Texas Instruments

TI-92
GUIDEBOOK

The TI-92 Geometry was jointly developed by TI and the authors of Cabri Geometry II™, who are with the Université Joseph Fourier, Grenoble, France.

The TI-92 Symbolic Manipulation was jointly developed by TI and the authors of the DERIVE® program, who are with Soft Warehouse, Inc., Honolulu, HI.

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.

Caution: Any changes or modifications to this equipment not expressly approved by Texas Instruments may void your authority to operate the equipment.

Table of Contents

This guidebook describes how to use the TI-92. The table of contents can help you locate “getting started” information as well as detailed information about the TI-92’s features.

Table of Contents (Continued)

How to Use this Guidebook

The last thing most people want to do is read a book of instructions before using a new product. With the TI-92, you can perform a variety of calculations without opening the guidebook. However, by reading at least parts of the book and skimming through the rest, you can learn about capabilities that let you use the TI-92 more effectively.

How the Guidebook Is Organized

The TI-92 has a wide variety of features and applications (Home screen, Y= Editor, Graph screen, Geometry, etc.) that are explained in this guidebook. Generally, the guidebook is divided into three major parts.

- Chapters 1 – 9 cover topics that are often used by people who are just getting started with the TI-92.
- Chapters 10 – 19 cover additional topics that may not be used right away (depending on your situation).
- The appendices provide useful reference information, as well as service and warranty information.

Which Chapters Should You Read?

Particularly when you first get started, you may not need to use all of the TI-92's capabilities. Therefore, you only need to read the chapters that apply to you. It's a little like the dictionary. If you're looking for *xylophone*, skip A through W.

If you want to:	Go to:
Get an overview of the TI-92 and its capabilities	**Chapter 1** — Contains step-by-step examples to get you started performing calculations, graphing functions, constructing geometric objects, etc. **Chapter 2** — Gives general information about operating the TI-92. Although this chapter primarily covers the Home screen, much of the information applies to any application.
Learn about a particular application or topic	**The applicable chapter** — For example, to learn how to graph a function, go to Chapter 3: Basic Function Graphing. Most chapters start with a step-by-step "preview" example that illustrates one or more of the topics covered in that chapter.

Although you don't need to read every chapter, skim through the entire guidebook and stop at anything that interests you. You may find a feature that could be very useful, but you might not know it exists if you don't look around.

How Do I Look Up Information?

Because the book is big, it's important that you know how to look things up quickly. Use the:

- Table of contents
- Index
- Appendix A (for detailed information about a particular TI-92 function or instruction)

Notes about Appendix A

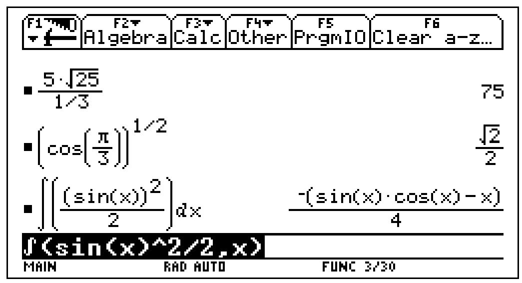
Long after you learn to use the TI-92, Appendix A can continue to be a valuable reference.

- You can access most of the TI-92's functions and instructions by selecting them from menus. Use Appendix A for details about the arguments and syntax used for each function and instruction.
  - You can also use the Help information that is displayed at the bottom of the CATALOG menu, as described in Chapter 2.
- At the beginning of Appendix A, the available functions and instructions are grouped into categories. This can help you locate a function or instruction if you don't know its name.
  - Also refer to Chapter 17, which categorizes program commands.

Getting Started

1

This chapter helps you to get started using the TI-92 quickly. This chapter takes you through several examples to introduce you to some of the principle operating and graphing functions of the TI-92.



After setting up your TI-92 and completing these examples, please read Chapter 2: Operating the TI-92. You then will be prepared to advance to the detailed information provided in the remaining chapters in this guidebook.

Getting the TI-92 Ready to Use

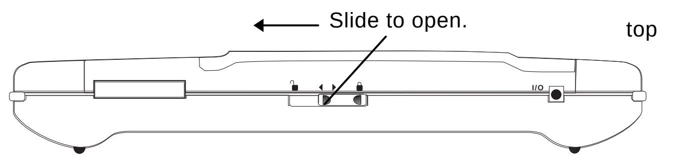
The TI-92 comes with four AA batteries. This section describes how to install these batteries, turn the unit on for the first time, set the display contrast, and view the Home screen.

Installing the AA Batteries

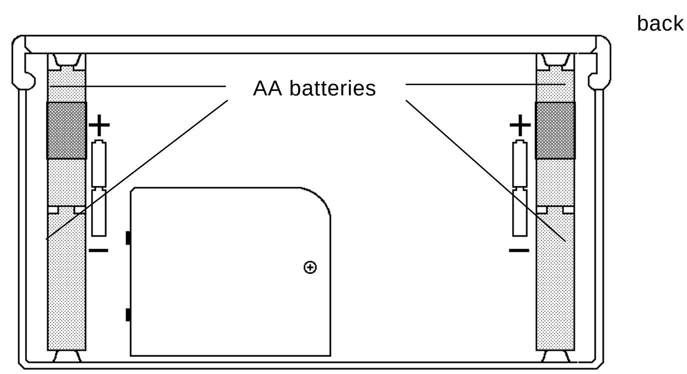
To install the four AA alkaline batteries:

1. Holding the TI-92 unit upright, slide the latch on the top of the unit to the right unlocked position; slide the rear cover down about one-eighth inch and remove it from the main unit.

Important: *When replacing batteries in the future, ensure that the TI-92 is turned off by pressing [2nd] [OFF].*



2. Place the TI-92 face down on a soft cloth to prevent scratching the display face.
3. Install the four AA batteries. Be sure to position the batteries according to the diagram inside the unit. The positive (+) terminal of each battery should point toward the top of the unit.



4. Replace the rear cover and slide the latch on the top of the unit to the left locked position to lock the cover back in place.

Turning the Unit On and Adjusting the Display Contrast

To turn the unit on and adjust the display after installing the batteries:

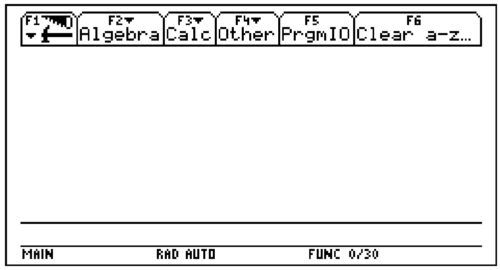
1. Press [ON] to turn the TI-92 on.

   The Home screen is displayed; however, the display contrast may be too dark or too dim to see anything. (When you want to turn the TI-92 off, press [2nd] [OFF].)

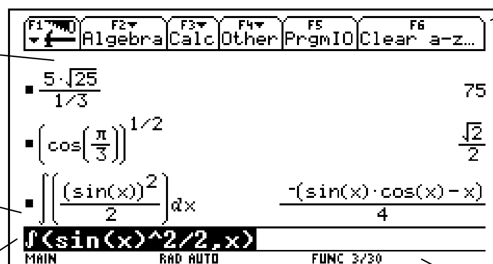
2. To adjust the display to your satisfaction, hold down [♦] (diamond symbol inside a green border) and momentarily press [−] (minus key) to lighten the display. Hold down [♦] and momentarily press [+] (plus key) to darken the display.

About the Home Screen

When you first turn on your TI-92, a blank Home screen is displayed. The Home screen lets you execute instructions, evaluate expressions, and view results.



The following example contains previously entered data and describes the main parts of the Home screen. Entry/answer pairs in the history area are displayed in "pretty print."



History Area
Lists entry/answer pairs you have entered. Pairs scroll up the screen as you make new entries.

Last Entry
Your last entry.

Entry Line
Where you enter expressions or instructions.

Toolbar
Lets you display menus for selecting operations applicable to the Home screen. To display a toolbar menu, press [F1], [F2], etc.

Last Answer
Result of your last entry. Note that results are not displayed on the entry line.

Status Line
Shows the current state of the calculator.

Performing Computations

This section provides several examples for you to perform that demonstrate some of the computational features of the TI-92. The history area in each screen was cleared by pressing [F1] and selecting 8:Clear Home, before performing each example, to illustrate only the results of the example's keystrokes.

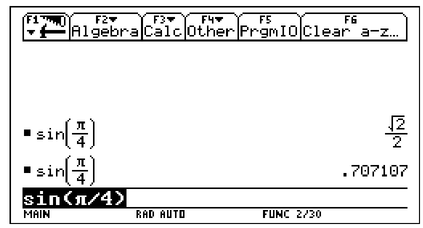
Steps	Keystrokes	Display

Showing Computations

1. Compute sin(π/4) and display the result in symbolic and numeric format.

   To clear the history area of previous calculations, press [F1] *and select* 8:Clear Home.

[SIN] [2nd] [π] [÷] 4 [)] [ENTER] [◆] [ENTER]



Finding the Factorial of Numbers

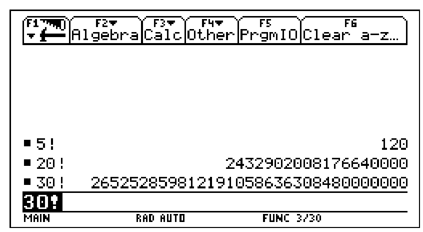
1. Compute the factorial of several numbers to see how the TI-92 handles very large integers.

   To get the factorial operator (!), press [2nd] [MATH], *select* 7:Probability, *and then select* 1:!.

5 [2nd] [MATH] 7 1 [ENTER]

2 0 [2nd] [MATH] 7 1 [ENTER]

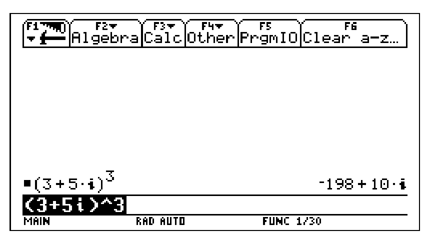
3 0 [2nd] [MATH] 7 1 [ENTER]



Expanding Complex Numbers

1. Compute $(3+5i)^3$ to see how the TI-92 handles computations involving complex numbers.

[(] 3 [+] 5 [2nd] [i] [)] [^] 3 [ENTER]



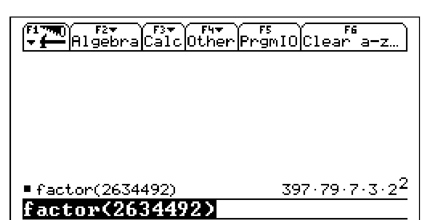
Finding Prime Factors

1. Compute the factors of the rational number 2634492.

   You can enter "factor" on the entry line by typing FACTOR *on the keyboard, or by pressing* [F2] *and selecting* 2:factor(.

2. (*Optional*) Enter other numbers on your own.

F A C T O R [(] 2 6 3 4 4 9 2 [)] [ENTER]



Steps	Keystrokes	Display

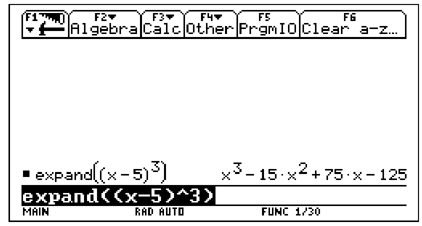
Expanding Expressions

1. Expand the expression $(x-5)^3$.

   You can enter "expand" on the entry line by typing EXPAND *on the keyboard, or by pressing* [F2] *and selecting* 3:expand(.

2. (*Optional*) Enter other expressions on your own.

Keystrokes: E X P A N D [(] [(] X [−] 5 [)] [^] 3 [)] [ENTER]

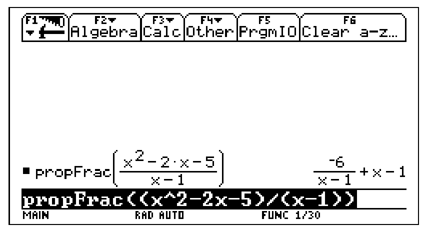


Reducing Expressions

1. Reduce the expression $(x^2-2x-5)/(x-1)$ to its simplest form.

   You can enter "propFrac" on the entry line by typing PROPFRAC *on the keyboard, or by pressing* [F2] *and selecting* 7:propFrac(.

Keystrokes: P R O P F R A C [(] [(] X [^] 2 [−] 2 X [−] 5 [)] [÷] [(] X [−] 1 [)] [)] [ENTER]

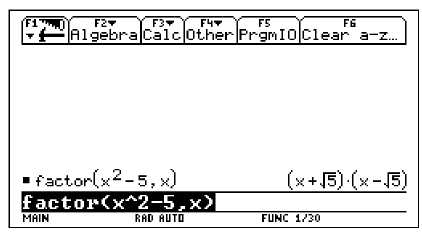


Factoring Polynomials

1. Factor the polynomial (x^2-5) with respect to x.

   You can enter "factor" on the entry line by typing FACTOR *on the keyboard or by pressing* [F2] *and selecting* 2:factor(.

Keystrokes: F A C T O R [(] X [^] 2 [−] 5 [,] X [)] [ENTER]



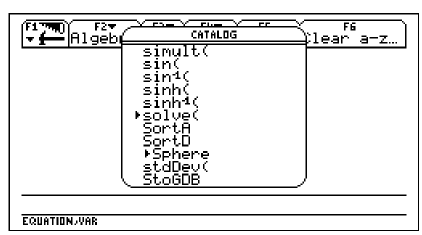
Solving Equations

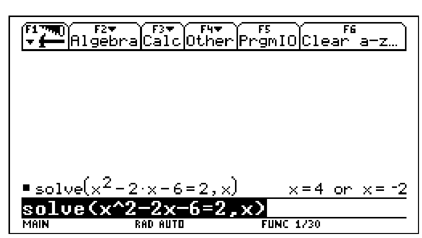
1. Solve the equation $x^2-2x-6=2$ with respect to x.

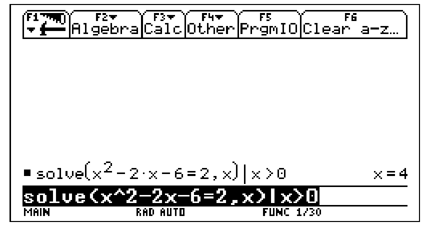
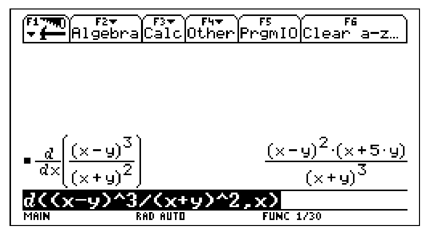
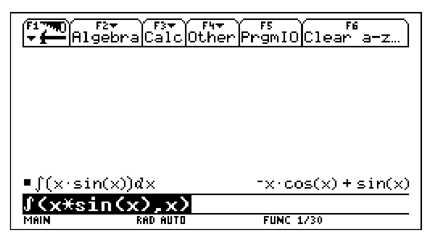
   You can enter "solve(" on the entry line by selecting "solve(" *from the Catalog menu, by typing* SOLVE(*on the keyboard, or by pressing* [F2] *and selecting* 1:solve(.

   The status line area shows the required syntax for the marked item in the Catalog menu.

Keystrokes: [2nd] [CATALOG] S (press ⊙ until the ▶ mark points to s o l v e ( ) [ENTER] X [^] 2 [−] 2 X [−] 6 [=] 2 [,] X [)] [ENTER]



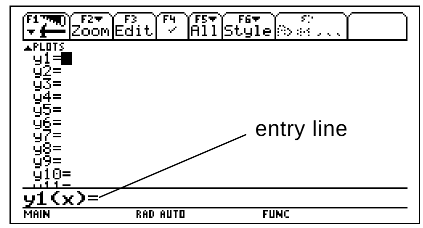
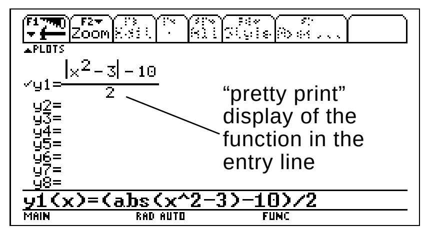
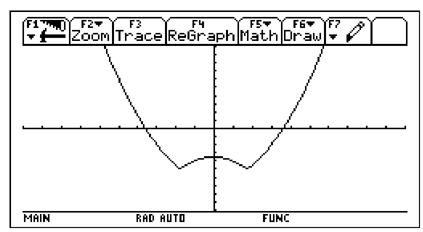
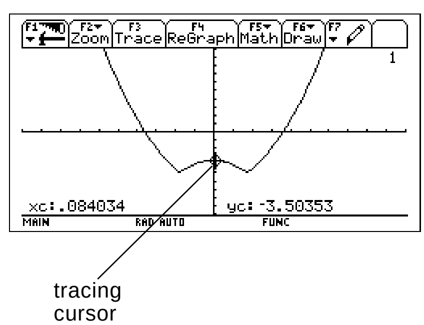


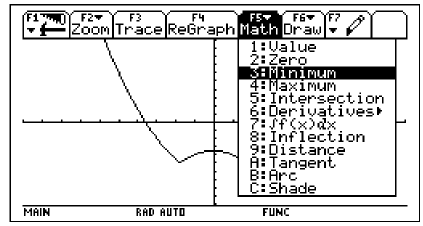
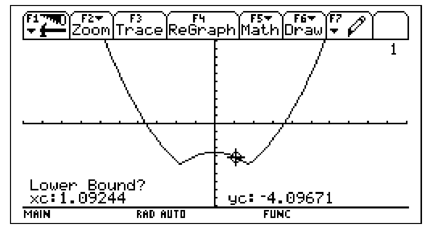
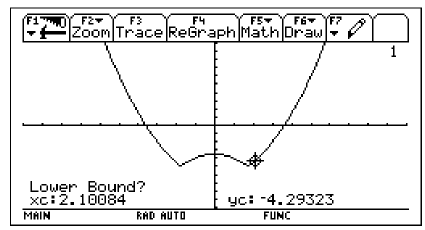
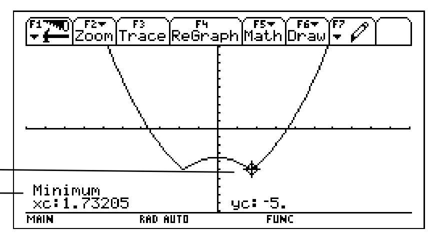
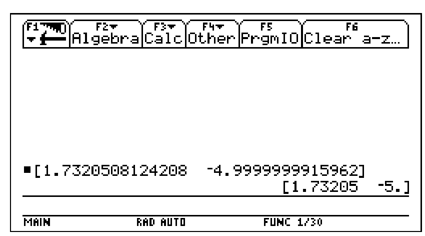
Steps	Keystrokes	Display
Solving Equations with a Domain Constraint		
1. Solve the equation $x^2-2x-6=2$ with respect to x where x is greater than zero. *Pressing [2nd] K produces the “with” (|) operator (domain constraint).*	[2nd] [CATALOG] S (press ⊙ until the ▶ mark points to solve() [ENTER] X [^] 2 [−] 2 X [−] 6 [=] 2 [,] X [)] [2nd] K X [2nd] [>] 0 [ENTER]	
Finding the Derivative of Functions		
1. Find the derivative of $(x-y)^3/(x+y)^2$ with respect to x. *This example illustrates using the calculus differentiation function and how the function is displayed in “pretty print” in the history area.*	[2nd] [d] [(] X [−] Y [)] [^] 3 [÷] [(] X [+] Y [)] [^] 2 [,] X [)] [ENTER]	
Finding the Integral of Functions		
1. Find the integral of x∗sin(x) with respect to x. *This example illustrates using the calculus integration function.*	[2nd] [∫] X [×] [SIN] X [)] [,] X [)] [ENTER]	

Graphing a Function

The example in this section demonstrates some of the graphing capabilities of the TI-92. It illustrates how to graph a function using the Y= Editor. You will learn how to enter a function, produce a graph of the function, trace a curve, find a minimum point, and transfer the minimum coordinates to the Home screen.

Explore the graphing capabilities of the TI-92 by graphing the function $y=(|x^2-3|-10)/2$.

Steps	Keystrokes	Display
1. Display the Y= Editor.	[♦] [Y=]	
2. Enter the function (abs(x^2−3)−10)/2.	[(] A B S [(] X [^] 2 [−] 3 [)] [−] 1 0 [)] [÷] 2 [ENTER]	
3. Display the graph of the function. *Select* 6:ZoomStd *by pressing* 6 *or by moving the cursor to* 6:ZoomStd *and pressing* [ENTER].	[F2] 6	
4. Turn on Trace. *The tracing cursor, and the x and y coordinates are displayed.*	[F3]	

Steps	Keystrokes	Display
5. Open the MATH menu and select 3:Minimum.	[F5] ⊙ ⊙	
6. Set the lower bound. *Press ⊙ (right cursor) to move the tracing cursor until the lower bound for x is just to the left of the minimum node before pressing* [ENTER] *the second time.*	[ENTER] ⊙ . . . ⊙ [ENTER]	
7. Set the upper bound. *Press ⊙ (right cursor) to move the tracing cursor until the upper bound for x is just to the right of the minimum node.*	⊙ . . . ⊙	
8. Find the minimum point on the graph between the lower and upper bounds.	[ENTER] minimum point minimum coordinates	
9. Transfer the result to the Home screen, and then display the Home screen.	[♦] H [♦] [HOME]	

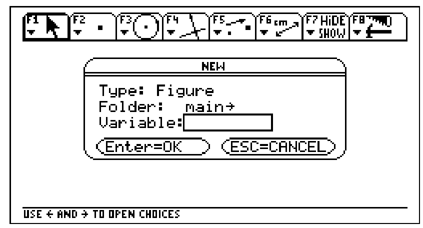
Constructing Geometric Objects

This section provides a multi-part example about constructing geometric objects using the Geometry application of the TI-92. You will learn how to construct a triangle and measure its area, construct perpendicular bisectors to two of the sides, and construct a circle centered at the intersection of the two bisectors that will circumscribe the triangle.

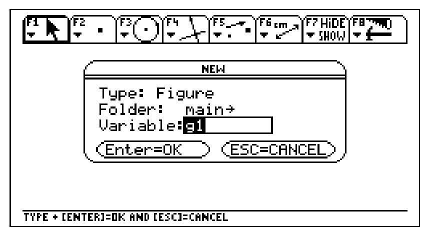
Getting Started in Geometry

To start a Geometry session, you first have to give it a name.

1. Press [APPS] 8 3 to display the New dialog box.

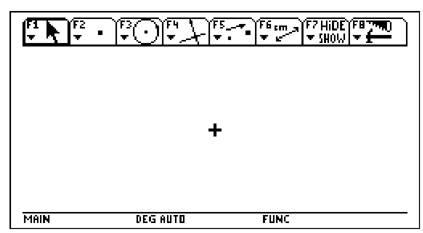


2. Press ⓖ G 1 as the name for the new construction, and press [ENTER].



__Note:__ Each of the following example modules require that you complete the previous module.

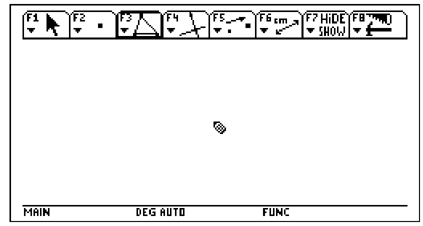
3. Press [ENTER] to display the Geometry drawing window.



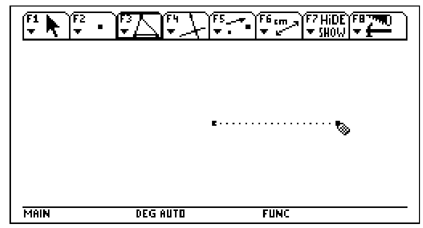
Creating a Triangle

To create a triangle:

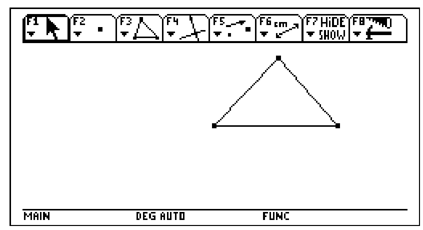
1. Press [F3] and select 3:Triangle.



2. Move the cursor (✎) to the desired location, and press [ENTER] to define the first point.
3. Move the cursor to another location, and press [ENTER] to define the second point.



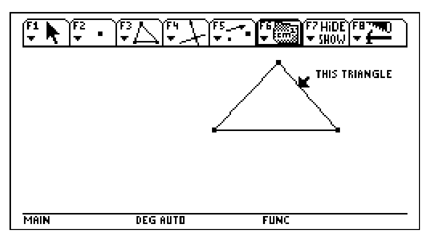
4. Move the cursor to the third location, and press [ENTER] again to complete the triangle.



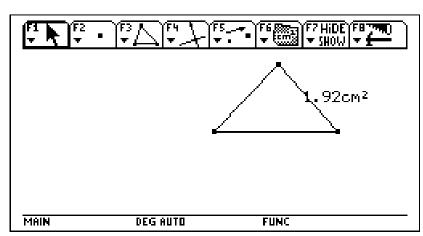
Measuring the Area of the Triangle

To measure the area of the triangle that you constructed in the previous example:

1. Press [F6] and select 2:Area.
2. Move the cursor, if necessary, until “THIS TRIANGLE” is displayed.



3. Press [ENTER] to display the result.

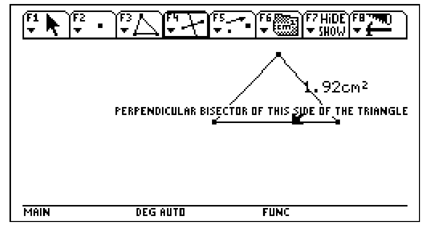


***Note:** Default measurements are in centimeters. See “Setting Application Preferences” in Chapter 7 to change to other unit measurements.*

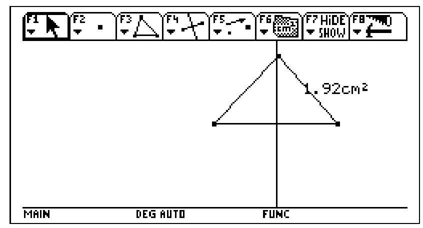
Constructing the Perpendicular Bisectors

To construct the perpendicular bisector to two sides of the triangle:

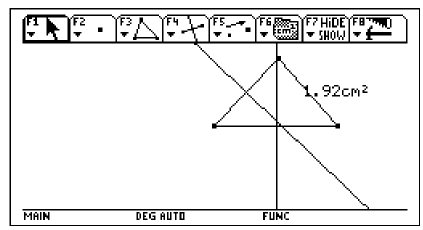
1. Press F4 and select 4:Perpendicular Bisector.



2. Move the cursor close to the triangle until a message is displayed that indicates a side of the triangle.
3. Press ENTER to construct the first bisector.



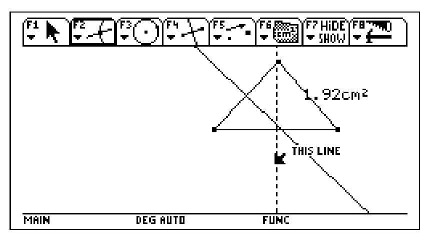
4. Move the cursor to one of the other two sides until the message is displayed (same as step 2), and press ENTER to construct the second bisector.



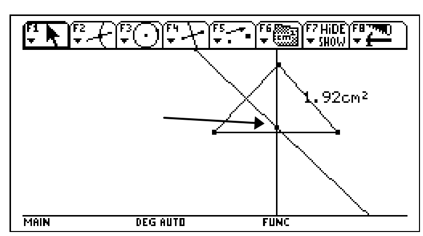
Finding the Intersection Point of Two Lines

To find the intersection point of the two bisectors:

1. Press F2 and select 3:Intersection Point.
2. Select the first line, and then press ENTER.



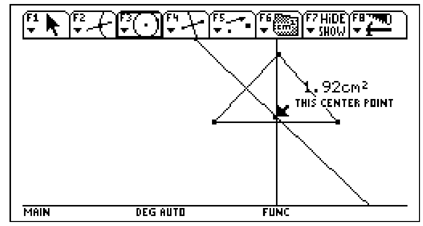
3. Select the second line, and then press ENTER to create the intersection point.



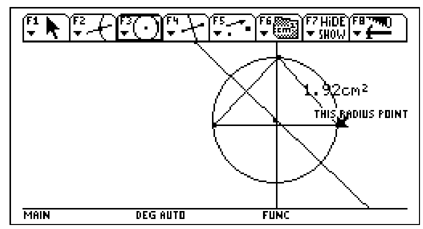
Creating a Circle

To create a circle whose centerpoint is at the intersection of the two bisectors and whose radius is attached to one of the triangle's vertex points:

1. Press [F3] and select 1:Circle.

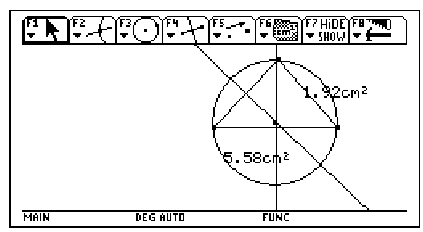


2. Move the cursor to the intersection point of the two perpendicular bisectors, and press [ENTER] to define the centerpoint of the circle.
3. Move the cursor away from the centerpoint to expand the circle until the cursor is near one of the vertices of the triangle and "THIS RADIUS POINT" appears.



Hint: See "Measuring the Area of the Triangle" on the previous page.

4. Press [ENTER] to construct the circle.
5. Measure the area of the circle.

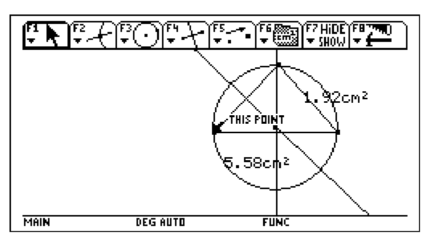


Effects of Modifying the Triangle

This example illustrates the interactive features of the TI-92. You will grab one vertex of the triangle to modify the triangle's shape. The size of the circle, as well as the areas of the triangle and circle, will change accordingly.

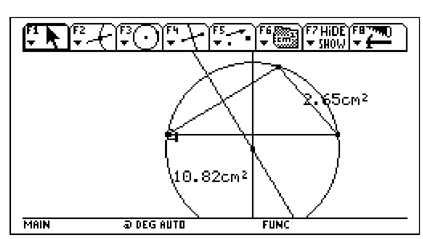
To observe the interactive features of the TI-92:

1. Press [F1] and select 1:Pointer. Move the cursor to one of the intersecting points of the circle and triangle until "THIS POINT" appears, and then press [ENTER].



Note: The circle stays attached to the triangle, and the areas of the triangle and circle change.

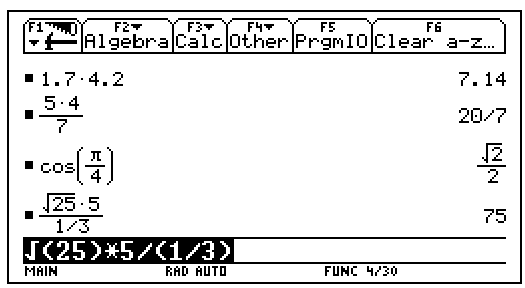
2. Press and hold [dragging hand key] (dragging hand) with your left thumb while pressing the cursor with your right thumb to drag the selected point to its new location.



Operating the TI-92

2

This chapter gives a general overview of the TI-92 and describes its basic operations. By becoming familiar with the information in this chapter, you can use the TI-92 to solve problems more effectively.



The Home screen is the most commonly used application on the TI-92. You can use the Home screen to perform a wide variety of mathematical operations.

Turning the TI-92 On and Off

You can turn the TI-92 on and off manually by using the [ON] and [2nd] [OFF] (or [♦] [OFF]) keys. To prolong battery life, the APD™ (Automatic Power Down) feature lets the TI-92 turn itself off automatically.

Turning the TI-92 On

Press [ON].

- If you turned the unit off by pressing [2nd] [OFF], the TI-92 shows the Home screen as it was when you last used it.
- If you turned the unit off by pressing [♦] [OFF] or if the unit turned itself off through APD, the TI-92 will be exactly as you left it.

Turning the TI-92 Off

Note: [OFF] *is the second function of the* [ON] *key.*

You can use either of the following keys to turn off the TI-92.

Press:	Description
[2nd] [OFF] (press [2nd] and then press [OFF])	Settings and memory contents are retained by the Constant Memory™ feature. However: • You cannot use [2nd] [OFF] if an error message is displayed. • When you turn the TI-92 on again, it always displays the Home screen (regardless of the last application you used).
[♦] [OFF] (press [♦] and then press [OFF])	Similar to [2nd] [OFF] except: • You can use [♦] [OFF] if an error message is displayed. • When you turn the TI-92 on again, it will be exactly as you left it.

APD (Automatic Power Down)

After several minutes without any activity, the TI-92 turns itself off automatically. This feature is called APD.

When you press [ON], the TI-92 will be exactly as you left it.

- The display, cursor, and any error conditions are exactly as you left them.
- All settings and memory contents are retained.

APD does not occur if a calculation or program is in progress, unless the program is paused.

Batteries

The TI-92 uses four AA alkaline batteries and a back-up lithium battery. To replace the batteries without losing any information stored in memory, follow the directions in Appendix C.

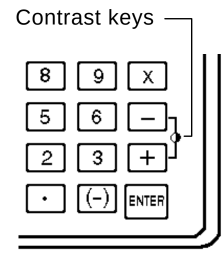
Setting the Display Contrast

The brightness and contrast of the display depend on room lighting, battery freshness, viewing angle, and the adjustment of the display contrast. The contrast setting is retained in memory when the TI-92 is turned off.

Adjusting the Display Contrast

You can adjust the display contrast to suit your viewing angle and lighting conditions.

To:	Press and hold both:
Increase (darken) the contrast	[◆] and [+]
Decrease (lighten) the contrast	[◆] and [−]

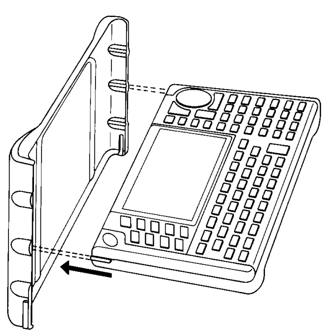


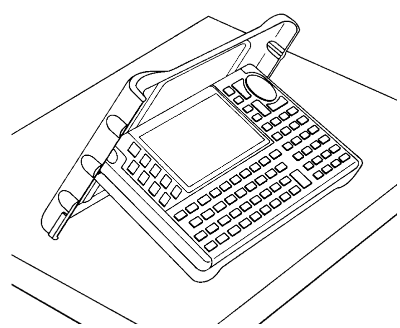
If you press and hold [◆] [+] or [◆] [−] too long, the display may go completely black or blank. To make finer adjustments, hold [◆] and then tap [+] or [−].

Using the Snap-on Cover as a Stand

When using the TI-92 on a desk or table top, you can use the snap-on cover to prop up the unit at one of three angles. This may make it easier to view the display under various lighting conditions.

__Note:__ Slide the tabs at the top-sides of the TI-92 into the slots in the cover.





When to Replace Batteries

__Tip:__ The display may be very dark after you change batteries. Use [◆] [−] to lighten the display.

As the batteries get low, the display begins to dim (especially during calculations) and you must increase the contrast. If you have to increase the contrast frequently, replace the four AA batteries.

The status line along the bottom of the display also gives battery information.

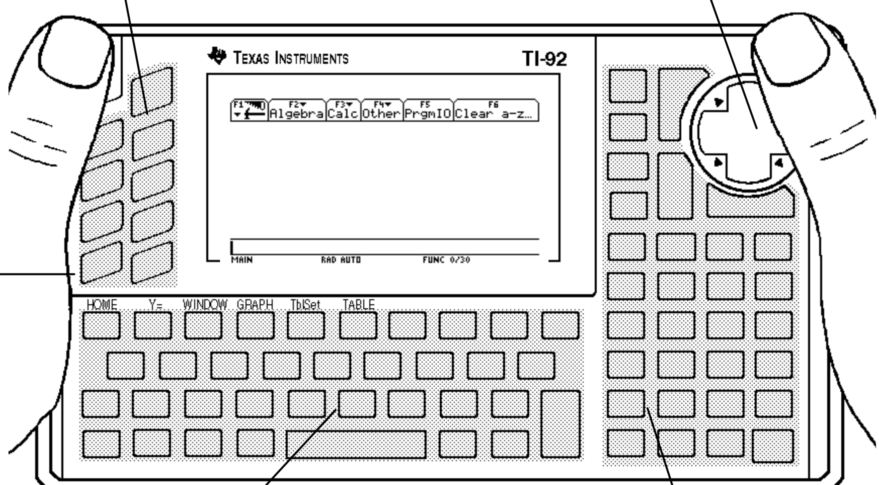
Indicator in status line	Description
BATT	Batteries are low.
BATT (reversed)	Replace batteries as soon as possible.

The Keyboard

With the TI-92's easy-to-hold shape and keyboard layout, you can quickly access any area of the keyboard even when you are holding the unit with two hands.

Keyboard Areas

The keyboard is divided into several areas of related keys.

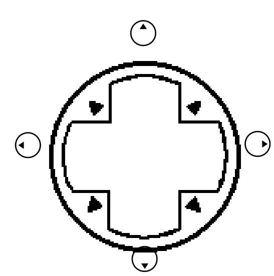


Cursor Pad

To move the cursor, press the applicable edge of the cursor pad. This guidebook uses key symbols such as ← and → to indicate which side of the cursor pad to press.

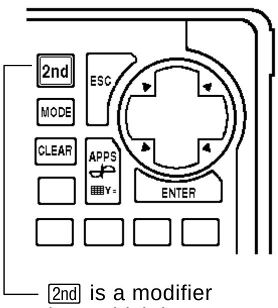
For example, press → to move the cursor to the right.

Note: The diagonal directions (↖, etc.) are used only for geometry and graphing applications.



Important Keys You Should Know About

The area around the cursor pad contains several keys that are important for using the TI-92 effectively.

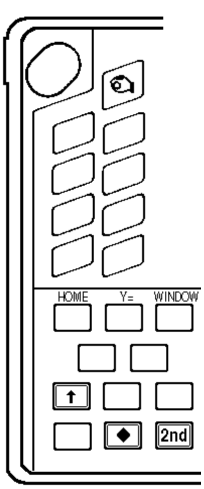


[2nd] is a modifier key, which is described below.

Key	Description
[APPS]	Displays a menu that lists all the applications available on the TI-92 and lets you select the one you want. Refer to page 33.
[ESC]	Cancels any menu or dialog box.
[ENTER]	Evaluates an expression, executes an instruction, selects a menu item, etc. Because this is commonly used in a variety of operations, the TI-92 has three [ENTER] keys placed at convenient locations.
[MODE]	Displays a list of the TI-92's current mode settings, which determine how numbers and graphs are interpreted, calculated, and displayed. You can change the settings as needed. Refer to "Setting Modes" on page 35.
[CLEAR]	Clears (erases) the entry line. Also used to delete an entry/answer pair in the history area.

Modifier Keys

Most keys can perform two or more functions, depending on whether you first press a modifier key.



Modifier	Description
[2nd] (Second)	Accesses the second function of the next key you press. On the keyboard, second functions are printed in the same color as the [2nd] key. The TI-92 has two [2nd] keys conveniently placed at opposite corners of the keyboard.
[♦] (Diamond)	Activates "shortcut" keys that select applications and certain menu items directly from the keyboard. On the keyboard, application shortcuts are printed in the same color as the [♦] key. Refer to page 34.
[↑] (Shift)	Types an uppercase character for the next letter key you press. [↑] is also used with ▶ and ◀ to highlight characters in the entry line for editing purposes.
[✋] (Hand)	Used with the cursor pad to manipulate geometric objects. [✋] is also used for drawing on a graph.

2nd Functions

On the TI-92's keyboard, a key's second function is printed above the key. For example:

SIN^{-1} ——— Second function
[SIN] ——— Primary function

Note: On the keyboard, second functions are printed in the same color as the [2nd] key.

To access a second function, press the [2nd] key and then press the key for that second function.

In this guidebook:

- Primary functions are shown in a box, such as [SIN].
- Second functions are shown in brackets, such as [2nd] [SIN^{-1}].

When you press [2nd], 2ND is shown in the status line at the bottom of the display. This indicates that the TI-92 will use the second function, if any, of the next key you press. If you press [2nd] by accident, press [2nd] again (or press [ESC]) to cancel its effect.

Entering Uppercase Letters with Shift ([↑]) or Caps Lock

Normally, the QWERTY keyboard types lowercase letters. To type uppercase letters, use Shift and Caps Lock just as on a typewriter.

To:	Do this:
Type a single uppercase letter	Press [↑] and then the letter key. • To type multiple uppercase letters, hold [↑] or use Caps Lock. • When Caps Lock is on, [↑] has no effect.
Toggle Caps Lock on or off	Press [2nd] [CAPS].

If You Need to Enter Special Characters

You can also use the QWERTY keyboard to enter a variety of special characters. For more information, refer to "Entering Special Characters" in Chapter 16.

Home Screen

When you first turn on your TI-92, the Home screen is displayed. The Home screen lets you execute instructions, evaluate expressions, and view results.

Displaying the Home Screen

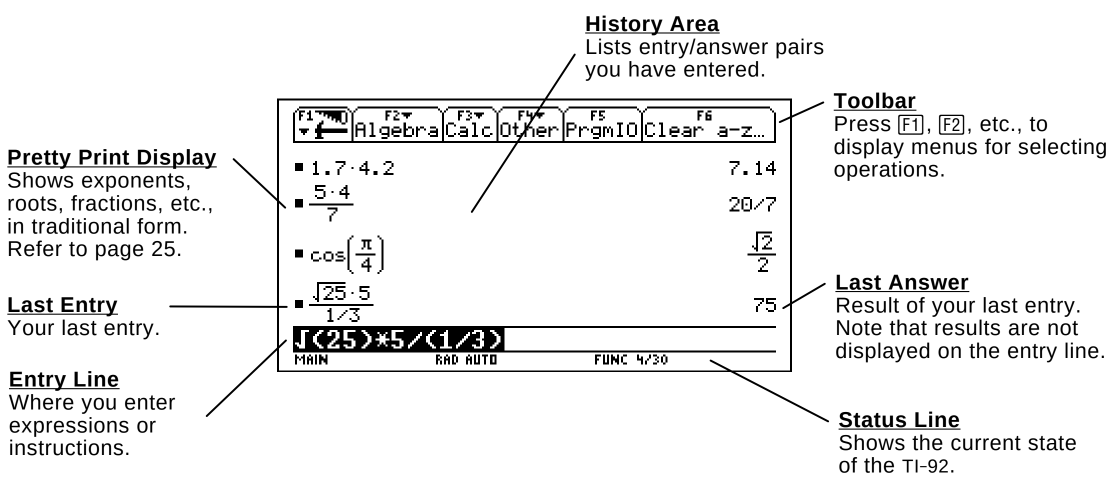
When you turn on the TI-92 after it has been turned off with [2nd] [OFF], the display always shows the Home screen. (If the TI-92 turned itself off through APD, the display shows the previous screen, which may or may not have been the Home screen.)

