                    [204 269]]
```

Power of 10: 10^

[2nd] [10^x]

10^*power* or **10^(***expression***)**

Returns 10 raised to *power* or *expression*, which can be real or complex.

```
10^1.5 [ENTER]      31.6227766017
10^-2 [ENTER]                 .01
```

10^*list*

Returns a list in which each element is 10 raised to the power specified by the corresponding element in *list*.

```
10^{1.5,-2} [ENTER]
                {31.6227766017 .01}
```

prod

LIST OPS menu

MATH MISC menu

prod *list*

Returns the product of all real or complex elements in *list*.

```
prod {1,2,4,8} [ENTER]                64
prod {2,7,-8} [ENTER]               -112
```

Prompt

‡ program editor
I/O menu
(Promp shows on menu)

Prompt *variableA*[**,***variableB***,** ...]

Prompts the user to enter a value for *variableA*, then *variableB*, and so on.

Program segment:

```
 :
:Prompt A,B,C
 :
```

PtChg(

† GRAPH DRAW menu

PtChg(*x*,*y***)**

Reverses the point at graph coordinates (x,y).

```
PtChg(-6,2)
```

PtOff(

† GRAPH DRAW menu

PtOff(*x*,*y***)**

Erases the point at graph coordinates (x,y).

```
PtOff(3,5)
```

PtOn(

† GRAPH DRAW menu

PtOn(*x*,*y***)**

Draws the point at graph coordinates (x,y).

```
PtOn(3,5)
```

PwrR

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

PwrR *xList*,*yList*,*frequencyList*,*equationVariable*

Fits a power regression model ($y=ax^b$) to positive real data pairs in *xList* and *yList*, using frequencies in *frequencyList*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to built-in variables **xStat**, **yStat**, and **fStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

PwrR *xList*,*yList*,*equationVariable*

Uses frequencies of 1.

PwrR *xList*,*yList*,*frequencyList*

Stores the regression equation to **RegEq** only.

PwrR *xList*,*yList*

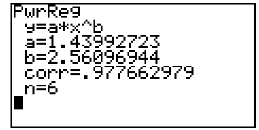
Uses frequencies of 1, and stores the regression equation to **RegEq** only.

PwrR *equationVariable*

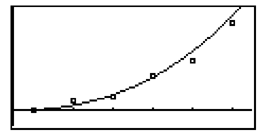
Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

In **Func** graphing mode:

```
{1,2,3,4,5,6}→L1 [ENTER]
                          {1 2 3 4 5 6}
{1,17,21,52,75,133}→L2 [ENTER]
                    {1 17 21 52 75 133}
PwrR L1,L2,y1 [ENTER]
```




```
Plot1(1,L1,L2) [ENTER]             Done
ZData [ENTER]
```



PwrR

Uses **xStat**, **yStat**, and **fStat**, and stores the regression equation to **RegEq** only.

PxChg(

GRAPH DRAW menu

PxChg(*row*,*column***)**

Reverses the pixel at (*row*, *column*), where $0 \le row \le 62$ and $0 \le column \le 126$.

```
PxChg(10,95)
```

PxOff(

GRAPH DRAW menu

PxOff(*row*,*column***)**

Erases the pixel at (*row*, *column*), where $0 \le row \le 62$ and $0 \le column \le 126$.

```
PxOff(10,95)
```

PxOn(

GRAPH DRAW menu

PxOn(*row*,*column***)**

Draws the pixel at (*row*, *column*), where $0 \le row \le 62$ and $0 \le column \le 126$.

```
PxOn(10,95)
```

PxTest(

GRAPH DRAW menu

PxTest(*row*,*column***)**

Returns **1** if the pixel at (*row*, *column*) is on, **0** if it is off; $0 \le row \le 62$ and $0 \le column \le 126$.

Assuming the pixel at (10,95) is already on:

```
PxTest(10,95) [ENTER]                 1
```

---

rAdd(

MATRX OPS menu

rAdd(*matrix*,*rowA*,*rowB***)**

Returns a matrix in which *rowA* of a real or complex *matrix* is added to (and stored in) *rowB*.

```
[[5,3,1][2,0,4][3,⁻1,2]]→MAT
[ENTER]                       [[5  3  1]
                               [2  0  4]
                               [3 ⁻1  2]]

rAdd(MAT,2,3) [ENTER]         [[5  3  1]
                               [2  0  4]
                               [5 ⁻1  6]]
```

Radian

† [2nd] [MODE]

Radian

Sets radian angle mode.

In **Radian** angle mode:

```
sin (π/2) [ENTER]                  1
sin 90 [ENTER]          .893996663601
```

Radian entry: r

MATH ANGLE menu

*number*r or **(***expression***)**r

Designates a real *number* or *expression* as radians, regardless of the angle mode setting.

*list*r

Designates each element in a real *list* as radians.

In **Degree** angle mode:

```
cos (π/2) [ENTER]       .999624216859
cos (π/2)r [ENTER]                 0

cos {π/2,π}r [ENTER]          {0 -1}
```

rand

MATH PROB menu

rand

Returns a random number between 0 and 1.

To control a random number sequence, first store an integer seed value to **rand** (such as **0→rand**).

You may have different results for the first two examples:

```
rand [ENTER]            .943597402492
rand [ENTER]            .146687829222

0→rand:rand [ENTER]     .943597402492
0→rand:rand [ENTER]     .943597402492
```

randBin(

MATH PROB menu
(randBi shows on menu)

randBin(*#ofTrials***,***probabilityOfSuccess***,***#ofSimulations***)**

Returns a list of random integers from a binomial distribution, where *#ofTrials* ≥ 1 and $0 \le$ *probabilityOfSuccess* ≤ 1. The *#ofSimulations* is an integer ≥ 1 that specifies the number of integers returned in the list.

A seed value stored to **rand** also affects **randBin(**.

```
1→rand:randBin(5,.2,3) [ENTER]
                             {0 3 2}
```

randBin(*#ofTrials***,***probabilityOfSuccess***)**

Returns a single random integer.

```
0→rand:randBin(5,.2) [ENTER]       1
```

randInt(

MATH PROB menu
(randIn shows on menu)

randInt(*lower*,*upper*,*#ofTrials***)**

Returns a list of random integers bound by the specified integers, *lower* ≤ integer ≤ *upper*. The *#ofTrials* is an integer ≥ 1 that specifies the number of integers returned in the list.

A seed value stored to **rand** also affects **randInt(**.

```
1→rand:randInt(1,10,3) [ENTER]
                       {8 9 3}
```

randInt(*lower*,*upper***)**

Returns a single random integer.

```
0→rand:randInt(1,10) [ENTER]    10
```

randM(

MATRX OPS menu

randM(*rows*,*columns***)**

Returns a *rows* × *columns* matrix filled with random one-digit integers (⁻9 to 9).

```
0→rand:randM(2,3) [ENTER]
                  [[4  ⁻2 0]
                   [⁻7 8  8]]
```

randNorm(

MATH PROB menu
(randN shows on menu)

randNorm(*mean*,*stdDeviation*,*#ofTrials***)**

Returns a list of random numbers from a normal distribution specified by *mean* and *stdDeviation*. The *#ofTrials* is an integer ≥ 1 that specifies how many numbers are returned. Each returned number could be any real number, but most will be within the interval:

[*mean*−3(*stdDeviation*), *mean*+3(*stdDeviation*)].

A seed value stored to **rand** also affects **randNorm(**.

```
1→rand:randNorm(0,1,3) [ENTER]
         {⁻.660585055265 ⁻1.0…
```

randNorm(*mean*,*stdDeviation***)**

Returns a single random number.

```
0→rand:randNorm(0,1) [ENTER]
              ⁻1.58570962271
```

RcGDB

† GRAPH menu

RcGDB *graphDataBaseName*

Restores all settings stored in *graphDataBaseName*. For a list of settings, refer to **StGDB** on page 361.

RcPic

† GRAPH menu

RcPic *pictureName*

Displays the current graph and adds the picture stored in *pictureName*.

real

CPLX menu

real (*complexNumber***)**

Returns the real part of *complexNumber*.

real (*real,imaginary***)** returns *real*.
real (*magnitude∠angle***)** returns *magnitude* ***** **cos** (*angle*).

In **Radian** angle mode:

```
real (3,4) [ENTER]                3
real (3∠4) [ENTER]    -1.96093086259
```

real *complexList*
real *complexMatrix*
real *complexVector*

Returns a list, matrix, or vector in which each element is the real part of the corresponding element in the argument.

In **Radian** angle mode:

```
real {-2,(3,4),(3∠4)} [ENTER]
              {-2 3 -1.96093086259}
```

►Rec

CPLX menu

complexNumber►**Rec**

Displays *complexNumber* in rectangular form (*real,imaginary*) regardless of the complex number mode.

In **PolarC** complex number mode:

```
√-2 [ENTER]       (1.41421356237∠1.570…
Ans►Rec [ENTER]       (0,1.41421356237)
```

complexList▶**Rec**
complexMatrix▶**Rec**
complexVector▶**Rec**

Returns a list, matrix, or vector in which each element of the argument is displayed in rectangular form.

In **PolarC** complex number mode:

```
[(3∠π/6),√-2] ENTER
                 [(3∠.523598775598) (…
Ans▶Rec ENTER
                 [(2.59807621135,1.5)…
```

RectC

† mode screen

RectC

Sets rectangular complex number mode (*real*,*imaginary*).

In **RectC** complex number mode:

```
√-2 ENTER            (0,1.41421356237)
```

RectGC

† graph format screen

RectGC

Displays graph coordinates in rectangular form.

RectV

† mode screen

RectV

Sets rectangular vector coordinate mode **[x y z]**.

In **RectV** vector coordinate mode:

```
3*[4∠5] ENTER
                 [3.40394622556 ⁻11.5…
```

ref

MATRX OPS menu

ref *matrix*

Returns the row-echelon form of a real or complex *matrix*. The number of columns must be greater than or equal to the number of rows.

```
[[4,5,6][7,8,9]]→MAT ENTER
                              [[4 5 6]
                               [7 8 9]]
ref MAT ENTER
                 [[1 1.14285714286 1.…
                  [0 1             2 …
```

Repeat

‡ program editor
CTL menu
(Repea shows
on menu)

:Repeat *condition*
:*commands-to-repeat*
:End
:*commands*

Executes *commands-to-repeat* until *condition* is true.

Program segment:

```
  :
:6→N
:1→Fact
:Repeat N<1
: Fact*N→Fact
: N-1→N
:End
:Disp "6!=",Fact
  :
```

Return

‡ program editor
CTL menu
(Retur shows
on menu)

Return

In a subroutine, exits the subroutine and returns to the calling program. In the main program, stops execution and returns to the home screen.

Program segment in the calling program:

```
  :
:Input "Diameter:",DIAM
:Input "Height:",HT
:AREACIRC
:VOL=AREA*HT
:Disp "Volume =",VOL
  :
```

AREACIRC subroutine program:

```
PROGRAM:AREACIRC
:RADIUS=DIAM/2
:AREA=π*RADIUS²
:Return
```

RK

† graph format screen
(scroll down to
second screen)

RK

In **DifEq** graphing mode, uses an algorithm based on the Runge-Kutta method to solve differential equations. Typically, **RK** is more accurate than **Euler** but takes longer to find the solutions.

rnorm

MATRX MATH menu

rnorm *matrix*

Returns the row norm of a real or complex *matrix*. For each row, **rnorm** sums the absolute values (magnitudes of complex elements) of all elements on that row. The returned value is the largest of the sums.