To display the Home screen at any time:

- Press [◆] [HOME].
  — or —
- Press [2nd] [QUIT].
  — or —
- Press [APPS] [ENTER] or [APPS] 1.

Parts of the Home Screen

The following example gives a brief description of the main parts of the Home screen.



History Area

The history area shows up to eight previous entry/answer pairs (depending on the complexity and height of the displayed expressions). When the display is filled, information scrolls off the top of the screen. You can use the history area to:

- Review previous entries and answers. You can use the cursor to view entries and answers that have scrolled off the screen.
- Recall or auto-paste a previous entry or answer onto the entry line so that you can re-use or edit it. Refer to pages 41 and 42.

Scrolling through the History Area

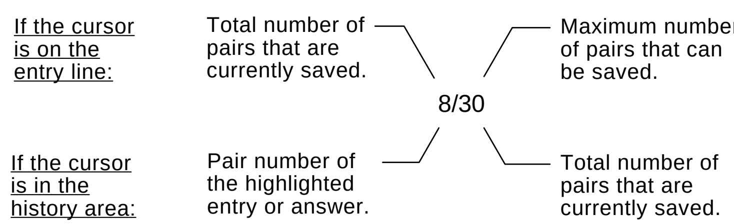
Normally, the cursor is in the entry line. However, you can move the cursor into the history area.

To:	Do this:
View entries or answers that have scrolled off the screen	1. From the entry line, press ↑ to highlight the last answer. 2. Continue using ↑ to move the cursor from answer to entry, up through the history area.
View an entry or answer that is too long for one line (▶ is at end of line)	Move the cursor to the entry or answer. Use → and ← to scroll left and right (or [2nd] → and [2nd] ← to go to the end or the beginning), respectively.
Return the cursor to the entry line	Press [ESC], or press ↓ until the cursor is back on the entry line.

***Note:** For an example of viewing a long answer, refer to page 24.*

History Information on the Status Line

Use the history indicator on the status line for information about the entry/answer pairs. For example:



By default, the last 30 entry/answer pairs are saved. If the history area is full when you make a new entry (indicated by 30/30), the new entry/answer pair is saved and the oldest pair is deleted. The history indicator does not change.

Modifying the History Area

To:	Do this:
Change the number of pairs that can be saved	Press [F1] and select 9:Format, or press [♦] F. Then press →, use ↑ or ↓ to highlight the new number, and press [ENTER] twice.
Clear the history area and delete all saved pairs	Press [F1] and select 8:Clear Home, or enter **ClrHome** on the entry line.
Delete a particular entry/answer pair	Move the cursor to either the entry or answer. Press [←] or [CLEAR].

Entering Numbers

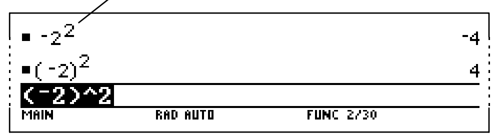
The TI-92's keypad lets you enter positive and negative numbers for your calculations. You can also enter numbers in scientific notation.

Entering a Negative Number

1. Press the negation key [(-)]. (Do not use the subtraction key [−].)
2. Type the number.

To see how the TI-92 evaluates a negation in relation to other functions, refer to the Equation Operating System (EOS) hierarchy in Appendix B. For example, it is important to know that functions such as x^2 are evaluated before negation.

Use [(] and [)] to include parentheses if you have any doubt about how a negation will be evaluated.



__Important:__ Use [−] for subtraction and use [(-)] for negation.

If you use [−] instead of [(-)] (or vice versa), you may get an error message or you may get unexpected results. For example:

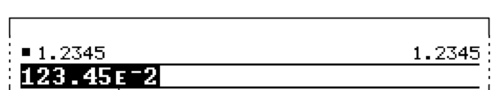
- 9 [×] [(-)] 7 = ⁻63
  — but —
  9 [×] [−] 7 displays an error message.
- 6 [−] 2 = 4
  — but —
  6 [(-)] 2 = ⁻12 since it is interpreted as 6(⁻2), implied multiplication.
- [(-)] 2 [+] 4 = 2
  — but —
  [−] 2 [+] 4 subtracts 2 from the previous answer and then adds 4.

Entering a Number in Scientific Notation

1. Type the part of the number that precedes the exponent. This value can be an expression.
2. Press [2nd] [EE]. E appears in the display.
3. Type the exponent as an integer with up to 3 digits. You can use a negative exponent.

Entering a number in scientific notation does not cause the answers to be displayed in scientific or engineering notation.

The display format is determined by the mode settings (pages 25 through 27) and the magnitude of the number.



Entering Expressions and Instructions

You perform a calculation by evaluating an expression. You initiate an action by executing the appropriate instruction. Expressions are calculated and results are displayed according to the mode settings described on page 25.

Definitions

Note: *Appendix A describes all of the TI-92's built-in functions and instructions.*

Note: *This guidebook uses the word* ***command*** *as a generic reference to both functions and instructions.*

Expression — Consists of numbers, variables, operators, functions, and their arguments that evaluate to a single answer. For example: πr^2+3.

- Enter an expression in the same order that it normally is written.
- In most places where you are required to enter a value, you can enter an expression.

Operator — Performs an operation such as +, −, ∗, ^.

- Operators require an argument before and after the operator. For example: 4+5 and 5^2.

Function — Returns a value.

- Functions require one or more arguments (enclosed in parentheses) after the function. For example: √(5) and **min**(5,8).

Instruction — Initiates an action.

- Instructions cannot be used in expressions.
- Some instructions do not require an argument. For example: **ClrHome**.
- Some require one or more arguments. For example: **Circle** 0,0,5.

For instructions, do not put the arguments in parentheses.

Implied Multiplication

The TI-92 recognizes implied multiplication, provided it does not conflict with a reserved notation.

	If you enter:	The TI-92 interprets it as:
Valid	2π	2∗π
	4 sin(46)	4∗sin(46)
	5(1+2) or (1+2)5	5∗(1+2) or (1+2)∗5
	[1,2]a	[a 2a]
	2(a)	2∗a
Invalid	xy	Single variable named xy
	a(2)	Function call
	a[1,2]	Matrix index to element a[1,2]

Parentheses

Expressions are evaluated according to the Equation Operating System (EOS) hierarchy described in Appendix B. To change the order of evaluation or just to ensure that an expression is evaluated in the order you require, use parentheses.

Calculations inside a pair of parentheses are completed first. For example, in 4(1+2), EOS first evaluates (1+2) and then multiplies the answer by 4.

Entering an Expression

Type the expression, and then press [ENTER] to evaluate it. To enter a function or instruction name on the entry line, you can:

- Press its key, if available. For example, press [SIN].
  — or —
- Select it from a menu, if available. For example, select 2:abs from the Number submenu of the MATH menu.
  — or —
- Type the name letter-by-letter from the keyboard. You can use any mixture of uppercase or lowercase letters. For example, type **sin(** or **Sin(** .

Example

Calculate 3.76 ÷ (⁻7.9 + √5) + 2 log 45.

3.76 [÷] [(] [(-)] 7.9 [+] [2nd] [√]

`3.76/(⁻7.9+√(`

[2nd] [√] inserts "√(" because its argument must be in parentheses.

5 [)] [)]

`3.76/(⁻7.9+√(5))`

Use [)] once to close √(5) and again to close (⁻7.9 + √5).

[+] 2 LOG [(] 45 [)]

Type the function name.

`3.76/(⁻7.9+√(5))+2log(45)`

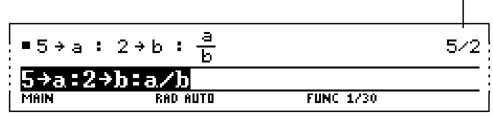
log requires ( ) around its argument.

[ENTER]

Entering Multiple Expressions on a Line

To enter more than one expression or instruction at a time, separate them with a colon by pressing [2nd] [:].

Displays the last result only.

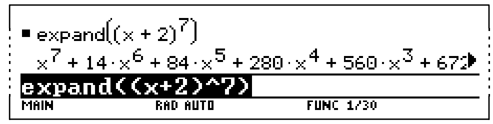


→ is displayed when you press [STO▸] to store a value to a variable.

If an Entry or Answer Is Too Long for One Line

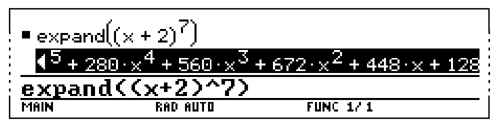
In the history area, if both the entry and its answer cannot be displayed on one line, the answer is displayed on the next line.

If an entry or answer is too long to fit on one line, ▶ is displayed at the end of the line.



To view the entire entry or answer:

1. Press ⓤ to move the cursor from the entry line up into the history area. This highlights the last answer.
2. As necessary, use ⓤ and ⓓ to highlight the entry or answer you want to view. For example, ⓤ moves from answer to entry, up through the history area.
3. Use ⓡ and ⓛ or [2nd] ⓡ and [2nd] ⓛ to scroll right and left.



Note: When you scroll to the right, ◀ is displayed at the beginning of the line.

4. To return to the entry line, press [ESC].

Continuing a Calculation

When you press [ENTER] to evaluate an expression, the TI-92 leaves the expression on the entry line and highlights it. You can continue to use the last answer or enter a new expression.

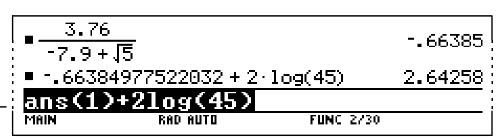
If you press:	The TI-92:
[+], [−], [×], [÷], [^], or [STO▶]	Replaces the entry line with the variable ans(1), which lets you use the last answer as the beginning of another expression.
Any other key	Erases the entry line and begins a new entry.

Example

Calculate 3.76 ÷ (⁻7.9 + √5). Then add 2 log 45 to the result.

3.76 [÷] [(] [(-)] 7.9 [+] [2nd] [√] 5 [)] [)] [ENTER]

[+] 2 LOG [(] 45 [)] [ENTER]

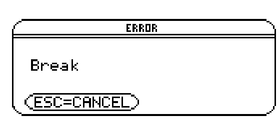


When you press [+], the entry line is replaced with the variable ans(1), which contains the last answer.

Stopping a Calculation

When a calculation is in progress, the BUSY indicator appears on the right end of the status line. To stop the calculation, press [ON].

There may be a delay before the “break” message is displayed.



Press [ESC] to return to the current application.

Formats of Displayed Results

A result may be calculated and displayed in any of several formats. This section describes the TI-92 modes and their settings that affect the display formats. To check or change your current mode settings, refer to page 35.

Pretty Print Mode

By default, Pretty Print = ON. Exponents, roots, fractions, etc., are displayed in the same form in which they are traditionally written. You can use [MODE] to turn pretty print off and on.

Pretty Print ON	Pretty Print OFF
$\pi^2, \frac{\pi}{2}, \sqrt{\frac{x-3}{2}}$	π^2, π/2, √((x−3)/2)

The entry line does not show an expression in pretty print. If pretty print is turned on, the history area will show both the entry and its result in pretty print after you press [ENTER].

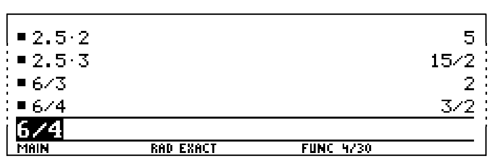
Exact/Approx Mode

By default, Exact/Approx = AUTO. You can use [MODE] to select from three settings.

Because AUTO is a combination of the other two settings, you should be familiar with all three settings.

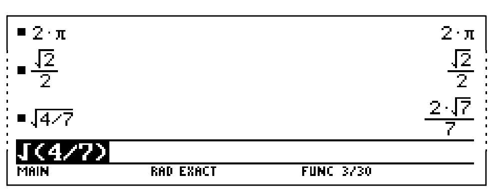
Note: By retaining fractional and symbolic forms, EXACT reduces rounding errors that could be introduced by intermediate results in chained calculations.

EXACT — Any result that is not a whole number is displayed in a fractional or symbolic form (1/2, π, $\sqrt{2}$, etc.).



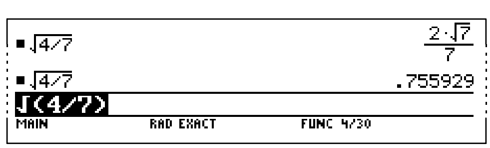
Shows whole-number results.

Shows simplified fractional results.



Shows symbolic π.

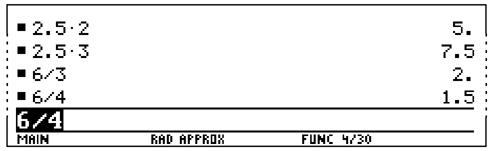
Shows symbolic form of roots that cannot be evaluated to a whole number.



Press [♦] [ENTER] to temporarily override the EXACT setting and display a floating-point result.

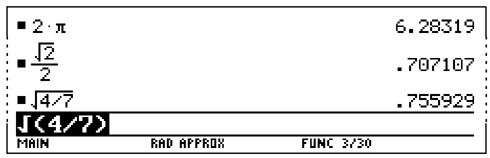
Exact/Approx Mode (Continued)

APPROXIMATE — All numeric results, where possible, are displayed in floating-point (decimal) form.



Fractional results are evaluated numerically.

Note: *Results are rounded to the precision of the TI-92 and displayed according to current mode settings.*

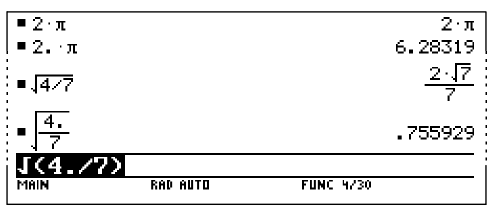


Symbolic forms, where possible, are evaluated numerically.

Because undefined variables cannot be evaluated, they are treated algebraically. For example, if the variable r is undefined, $\pi r^2 = 3.14159 \cdot r^2$.

AUTO — Uses the EXACT form where possible, but uses the APPROXIMATE form when your entry contains a decimal point. Also, certain functions may display APPROXIMATE results even if your entry does not contain a decimal point.

Tip: *To retain an EXACT form, use fractions instead of decimals. For example, use 3/2 instead of 1.5.*



A decimal in the entry forces a floating-point result.

The following chart compares the three settings.

Entry	Exact Result	Approximate Result	Auto Result
8/4	2	2.	2
8/6	4/3	1.33333	4/3
8.5∗3	51/2	25.5	25.5
√(2)/2	$\frac{\sqrt{2}}{2}$	.707107	$\frac{\sqrt{2}}{2}$
π∗2	2·π	6.28319	2·π
π∗2.	2·π	6.28319	6.28319

A decimal in the entry forces a floating-point result in AUTO.

Tip: *To evaluate an entry in APPROXIMATE form, regardless of the current setting, press [♦] [ENTER].*

Display Digits Mode

By default, Display Digits = FLOAT 6, which means that results are rounded to a maximum of six digits. You can use [MODE] to select different settings. The settings apply to all exponential formats.

Internally, the TI-92 calculates and retains all decimal results with up to 14 significant digits (although a maximum of 12 are displayed).

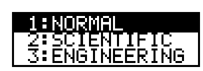
Note: *Regardless of the Display Digits setting, the full value is used for internal floating-point calculations to ensure maximum accuracy.*

Note: *A result is automatically shown in scientific notation if its magnitude cannot be displayed in the selected number of digits.*

Setting	Example		Description
FIX (0 – 12)	123. 123.5 123.46 123.457	(FIX 0) (FIX 1) (FIX 2) (FIX 3)	Results are rounded to the selected number of decimal places.
FLOAT	123.456789012		Number of decimal places varies, depending on the result.
FLOAT (1 – 12)	1.E 2 1.2E 2 123. 123.5 123.46 123.457	(FLOAT 1) (FLOAT 2) (FLOAT 3) (FLOAT 4) (FLOAT 5) (FLOAT 6)	Results are rounded to the total number of selected digits.

Exponential Format Mode

By default, Exponential Format = NORMAL. You can use [MODE] to select from three settings.



Note: *In the history area, a number in an entry is displayed in SCIENTIFIC if its absolute value is less than .001.*

Setting	Example	Description
NORMAL	12345.6	If a result cannot be displayed in the number of digits specified by the Display Digits mode, the TI-92 switches from NORMAL to SCIENTIFIC for that result only.
SCIENTIFIC	1.23456E 4	1.23456×10^4 Exponent (power of 10). Always 1 digit to the left of the decimal point.
ENGINEERING	12.3456E 3	12.3456×10^3 Exponent is a multiple of 3. May have 1, 2, or 3 digits to the left of the decimal point.

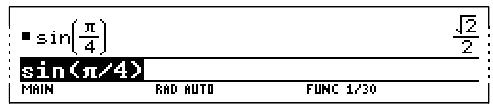
Editing an Expression in the Entry Line

Knowing how to edit an entry can be a real time-saver. If you make an error while typing an expression, it's often easier to correct the mistake than to retype the entire expression.

Removing the Highlight from the Previous Entry

After you press [ENTER] to evaluate an expression, the TI-92 leaves that expression on the entry line and highlights it. To edit the expression, you must first remove the highlight; otherwise, you may clear the expression accidentally by typing over it.

To remove the highlight, move the cursor toward the side of the expression you want to edit.



▶ moves the cursor to the end of the expression.

◀ moves the cursor to the beginning.

Moving the Cursor

After removing the highlight, move the cursor to the applicable position within the expression.

__Note:__ If you accidentally press ▲ instead of ◀ or ▶, the cursor moves up into the history area. Press [ESC] or press ▼ until the cursor returns to the entry line.

To move the cursor:	Press:	
Left or right within an expression.	◀ or ▶	Hold the pad to repeat the movement.
To the beginning of the expression.	[2nd] ◀	
To the end of the expression.	[2nd] ▶	

Deleting a Character

To delete:	Press:	
The character to the left of the cursor.	[⇐]	Hold [⇐] to delete multiple characters.
The character to the right of the cursor.	[♦] [⇐]	
All characters to the right of the cursor.	[CLEAR] (once only)	If there are no characters to the right of the cursor, [CLEAR] erases the entire entry line.

Clearing the Entry Line

To clear the entry line, press:

- [CLEAR] if the cursor is at the beginning or end of the entry line.
  — or —
- [CLEAR] [CLEAR] if the cursor is not at the beginning or end of the entry line. The first press deletes all characters to the right of the cursor, and the second clears the entry line.

Inserting or Overtyping a Character

The TI-92 has both an insert and an overtype mode. By default, the TI-92 is in the insert mode. To toggle between the insert and overtype modes, press [2nd] [INS].

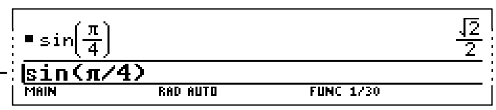
Tip: Look at the cursor to see if you're in insert or overtype mode.

If the TI-92 is in:	The next character you type:
Insert mode Thin cursor between characters	Will be inserted at the cursor.
Overtype mode Cursor highlights a character	Will replace the highlighted character.

Replacing or Deleting Multiple Characters

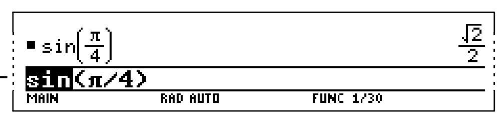
First, highlight the applicable characters. Then, replace or delete all the highlighted characters.

<table>
<tr><th>To:</th><th>Do this:</th></tr>
<tr><td>Highlight multiple characters</td><td>1. Move the cursor to either side of the characters you want to highlight.



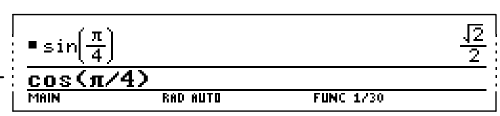
To replace **sin** with **cos**, place the cursor beside **sin**.

2. Hold [↑] and press ◀ or ▶ to highlight characters left or right of the cursor.



Hold [↑] and press ▶ ▶ ▶.</td></tr>
<tr><td>Replace the highlighted characters

— or —</td><td>Type the new characters.



Type COS.</td></tr>
<tr><td>Delete the highlighted characters</td><td>Press [←].</td></tr>
</table>

Tip: When you highlight characters to replace, remember that some function keys automatically add an open parenthesis. For example, pressing [COS] types **cos(**.

To leave the keyboard uncluttered, the TI-92 uses menus to access many operations. This section gives an overview of how to select an item from any menu. Specific menus are described in the appropriate chapters of this guidebook.

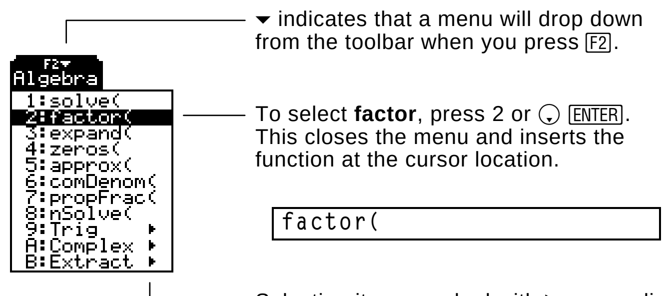
Displaying a Menu

Press:	To display:
F1, F2, etc.	A toolbar menu — Drops down from the toolbar at the top of most application screens. Lets you select operations useful for that application.
APPS	APPLICATIONS menu — Lets you select from the list of TI-92 applications. Refer to page 33.
2nd [CHAR]	CHAR menu — Lets you select from categories of special characters (Greek, math, etc.).
2nd [MATH]	MATH menu — Lets you select from categories of math operations.
2nd [CATALOG]	CATALOG menu — Lets you select from a complete, alphabetic list of the TI-92's built-in functions and instructions.

Selecting an Item from a Menu

To select an item from the displayed menu, either:

- Press the number or letter shown to the left of that item.
  — or —
- Use the cursor pad ⊙ and ⊙ to highlight the item, and then press ENTER. (Note that pressing ⊙ from the first item does *not* move the highlight to the last item, nor vice versa.)



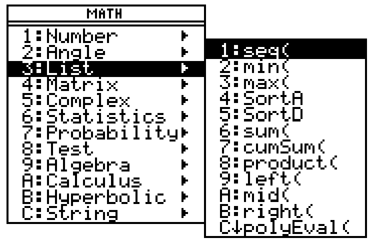
▾ indicates that a menu will drop down from the toolbar when you press F2.

To select **factor**, press 2 or ⊙ ENTER. This closes the menu and inserts the function at the cursor location.

Selecting items marked with ▸ or . . . displays a submenu or dialog box, respectively.

Items Ending with ▶ (Submenus)

If you select a menu item ending with ▶, a submenu is displayed. You then select an item from the submenu.



For example, **List** displays a submenu that lets you select a specific List function.

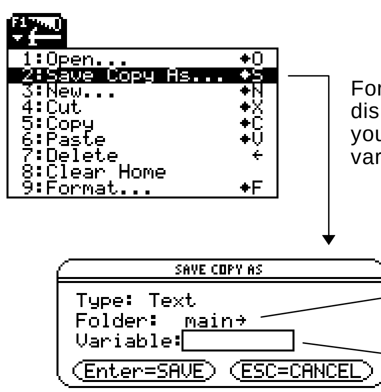
↓ indicates that you can use the cursor pad to scroll down for additional items.

For items that have a submenu, you can use the cursor pad as described below.

- To display the submenu for the highlighted item, press ⊙. (This is the same as selecting that item.)
- To cancel the submenu without making a selection, press ⊙. (This is the same as pressing [ESC].)

Items Containing "..." (Dialog Boxes)

If you select a menu item containing ". . ." (ellipsis marks), a dialog box is displayed for you to enter additional information.



For example, **Save Copy As ...** displays a dialog box that prompts you to enter a folder name and a variable name.

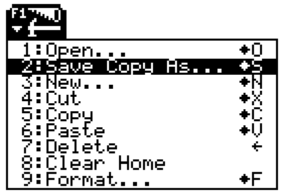
→ indicates that you can press ⊙ to display and select from a menu.

An input box indicates that you must type a value.

After typing in an input box such as Variable, you must press [ENTER] twice to save the information and close the dialog box.

Keyboard Shortcuts

You can select certain menu items directly from the keyboard, without first having to display a menu. If an item has a keyboard shortcut, it is indicated on the menu.



Without even displaying this menu, you can press [♦] S to select **Save Copy As**.

Moving from One Toolbar Menu to Another

To move from one toolbar menu to another without making a selection, either:

- Press the key ([F1], [F2], etc.) for the other toolbar menu.
  — or —
- Use the cursor pad to move to the next (press ▸) or previous (press ◂) toolbar menu. Pressing ▸ from the last menu moves to the first menu, and vice versa.

When using ▸, be sure that an item with a submenu is not highlighted. If so, ▸ displays that item's submenu instead of moving to the next toolbar menu.

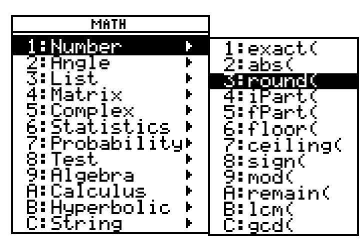
Canceling a Menu

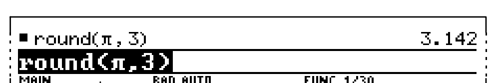
To cancel the current menu without making a selection, press [ESC]. Depending on whether any submenus are displayed, you may need to press [ESC] several times to cancel all displayed menus.

Example: Selecting a Menu Item

Round the value of π to three decimal places. Starting from a clear entry line on the Home screen:

1. Press [2nd] [MATH] to display the MATH menu.
2. Press 1 to display the Number submenu. (Or press [ENTER] since the first item is automatically highlighted.)
3. Press 3 to select **round**. (Or press ▾ ▾ and [ENTER].)
4. Press [2nd] [π] [,] 3 [)] and then [ENTER] to evaluate the expression.





Selecting the function in Step 3 automatically typed **round(** on the entry line.

Selecting an Application

The TI-92 has different applications that let you solve and explore a variety of problems. You can select an application from a menu, or you can access commonly used applications directly from the keyboard.

From the APPLICATIONS Menu

__Note:__ To cancel the menu without making a selection, press [ESC].

1. Press [APPS] to display a menu that lists the applications.
2. Select an application. Either:

- Use the cursor pad ↓ or ↑ to highlight the application and then press [ENTER].

  — or —
- Press the number for that application.

Application:	Lets you:
Home	Enter expressions and instructions, and perform calculations.
Y= Editor	Define, edit, and select functions or equations for graphing (Chapter 3 and Chapters 11 – 15).
Window Editor	Set window dimensions for viewing a graph (Chapter 3).
Graph	Display graphs (Chapter 3).
Table	Display a table of variable values that correspond to an entered function (Chapter 4).
Data/Matrix Editor	Enter and edit lists, data, and matrices. You can perform statistical calculations and graph statistical plots (Chapters 8 and 9).
Program Editor	Enter and edit programs and functions (Chapter 17).
Geometry	Construct geometric objects, and perform analytical and transformational operations (Chapter 7).
Text Editor	Enter and edit a text session (Chapter 16).

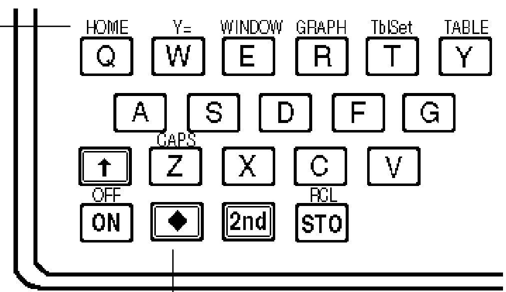
From the Keyboard

You can access six commonly used applications from the QWERTY keyboard.

1. Press the diamond ([♦]) key.
2. Press the QWERTY key for the application.

Note: On your keyboard, the application names above Q, W, etc., are printed in the same color as the [♦] key.

Applications are listed above the QWERTY keys.



Diamond key

For example, press [♦] and then Q to display the Home screen. This guidebook uses the notation [♦] [HOME], similar to the notation used for second functions.

Setting Modes

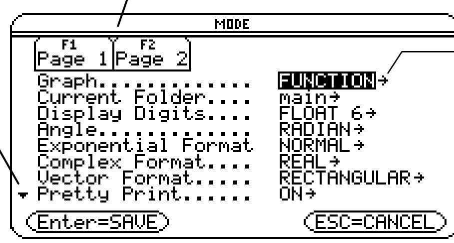
Modes control how numbers and graphs are displayed and interpreted. Mode settings are retained by the Constant Memory™ feature when the TI-92 is turned off. All numbers, including elements of matrices and lists, are displayed according to the current mode settings.

Checking Mode Settings

Press [MODE] to display the MODE dialog box, which lists the modes and their current settings.

There are two pages of mode listings. Press [F1] or [F2] to quickly display the first or second page.

Indicates you can scroll down to see additional modes.



Indicates that you can press → or ← to display and select from a menu.

Note: Modes that are not currently valid are dimmed. For example, on the second page, Split 2 App is not valid when Split Screen = FULL. When you scroll through the list, the cursor skips dimmed settings.

Changing Mode Settings

Tip: ***To cancel a menu and return to the MODE dialog box without making a selection, press [ESC].***

From the MODE dialog box:

1. Highlight the mode setting you want to change. Use ↓ or ↑ (with [F1] and [F2]) to scroll through the list.
2. Press → or ← to display a menu that lists the valid settings. The current setting is highlighted.
3. Select the applicable setting. Either:
   - Use ↓ or ↑ to highlight the setting and press [ENTER].

     — or —
   - Press the number or letter for that setting.
4. Change other mode settings, if necessary.
5. When you finish all your changes, press [ENTER] to save the changes and exit the dialog box.

Important: If you press [ESC] instead of [ENTER] to exit the MODE dialog box, any mode changes you made will be canceled.

Overview of the Modes

Note: *For detailed information about a particular mode, look in the applicable section of this guidebook.*

Mode	Description
Graph	Type of graphs to plot: FUNCTION, PARAMETRIC, POLAR, SEQUENCE, or 3D.
Current Folder	Folder used to store and recall variables. Unless you have created additional folders, only the MAIN folder is available. Refer to “Using Folders to Store Independent Sets of Variables” in Chapter 10.
Display Digits	Maximum number of digits (FLOAT) or fixed number of decimal places (FIX) displayed in a floating-point result. Regardless of the setting, the total number of displayed digits in a floating-point result cannot exceed 12. Refer to page 27.
Angle	Units in which angle values are interpreted and displayed: RADIAN or DEGREE.
Exponential Format	Notation used to display results: NORMAL, SCIENTIFIC, or ENGINEERING. Refer to page 27.
Complex Format	Format used to display complex results, if any: REAL (complex results are not displayed unless you use a complex entry), RECTANGULAR, or POLAR.
Vector Format	Format used to display 2- and 3-element vectors: RECTANGULAR, CYLINDRICAL, or SPHERICAL.
Pretty Print	Turns the pretty print display feature OFF or ON. Refer to page 25.
Split Screen	Splits the screen into two parts and specifies how the parts are arranged: FULL (no split screen), TOP-BOTTOM, or LEFT-RIGHT. Refer to Chapter 5.
Split 1 App	Application in the top or left side of a split screen. If you are not using a split screen, this is the current application.
Split 2 App	Application in the bottom or right side of a split screen. This is active only for a split screen.
Number of Graphs	For a split screen, lets you set up both sides of the screen to display independent sets of graphs.
Graph 2	If Number of Graphs = 2, selects the type of graph in the Split 2 part of the screen. Refer to Chapter 15.
Split Screen Ratio	Proportional sizes of the two parts of a split screen: 1:1, 1:2, or 2:1.
Exact/Approx	Calculates expressions and displays results in numeric form or in rational/symbolic form: AUTO, EXACT, or APPROXIMATE. Refer to page 25.

Using the Catalog to Select a Command

The CATALOG is an alphabetic list of all commands (functions and instructions) on the TI-92. Although the commands are available on various menus, the CATALOG lets you access any command from one convenient list. It also gives help information that describes a command's parameters.

Selecting from the CATALOG

When you select a command, its name is inserted in the entry line at the cursor location. Therefore, you should position the cursor as necessary before selecting the command.

Note: The first time you display the CATALOG, it starts at the top of the list. The next time you display the CATALOG, it starts at the same place you left it.

1. Press [2nd] [CATALOG].

- Commands are listed in alphabetical order. Commands that do not start with a letter (+, %, √, Σ, etc.) are at the end of the list.
- To exit the CATALOG without selecting a command, press [ESC].

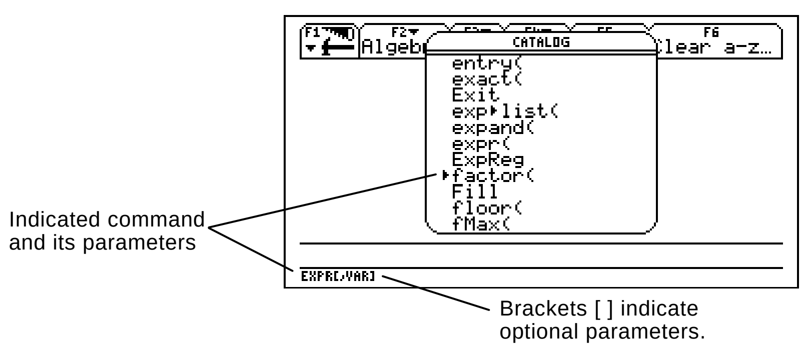
2. Move the ▸ indicator to the command, and press [ENTER].

Tip: From the top of the list, press ↑ to move to the bottom. From the bottom, press ↓ to move to the top.

To move the ▸ indicator:	Press or type:
One command at a time	↓ or ↑
One page at a time	[2nd] ↓ or [2nd] ↑
To the first command that begins with a specified letter	The letter. For example, type Z to go to the **Zoom** commands.

Help Information about Parameters

For the command indicated by ▸, the status line shows the required and optional parameters, if any, and their type.



Note: For details about the parameters, refer to that command's description in Appendix A.

From the example above, the syntax for **factor** is:

factor(*expression*)	required
— or —	
factor(*expression*,*variable*)	optional

Storing and Recalling Variable Values

When you store a value, you store it as a named variable. You can then use the name instead of the value in expressions. When the TI-92 encounters the name in an expression, it substitutes the variable's stored value.

Rules for Variable Names

A variable name:

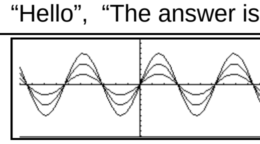
- Can use 1 to 8 characters consisting of letters and digits. This includes Greek letters (but not π), accented letters, and international letters. Do not include spaces.
  - The first character cannot be a digit.
- Can use uppercase or lowercase letters. The names AB22, Ab22, aB22, and ab22 all refer to the same variable.
- Cannot be the same as a name that is preassigned by the TI-92. Preassigned names include:
  - Built-in functions (such as **abs**) and instructions (such as **LineVert**). Refer to Appendix A.
  - System variables (such as xmin and xmax, which are used to store graph-related values). Refer to Appendix B for a list.

Examples

Variable	Description
myvar	OK.
a	OK.
Log	Not OK, name is preassigned to the **log** function.
Log1	OK.
3rdTotal	Not OK, starts with a digit.
circumfer	Not OK, more than 8 characters.

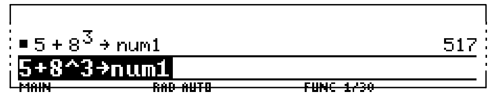
Data Types

You can save any TI-92 data type as a variable. For a list of data types, refer to **getType()** in Appendix A. Some examples are:

Data Types	Examples
Expressions	2.54, 1.25E6, 2π, xmin/10, 2+3i, $(x-2)^2$, $\sqrt{2}/2$
Lists	{2 4 6 8}, {1 1 2}
Matrices	[1 0 0], $\begin{bmatrix}1&0&0\\3&4&6\end{bmatrix}$
Character strings	"Hello", "The answer is:", "xmin/10"
Pictures	
Functions	myfunc(arg), ellipse(x,y,r1,r2)

Storing a Value in a Variable

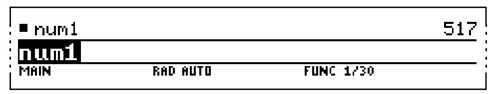
1. Enter the value you want to store, which can be an expression.
2. Press [STO▸]. The store symbol (→) is displayed.
3. Type the variable name.
4. Press [ENTER].



To store to a variable temporarily, you can use the “with” operator. Refer to “Substituting Values and Setting Constraints” in Chapter 6.

Displaying a Variable

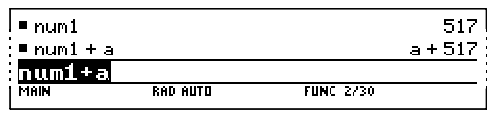
1. Type the variable name.
2. Press [ENTER].



If the variable is undefined, the variable name is shown in the result.

Note: Refer to Chapter 6 for information about symbolic manipulation.

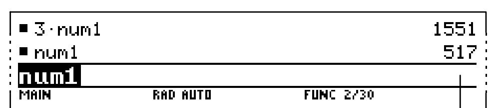
In this example, the variable a is undefined. Therefore, it is used as a symbolic variable.



Using a Variable in an Expression

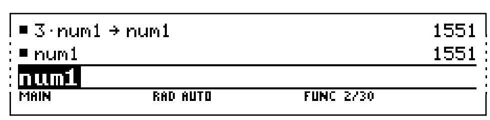
Tip: To view a list of existing variable names, use [2nd] [VAR-LINK] as described in Chapter 18.

1. Type the variable name into the expression.
2. Press [ENTER] to evaluate the expression.



The variable’s value did not change.

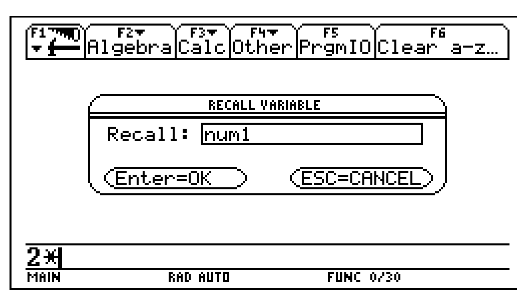
If you want the result to replace the variable’s previous value, you must store the result.



Recalling a Variable’s Value

In some cases, you may want to use a variable’s actual value in an expression instead of the variable name.

1. Press [2nd] [RCL] to display a dialog box.
2. Type the variable name.
3. Press [ENTER] twice.



In this example, the value stored in num1 will be inserted at the cursor position in the entry line.

Reusing a Previous Entry or the Last Answer

You can reuse a previous entry by reexecuting the entry "as is" or by editing the entry and then reexecuting it. You can also reuse the last calculated answer by inserting it into a new expression.

Reusing the Expression on the Entry Line

Tip: ***Reexecuting an entry "as is" is useful for iterative calculations that involve variables.***

When you press [ENTER] to evaluate an expression, the TI-92 leaves that expression on the entry line and highlights it. You can type over the entry, or you can reuse it as necessary.

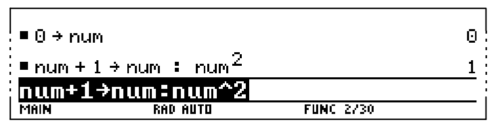
For example, using a variable, find the square of 1, 2, 3, etc.

1. Set the initial variable value.

   0 [STO▸] NUM [ENTER]

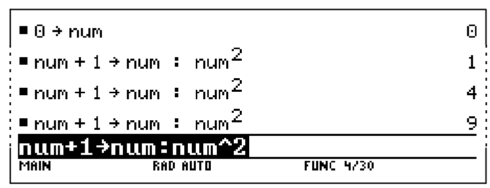
2. Enter the variable expression.

   NUM [+] 1 [STO▸] NUM [2nd] [:] NUM [^] 2 [ENTER]



3. Reenter to increment the variable and calculate the square.

   [ENTER]
   [ENTER]

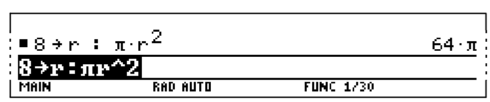


Tip: ***Editing an entry lets you make minor changes without retyping the entire entry.***

Using the equation A=πr^2, use trial and error to find the radius of a circle that covers 200 square centimeters.

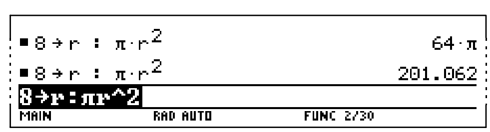
1. Use 8 as your first guess.

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



2. Display the answer in its approximate floating-point form.

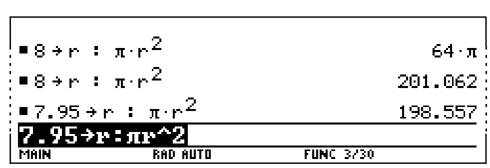
   [♦] [ENTER]



Note: ***When the entry contains a decimal point, the result is automatically displayed in floating-point.***

3. Edit and reexecute with 7.95.

   [◀] [♦] [⇐]
   7.95 [ENTER]



4. Continue until the answer is as accurate as you want.

Recalling a Previous Entry

You can recall any previous entry that is stored in the history area, even if the entry has scrolled off the top of the screen. The recalled entry *replaces* whatever is currently shown on the entry line. You can then reexecute or edit the recalled entry.