```
[[-5,6,-7][3,3,9][9,-9,-7]]
→MAT [ENTER]                [[-5 6  -7]
                             [3  3  9 ]
                             [9  -9 -7]]
rnorm MAT [ENTER]                    25
```

rnorm *vector*

Returns the largest absolute value (or magnitude) in a real or complex *vector*.

```
rnorm [15,-18,7] [ENTER]             18
```

Root: $^x\sqrt{\ }$

MATH MISC menu

$x^{th}root\ {}^{x}\sqrt{number}$ or $x^{th}root\ {}^{x}\sqrt{}$**(***expression***)**

Returns the $x^{th}root$ of *number* or *expression*. The arguments can be real or complex.

```
5ˣ√32 [ENTER]                         2
```

$x^{th}root\ {}^{x}\sqrt{list}$

Returns a list in which each element is the $x^{th}root$ of the corresponding element in *list*.

```
5ˣ√{32,243} [ENTER]               {2 3}
```

$x^{th}rootList\ {}^{x}\sqrt{list}$

Returns a list in which each element is the root specified by the corresponding elements in $x^{th}rootList$ and *list*.

```
{5,2}ˣ√{32,25) [ENTER]            {2 5}
```

rotL

BASE BIT menu

rotL *integer*

Returns a real *integer* with bits rotated one to the left. Internally, *integer* is represented as a 16-bit binary number. When the bits are rotated left, the leftmost bit rotates to the rightmost bit.

rotL 0000111100001111b = 0001111000011110b

rotL is not valid in **Dec** number base mode. To enter hexadecimal numbers A through **F**, use the BASE A-F menu. Do not use [ALPHA] to type a letter.

In **Bin** number base mode:

```
rotL 0000111100001111 [ENTER]
                    1111000011110b
```

Leading zeros are not displayed.

rotR

BASE BIT menu

rotR *integer*

Returns a real *integer* with bits rotated one to the right. Internally, *integer* is represented as a 16-bit binary number. When the bits are rotated right, the rightmost bit rotates to the leftmost bit.

rotR 0000111100001111b = 1000011110000111b

rotR is not valid in **Dec** number base mode. To enter hexadecimal numbers A through **F**, use the BASE A-F menu. Do not use [ALPHA] to type a letter.

In **Bin** number base mode:

```
rotR 0000111100001111 [ENTER]
                 1000011110000111b
```

round(

MATH NUM menu

round(*number*,*#ofDecimals***)**
round(*number***)**

Returns a real or complex *number* rounded to the specified *#ofDecimals* (0 to 11). If *#ofDecimals* is omitted, *number* is rounded to 12 decimal places.

```
round(π,4) [ENTER]                3.1416
round(π/4,4) [ENTER]               .7854
round(π/4) [ENTER]         .785398163397
```

round(*list*,*#ofDecimals***)**
round(*matrix*,*#ofDecimals***)**
round(*vector*,*#ofDecimals***)**

Returns a list, matrix, or vector in which each element is the rounded value of the corresponding element in the argument. *#ofDecimals* is optional.

```
round({π,√2,ln 2},3) [ENTER]
                   {3.142 1.414 .693}
round([[ln 5,ln 3][π,e^1]],2)
[ENTER]                   [[1.61 1.1 ]
                           [3.14 2.72]]
```

rref

MATRX OPS menu

rref *matrix*

Returns the reduced row-echelon form of a real or complex *matrix*. The number of columns must be greater than or equal to the number of rows.

```
[[4,5,6][7,8,9]]→MAT [ENTER]
                             [[4 5 6]
                              [7 8 9]]
rref MAT [ENTER]
                [[1 0 -.999999999999…
                 [0 1 2              …
```

rSwap(

MATRX OPS menu

rSwap(*matrix*,*rowA*,*rowB***)**

Returns a matrix with *rowA* of a real or complex *matrix* swapped with *rowB*.

```
[[5,3,1][2,0,4][3,-1,2]]→MAT
[ENTER]                  [[5 3  1]
                          [2 0  4]
                          [3 -1 2]]
rSwap(MAT,2,3) [ENTER]
                         [[5 3  1]
                          [3 -1 2]
                          [2 0  4]]
```

Scatter

† STAT DRAW menu (Scatte shows on menu)

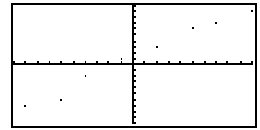
Scatter *xList*,*yList*

Draws a scatter plot on the current graph, using the real data pairs in *xList* and *yList*.

Scatter

Uses the data in built-in variables **xStat** and **yStat**. These variables must contain valid data of the same dimension; otherwise, an error occurs.

```
{⁻9,⁻6,⁻4,⁻1,2,5,7,10}→XL [ENTER]
                    {⁻9 ⁻6 ⁻4 ⁻1 2 5 7 1…
{⁻7,⁻6,⁻2,1,3,6,7,9}→YL [ENTER]
                     {⁻7 ⁻6 ⁻2 1 3 6 7 9}
ZStd:Scatter XL,YL [ENTER]
```



Sci

† mode screen

Sci

Sets scientific notation display mode.

In **Sci** notation mode:

```
123456789 [ENTER]          1.23456789E8
```

In **Normal** notation mode:

```
123456789 [ENTER]             123456789
```

Select(

LIST OPS menu

Select(*xListName*,*yListName***)**

If a scatter plot or xyline plot is currently selected and plotted on the graph screen, you can select a subset (range) of those data points. The selected data points are stored to *xListName* and *yListName*.

Select(*xListName*,*yListName***)** displays the current graph screen and starts an interactive session during which you select a range of data points.

a. Move the cursor to the leftmost (left bound) point of the range you want to select and press [ENTER].

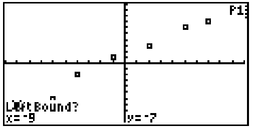
b. Then move the cursor to the rightmost (right bound) point of the range you want to select and press [ENTER].

A new stat plot of *xListName* and *yListName* replaces the plot from which you selected the points.

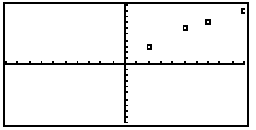
```
{⁻9,⁻6,⁻4,⁻1,2,5,7,10}→L1 [ENTER]
                {⁻9 ⁻6 ⁻4 ⁻1 2 5 7 1…
{⁻7,⁻6,⁻2,1,3,6,7,9}→L2 [ENTER]
                 {⁻7 ⁻6 ⁻2 1 3 6 7 9}
Plot1(1,L1,L2):ZStd [ENTER]
```

After the graph is displayed:

```
Select(L10,L20) [ENTER]
```



Move the cursor to point (2,3) and press [ENTER]. Then move to (10,9) and press [ENTER].



```
L10 [ENTER]                   {2 5 7 10}
L20 [ENTER]                    {3 6 7 9}
```

Send(

‡ program editor I/O menu

Send(*listName***)**

Sends the contents of *listName* to the CBL or CBR System.

```
{1,2,3,4,5}→L1:Send(L1) [ENTER]
                                Done
```

seq(

MATH MISC menu

seq(*expression*,*variable*,*begin*,*end*,*step***)**

Returns a list containing a sequence of numbers created by evaluating *expression* from *variable* = *begin* to *variable* = *end* in increments of *step*.

```
seq(x²,x,1,8,2) [ENTER]
                    {1 9 25 49}
```

seq(*expression*,*variable*,*begin*,*end***)**

Uses a *step* of 1.

```
seq(x²,x,1,8) [ENTER]
         {1 4 9 16 25 36 49 6…
```

SeqG

† graph format screen

SeqG

Sets sequential graphing format, in which selected functions are plotted one at a time.

SetLEdit

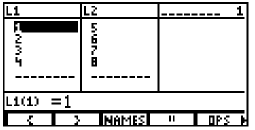
LIST OPS menu
(SetLE shows on menu)

SetLEdit *column1ListName*[**,** … **,***column20ListName*]

Removes all lists from the list editor and then stores one or more *ListNames* in the specified order, starting with column 1.

```
{1,2,3,4}→L1 [ENTER]          {1 2 3 4}
{5,6,7,8}→L2 [ENTER]          {5 6 7 8}
SetLEdit L1,L2 [ENTER]             Done
```

The list editor now contains:



SetLEdit

Removes all lists from the list editor and stores built-in lists **xStat**, **yStat**, and **fStat** in columns 1 through 3, respectively.

Shade(

GRAPH DRAW menu

Shade(*lowerFunc*,*upperFunc*,*xLeft*,*xRight*,*pattern*,*patternRes***)**

Draws *lowerFunc* and *upperFunc* in terms of **x** on the current graph and shades the area bounded by *lowerFunc*, *upperFunc*, *xLeft*, and *xRight*. The shading style is determined by *pattern* (**1** through **4**) and *patternRes* (**1** through **8**).

pattern:

1 = vertical (default)	**3** = negative-slope 45°
2 = horizontal	**4** = positive-slope 45°

patternRes (resolution):

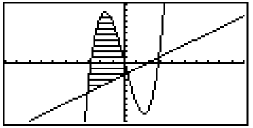
1 = every pixel (default)	5 = every 5th pixel
2 = every 2nd pixel	6 = every 6th pixel
3 = every 3rd pixel	7 = every 7th pixel
4 = every 4th pixel	8 = every 8th pixel

Shade(*lowerFunc*,*upperFunc***)**

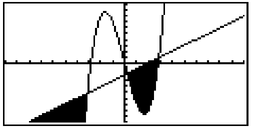
Sets *xLeft* and *xRight* to **xMin** and **xMax**, respectively, and uses the defaults for *pattern* and *patternRes*.

In **Func** graphing mode:

```
Shade(x-2,x^3-8 x,-5,1,2,3) [ENTER]
```



```
ClDrw:Shade(x^3-8 x,x-2) [ENTER]
```



shftL

BASE BIT menu

shftL *integer*

Returns a real *integer* with bits shifted one to the left. Internally, *integer* is represented as a 16-bit binary number. When the bits are shifted left, the leftmost bit is dropped and 0 is used as the rightmost bit.

shftL 0000111100001111b = 0001111000011110b

0

shftL is not valid in **Dec** number base mode. To enter hexadecimal numbers **A** through **F**, use the BASE A-F menu. Do not use [ALPHA] to type a letter.

In **Bin** number base mode:

```
shftL 0000111100001111 [ENTER]
                    1111000011110b
```

Leading zeros are not displayed.

shftR

BASE BIT menu

shftR *integer*

Returns a real *integer* with bits shifted one to the right. Internally, *integer* is represented as a 16-bit binary number. When the bits are shifted right, the rightmost bit is dropped and 0 is used as the leftmost bit.

shftR 0000111100001111b = 0000011110000111b

0

shftR is not valid in **Dec** number base mode. To enter hexadecimal numbers **A** through **F**, use the BASE A-F menu. Do not use [ALPHA] to type a letter.

In **Bin** number base mode:

```
shftR 0000111100001111 [ENTER]
                       11110000111b
```

Leading zeros are not displayed.

ShwSt

CATALOG

ShwSt

Displays the results of the most recent stat calculation.

sign

MATH NUM menu

sign *number* or **sign (***expression***)**

Returns ⁻**1** if the argument is < 0, **1** if it is > 0, or **0** if it is = 0. The argument must be real.

```
sign ⁻3.2 [ENTER]                    ⁻1
sign (6+2-8) [ENTER]                  0
```

sign *list*

Returns a list in which each element is ⁻1, 1, or 0 to indicate the sign of the corresponding element in *list*.

```
sign {⁻3.2,16.8,6+2-8} [ENTER]
                              {⁻1 1 0}
```

SimulG

† graph format screen

SimulG

Sets simultaneous graphing format, in which all selected functions are plotted at the same time.

simult(

† [2nd] [SIMULT]

simult(*squareMatrix*,*vector***)**

Returns a vector containing the solutions to a system of simultaneous linear equations that have the form:

$$a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 + ... = b_1$$
$$a_{2,1}x_1 + a_{2,2}x_2 + a_{2,3}x_3 + ... = b_2$$
$$a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 + ... = b_3$$

Each row in *squareMatrix* contains the **a** coefficients of an equation, and *vector* contains the **b** constants.

Solve the following for x and y:

```
3x - 4y = 7
 x + 6y = 6
```

```
[[3,⁻4][1,6]]→MAT [ENTER]      [[3 ⁻4]
                                [1 6 ]]
[7,6]→VEC [ENTER]               [7 6]
simult(MAT,VEC) [ENTER]         [3 .5]
```

The solution is x=3 and y=.5.

sin

[SIN]

sin *angle* or **sin (***expression***)**

Returns the sine of *angle* or *expression*, which can be real or complex.

An angle is interpreted as degrees or radians according to the current angle mode. In any angle mode, you can designate an angle as degrees or radians by using the ° or ^{r} designator, respectively, from the MATH ANGLE menu.

In **Radian** angle mode:

```
sin π/2 [ENTER]                    0
sin (π/2) [ENTER]                  1
sin 45° [ENTER]        .707106781187
```

In **Degree** angle mode:

```
sin 45 [ENTER]         .707106781187
sin (π/2)ʳ [ENTER]                 1
```

sin *list*

Returns a list in which each element is the sine of the corresponding element in *list*.

In **Radian** angle mode:

```
sin {0,π/2,π} [ENTER]        {0 1 0}
```

In **Degree** angle mode:

```
sin {0,30,90} [ENTER]       {0 .5 1}
```

sin *squareMatrix*

The squareMatrix cannot have repeated eigenvalues.

Returns a square matrix that is the matrix sine of *squareMatrix*. The matrix sine corresponds to the result calculated using power series or Cayley-Hamilton Theorem techniques. This is *not* the same as simply calculating the sine of each element.

$\sin^{-1}$

[2nd] [SIN⁻¹]

$\sin^{-1}$ *number* or **$\sin^{-1}$ (***expression***)**

Returns the arcsine of *number* or *expression*, which can be real or complex.

In **Radian** angle mode:

```
sin⁻¹ .5 [ENTER]       .523598775598
sin⁻¹ {0,.5} [ENTER]
                  {0 .523598775598}
```

In **Degree** angle mode:

```
sin⁻¹ 1 [ENTER]                   90
```

$\sin^{-1}$ *list*

Returns a list in which each element is the arcsine of the corresponding element in *list*.

sinh

MATH HYP menu

sinh *number* or **sinh (***expression***)**

Returns the hyperbolic sine of *number* or *expression*, which can be real or complex.

```
sinh 1.2 [ENTER]          1.50946135541
```

sinh *list*

Returns a list in which each element is the hyperbolic sine of the corresponding element in *list*.

```
sinh {0,1.2} [ENTER]
                    {0 1.50946135541}
```

sinh^{-1}

MATH HYP menu

sinh^{-1} *number* or **sinh^{-1}(***expression***)**

Returns the inverse hyperbolic sine of *number* or *expression*, which can be real or complex.

```
sinh⁻¹ 1 [ENTER]           .88137358702
```

sinh^{-1} *list*

Returns a list in which each element is the inverse hyperbolic sine of the corresponding element in *list*.

```
sinh⁻¹ {1,2.1,3} [ENTER]
              {.88137358702 1.4874…
```

SinR

STAT CALC menu

*Built-in equation variables such as **y1**, **r1**, and **xt1** are case-sensitive. Do not use **Y1**, **R1**, and **XT1**.*

SinR [*iterations*,] *xList*,*yList* [,*period*],*equationVariable*

Attempts to fit a sinusoidal regression model (y=a sin(bx+c)+d) to real data pairs in *xList* and *yList*, using an optional estimated *period*. The regression equation is stored to *equationVariable*, which must be a built-in equation variable such as **y1**, **r1**, and **xt1**. The equation's coefficients always are stored as a list to built-in variable **PRegC**.

iterations is optional; it specifies the maximum number of times (1 through 16) the TI-86 will attempt to find a solution. If omitted, 8 is used. Typically, larger values result in better accuracy but longer execution times, and vice versa.

If you specify a period, the TI-86 may find a solution more quickly or it may find a solution when one would not have been found otherwise.

If you omit the optional *period*, the difference between values in *xList* should be equal and in sequential order. If you specify *period*, the differences between x values can be unequal.

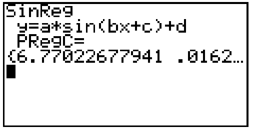
Values used for *xList* and *yList* are stored automatically to built-in variables **xStat** and **yStat**, respectively. The regression equation is stored also to built-in equation variable **RegEq**.

The output of **SinR** is always in radians, regardless of the angle mode setting.

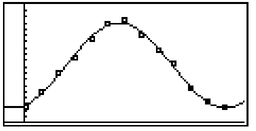
SinR [*iterations*,] *xList*,*yList* [,*period*]

Stores the regression equation to **RegEq** only.

```
seq(x,x,1,361,30)→L1 [ENTER]
          {1 31 61 91 121 151 …
{5.5,8,11,13.5,16.5,19,19.5,17,
14.5,12.5,8.5,6.5,5.5}→L2 [ENTER]
          {5.5 8 11 13.5 16.5…
SinR L1,L2,y1 [ENTER]
```




```
Plot1(1,L1,L2) [ENTER]          Done
ZData [ENTER]
```



SinR [*iterations*,] *equationVariable*

Uses **xStat** and **yStat** for *xList* and *yList*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs. The regression equation is stored to *equationVariable* and **RegEq**.

SinR [*iterations*]

Uses **xStat** and **yStat**, and stores the regression equation to **RegEq** only.

SlpFld

† graph format screen (scroll down to second screen)

SlpFld

In **DifEq** graphing mode, turns on slope fields. To turn off direction and slope fields, use **FldOff**.

Solver(

† [2nd] [SOLVER]

Solver(*equation*,*variable*,*guess*,**{***lower*,*upper***})**

Solves *equation* for *variable*, given an initial *guess* and *lower* and *upper* bounds within which the solution is sought. *equation* can be an expression, which is assumed to equal 0.

If y=5, solve $x^3+y^2=125$ for x. You guess the solution is approximately 4:

```
5→y [ENTER]                        5
Solver(x^3+y²=125,x,4) [ENTER]  Done
x [ENTER]              4.64158883361
```

Solver(*equation*,*variable*,*guess***)**

Uses ⁻1E99 and 1E99 for *upper* and *lower*, respectively.

Solver(*equation*,*variable*,**{***guessLower*,*guessUpper***})**

Uses the secant line between *guessLower* and *guessUpper* to start the search. **Solver(** will still search for a solution outside of this range.

sortA

LIST OPS menu

SortA *list*

Returns a list in which the real or complex elements of *list* are sorted in ascending order.

```
{5,8,-4,0,-6}→L1 [ENTER]
                      {5 8 -4 0 -6}
SortA L1 [ENTER]      {-6 -4 0 5 8}
```

sortD

LIST OPS menu

SortD *list*

Returns a list in which the real or complex elements of *list* are sorted in descending order.

```
{5,8,-4,0,-6}→L1 [ENTER]
                      {5 8 -4 0 -6}
SortD L1 [ENTER]      {8 5 0 -4 -6}
```

Sortx

LIST OPS menu

Sortx *xListName*,*yListName*,*frequencyListName*
Sortx *xListName*,*yListName*

In ascending order of **x** elements, sorts real or complex **x** and **y** data pairs and, optionally, their frequencies in *xListName*, *yListName*, and *frequencyListName*. The lists' contents are updated to reflect the changes.

Sortx

Uses built-in variables **xStat** and **yStat** for *xListName* and *yListName*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

```
{3,1,2}→XL [ENTER]          {3 1 2}
{0,8,-4}→YL [ENTER]        {0 8 -4}
Sortx XL,YL [ENTER]            Done
XL [ENTER]                  {1 2 3}
YL [ENTER]                 {8 -4 0}
```

Sorty

LIST OPS menu

Sorty *xListName*,*yListName*,*frequencyListName*
Sorty *xListName*,*yListName*

In ascending order of **y** elements, sorts real or complex **x** and **y** data pairs and, optionally, their frequencies in *xListName*, *yListName*, and *frequencyListName*. The lists' contents are updated to reflect the changes.

```
{3,1,2}→XL [ENTER]          {3 1 2}
{0,8,-4}→YL [ENTER]        {0 8 -4}
Sorty XL,YL [ENTER]            Done
YL [ENTER]                 {-4 0 8}
XL [ENTER]                  {2 3 1}
```

Sorty

Uses built-in variables **xStat** and **yStat** for *xListName* and *yListName*, respectively. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

▶Sph

VECTR OPS menu

vector▶**Sph**

Displays a 2- or 3-element *vector* as spherical coordinates in $[r \angle\theta \ \angle 0]$ or $[r \angle\theta \ \angle\phi]$ form, respectively, even if the display mode is not set for spherical (**SphereV**).

In **RectV** vector coordinate mode:

```
[0,-1]▶Sph [ENTER]
                  [1∠-1.57079632679∠1.…
[0,0,-1]▶Sph [ENTER]
                  [1∠0∠3.14159265359]
```

SphereV

† [2nd] [MODE]

SphereV

Sets spherical vector coordinate mode $[r \angle\theta \ \angle\phi]$.

In **SphereV** vector coordinate mode:

```
[1,2] [ENTER]
                  [2.2360679775∠1.1071…
```

Square: ²

[x²]

*number*² or (*expression*)²
*list*²
*squareMatrix*²

Returns a real or complex argument multiplied by itself. To square a negative number, enclose it in parentheses.

A *squareMatrix* multiplied by itself is not the same as simply squaring each element.

```
25² [ENTER]                       625
(16+9)² [ENTER]                   625

-2² [ENTER]                        -4
(-2)² [ENTER]                       4

{-2,4,25}² [ENTER]        {4 16 625}

[[2,3][4,5]]² [ENTER]       [[16 21]
                             [28 37]]
```

Square root: √

[2nd] [√]

√*number* or √(*expression*)

Returns the square root of *number* or *expression*, which can be real or complex.

```
√25 [ENTER]                         5
√(25+11) [ENTER]                    6
```

√*list*

Returns a list in which element is the square root of the corresponding element in *list*.

In **RectC** complex number mode:

```
√{-2,25} [ENTER]
                 {(0,1.41421356237) (…
```

St▸Eq(

STRNG menu

St▸Eq(*stringVariable*,*equationVariable***)**

Converts *stringVariable* to a number, expression, or equation, and stores it in *equationVariable*.

To convert the string and retain the same variable name, you can set *equationVariable* equal to *stringVariable*.