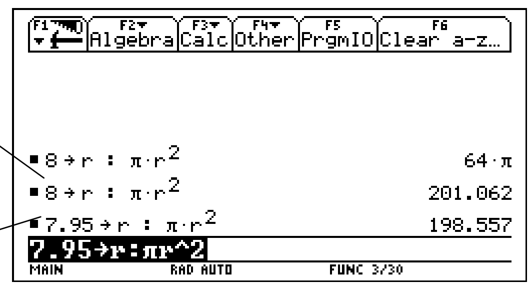
Note: ***Note:*** *You can also use the* ***entry*** *function to recall any previous entry. Refer to* ***entry()*** *in Appendix A.*

To recall:	Press:	Effect:
The last entry (if you've changed the entry line)	[2nd] [ENTRY] once	If the last entry is still shown on the entry line, this recalls the entry prior to that.
Previous entries	[2nd] [ENTRY] repeatedly	Each press recalls the entry prior to the one shown on the entry line.

For example:

If the entry line contains the last entry, [2nd] [ENTRY] recalls this entry.

If the entry line is edited or cleared, [2nd] [ENTRY] recalls this entry.



Recalling the Last Answer

Each time you evaluate an expression, the TI-92 stores the answer to the variable ans(1). To insert this variable in the entry line, press [2nd] [ANS].

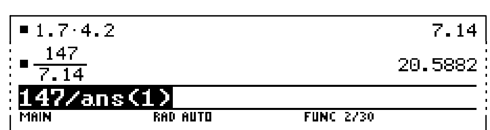
For example, calculate the area of a garden plot that is 1.7 meters by 4.2 meters. Then calculate the yield per square meter if the plot produces a total of 147 tomatoes.

1. Find the area.

   1.7 [×] 4.2 [ENTER]

2. Find the yield.

   147 [÷] [2nd] [ANS] [ENTER]



Variable ans(1) is inserted, and its value is used in the calculation.

Note: *Refer to* ***ans()*** *in Appendix A.*

Just as ans(1) always contains the last answer, ans(2), ans(3), etc., also contain previous answers. For example, ans(2) contains the next-to-last answer.

Auto-Pasting an Entry or Answer from the History Area

You can select any entry or answer from the history area and "auto-paste" a duplicate of it on the entry line. This lets you insert a previous entry or answer into a new expression without having to retype the previous information.

Why Use Auto-Paste

The effect of using auto-paste is similar to [2nd] [ENTRY] and [2nd] [ANS] as described in the previous section, but there are differences.

Note: You can also paste information by using the [F1] toolbar menu. Refer to "Cutting, Copying, and Pasting Information" in Chapter 10.

For entries:

Pasting lets you:	[2nd] [ENTRY] lets you:
Insert any previous entry into the entry line.	*Replace* the contents of the entry line with any previous entry.

For answers:

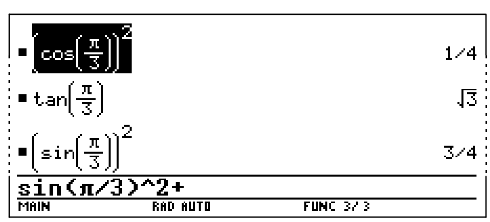
Pasting lets you:	[2nd] [ANS] lets you:
Insert the displayed value of *any previous answer* into the entry line.	Insert the variable ans(1), which contains *the last answer only*. Each time you enter a calculation, ans(1) is updated to the latest answer.

Auto-Pasting an Entry or Answer

Tip: To cancel auto-paste and return to the entry line, press [ESC].

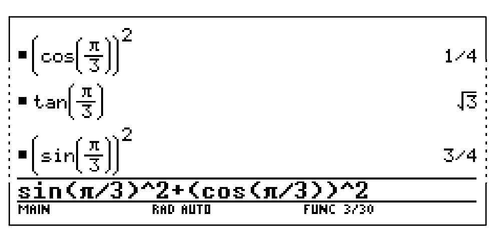
Tip: To view an entry or answer too long for one line (indicated by ▶ at the end of the line), use → and ← or [2nd] → and [2nd] ←.

1. On the entry line, place the cursor where you want to insert the entry or answer.
2. Press ↑ to move the cursor up into the history area. This highlights the last answer.
3. Use ↑ and ↓ to highlight the entry or answer to auto-paste.
   - ↑ moves from answer to entry up through the history area.
   - You can use ↑ to highlight items that have scrolled off the screen.



4. Press [ENTER].

   The highlighted item is inserted in the entry line.

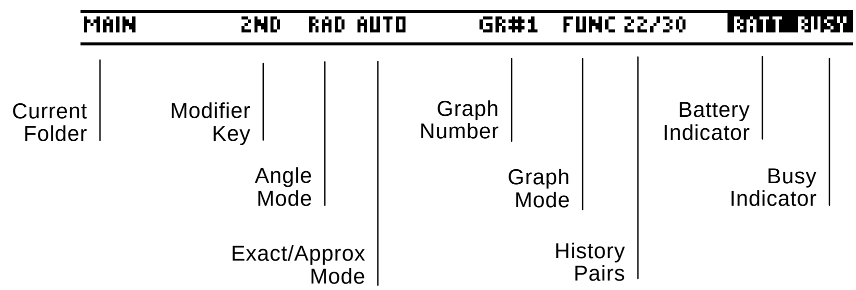


This pastes the entire entry or answer. If you need only a part of the entry or answer, edit the entry line to delete the unwanted parts.

Status Line Indicators in the Display

The status line is displayed at the bottom of all application screens. It shows information about the current state of the TI-92, including several important mode settings.

Status Line Indicators



Indicator	Meaning
Current Folder	Shows the name of the current folder. Refer to "Using Folders to Store Independent Sets of Variables" in Chapter 10. MAIN is the default folder that is set up automatically when you use the TI-92.
Modifier Key	Displayed when you press [↑], [◆], [2nd], or [✋].
⇑	The TI-92 will type an uppercase character for the next letter key you press.
◆	The TI-92 will access the diamond feature of the next key you press.
2ND	The TI-92 will use the second function of the next key you press.
∂	When used in combination with the cursor pad, the TI-92 will use any "dragging" features that are available in graphing and geometry.
Angle Mode	Shows the units in which angle values are interpreted and displayed. To change the Angle mode, use the [MODE] key.
RAD	Radians
DEG	Degrees
Exact/ Approx Mode	Shows how answers are calculated and displayed. Refer to page 25. To change the Exact/Approx mode, use the [MODE] key.
AUTO	Auto
EXACT	Exact
APPROX	Approximate

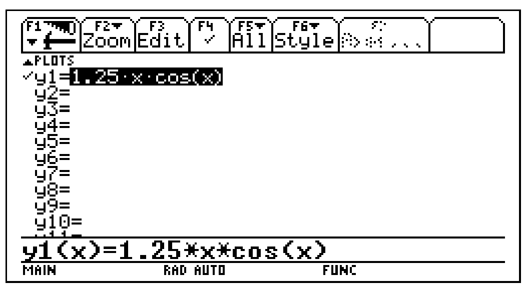
Status Line Indicators (Continued)

Indicator	Meaning
Graph Number	If the screen is split to show two independent graphs, this indicates which graph is active (GR#1 or GR#2).
Graph Mode	Indicates the type of graphs that can be plotted. (To change the Graph mode, use the [MODE] key.)
FUNC	y(x) functions
PAR	x(t) and y(t) parametric equations
POL	r(θ) polar equations
SEQ	u(n) sequences
3D	z(x,y) 3D equations
History Pairs	Displayed only on the Home screen to show information about the number of entry/answer pairs in the history area. Refer to page 20.
Battery Indicator	Displayed only when the batteries are getting low. If BATT is shown with a black background, change the batteries as soon as possible.
Busy Indicator	Displayed only when the TI-92 is performing a calculation or plotting a graph.
BUSY	A calculation or graph is in progress.
PAUSE	You have paused a graph or program.

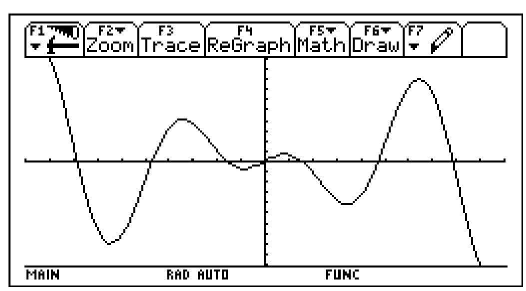
Basic Function Graphing

3

This chapter describes the steps used to display and explore a graph. Before using this chapter, you should be familiar with Chapter 2: Operating the TI-92.



Y= Editor shows an algebraic representation.

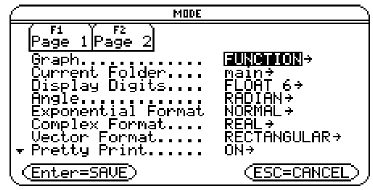
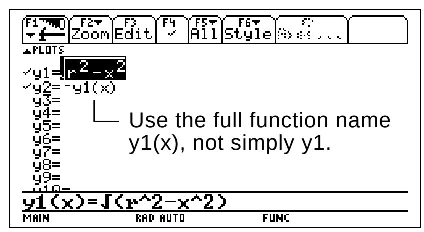
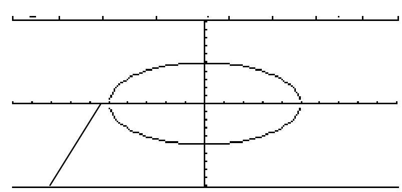
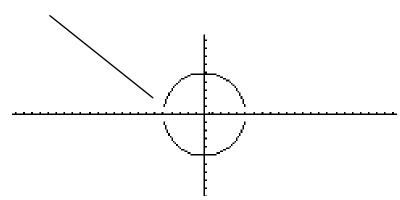


Graph screen shows a graphic representation.

Although this chapter describes how to graph y(x) functions, the basic steps apply to all graphing modes. Later chapters give specific information about the other graphing modes.

Preview of Basic Function Graphing

Graph a circle of radius 5, centered on the origin of the coordinate system. View the circle using the standard viewing window (**ZoomStd**). Then use **ZoomSqr** to adjust the viewing window.

Steps	Keystrokes	Display
1. Display the MODE dialog box. For Graph mode, select FUNCTION.	[MODE] ⊙ 1 [ENTER]	
2. Display the Home screen. Then store the radius, 5, in variable r.	[◆] [HOME] 5 [STO▸] R [ENTER]	5→r 5
3. Display and clear the Y= Editor. Then define $y1(x) = \sqrt{r^2 - x^2}$, the top half of a circle. *In function graphing, you must define separate functions for the top and bottom halves of a circle.*	[◆] [Y=] [F1] 8 [ENTER] [ENTER] [2nd] [√] R [^] 2 [−] X [^] 2 [)] [ENTER]	Use the full function name y1(x), not simply y1.
4. Define $y2(x) = ^{-}\sqrt{r^2 - x^2}$, the function for the bottom half of the circle. *The bottom half is the negative of the top half, so you can define y2(x) = ⁻y1(x).*	[ENTER] [(-)] Y 1 [(] X [)] [ENTER]	
5. Select the ZoomStd viewing window, which automatically graphs the functions. *In the standard viewing window, both the x and y axes range from ⁻10 to 10.* *However, this range is spread over a longer distance along the x axis than the y axis. Therefore, the circle appears as an ellipse.*	[F2] 6	Notice slight gap between top and bottom halves.
6. Select ZoomSqr. *ZoomSqr increases the range along the x axis so that circles and squares are shown in correct proportion.*	[F2] 5	

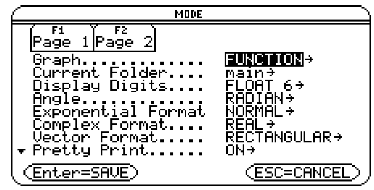
Note: There is a gap between the top and bottom halves of the circle because each half is a separate function. The mathematical endpoints of each half are (−5,0) and (5,0). Depending on the viewing window, however, the *plotted* endpoints for each half may be slightly different from their *mathematical* endpoints.

Overview of Steps in Graphing Functions

To graph one or more y(x) functions, use the general steps shown below. For a detailed description of each step, refer to the following pages. You may not need to do all the steps each time you graph a function.

Graphing Functions

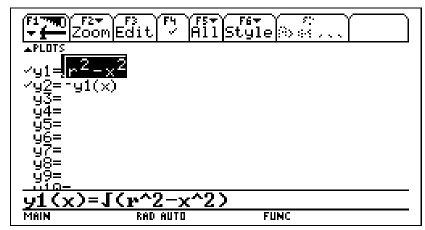
Set Graph mode ([MODE]) to FUNCTION. Also set Angle mode, if necessary.



Define functions on Y= Editor ([♦] [Y=]).

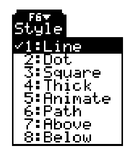
Tip: To turn off any stat data plots (Chapter 9), press [F5] 5 *or use* [F4] *to deselect them.*

Select ([F4]) which defined functions to graph.



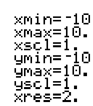
Tip: This is optional. For multiple functions, this helps visually distinguish one from another.

Set the display style ([F6]) for a function.

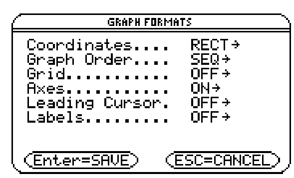


Tip: [F2] Zoom *also changes the viewing window.*

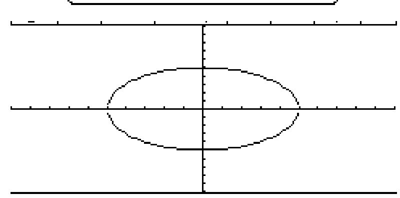
Define the viewing window ([♦] [WINDOW]).



Change the graph format ([♦] F or [F1] 9), if necessary.



Graph the selected functions ([♦] [GRAPH]).



Exploring the Graph

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a function.
- Use the [F2] Zoom toolbar menu to zoom in or out on a portion of the graph.
- Use the [F5] Math toolbar menu to find a zero, minimum, maximum, etc.

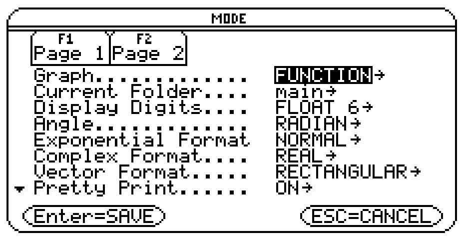
Setting the Graph Mode

Before graphing y(x) functions, you must select **FUNCTION** graphing. You may also need to set the **Angle** mode, which affects how the TI-92 graphs trigonometric functions.

Graph Mode

1. Press [MODE] to display the MODE dialog box, which shows the current mode settings.
2. Set the Graph mode to FUNCTION. Refer to “Setting Modes” in Chapter 2.

__Note:__ For graphs that do not use complex numbers, set Complex Format = REAL. Otherwise, it may affect graphs that use powers, such as $x^{1/3}$.



While this chapter specifically describes y(x) function graphs, the TI-92 lets you select from five Graph mode settings.

__Note:__ Other Graph mode settings are described in later chapters.

Graph Mode Setting	Description
FUNCTION	y(x) functions
PARAMETRIC	x(t) and y(t) parametric equations
POLAR	r(θ) polar equations
SEQUENCE	u(n) sequences
3D	z(x,y) 3D equations

Angle Mode

When using trigonometric functions, set the Angle mode for the units (RADIAN or DEGREE) in which you want to enter and display angle values.

Checking the Status Line

To see the current Graph mode and Angle mode, check the status line at the bottom of the screen.

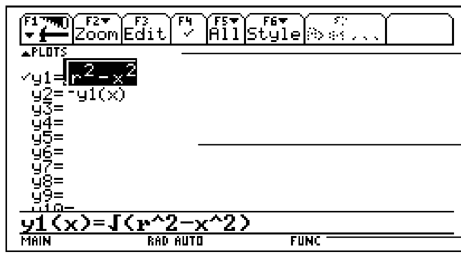
Defining Functions for Graphing

In FUNCTION graphing mode, you can graph functions named y1(x) through y99(x). To define and edit these functions, use the Y= Editor. (The Y= Editor lists function names for the current graphing mode. For example, in POLAR graphing mode, function names are r1(θ), r2(θ), etc.)

Defining a New Function

Note: *The function list shows abbreviated function names such as y1, but the entry line shows the full name y1(x).*

1. Press [♦] [Y=] or [APPS] 2 to display the Y= Editor.



Plots — You can scroll above y1= to see a list of stat plots. See Chapter 9.

Function List — You can scroll through the list of functions and definitions.

Entry Line — Where you define or edit the function highlighted in the list.

2. Press ⓓ and ⓤ to move the cursor to any undefined function. (Use [2nd] ⓓ and [2nd] ⓤ to scroll one page at a time.)

Tip: *For an undefined function, you do not need to press [ENTER] or [F3]. When you begin typing, the cursor moves to the entry line.*

3. Press [ENTER] or [F3] to move the cursor to the entry line.
4. Type the expression to define the function.
   - The independent variable in function graphing is x.
   - The expression can refer to other variables, including matrices, lists, and other functions.

Tip: *If you accidentally move the cursor to the entry line, press [ESC] to move it back to the function list.*

5. When you complete the expression, press [ENTER].

The function list now shows the new function, which is automatically selected for graphing.

Editing a Function

From the Y= Editor:

1. Press ⓓ and ⓤ to highlight the function.
2. Press [ENTER] or [F3] to move the cursor to the entry line.
3. Do any of the following.
   - Use ⓡ and ⓛ to move the cursor within the expression and edit it. Refer to “Editing an Expression in the Entry Line” in Chapter 2.
   — or —
   - Press [CLEAR] once or twice to clear the old expression, and then type the new one.

Tip: *To cancel any editing changes, press [ESC] instead of [ENTER].*

4. Press [ENTER].

The function list now shows the edited function, which is automatically selected for graphing.

Clearing a Function

From the Y= Editor:

To erase:	Do this:
A function from the function list	Highlight the function and press [←] or [CLEAR].
A function from the entry line	Press [CLEAR] once or twice (depending on the cursor's location) and then press [ENTER].
All functions	Press [F1] and then select 8:Clear Functions. When prompted for confirmation, press [ENTER].

Note: *[F1] 8 does not erase any stat plots (Chapter 9).*

You don't have to clear a function to prevent it from being graphed. As described on page 51, you can select the functions you want to graph.

From the Home Screen or a Program

Tip: *User-defined functions can have almost any name. However, if you want them to appear in the Y= Editor, use function names y1(x), y2(x), etc.*

You can also define and evaluate a function from the Home screen or a program.

- Use the **Define** and **Graph** commands. Refer to:
  - "Graphing a Function Defined on the Home Screen" and "Graphing a Piecewise Defined Function" in Chapter 15.
  - "Overview of Entering a Function" in Chapter 17.
- Store an expression directly to a function variable. Refer to:
  - "Storing and Recalling Variable Values" in Chapter 2.
  - "Creating and Evaluating User-Defined Functions" in Chapter 10.

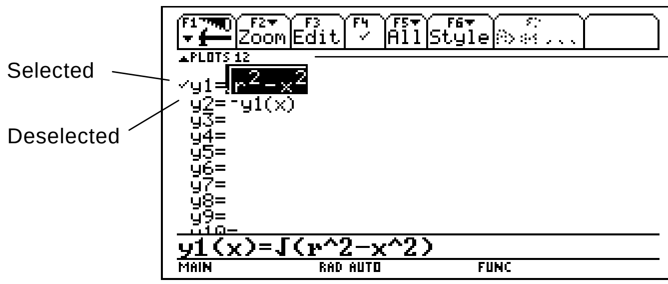
Selecting Functions to Graph

Regardless of how many functions are defined in the Y= Editor, you can select the ones you want to graph.

Selecting or Deselecting Functions

Press [♦] [Y=] or [APPS] 2 to display the Y= Editor.

A “✓” indicates which functions will be graphed the next time you display the Graph screen.



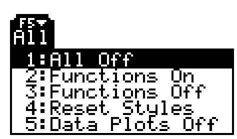
If PLOT numbers are displayed, those stat plots are selected.

In this example, Plots 1 and 2 are selected. To view them, scroll above y1=.

__Tip:__ You don’t have to select a function when you enter or edit it; it is selected automatically.

__Tip:__ To turn off any stat plots, press [F5] 5 or use [F4] to deselect them.

To select or deselect:	Do this:
A specified function	1. Move the cursor to highlight the function. 2. Press [F4]. This procedure selects a deselected function or deselects a selected function.
All functions	1. Press [F5] to display the **All** toolbar menu. 2. Select the applicable item.



From the Home Screen or a Program

You can also select or deselect functions from the Home screen or a program.

- Use the **FnOn** and **FnOff** commands (available from the Home screen’s [F4] Other toolbar menu) for functions. Refer to Appendix A.
- Use the **PlotsOn** and **PlotsOff** commands for stat plots. Refer to Appendix A.

Setting the Display Style for a Function

For each defined function, you can set a style that specifies how that function will be graphed. This is useful when graphing multiple functions. For example, set one as a solid line, another as a dotted line, etc.

Displaying or Changing a Function's Style

From the Y= Editor:

1. Move the cursor to highlight the applicable function.
2. Press [F6].

- Although the Line item is initially highlighted, the function's current style is indicated by a ✓ mark.
- To exit the menu without making a change, press [ESC].

3. To make a change, select the applicable style.

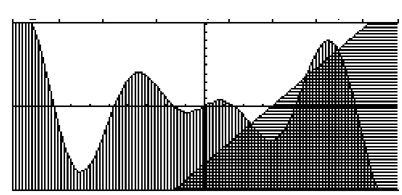
Tip: *To set* Line *as the style for all functions, press* [F5] *and select* 4:Reset Styles.

Style	Description
Line	Connects plotted points with a line. This is the default.
Dot	Displays a dot at each plotted point.
Square	Displays a solid box at each plotted point.
Thick	Connects plotted points with a thick line.
Animate	A round cursor moves along the leading edge of the graph but *does not* leave a path.
Path	A round cursor moves along the leading edge of the graph and *does* leave a path.
Above	Shades the area above the graph.
Below	Shades the area below the graph.

If You Use Above or Below Shading

The TI-92 has four shading patterns, used on a rotating basis. If you set one function as shaded, it uses the first pattern. The next shaded function uses the second pattern, etc. The fifth shaded function reuses the first pattern.

When shaded areas intersect, their patterns overlap.



From the Home Screen or a Program

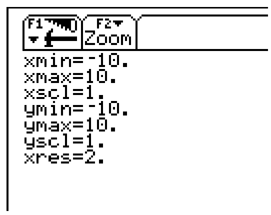
You can also set a function's style from the Home screen or a program. Refer to the **Style** command in Appendix A.

Defining the Viewing Window

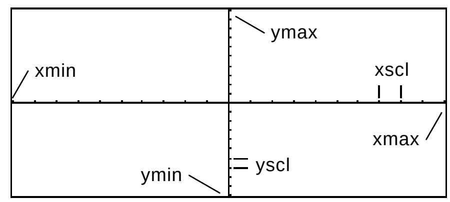
The viewing window represents the portion of the coordinate plane displayed on the Graph screen. By setting Window variables, you can define the viewing window's boundaries and other attributes. Function graphs, parametric graphs, etc., have their own independent set of Window variables.

Displaying Window Variables in the Window Editor

Press [♦] [WINDOW] or [APPS] 3 to display the Window Editor.



Window Variables
(shown in Window Editor)



Corresponding Viewing Window
(shown on Graph screen)

Tip: To turn off tick marks, set xscl=0 and/or yscl=0.

Tip: Small values of xres improve the graph's resolution but may reduce the graphing speed.

Variable	Description
xmin, xmax, ymin, ymax	Boundaries of the viewing window.
xscl, yscl	Distance between tick marks on the x and y axes.
xres	Sets pixel resolution (1 through 10) for function graphs. The default is 2. • At 1, functions are evaluated and graphed at each pixel along the x axis. • At 10, functions are evaluated and graphed at every 10th pixel along the x axis.

Changing the Values

From the Window Editor:

1. Move the cursor to highlight the value you want to change.
2. Do any of the following:
   - Type a value or an expression. The old value is erased when you begin typing.
     — or —
   - Press [CLEAR] to clear the old value; then type the new one.
     — or —
   - Press ⊲ or ⊳ to remove the highlighting; then edit the value.

Note: If you type an expression, it is evaluated when you move the cursor to a different Window variable or leave the Window Editor.

Values are stored as you type them; you do not need to press [ENTER]. [ENTER] simply moves the cursor to the next Window variable.

From the Home Screen or a Program

You can also store values directly to the Window variables from the Home screen or a program. Refer to "Storing and Recalling Variable Values" in Chapter 2.

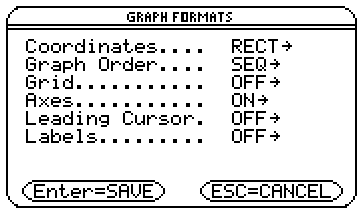
Changing the Graph Format

You can set the graph format to show or hide reference elements such as the axes, a grid, and the cursor's coordinates. Function graphs, parametric graphs, etc., have their own independent set of graph formats.

Displaying Graph Format Settings

Tip: You also can press [♦] F *from the Y= Editor, Window Editor, or Graph screen.*

From the Y= Editor, Window Editor, or Graph screen, press [F1] and select 9:Format.



- The GRAPH FORMATS dialog box shows the current settings.
- To exit without making a change, press [ESC].

Tip: To turn off tick marks, define the viewing window so that xscl and/or yscl = 0.

Format	Description
Coordinates	Shows cursor coordinates in rectangular (RECT) or polar (POLAR) form, or hides (OFF) the coordinates.
Graph Order	Graphs functions one at a time (SEQ) or all at the same time (SIMUL).
Grid	Shows (ON) or hides (OFF) grid points that correspond to the tick marks on the axes.
Axes	Shows (ON) or hides (OFF) the x and y axes.
Leading Cursor	Shows (ON) or hides (OFF) a reference cursor that tracks the functions as they are graphed.
Labels	Shows (ON) or hides (OFF) labels for the x and y axes.

Changing Settings

Tip: To cancel a menu or exit the dialog box without saving any changes, use [ESC] *instead of* [ENTER]*.*

From the GRAPH FORMATS dialog box:

1. Move the cursor to highlight the format setting.
2. Press ▸ to display a menu of valid settings for that format.
3. Select a setting. Either:
   - Move the cursor to highlight the setting, and then press [ENTER].
     — or —
   - Press the number for that setting.
4. After changing all applicable format settings, press [ENTER] to save your changes and close the GRAPH FORMATS dialog box.

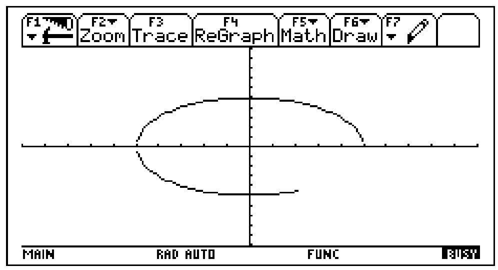
Graphing the Selected Functions

When you are ready to graph the selected functions, display the Graph screen. This screen uses the display style and viewing window that you previously defined.

Displaying the Graph Screen

Press ◆ [GRAPH] or APPS 4. The TI-92 automatically graphs the selected functions.

__Note__: If you select an F2 Zoom operation from the Y= Editor or Window Editor, the TI-92 automatically displays the Graph screen.



BUSY indicator shows while graphing is in progress.

Interrupting Graphing

While graphing is in progress:

- To pause graphing temporarily, press ENTER. (The PAUSE indicator replaces BUSY.) To resume, press ENTER again.
- To cancel graphing, press ON. To start graphing again from the beginning, press F4 (ReGraph).

If You Need to Change the Viewing Window

Depending on various settings, a function may be graphed such that it is too small, too large, or offset too far to one side of the screen. To correct this:

- Redefine the viewing window with different boundaries (page 53).
- Use a Zoom operation (page 59).

Smart Graph

When you display the Graph screen, the Smart Graph feature displays the previous window contents immediately, provided nothing has changed that requires regraphing.

Smart Graph updates the window and regraphs only if you have:

- Changed a mode setting that affects graphing, a function's graphing attribute, a Window variable, or a graph format.
- Selected or deselected a function or stat plot. (If you only select a new function, Smart Graph adds that function to the Graph screen.)
- Changed the definition of a selected function or the value of a variable in a selected function.
- Cleared a drawn object (Chapter 15).
- Changed a stat plot definition (Chapter 9).

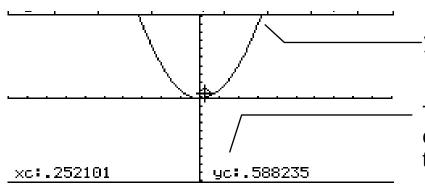
Displaying Coordinates with the Free-Moving Cursor

To display the coordinates of any location on the Graph screen, use the free-moving cursor. You can move the cursor to any pixel on the screen; the cursor is not confined to a graphed function.

Free-Moving Cursor

When you first display the Graph screen, no cursor is visible. To display the cursor, press the cursor pad. The cursor moves from the center of the screen, and its coordinates are displayed.

Tip: *If your screen does not show coordinates, set the graph format (♦ F) so that Coordinates = RECT or POLAR.*



$y1(x)=x^2$

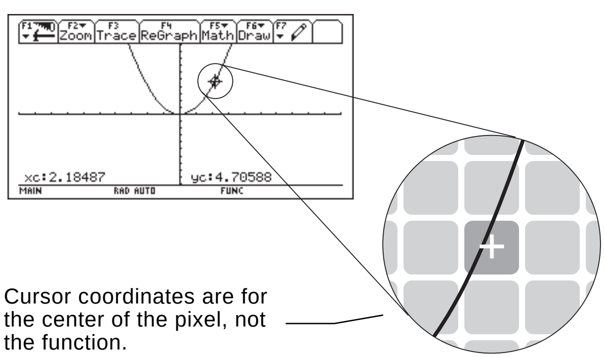
The "c" indicates these are cursor coordinates. The values are stored in the xc and yc system variables.

Rectangular coordinates use xc and yc. Polar coordinates use rc and θc.

Tip: *To hide the cursor and its coordinates temporarily, press* CLEAR*,* ESC*, or* ENTER*. The next time you move the cursor, it moves from its last position.*

To move the free-moving cursor:	Press:
To an adjoining pixel	The cursor pad for any direction.
In increments of 10 pixels	2nd and then the cursor pad.

When you move the cursor to a pixel that appears to be "on" the function, it may be near the function but not on it.



Cursor coordinates are for the center of the pixel, not the function.

To increase the accuracy:

- Use the **Trace** tool described on the next page to display coordinates that are on the function.
- Use a Zoom operation to zoom in on a portion of the graph.

Tracing a Function

To display the exact coordinates of any plotted point on a graphed function, use the [F3] **Trace** tool. Unlike the free-moving cursor, the trace cursor moves only along a function's plotted points.

Beginning a Trace

From the Graph screen, press [F3].

The trace cursor appears on the function, at the middle x value on the screen. The cursor's coordinates are displayed at the bottom of the screen.

***Note:** If any stat plots are graphed (Chapter 9), the trace cursor appears on the lowest-numbered stat plot.*

If multiple functions are graphed, the trace cursor appears on the lowest-numbered function selected in the Y= Editor. The function number is shown in the upper right part of the screen.

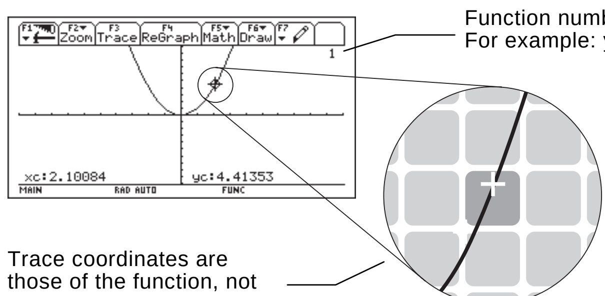
Moving along a Function

To move the trace cursor:	Do this:
To the previous or next plotted point	Press ◀ or ▶.
Approximately 5 plotted points (it may be more or less than 5, depending on the xres Window variable)	Press [2nd] ◀ or [2nd] ▶.
To a specified x value on the function	Type the x value and press [ENTER].

***Note:** If you enter an x value, it must be between xmin and xmax.*

The trace cursor moves only from plotted point to plotted point along the function, not from pixel to pixel.

***Tip:** If your screen does not show coordinates, set the graph format ([♦] F) so that Coordinates = RECT or POLAR.*



Each displayed y value is calculated from the x value; that is, y=y*n*(x). If the function is undefined at an x value, the y value is blank.

***Tip**: Use QuickCenter, described on the next page, to trace a function that goes above or below the window.*

You can continue to trace a function that goes above or below the viewing window. You cannot see the cursor as it moves in that "off the screen" area, but the displayed coordinate values show its correct coordinates.

Moving from Function to Function

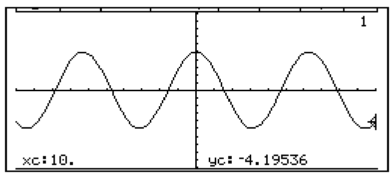
Press ↑ or ↓ to move to the previous or next selected function at the same x value. The new function number is shown on the screen.

The "previous or next" function is based on the order of the selected functions in the Y= Editor, not the appearance of the functions as graphed on the screen.

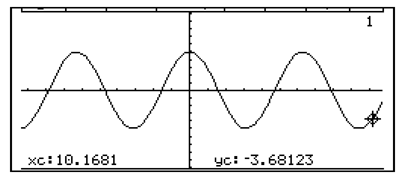
Automatic Panning

If you trace a function off the left or right edge of the screen, the viewing window automatically pans to the left or right. There is a slight pause while the new portion of the graph is drawn.

Note: *Automatic panning does not work if stat plots are displayed or if a function uses a shaded display style.*



Before automatic pan



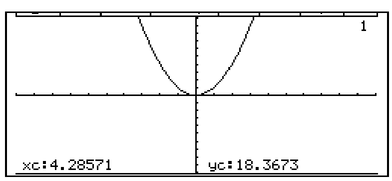
After automatic pan

After an automatic pan, the cursor continues tracing.

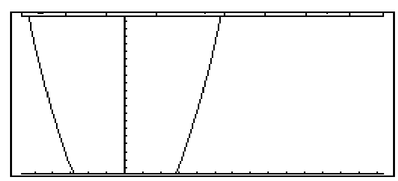
Using QuickCenter

If you trace a function off the top or bottom of the viewing window, you can press [ENTER] to center the viewing window on the cursor location.

Tip*: You can use QuickCenter at any time during a trace, even when the cursor is still on the screen.*



Before using QuickCenter


After using QuickCenter

After QuickCenter, the cursor stops tracing. If you want to continue tracing, press [F3].

Canceling Trace

To cancel a trace at any time, press [ESC].

A trace is also canceled when you display another application screen such as the Y= Editor. When you return to the Graph screen and press [F3] to begin tracing:

- If Smart Graph regraphed the screen, the cursor appears at the middle x value.
- If Smart Graph *does not* regraph the screen, the cursor appears at its previous location (before you displayed the other application).

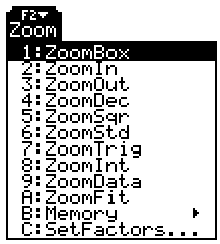
Using Zooms to Explore a Graph

The [F2] **Zoom** toolbar menu has several tools that let you adjust the viewing window. You can also save a viewing window for later use.

Overview of the Zoom Menu

Note: *If you select a Zoom tool from the Y=Editor or Window Editor, the TI-92 automatically displays the Graph screen.*

Press [F2] from the Y= Editor, Window Editor, or Graph screen.



Procedures for using ZoomBox, ZoomIn, ZoomOut, ZoomStd, Memory, and SetFactors are given later in this section.

For more information about the other items, refer to Appendix A.

Note: *Δx and Δy are the distances from the center of one pixel to the center of an adjoining pixel.*

Zoom Tool	Description
ZoomBox	Lets you draw a box and zoom in on that box.
ZoomIn, ZoomOut	Let you select a point and zoom in or out by an amount defined by SetFactors.
ZoomDec	Sets Δx and Δy to .1, and centers the origin.
ZoomSqr	Adjusts Window variables so that a square or circle is shown in correct proportion (instead of a rectangle or ellipse).
ZoomStd	Sets Window variables to their default values. xmin = -10 ymin = -10 xres = 2 xmax = 10 ymax = 10 xscl = 1 yscl = 1
ZoomTrig	Sets Window variables to preset values that are often appropriate for graphing trig functions. Centers the origin and sets: $\Delta x = \pi/24$ (.130899... radians or 7.5 degrees) ymin = -4 ymax = 4 xscl = $\pi/2$ (1.570796... radians or 90 degrees) yscl = 0.5
ZoomInt	Lets you select a new center point, and then sets Δx and Δy to 1 and sets xscl and yscl to 10.
ZoomData	Adjusts Window variables so that all selected stat plots are in view. Refer to Chapter 9.
ZoomFit	Adjusts the viewing window to display the full range of dependent variable values for the selected functions. In function graphing, this maintains the current xmin and xmax and adjusts ymin and ymax.
Memory	Lets you store and recall Window variable settings so that you can recreate a custom viewing window.
SetFactors	Lets you set Zoom factors for ZoomIn and ZoomOut.

Zooming In with a Zoom Box

Tip: *To move the cursor in larger increments, use* [2nd] ▶, [2nd] ▼, *etc.*

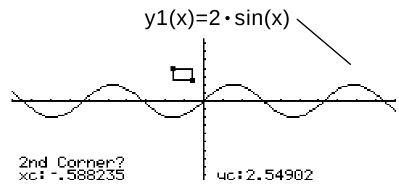
Tip: *You can cancel ZoomBox by pressing* [ESC] *before you press* [ENTER].

1. From the [F2] Zoom menu, select 1:ZoomBox.

   The screen prompts for 1st Corner?

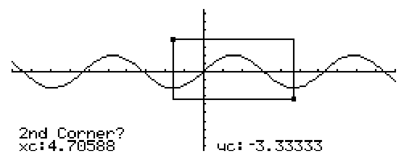
2. Move the cursor to any corner of the box you want to define, and then press [ENTER].

   The cursor changes to a small square, and the screen prompts for 2nd Corner?



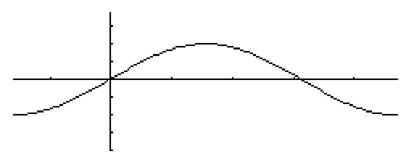
3. Move the cursor to the opposite corner of the zoom box.

   As you move the cursor, the box stretches.



4. When you have outlined the area you want to zoom in on, press [ENTER].

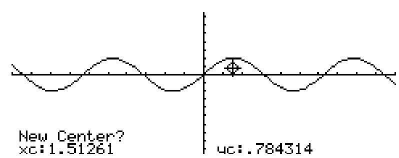
   The Graph screen shows the zoomed area.



Zooming In and Out on a Point

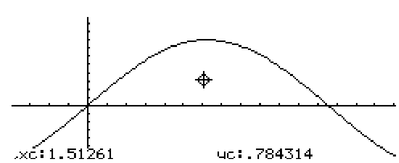
1. From the [F2] Zoom menu, select 2:ZoomIn or 3:ZoomOut.

   A cursor appears, and the screen prompts for New Center?



2. Move the cursor to the point where you want to zoom in or out, and then press [ENTER].

   The TI-92 adjusts the Window variables by the Zoom factors defined in SetFactors.



   - For a ZoomIn, the x variables are divided by xFact, and the y variables are divided by yFact.

     new xmin = $\frac{\text{xmin}}{\text{xFact}}$, etc.

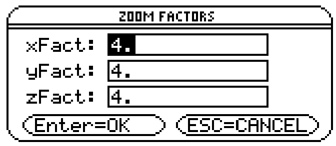
   - For a ZoomOut, the x variables are multiplied by xFact, and the y variables are multiplied by yFact.

     new xmin = xmin ∗ xFact , etc.

Changing Zoom Factors

The Zoom factors define the magnification and reduction used by ZoomIn and ZoomOut.

1. From the [F2] Zoom menu, select C:SetFactors to display the ZOOM FACTORS dialog box.



Zoom factors must be ≥ 1, but they do not have to be integers. The default setting is 4.

Tip: *To exit without saving any changes, press [ESC].*

2. Use ↓ and ↑ to highlight the value you want to change. Then:
   - Type the new value. The old value is cleared automatically when you begin typing.
     — or —
   - Press ← or → to remove the highlighting, and then edit the old value.
3. Press [ENTER] (after typing in an input box, you must press [ENTER] twice) to save any changes and exit the dialog box.

Saving or Recalling a Viewing Window

After using various Zoom tools, you may want to return to a previous viewing window or save the current one.

1. From the [F2] Zoom menu, select B:Memory to display its submenu.

2. Select the applicable item.

Select:	To:
1:ZoomPrev	Return to the viewing window displayed before the previous zoom.
2:ZoomSto	Save the current viewing window. (The current Window variable values are stored to the system variables zxmin, zxmax, etc.)
3:ZoomRcl	Recall the viewing window last stored with ZoomSto.

Note: *You can store only one set of Window variable values at a time. Storing a new set overwrites the old set.*

Restoring the Standard Viewing Window

You can restore the Window variables to their default values at any time.

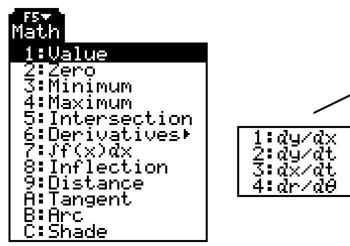
From the [F2] Zoom menu, select 6:ZoomStd.

Using Math Tools to Analyze Functions

On the Graph screen, the [F5] **Math** toolbar menu has several tools that help you analyze graphed functions.

Overview of the Math Menu

Press [F5] from the Graph screen.



On the Derivatives submenu, only dy/dx is available for function graphing. The other derivatives are available for other graphing modes (parametric, polar, etc.).