```
"5"→x:6 x [ENTER]
                          ERROR 10 DATA TYPE
"5"→x:St▸Eq(x,x):6 x [ENTER]              30
```

Program segment:

```
   :
:InpSt "Enter y1(x):",STR
:St▸Eq(STR,y1)
:Input "Enter x:",x
:Disp "Result is:",y1(x)
   :
```

If you use ***Input*** *instead of* ***InpSt*** *here, the entered expression is evaluated at the current value of x and the result (not the expression) is stored.*

You cannot store a string directly to a built-in equation variable.

StGDB

† GRAPH menu

StGDB *graphDataBaseName*

Creates a graph database (GDB) variable that contains the current:

- Graphing mode, graph format settings, and range variables.
- Functions in the equation editor, whether they are selected, and their graph styles.

To restore the database and recreate the graph, use **RcGDB** (page 343).

Stop

‡ program editor CTL menu

Stop

Ends program execution and returns to the home screen.

Program segment:

```
    :
:Input N
:If N==999
:Stop
    :
```

Use N==999, not N=999.

Store to variable: →

[STO▸]

number→*variable* or (*expression*)→*variable*
string→*variable*
list→*variable*
vector→*variable*
matrix→*variable*

Stores the specified argument to *variable*.

```
10→A:4*A [ENTER]                 40
"Hello"→STR [ENTER]     Hello
{1,2,3}→L1 [ENTER]          {1 2 3}
[1,2,3]→VEC [ENTER]         [1 2 3]
[[1,2,3][4,5,6]]→MAT [ENTER]
                           [[1 2 3]
                            [4 5 6]]
```

StPic

† GRAPH menu

StPic *pictureName*

Stores a picture of the current graph screen to *pictureName*.

StReg(

STAT CALC menu

StReg(*variable***)**

Stores the most recently calculated regression equation to *variable*. This lets you save a regression equation by storing it to any variable as opposed to a built-in equation variable.

```
{1,2,3,4,5}→L1 [ENTER]
                      {1 2 3 4 5}
{1,20,55,230,742}→L2 [ENTER]
                 {1 20 55 230 742}
ExpR L1,L2:StReg(EQ) [ENTER]
                              Done
8→x [ENTER]                      8
Rcl EQ [ENTER]
  .41138948780597*4.7879605684671^x
[ENTER]               113620.765451
```

[2nd] [RCL] EQ [ENTER] recalls the equation. Then [ENTER] evaluates it at the current value of x.

String entry: "

STRNG menu

‡ program editor I/O menu

"*string***"**

Defines a string. When you display a string, it is left-justified on the screen.

Strings are interpreted as text characters, not numbers. For example, you cannot perform a calculation with strings such as "4" or "A∗8". To convert between string variables and equation variables, use **Eq▸St(** and **St▸Eq(** as described on pages 290 and 361, respectively.

```
"Hello"→STR [ENTER]
                         Hello
Disp STR+", Jan" [ENTER]
                         Hello, Jan
                               Done
```

sub(

STRNG menu

sub(*string*,*begin*,*length***)**

Returns a new string that is a subset of *string*, starting at character number *begin* and continuing for the specified *length*.

```
"The answer is:"→STR [ENTER]
                    The answer is:
sub(STR,5,6) [ENTER]
                    answer
```

Subtraction: −

[−]

numberA − *numberB*

Returns the value of *numberB* subtracted from *numberA*. The arguments can be real or complex.

list − *number*

Returns a list in which *number* is subtracted from each element of *list*. The arguments can be real or complex.

```
6-2 [ENTER]                        4
10--4.5 [ENTER]                 14.5
{10,9,8}-4 [ENTER]           {6 5 4}
```

In **RectC** complex number mode:

```
{8,1,(5,2)}-3 [ENTER]
               {(5,0) (⁻2,0) (2,2)}
```

listA – *listB*
matrixA – *matrixB*
vectorA – *vectorB*

Returns a list, matrix, or vector that is the result of each element in the second argument subtracted from the corresponding element in the first argument. The two real or complex arguments must have the same dimension.

```
{5,7,9}-{4,5,6} [ENTER]        {1 2 3}

[[5,7,9][11,13,15]]-[[4,5,6][7,8,
9]] [ENTER]                  [[1 2 3]
                              [4 5 6]]

[5,7,9]-[1,2,3] [ENTER]        [4 5 6]
```

sum

MATH MISC menu

LIST OPS menu

sum *list*

Returns the sum of all real or complex elements in *list*.

```
sum {1,2,4,8} [ENTER]              15
sum {2,7,-8,0} [ENTER]              1
```

tan

[TAN]

tan *angle* or **tan (***expression***)**

Returns the tangent of *angle* or *expression*, which can be real or complex.

An angle is interpreted as degrees or radians according to the current angle mode. In any angle mode, you can designate an angle as degrees or radians by using the ° or ʳ designator, respectively, from the MATH ANGLE menu.

In **Radian** angle mode:

```
tan π/4 [ENTER]                 0
tan (π/4) [ENTER]               1
tan 45° [ENTER]                 1
```

In **Degree** angle mode:

```
tan 45 [ENTER]                  1
tan (π/4)ʳ [ENTER]              1
```

tan *list*

Returns a list in which each element is the tangent of the corresponding element in *list*.

In **Degree** angle mode:

```
tan {0,45,60} [ENTER]
          {0 1 1.73205080757}
```

$\tan^{-1}$

[2nd] [TAN⁻¹]

$\tan^{-1}$ *number* or **$\tan^{-1}$ (***expression***)**

Returns the arctangent of *number* or *expression*, which can be real or complex.

In **Radian** angle mode:

`tan⁻¹ .5 [ENTER]` `.463647609001`

In **Degree** angle mode:

`tan⁻¹ 1 [ENTER]` `45`

$\tan^{-1}$ *list*

Returns a list in which each element is the arctangent of the corresponding element in *list*.

In **Radian** angle mode:

`tan⁻¹ {0,.2,.5} [ENTER]`
`{0 .19739555985 .463…`

tanh

MATH HYP menu

tanh *number* or **tanh (***expression***)**

Returns the hyperbolic tangent of *number* or *expression*, which can be real or complex.

`tanh 1.2 [ENTER]` `.833654607012`

tanh *list*

Returns a list in which each element is the hyperbolic tangent of the corresponding element in *list*.

`tanh {0,1.2} [ENTER]`
`{0 .833654607012}`

$\tanh^{-1}$

MATH HYP menu

$\tanh^{-1}$ *number* or **$\tanh^{-1}$(***expression***)**

Returns the inverse hyperbolic tangent of *number* or *expression*, which can be real or complex.

`tanh⁻¹ 0 [ENTER]` `0`

$\tanh^{-1}$ *list*

Returns a list in which each element is the inverse hyperbolic tangent of the corresponding element in *list*.

In **RectC** complex number mode:

`tanh⁻¹ {0,2.1} [ENTER]`
`{(0,0) (.51804596584…`

TanLn(

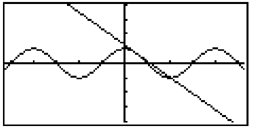
GRAPH DRAW menu

TanLn(*expression*,*xValue***)**

Draws *expression* on the current graph and then draws a tangent line at *xValue*.

In **Func** graphing mode and **Radian** angle mode:

```
ZTrig:TanLn(cos x,π/4) [ENTER]
```



Text(

† GRAPH DRAW menu

Text(*row*,*column*,*string***)**

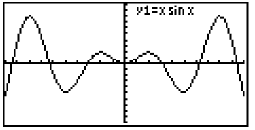
Writes a text *string* on the current graph beginning at pixel (*row*,*column*), where $0 \le row \le 57$ and $0 \le column \le 123$.

Text at the bottom of the graph may be covered by a displayed menu. To remove the menu, press [CLEAR].

Program segment in **Func** graphing mode and a **ZStd** graph screen:

```
  :
:y1=x sin x
:Text(0,70,"y1=x sin x")
  :
```

When executed:



Then

‡ program editor
CTL menu

Refer to syntax information for **If**, beginning on page 305. See the **If:Then:End** and **If:Then:Else:End** syntax.

Trace

† GRAPH menu

Trace

Displays the current graph and lets the user trace a function. From a program, press [ENTER] to stop tracing and continue with the program.

Transpose: ᵀ

MATRX MATH menu

*matrix*ᵀ

Returns a transposed real or complex matrix in which element *row*,*column* is swapped with element *column*,*row* of *matrix*. For example:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{T} \text{ returns } \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

For complex matrices, the complex conjugate of each element is taken.

```
[[1,2][3,4]]→MATA [ENTER]
                                [[1 2]
                                 [3 4]]

MATAᵀ [ENTER]                   [[1 3]
                                 [2 4]]

[[1,2,3][4,5,6][7,8,9]]→MATB
[ENTER]                       [[1 2 3]
                               [4 5 6]
                               [7 8 9]]

MATBᵀ [ENTER]                 [[1 4 7]
                               [2 5 8]
                               [3 6 9]]
```

In **RectC** complex number mode:

```
[[(1,2),(1,1)][(3,2),(4,3)]]
→MATC [ENTER]
                     [[(1,2) (1,1)]
                      [(3,2) (4,3)]]

MATCᵀ [ENTER]      [[(1,-2) (3,-2)]
                    [(1,-1) (4,-3)]]
```

TwoVar

STAT CALC menu
(TwoVa shows on menu)

TwoVar *xList*,*yList*,*frequencyList*

Performs two-variable statistical analysis on the real data pairs in *xList* and *yList*, using the frequencies in *frequencyList*.

Values used for *xList*, *yList*, and *frequencyList* are stored automatically to the built-in variables **xStat**, **yStat**, and **fStat**, respectively.

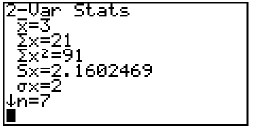
TwoVar *xList*,*yList*

Uses frequencies of 1.

TwoVar

Uses **xStat**, **yStat**, and **fStat** for *xList*, *yList*, and *frequencyList*. These built-in variables must contain valid data of the same dimension; otherwise, an error occurs.

```
{0,1,2,3,4,5,6}→L1 [ENTER]
                    {0 1 2 3 4 5 6}
{0,1,2,3,4,5,6}→L2 [ENTER]
                    {0 1 2 3 4 5 6}
TwoVar L1,L2 [ENTER]
```



Scroll down to see more results.

unitV

VECTR MATH menu

unitV *vector*

Returns a unit vector of a real or complex *vector*, where:

unitV [a,b,c] returns $\left[\frac{\mathbf{a}}{\mathbf{norm}}\ \frac{\mathbf{b}}{\mathbf{norm}}\ \frac{\mathbf{c}}{\mathbf{norm}}\right]$

and

norm is $\sqrt{(\mathbf{a}^2+\mathbf{b}^2+\mathbf{c}^2)}$.

In **RectV** vector coordinate mode:

```
unitV [1,2,1] [ENTER]
              [.408248290464 .8164…
```

vc▶li

LIST OPS menu
VECTR OPS menu

vc▶li *vector*

Returns a real or complex *vector* converted to a list.

```
vc▶li [2,7,-8,0] ENTER     {2 7 -8 0}
(vc▶li [2,7,-8,0])² ENTER
                          {4 49 64 0}
```

Vector entry: [ ]

2nd [[] and 2nd []]

[*element1*,*element2*, ... **]**

Defines a vector in which each element is a real or complex number or variable.

```
[4,5,6]→VEC ENTER           [4 5 6]
```

In **PolarC** complex number mode:

```
[5,(2∠π/4)]→VEC ENTER
             [(5∠0) (2∠.785398163…
```

Vert

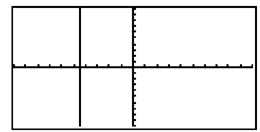
† GRAPH DRAW menu

Vert *xValue*

Draws a vertical line on the current graph at *xValue*.

In a **ZStd** graph screen:

```
Vert -4.5 ENTER
```



While

‡ program editor
CTL menu

:While *condition*
:*commands-while-true*
:End
:*command*

Executes *commands-while-true* as long as *condition* is true.

Program segment:

```
  :
:1→J
:0→TEMP
:While J≤20
: TEMP+1/J→TEMP
: J+1→J
:End
:Disp "Reciprocal sums to
20",TEMP
  :
```

xor

BASE BOOL menu

integerA **xor** *integerB*

Compares two real integers bit by bit. Internally, both integers are converted to binary. When corresponding bits are compared, the result is 1 if either bit (but not both) is 1; the result is 0 if both bits are 0 or both bits are 1. The returned value is the sum of the bit results.

For example, 78 **xor** 23 = 89.

```
78  =  1001110b
23  =  0010111b
       1011001b  =  89
```

You can enter real numbers instead of integers, but they are truncated automatically before the comparison.

In **Dec** number base mode:

```
78 xor 23 [ENTER]                89
```

In **Bin** number base mode:

```
1001110 xor 10111 [ENTER]
                           1011001b
Ans▸Dec [ENTER]                 89d
```

xyline

† STAT DRAW menu

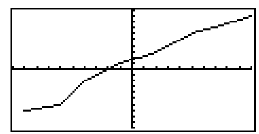
xyline *xList*,*yList*

Draws a line plot on the current graph, using the real data pairs in *xList* and *yList*.

xyline

Uses the data in built-in variables **xStat** and **yStat**. These variables must contain valid data of the same dimension; otherwise, an error occurs.

```
{-9,-6,-4,-1,2,5,7,10}→XL [ENTER]
                 {-9 -6 -4 -1 2 5 7 1…
{-7,-6,-2,1,3,6,7,9}→YL [ENTER]
                  {-7 -6 -2 1 3 6 7 9}
ZStd:xyline XL,YL [ENTER]
```



ZData

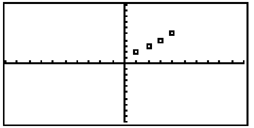
† GRAPH ZOOM menu

ZData

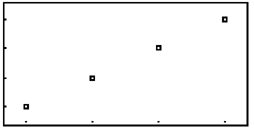
Adjusts the window variable values based on the currently defined statistical plots so that all stat data points will be plotted, and then updates the graph screen.

In **Func** graphing mode:

```
{1,2,3,4}→XL [ENTER]          {1 2 3 4}
{2,3,4,5}→YL [ENTER]          {2 3 4 5}
Plot1(1,XL,YL) [ENTER]             Done
ZStd [ENTER]
```



```
ZData [ENTER]
```



ZDecm

† GRAPH ZOOM menu

ZDecm

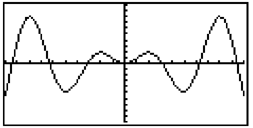
Sets the window variable values such that Δ**x**=Δ**y**=.**1**, and then updates the graph screen with the origin centered on the screen.

xMin=⁻6.3 **yMin=⁻3.1**
xMax=6.3 **yMax=3.1**
xScl=1 **yScl=1**

One of the benefits of **ZDecm** is that you can trace in .1 increments.

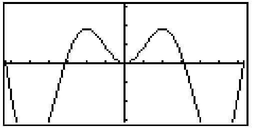
In **Func** graphing mode:

```
y1=x sin x [ENTER]          Done
ZStd [ENTER]
```



If you trace the graph above, **x** values start at 0 and increment by .1587301587.

```
ZDecm [ENTER]
```



If you trace this graph, the **x** values increment by .1.

ZFit

† GRAPH ZOOM menu

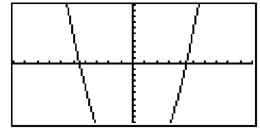
ZFit

Recalculates **yMin** and **yMax** to include the minimum and maximum **y** values of the selected functions between the current **xMin** and **xMax**, and then updates the graph screen.

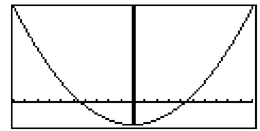
This does not affect **xMin** and **xMax**.

In **Func** graphing mode:

```
y1=x²-20 [ENTER]                  Done
ZStd [ENTER]
```



```
ZFit [ENTER]
```



ZIn

† GRAPH ZOOM menu

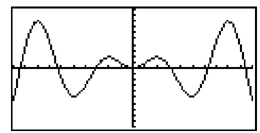
ZIn

Zooms in on the part of the graph centered around the current cursor location.

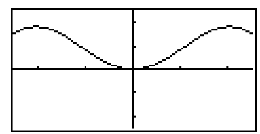
Zoom factors are set by the values of built-in variables **xFact** and **yFact**; the default is 4 for both factors.

In **Func** graphing mode:

```
y1=x sin x [ENTER]                Done
ZStd [ENTER]
```



```
ZIn [ENTER]
```



ZInt

† GRAPH ZOOM menu

ZInt

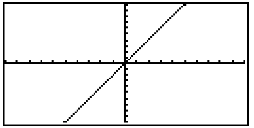
Sets the window variable values so that each pixel is an integer in all directions (Δ**x**=Δ**y**=**1**), sets **xScl**=**yScl**=**10**, and then updates the graph screen.

The current cursor location becomes the center of the new graph.

One of the benefits of **ZInt** is that you can trace in whole number increments.

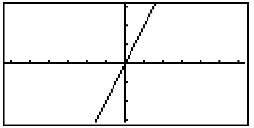
In **Func** graphing mode:

`y1=der1(x²-20,x)` [ENTER] Done
`ZStd` [ENTER]



If you trace the graph above, **x** values start at 0 and increment by .1587301587.

`ZInt` [ENTER]



If you trace this graph, **x** values increment by 1.

ZOut

† GRAPH ZOOM menu

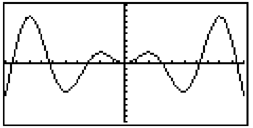
ZOut

Zooms out to display more of the graph, centered around the current cursor location.

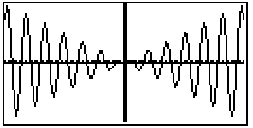
Zoom factors are set by the values of built-in variables **xFact** and **yFact**; the default is 4 for both factors.

In **Func** graphing mode:

```
y1=x sin x [ENTER]                Done
ZStd [ENTER]
```



```
ZOut [ENTER]
```



ZPrev

† GRAPH ZOOM menu

ZPrev

Replots the graph using the window variable values of the graph that was displayed before you executed the previous **ZOOM** instruction.

ZRcl

† GRAPH ZOOM menu

ZRcl

Sets the window variables to values stored previously in the user-defined zoom-window variables, and then updates the graph screen.

To set user-defined zoom-window variables, either:

- Press [GRAPH] [F3] [MORE] [MORE] [MORE] [F1] (**ZSTO**) to store the current graph's window variables.
  – or –
- Store the applicable values to the zoom-window variables, whose names begin with **z** followed by the regular window variable name. For example, store a value for xMin to **zxMin**, yMin to **zyMin**, etc.

ZSqr

† GRAPH ZOOM menu

ZSqr

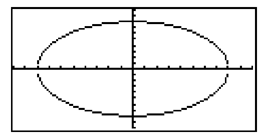
Sets the window variable values to produce "square" pixels where Δ**x**=Δ**y**, and then updates the graph screen.

The center of the current graph (not necessarily the axes intersection) becomes the center of the new graph.

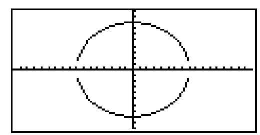
In other types of zooms, squares may look like rectangles and circles may look like ovals. Use **ZSqr** for a more accurate shape.

In **Func** graphing mode:

```
y1=√(8²−x²):y2=⁻y1 [ENTER]          Done
ZStd [ENTER]
```



```
ZSqr [ENTER]
```



ZStd

† GRAPH ZOOM menu

ZStd

Sets the window variables to the standard default values, and then updates the graph screen.

Func graphing mode:

xMin=⁻10	**yMin=⁻10**
xMax=10	**yMax=10**
xScl=1	**yScl=1**

Pol graphing mode:

θMin=0	**xMin=⁻10**	**yMin=⁻10**
θMax=6.28318530718 (2π)	**xMax=10**	**yMax=10**
θStep=.130899693899… (π/24)	**xScl=1**	**yScl=1**

Param graphing mode:

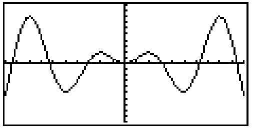
tMin=0	**xMin=⁻10**	**yMin=⁻10**
tMax=6.28318530718 (2π)	**xMax=10**	**yMax=10**
tStep=.130899693899… (π/24)	**xScl=1**	**yScl=1**

DifEq graphing mode:

tMin=0	**xMin=⁻10**	**yMin=⁻10**
tMax=6.28318530718 (2π)	**xMax=10**	**yMax=10**
tStep=.130899693899… (π/24)	**xScl=1**	**yScl=1**
tPlot=0		**difTol=.001**

In **Func** graphing mode:

```
y1=x sin x [ENTER]                Done
ZStd [ENTER]
```



ZTrig

† GRAPH ZOOM menu

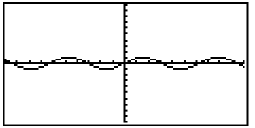
ZTrig

Sets the window variables to preset values appropriate for plotting trig functions in **Radian** angle mode (Δ**x**=π/24), and then updates the graph screen.

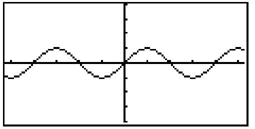
xMin=⁻8.24668071567	**yMin=⁻4**
xMax=8.24668071567	**yMax=4**
xScl=1.5707963267949 (π/2)	**yScl=1**

In **Func** graphing mode:

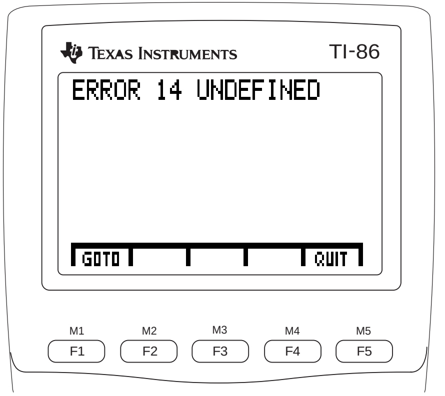
```
y1=sin x [ENTER]                    Done
ZStd [ENTER]
```