Note: *For Math results, cursor coordinates are stored in system variables xc and yc (rc and θc if you use polar coordinates). Derivatives, integrals, distances, etc., are stored in the system variable sysMath.*

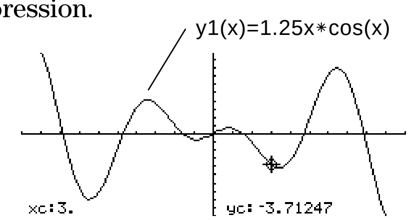
Math Tool	Description
Value	Evaluates a selected y(x) function at a specified x value.
Zero, Minimum, Maximum	Finds a zero (x-intercept), minimum, or maximum point within an interval.
Intersection	Finds the intersection of two functions.
Derivatives	Finds the derivative (slope) at a point.
∫f(x)dx	Finds the approximate numerical integral over an interval.
Inflection	Finds the inflection point of a curve, where its second derivative changes sign (where the curve changes concavity).
Distance	Draws and measures a line between two points on the same function or on two different functions.
Tangent	Draws a tangent line at a point and displays its equation.
Arc	Finds the arc length between two points along a curve.
Shade	Depends on the number of functions graphed. • If only one function is graphed, this shades the function's area above or below the x axis. • If two or more functions are graphed, this shades the area between any two functions within an interval.

Finding y(x) at a Specified Point

__Tip:__ You can also display function coordinates by tracing the function ([F3]), typing an x value, and pressing [ENTER].

1. From the Graph screen, press [F5] and select 1:Value.
2. Type the x value, which must be a real value between xmin and xmax. The value can be an expression.
3. Press [ENTER].

   The cursor moves to that x value on the first function selected in the Y= Editor, and its coordinates are displayed.



4. Press ↓ or ↑ to move the cursor between functions at the entered x value. The corresponding y value is displayed.

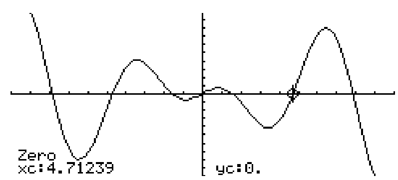
   Note: If you press ← or →, the free-moving cursor appears. You may not be able to move it back to the entered x value.

Finding a Zero, Minimum, or Maximum within an Interval

__Tip:__ Typing x values is a quick way to set bounds.

1. From the Graph screen, press [F5] and select 2:Zero, 3:Minimum, or 4:Maximum.
2. As necessary, use ↓ and ↑ to select the applicable function.
3. Set the lower bound for x. Either use ← and → to move the cursor to the lower bound or type its x value.
4. Press [ENTER]. A ▶ at the top of the screen marks the lower bound.
5. Set the upper bound, and press [ENTER].

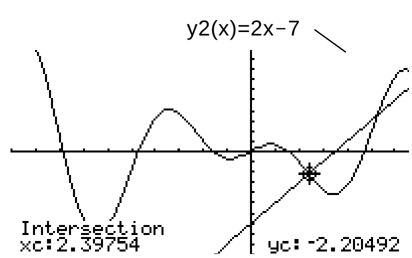
   The cursor moves to the solution, and its coordinates are displayed.



Finding the Intersection of Two Functions within an Interval

1. From the Graph screen, press [F5] and select 5:Intersection.
2. Select the first function, using ↓ or ↑ as necessary, and press [ENTER]. The cursor moves to the next graphed function.
3. Select the second function, and press [ENTER].
4. Set the lower bound for x. Either use ← and → to move the cursor to the lower bound or type its x value.
5. Press [ENTER]. A ▶ at the top of the screen marks the lower bound.
6. Set the upper bound, and press [ENTER].

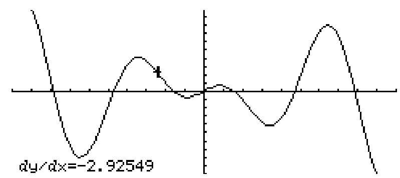
   The cursor moves to the intersection, and its coordinates are displayed.



Finding the Derivative (Slope) at a Point

1. From the Graph screen, press [F5] and select 6:Derivatives. Then select 1:dy/dx from the submenu.
2. As necessary, use ⊙ and ⊙ to select the applicable function.
3. Set the derivative point. Either move the cursor to the point or type its x value.
4. Press [ENTER].

   The derivative at that point is displayed.



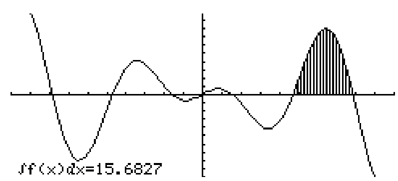
Finding the Numerical Integral over an Interval

Tip: *Typing x values is a quick way to set the limits.*

Tip: *To erase the shaded area, press [F4] (ReGraph).*

1. From the Graph screen, press [F5] and select 7:∫f(x)dx.
2. As necessary, use ⊙ and ⊙ to select the applicable function.
3. Set the lower limit for x. Either use ⊙ and ⊙ to move the cursor to the lower limit or type its x value.
4. Press [ENTER]. A ▶ at the top of the screen marks the lower limit.
5. Set the upper limit, and press [ENTER].

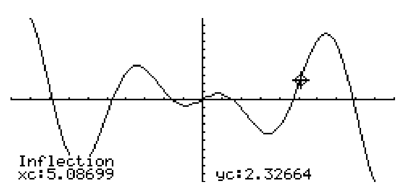
   The interval is shaded, and its approximate numerical integral is displayed.



Finding an Inflection Point within an Interval

1. From the Graph screen, press [F5] and select 8:Inflection.
2. As necessary, use ⊙ and ⊙ to select the applicable function.
3. Set the lower bound for x. Either use ⊙ and ⊙ to move the cursor to the lower bound or type its x value.
4. Press [ENTER]. A ▶ at the top of the screen marks the lower bound.
5. Set the upper bound, and press [ENTER].

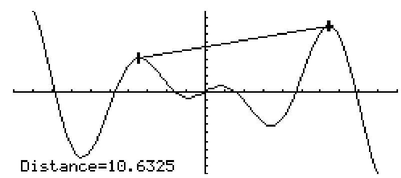
   The cursor moves to the inflection point (if any) within the interval, and its coordinates are displayed.



Finding the Distance between Two Points

1. From the Graph screen, press [F5] and select 9:Distance.
2. As necessary, use ⓓ and ⓤ to select the function for the first point.
3. Set the first point. Either use ⓛ or ⓡ to move the cursor to the point or type its x value.
4. Press [ENTER]. A + marks the point.
5. If the second point is on a different function, use ⓓ and ⓤ to select the function.
6. Set the second point. (If you use the cursor to set the point, a line is drawn as you move the cursor.)
7. Press [ENTER].

   The distance between the two points is displayed, along with the connecting line.

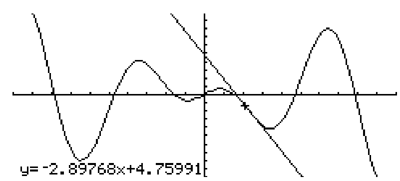


Drawing a Tangent Line

Tip: To erase a drawn tangent line, press [F4] (ReGraph).

1. From the Graph screen, press [F5] and select A:Tangent.
2. As necessary, use ⓓ and ⓤ to select the applicable function.
3. Set the tangent point. Either move the cursor to the point or type its x value.
4. Press [ENTER].

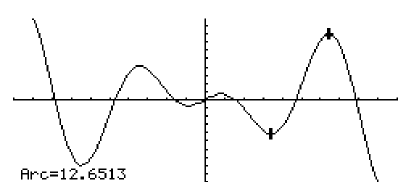
   The tangent line is drawn, and its equation is displayed.



Finding an Arc Length

1. From the Graph screen, press [F5] and select B:Arc.
2. As necessary, use ⓓ and ⓤ to select the applicable function.
3. Set the first point of the arc. Either use ⓛ or ⓡ to move the cursor or type the x value.
4. Press [ENTER]. A + marks the first point.
5. Set the second point, and press [ENTER].

   A + marks the second point, and the arc length is displayed.

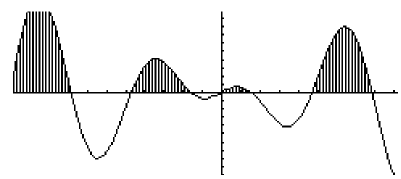


Shading the Area between a Function and the X Axis

You must have only one function graphed. If you graph two or more functions, the Shade tool shades the area between two functions.

1. From the Graph screen, press [F5] and select C:Shade. The screen prompts for Above X axis?
2. Select one of the following. To shade the function's area:
   - Above the x axis, press [ENTER].
   - Below the x axis, press N.
3. Set the lower bound for x. Either use ◀ and ▶ to move the cursor to the lower bound or type its x value.
4. Press [ENTER]. A ▶ at the top of the screen marks the lower bound.
5. Set the upper bound, and press [ENTER].

   The bounded area is shaded.



Note: *If you do not press ◀ or ▶, or type an x value when setting the lower and upper bound, xmin and xmax will be used as the lower and upper bound, respectively.*

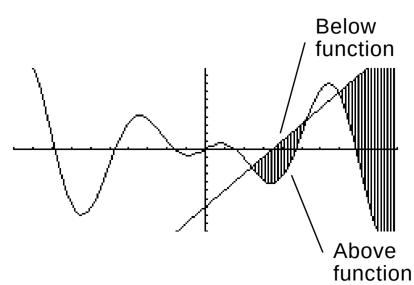
Tip: *To erase the shaded area, press [F4] (ReGraph).*

Shading the Area between Two Functions within an Interval

You must have at least two functions graphed. If you graph only one function, the Shade tool shades the area between the function and the x axis.

1. From the Graph screen, press [F5] and select C:Shade. The screen prompts for Above?
2. As necessary, use ▼ or ▲ to select a function. (Shading will be *above* this function.)
3. Press [ENTER]. The cursor moves to the next graphed function, and the screen prompts for Below?
4. As necessary, use ▼ or ▲ to select another function. (Shading will be *below* this function.)
5. Press [ENTER].
6. Set the lower bound for x. Either use ◀ and ▶ to move the cursor to the lower bound or type its x value.
7. Press [ENTER]. A ▶ at the top of the screen marks the lower bound.
8. Set the upper bound, and press [ENTER].

   The bounded area is shaded.



Note: *If you do not press ◀ or ▶, or type an x value when setting the lower and upper bound, xmin and xmax will be used as the lower and upper bound, respectively.*

Tip: *To erase the shaded area, press [F4] (ReGraph).*

Tables

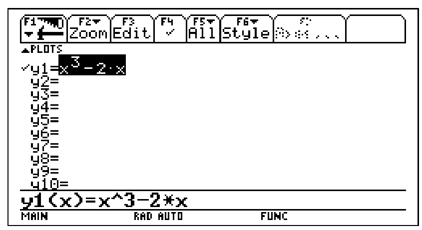
4

Note: ***Tables are not available in 3D Graph mode.***

Previously, in Chapter 3: Basic Function Graphing, you learned how to define and graph a function.

By using a table, you can display a defined function in a tabular form.

Y= Editor shows an algebraic representation.



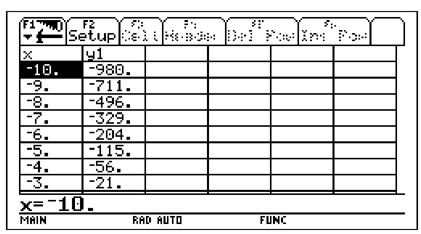
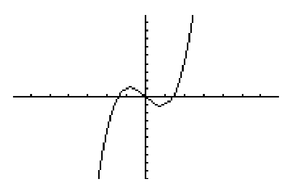


Table screen shows a numeric representation.



Graph screen shows a graphic representation.

The table lists a series of values for the independent variable and shows the corresponding value of the dependent variable.

$y(x) = x^3 - 2x$

Independent variable: x

Dependent variable: $y(x)$

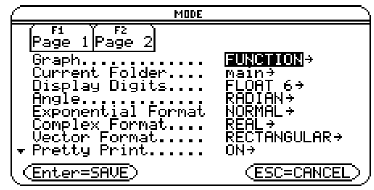
Preview of Tables

Evaluate the function $y=x^3-2x$ at each integer between -10 and 10. How many sign changes are there, and where do they occur?

Steps	Keystrokes	Display

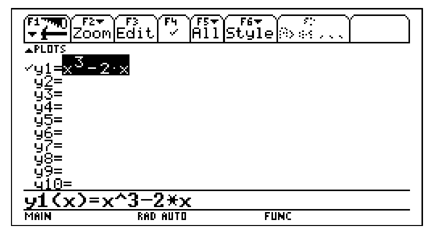
1. Display the MODE dialog box. For the Graph mode, select FUNCTION.

Keystrokes: [MODE] → 1 [ENTER]



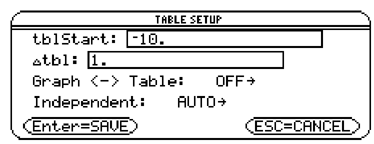
2. Display and clear the Y= Editor. Then define $y1(x) = x^3 - 2x$.

Keystrokes: [♦] [Y=] / [F1] 8 [ENTER] / [ENTER] / X [^] 3 [−] 2 X / [ENTER]



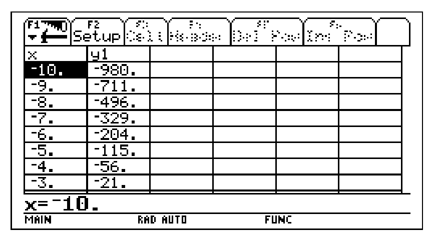
3. Set the table parameters to:
   tblStart = -10
   Δtbl = 1
   Graph < - > Table = OFF
   Independent = AUTO

Keystrokes: [♦] [TblSet] / [(-)] 1 0 / ↓ 1 / ↓ → 1 / ↓ → 1 [ENTER]



4. Display the Table screen.

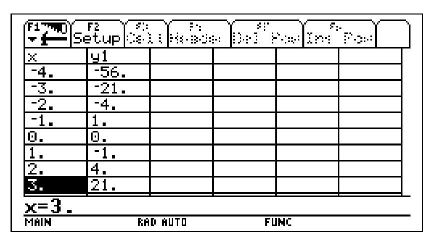
Keystrokes: [♦] [TABLE]



5. Scroll through the table. Notice that y1 changes sign at x = -1, 1, and 2.

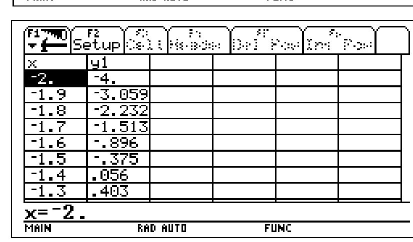
   To scroll one page at a time, use [2nd] ↓ and [2nd] ↑.

Keystrokes: ↓ and ↑ as necessary



6. Zoom in on the sign change between x = -2 and x = -1 by changing the table parameters to:
   tblStart = -2
   Δtbl = .1

Keystrokes: [F2] / [(-)] 2 / ↓ . 1 / [ENTER] [ENTER]



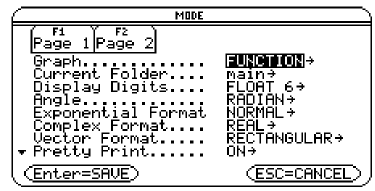
Overview of Steps in Generating a Table

To generate a table of values for one or more functions, use the general steps shown below. For specific information about setting table parameters and displaying the table, refer to the following pages.

Generating a Table

Note: *Tables are not available in 3D Graph mode.*

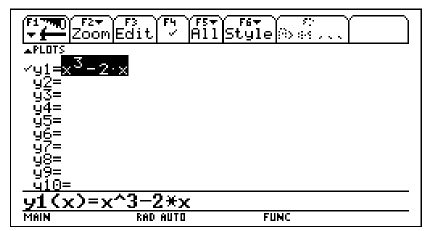
Set Graph mode and, if necessary, Angle mode ([MODE]).



Tip: *For information on defining and selecting functions with the Y= Editor, refer to Chapter 3.*

Define functions on Y= Editor ([◆] [Y=]).

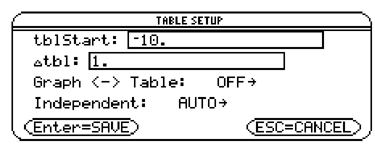
Select ([F4]) which defined functions to display in the table.



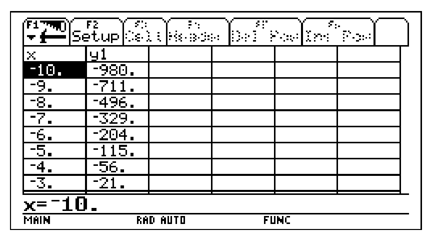
Tip: *You can specify:*

- *An automatic table*
  - *Based on initial values.*
  - *That matches a graph.*
- *A manual (ask) table.*

Set up the initial table parameters ([◆] [TblSet]).



Display the table ([◆] [TABLE]).



Exploring the Table

From the Table screen, you can:

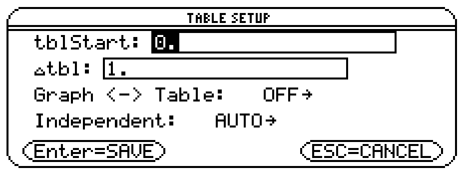
- Scroll through the table to see values on other pages.
- Highlight a cell to see its full value.
- Change the table's setup parameters. By changing the starting or incremental value used for the independent variable, you can zoom in or out on the table to see different levels of detail.
- Change the cell width.
- Edit selected functions.
- Build or edit a manual table to show only specified values of the independent variable.

Setting Up the Table Parameters

To set up the initial parameters for a table, use the TABLE SETUP dialog box. After the table is displayed, you can also use this dialog box to change the parameters.

Displaying the TABLE SETUP Dialog Box

To display the TABLE SETUP dialog box, press [♦] [TblSet] . From the Table screen, you can also press [F2].



***Note:** The table initially starts at tblStart, but you can use ↑ to scroll to prior values.*

Setup Parameter	Description
tblStart	If Independent = AUTO and Graph < - > Table = OFF, this specifies the starting value for the independent variable.
∆tbl	If Independent = AUTO and Graph < - > Table = OFF, this specifies the incremental value for the independent variable. ∆tbl can be positive or negative, but not zero.
Graph < - > Table	If Independent = AUTO: OFF — The table is based on the values you enter for tblStart and ∆tbl. ON — The table is based on the same independent variable values that are used to graph the functions on the Graph screen. These values depend on the Window variables set in the Window Editor (Chapter 3) and the split screen size (Chapter 5).
Independent	AUTO —The TI-92 automatically generates a series of values for the independent variable based on tblStart, ∆tbl, and Graph < - > Table. ASK — Lets you build a table manually by entering specific values for the independent variable.

Which Setup Parameters to Use

To generate:	tblStart	Δtbl	Graph < - > Table	Independent
An automatic table				
• Based on initial values	value	value	OFF	AUTO
• That matches Graph screen	—	—	ON	AUTO
A manual table	—	—	—	ASK

"—" means that any value entered for this parameter is ignored for the indicated type of table.

In SEQUENCE graphing mode (Chapter 13), use integers for tblStart and Δtbl.

Changing the Setup Parameters

From the TABLE SETUP dialog box:

1. Use ⊙ (down) and ⊙ (up) to highlight the value or setting to change.
2. Specify the new value or setting.

To change:	Do this:
tblStart or Δtbl	Type the new value. The existing value is erased when you start to type. — or — Press ⊙ (left) or ⊙ (right) to remove the highlighting. Then edit the existing value.
Graph < - > Table or Independent	Press ⊙ (left) or ⊙ (right) to display a menu of valid settings. Then either: • Move the cursor to highlight the setting and press [ENTER]. — or — • Press the number for that setting.

3. After changing all applicable values or settings, press [ENTER] to save your changes and close the dialog box.

***Tip:** To cancel a menu or exit the dialog box without saving any changes, press [ESC] instead of [ENTER].*

From the Home Screen or a Program

You can set up a table's parameters from the Home screen or a program. You can:

- Store values directly to the system variables tblStart and Δtbl. Refer to "Storing and Recalling Variable Values" in Chapter 2.
- Set Graph < - > Table and Independent by using the **setTable** function. Refer to Appendix A.

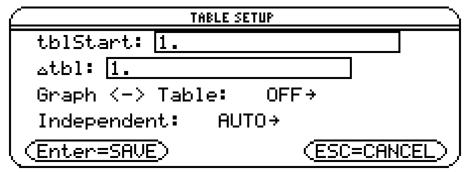
Displaying an Automatic Table

If **Independent = AUTO** on the TABLE SETUP dialog box, a table is generated automatically when you display the Table screen. If **Graph <-> Table = ON**, the table matches the trace values from the Graph screen. If **Graph <-> Table = OFF**, the table is based on the values you entered for **tblStart** and **∆tbl**.

Before You Begin

Define and select the applicable functions on the Y= Editor ([◆] [Y=]). This example uses y1(x) = x^3 – x/3.

Then enter the initial table parameters ([◆] [TblSet]).



Displaying the Table Screen

To display the Table screen, press [◆] [TABLE] or [APPS] 5.

The cursor initially highlights the cell that contains the starting value of the independent variable. You can move the cursor to any cell that contains a value.

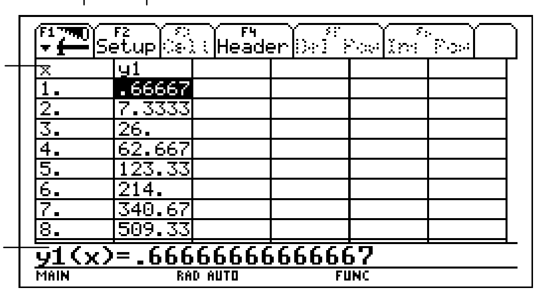
__Tip:__ You can scroll back from the starting value by pressing ↑ or [2nd] ↑.

First column shows values of the independent variable.

Other columns show corresponding values of the functions selected in the Y= Editor.

Header row shows names of independent variable (x) and selected functions (y1).

Entry line shows full value of highlighted cell.



To move the cursor:	Press:
One cell at a time	↓, ↑, →, or ←
One page at a time	[2nd] and then ↓, ↑, →, or ←

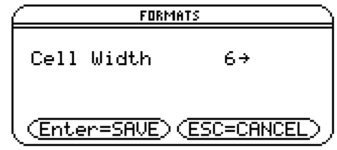
The header row and the first column are fixed so that they cannot scroll off the screen.

- When you scroll down or up, the variable and function names are always visible across the top of the screen.
- When you scroll right or left, the values of the independent variable are always visible along the left side of the screen.

Changing the Cell Width

Cell width determines the maximum number of digits and symbols (decimal point, minus sign, and "E" for scientific notation) that can be displayed in a cell. All cells in the table have the same width.

To change the cell width from the Table screen:



Note: *By default, the cell width is 6.*

1. Press [♦] F or [F1] 9.
2. Press ⊙ or ⊙ to display a menu of valid widths (3 – 12).
3. Move the cursor to highlight a number and press [ENTER]. (For single-digit numbers, you can type the number and press [ENTER].)
4. Press [ENTER] to close the dialog box and update the table.

How Numbers Are Displayed in a Cell

Note: *If a function is undefined at a particular value,* undef *is displayed in the cell.*

Tip: *Use* [MODE] *to set the display modes.*

Whenever possible, a number is shown according to the currently selected display modes (Display Digits, Exponential Format, etc.). The number may be rounded as necessary. However:

- If a number's magnitude is too large for the current cell width, the number is rounded and shown in scientific notation.
- If the cell width is too narrow even for scientific notation, "..." is shown.

By default, Display Digits = FLOAT 6. With this mode setting, a number is shown with up to six digits, even if the cell is wide enough to show more. Other settings similarly affect a displayed number.

Tip: *To see a number in full precision, highlight the cell and look at the entry line.*

	If cell width is:			
Full Precision	**3**	**6**	**9**	**12**
1.2345678901	1.2	1.2346	1.23457	1.23457
⁻123456.78	...	⁻1.2E5	⁻123457.	⁻123457.
.000005	...	5.E⁻6	.000005	.000005
1.2345678E19	...	1.2E19	1.2346E19	1.23457E19
⁻1.23456789012E⁻200	...	...	⁻1.2E⁻200	⁻1.2346E⁻200

Note: Depending on display mode settings, some values are not shown in full precision even when the cell is wide enough.

If You Are Using Complex Numbers

A cell shows as much as possible of a complex number (according to the current display modes) and then shows "..." at the end of the displayed portion.

When you highlight a cell containing a complex number, the entry line shows the real and imaginary parts with a maximum of four digits each (FLOAT 4).

Editing a Selected Function

From a table, you can change a selected function without having to use the Y= Editor.

1. Move the cursor to any cell in the column for that function. The table's header row shows the function names (y1, etc.).
2. Press [F4] to move the cursor to the entry line, where the function is displayed and highlighted.
3. Make any changes, as necessary.
   - Type the new function. The old function is erased when you begin typing.
     — or —
   - Press [CLEAR] to clear the old function. Then type the new one.
     — or —
   - Press ← or → to remove the highlighting. Then edit the function.
4. Press [ENTER] to save the edited function and update the table. The edited function is also saved in the Y= Editor.

Tip: *You can use this feature to view a function without leaving the table.*

Tip: *To cancel any changes and return the cursor to the table, press [ESC] instead of [ENTER].*

If You Want to Change the Setup Parameters

After generating an automatic table, you can change its setup parameters as necessary.

Press [F2] or [♦] [TblSet] to display the TABLE SETUP dialog box. Then make your changes as described on pages 70 and 71.

Building a Manual (Ask) Table

If **Independent = ASK** on the TABLE SETUP dialog box, the TI-92 lets you build a table manually by entering specific values for the independent variable.

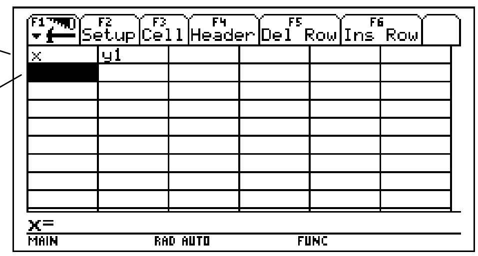
Displaying the Table Screen

To display the Table screen, press [♦] [TABLE] or [APPS] 5.

If you set Independent = ASK (with [♦] [TblSet]) before displaying a table for the first time, a blank table is displayed. The cursor highlights the first cell in the independent variable column.

Header row shows names of independent variable (x) and selected functions (y1).

Enter a value here.



If you first display an automatic table and then change it to Independent = ASK, the table continues to show the same values. However, you can no longer see additional values by scrolling up or down off the screen.

Entering or Editing an Independent Variable Value

You can enter a value in column 1 (independent variable) only.

1. Move the cursor to highlight the cell you want to enter or edit.
   - If you start with a blank table, you can enter a value in consecutive cells only (row 1, row 2, etc.). You cannot skip cells (row1, row3).
   - If a cell in column 1 contains a value, you can edit that value.
2. Press [F3] to move the cursor to the entry line.
3. Type a new value or expression, or edit the existing value.
4. Press [ENTER] to move the value to the table and update the corresponding function values.

***Tip:** To enter a new value in a cell, you do not need to press [F3]. Simply begin typing.*

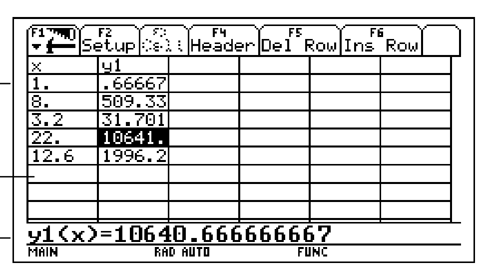
The cursor returns to the entered cell. You can use ⓓ to move to the next row.

Note: *In this example, you can move the cursor to column 2, but you can enter values in column 1 only.*

Enter values in any numerical order.

Enter a new value here.

Shows full value of highlighted cell.



Entering a List in the Independent Variable Column

Note: If the independent variable column contains existing values, they are shown as a list (which you can edit).

1. Move the cursor to highlight any cell in the independent variable column.
2. Press [F4] to move the cursor to the entry line.
3. Type a series of values, enclosed in braces { } and separated by commas. For example:

```
x={1,1.5,1.75,2}
```

   You can also enter a list variable or an expression that evaluates to a list.
4. Press [ENTER] to move the values into the independent variable column. The table is updated to show the corresponding function values.

Adding, Deleting, or Clearing

To:	Do this:
Insert a new row above a specified row	Highlight a cell in the specified row and press [F6]. The new row is undefined (undef) until you enter a value for the independent variable.
Delete a row	Highlight a cell in the row and press [F5]. If you highlight a cell in the independent variable column, you can also press [←].
Clear the entire table (but *not* the selected Y= functions)	Press [F1] 8. When prompted for confirmation, press [ENTER].

Cell Width and Display Formats

Several factors affect how numbers are displayed in a table. Refer to "Changing the Cell Width" and "How Numbers Are Displayed in a Cell" on page 73.

From the Home Screen or a Program

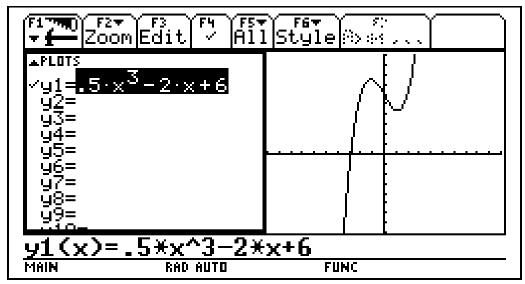
System variable tblInput contains a list of all independent variable values entered in the table, even those not currently displayed. tblInput is also used for an automatic table, but it contains only the independent variable values that are currently displayed.

Before displaying a table, you can store a list of values directly to the tblInput system variable.

Split Screens

5

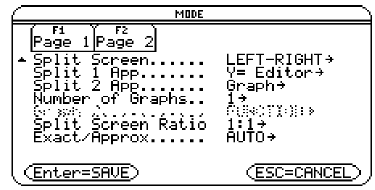
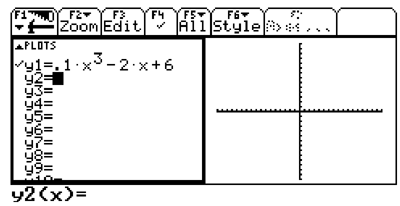
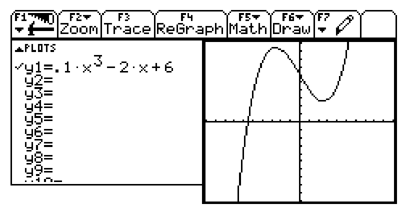
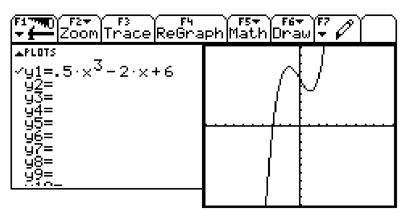
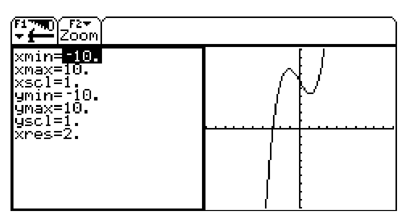
On the TI-92, you can split the screen to show two applications at the same time.



For example, it may be helpful to show both the Y= Editor and the Graph screen so that you can see the list of functions and how they are graphed.

Preview of Split Screens

Split the screen to show the Y= Editor and the Graph screen. Then explore the behavior of a polynomial as its coefficients change.

Steps	Keystrokes	Display
1. Display the MODE dialog box. For Graph, select FUNCTION. For Split Screen, select LEFT-RIGHT. For Split 1 App, select Y= Editor. For Split 2 App, select Graph.	[MODE] → 1 [F2] → 3 ↓ → 2 ↓ → 4 [ENTER]	
2. Clear the Y= Editor and turn off any stat data plots. Then define y1(x) = $.1x^3$–2x+6. *A thick border around the Y= Editor indicates it is active. When active, its entry line goes all the way across the display.*	[F1] 8 [ENTER] [F5] 5 [ENTER] . 1 X [^] 3 [-] 2 X [+] 6 [ENTER]	
3. Select the ZoomStd viewing window, which switches to the Graph screen and graphs the function. *The thick border is now around the Graph screen.*	[F2] 6	
4. Switch to the Y= Editor. Then edit y1(x) to change $.1x^3$ to $.5x^3$. *[2nd] [⊞] is the second function of [APPS].* *The thick border is around the Y= Editor.*	[2nd] [⊞] ↑ [ENTER] ← → → [←] 5 [ENTER]	
5. Switch to the Graph screen, which regraphs the edited function. *The thick border is around the Graph screen.*	[2nd] [⊞]	
6. Switch to the Y= Editor. Then open the Window Editor in its place.	[2nd] [⊞] [♦] [WINDOW]	
7. Open the Home screen. Then exit to a full-sized Home screen.	[2nd] [QUIT] [2nd] [QUIT]	

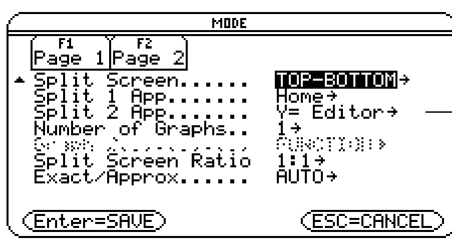
Setting and Exiting the Split Screen Mode

To set up a split screen, use the MODE dialog box to specify the applicable mode settings. After you set up the split screen, it remains in effect until you change it.

Setting the Split Screen Mode

1. Press [MODE] to display the MODE dialog box.
2. Because the modes related to split screens are listed on the second page of the MODE dialog box, either:
   - Use ⓓ to scroll down.
     — or —
   - Press [F2] to display Page 2.
3. Set the Split Screen mode to either of the following settings. For the procedure used to change a mode setting, refer to Chapter 2.

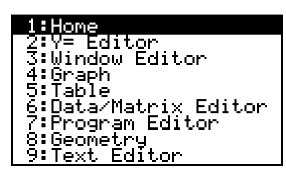
Split Screen Settings
TOP-BOTTOM
LEFT-RIGHT



When you set Split Screen = TOP-BOTTOM or LEFT-RIGHT, previously dimmed modes such as Split 2 App become active.

Setting the Initial Applications

Before pressing [ENTER] to close the MODE dialog box, you can use the Split 1 App and Split 2 App modes to select the applications you want to use.



Mode	Specifies the application in the:
Split 1 App	Top or left part of the split screen.
Split 2 App	Bottom or right part of the split screen.

Note: *In two-graph mode, described in Chapter 15, the same application can be in both parts of a split screen.*

If you set Split 1 App and Split 2 App to the same application, the TI-92 exits the split screen mode and displays the application full screen.

You can open different applications after the split screen is displayed, as described on page 81.

Other Modes that Affect a Split Screen

Mode	Description
Number of Graphs *Note: Leave this set to 1 unless you have read the applicable section in Chapter 15.*	Lets you set up and display two independent sets of graphs. This is an advanced graphing feature as described in "Using the Two-Graph Mode" in Chapter 15.
Split Screen Ratio	Sets the proportional sizes (1:1, 1:2, 2:1) of Split 1 App and Split 2 App.

Split Screens and Pixel Coordinates

The TI-92 has commands that use pixel coordinates to draw lines, circles, etc., on the Graph screen. The following chart shows how the Split Screen and Split Screen Ratio mode settings affect the number of pixels available on the Graph screen.

***Tip:** For a list of drawing commands, refer to "Drawing on the Graph Screen" in Chapter 17.*

***Note:** Due to the border that indicates the active application, split screens have a smaller displayable area than a full screen.*

		Split 1 App		Split 2 App	
Split	**Ratio**	**x**	**y**	**x**	**y**
FULL	N/A	0 – 238	0 – 102	N/A	N/A
TOP–BOTTOM	1:1	0 – 234	0 – 46	0 – 234	0 – 46
	1:2	0 – 234	0 – 26	0 – 234	0 – 68
	2:1	0 – 234	0 – 68	0 – 234	0 – 26
LEFT–RIGHT	1:1	0 – 116	0 – 98	0 – 116	0 – 98
	1:2	0 – 76	0 – 98	0 – 156	0 – 98
	2:1	0 – 156	0 – 98	0 – 76	0 – 98

Exiting the Split Screen Mode

Method 1: Press [MODE] to display the MODE dialog box. Then set Split Screen = FULL. When you press [ENTER] to close the dialog box, the full-sized screen shows the application specified in Split 1 App.

Method 2: Press [2nd] [QUIT] twice to display a full-sized Home screen.

When You Turn Off the TI-92

Turning the TI-92 off does not exit the split screen mode.

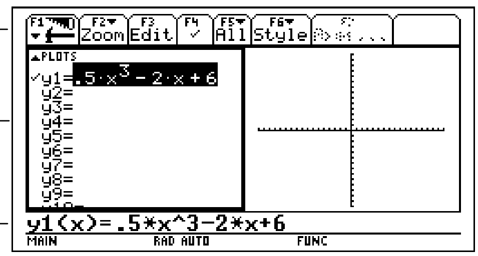
If the TI-92 is turned off:	When you turn the TI-92 on again:
When you press [2nd] [OFF]	The split screen is still in effect, but the Home screen is always displayed in place of the application that was active when you pressed [2nd] [OFF].
By the Automatic Power Down (APD) feature, or when you press [♦] [OFF].	The split screen is just as you left it.

Selecting the Active Application

With a split screen, only one of the two applications can be active at a time. You can easily switch between existing applications, or you can open a different application.

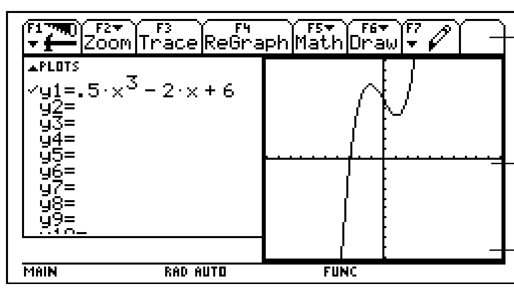
The Active Application

- The active application is indicated by a thick border.
- The toolbar and status line, which are always the full width of the display, are associated with the active application.
- For applications that have an entry line (such as the Home screen and Y= Editor), the entry line is the full width of the display *only when that application is active.*



Switching between Applications

Press [2nd] [⊞] (second function of [APPS]) to switch from one application to the other.



Opening a Different Application

Note: *Also refer to "Using [2nd] [QUIT] to Display the Home Screen" on page 82.*

Method 1:
1. Use [2nd] [⊞] to switch to the application you want to replace.
2. Use [APPS] or [♦] (such as [APPS] 1 or [♦] [HOME]) to select the new application.

If you select an application that is already displayed, the TI-92 switches to that application.

Method 2:
1. Press [MODE] and then [F2].
2. Change Split 1 App and/or Split 2 App.

If you set Split 1 App and Split 2 App to the same application, the TI-92 exits the split screen mode and displays the application full screen.

Note: *In two-graph mode, described in Chapter 15, the same application can be in both parts of a split screen.*

Using 2nd [QUIT] to Display the Home Screen

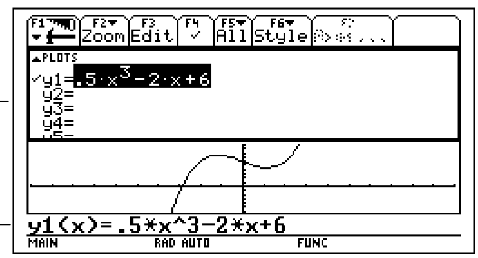
Tip: Pressing 2nd [QUIT] twice always exits the split screen mode.

If the Home screen:	Pressing 2nd [QUIT]:
Is not already displayed	Opens the Home screen in place of the active application.
Is displayed, but is not the active application	Switches to the Home screen and makes it the active application.
Is the active application	Exits the split screen mode and displays a full-sized Home screen.

When Using a Top-Bottom Split

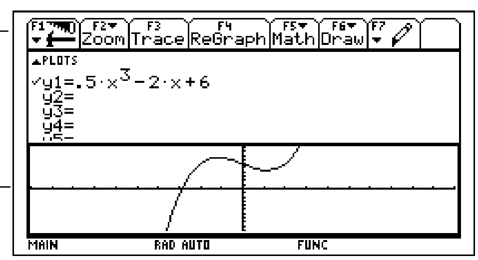
When you select a TOP-BOTTOM split, remember that the entry line and the toolbar are always associated with the active application. For example:

Entry line is for the active Y= Editor, *not* the Graph screen.



Note: Both Top-Bottom and Left-Right splits use the same methods to select an application.

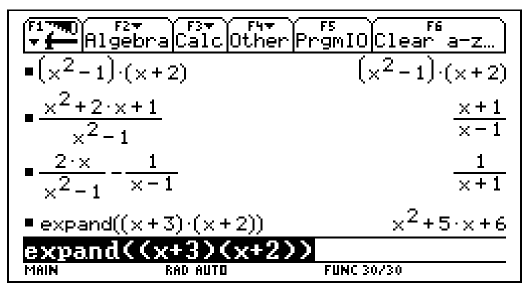
Toolbar is for the active Graph screen, *not* the Y= Editor.



Symbolic Manipulation

6

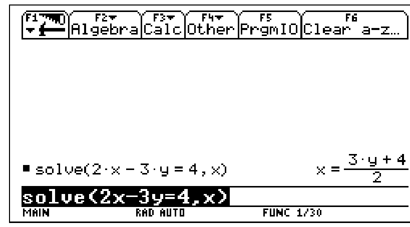
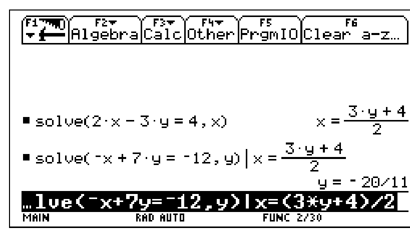
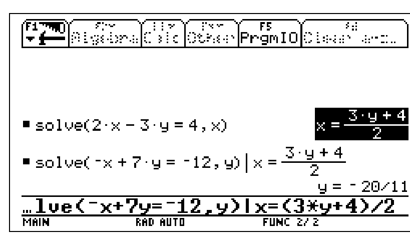
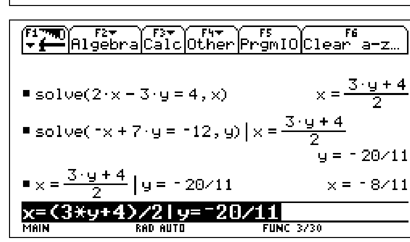
This chapter is an overview of the fundamentals of using symbolic manipulation to perform algebraic or calculus operations.



You can easily perform symbolic calculations from the Home screen.

Preview of Symbolic Manipulation

Solve the system of equations 2x − 3y = 4 and ⁻x + 7y = ⁻12. Solve the first equation so that x is expressed in terms of y. Substitute the expression for x into the second equation, and solve for the value of y. Then substitute the y value back into the first equation to solve for the value of x.

Steps	Keystrokes	Display
1. Display the Home screen and clear the entry line. Solve the equation 2x − 3y = 4 for x. *[F2] 1 selects* **solve(** *from the* Algebra *menu. You can also type* **solve(** *directly from the keyboard.*	[♦] [HOME] [CLEAR] [CLEAR] [F2] 1 2 X [−] 3 Y [=] 4 [,] X [)] [ENTER]	
2. Begin to solve the equation ⁻x + 7y = ⁻12 for y, but do not press [ENTER] yet.	[F2] 1 [(-)] X [+] 7 Y [=] [(-)] 1 2 [,] Y [)]	
3. Use the “with” operator ([2nd] K) to substitute the expression for x that was calculated from the first equation. This gives the value of y. *The “with” operator is displayed as* | *on the screen.* *Use the auto-paste feature to highlight the last answer in the history area and paste it to the entry line.*	[2nd] K [⊙] [ENTER] [ENTER]	
4. Highlight the equation for x in the history area.	[⊙] [⊙] [⊙]	
5. Auto-paste the highlighted expression to the entry line. Then substitute the value of y that was calculated from the second equation. The solution is: x = ⁻8/11 and y = ⁻20/11	[ENTER] [2nd] K [⊙] [ENTER] [ENTER]	

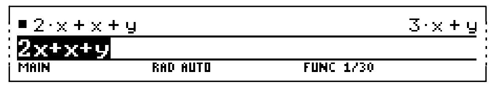
Using Undefined or Defined Variables

When performing algebraic or calculus operations, it is important that you understand the effect of using undefined and defined variables. Otherwise, you may get a number for a result instead of the algebraic expression that you anticipated.

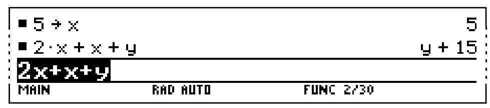
How Undefined and Defined Variables Are Treated

When you enter an expression that contains a variable, the TI-92 treats the variable in one of two ways.

- If the variable is undefined, it is treated as an algebraic symbol.



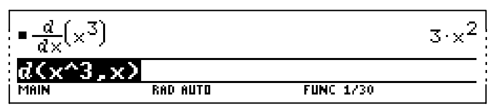
- If the variable is defined (even if defined as 0), its value replaces the variable.



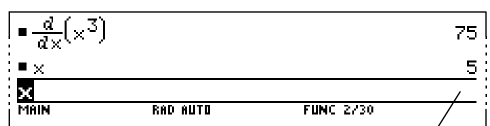
Tip: *When defining a variable, it's a good practice to use more than one character in the name. Leave one-character names undefined for symbolic calculations.*

To see why this is important, suppose you want to find the first derivative of x^3 with respect to x.

- If x is undefined, the result is in the form you probably expected.



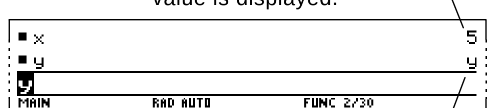
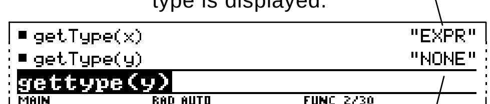
- If x is defined, the result may be in a form you did not expect.



Unless you knew that 5 had been stored to x previously, the answer 75 could be misleading.

Determining If a Variable Is Undefined

Note: *Use* [2nd] [VAR-LINK] *to view a list of defined variables, as described in Chapter 18.*

Method:	Example:
Enter the variable name.	If defined, the variable's value is displayed.  If undefined, the variable name is displayed.
Use the **getType** function.	If defined, the variable's type is displayed.  If undefined, "NONE" is displayed.

Deleting a Defined Variable

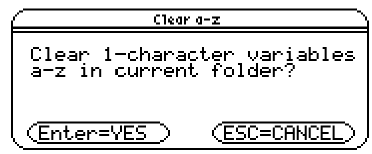
You can “undefine” a defined variable by deleting it.

To delete:	Do this:
One or more specified variables	Use the **DelVar** function.

You can also delete variables by using the VAR-LINK screen ([2nd] [VAR-LINK]) as described in Chapter 18.

__Note:__ For information about folders, refer to Chapter 10.

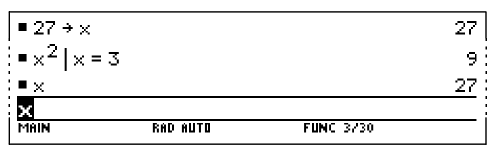
To delete:	Do this:
All one-letter variables (a – z) in the current folder	From the Home screen, press [F6] Clear a-z. You will be prompted to press [ENTER] to confirm the deletion.



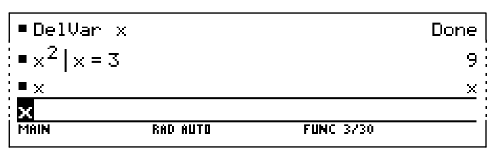
Temporarily Overriding a Variable

By using [2nd] K to type the “with” operator (|), you can:

- Temporarily override a variable’s defined value.



- Temporarily define a value for an undefined variable.



__Note:__ For more information about the | operator, refer to page 93.

Using Exact, Approximate, and Auto Modes

The Exact/Approx mode settings, which are described briefly in Chapter 2, directly affect the precision and accuracy with which the TI-92 calculates a result. This section describes these mode settings as they relate to symbolic manipulation.

EXACT Setting

When Exact/Approx = EXACT, the TI-92 uses exact rational arithmetic with up to 614 digits in the numerator and 614 digits in the denominator. The EXACT setting:

- Transforms irrational numbers to standard forms as much as possible without approximating them. For example, $\sqrt{12}$ transforms to $2\sqrt{3}$ and ln(1000) transforms to 3 ln(10).
- Converts floating-point numbers to rational numbers. For example, 0.25 transforms to 1/4.

The functions **solve**, **cSolve**, **zeros**, **cZeros**, **factor**, ∫, **fMin**, and **fMax** use only exact symbolic algorithms. These functions do not compute approximate solutions in the EXACT setting.

- Some equations, such as $2^{-x} = x$, have solutions that cannot all be finitely represented in terms of the functions and operators on the TI-92.
- With this kind of equation, EXACT will not compute approximate solutions. For example, $2^{-x} = x$ has an approximate solution $x \approx 0.641186$, but it is not displayed in the EXACT setting.

Advantages	Disadvantages
Results are exact.	As you use more complicated rational numbers and irrational constants, calculations can: • Use more memory, which may exhaust the memory before a solution is completed. • Take more computing time. • Produce bulky results that are harder to comprehend than a floating-point number.

APPROXIMATE Setting

When Exact/Approx = APPROXIMATE, the TI-92 converts rational numbers and irrational constants to floating-point. However, there are exceptions:

- Certain built-in functions that expect one of their arguments to be an integer will convert that number to an integer if possible. For example: ***d***(y(x), x, 2.0) transforms to ***d***(y(x), x, 2).
- Whole-number floating-point exponents are converted to integers. For example: $x^{2.0}$ transforms to x^2 even in the APPROXIMATE setting.

Functions such as **solve** and ∫ (integrate) can use both exact symbolic and approximate numeric techniques. These functions skip all or some of their exact symbolic techniques in the APPROXIMATE setting.

Advantages	Disadvantages
If exact results are not needed, this might save time and/or use less memory than the EXACT setting.	Results with undefined variables or functions often exhibit incomplete cancellation. For example, a coefficient that should be 0 might be displayed as a small magnitude such as 1.23457E-11.
Approximate results are sometimes more compact and comprehensible than exact results.	Symbolic operations such as limits and integration are less likely to give satisfying results in the APPROXIMATE setting.
If you do not plan to use symbolic computations, approximate results are similar to familiar, traditional numeric calculators.	Approximate results are sometimes less compact and comprehensible than exact results. For example, you may prefer to see 1/7 instead of .142857.

AUTO Setting

When Exact/Approx = AUTO, the TI-92 uses exact rational arithmetic wherever all of the operands are rational numbers. Otherwise, floating-point arithmetic is used after converting any rational operands to floating-point. In other words, floating-point is "infectious." For example:

1/2 – 1/3 transforms to 1/6
but
0.5 – 1/3 transforms to .166666666667

This floating-point infection does not leap over barriers such as undefined variables or between elements of lists or matrices. For example:

(1/2 - 1/3) x + (0.5 – 1/3) y transforms to x/6 + .166666666667 y
and
{1/2 - 1/3, 0.5 – 1/3} transforms to {1/6, .166666666667}

In the AUTO setting, functions such as **solve** determine as many solutions as possible exactly, and then use approximate numerical methods if necessary to determine additional solutions. Similarly, ∫ (integrate) uses approximate numerical methods if appropriate where exact symbolic methods fail.

Advantages	Disadvantages
You see exact results when practical, and approximate numeric results when exact results are impractical. You can often control the format of a result by choosing to enter some coefficients as either rational or floating-point numbers.	If you are interested only in exact results, some time may be wasted seeking approximate results. If you are interested only in approximate results, some time may be wasted seeking exact results. Moreover, you might exhaust the memory seeking those exact results.

Automatic Simplification

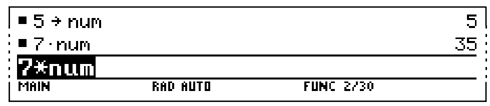
When you type an expression on the entry line and press [ENTER], the TI-92 automatically simplifies the expression according to its default simplification rules.

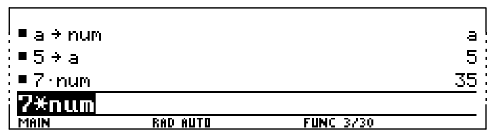
Default Simplification Rules

All of the following rules are applied automatically. You do not see intermediate results.

- If a variable has a defined value, that value replaces the variable.

  If the variable is defined in terms of another variable, the variable is replaced with its “lowest level” value (called infinite lookup).



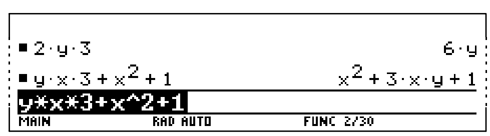


Note: For information about folders, refer to Chapter 10.

  Default simplification does not modify variables that use pathnames to indicate a folder. For example, x+class\x does not simplify to 2x.

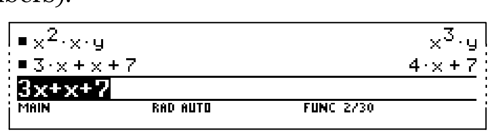
Note: Refer to “Delayed Simplification for Certain Built-In Functions” on page 92.

- For functions:
  - The arguments are simplified. (Some built-in functions delay simplification of some of their arguments.)
  - If the function is a built-in or user-defined function, the function definition is applied to the simplified arguments. Then the functional form is replaced with this result.
- Numeric subexpressions are combined.
- Products and sums are sorted into order.

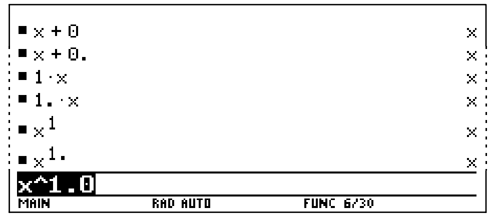


  Products and sums involving undefined variables are sorted according to the first letter of the variable name.

  - Undefined variables r through z are assumed to be true variables, and are placed in alphabetical order at the beginning of a sum.
  - Undefined variables a through q are assumed to represent constants, and are placed in alphabetical order at the end of a sum (but before numbers).
- Similar factors and similar terms are collected.

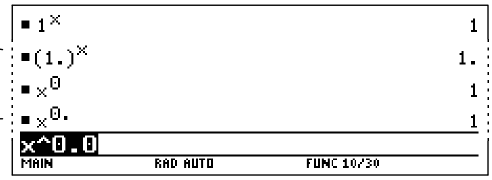


- Identities involving zeros and ones are exploited.

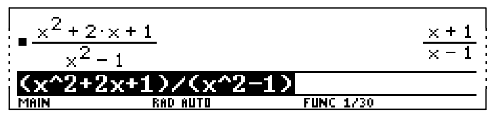


This floating-point number causes numeric results to be shown as floating-point.

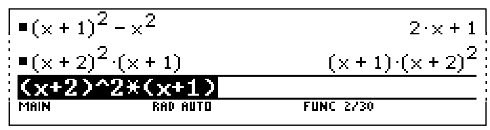
If a floating-point whole number is entered as an exponent, it is treated as an integer (and does not produce a floating-point result).



- Polynomial greatest common divisors are canceled.



- Polynomials are expanded unless no key cancellation can occur.



No key cancellation

- Common denominators are formed unless no key cancellation can occur.

No key cancellation

- Functional identities are exploited. For example:

  ln(2x) = ln(2) + ln(x)
  and
  $\sin(x)^2 + \cos(x)^2 = 1$

How Long Is the Simplification Process?

Depending on the complexity of an entry, result, or intermediate expression, it can take a long time to expand an expression and cancel common divisors as necessary for simplification.

To interrupt a simplification process that is taking too long, press [ON]. You can then try simplifying only a portion of the expression. (Auto-paste the entire expression on the entry line, and then delete the unwanted parts.)

Delayed Simplification for Certain Built-In Functions

Usually, variables are automatically simplified to their lowest possible level before they are passed to a function. For certain functions, however, complete simplification is delayed until after the function is performed.

Functions that Use Delayed Simplification

Functions that use delayed simplification have a required *var* argument that performs the function with respect to a variable. These functions have at least two arguments with the general form:

function(*expression*, *var* [, ...])

Note: *Not all functions that use a var argument use delayed simplification.*

For example: **solve**(x^2−x−2=0,x)
d(x^2−x−2,x)
∫(x^2−x−2,x)
limit(x^2−x−2,x,5)

For a function that uses delayed simplification:

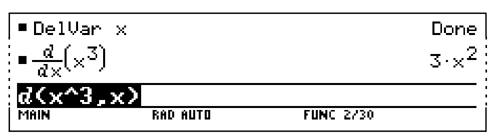
1. The *var* variable is simplified to the lowest level at which it remains a variable (even if it could be further simplified to a non-variable value).
2. The function is performed using the variable.
3. If *var* can be further simplified, that value is then substituted into the result.

Note: *You may or may not want to define a numeric value for var, depending on the situation.*

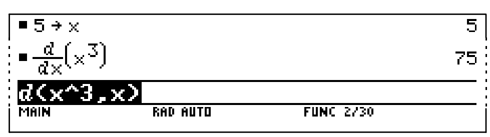
Note: *The example to the right finds the derivative of x^3 at x=5. If x^3 was initially simplified to 75, you would find the derivative of 75, which is not what you want.*

For example:

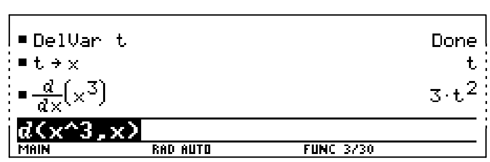
x cannot be simplified.



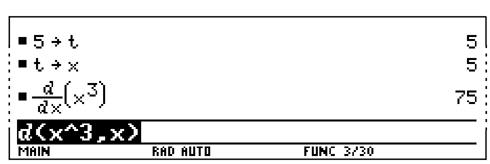
x is not simplified. The function uses x^3, and then substitutes 5 for x.



x is simplified to t. The function uses t^3.



x is simplified to t. The function uses t^3, and then substitutes 5 for t.



Substituting Values and Setting Constraints

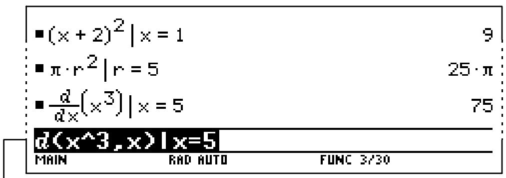
The “with” operator (|) lets you temporarily substitute values into an expression or specify domain constraints.

Typing the “With” Operator

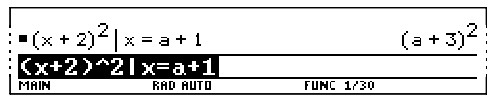
To type the “with” operator (|), type [2nd] K on the QWERTY keyboard.

Substituting for a Variable

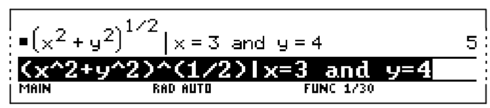
For every occurrence of a specified variable, you can substitute a numeric value or an expression.



First derivative of x^3 at x = 5

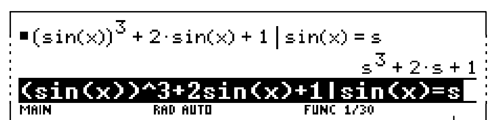


To substitute for multiple variables at the same time, use the Boolean **and** operator.



Substituting for a Simple Expression

For every occurrence of a simple expression, you can substitute a variable, numeric value, or another expression.

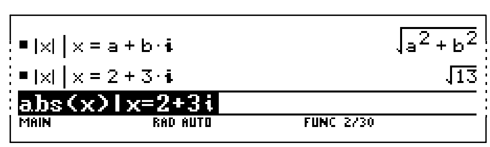


Substituting s for sin(x) shows that the expression is a polynomial in terms of sin(x).

By replacing a commonly used (or long) term, you can display results in a more compact form.

Substituting Complex Values

You can substitute complex values just as you would for other values.



***Note:** For an overview of complex numbers, refer to Appendix B.*

All undefined variables are treated as real numbers in symbolic calculations. To perform complex symbolic analysis, you must define a complex variable. For example:

x+y*i*→z

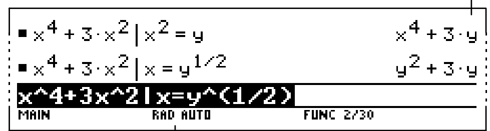
Then you can use z as a complex variable.

Tip:** To get the complex* ***i*, press [2nd] [i]. Do not simply type I on the keyboard.*

Be Aware of the Limitations of Substitutions

- Substitution occurs only where there is an *exact* match for the substitution.

Only x^2 was replaced, not x^4.



Define the substitution in simpler terms for a more complete substitution.

- Infinite recursions can occur when you define a substitution variable in terms of itself.

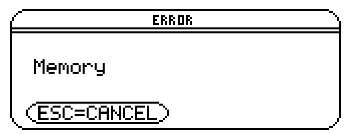
Substitutes sin(x+1), sin(x+1+1), sin(x+1+1+1), etc.

```
sin(x)|x=x+1
sin(x)|x=y and y=x
```

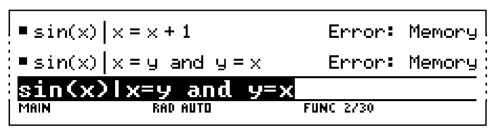
Substitutes sin(y), sin(x), sin(y), sin(x), repeatedly.

When you enter a substitution that causes an infinite recursion:

– An error message is displayed.



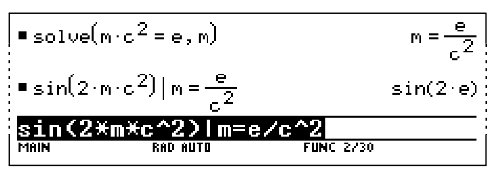
– When you press [ESC], an error is shown in the history area.



- Internally, an expression is sorted according to the automatic simplification rules. Therefore, products and sums may not match the order in which you entered them.

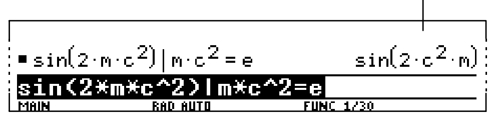
Tip: ***Use the*** **solve** *function to help determine the single-variable substitution.*

– As a general rule, you should substitute for a single variable.



– Substituting for more general expressions (either m•c²=e or c²•m=e) may not work as you anticipate.

No match for substitution



Specifying Domain Constraints

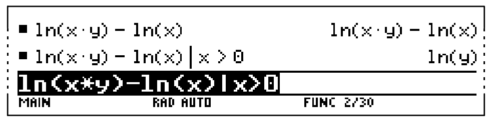
Many identities and transformations are valid for only a particular domain. For example:

ln(x∗y) = ln(x) + ln(y)	only if x and/or y is not negative
$\sin^{-1}(\sin(\theta)) = \theta$	only if $\theta \geq {}^{-}\pi/2$ and $\theta \leq \pi/2$ radians

Use the “with” operator to specify the domain constraint.

Tip: *Enter* ln(x∗y) *instead of* ln(xy)*; otherwise, xy is interpreted as a single variable named xy.*

Because ln(x∗y) = ln(x) + ln(y) is not always valid, the logarithms are not combined.

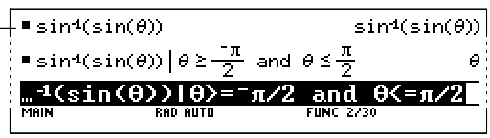


With a constraint, the identity is valid and the expression is simplified.

Tip: *For ≥ or ≤, type >= or <= from the keyboard. You can also use* [2nd] [MATH] 8 *or* [2nd] [CHAR] 2 *to select them from a menu.*

Because $\sin^{-1}(\sin(\theta)) = \theta$ is not always valid, the expression is not simplified.

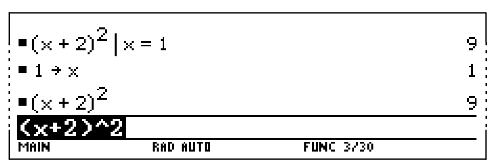
[2nd] [SIN-1]



With a constraint, the expression can be simplified.

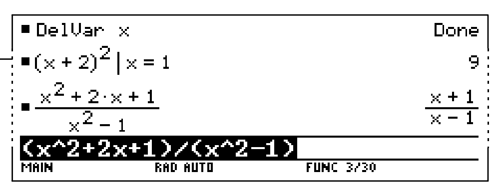
Using Substitutions vs. Defining a Variable

In many cases, you can achieve the same effect as a substitution by defining the variable.



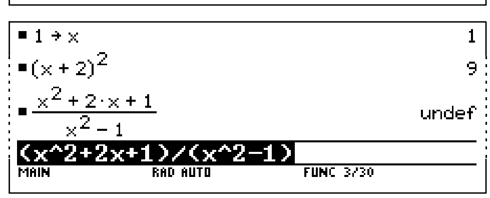
However, substitution is preferable for most cases because the variable is defined only for the current calculation and does not accidentally affect later calculations.

Substituting x=1 does not affect the next calculation.



Caution: *After x is defined, it can affect all calculations that involve x (until you delete x).*

Storing 1→x affects the subsequent calculations.



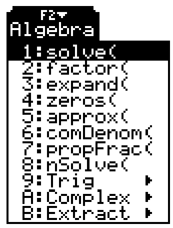
Overview of the Algebra Menu

You can use the [F2] **Algebra** toolbar menu to select the most commonly used algebraic functions.

The Algebra Menu

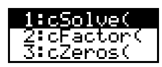
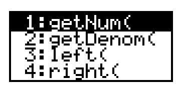
__Note:__ For a complete description of each function and its syntax, refer to Appendix A.

From the Home screen, press [F2] to display:



This menu is also available from the MATH menu. Press [2nd] [MATH] and then select 9:Algebra.

Menu Item	Description
solve	Solves an expression for a specified variable. This returns real solutions only, regardless of the Complex Format mode setting. (For complex solutions, select A:Complex from the Algebra menu.)
factor	Factors an expression with respect to all its variables or with respect to only a specified variable.
expand	Expands an expression with respect to all its variables or with respect to only a specified variable.
zeros	Determines the values of a specified variable that make an expression equal to zero.
approx	Evaluates an expression using floating-point arithmetic, where possible. This is equivalent to using [MODE] to set Exact/Approx = APPROXIMATE (or using [♦] [ENTER] to evaluate an expression).
comDenom	Calculates a common denominator for all terms in an expression and transforms the expression into a reduced ratio of a numerator and denominator.
propFrac	Returns an expression as a proper fraction expression.
nSolve	Calculates a single solution for an equation as a floating-point number (as opposed to **solve**, which may display several solutions in a rational or symbolic form).

Menu Item	Description
Trig	Displays the submenu:  tExpand — Expands trig expressions with angle sums and multiple angles. tCollect — Collects the products of integer powers of trig functions into angle sums and multiple angles. **tCollect** is the opposite of **tExpand**.
Complex	Displays the submenu:  These are the same as **solve**, **factor**, and **zeros**; but they also compute complex results.
Extract	Displays the submenu:  getNum — Applies **comDenom** and then returns the resulting numerator. getDenom — Applies **comDenom** and then returns the resulting denominator. left — Returns the left-hand side of an equation or inequality. right — Returns the right-hand side of an equation or inequality.

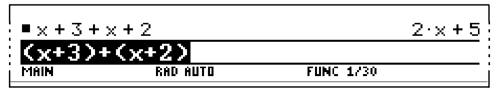
Note: *The* **left** *and* **right** *functions are also used to return a specified number of elements or characters from the left or right side of a list or character string.*

Common Algebraic Operations

This section gives examples for some of the functions available from the [F2] **Algebra** toolbar menu. For complete information about any function, refer to Appendix A. Some algebraic operations do not require a special function.

Adding or Dividing Polynomials

You can add or divide polynomials directly, without using a special function.



Factoring and Expanding Polynomials

Use the **factor** ([F2] 2) and **expand** ([F2] 3) functions.

factor(*expression* [,*var*])

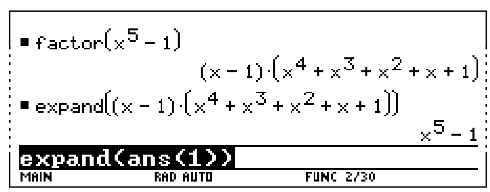
└─ for factoring with respect to a variable

expand(*expression* [,*var*])

└─ for partial expansion with respect to a variable

Factor $x^5 - 1$. Then expand the result.

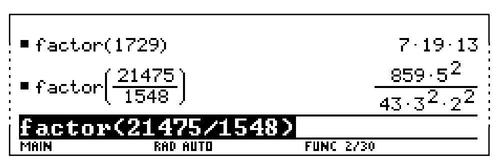
Notice that **factor** and **expand** perform opposite operations.



Finding Prime Factors of a Number

The **factor** ([F2] 2) function lets you do more than simply factor an algebraic polynomial.

You can find prime factors of a rational number (either an integer or a ratio of integers).

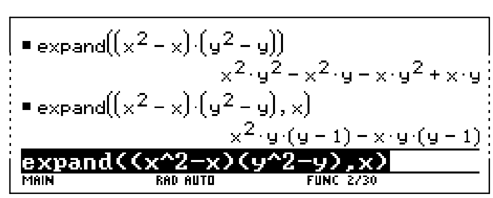


Finding Partial Expansions

With the **expand** ([F2] 3) function's optional *var* value, you can do a partial expansion that collects similar powers of a variable.

Do a full expansion of $(x^2 - x)(y^2 - y)$ with respect to all variables.

Then do a partial expansion with respect to x.

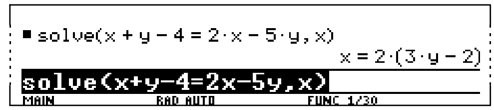


Solving an Equation

Use the **solve** ([F2] 1) function to solve an equation for a specified variable.

solve(*equation*, *var*)

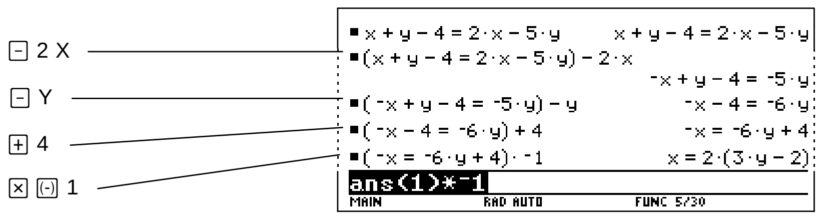
Solve x + y − 4 = 2x − 5y for x.



Notice that **solve** displays only the final result.

To see intermediate results, you can manually solve the equation step-by-step.

Note: An operation such as [−] 2 X *subtracts 2x from both sides.*



Notice that the result in this example is automatically transformed to x = 2 (3y − 2). You can use **expand** to obtain x = 6y − 4.

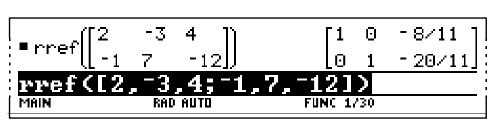
Solving a System of Linear Equations

Consider a set of two equations with two unknowns:

2x − 3y = 4
⁻x + 7y = ⁻12

To solve this system of equations, use any of the following methods.

Note: *The* **simult** *and* **rref** *matrix functions are not on the* [F2] Algebra *menu. Use* [2nd] [MATH] 4 *or* [2nd] [CATALOG].

Method	Example
Use the **solve** function with substitution (|).	Refer to the preview at the beginning of this chapter, which solved for x = ⁻8/11 and y = ⁻20/11.
Use the **simult** function with a matrix.	Enter the coefficients as a matrix and the results as a constant column matrix. 
Use the **rref** function with a matrix.	Enter the coefficients as an augmented matrix. 

Finding the Zeros of an Expression

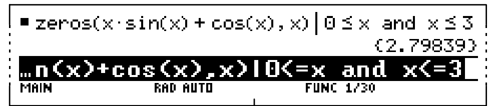
Tip: For ≥ or ≤, type >= or <= from the keyboard. You can also use [2nd] [MATH] 8 *or* [2nd] [CHAR] 2 *to select them from a menu.*

Use the **zeros** ([F2] 4) function.

zeros(*expression*, *var*)

Use the expression x ∗ sin(x) + cos(x).

Find the zeros with respect to x in the interval 0 ≤ x and x ≤ 3.



Use the “with” operator ([2nd] K) to specify the interval.

Finding Proper Fractions and Common Denominators

Use the **propFrac** ([F2] 7) and **comDenom** ([F2] 6) functions.

propFrac(*rational expression* [,*var*])

└ for proper fractions with respect to a variable

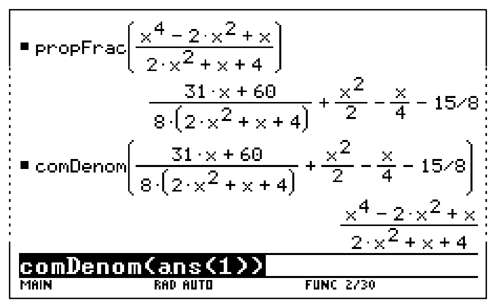
comDenom(*expression* [,*var*])

└ for common denominators that collect similar powers of this variable

Note: *You can use* **comDenom** *with an expression, list, or matrix.*

Find a proper fraction for the expression (x^4−2x^2+ x) / (2x^2+ x + 4).

Then transform the answer into a ratio of a fully expanded numerator and a fully expanded denominator.



If you do this example on your TI-92, the **propFrac** function scrolls off the top of the screen.

Notice that **propFrac** and **comDenom** perform opposite operations.

In this example:

- $\frac{31\,x + 60}{8}$ is the remainder of x^4−2x^2+x divided by 2x^2+x+4.
- $\frac{x^2}{2} - \frac{x}{4} - 15/8$ is the quotient.

Overview of the Calc Menu

You can use the F3 **Calc** toolbar menu to select commonly used calculus functions.

The Calc Menu

From the Home screen, press F3 to display:

This menu is also available from the MATH menu. Press 2nd [MATH] and then select A:Calculus.

***Note:** For a complete description of each function and its syntax, refer to Appendix A.*

***Note:** The d symbol for differentiate is a special symbol. It is not the same as typing D on the keyboard.*

Menu Item	Description
d differentiate	Differentiates an expression with respect to a specified variable.
∫ integrate	Integrates an expression with respect to a specified variable.
limit	Calculates the limit of an expression with respect to a specified variable.
Σ sum	Evaluates an expression at discrete variable values within a range and then calculates the sum.
Π product	Evaluates an expression at discrete variable values within a range and then calculates the product.
fMin	Finds candidate values of a specified variable that minimize an expression.
fMax	Finds candidate values of a specified variable that maximize an expression.
arcLen	Returns the arc length of an expression with respect to a specified variable.
taylor	Calculates a Taylor polynomial approximation to an expression with respect to a specified variable.
nDeriv	Calculates the numerical derivative of an expression with respect to a specified variable.
nInt	Calculates an integral as a floating-point number using quadrature (an approximation using weighted sums of integrand values).

Common Calculus Operations

This section gives examples for some of the functions available from the [F3] **Calc** toolbar menu. For complete information about any calculus function, refer to Appendix A.

Integrating and Differentiating

Use the ∫ integrate ([F3] 2) and ***d*** differentiate ([F3] 1) functions.

∫ (*expression*, *var* [,*low*] [,*up*])

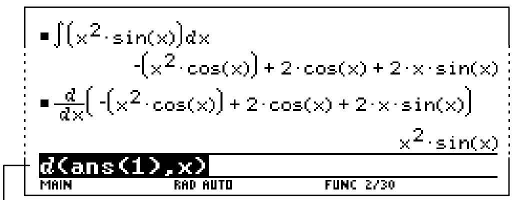
— lets you specify limits or a constant of integration

d (*expression*, *var* [,*order*])

Note: *You can integrate an expression only; you can differentiate an expression, list, or matrix.*

Integrate $x^2*\sin(x)$ with respect to x.

Differentiate the answer with respect to x.



To get *d*, use [F3] 1. Do not simply type D on the keyboard.

Finding a Limit

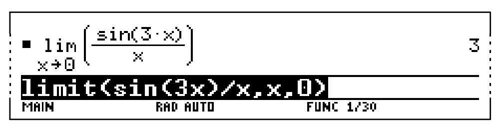
Use the **limit** ([F3] 3) function.

limit(*expression*, *var*, *point* [,*direction*])

— negative = from left
positive = from right
omitted or 0 = both

Note: *You can find a limit for an expression, list, or matrix.*

Find the limit of sin(3x) / x as x approaches 0.



Finding a Taylor Polynomial

Use the **taylor** ([F3] 9) function.

taylor(*expression*, *var*, *order* [,*point*])

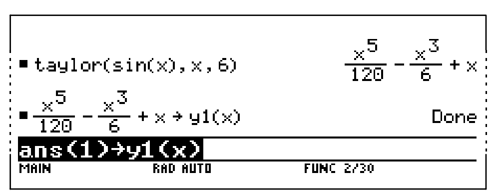
— if omitted, expansion point is 0

Important: *Degree-mode scaling by π/180 may cause calculus application results to appear in a different form.*

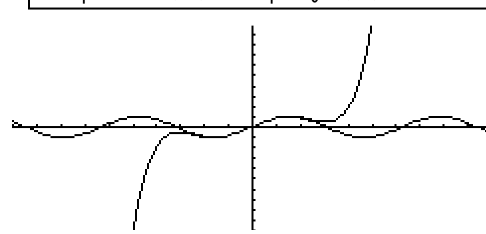
Find a 6th order Taylor polynomial for sin(x) with respect to x.

Store the answer as a user-defined function named y1(x).

Then graph sin(x) and the Taylor polynomial.




```
Graph sin(x):Graph y1(x)
```



You can use a user-defined function as an argument for the TI-92's built-in algebra and calculus functions.

For Information about Creating a User-Defined Function

Refer to:

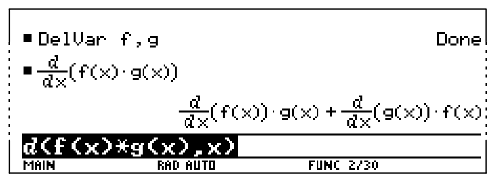
- "Creating and Evaluating User-Defined Functions" in Chapter 10.
- "Graphing a Function Defined on the Home Screen" and "Graphing a Piecewise Defined Function" in Chapter 15.
- "Overview of Entering a Function" in Chapter 17.

Undefined Functions

You can use functions such as f(x), g(t), r(θ), etc., that have not been assigned a definition. These "undefined" functions yield symbolic results. For example:

Use **DelVar** to ensure that f(x) and g(x) are not defined.

Then find the derivative of f(x)*g(x) with respect to x.



Tip: *To select* d *from the* Calc *toolbar menu, press* F3 1.

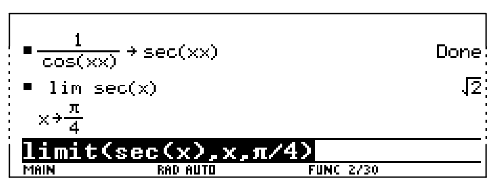
Single-Statement Functions

You can use user-defined functions consisting of a single expression. For example:

- Use STO▸ to create a user-defined secant function, where:

  $$\sec(xx) = \frac{1}{\cos(xx)}$$

  Then find the limit of sec(x) as x approaches π/4.



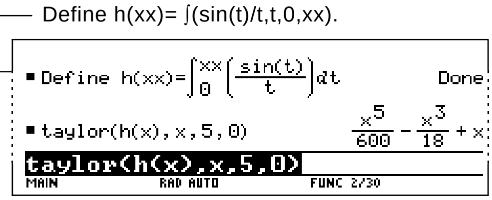
Tip: *To select* ***limit*** *from the* Calc *toolbar menu, press* F3 3.

- Use **Define** to create a user-defined function h(xx), where:

  $$h(xx)=\int_0^{xx} \sin(t) / t$$

  Then find a 5th order Taylor polynomial for h(x) with respect to x.

Define h(xx)= ∫(sin(t)/t,t,0,xx).



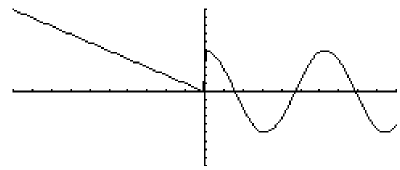
Tip: *To select* ∫ *from the* Calc *toolbar menu, press* F3 2. *To select* ***taylor****, press* F3 9.

Multi-Statement vs. Single-Statement Functions

Multi-statement user-defined functions should be used as an argument for numeric functions (such as **nDeriv** and **nInt**) only.

In some cases, you may be able to create an equivalent single-statement function. For example, consider a piecewise function with two pieces.

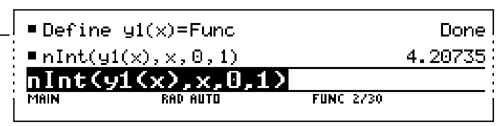
When:	Use expression:
x < 0	⁻x
x ≥ 0	5 cos(x)



- Create a multi-statement user-defined function with the form:

```
Func
  If x<0 Then
  Return ⁻x
Else
  Return 5cos(x)
EndIf
EndFunc
```

Define y1(x)=Func:If x<0 Then: ... :EndFunc



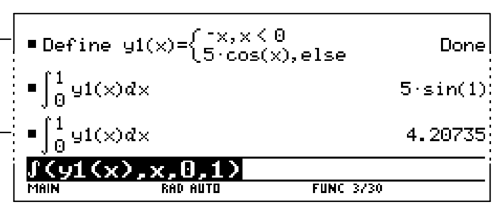
Tip: ***To select*** **nInt** ***from the*** Calc ***toolbar menu, press*** [F3] B.

Then numerically integrate y1(x) with respect to x.

- Create an equivalent single-statement user-defined function.

Define y1(x)=when(x<0, ⁻x, 5cos(x))

Use the TI-92's built-in **when** function.



Tip: ***To select ∫ from the*** Calc ***toolbar menu, press*** [F3] 2.

Then integrate y1(x) with respect to x.

Press [♦] [ENTER] for a floating-point result.

If You Get an Out-of-Memory Error

The TI-92 stores intermediate results in memory and then deletes them when the calculation is complete. Depending on the complexity of the calculation, the TI-92 may run out of memory before a result can be calculated.

Freeing Up Memory

- Delete unneeded variables, particularly large-sized ones.
  - Use [2nd] [VAR-LINK] as described in Chapter 18 to view and delete variables.
- On the Home screen:
  - Clear the history area ([F1] 8) or delete unneeded history pairs.
  - You can also use [F1] 9 to reduce the number of history pairs that will be saved.
- Use [MODE] to set Exact/Approx = APPROXIMATE. (For results that have a large number of digits, this uses less memory than AUTO or EXACT. For results that have only a few digits, this uses more memory.)

Simplifying Problems

- Split the problem into parts.
  - Split **solve**(a∗b=0,*var*) into **solve**(a=0,*var*) and **solve**(b=0,*var*). Solve each part and combine the results.
- If several undefined variables occur only in a certain combination, replace that combination with a single variable.
  - If m and c occur only as $m*c^2$, substitute e for $m*c^2$.
  - In the expression $\frac{(a+b)^2 + \sqrt{(a+b)^2}}{1 - (a+b)^2}$, substitute c for (a+b) and use $\frac{c^2 + \sqrt{c^2}}{1 - c^2}$. In the solution, replace c with (a+b).
- For expressions combined over a common denominator, replace sums in denominators with unique new undefined variables.
  - In the expression $\frac{x}{\sqrt{a^2+b^2} + c} + \frac{y}{\sqrt{a^2+b^2} + c}$, substitute d for $\sqrt{a^2+b^2} + c$ and use $\frac{x}{d} + \frac{y}{d}$. In the solution, replace d with $\sqrt{a^2+b^2} + c$.
- Substitute known numeric values for undefined variables at an earlier stage, particularly if they are simple integers or fractions.
- Reformulate a problem to avoid fractional powers.
- Omit relatively small terms to find an approximation.

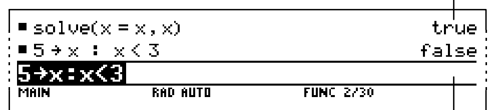
Special Constants Used in Symbolic Manipulation

The result of a calculation may include one of the special constants described in this section. In some cases, you may also need to enter a constant as part of your entry.

true, false

These indicate the result of an identity or a Boolean expression.

x=x is true for any value of x.



5<3 is false.

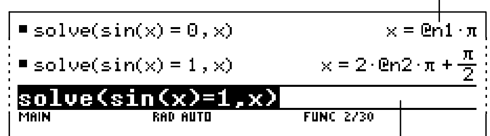
@n1 ... @n255

This notation indicates an “arbitrary integer” that represents any integer.

Tip: For @, press [2nd] *R.*

When an arbitrary integer occurs multiple times in the same session, each occurrence is numbered consecutively. After it reaches 255, arbitrary integer consecutive numbering restarts at @n0. There is no way to reset this number other than resetting the TI-92.

A solution is at every integer multiple of π.