```
ZTrig [ENTER]
```



A Appendix



Texas Instruments
TI-86
ERROR 14 UNDEFINED
GOTO
QUIT
M1
M2
M3
M4
M5
F1
F2
F3
F4
F5

TI-86 Menu Map

This section presents the TI-86 menus as they appear on the TI-86 keyboard, starting at the top. If a menu has items that display other menus, the other menus follow directly below the main menu. In the program editor, the appearance of some menus changes slightly. The menu map omits user-created-name menus, such as the LIST NAMES and CONS USER menus.

The link menus are not available in the program editor.

LINK Menu [2nd] [LINK]

SEND	RECV	SND85		

LINK SEND Menu [2nd] [LINK] [F1]

BCKUP	PRGM	MATRX	GDB	ALL	▸	LIST	VECTR	REAL	CPLX	EQU	▸	CONS	PIC	WIND	STRNG	

SEND BCKUP Menu [2nd] [LINK] [F1] [F1]

XMIT				

LINK SEND Selection Screen Menu [2nd] [LINK] [F1] *data type*

XMIT	SELCT	ALL+	ALL−	

LINK SND85 Menu [2nd] [LINK] [F3]

MATRX	LIST	VECTR	REAL	CPLX	▸	CONS	PIC	STRNG		

In the program editor, **DrEqu** *is available as a* GRAPH *menu item.*

GRAPH Menu [GRAPH] in Func graphing mode

y(x)=	WIND	ZOOM	TRACE	GRAPH	▸	MATH	DRAW	FORMT	STGDB	RCGDB	▸	EVAL	STPIC	RCPIC		

GRAPH Menu [GRAPH] in Pol graphing mode

r(θ)=	WIND	ZOOM	TRACE	GRAPH	▸	MATH	DRAW	FORMT	STGDB	RCGDB	▸	EVAL	STPIC	RCPIC		

GRAPH Menu [GRAPH] in Param graphing mode

E(t)=	WIND	ZOOM	TRACE	GRAPH	▸	MATH	DRAW	FORMT	STGDB	RCGDB	▸	EVAL	STPIC	RCPIC		

GRAPH Menu [GRAPH] in DifEq graphing mode

Q'(t)=	WIND	INITC	AXES	GRAPH	▸	FORMT	DRAW	ZOOM	TRACE	EXPLR	▸	EVAL	STGDB	RCGDB	STPIC	RCPIC

Equation Editor Menu [GRAPH] [F1] in Func graphing mode

y(x)=	WIND	ZOOM	TRACE	GRAPH						
x	y	INSf	DELf	SELCT	▸	ALL+	ALL−	STYLE		

Equation Editor Menu [GRAPH] [F1] in Pol graphing mode

r(θ)=	WIND	ZOOM	TRACE	GRAPH						
θ	r	INSf	DELf	SELCT	▸	ALL+	ALL−	STYLE		

Equation Editor Menu [GRAPH] [F1] in Param graphing mode

E(t)=	WIND	ZOOM	TRACE	GRAPH						
t	xt	yt	DELf	SELCT	▸	INSf	ALL+	ALL−	STYLE	

Equation Editor Menu [GRAPH] [F1] in DifEq graphing mode

Q'(t)=	WIND	INITC	AXES	GRAPH						
t	Q	INSf	DELf	SELCT	▸	ALL+	ALL−	STYLE		

GRAPH VARS (Graph Variables) Menu GRAPH F1 in the program editor only

y(x)=	WIND	ZOOM	TRACE	GRAPH
y	x	xt	yt	t

▸

r	θ	Q1	Q'1	t

▸

FnOn	FnOff	Axes	QI	dTime

▸

fldRes				

GRAPH WIND (Window Variables) Menu GRAPH F2 in the program editor only

y(x)=	WIND	ZOOM	TRACE	GRAPH
xMin	xMax	xScl	yMin	yMax

▸

yScl	tMin	tMax	tStep	θMin

▸

θMax	θStep	tPlot	difTol	xRes

▸

EStep				

GRAPH ZOOM Menu GRAPH F3

To display the GRAPH ZOOM *menu in* **DifEq** *mode, press* GRAPH MORE F3.

y(x)=	WIND	ZOOM	TRACE	GRAPH
BOX	ZIN	ZOUT	ZSTD	ZPREV

▸

ZFIT	ZSQR	ZTRIG	ZDECM	ZDATA

▸

ZRCL	ZFACT	ZOOMX	ZOOMY	ZINT

▸

ZSTO				

GRAPH MATH Menu GRAPH MORE F1 in Func graphing mode

DifEq *graphing mode has no* GRAPH MATH *menu.*

MATH	DRAW	FORMT	STGDB	RCGDB
ROOT	dy/dx	∫f(X)	FMIN	FMAX

▸

INFLC	YICPT	ISECT	DIST	ARC

▸

TANLN				

GRAPH MATH Menu GRAPH MORE F1 in Pol graphing mode

MATH	DRAW	FORMT	STGDB	RCGDB
DIST	dy/dx	dr/dθ	ARC	TANLN

GRAPH MATH Menu [GRAPH] [MORE] [F1] in Param graphing mode

MATH	DRAW	FORMT	STGDB	RCGDB						
DIST	dy/dx	dy/dt	dx/dt	ARC	▸	TANLN				

GRAPH DRAW Menu [GRAPH] [MORE] [F2]

DrInv *is available only in Func graphing mode.* **DrEqu** *is available only in* **DifEq** *graphing mode.*

MATH	DRAW	FORMT	STGDB	RCGDB											
Shade	LINE	VERT	HORIZ	CIRCL	▸	DrawF	PEN	PTON	PTOFF	PTCHG	▸	CLDRW	PxOn	PxOff	PxChg	PxTest

▸

TEXT	TanLn	DrInv		

SOLVER Menu [2nd] [SOLVER] *equation* [ENTER]

GRAPH	WIND	ZOMM	TRACE	SOLVE

SOLVER ZOOM Menu [2nd] [SOLVER] *equation* [ENTER] [F3]

BOX	ZINT	ZOUT	ZFACT	ZSTD

TABLE Menu [TABLE]

TABLE	TBLST			

TABLE SETUP Menu [TABLE] [F2]

TABLE				

Table Screen Menu [TABLE] [F1]

in Func graphing mode

TBLST	SELCT	x	y	

in Param graphing mode

TBLST	SELCT	t	xt	yt

in Pol graphing mode

TBLST	SELCT	θ	r	

in DifEq graphing mode

TBLST	SELCT	t	Q	

SIMULT ENTRY Menu [2nd] [SIMULT] *(integer* ≥ 2 *&* ≤ 30*)* [ENTER]

PREV	NEXT	CLRq		SOLVE

SIMULT RESULT Menu [F5]

COEFS	STOa	STOb	STOx	

PRGM Menu [PRGM]

NAMES	EDIT			

Program Editor Menu [PRGM] [F2] *program name* [ENTER]

PAGE↓	PAGE↑	I/O	CTL	INSc	▸	DELc	UNDEL	:		

PRGM I/O (Input/Output) Menu [PRGM] [F2] *program name* [ENTER] [F3]

PAGE↓	PAGE↑	I/O	CTL	INSc												
Input	Promp	Disp	DispG	DispT	▸	ClTbl	Get	Send	getKy	ClLCD	▸	"	Outpt	InpSt		

PRGM CTL (Control) Menu [PRGM] [F2] *programName* [ENTER] [F4]

PAGE↓	PAGE↑	I/O	CTL	INSc												
If	Then	Else	For	End	▸	While	Repea	Menu	Lbl	Goto	▸	IS>	DS<	Pause	Retur	Stop
											▸	DelVa	GrStl	LCust		

POLY ENTRY Menu [2nd] [POLY] *(integer ≥ 2 & ≤ 30)* [ENTER]

CLRq				SOLVE

POLY RESULT Menu [F5]

COEFS	STOa			

CUSTOM Menu [CUSTOM]

Use the CUSTOM *menu to create your own menu (Chapter 2).*

					▸						▸					

CATLG-VARS Menu [2nd] [CATLG-VARS]

CATLG	ALL	REAL	CPLX	LIST	▸	VECTR	MATRX	STRNG	EQU	CONS	▸	PRGM	GDB	PIC	STAT	WIND

CATLG-VARS Selection Menu [2nd] [CATLG-VARS] [F1] or select a data type

PAGE↓	PAGE↑	CUSTM	BLANK	

CALC Menu 2nd [CALC]

evalF	nDer	der1	der2	fnInt	▸	fMin	fMax	arc		

MATRX Menu 2nd [MATRX]

NAMES	EDIT	MATH	OPS	CPLX

Matrix Editor Menu 2nd [MATRX] F2 *matrixName* ENTER

INSr	DELr	INSc	DELc	▸REAL

MATRX MATH Menu 2nd [MATRX] F3

NAMES	EDIT	MATH	OPS	CPLX						
det	T	norm	eigVl	eigVc	▸	rnorm	cnorm	LU	cond	

MATRX OPS (Operations) Menu 2nd [MATRX] F4

NAMES	EDIT	MATH	OPS	CPLX												
dim	Fill	ident	ref	rref	▸	aug	rSwap	rAdd	multR	mRAdd	▸	randM				

MATRX CPLX Menu 2nd [MATRX] F5

NAMES	EDIT	MATH	OPS	CPLX
conj	real	imag	abs	angle

VECTR Menu 2nd [VECTR]

NAMES	EDIT	MATH	OPS	CPLX

Vector Editor Menu 2nd [VECTR] F2 *vectorName* ENTER

INSi	DELi	▸REAL		

VECTR MATH Menu 2nd [VECTR] F3

NAMES	EDIT	MATH	OPS	CPLX
cross	unitV	norm	dot	

VECTR OPS (Operations) Menu [2nd] [VECTR] [F4]

NAMES	EDIT	MATH	OPS	CPLX						
dim	Fill	▸Pol	▸Cyl	▸Sph	▸	▸Rec	li▸vc	vc▸li		

VECTR CPLX Menu [2nd] [VECTR] [F5]

NAMES	EDIT	MATH	OPS	CPLX
conj	real	imag	abs	angle

CPLX (Complex Number) Menu [2nd] [CPLX]

conj	real	imag	abs	angle	▸	▸Rec	▸Pol			

MATH Menu [2nd] [MATH]

NUM	PROB	ANGLE	HYP	MISC	▸	INTER				

MATH NUM (Number) Menu [2nd] [MATH] [F1]

NUM	PROB	ANGLE	HYP	MISC						
round	iPart	fPart	int	abs	▸	sign	min	max	mod	

MATH PROB (Probability) Menu [2nd] [MATH] [F2]

NUM	PROB	ANGLE	HYP	MISC						
!	nPr	nCr	rand	randIn	▸	randN	randBi			

MATH ANGLE Menu [2nd] [MATH] [F3]

NUM	PROB	ANGLE	HYP	MISC
°	r	'	▸DMS	

MATH HYP (Hyperbolic) Menu

[2nd] [MATH] [F4]

NUM	PROB	ANGLE	HYP	MISC
sinh	cosh	tanh	$sinh^{-1}$	$cosh^{-1}$

▸

$tanh^{-1}$				

MATH MISC (Miscellaneous) Menu

[2nd] [MATH] [F5]

NUM	PROB	ANGLE	HYP	MISC
sum	prod	seq	lcm	gcd

▸

▸Frac	%	pEval	x√	eval

CONS (Constants) Menu

[2nd] [CONS]

BLTIN	EDIT	USER		

CONS BLTIN (Built-In Constants) Menu

[2nd] [CONS] [F1]

BLTIN	EDIT	USER		
Na	k	Cc	ec	Rc

▸

Gc	g	Me	Mp	Mn

▸

μ0	ε0	h	c	u

CONV (Conversions) Menu

[2nd] [CONV]

LNGTH	AREA	VOL	TIME	TEMP

▸

MASS	FORCE	PRESS	ENRGY	POWER

▸

SPEED				

CONV LNGTH (Length) Menu

[2nd] [CONV] [F1]

LNGTH	AREA	VOL	TIME	TEMP
mm	cm	m	in	ft

▸

yd	km	mile	nmile	lt-yr

▸

mil	Ang	fermi	rod	fath

CONV AREA Menu

[2nd] [CONV] [F2]

LNGTH	AREA	VOL	TIME	TEMP
ft^2	m^2	mi^2	km^2	acre

▸

in^2	cm^2	yd^2	ha	

CONV VOL (Volume) Menu 2nd [CONV] F3

LNGTH	AREA	VOL	TIME	TEMP
liter	gal	qt	pt	oz

cm^3	in^3	ft^3	m^3	cup

tsp	tbsp	ml	galUK	ozUK

CONV TIME Menu 2nd [CONV] F4

LNGTH	AREA	VOL	TIME	TEMP
sec	mn	hr	day	yr

week	ms	µs	ns	

CONV TEMP (Temperature) Menu 2nd [CONV] F5

LNGTH	AREA	VOL	TIME	TEMP
°C	°F	°K	°R	

CONV MASS Menu 2nd [CONV] MORE F1

MASS	FORCE	PRESS	ENRGY	POWER
gm	kg	lb	amu	slug

ton	mton			

CONV FORCE Menu 2nd [CONV] MORE F2

MASS	FORCE	PRESS	ENRGY	POWER
N	dyne	tonf	kgf	lbf

CONV PRESS (Pressure) Menu 2nd [CONV] MORE F3

MASS	FORCE	PRESS	ENRGY	POWER
atm	bar	N/m^2	lb/in^2	mmHg

mmH_2	inHg	inH_20		

CONV ENRGY (Energy) Menu 2nd [CONV] MORE F4

MASS	FORCE	PRESS	ENRGY	POWER
J	cal	Btu	ft-lb	kw-hr

eV	erg	l-atm		

CONV POWER Menu [2nd] [CONV] [MORE] [F5]

MASS	FORCE	PRESS	ENRGY	POWER
hp	W	ftlb/s	cal/s	Btu/m

CONV SPEED Menu [2nd] [CONV] [MORE] [MORE] [F1]

SPEED				
ft/s	m/s	mi/hr	km/hr	knot

STRNG Menu [2nd] [STRNG]

"	sub	lngth	Eq▸St	St▸Eq

LIST Menu [2nd] [LIST]

{	}	NAMES	EDIT	OPS

LIST NAMES Menu [2nd] [LIST] [F3]

{	}	NAMES	EDIT	OPS
fStat	xStat	yStat		

List Editor Menu [2nd] [LIST] [F4]

{	}	NAMES	"	OPS

▸

▸REAL				

LIST OPS (Operations) Menu [2nd] [LIST] [F5]

{	}	NAMES	EDIT	OPS
dimL	sortA	sortD	min	max

▸

sum	prod	seq	li▸vc	vc▸li

▸

Fill	aug	cSum	Deltal	Sortx

▸

Sorty	Select	SetLE	Form	

The (Number) BASE Menu [2nd] [BASE]

A-F	TYPE	CONV	BOOL	BIT

BASE A-F (Hexadecimal) Menu [2nd] [BASE] [F1]

A	TYPE	CONV	BOOL	BIT
B	C	D	E	F

BASE TYPE Menu [2nd] [BASE] [F2]

A-F	TYPE	CONV	BOOL	BIT
b	h	o	d	

BASE CONV (Conversions) Menu [2nd] [BASE] [F3]

A-F	TYPE	CONV	BOOL	BIT
▸Bin	▸Hex	▸Oct	▸Dec	

BASE BOOL (Boolean) Menu [2nd] [BASE] [F4]

A-F	TYPE	CONV	BOOL	BIT
and	or	xor	not	

BASE BIT Menu [2nd] [BASE] [F5]

A-F	TYPE	CONV	BOOL	BIT
rotR	rotL	shftR	shftL	

TEST (Relational) Menu [2nd] [TEST]

==	<	>	≤	≥	▸	≠				

MEM (Memory) Menu [2nd] [MEM]

RAM	DELET	RESET	TOL	ClrEnt

MEM DELET (Delete) Menu [2nd] [MEM] [F2]

ALL	REAL	CPLX	LIST	VECTR	▸	MATRX	STRNG	EQU	CONS	PRGM	▸	GDB	PIC			

MEM RESET Menu [2nd] [MEM] [F3]

RAM	DELET	RESET	TOL	ClrEnt
ALL	MEM	DFLTS		

MEM RESET Are You Sure? Menu

			YES	NO

STAT (Statistics) Menu [2nd] [STAT]

When you press [2nd] [STAT] [F2], the list editor and list menu are displayed.