Both @n1 and @n2 represent any arbitrary integer, but this notation identifies separate arbitrary integers.

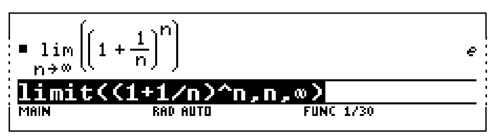
∞, *e*

∞ represents infinity, and *e* represents the constant 2.71828... (base of the natural logarithms).

These constants are often used in entries as well as results.

Tip: For ∞, press [2nd] [∞] *(same as* [2nd] *J).*

Tip: For e, press [2nd] [e^x]*. This is not the same as typing E on the keyboard.*



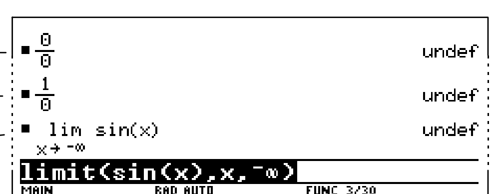
undef

This indicates that the result is undefined.

Mathematically undefined

±∞ (undetermined sign)

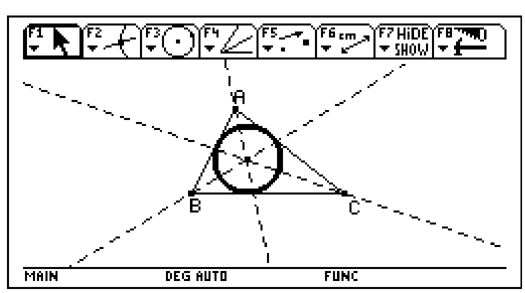
Non-unique limit



Geometry

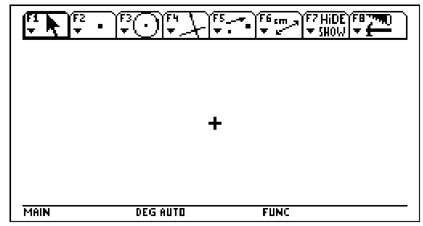
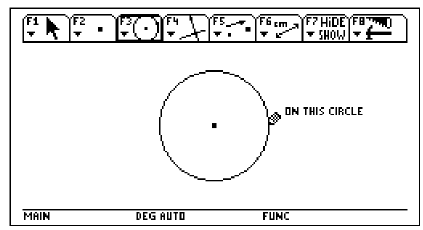
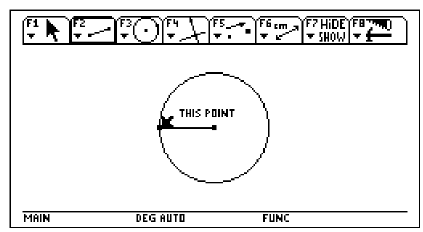
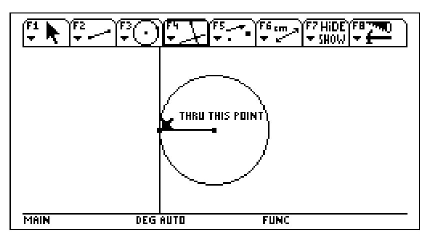
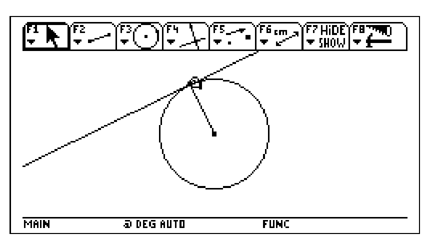
7

This chapter describes the Geometry application of the TI-92. It provides descriptions, procedures, illustrations, and examples for using the TI-92 to perform analytic, transformational, and Euclidean geometric functions.



Preview of Geometry

Create a circle and construct a perpendicular line that is tangent to the circle.

Steps	Keystrokes	Display
1. Open a geometry session. *In this example,* `G2` *is the name of the construction. You can use up to eight characters to name constructions.*	[APPS] 8 3 ▽ G 2 [ENTER] [ENTER]	
2. Construct a circle. *Pressing [ENTER] the first time defines the center point. Pressing [ENTER] the second time draws the circle.*	[F3] 1 [ENTER] ▷ (hold momentarily to expand the circle) [ENTER]	
3. Construct a segment from the center of the circle and attach it to the circumference.	[F2] 5 ◁ (until you see "THIS POINT") [ENTER] ◁ (until you see "ON THIS CIRCLE") [ENTER]	
4. Construct a line perpendicular to the segment at the intersection point of the segment and the circle. *Observe each displayed message before pressing [ENTER].* *The resultant perpendicular line is tangent to the circle.*	[F4] 1 [ENTER] [ENTER]	
5. Observe what happens when the endpoint of the segment is dragged around the circle.	Press and hold [✋], then press the cursor pad.	

Learning the Basics

This section describes the basic operations that you need to know, such as selecting items from the various menus, navigating with the cursor pad, and starting a construction.

Starting Geometry

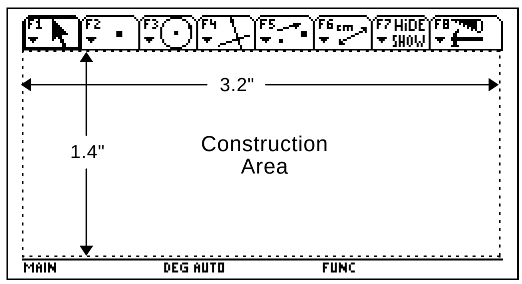
Important: TI-92 Geometry requires 25 Kbytes minimum of free memory

To start a new geometry session:

1. Press [ON] to turn on the TI-92.
2. Press [APPS] and select 8:Geometry.
3. Select 3:New to select a new session.

Note: The variable name can be up to eight characters.

4. Type a variable name in the NEW dialog box and press [ENTER] twice. The Geometry application window opens as shown below.



You construct objects in the active drawing window, which is 240 pixels horizontally and 105 pixels vertically. This is about 3.2 by 1.4 inches (8.1 by 3.6 centimeters).

Selecting a Tool/Command

The toolbar is comprised of eight separate menus (see pages 167 and 168), which are selected when you press [F1] through [F8]. Each menu in the toolbar (except [F8]) contains an icon that graphically illustrates a geometry tool or command. The active menu is framed as shown by the first menu item in the above figure. Press:

[F1] to perform freehand transformations.
[F2] to construct points or linear objects.
[F3] to construct curves and polygons.
[F4] to build Euclidean constructions and create macros.
[F5] to build transformational geometry constructions.
[F6] to perform measurements and calculations.
[F7] to annotate constructions or animate objects.
[F8] to perform file operations and edit functions.

You select tools or commands in a menu by pressing the number that corresponds to the menu item, or by using the cursor pad to move up and down through the menu and pressing [ENTER] to select the highlighted menu item.

For most menu items, once a menu item is selected, it remains in effect until another menu item is selected. The exceptions default to the **Pointer** tool; they are the **Define Macro** tool in the [F4] **Construct** toolbar menu and all [F8] **File** toolbar menu items.

Moving the Cursor

Pressing the cursor pad allows you to move the current active cursor in one of eight directions: up, down, left, right, and the four corresponding diagonals. The cursor moves one pixel for each keypress. When used in combination with the drag key (), the cursor moves one pixel for each keypress and five pixels in repeat mode (cursor pad is held down).

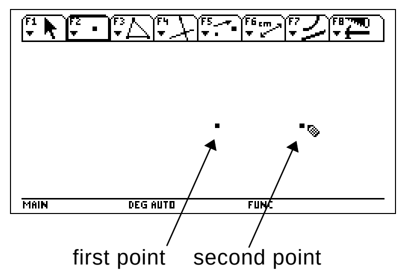
Placing Points

All objects are constructed using one or more points. You create or select points when a tool is active. The order of operation is:

1. Select a construction tool.
2. Create or select the required points that define the object.

A point is created when the **Point** tool is selected and [ENTER] is pressed. You can create points anywhere in the plane when the construction pencil () is active. For example, to construct the two points in the plane below:

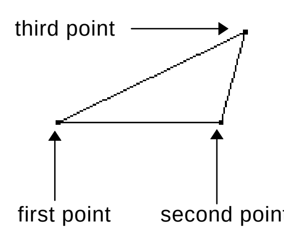
1. Press [F2] and select 1:Point.
2. Move the () cursor to the desired location, and press [ENTER] to create the first point.
3. To create the second point, press the right side of the cursor pad () until the cursor is at the desired location, and then press [ENTER].



Creating a Simple Triangle

All other objects require multiple points to complete their construction. For example, to construct a triangle you create three points as shown below:

1. Press [F3] and select 3:Triangle.
2. Move the () cursor to the desired location, and press [ENTER] to define the first point.
3. Move the cursor to another location, and press [ENTER] to define the second point.
4. Move the cursor to the third location, and press [ENTER] again to complete the triangle.



Selecting Objects

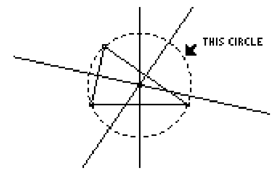
You can select objects by pointing to the object and pressing [ENTER] or by drawing a marquee (dotted) rectangle around the objects. You deselect selected objects by moving the cursor to an unoccupied location in the plane and pressing [ENTER].

Selecting one object.

1. Move the cursor using the **Pointer** tool until the object's name appears, and press [ENTER].

   The selected object appears as a marquee outline.

Select an object.



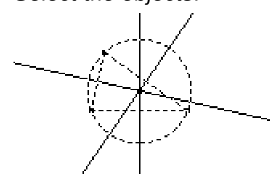
Method #1: Selecting multiple objects.

Hint: *Press [↑] when pressing [ENTER] to select multiple objects.*

1. Move the cursor using the **Pointer** tool until the object's name appears, and then hold [↑] and press [ENTER].
2. Repeat step 1 for other objects that you want to select. (The circle and triangle in this example.)

   All selected objects appear as a marquee outline.

Select the objects.



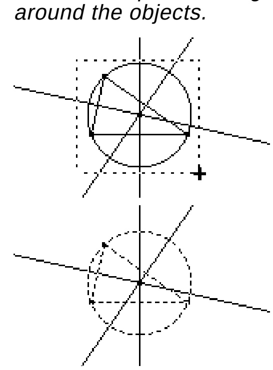
Method #2: Selecting multiple objects.

Note: *The Pointer must begin in an unoccupied location in the plane.*

1. Press and hold [✋] and press the cursor pad to draw a marquee rectangle around the objects that you want to select.
2. Release [✋]. (The circle, triangle, and their points are selected in this example.)

   All selected objects appear as a marquee outline.

Draw a marquee rectangle around the objects.



Deleting Objects

Delete objects by selecting them using the procedures described on the previous page, and then pressing [←] (backspace key) or [F8] and select 7:Delete (delete option in the **File** toolbar menu).

Labeling Points and Objects

You can label points and objects in the following two ways:

- As you create them (see below).
- With the **Label** tool in the **Display** menu (see page 161).

Labeling objects as they are created is intended for quick access and is limited to five alphanumeric characters. Label editing is not available; however, after constructing the object, you can edit a label with the **Label** tool.

1. Press [F3] and select 3:Triangle.

Note: *A point appears with a label "a" beside it.*

2. Move the (✎) cursor to the desired location, press [ENTER] to create the first point, and then press A (for lowercase letters) or [↑] A (for uppercase letters).

Define and label the first point.

Note: *Another point, a segment connecting the two points, and a label "b" appear.*

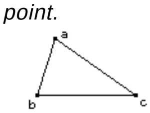
3. Move the cursor and press [ENTER] to create the second point, and then press B.

Define and label the second point.

Note*: The completed triangle appears as well as the label "c" beside the last point created.*

4. Move the cursor and press [ENTER] to create the third point, and then press C.

Define and label the third point.



Dependent and Independent Objects

All objects are created using one or more points. The manner in which you create an object determines whether or not it is dependent or independent of the object. This distinction becomes important with respect to dragging objects.

A point constructed by itself is called a *basic point*. You can identify basic points by selecting the **Pointer** tool and pressing [✋] once. All basic points will flash and can be dragged.

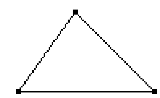
An **independent object** is an object created using only basic points. Independent objects can be moved (dragged) but cannot be modified directly. By moving the basic points used for their construction, you can modify them indirectly.

A **dependent object** is an object constructed using an independent object (or another dependent object). Dependent objects cannot be moved (dragged) or modified directly. You can move or modify them indirectly by moving the basic points or independent objects responsible for their existence.

Dragging Objects

You can move constructed objects that you define with the **Pointer** tool anywhere in the plane. For example, to reposition a constructed object:

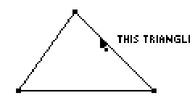
1. Construct a triangle as previously described on page 110.



2. Press F1 and select 1:Pointer.

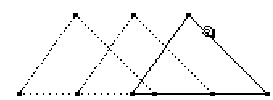
3. Position the (+) cursor until it changes to the (↙) cursor.

   The message “THIS TRIANGLE” appears.



Hint: ***Press*** *2nd* *to lock the cursor in drag mode.*

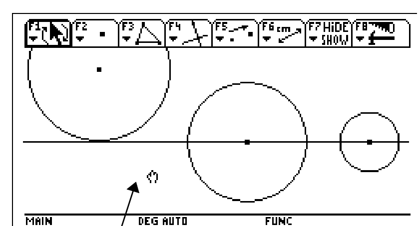
4. Press and hold to use the dragging hand, and then press and hold ▶ to move the triangle to the right.



Positioning a Construction

You can scroll the drawing window to anywhere within the virtual working area (see page 159) by pressing 2nd and the cursor pad at the same time. The default position of the active drawing window is at the center of the virtual working area.

1. Construct several geometric objects as shown.
2. Press F1 and select 1:Pointer.
3. Press and hold 2nd, and then press the cursor pad to scroll all objects in the active drawing window.



open hand scroll cursor

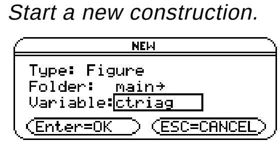
Multi-Step Constructions

You perform multi-step constructions by repeating the construction of individual points described in this section. Lines require one point and a direction, line segments require two points, triangles and arcs require three points, and polygons require n points where n is greater than two.

As an example, to illustrate the basic steps in this section, the procedure below will construct and measure a circle circumscribed around a triangle.

1. Press F8 and select 3:New.

Start a new construction.

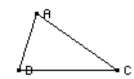


2. Type in a name for the variable to start a new construction, and press ENTER twice.

Multi-Step Constructions (Continued)

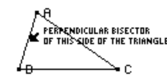
3. Construct and label a triangle. (Perform steps 1 through 4 in “Labeling Points and Objects” described on page 112.)

Construct and label a triangle.

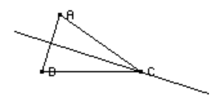


4. Construct perpendicular bisectors for two sides of the triangle by pressing [F4] and selecting 4:Perpendicular Bisector.

Construct the first perpendicular bisector.

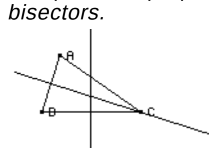


5. Select side AB and press [ENTER].



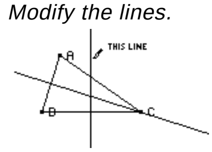
6. Select side BC and press [ENTER].

Complete the perpendicular bisectors.

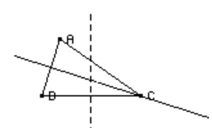


7. Modify the appearance of the perpendicular bisectors from solid to dotted lines by pressing [F7] and selecting 9:Dotted.

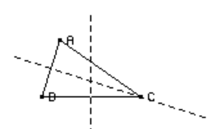
Modify the lines.



8. Select a line and press [ENTER] .



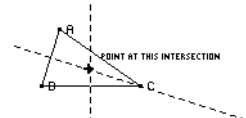
9. Repeat step 8 for the other perpendicular bisector.



10. Press [F3] and select 1:Circle.

11. Define the center point of the circle by moving the cursor near the intersection of the perpendicular bisectors until the message “POINT AT THIS INTERSECTION” appears and pressing [ENTER].

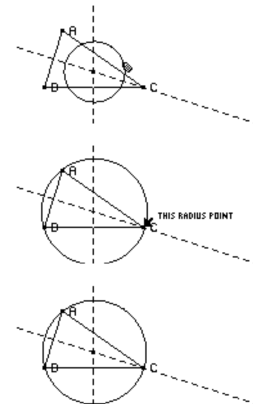
Define the center point.



12. Complete construction of the circle by pressing the cursor pad (→) to expand the circle.

    Press the cursor pad (→ and ↓) until the cursor is near one vertex of the triangle and the message, “THIS RADIUS POINT” appears, and then press [ENTER] to complete the circle.

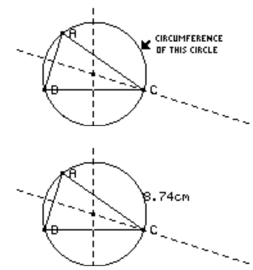
Complete the circle.



13. Measure the circumference of the circle by pressing [F6] and selecting 1:Distance & Length.

14. Position the cursor near the circle until the message “CIRCUMFERENCE OF THIS CIRCLE” appears, and then press [ENTER].

Measure the circumference.



Using Undo

Pressing [F8] and selecting D:Undo, or pressing [♦] Z, will undo the last fully constructed object or operation.

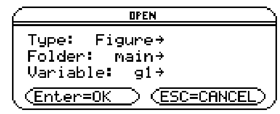
Managing File Operations

The F8 **File** toolbar menu contains file-management commands that allow you to open, close, and save geometry constructions.

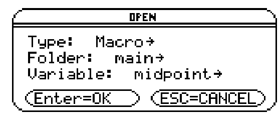
Opening a Construction or Macro

The **Open** command opens a dialog box for opening an existing geometry figure or macro.

1. Press F8 and select 1:Open.
   — or —
   press ♦ O.



2. Select the type of variable that you want to open, Figure or Macro.
3. Press the cursor pad to highlight the variable name that you want to open, and press ENTER twice.



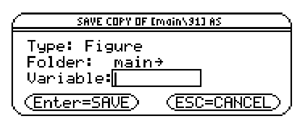
Note: *Pressing* ◯ *and selecting* 2:Macro *after selecting the* Open *command lets you open and use a previously saved macro.*

Saving a Construction as Another Name

To preserve memory, the TI-92 uses an "edit-in-place" method while you are constructing objects. This means the variable that you named when you first opened the geometry session is constantly updated during your constructions.

The **Save Copy As** command opens a dialog box that allows you to save the current construction to a variable name that you specify.

1. Press F8 and select 2:Save Copy As.
   — or —
   Press ♦ S.

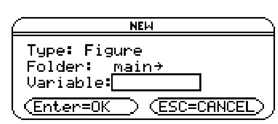


2. Enter a name for your construction in the Variable box, and then press ENTER twice.

Starting a New Construction

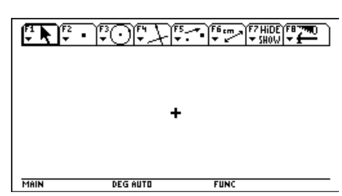
The **New** command opens a new, blank Geometry drawing window for creating a construction or macro.

1. Press F8 and select 3:New.
   — or —
   Press ♦ N.



2. Press ◯ and enter a name, up to eight characters, for your new construction; and then press ENTER twice.

   A blank construction area appears.

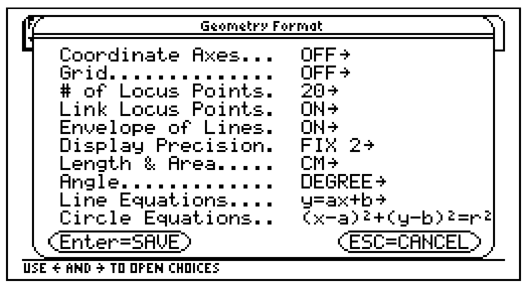


Setting Application Preferences

The [F8] **File** toolbar menu contains the **Format** command that opens a dialog box to specify application preferences, such as angles in degrees or radians, and the display precision of calculations.

Geometry Format Dialog Box Options

The **Format** command opens the Geometry Format dialog box that allows you to specify application preferences. The default formats are shown below.



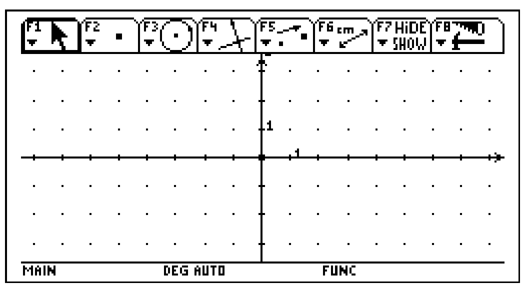
The contents of the Geometry Format dialog box are included in your saved construction files. Consequently, when you open a saved construction, the application returns to the same configuration that was used when you developed the construction.

Defining Application Preferences

1. Press [F8] and select 9:Format.
   — or —
   Press [♦] F.
2. Press ↓ until the cursor is on the same line as the item that you want to change, and then press → to display all options.
3. Select the desired option. (Press the appropriate digit, or highlight the option and press [ENTER].)
4. Press [ENTER] to save your changes and close the dialog box.

Format Options and Descriptions

The table below describes each option in the **Geometry Format** dialog box. (Default settings are in boldface.)

Option	Description
Coordinate Axes **1:OFF** 2:RECTANGULAR 3:POLAR 4:DEFAULT	Displays the rectangular or polar axes. The default distance for the tick marks is approximately 5 mm each. You can change this scale by selecting any tick mark on the horizontal axis and dragging it to a location that approximates the desired scale. All the tick marks in the horizontal and vertical axes will change accordingly. You can change the scale for only the y axis by dragging any tick mark on the vertical axis. The scale of constructed objects is not affected when you change the coordinate scale. You can rotate the axes 360 degrees to redefine the major axes by dragging the x axis in a circular direction. You can also rotate the y axis independently to create an oblique coordinate system. Constructed objects do not change.
Grid **1:OFF** 2:ON	Displays a grid that is composed of a dot at each coordinate. The example below shows the rectangular coordinate axes with grid marks turned ON. The grid does not represent a polar coordinate system. 
# of Locus Points 5 10 15 **20** : 99	Determines how many objects will be constructed along the designated path when you construct a locus. The complete option list is: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 99. You can modify this value dynamically in your construction by selecting the locus and pressing [+] to increase the number of locus points or [−] to decrease the number of locus points.

Option	Description
Link Locus Points 1:OFF **2:ON**	When this option is ON, the points of a locus are linked by way of linear interpolation. When this option is OFF, only the points are displayed.
Envelope of Lines 1:OFF **2:ON**	When this option is ON, only the envelope of the line is displayed when you construct the locus of a line. When this option is OFF, each line of the locus is displayed.
Display Precision 1:FIX 1 **2:FIX 2** ⋮ C:FIX 12	Determines the display precision for calculations and measurements in your constructions. You can modify this value dynamically in a construction by selecting the number and pressing [+] or [−] to increase or decrease the displayed precision of that number.
Length & Area 1:PIXELS 2:MM **3:CM** 4:M	Determines the default units for measurements in your constructions. All values are converted to the selected unit.
Angle **1:DEGREE** 2:RADIAN	Determines the angle units that are displayed and the geometry calculator mode. All angles are converted to the selected unit. This Angle preference is independent from the Angle preference in the Mode dialog box, which applies to other applications.
Line Equations **1:y=ax+b** 2:ax+by+c=0	Determines the format for displayed line equations.
Circle Equations **1:$(x-a)^2+(y-b)^2=r^2$** 2:$x^2+y^2+ax+by+c=0$	Determines the format for displayed circle equations.

Selecting and Moving Objects

The [F1] **Pointer** toolbar menu contains the tools associated with geometry pointer features. These features allow you to select objects and to perform freehand transformations.

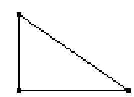
Selecting and Moving Objects Using the Pointer Tool

The **Pointer** tool allows you to select, move, or modify objects. Pressing the cursor pad lets you move the **Pointer** in one of eight directions. The primary functions of the **Pointer** are selection, dragging, and scrolling.

You can return to the **Pointer** at any time by pressing [ESC].

To see how the **Pointer** tool works:

1. Construct a triangle as previously described.
2. Press [F1] and select 1:Pointer.

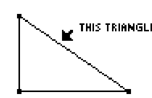


Tip: Press [↑] while selecting an object to select multiple objects.

3. *Selecting:* Select an object by pointing to it and pressing [ENTER] when the cursor message appears for that object.

   Deselect an object by pointing to an unoccupied location and pressing [ENTER].

Point to the object.



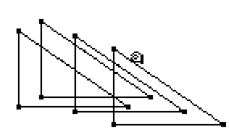
Select the object.

Note: Sometimes multiple objects cannot be moved concurrently. Dependent objects cannot be moved directly. If a selected object cannot be moved directly, the cursor reverts to the cross hair (+) cursor instead of the dragging hand ([✋]) cursor.

4. *Moving:* Move an object by dragging it to a new location. (Only the last object is actually displayed.)

   To show all the points that can be moved, position the cursor to an unoccupied location and press [✋] once. The points that you can drag will flash.

Drag the object.



Deleting Objects from a Construction

The [F8] **File** toolbar menu contains commands that let you delete selected objects or all objects from a construction.

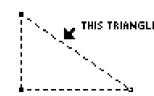
Delete Defined Objects

The **Delete** command allows you to delete selected objects.

1. Select the object that you want to delete. (To select additional objects, press [↑] while selecting each item.)

   Note: In this example, only the triangle and not the points of the vertices are selected.

Select the object.



Hint: Use Undo ([♦] Z) to recover an inadvertent deletion.

2. Press [F8] and select 7:Delete to delete the selected objects.
   — or —
   Press [←].

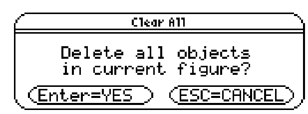
Delete the selected object.

Deleting All Objects

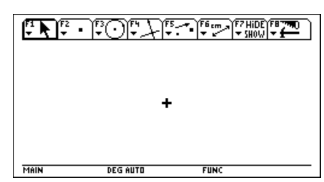
The **Clear All** command deletes every item in the construction and clears the screen.

1. Press [F8] and select 8:Clear All.

   A dialog box is displayed for you to confirm this command.



2. Press [ENTER] to clear the entire construction area, or press [ESC] to cancel.



Creating Points

The [F2] **Points and Lines** toolbar menu contains tools for creating and constructing points in geometry. The three point tools allow you to create points anywhere in the plane, on objects, or at the intersection of two objects.

Creating Points in Free Space and on Objects

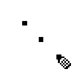
The **Point** tool creates points that can be placed anywhere in the plane, on existing objects, or at the intersection of any two objects.

- If the point created is on an object, it will remain on the object throughout any changes made to the point or to the object.
- If the point is at the intersection of two objects, the point will remain at the intersection when changes are made to the object or objects.
- If the objects are changed such that they no longer intersect, the intersection point disappears. The intersection point reappears when the objects again intersect.

To create points:

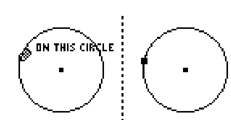
1. Press [F2] and select 1:Point.
2. *Creating points in free space:* Move the cursor to any location in the plane where you want a point, and then press [ENTER] to create the point.

Create points in free space.



3. *Creating points on objects:* Move the cursor to the location on an object where you want a point. When the cursor message appears, press [ENTER] to create the point.

Create points on objects.

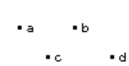


before *after*

Note: ***You can attach a label to the point by entering text (five-character maximum) from the keyboard immediately after creating a point.***

4. *Creating points with labels:* Create a point as defined in step 2 or 3, and then press an appropriate character key to create a label for the point.

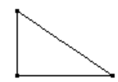
Create points with labels.



Creating a Point on an Object

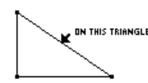
The **Point on Object** tool creates points on any existing object. The point is placed at the location of the cursor. It remains permanently attached to the object—you can drag the point to move it, but it will always remain on the object.

1. Create any object, such as the triangle shown in this example.



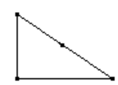
2. Press [F2] and select 2:Point on Object.
3. Move the cursor toward the object until a cursor message appears for the object.

Point to the object.



4. Press [ENTER] to create the point.

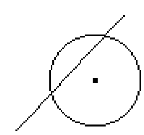
Create the point.



Creating an Intersection Point

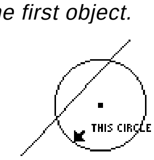
The **Intersection Point** tool creates a point at the intersection (or intersections) of any two defined objects. If the objects are changed so that they no longer intersect, the intersection point disappears. The intersection point reappears when the objects again intersect.

1. Create any two intersecting objects, such as the circle and line shown in this example. (If necessary, see pages 124 and 127.)



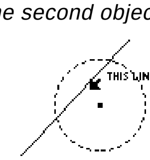
2. Press [F2] and select 3:Intersection Point.
3. Select the first object of two intersecting objects, and then press [ENTER].

Select the first object.

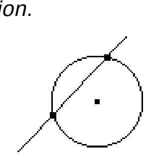


4. Select the second object, and then press [ENTER] to create the intersection point or points.

Select the second object.



Points are created at each intersection.



Creating Lines, Segments, Rays, and Vectors

The F2 **Points and Lines** toolbar menu contains tools for creating and constructing linear objects such as lines, segments, rays, and vectors. The **Construction** menu (F4) contains a tool for creating resultant vectors.

Creating a Line

The **Line** tool creates a line that extends infinitely in both directions through a point at a specified slope. You can control the slope of the line in free space or create the line to go through another point.

1. Press F2 and select 4:Line.
2. Move the (✎) cursor to the desired location, and press ENTER to create the initial point of the line.

Create a point.

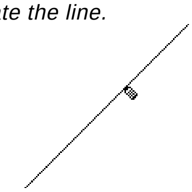
Tip: *To limit the slope to 15-degree increments, press ↑ while pressing the cursor pad.*

Tip: *To label a line, type up to five characters immediately after creating the line or use the Label tool.*

3. Move the cursor away from the point to create the line.

   The line is drawn in the same direction as the keypress. When the line appears, you control the slope of the line by continuing to press the cursor pad.

Create the line.



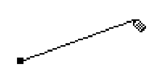
4. Press ENTER to complete the construction.

Creating a Segment

The **Segment** tool creates a line segment between two endpoints.

1. Press F2 and select 5:Segment.
2. Move the (✎) cursor to the desired location, and press ENTER to create the initial endpoint of the segment.

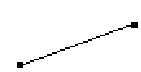
Create the initial point.



Tip: *To limit the slope to 15-degree increments, press ↑ while pressing the cursor pad.*

3. Move the pointer to the location for the final endpoint of the segment.
4. Press ENTER.

Create the final point.



Creating a Ray

The **Ray** tool creates a ray defined by an initial endpoint and extending infinitely in a specified direction. You can control the slope of the ray in free space or create the ray to go through another point.

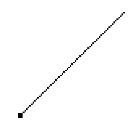
1. Press [F2] and select 6:Ray.
2. Move the (✎) cursor to the desired location, and press [ENTER] to create the endpoint of the ray.

Create a point.

Tip: *To limit the slope to 15-degree increments, press [↑] while pressing the cursor pad.*

3. Position the ray in the desired orientation using the cursor pad.
4. Press [ENTER].

Create the ray.



Creating a Vector

The **Vector** tool creates a vector between two points. A vector is a segment defined by magnitude and direction with a tail (initial endpoint) and head (final endpoint).

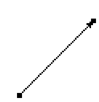
1. Press [F2] and select 7:Vector.
2. Move the (✎) cursor to the desired location, and press [ENTER] to create the tail of the vector.

Create the tail.

Tip: *To limit the slope to 15-degree increments, press [↑] while pressing the cursor pad.*

3. Move the pointer to the location for the head.
4. Press [ENTER].

Create the head.

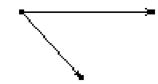


Creating a Resultant Vector

Note: *The selected vectors do not have to share a common endpoint (tail) and may also be previously defined vector sums.*

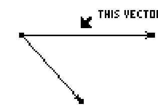
The **Vector Sum** tool in the **Construction** menu creates a resultant vector that is the sum of two selected vectors.

1. Create two vectors as shown in this example.
2. Press [F4] and select 7:Vector Sum.



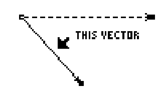
3. Move the pointer and select the first vector.

Select the first vector.



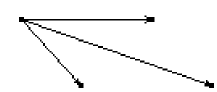
4. Move the pointer and select the second vector.

Select the second vector.



5. Select the initial point for the resultant vector, and then press [ENTER].

Select a tail point for the vector sum.



Creating Circles and Arcs

The [F3] **Curves and Polygons** toolbar menu contains the tools for creating and constructing circles and arcs. The **Construction** menu (F4) also contains a tool for creating circles.

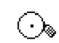
Creating a Circle Using the Circle Tool

The **Circle** tool in the **Curves and Polygons** menu creates a circle defined by a center point and the circle's circumference. The circumference of the circle also can be attached to a point.

You can resize the circle by dragging its circumference. You can move the circle by dragging the center point.

1. Press [F3] and select 1:Circle.
2. Move the (✎) cursor to the desired location and press [ENTER] to create the center point of the circle. Moving the cursor expands the circle.

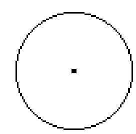
Create the center point.



Tip: To label a circle, type up to five characters immediately after creating the circle or use the Label tool.

3. Continue to move the cursor away from the center point to specify the radius, and then press [ENTER] to create the circle.

Specify the radius and create the circle.



Creating a Circle Using the Compass Tool

The **Compass** tool in the **Construction** menu creates a circle with a radius equal to the length of an existing segment or the distance between two points.

You can change the radius of the circle by dragging the endpoints of the segment that defines the radius. You can move the circle by dragging its center point.

1. Create a segment or two points to define the radius of the circle.

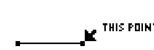
2. Press [F4] and select 8:Compass.
3. Move the pointer to the segment, and press [ENTER].

Select a *segment.*

Note: The center point can actually be anywhere in the plane.

4. Move the pointer to one of the endpoints of the segment, and press [ENTER] to create the circle.

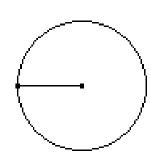
Select a center point.



Note: The first two points determine the radius; the third point becomes the center point of the circle.

5. *(Optional)* Follow the same basic steps to create a compass circle using points. Select three points to perform the construction.

Create the circle.



Creating an Arc

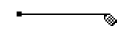
The **Arc** tool creates an arc defined by two endpoints and a curvature point that specifies the curvature of the arc.

1. Press [F3] and select 2:Arc.
2. Move the (✎) cursor to the desired location, and press [ENTER] to create the initial endpoint of the arc.

Create the initial point.

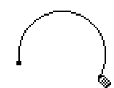
3. Move the pointer away from the initial endpoint.

Move the pointer.



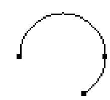
4. Press [ENTER], and then move the cursor to create the curvature point.

Create the curvature point.



5. Move the pointer from the curvature point, and then press [ENTER] to create the final endpoint.

Create the final point.

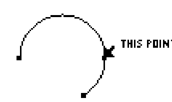


Resizing an Arc

You can resize an arc or change its curvature by dragging any of the three defined points.

1. Move the cursor to one of the points that define the arc.
2. Press and hold [✋] while pressing the cursor pad to resize the arc.

Drag a point to resize the arc.

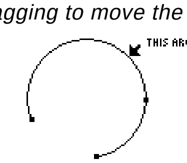


Moving an Arc

You can move the arc by grabbing the arc away from the points that define it and dragging it to a new location.

1. Move the cursor to any location on the arc that is away from the points.
2. Press and hold [✋] while pressing the cursor pad to move the arc.

Select the arc before dragging to move the arc.



Creating Triangles

The [F3] **Curves and Polygons** toolbar menu contains tools for creating and constructing triangles.

Creating a Triangle

The **Triangle** tool creates a triangle defined by three points (vertices).

- **Modifying:** You can modify a triangle by dragging one of its vertices.
- **Moving:** You can move a triangle as an object by grabbing it away from the vertices and moving it to a new location.
- **Moving a point:** You can move a point placed on a triangle along the entire perimeter of the triangle.

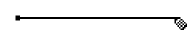
1. Press [F3] and select 3:Triangle.
2. Move the (✎) cursor to the desired location, and press [ENTER] to create the initial vertex.

   Create the first vertex.

3. Move the pointer from the initial vertex, and then press [ENTER] to create the second vertex.

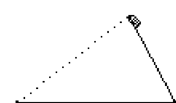
   Create the second vertex.

   Note: *You can limit the slope of its sides to 15-degree increments by pressing [↑] while constructing the triangle.*

4. Move the pointer to the location for the final vertex.

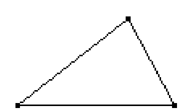
   Locate the final vertex.

   Note: *An outline of the third side is displayed as you move the cursor.*

5. Press [ENTER] to create the final vertex to complete the triangle.

   Create the triangle.

Creating Polygons

The F3 **Curves and Polygons** toolbar menu contains tools for creating and constructing polygons in geometry.

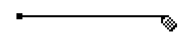
Creating a Polygon

Tip: You can limit the slope of the sides of a polygon to 15-degree increments by pressing ↑ while constructing the polygon.

The **Polygon** tool constructs an n-sided polygon of any shape defined by n points (vertices) where n is a number greater than two.

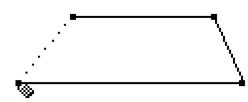
1. Press F3 and select 4:Polygon.
2. Move the (pencil) cursor to the desired location.
3. Press ENTER to create the initial vertex, and then press the cursor pad to create the first side.

Create the initial vertex and the first side.



4. Press ENTER, and then move the pointer to create each of the other vertices.

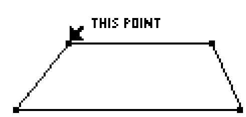
Create additional vertices.



5. To terminate construction of a polygon:
   - Move the pointer to the initial vertex until "THIS POINT" is displayed, and then press ENTER.

     — or —
   - Press ENTER a second time on the last point of a polygon.

Select the original point.



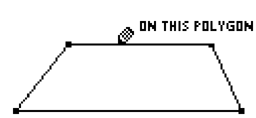
Polygon is complete.

Placing and Moving a Point on a Polygon

You can move a point placed on a polygon along the entire perimeter of the polygon.

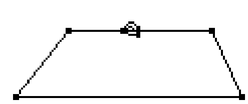
1. Press F2 and select 1:Point.
2. Move the (pencil) cursor to the perimeter of the polygon, and press ENTER.

Create a point.



3. Press and hold [hand] while pressing the cursor pad to move the point along the perimeter of the polygon.

Grab and move the point.



Creating a Regular Polygon

The **Regular Polygon** tool constructs a regular convex or star polygon defined by a center point and n sides.

To begin creating either type polygon, perform steps 1 through 3, and then go to the appropriate step 4 depending on the type of polygon that you want to create.

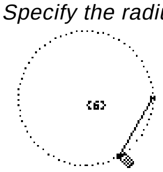
Note: *After creating a regular polygon, you can move a point placed on it along the entire perimeter of the polygon. (See previous page.)*

1. Press [F3] and select 5:Regular Polygon.
2. Move the (✎) cursor to the desired location.
3. Press [ENTER] to create the center point, press the cursor pad to expand the radius, and then press [ENTER].

   The number of sides is displayed at the center point. (Default = 6.)

Create the center point.

Specify the radius.

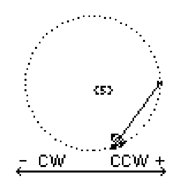


To create a regular *convex* polygon:

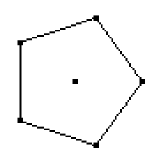
Note: *The polygon can have a minimum of 3 and maximum of 17 sides. If you move beyond 17 sides or 180 degrees from the initial vertex and the center point, the convex polygon becomes a star polygon, and a fraction is displayed at the center point.*

4. Move the pointer *clockwise* from its current position to decrease (−) the number of sides or *counterclockwise* from its current position to increase (+) the number of sides.
5. Press [ENTER] to complete the convex polygon.

Determine # of sides.



Completed polygon.

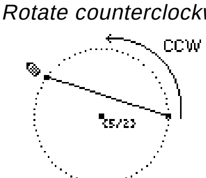


To create a regular *star* polygon:

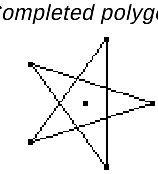
Note: *The minimum value is 5/2 and the maximum value is 17/3. The numerator is the number of sides. The denominator is the number of times the star is crossed.*

6. Move the cursor *counterclockwise* from its current position until a fraction is displayed at the center point. Continue to move the cursor until the desired number of sides is reached.
7. Press [ENTER] to complete the star polygon.

Rotate counterclockwise.



Completed polygon.



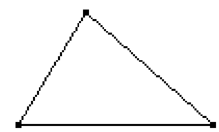
Constructing Perpendicular and Parallel Lines

The [F4] **Construction** toolbar menu contains tools for constructing objects in relation to other objects, such as perpendicular and parallel lines.

Constructing a Perpendicular Line

The **Perpendicular Line** tool creates a line passing through a point and perpendicular to a selected linear object (line, segment, ray, vector, side of a polygon, or axis).

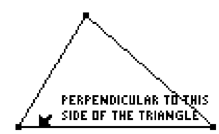
1. Create any object having linear properties such as the triangle shown in this example.



2. Press [F4] and select 1:Perpendicular Line.

3. Move the cursor to a side or object through which you want the perpendicular line to pass, and then press [ENTER].

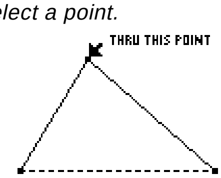
Select a linear object.



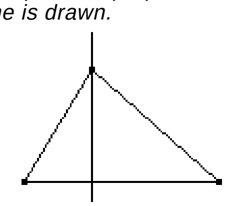
Note: *The order of steps 3 and 4 can be reversed.*

4. Move the cursor to the point through which you want the perpendicular line to pass, and then press [ENTER].

Select a point.



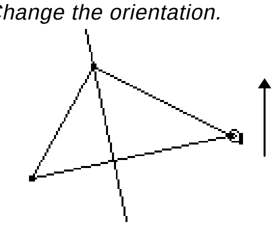
A dependent perpendicular line is drawn.