CALC	EDIT	PLOT	DRAW	VARS	▸	FCST				

STAT CALC (Calculations) Menu [2nd] [STAT] [F1]

CALC	EDIT	PLOT	DRAW	VARS												
OneVa	TwoVa	LinR	LnR	ExpR	▸	PwrR	SinR	LgstR	P2Reg	P3Reg	▸	P4Reg	StReg			

STAT PLOT Menu [2nd] [STAT] [F3]

PLOT1	PLOT2	PLOT3	PlOn	PlOff

Plot Type Menu [2nd] [STAT] [F3] ([F1], [F2], or [F3]) [▼]

PLOT1	PLOT2	PLOT3	PlOn	PlOff
SCAT	xyLINE	MBOX	HIST	BOX

Plot Mark Menu [2nd] [STAT] [F3] ([F1], [F2], or [F3]) [▼] ([F1], [F2], or [F3]) [▼] [▼] [▼]

PLOT1	PLOT2	PLOT3	PlOn	PlOff
□	+	•		

STAT DRAW Menu [2nd] [STAT] [F4]

CALC	EDIT	PLOT	DRAW	VARS
HIST	SCAT	xyLINE	BOX	MBOX

▸

DRREG	CLDRW	DrawF	STPIC	RCPIC

STAT VARS (Statistical Result Variables) Menu [2nd] [STAT] [F5]

CALC	EDIT	PLOT	DRAW	VARS
$\bar{x}$	σx	Sx	$\bar{y}$	σy

▸

Sy	Σx	Σx²	Σy	Σy²

▸

Σxy	RegEq	corr	a	b

▸

n	minX	maxX	minY	maxY

▸

Med	PRegC	Qrtl1	Qrtl3	tolMe

CHAR (Character) Menu [2nd] [CHAR]

MISC	GREEK	INTL		

CHAR MISC (Miscellaneous) Menu [2nd] [CHAR] [F1]

Ñ, ñ, Ç, and ç are valid as the first letter of a variable name.

%, ' , and ! can be functions.

MISC	GREEK	INTL		
?	#	&	%	'

▸

!	@	$	~	|

▸

¿	Ñ	ñ	Ç	ç

All CHAR GREEK *menu items are valid variable-name characters, including the first letter.* π ([2nd] [π]) *is not valid as a character;* π *is a constant on the TI-86.*

CHAR GREEK Menu [2nd] [CHAR] [F2]

MISC	GREEK	INTL		
α	β	γ	Δ	δ

▸

ε	θ	λ	μ	ρ

▸

Σ	σ	ι	φ	Ω

CHAR INTL (International Letter Symbols) Menu [2nd] [CHAR] [F3]

MISC	GREEK	INTL		
´	`	^	¨	

Handling a Difficulty

1. If you cannot see anything on the screen, you may need to adjust the contrast (Chapter 1).
   - To darken the screen, press and release [2nd], and then press and hold [▲].
   - To lighten the screen, press and release [2nd], and then press and hold [▼].
2. If an error menu is displayed, follow the steps in Chapter 1. Refer to the Error Conditions section of the Appendix (page 393) for details about specific errors, if necessary.
3. If a checkerboard cursor (▦) is displayed, then either you have entered the maximum number of characters in a prompt or memory is full. If memory is full, press [2nd] [MEM] [F2], select a data type, and then delete some items from memory (Chapter 17).
4. If the busy indicator (dotted line) is displayed in the top-right corner, a graph or program has paused; the TI-86 is waiting for input. Press [ENTER] to continue or press [ON] to break.
5. If the calculator does not seem to work at all, be sure the batteries are fresh and that they are installed properly. Refer to battery information in Chapter 1.

Error Conditions

When the TI-86 detects an error, it displays an error message **ERROR** # *type* and the error menu. Chapter 1 describes how to correct an error. This section describes possible causes for the errors and examples. To find the proper arguments for a function or instruction, as well as restrictions on those arguments, refer to Chapter 20: A to Z Function and Instruction Reference.

Errors 1 through 5 do not occur during graphing. The TI-86 allows for undefined values on a graph.

01 OVERFLOW
- You attempted to enter a number that is beyond the calculator's range.
- You attempted to execute an expression with a result that is beyond the calculator's range.

02 DIV BY ZERO
- You attempted to divide by zero.
- You attempted a linear regression with a vertical line.

03 SINGULAR MAT
- You attempted to use a singular matrix (determinate = 0) as the argument for $^{-1}$, **Simult**, or **LU**.
- You attempted a regression with at least one inappropriate list.
- You attempted to use a matrix with repeated eigenvalues as the argument for **exp**, **cos**, or **sin**.

04 DOMAIN
- You attempted to use an argument that is out of the range of valid values for the function or instruction.
- You attempted a logarithmic or power regression with a ⁻x or an exponential regression with a ⁻y.

05 INCREMENT The increment in **seq** is **0** or has the wrong sign; the increment for a loop is **0**.

06 BREAK You pressed [ON] to break a program, DRAW instruction, or expression evaluation.

07 SYNTAX You entered a value; look for misplaced functions, arguments, parentheses, or commas; check the syntax description in the A to Z Reference.

08 NUMBER BASE	♦ You entered an invalid digit in a number base, such as **7b**. ♦ You attempted an operation that is not allowed in **Bin**, **Oct** , or **Hex** base mode.
09 MODE	You attempted to store to a window variable of a noncurrent graphing mode. or to use an instruction valid only in noncurrent graphing modes; for example, using **DrInv** in **Pol**, **Param**, or **DifEq** graphing mode.
10 DATA TYPE	♦ You entered a value or variable that is an inappropriate data type. ♦ You entered an argument that is an inappropriate data type for a function or an instruction, such as a program name for **sortA**. ♦ In an editor, you entered a data type that is not allowed; check the appropriate chapter. ♦ You attempted to store data to a protected data type, such as a constant, program, picture, or graph database. ♦ You attempted to store inappropriate data to a restricted built-in variable, such as the list names **xStat**, **yStat**, and **fStat**.
11 ARGUMENT	You attempted to execute a function or instruction without all the arguments.
12 DIM MISMATCH	You attempted to use two or more lists, matrices, or vectors as arguments, but the dimensions of all arguments are not equal, such as **{1,2}+{1,2,3}**.
13 DIMENSION	♦ You entered an argument with an inappropriate dimension. ♦ You entered a matrix or vector dimension < 1 or > 255 or a noninteger. ♦ You attempted to invert a matrix that is not a square matrix.
14 UNDEFINED	You are referencing a variable that currently is not defined.
15 MEMORY	Memory is insufficient to perform the desired command; you must delete items from memory (Chapter 17) before executing this command.
16 RESERVED	You attempted to use a built-in variable inappropriately.
17 INVALID	You attempted to reference a variable or use a function where it is not valid.

Errors 26 through 29 occur during the solving process. Examine a graph of the function or a graph of the variable vs. **left−rt** *in the* SOLVER. *If the equation has a solution, change bounds and/or the initial guess.*

18 ILLEGAL NEST	You attempted to use an invalid function in an argument for **seq(** or a CALC function; for example, **der1(der1(x^3,x),x))**.
19 BOUND	You defined an upper bound that is less than the specified lower bound or a lower bound that is greater than the specified upper bound.
20 GRAPH WINDOW	♦ One or more window variable values is incompatible with the others for defining the graph screen; for example, you defined **xMax** < **xMin**. ♦ Window variables are too small or too large to graph correctly; for example, you attempted to zoom out beyond the calculator's range.
21 ZOOM	A ZOOM operation resulted in an error; you attempted to define **ZBOX** with a line.
22 LABEL	In programming, the **Goto** instruction label is not defined with a **Lbl** instruction.
23 STAT	♦ You attempted a stat calculation with at least one inappropriate list, such as a list with less than two data points. ♦ At least one element of a frequency list is < 0. ♦ (**xMax** − **xMin**)/**xScl** ≤ 63 must be true when plotting a histogram.
24 CONVERSION	When converting measurements, the units are incompatible, as in volts to liters.
25 SOLVER	♦ In the solver editor, the equation does not contain a variable. ♦ You attempted to graph with the cursor positioned on bound.
26 SINGULARITY	In the solver editor, the equation contains a singularity, which is a point at which the function is not defined.
27 NO SIGN CHNG	The solver did not detect a sign change.
28 ITERATIONS	The solver has exceeded the maximum permitted number of iterations.
29 BAD GUESS	♦ The initial guess was outside the specified bounds. ♦ The initial guess and several points around the guess are undefined.

30 DIF EQ SETUP	In **DifEq** graphing mode, equations in the equation editor must be from **Q'1** to **Q'9** and each must have an associated initial condition from **QI1** to **QI9.**
31 DIF EQ MATH	The step size used by the fitting algorithm has become too small; check the equations and initial values; try a larger value for the window variable **difTol**; try changing **tMin** or **tMax** to examine a different region of the solution.
32 POLY	All coefficients are **0**.
33 TOL NOT MET	The algorithm cannot return a result accurate to the requested tolerance.
34 STAT PLOT	You attempted to display a stat plot that references an undefined list.
35 AXES	You attempted to plot a **DifEq** graph with improper axes set.
36 FLD/ORDER	♦ You attempted to plot a 2nd-order or higher differential equation with **SlpFld** field format set; change field format or modify the order. ♦ You attempted to plot a 3rd-order or higher differential equation with **DirFld** field format set; change field format or modify the order.
37 LINK MEMORY FULL	You attempted to transmit an item with insufficient available memory in the receiving unit; skip the item or cancel the transmission.
38 LINK TRANSMISSION ERROR	♦ Unable to transmit item; check to see that the cable is firmly connected to both units and the receiving unit is ready to receive data (Chapter 18). ♦ You pressed [ON] to break during transmission.