Note: *You can move the perpendicular line by dragging the point through which the line passes or by changing the orientation of the object to which it is perpendicular.*

5. Drag one of the vertices of the triangle to change its orientation.

Change the orientation.

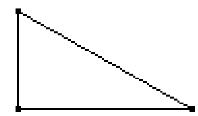


Constructing a Parallel Line

The **Parallel Line** tool creates a line that passes through a point and is parallel to a selected linear object (line, segment, ray, vector, side of a polygon, or axis).

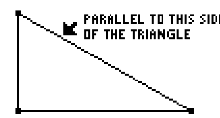
1. Create any object having linear properties such as the triangle shown in this example.

2. Press [F4] and select 2:Parallel Line.



3. Move the pointer to the line, segment, ray, vector, or side of a polygon that will be parallel to the constructed line, and then press [ENTER].

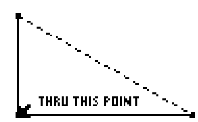
Select a linear object.



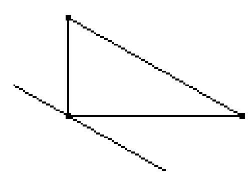
Note: *The order of steps 3 and 4 can be reversed.*

4. Move the pointer to a point through which the parallel line will pass, and then press [ENTER].

Select a point.



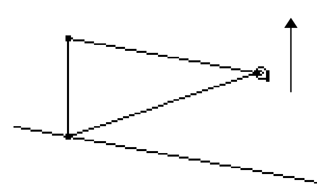
A dependent parallel line is drawn.



Note: *You can move the parallel line by dragging the point through which the line passes or by changing the orientation of the object to which it is parallel.*

5. Drag one of the vertices of the triangle to change its orientation.

Change the orientation.



Constructing Perpendicular and Angle Bisectors

The [F4] **Construction** toolbar menu contains tools for constructing objects in relation to other objects, such as perpendicular and angle bisectors.

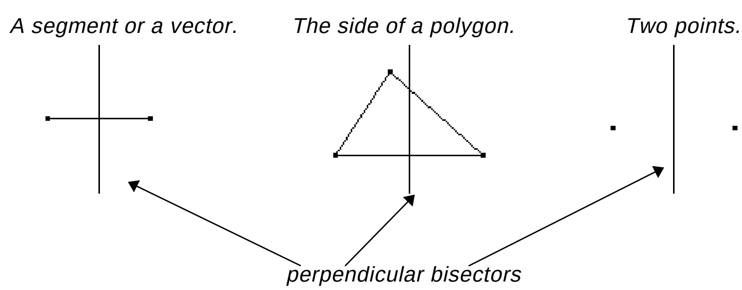
Constructing a Perpendicular Bisector

The **Perpendicular Bisector** tool creates a line that is perpendicular to a segment, a vector, a side of a polygon, or between two points, and passes through the midpoint of the object.

You can move the perpendicular bisector by moving one of the endpoints that define the bisected line segment. A perpendicular bisector cannot be translated directly unless it is constructed between two basic points.

1. Create any object or objects such as those shown below.
2. Press [F4] and select 4:Perpendicular Bisector.
3. Move the pointer to one of the following, and press [ENTER].

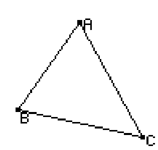
Note: *For two points, select and press [ENTER] for each point.*



Constructing an Angle Bisector

The **Angle Bisector** tool creates a line that bisects an angle identified by three selected or created points. The second point defines the *vertex* of the angle through which the line passes.

1. Create a labeled triangle such as the one shown in this example.
2. Press [F4] and select 5:Angle Bisector.

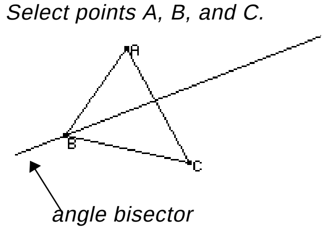


Tip: *You can change the angle bisector by dragging any of the three points that define the angle.*

3. Select three points to define the angle that you want to be bisect. (The second point that you select is the vertex of the angle.)

   The angle bisector is created when you select the third vertex.

Select points A, B, and C.



Creating Midpoints

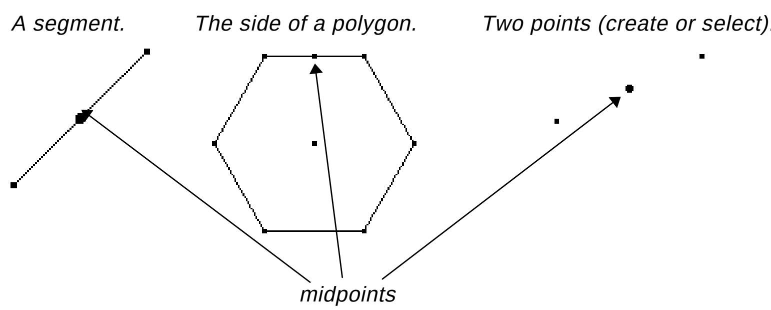
The [F4] **Construction** toolbar menu contains a tool for constructing the midpoint of a segment.

Creating a Midpoint

The **Midpoint** tool creates a point at the midpoint of a segment, a vector, the side of a polygon, or between two points.

1. Create any object or objects such as those shown below.
2. Press [F4] and select 3:Midpoint.
3. Move the pointer to one of the following, and press [ENTER].

Note: *For two points, select and press [ENTER] for each point.*



Transferring Measurements

The F4 **Construction** toolbar menu contains a tool for transferring measurements between objects.

About Transferring Measurements

The **Measurement Transfer** tool creates:

- A point on a ray or vector from the initial point of a line, segment, polygon, or axis.
- A point at a proportional distance from another point.
- A point on a circle that is at an equivalent arc length from another point on the circle.

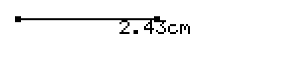
The point created by the measurement transfer is dynamically updated. The magnitude of the measurement that is transferred defaults to the specified unit of length.

Note: See “Measuring Distance and Length of an Object” on page 149 and “Creating and Editing Numerical Values” on page 162 to create the numerical values shown in the examples in this section.

Creating a Measurement Transfer Point on a Ray

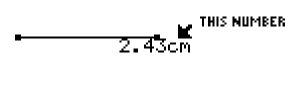
Perform the following steps to transfer the measurement of a segment to a ray.

1. Construct and measure a segment, and construct a ray as shown in this example.



2. Press F4 and select 9:Measurement Transfer.
3. Point to any measurement or numerical value, and press ENTER to select the value.

Select a numerical value.

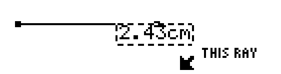


Note: If you select a point, a dotted line appears. Position the dotted line as you want it, and then press ENTER *to set the position.*

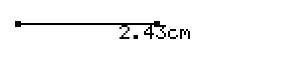
4. Select a ray, vector, polygon, point, or axis; and press ENTER to transfer the measurement to the object.

   A point is created that is an equivalent distance from the endpoint of the ray.

Select a ray.



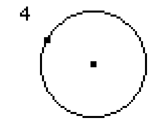
Transfer the measurement.



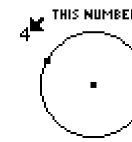
Creating a Measurement Transfer Point on a Circle

Perform the following steps to create a point on a circle at a proportional arc length away from a selected point.

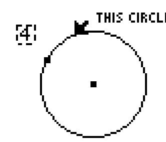
1. Create a circle with a point on it, and then create a numerical value as shown in this example.



2. Press [F4] and select 9:Measurement Transfer.
3. Move the cursor and press [ENTER] to select the numerical value.

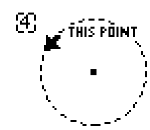


4. Move the cursor and press [ENTER] to select the circle

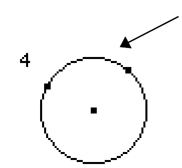


__Note:__ The direction of the distance or arc length is counterclockwise for positive values and clockwise for negative values. The direction is determined by the sign of the selected numerical value.

5. Move the cursor to the existing point on the circle.



6. Press [ENTER] to create a point on the circle that is a proportional arc length away from the initial point.



Creating a Locus

The [F4] **Construction** toolbar menu contains the Locus tool, which generates a set of points while a point moves along a path.

Creating a Locus

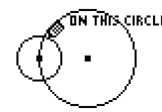
The **Locus** tool creates a set of objects defined by the movement of a point along a path. A path is any defined object on which a point can be placed.

Note: ***The number of points calculated in the construction of the locus is defined in the*** Geometry Format *dialog box.*

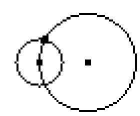
1. Construct two circles as shown.

   The center point and circumference of the small circle *must be attached* to the circumference of the large circle.

Construct and attach two circles.



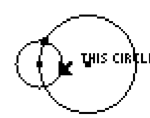
This point indicates that the circles are attached.



2. Press [F4] and select A:Locus.
3. Select the small circle as the object for which to construct the locus.

Note: *The locus is dynamically recalculated when you modify the objects that define the locus.*

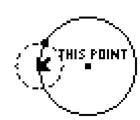
Select the object.



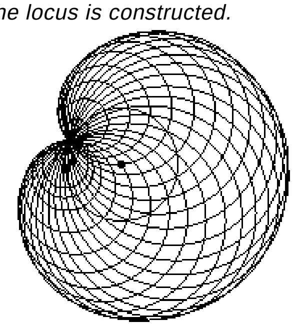
4. Select the center point of the small circle as the point that lies on a path.

   When you select a point on a path (object), the locus is constructed in its entirety and is considered a defined object.

Select a point on the path.



The locus is constructed.



Redefining Point Definitions

The F4 **Construction** toolbar menu contains the **Redefine** tool, which redefines the definition of points.

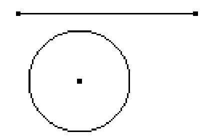
Redefining the Definition of a Point

__Note:__ The new definition cannot be a circular reference. A circular reference occurs when a point that defines an object is redefined to be on that object. For example, defining the center point of a circle to be a point on the circle is not allowed.

The **Redefine Point** tool modifies the current definition of a point.

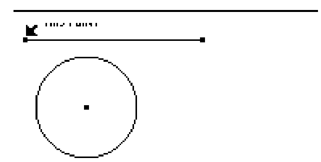
To redefine a point in the following construction:

1. Create a segment and circle as shown in this example.



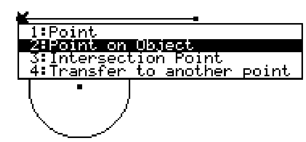
2. Press F4 and select B:Redefine Point.

3. Move the pointer to a point, and then press ENTER.



Select the endpoint of the segment.

A pop-up menu opens to let you select a point redefinition option.

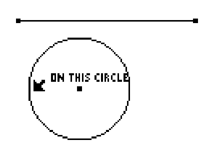


- Point – Redefines the point as a basic point at the same location.
- Point on Object – Redefines the point to be on an object.
- Intersection Point – Redefines the point to be at the intersection of two objects.
- Transfer to another point – Transfers the point to another existing point.

4. Select 2:Point on Object.

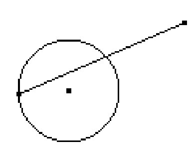
Select a point on the circle.

5. Move the pointer to an object compatible with the selected option, and press ENTER.



The point is redefined.

The segment is attached to the circle.



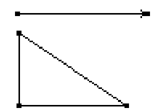
Translating Objects

The [F5] **Transformations** toolbar menu contains a tool that is used to translate (copy and move) geometry objects.

Translating an Object

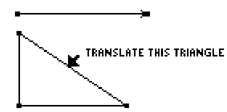
The **Translation** tool creates the image of an object translated by a specified, previously defined vector.

1. Create a vector and triangle as shown in this example.
2. Press [F5] and select 1:Translation.



3. Select the object to translate.

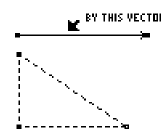
Select the object to translate.



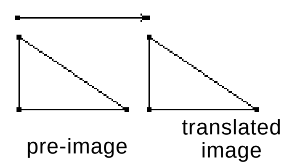
4. Select the vector that defines the translation direction and distance.

   The image of the "pre-image" is translated to the selected location. The pre-image remains in its original location.

Select the translation vector.



The image is translated.



Modifying a Translation

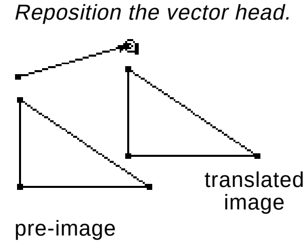
Note: *Because it is a dependent object, you cannot change the translated image directly.*

You can modify a translated image by dragging the vector head to a new location.

- Grab and drag the vector head.
  —or—
- Grab and drag the vector tail to change the magnitude of the translation.

The translated image changes according to the changes made to the vector.

Reposition the vector head.



Rotating and Dilating Objects

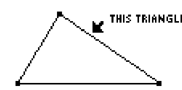
The [F1] **Pointer** toolbar menu contains tools to rotate and dilate objects by freehand manipulation. The [F5] **Transformations** toolbar menu contains tools for rotating and dilating objects using specific values to create translated images.

Rotating Objects by Freehand

The **Rotate** tool in the **Pointer** menu rotates an object about its geometric center or a defined point.

To rotate an object about its geometric center:

1. Create a triangle as shown in this example.

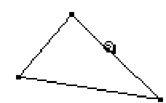


2. Press [F1] and select 2:Rotate.

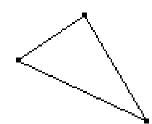
__Hint:__ Press and hold [hand key] while pressing the cursor pad.

3. Point to the object (not a point) and drag in the direction that you want to rotate the object.

Drag the object around its geometric center

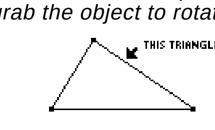


Complete the rotation.



To rotate an object about a defined point:

Select the rotation point and grab the object to rotate.

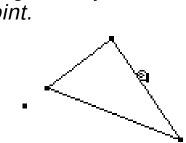


1. Create a triangle and a point as shown in this example.
2. Press [F1] and select 2:Rotate.

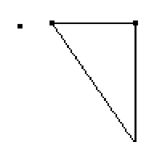
__Note:__ Move the cursor to an unoccupied location and press [ENTER] to deselect the rotation point.

3. Select the rotation point. The point will blink on and off.
4. Point to the object and drag in the direction that you want to rotate the object.

Drag the object around the point.



Complete the rotation.

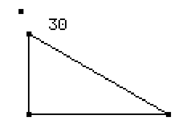


Rotating Objects by a Specified Angular Value

The **Rotation** tool in the [F5] **Transformations** toolbar menu translates and rotates an object by a specified angular value with respect to a point.

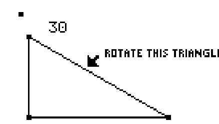
Note: See “Measuring Distance and Length of an Object” on page 149 and “Creating and Editing Numerical Values” on page 162 to create the numerical values shown in the examples below.

1. Create a triangle, a point, and a numerical value as shown in this example.



2. Press [F5] and select 2:Rotation.
3. Select the object to rotate.

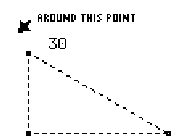
Select the object to rotate.



Note: The angular value may be any measurement or numerical value regardless of unit assignment. Rotation assumes that the value is in degrees or radians, and is consistent with the Angle *setting in the* Geometry Format *dialog box. Positive values = CCW rotation. Negative values = CW rotation.*

4. Select the point of rotation.

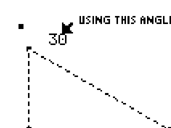
Select the rotation point.



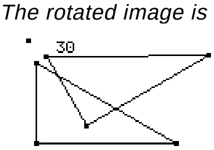
5. Select the angular value of rotation.

   The rotated image is created. The original object is still displayed at its original location.

Select the angular value.



The rotated image is created.



Modifying a Rotation

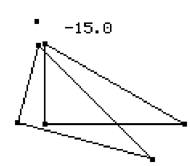
You can modify a rotated image by changing the number that defines the angle of rotation, moving the rotation point, or modifying the original object.

Note: Because the rotated image is a dependent object, you cannot change it directly.

1. Select the number, press [F7] and select 6:Numerical Edit.
2. Change the number to a different value and press [ENTER].

   The rotated image moves according to the numerical value that defines the rotation.

The rotated image is modified.

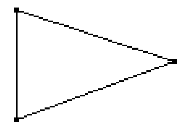


Dilating Objects by Freehand

The **Dilate** tool in the **Pointer** menu expands or contracts an object about its geometric center or a defined point.

To dilate an object about its geometric center:

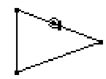
1. Create a triangle as shown in this example.
2. Press [F1] and select 3:Dilate.



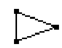
Tip: Press and hold [✋] while pressing the cursor pad.

3. Point to the object (not a point) and drag to dilate the object about its geometric center.
4. Drag the object away from its center to expand or toward its center to contract.

Drag the object.



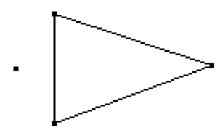
Complete the dilation.



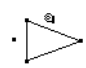
To dilate an object about a defined point:

1. Create a triangle and a point as shown in this example.
2. Press [F1] and select 3:Dilate.
3. Select the dilation point. The point will blink on and off.
4. Point to the object and drag to dilate the object with respect to the dilation point.
5. Drag the object away from its center to expand or toward its center to contract.

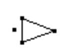
Select a dilation point.



Drag the object.



Complete the dilation.



***Note:** Dragging an object through the dilation point causes a negative dilation. The cursor must travel through the dilation point.*

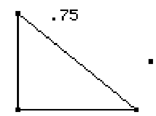
Dilating Objects by a Specified Factor

The **Dilation** tool in the **Transformations** menu translates and dilates an object by a specified factor with respect to a specified point.

Note: See “Creating and Editing Numerical Values” on page 162 to create the numerical values shown in the examples below.

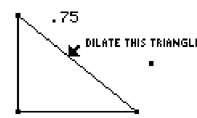
***Note:** Negative numerical values result in a negative dilation.*

1. Create a triangle, a point, and a numerical value as shown in this example.



2. Press [F5] and select 3:Dilation.
3. Select the object to dilate.

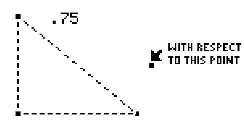
Select the object to dilate.



***Note:** The factor can be any measurement or numerical value regardless of unit assignment. Dilation assumes that the selected value is without a defined unit.*

4. Select the point of dilation.

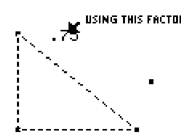
Select the dilation point.



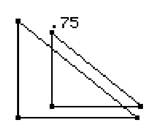
5. Select the factor of dilation.

   The dilated image is created. The original object is still displayed at its original location.

Select the dilation factor.



The dilated image is created.



Modifying a Dilation

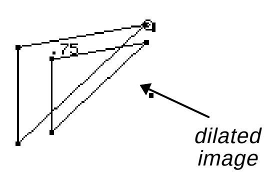
You can modify a dilated image by changing the number that defines the factor of dilation, moving the dilation point, or modifying the original object.

***Note:** Because it is a dependent object, you cannot change the dilated image directly.*

1. Grab and drag a vertex of the original object.

   The dilated image moves according to the changes made to the original object.

The dilated image is modified.

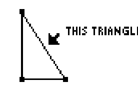


Rotating and Dilating Objects by Freehand

The **Rotate & Dilate** tool in the **Pointer** menu rotates and dilates a selected object about its geometric center or a defined point.

To rotate and dilate an object about its geometric center:

1. Create a triangle as shown in this example.

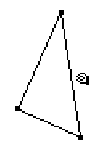


2. Press [F1] and select 4:Rotate & Dilate.

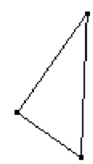
Tip: Drag the object away from its center to expand, or toward its center to contract. Drag the object in a circular motion to rotate.

3. Point to the object, and drag to rotate and dilate the object.

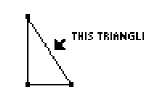
Drag the object in a circular or linear path.



Complete the rotation and dilation.



To rotate and dilate an object about a defined point:

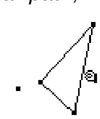


1. Create a triangle and a point as shown in this example.
2. Press [F1] and select 4:Rotate & Dilate.
3. Select the point of rotation and dilation. The point will blink on and off.

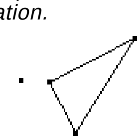
Tip: Drag the object away from its defined point to expand and rotate or toward its center to contract and rotate.

4. Point to the object, and drag to rotate and dilate the object with respect to the point.

Drag object in a circular or linear path,



Complete the rotation and dilation.



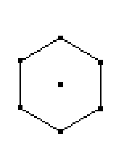
Creating Reflections and Inverse Objects

The [F5] **Transformations** toolbar menu contains the tools associated with transformational geometry for creating reflections and inverse objects.

Creating a Reflection

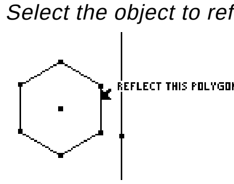
The **Reflection** tool creates a mirror image of an object reflected across a line, segment, ray, vector, axis, or side of a polygon.

1. Create a polygon and a line as shown in this example.
2. Press [F5] and select 4:Reflection.



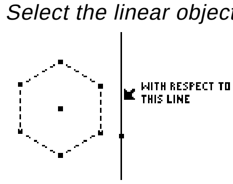
3. Select the object to reflect.

Select the object to reflect.

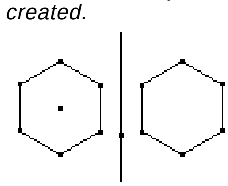


4. Select the line, segment, ray, vector, axis, or side of a polygon to reflect the object across.

Select the linear object.



The reflected object is created.



Modifying a Reflection

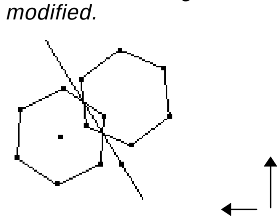
Note: *Because the reflected image is a dependent object, you cannot change it directly.*

You can modify a reflected image by changing the original object or by modifying the line of reflection.

1. Select, reposition, and rotate the line.

   The reflected image moves according to the changes made to the line.

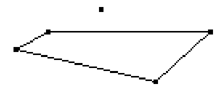
The reflected image is modified.



Creating a Symmetrical Image

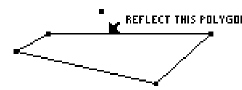
The **Symmetry** tool creates the image of an object that is rotated 180 degrees around a point.

1. Create a polygon and a point as shown in this example.



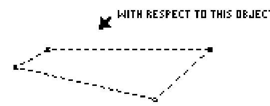
2. Press [F5] and select 5:Symmetry.
3. Select the object to rotate 180 degrees.

Select the object to rotate.

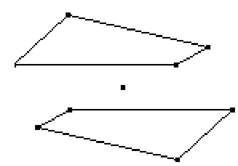


4. Select the point of symmetry.

Select a point.



The symmetrical image is created.



Modifying a Symmetrical Image

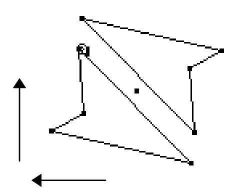
__Note:__ Because a symmetrical image is a dependent object, you cannot change it directly.

You can modify a symmetrical image by changing the original object or by moving the point of symmetry.

1. Grab and drag a vertex of the original object. (Upper right vertex of the original object shown in step 1.)

   The symmetrical image is modified according to the changes made to the original object.

The symmetrical image is modified.



Creating an Inverse Point

The **Inverse** tool constructs an inverse point with respect to a circle and a point, according to the equation OM • OM’ = r^2

where:

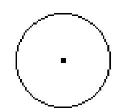
M and M’ are points that lie on a ray with endpoint O.

O	=	center of circle.
M	=	selected point.
M’	=	inverse point.
r	=	radius of selected circle.

As the selected point approaches the center point, the inverse point approaches a point at infinity. If M is defined to be on a line, the locus of M’ constructs a circle that passes through the center of the original circle.

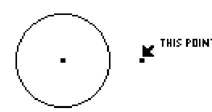
If the original point lies in the interior of the circle, the inverse point is constructed in the exterior, and vice versa. The inverse point lies on a ray with the center point as the endpoint.

1. Create a circle and a point as shown in this example.
2. Press [F5] and select 6:Inverse.



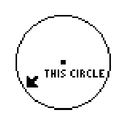
3. Select the point as the original point.

Select a point.

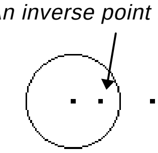


4. Select the circle.

Select a circle.



An inverse point is created.



Modifying an Inverse Point

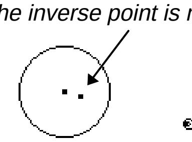
Note: *Because an inverse point is a dependent point, you cannot change it directly.*

You can modify an inverse point by dragging the point or by modifying the circle that defines it.

1. Grab and drag the original point.

   The inverse point inside the circle moves according to the changed position of the original point.

The inverse point is modified.



Measuring Objects

The [F6] **Measurement** toolbar menu contains the tools associated with measurement features in geometry. These features allow you to perform different measurements and calculations on your constructions.

About Measuring Objects

For all measurements described in this section:

- You can add a descriptive comment to a measurement by entering text immediately after creating the measurement, or by using the **Comment** tool in the [F7] **Display** toolbar menu.
- You can change the location of a measurement result by dragging it to a different location.

Measuring Distance and Length of an Object

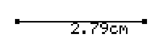
The **Distance & Length** tool measures length, arc length, perimeter, circumference, radius, or the distance between two points.

1. Create a segment as shown in this example.

2. Press [F6] and select 1:Distance & Length.
3. To measure:
   - Length, perimeter, or circumference – Select a segment, arc, polygon, or circle.
   - Distance – Select two points.
   - Radius – Select the center point, and then the circumference of the circle.

Select an object.

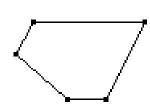
The result is displayed.



Measuring the Area of a Closed Object

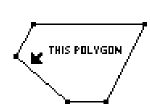
The **Area** tool measures the area of a selected polygon or circle.

1. Create a polygon or circle.

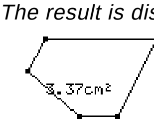


2. Press [F6] and select 2:Area.
3. Select the polygon or circle whose area you want to measure, and then press [ENTER].

Select an object.



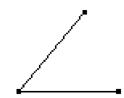
The result is displayed.



Measuring an Angle

The **Angle** tool measures an angle defined by three selected points or an angle mark. The second point selected is the vertex of the angle. The result is displayed in degrees or radians consistent with the Angle option in the **Geometry Format** dialog box.

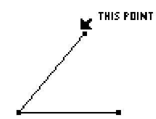
1. Create two segments that have a common point, or any polygon.
2. Press [F6] and select 3:Angle.



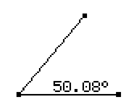
Hint: ***Hint:** If an angle mark is displayed on the angle, select the angle mark to measure the angle.*

3. Select three points to specify the angle. The second point that you select is the vertex.

Select three points.



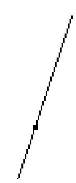
The result is displayed.



Measuring the Slope of a Linear Object

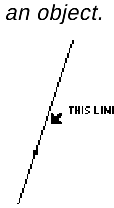
The **Slope** tool measures the slope of a selected segment, ray, vector, or line.

1. Create any linear object.
2. Press [F6] and select 4:Slope.



3. Select the segment, ray, vector, or line whose slope you want to measure.

Select an object.



The result is displayed.

Determining Equations and Coordinates

The [F6] **Measurement** toolbar menu contains the **Equation & Coordinates** tool that generates and displays equations and coordinates of lines, circles, and points.

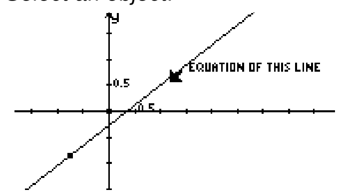
About the Equation & Coordinates Tool

The **Equation & Coordinates** tool displays the equation of a line, circle, or coordinates of a point with respect to a default coordinate system. The equation or coordinates are updated when the object is modified or moved.

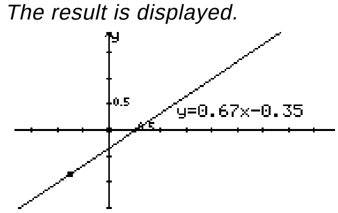
Checking the Equation and Coordinates of a Point or Line

1. (*Optional*) To display the x and y axes, press [F8] and select 9:Format; and then select 2:RECTANGULAR from the Coordinate Axes option.
2. Press [F6] and select 5:Equation & Coordinates.
3. Select the point or line whose coordinates or equation you want to find.

Select an object.



The result is displayed.

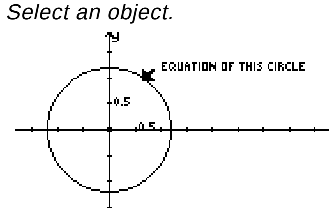


Checking the Equation and Coordinates of a Circle

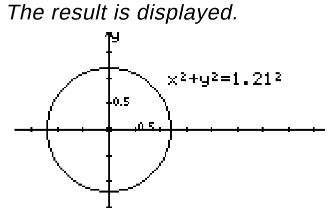
The **Equation & Coordinates** tool displays the equation of a circle with respect to a default coordinate system. The equation or coordinates are updated when the object is modified or moved.

1. (*Optional*) To display the x and y axes, press [F8] and select 9:Format; and then select 2:RECTANGULAR from the Coordinate Axes option.
2. Press [F6] and select 5:Equation & Coordinates.
3. Select the circle whose equation you want to find.
4. Select the center point of the circle to find the coordinates of the point.

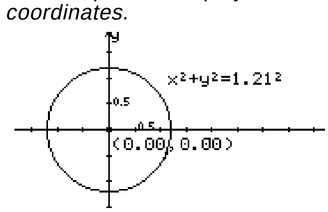
Select an object.



The result is displayed.



Select a point to display its coordinates.



The F6 **Measurement** toolbar menu contains the **Calculate** tool that performs measurement calculations on your constructions.

Performing Calculations on Constructed Objects

The **Calculate** tool opens a calculation entry line near the bottom of the screen. The entry line is the interface for entering mathematical expressions involving geometric objects. This tool lets you do the following:

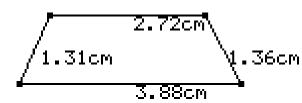
- Perform calculations on constructed objects.
- Access various features of the TI-92 calculator.

__Note__: The result of a calculation must be a single floating-point number to be displayed.

Follow the steps below to perform calculations using measurements, numerical values, calculation results, and numerical inputs from the keyboard.

1. Construct a polygon, and then measure the distance between each point (see page 149).

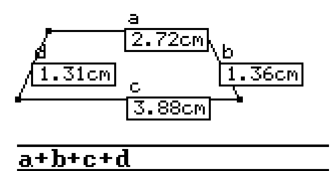
Construct and measure an object.



2. To calculate the perimeter, press F6 and select 6:Calculate.
3. Press ⓒ to select the first measurement, and then press ENTER.
4. Press +.
5. Press ⓒ as necessary to select the second, third, and fourth measurements, and then press ENTER each time. (Press + before each variable.)

__Note:__ The characters assigned to each value are copied from the drawing window and indicate that the value is a variable. The characters are an internal variable representation and do not affect other system-level variables with the same name. You can have up to 10 variables per calculation.

Assign variables.

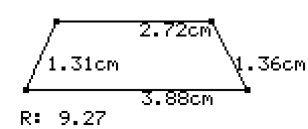


6. With the cursor in the entry line, press ENTER.

   The sum is calculated and displayed after R:.

__Note:__ You can recall a calculation by selecting the result and pressing 2nd ENTER.

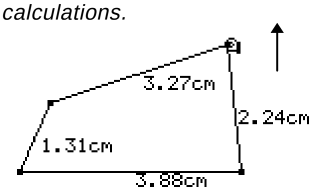
Perform the calculation.



7. To see interactive calculations, grab a vertex of the polygon and drag it to another location.

   Observe the dynamic changes in the result (R:) as the object is changed.

Observe interactive calculations.



The F6 **Measurement** toolbar menu contains the **Collect Data** tool that lets you define and store data from your constructions into lists for later review in the Data/Matrix Editor.

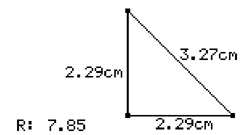
Collecting Data about an Object into a Table

The **Collect Data** tool collects selected measurements, calculations, and numerical values into the variable sysData. You can collect up to 10 data measurements simultaneously.

1. Construct an object, and then measure its dimensions.

   For example, measure the sides of a triangle and calculate its perimeter.

Construct and measure.

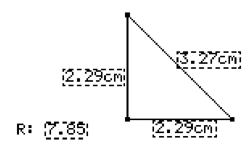


2. Press F6 and select 7:Collect Data, and then select 2:Define Entry.

3. Select each measurement and calculated value to define the data to collect.

   The data will appear in the Data/Matrix Editor in the order in which the data was selected.

Define the data to collect.

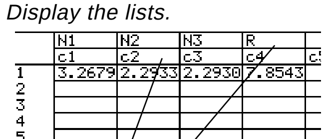


Tip: Press ♦ H *to place the collected data as a vector in the history area of the Home screen for later review.*

4. Press F6 and select 7:Collect Data, and then select 1:Store Data.
   — or —
   Press ♦ D.

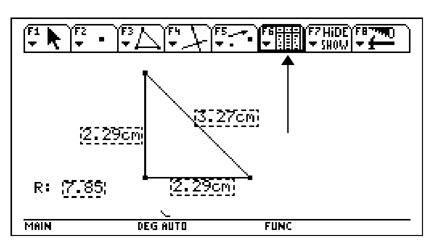
5. Press APPS and select 6:Data/Matrix Editor, and then open the variable sysData to display the lists of collected data.

Display the lists.



(**Note:** Labels are also copied to the table, if available.)

Note: You can automatically collect defined data entries if the Store Data icon appears in the toolbar while you are animating your construction. (See "Putting Objects in Motion" on page 156).



Checking Properties of Objects

The [F6] **Measurement** toolbar menu contains the **Check Property** tool, which allows you to verify specific properties related to a construction.

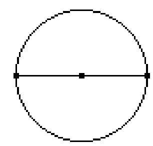
Editing Check Property Text

For all properties described in this section, you can edit the **Check Property** text using the **Comment** tool (see page 162) to customize the result.

Determining If Points Are Collinear

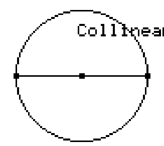
The **Collinear** tool verifies whether or not three selected points lie on the same line.

1. Construct a circle and a segment such that the segment passes through the center point and its endpoints are attached to the circle.



2. Press [F6] and select 8:Check Property, and then select 1:Collinear.
3. Point to each endpoint of the segment and the center point of the circle, pressing [ENTER] each time.
4. Press [ENTER] to display the property.

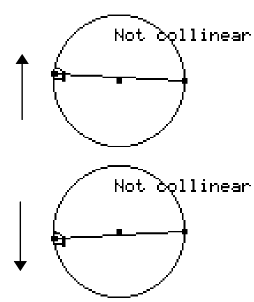
Select three points.



Tip: *Position the text box to the desired location before pressing [ENTER] to display the result.*

Note: *The displayed property changes when the third point (center point) is no longer collinear with the endpoints of the segment.*

5. Drag one of the endpoints of the segment a few pixels up and a few pixels down.



Determining If Lines Are Parallel

The **Parallel** tool verifies whether or not two lines, segments, rays, vectors, axes, or sides of a polygon are parallel.

1. Construct two segments as shown.

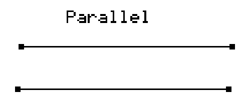
2. Press [F6] and select 8:Check Property, and then select 2:Parallel.

3. Point to the first segment and press [ENTER]. Then point to the second segment and press [ENTER].

Select the objects.

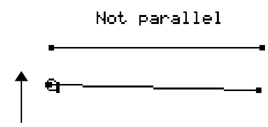
Tip: Position the text box to the desired location before pressing [ENTER] to display the result.

4. Press [ENTER] to display the property of the two segments.



Note: The displayed property changes when the two segments are no longer parallel.

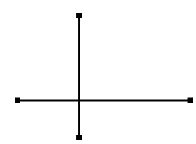
5. Drag the endpoint of one of the segments a few pixels up or down.



Determining If Lines Are Perpendicular

The **Perpendicular** tool verifies whether or not two lines, segments, rays, vectors, axes, or sides of a polygon are perpendicular.

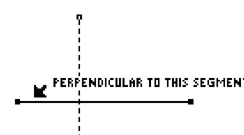
1. Construct two segments as shown.



2. Press [F6] and select 8:Check Property, and then select 3:Perpendicular.

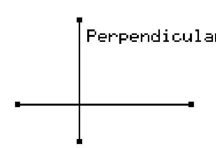
3. Point to each segment, pressing [ENTER] each time.

Select the objects.



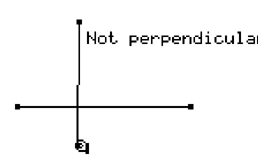
Tip: Position the text box to the desired location before pressing [ENTER] to display the result.

4. Press [ENTER] to display the property.



Note: The displayed property changes when the two segments are no longer perpendicular.

5. Drag the endpoint of one of the segments so that they are no longer perpendicular.



Putting Objects in Motion

The [F7] **Display** toolbar menu contains the tools that let you animate and trace objects.

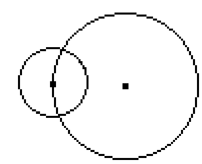
Animating Independent Objects

The **Animation** tool automatically moves an independent object along a specified path.

- If the **Pointer** tool is visible in the toolbar and the object does not lie on a defined path, the animated direction is 180 degrees from the spring. Otherwise, the object is animated along its defined path.
- If the **Rotate**, **Dilate**, or **Rotate & Dilate** tool is visible in the **Pointer** toolbox and the object can be transformed, the animation will be relative to the visible **Pointer** tool. For example, if the **Rotate** tool is visible, the object is rotated automatically.
- Pressing [ENTER] pauses the animation; pressing [ENTER] again resumes the animation. Pressing [ESC] or [ON] cancels the animation.

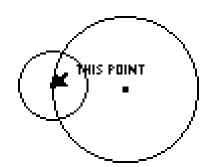
To animate an object:

1. Construct two circles as shown in this example.
2. Press [F7] and select 3:Animation.



3. Select the point of the object to animate.

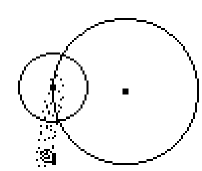
Select the point.



4. Drag the animation spring in the opposite direction of the intended animation, and then release [hand].
   —or—
   Press and release [hand] twice quickly.

   The small circle moves around the circumference of the large circle.

Drag the animation spring.



Note: *The farther away the spring is pulled, the faster the object is animated. You can also increase or decrease the animation while the object is in motion by pressing [+] or [−], respectively.*

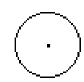
Tracing the Path of an Object

The **Trace On/Off** tool traces the path of an object as it is moved.

- You can trace objects manually by dragging them, or automatically by using the **Animate** tool.
- You can select multiple objects for tracing, or deselect all trace objects by pressing [↑]+[ENTER] with the cursor in an unoccupied location in the plane.
- You can clear the results of a trace by pressing [CLEAR].

To trace the path of a moving object:

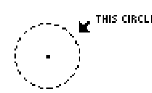
1. Create a circle as shown in this example.
2. Press [F7] and select 2:Trace On / Off.



3. Select the objects to trace.

   Selected objects are displayed in a marquee outline.

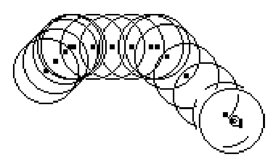
Select any object or objects.



Note: The Trace On / Off *tool works as a toggle function on an object.*

4. To disable the trace on an object, press [F7] and select 2:Trace On / Off. Then select the object displayed in marquee outline.

Move the object to show the trace.



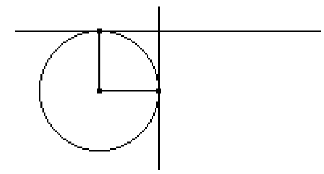
Controlling How Objects Are Displayed

The [F7] **Display** toolbar menu contains tools for controlling the display features of objects. The [F8] **File** toolbar menu contains several tools that determine how objects are viewed.

Hiding and Showing Objects

The **Hide/Show** tool in the **Display** toolbar menu hides selected visible objects and shows selected hidden objects. Hidden objects do not alter their geometric role in the construction.

1. Construct several objects such as those shown in this example.
2. Press [F7] and select 1:Hide / Show.

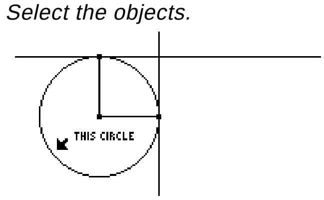


Note: Hidden objects are shown in dotted outline when the Hide / Show tool is active.

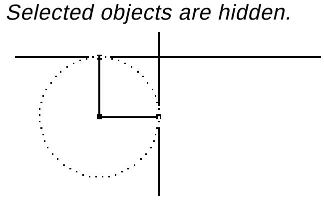
Tip: Hiding objects improves performance because fewer objects must be drawn.

3. Point to each object that you want to hide, and press [ENTER].

Select the objects.



Selected objects are hidden.

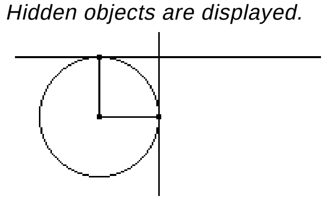


Note: *When the* Hide / Show *tool is active, pressing* [↑] *and* [ENTER] *at the same time in free space makes all hidden objects visible.*

4. Select a hidden object to make it visible again.

   The **Hide / Show** tool works as a toggle function on an object.

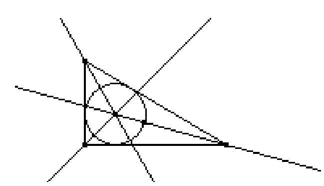
Hidden objects are displayed.



Changing the Line Thickness of Objects

The **Thick** tool in the **Display** toolbar menu changes the outline thickness of an object between Normal (one pixel) and Thick (three pixels) outlines.

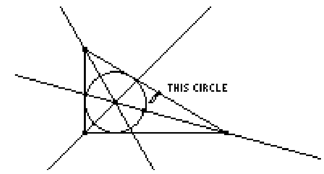
1. Construct several objects such as those shown in this example.
2. Press [F7] and select 8:Thick.