39 LINK DUPLICATE NAME	You attempted to transmit an item when an item with the same name already exists in the receiving unit.

Equation Operating System (EOS™)

The Equation Operating System (EOS) governs the order of evaluation on the TI-86. Calculations within parentheses are evaluated first, and then EOS evaluates functions within an expression in this order:

Within a priority level, EOS evaluates functions from left to right.

Multi-argument functions, such as **nDeriv(A2,A,6)**, *are evaluated as they are encountered.*

1st	Functions that are entered after the argument, such as ^{2}, $^{-1}$, **!**, °, ^{r}, and conversions
2nd	Powers and roots, such as **2^5** or **5**^{x}**√32**
3rd	Single-argument functions that precede the argument, such as **√(** , **sin(** , or **log(**
4th	Permutations (**nPr**) and combinations (**nCr**)
5th	Multiplication, implied multiplication, and division
6th	Addition and subtraction
7th	Relational functions, such as **>** or ≤
8th	Logic operator **and**
9th	Logic operators **or** and **xor**

TI-86 implied multiplication rules differ from those of the TI-85. For example, the TI-86 evaluates **1/2x** *as* **(1/2)∗x**, *while the TI-85 evaluates* **1/2x** *as* **1/(2∗x)**.

Implied Multiplication

The TI-86 recognizes implied multiplication, so you need not press [×] to express multiplication in all cases. For example, the TI-86 interprets **2π**, **4sin(46)**, **5(1+2)**, and **(2∗5)7** as implied multiplication.

Parentheses

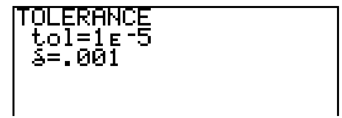
All calculations inside a pair of parentheses are completed first. For example, in the expression **4(1+2)**, EOS evaluates **1+2** inside the parentheses first, and then multiplies **3** by **4**.

You can omit the close parenthesis (**)**) at the end of an expression. All open parenthetical elements are closed automatically at the end of an expression. This is also true for open parenthetical elements that precede the store or display-conversion instructions.

Open parentheses after list names, matrix names, or equation function names are not interpreted as implied multiplication. Arguments that follow these open parentheses are specified list elements, matrix elements, or values for which to solve the equation function.

TOL (The Tolerance Editor) [2nd] [MEM] [F4]

On the TI-86, the computational accuracy of some functions is controlled by the variables **tol** and δ. The values stored to these variables may affect the speed at which the TI-86 calculates or plots.



The variable **tol** defines the tolerance in calculating the functions **fnInt(**, **fMin(**, **fMax(**, and **arc(**, and the GRAPH MATH operations **Σf(x)**, **FMIN**, **FMAX**, and **ARC** (Chapter 6). **tol** must be a positive value ≥ 1ᴇ-12.

The value stored to δ must be a positive real number. δ defines the step size the TI-86 uses to calculate the functions **arc** in **dxNDer** mode; **nDer**; and the operations **dy/dx**, **dr/dθ**, **dy/dt**, **dx/dt**, **INFLC**, **TANLN**, and **ARC**, all in **dxNDer** mode (Chapter 6).

To store a value to **tol** or δ on the home screen or in a program, use [STO▸]. You can select **tol** and δ from the CATALOG. Also, you can enter **tol** directly and select δ from the CHAR GREEK menu.

Computational Accuracy

To maximize accuracy, the TI-86 carries more digits internally than it displays. Values are stored in memory using up to 14 digits with a 3-digit exponent.

- ♦ You can store values up to 12 digits long to most window variables. To **xScl**, **yScl**, **tStep**, and θ**Step**, you can store values up to 14 digits long.
- ♦ When a value is displayed, the displayed value is rounded as specified by the mode setting (Chapter 1), with a maximum of 12 digits and a 3-digit exponent.
- ♦ Chapter 4 describes calculations in hexadecimal, octal, and binary number bases.

Support and Service Information

Product Support

Customers in the U.S., Canada, Puerto Rico, and the Virgin Islands

For general questions, contact Texas Instruments Customer Support:

phone: **1-800-TI-CARES (1-800-842-2737)**
e-mail: **ti-cares@ti.com**

For technical questions, call the Programming Assistance Group of Customer Support:

phone: **1-972-917-8324**

Customers outside the U.S., Canada, Puerto Rico, and the Virgin Islands

Contact TI by e-mail or visit the TI **Calculator** home page on the World Wide Web.

e-mail: **ti-cares@ti.com**
Internet: **education.ti.com**

Product Service

Customers in the U.S. and Canada Only

Always contact Texas Instruments Customer Support before returning a product for service.

Customers outside the U.S. and Canada

Refer to the leaflet enclosed with this product or contact your local Texas Instruments retailer/distributor.

Other TI Products and Services

Visit the TI **Calculator** home page on the World Wide Web.

education.ti.com

Warranty Information

Customers in the U.S. and Canada Only

One-Year Limited Warranty for Commercial Electronic Product

This Texas Instruments electronic product warranty extends only to the original purchaser and user of the product.

Warranty Duration. This Texas Instruments electronic product is warranted to the original purchaser for a period of one (1) year from the original purchase date.

Warranty Coverage. This Texas Instruments electronic product is warranted against defective materials and construction. **THIS WARRANTY IS VOID IF THE PRODUCT HAS BEEN DAMAGED BY ACCIDENT OR UNREASONABLE USE, NEGLECT, IMPROPER SERVICE, OR OTHER CAUSES NOT ARISING OUT OF DEFECTS IN MATERIALS OR CONSTRUCTION.**

WARRANTY DISCLAIMERS. ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE ONE-YEAR PERIOD. TEXAS INSTRUMENTS SHALL NOT BE LIABLE FOR LOSS OF USE OF THE PRODUCT OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER.

Some states/provinces do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you.

Legal Remedies. This warranty gives you specific legal rights, and you may also have other rights that vary from state to state or province to province.

Warranty Performance. During the above one (1) year warranty period, your defective product will be either repaired or replaced with a reconditioned model of an equivalent quality, (at TI's option) when the product is returned, postage prepaid, to Texas Instruments Service Facility. The warranty for the repaired or replacement unit will continue for the warranty of the original unit or six (6) months, whichever is longer. Other than the postage requirement, no charge will be made for such repair and/or replacement. TI strongly recommends that you insure the product for value prior to mailing.

Software. Software is licensed, not sold. TI and its licensors do not warrant that the software will be free from errors or meet your specific requirements. **All software is provided "AS IS."**

Copyright. The software and any documentation supplied with this product are protected by copyright.

Australia & New Zealand Customers only

One-Year Limited Warranty for Commercial Electronic Product

This Texas Instruments electronic product warranty extends only to the original purchaser and user of the product.

Warranty Duration. This Texas Instruments electronic product is warranted to the original purchaser for a period of one (1) year from the original purchase date.

Warranty Coverage. This Texas Instruments electronic product is warranted against defective materials and construction. This warranty is void if the product has been damaged by accident or unreasonable use, neglect, improper service, or other causes not arising out of defects in materials or construction.

Warranty Disclaimers. Any implied warranties arising out of this sale, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, are limited in duration to the above one-year period. Texas Instruments shall not be liable for loss of use of the product or other incidental or consequential costs, expenses, or damages incurred by the consumer or any other user.

Some jurisdictions do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you.

Legal Remedies. This warranty gives you specific legal rights, and you may also have other rights that vary from jurisdiction to jurisdiction.

Warranty Performance. During the above one (1) year warranty period, your defective product will be either repaired or replaced with a new or reconditioned model of an equivalent quality (at TI's option) when the product is returned to the original point of purchase. The repaired or replacement unit will continue for the warranty of the original unit or six (6) months, whichever is longer. Other than your cost to return the product, no charge will be made for such repair and/or replacement. TI strongly recommends that you insure the product for value if you mail it.

Software. Software is licensed, not sold. TI and its licensors do not warrant that the software will be free from errors or meet your specific requirements. All software is provided "AS IS."

Copyright. The software and any documentation supplied with this product are protected by copyright.

All Customers outside the U.S. and Canada

For information about the length and terms of the warranty, refer to your package and/or to the warranty statement enclosed with this product, or contact your local Texas Instruments retailer/distributor.

Index

C

D

E

P