Tip: *Change the thickness of a point to set it apart from other points.*

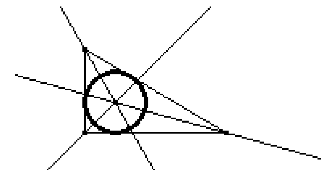
3. Point to the object to be outlined in thick outline.

Select the object.



Note*: This option works as a toggle. Reselecting the object changes the outline back to normal.*

4. Press ENTER to change the outline as shown, and then press ENTER again to change it back to normal.

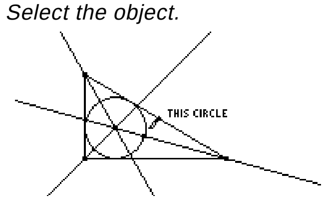


Changing the Line Pattern of Objects

The **Dotted** tool in the **Display** toolbar menu changes the outline pattern of objects between solid and dotted outlines.

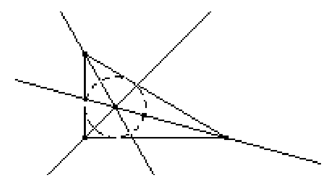
1. Press F7 and select 9:Dotted.
2. Point to a solid outlined object that is to be displayed in dotted outline.

Select the object.



Note*: This option works as a toggle. Reselecting the object changes the outline pattern back to normal.*

3. Press ENTER to change the outline as shown, and then press ENTER again to change it back to normal.

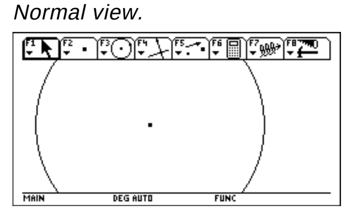


Showing the Entire Drawing Page

The **Show Page** command in the **File** toolbar menu allows you to view an entire construction, which can be larger than the drawing window. It displays the full-page picture of the construction in miniature.

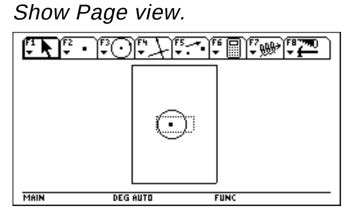
1. Construct a circle that is larger than the drawing window.
2. Press F8 and select A:Show Page.

Normal view.



3. Drag the small window to move the drawing view to a new location.
4. Press ENTER to accept the change or ESC to cancel and return to the normal drawing window.

Show Page view.



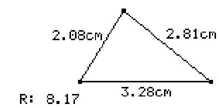
Viewing Data and Objects at the Same Time

***Note:** When you select Data View, the construction is in the left screen, and the Data Matrix Editor is in the right screen. The Data/Matrix Editor stores collected data in the variable* sysData. *If you have not collected data,* sysData *may be empty and no data will be displayed.*

The **Data View** command in the [F8] **File** toolbar menu displays a split screen for viewing a geometry construction and collected data in the Data/Matrix Editor at the same time.

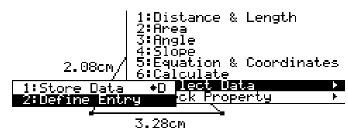
1. Construct and measure an object.

Construct and measure.



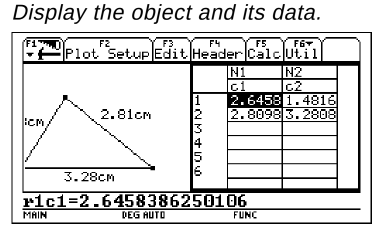
2. Press [F6], select 7:Collect Data, and then 2:Define Entry.
3. Select each data item that you want to define.
4. Press [F6], select 7:Collect Data, and then select 1:Store Data.

Define and store the data.



5. Press [F8] and select B:Data View.
6. Press [2nd] [APPS] to display the Data/Matrix Editor and the stored data and to switch between the two applications.

Display the object and its data.

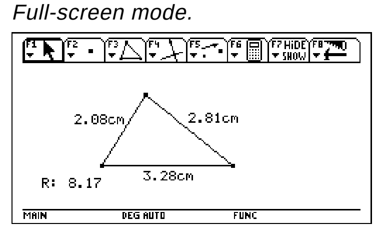


Clearing Data View

The **Clear Data View** command in the **File** toolbar menu brings you back to full-screen mode.

1. Press [F8] and select C:Clear Data View.

Full-screen mode.



Adding Descriptive Information to Objects

The [F7] **Display** toolbar menu contains the tools that let you annotate your constructions.

Creating a Label Using the Label Tool

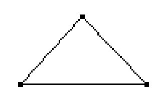
The **Label** tool attaches a label to a point, line, or circle. When you select an object with the **Label** tool, an edit box appears in which you can enter the label text or numbers.

- The label is a textual object that you can move anywhere within a specified distance from the object. The relative position of the label is maintained.
- To edit an existing label, place the cursor on the label and press [ENTER]. A text cursor appears that allows you to edit the text in the label.
- The text cursor is controlled by pressing [♦] and the cursor pad simultaneously.
- All label text is horizontally oriented.

Note: *You also can attach a label to a point immediately after it is created by entering text from the keyboard.*

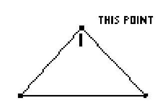
To label an object:

1. Construct any object such as the triangle shown in this example.
2. Press [F7] and select 4:Label.



3. Select a point, line, or circle.

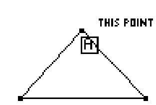
Select a point, line, or circle.



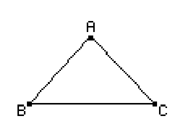
Note: *You can reposition the label by selecting it and then dragging it to the desired location.*

4. Type the label text on the keyboard, and then press [ESC].

Enter a label.



Reposition and complete the labels.



Creating a Descriptive Comment

The **Comment** tool creates a text box in unoccupied space or next to a measurement. It is similar to the **Label** tool except that a comment text box does not attach itself to an object.

1. Press [F7] and select 5:Comment.

Note: The text cursor is controlled by pressing [♦] and the cursor pad simultaneously.

2. Press [ENTER] to create a comment box anywhere in the plane. Drag the comment box by the lower right corner to specify the size of the comment.

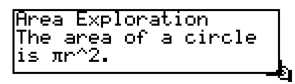
Drag an appropriately sized box.

Hint: Use the Comment tool to add a descriptive label/comment to a measurement.

3. Type the comment text on the keyboard, and then press [ESC].

   You can reposition the comment by dragging it to the desired location.

Enter a comment.



Creating and Editing Numerical Values

The **Numerical Edit** tool creates an edit box for editing numerical values, including interactive numbers or measurements. Interactive numbers must be created with this tool; and they can be interactively modified and used to define rotations, dilations, or measurement transfer values.

1. Press [F7] and select 6:Numerical Edit.

Note: The text cursor is controlled by pressing [♦] and the cursor pad simultaneously.

2. Press [ENTER] to place an edit box anywhere in the drawing for creating an interactive number.

Position the edit box.

3. Type a numerical value, and then press [ESC].

Enter a numerical value.

4. *(Optional)* Add a unit description to a number by pressing [♦] U and selecting from: Number, Length, Area, Volume, Angle.

Assign a unit of measurement.

Moving and Modifying a Number

You can move a number by selecting it and dragging it anywhere in the plane with the **Pointer** tool. You can modify a number when the edit box is active.

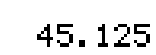
1. Select the number that you want to change.

Select a number to modify.

Note: The I cursor is placed at the right of the least-significant digit.

2. Press [←] to delete the necessary digits, and then re-type the corrected number.

Edit the number with delete and replace.



Tip: *Point to a label, comment, or numerical edit value and press [ENTER] twice to open the appropriate tool automatically.*

3. Press [♦] [↑] or [♦] [↓] to increase or decrease the digit to the left of the cursor, respectively.

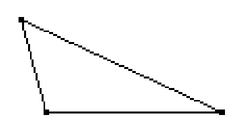
Edit the number with [♦] [↑].

4. Press [ESC] when finished.

Creating a Marked Angle

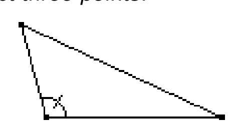
The **Mark Angle** tool labels an angle specified by three points with an angle mark.

1. Create a triangle as shown in this example.
2. Press [F7] and select 7:Mark Angle.



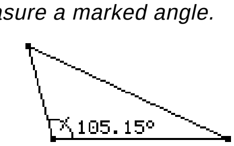
3. Specify the angle by selecting three points. The second point that you select becomes the vertex.

Select three points.



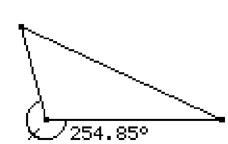
4. Press [F6] and select 3:Angle, and then select the marked angle.

Measure a marked angle.



5. To measure the exterior angle, drag the angle mark through the vertex of the angle.

Measure the exterior angle.



Creating Macros

The [F4] **Construction** toolbar menu contains the tools for constructing macros.

Introduction to Creating Macros

The **Macro Construction** menu item contains the tools for constructing macros in the Geometry application. A macro is a sequence of interdependent constructions. Macros are useful for creating new tools that construct unique objects or perform repetitive tasks.

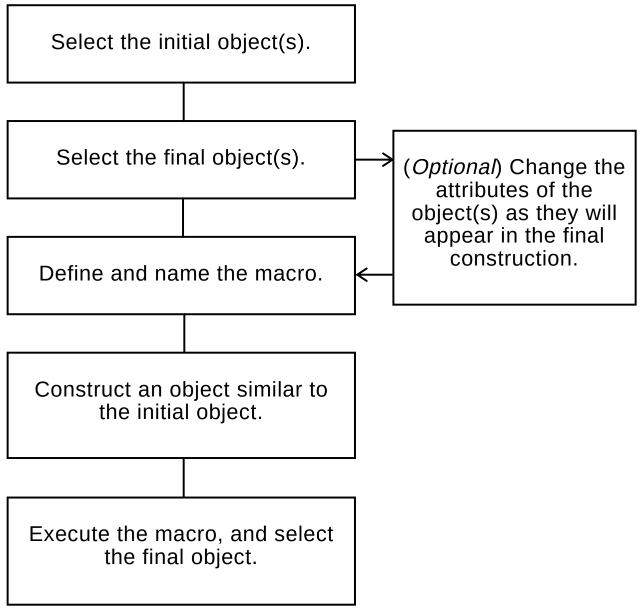
A macro constructs “final” objects based on “initial” objects. Intermediate objects are not constructed. This feature allows for easy construction of complex figures and is the primary method for constructing fractals. You can save macros for later use. Macros are saved automatically with any construction in which they are used. The number of objects created by a macro is limited only by available system memory.

Rules for Creating Macros

Rule	Explanation
• Initial objects must allow for the construction of all final objects.	Final objects are determined by the initial objects. A macro must respect the logical structure of the figure as it was constructed.
• An object cannot exist without the points that define it.	For example, a triangle cannot exist without its vertices. Therefore, when you select an object as an initial object, the macro is able to refer to the points that define the object.
• When you select Define Macro, a macro generates its final objects with the object’s existing attributes.	You can change these attributes during an intermediate step before you select Define Macro. In this way, you can hide objects (using Hide/Show in the **Display** menu that were selected as initial objects.
• Comments and labels cannot be defined as final objects.	Macros are intended as general purpose construction tools, like those in the **Construction** menu. You can select measurements and numerical values as final objects, but any text attached will not be duplicated when the macro executes.
• The location of an arbitrary point on an object is determined by random-number generation.	The position of the point will be uncertain if it is selected as a final object and may result in an incorrectly defined macro.
• The order that initial objects are used depends upon the similarity of their types.	For example, lines and circles are different types, and they are not used in any order. When they are the same type, the macro uses them in the order in which they were selected as initial objects.

Overview: Creating and Executing a Macro

The flowchart below shows an overview of the basic steps required to create macros.



The **Execute Macro** command displays a pop-up menu that lists all defined macros. If the initial conditions of the selected macro are satisfied, the macro executes and generates the final object or objects.

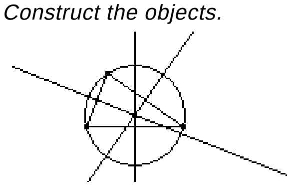
Example: Creating and Executing a Macro

To create and execute a macro:

1. Construct the initial and final objects.

   For example, construct a triangle (initial object) and its perpendicular bisectors, and then construct a circle (final object) through all vertices of the triangle.

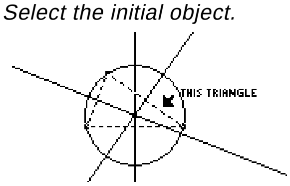
Construct the objects.



2. Press [F4] and select 6:Macro Construction.

3. Select 2:Initial Objects, and then select the triangle as the initial object.

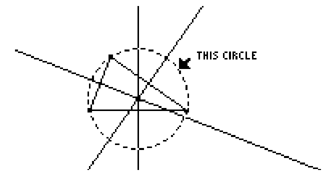
Select the initial object.



Example: Creating and Executing a Macro (Continued)

4. Press [F4] and select 6:Macro Construction.

5. Select 3:Final Objects, and then select the circle as the final object.

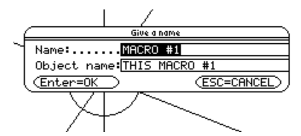


Select the final object.

6. (Optional) You can change the appearance of your construction by using the **Hide/Show**, **Thick**, and **Dotted** tools in the [F7] **Display** toolbar menu.

7. Press [F4] and select 6:Macro Construction.

8. Select 4:Define Macro, and then type a name for the macro.

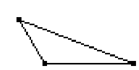


Name the macro.

The Name you enter will help you identify the macro later. The Object name you enter will appear in cursor messages when appropriate. Both names can be up to 25 characters.

Note: After the Name Macro dialog has been completed, the Save Macro dialog will appear. You must provide a valid name to save your macro as a separate file. If you do not want to save the macro to a separate file, the macro will be saved with your construction. In this case, you will not be able to open the macro from the [F8] **File** toolbar menu.

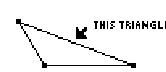
9. Construct the initial object (any triangle).



Construct an object.

10. Press [F4] and select 6:Macro Construction, and then select 1:Execute Macro.

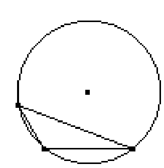
11. Select the macro that you previously defined, and then select the triangle to execute the macro.



Select the object.

***Note:** Defined macros appear in a pop-up menu. Highlight the desired macro, and press [ENTER] to select it.*

This macro determines the center and radius of the circle and constructs a circle thorough all vertices of the triangle.



Execute the macro.

Geometry Toolbar Menu Items

This section shows the geometry toolbar and the subsequent Tool/Command menu items that are opened when you press one of the function keys F1 through F8.

Pointer Toolbar Menu

The [F1] **Pointer** toolbar menu contains tools for selecting and performing freehand transformations.

F1	
1:Pointer	see page 120
2:Rotate	see page 141
3:Dilate	see page 143
4:Rotate & Dilate	see page 145

Points and Lines Toolbar Menu

The [F2] **Points and Lines** toolbar menu contains tools for constructing points or linear objects.

F2	
1:Point	see page 122
2:Point on Object	see page 123
3:Intersection Point	see page 123
4:Line	see page 124
5:Segment	see page 124
6:Ray	see page 125
7:Vector	see page 125

Curves and Polygons Toolbar Menu

The [F3] **Curves and Polygons** toolbar menu contains tools for constructing circles, arcs, triangles, and polygons.

F3	
1:Circle	see page 127
2:Arc	see page 128
3:Triangle	see page 129
4:Polygon	see page 130
5:Regular Polygon	see page 131

Construction Toolbar Menu

The [F4] **Construction** toolbar menu contains Euclidean geometry construction tools as well as a **Macro Construction** tool for creating new tools.

F4	
1:Perpendicular Line	see page 132
2:Parallel Line	see page 133
3:Midpoint	see page 135
4:Perpendicular Bisector	see page 134
5:Angle Bisector	see page 134
6:Macro Construction ▸	see page 164
7:Vector Sum	see page 126
8:Compass	see page 127
9:Measurement Transfer	see page 136
A:Locus	see page 138
B:Redefine Point	see page 139

Transformations Menu

The [F5] **Transformations** toolbar menu contains tools for transformational geometry.

F5	
1:Translation	see page 140
2:Rotation	see page 142
3:Dilation	see page 144
4:Reflection	see page 146
5:Symmetry	see page 147
6:Inverse	see page 148

Measurement Menu

The [F6] **Measurement** toolbar menu contains tools for performing measurements and calculations.

F6		
1:Distance & Length		see page 149
2:Area		see page 149
3:Angle		see page 150
4:Slope		see page 150
5:Equation & Coordinates		see page 151
6:Calculate		see page 152
7:Collect Data	▶	see page 153
B:Check Property	▶	see page 154

Display Menu

The [F7] **Display** toolbar menu contains tools for annotating constructions or animating objects.

F7	
1:Hide / Show	see page 158
2:Trace On / Off	see page 157
3:Animation	see page 156
4:Label	see page 161
5:Comment	see page 162
6:Numerical Edit	see page 162
7:Mark Angle	see page 163
8:Thick	see page 158
9:Dotted	see page 159

File Menu

The [F8] **File** toolbar menu contains file operations and editing functions.

***Note:** Cut, copy, and paste are not available in the Geometry application.*

F8		
1:Open...	[♦] O	see page 116
2:Save as...	[♦] S	see page 116
3:New...	[♦] N	see page 116
4:Cut		see Note
5:Copy		see Note
6:Paste		see Note
7:Delete	[←]	see page 121
8:Clear All		see page 121
9:Format...	[♦] F	see page 117
A:Show Page		see page 159
B:Data View		see page 160
C:Clear Data View		see page 160
D:Undo	[♦] Z	see page 115

Pointing Indicators and Terms Used in Geometry

This section describes the various pointing indicators that are used in the procedures, and a glossary of terms

Pointers That Guide You

Several types of pointers exist to help guide you through your constructions. The pointers are shown and described below.

Cursor Display/Name	Active when...
arrow	The pointer is on an object.
+ cross hair	A Pointer indicator is selected or the cursor is in motion.
construction pencil	A construction tool is active.
selection pencil	A construction tool is active and a point can be placed on an object.
dragging hand	A selected object can be moved.
open hand	[2nd] and the cursor pad (↑, ↗, →, ↘, ↓, ↙, ←, ↖) are pressed at the same time to scroll the display anywhere within the plane.
I-beam	Text or numbers can be entered or edited in a label or comment box.
crossed lines	The comment box is active.
paint brush	Thick or dotted lines are selected.

Glossary of Geometry Definitions

The following terms are used in this chapter to describe specific TI-92 Geometry operations.

[ENTER]	Press any of the three [ENTER] keys on the TI-92 to execute a command or to confirm an action.
drag	Drag means to point to the object that you want to move, press and hold (drag key) to select the object, and then move the screen pointer to a new location. Release to stop dragging.
marquee outline	A marquee outline shows the outline of an object using animated dots instead of a solid line.
page/plane	The page is a virtual working area of the plane. The plane is 7.5 by 10.0 inches (19.05 by 25.4 centimeters).
point	When used as an instruction, point means to place the screen pointer on the object you want to select.
select	When used as an instruction, select means pointing to an object and pressing [ENTER].

Helpful Shortcuts

Use the suggestions in the following table to quickly access or perform specific geometry functions.

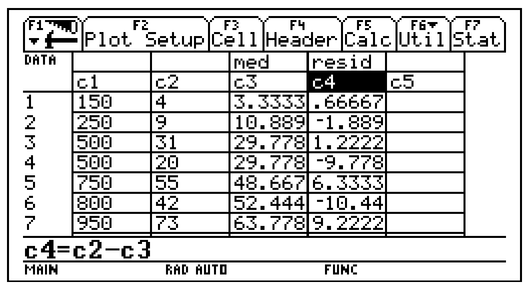
Press [♦] [ON].	• To turn off the TI-92 without exiting Geometry.
Press [♦] Z.	• To undo the last completed operation.
Press [ESC].	• To return to the **Pointer** tool from anywhere.
Select an object and press [+] or [−].	• To increase or decrease the displayed precision of selected numerical values. • To increase or decrease the number of objects in a selected locus. • To increase or decrease the animation speed.
Press [↑].	• To limit the slope of lines, rays, segments, vectors, triangles, or polygons to increments of 15 degrees when creating these objects. • To select multiple objects.
Press [hand] once.	• To display all basic points (those points which you can drag) as flashing points. The cursor must be in unoccupied space.
Press [hand] twice.	• To begin animation of an object. The **Animation** tool must be selected and the cursor pointing to the object.
Press [ENTER] once.	• To deselect selected objects. The pointer must be in unoccupied space.
Press [ENTER] twice.	• On the final point of a polygon, to complete construction of the polygon. • On a label, comment, or numerical value to invoke the appropriate editor.
Press [↑] and [ENTER].	• To deselect all hidden or traced objects. The appropriate tool must be selected and the cursor must be in unoccupied space.
Press [♦] and the cursor key.	• To edit or change numerical values, comments, or labels.
Begin typing immediately after:	• Creating a point, line, or circle to add a label to an object. The label is limited to five characters and can only be edited with the **Label** tool. • Creating a measurement to add a comment to the measurement.

Data/Matrix Editor

8

The Data/Matrix Editor serves two main purposes.

- This chapter describes how to use the Data/Matrix Editor to create and maintain a list, matrix, or data variable.



- Chapter 9 describes how to use the Data/Matrix Editor to perform statistical calculations and graph statistical plots.

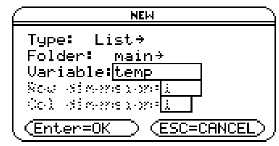
Preview of the Data/Matrix Editor

Use the Data/Matrix Editor to create a one-column list variable. Then add a second column of information. Notice that the list variable (which can have only one column) is automatically converted into a data variable (which can have multiple columns).

Steps | **Keystrokes** | **Display**

1. Start the Data/Matrix Editor and create a new list variable named TEMP.

   Keystrokes: [APPS] 6 3 / [→] 3 / [↓] [↓] / T E M P / [ENTER] [ENTER]

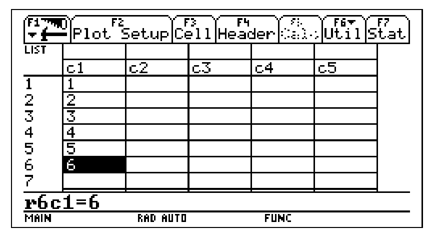


2. Enter a column of numbers. Then move the cursor up one cell (just to see that a highlighted cell's value is shown on the entry line).

   LIST is shown in the upper-left corner to indicate a list variable.

   You can use [↓] instead of [ENTER] to enter information in a cell.

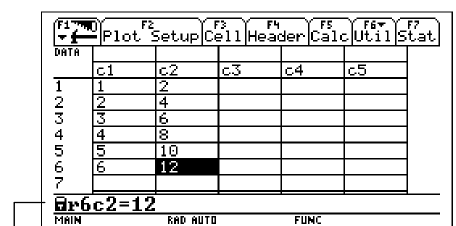
   Keystrokes: 1 [ENTER] / 2 [ENTER] / 3 [ENTER] / 4 [ENTER] / 5 [ENTER] / 6 [ENTER] / [↑]



3. Move to column 2, and define its column header so that it is twice the value of column 1.

   DATA is shown in the upper-left corner to indicate that the list variable was converted to a data variable.

   Keystrokes: [→] / [F4] / 2 [×] C 1 / [ENTER]

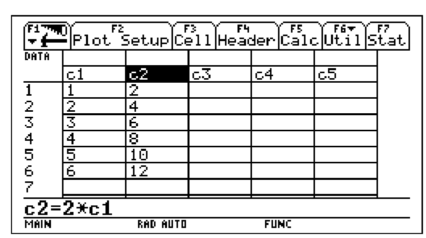


means the cell is in a defined column.

4. Move to the column 2 header cell to show its definition in the entry line.

   When the cursor is on the header cell, you do not need to press [F4] to define it. Simply begin typing the expression.

   Keystrokes: [2nd] [↑] / [↑]



5. Go to the Home screen, and then return to the current variable.

   Keystrokes: [♦] [HOME] / [APPS] 6 1

6. Clear the contents of the variable.

   Simply clearing the data does not convert the data variable back into a list variable.

   Keystrokes: [F1] 8 [ENTER]

Tip: If you don't need to save the current variable, use it as a *scratchpad*. The next time you need a variable for temporary data, clear the current variable and re-use it. This lets you enter temporary data without creating a new variable each time, which uses up memory.

Overview of List, Data, and Matrix Variables

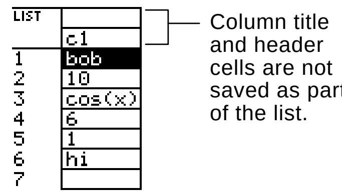
To use the Data/Matrix Editor effectively, you must understand list, data, and matrix variables.

List Variable

__Note:__ If you enter more than one column of elements in a list variable, it is converted automatically into a data variable.

A list is a series of items (numbers, expressions, or character strings) that may or may not be related. Each item is called an element. In the Data/Matrix Editor, a list variable:

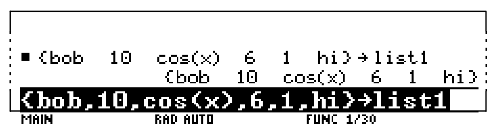
- Is shown as a single column of elements, each in a separate cell.
- Must be continuous; blank or empty cells are not allowed within the list.
- Can have up to 999 elements.



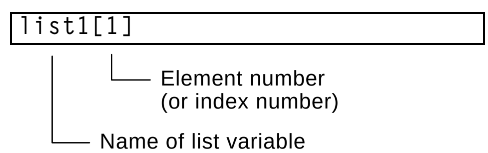
__Tip:__ After creating a list in the Data/Matrix Editor, you can use the list in any application (such as the Home screen).

On the Home screen (or anywhere else you can use a list), you can enter a list as a series of elements enclosed in braces { } and separated by commas.

Although you must use commas to separate elements on the entry line, spaces separate the elements in the history area.



To refer to a specified element in a list, use the format shown to the right.



Data Variable

__Note:__ For stat calculations, columns must have the same length.

A data variable is essentially a collection of lists that may or may not be related. In the Data/Matrix Editor, a data variable:

- Can have up to 99 columns.
- Can have up to 999 elements in each column. Depending on the kind of data, all columns may not have to be the same length.

DATA	c1	c2	c3	c4	c5
1	fred	stone	95	86	94
2	sally	ross	75	79	83
3	jane	smith	97	96	97
4	nick	castle	83	88	91
5	betty	brant	90	93	100
6	terry	miller	86	91	86
7	mike	reid	69	75	78

- Must have continuous columns; blank or empty cells are not allowed within a column.

Data Variable (Continued)

From the Home screen or a program, you can use the **NewData** command to create a data variable that consists of existing lists.

```
NewData data1,list1,list2
```

- Name of data variable to create: data1
- Names of existing list variables: list1,list2

Although you cannot directly display a data variable on the Home screen, you can display a specified column or element.

```
data1[1]
(data1[1])[1]
```

- data1[1]: Name of data variable (data1); Column number ([1])
- (data1[1])[1]: Column number (first [1]); Element number in the column (second [1])

For example:

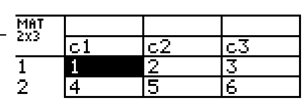
Displays column 1 of the variable data1.

Displays element 1 in column 1 of the variable data1.

Matrix Variable

A matrix is a rectangular array of elements. When you create a matrix in the Data/Matrix Editor, you must specify the number of rows and columns (although you can add or delete rows and columns later). In the Data/Matrix Editor, a matrix variable:

- Looks similar to a data variable, but all columns must have the same length.
- Is initially created with 0 in each cell. You can then enter the applicable value in place of 0.



Shows the size of the matrix.

Tip: After creating a matrix in the Data/Matrix Editor, you can use the matrix in any application (such as the Home screen).

From the Home screen or a program, you can use [STO▶] to store a matrix with either of the equivalent methods shown to the right.

```
[[1,2,3][4,5,6]]→mat1
[1,2,3;4,5,6]→mat1
```

- In `[[1,2,3][4,5,6]]→mat1`: row 1 is [1,2,3], row 2 is [4,5,6].
- In `[1,2,3;4,5,6]→mat1`: row 1 is 1,2,3, row 2 is 4,5,6.

Note: Use brackets to refer to a specific element in a matrix. For example, enter mat1[2,1] to access the 1st element in the 2nd row.

Although you enter the matrix as shown above, it is pretty printed in the history area in traditional matrix form.

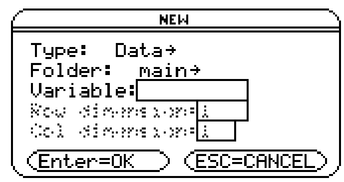
Starting a Data/Matrix Editor Session

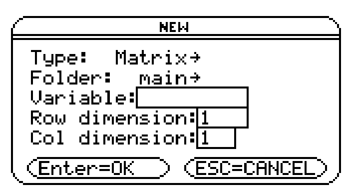
Each time you start the Data/Matrix Editor, you can create a new variable, resume using the current variable (the variable that was displayed the last time you used the Data/Matrix Editor), or open an existing variable.

Creating a New Data, Matrix, or List Variable

1. Press [APPS] and then select 6:Data/Matrix Editor.
2. Select 3:New.

3. Specify the applicable information for the new variable.



Item	Lets you:
Type	Select the type of variable to create. Press ► to display a menu of available types. 
Folder	Select the folder in which the new variable will be stored. Press ► to display a menu of existing folders. For information about folders, refer to Chapter 10.
Variable	Type a new variable name. If you specify a variable that already exists, an error message will be displayed when you press [ENTER]. When you press [ESC] or [ENTER] to acknowledge the error, the NEW dialog box is redisplayed.
Row dimension and Col dimension	If Type = Matrix, type the number of rows and columns in the matrix. 

Note: *If you do not type a variable name, the TI-92 will display the Home screen.*

4. Press [ENTER] (after typing in an input box such as Variable, press [ENTER] twice) to create and display an empty variable in the Data/Matrix Editor.

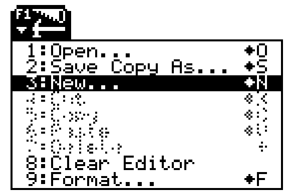
Using the Current Variable

You can leave the Data/Matrix Editor and go to another application at any time. To return to the variable that was displayed when you left the Data/Matrix Editor, press [APPS] 6 and select 1:Current.

Creating a New Variable from the Data/Matrix Editor

From the Data/Matrix Editor:

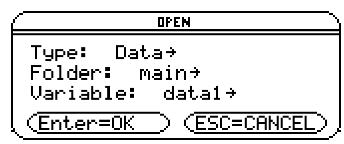
1. Press [F1] and select 3:New. (You can press [♦] N instead of using the [F1] toolbar menu.)
2. Specify the type, folder, and variable name. For a matrix, also specify the number of rows and columns.



Opening Another Variable

You can open another variable at any time.

1. From the Data/Matrix Editor, press [F1] and select 1:Open. (You can press [♦] O instead of using the [F1] toolbar menu.)
   — or —
   From any application, press [APPS] 6 and select 2:Open.
2. Select the type, folder, and variable to open.
3. Press [ENTER].



Note: Variable *shows the first existing variable in alphabetic order. If there are no existing variables, nothing is displayed.*

Note about Deleting a Variable

Because all Data/Matrix Editor variables are saved automatically, you can accumulate quite a few variables, which take up memory.

To delete a variable, use the VAR-LINK screen ([2nd] [VAR-LINK]). For information about VAR-LINK, refer to Chapter 18.

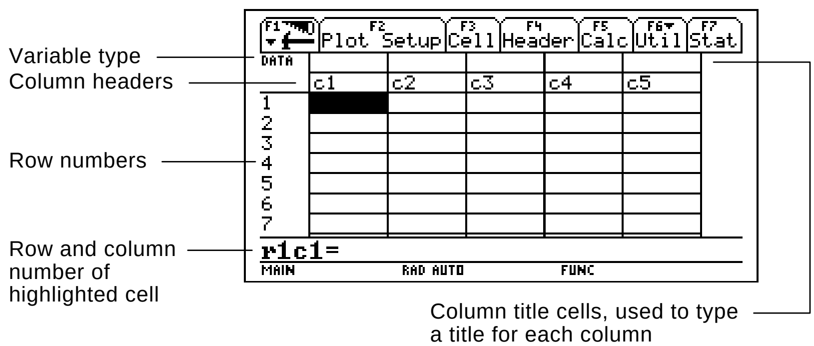
Entering and Viewing Cell Values

If you create a new variable, the Data/Matrix Editor is initially blank (for a list or data variable) or filled with zeros (for a matrix). If you open an existing variable, the values in that variable are displayed. You can then enter additional values or edit the existing ones.

The Data/Matrix Editor Screen

__Tip:__ Use the title cell at the very top of each column to identify the information in that column.

A blank Data/Matrix Editor screen is shown below. When the screen is displayed initially, the cursor highlights the cell at row 1, column1.



When values are entered, the entry line shows the full value of the highlighted cell.

Entering or Editing a Value in a Cell

__Tip:__ To enter a new value, you can start typing without pressing [ENTER] or [F3] first. However, you must use [ENTER] or [F3] to edit an existing value.

__Note:__ To enter a value from the entry line, you can also use ⓓ or ⓤ.

You can enter any type of expression in a cell (number, variable, function, string, etc.).

1. Move the cursor to highlight the cell you want to enter or edit.
2. Press [ENTER] or [F3] to move the cursor to the entry line.
3. Type a new value or edit the existing one.
4. Press [ENTER] to enter the value into the highlighted cell.

When you press [ENTER], the cursor automatically moves to highlight the next cell so that you can continue entering or editing values. However, the variable type affects the direction that the cursor moves.

Variable Type	After [ENTER], the cursor moves:
List or data	Down to the cell in the next row.
Matrix	Right to the cell in the next column. From the last cell in a row, the cursor automatically moves to the first cell in the next row. This lets you enter values for row1, row2, etc.

Scrolling through the Editor

To move the cursor:	Press:
One cell at a time	↓, ↑, →, or ←
One page at a time	[2nd] and then ↓, ↑, →, or ←

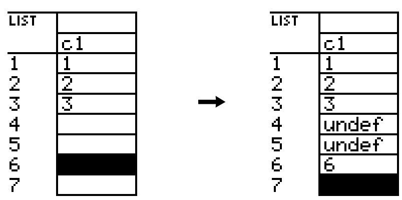
When you scroll down/up, the header row remains at the top of the screen so that the column numbers are always visible. When you scroll right/left, the row numbers remain on the left side of the screen so that they are always visible.

How Rows and Columns Are Filled Automatically

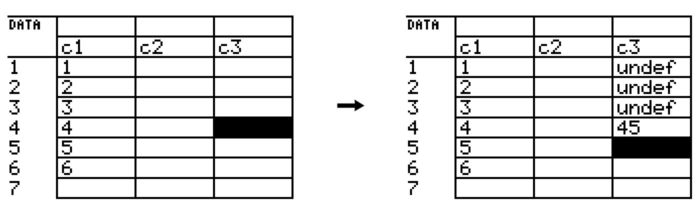
When you enter a value in a cell, the cursor moves to the next cell. However, you can move the cursor to any cell and enter a value. If you leave gaps between cells, the TI-92 handles the gaps automatically.

- In a list variable, a cell in the gap is *undefined* until you enter a value for the cell.

Note: *If you enter more than one column of elements in a list variable, it is converted automatically into a data variable.*

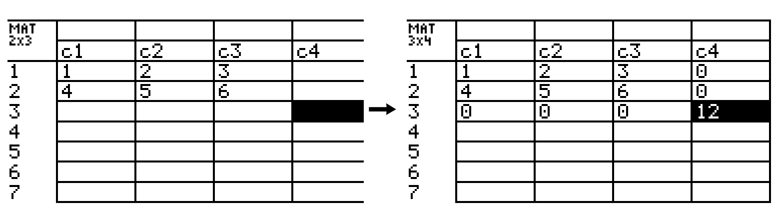


- In a data variable, gaps in a column are handled the same as a list. However, if you leave a gap between columns, that column is blank.



Note: *Although you specify the size of a matrix when you create it, you can easily add additional rows and/or columns.*

- In a matrix variable, when you enter a value in a cell outside the current boundaries, additional rows and/or columns are added automatically to the matrix to include the new cell. Other cells in the new rows and/or columns are filled with zeros.

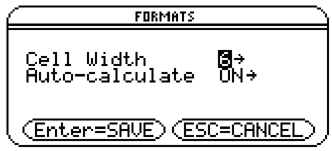


Changing the Cell Width

Tip: ***Remember, to see a number in full precision, you can always highlight the cell and look at the entry line.***

The cell width affects how many characters are displayed in any cell. To change the cell width in the Data/Matrix Editor:

1. Press [♦] F or [F1] 9 to display the FORMATS dialog box.



Cell width is the maximum number of characters that can be displayed in a cell.

All cells have the same cell width.

2. With the current Cell Width setting highlighted, press ▶ or ◀ to display a menu of digits (3 through 12).
3. Move the cursor to highlight a number and press [ENTER]. (For single-digit numbers, you can type the number and press [ENTER].)
4. Press [ENTER] to close the dialog box.

Clearing a Column or all Columns

Note: *For a list or data variable, a clear column is empty. For a matrix, a clear column contains zeros.*

This procedure erases the contents of a column. It does not delete the column.

To clear:	Do this:
A column	1. Move the cursor to any cell in the column. 2. Press [F6] and select 5:Clear Column. (This item is not available for a matrix.)
All columns	Press [F1] and select 8:Clear Editor. When prompted for confirmation, press [ENTER] (or [ESC] to cancel).

Inserting and Deleting a Row, Column, or Cell

The general procedures for inserting and deleting a cell, row, or column are simple and straightforward. You can have up to 99 columns with up to 999 elements in each column.

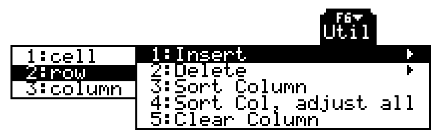
Note About Column Titles and Headers

You cannot delete the rows or cells that contain column titles or headers. Also, you cannot insert a row or cell before a column title or header.

Inserting a Row or Column

The new row or column is inserted *before* the row or column that contains the highlighted cell. In the Data/Matrix Editor:

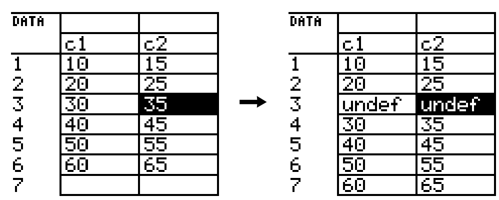
1. Move the cursor to any cell in the applicable row or column.
2. Press [F6] and select 1:Insert.
3. Select either 2:row or 3:column.



Note: *For a list variable, inserting a row is the same as inserting a cell.*

When you insert a row:

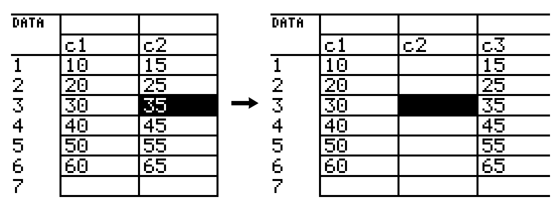
- In a list or data variable, the row is *undefined*.
- In a matrix variable, the row is filled with zeros.



Note: *For a list variable, you cannot insert a column because a list has only one column.*

When you insert a column:

- In a data variable, the column is blank.
- In a matrix variable, the column is filled with zeros.

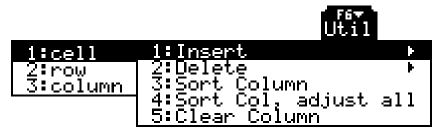


You can then enter values in the undefined or blank cells.

Inserting a Cell

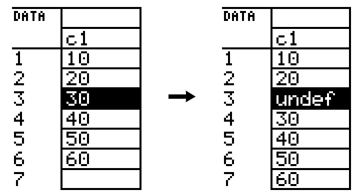
The new cell is inserted *before* the highlighted cell in the same column. (You cannot insert a cell into a locked column, which is defined by a function in the column header. Refer to page 182.) In the Data/Matrix Editor:

1. Move the cursor to the applicable cell.
2. Press [F6] and select 1:Insert.
3. Select 1:cell.



Note: For a matrix variable, you cannot insert a cell because the matrix must retain a rectangular shape.

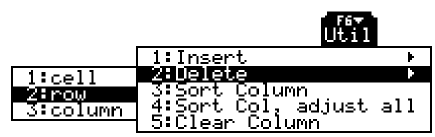
The inserted cell is undefined. You can then enter a value in the cell.



Deleting a Row or Column

In the Data/Matrix Editor:

1. Move the cursor to any cell in the row or column you want to delete.
2. Press [F6] and select 2:Delete.
3. Select either 2:row or 3:column.

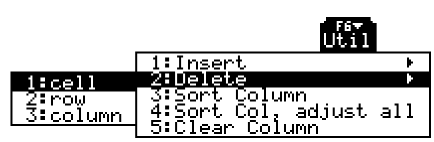


If you delete a row, any rows below the deleted row are shifted up. If you delete a column, any columns to the right of the deleted column are shifted left.

Deleting a Cell

In the Data/Matrix Editor:

1. Move the cursor to the cell you want to delete. (You cannot delete a cell in a locked column, which is defined by a function in the column header. Refer to page 182.)
2. Press [F6] and select 2:Delete.
3. Select 1:cell.



Note: For a matrix variable, you cannot delete a cell because the matrix must retain a rectangular shape.

Any cells below the deleted cell are shifted up.

If You Need to Add a New "Last" Row, Column, or Cell

You do *not* need to use the [F6] Util toolbar menu to:

- Add a new row or cell at the bottom of a column.
  — or —
- Add a new column to the right of the last column.

Simply move the cursor to the applicable cell and enter a value.

Defining a Column Header with an Expression

For a list variable or a column in a data variable, you can enter a function in the column header that automatically generates a list of elements. In a data variable, you can also define one column in terms of another.

Entering a Header Definition

In the Data/Matrix Editor:

1. Move the cursor to any cell in the column and press [F4].
   — or —
   Move the cursor to the header cell (c1, c2, etc.) and press [ENTER].

   Note: [ENTER] is not required if you want to type a new definition or replace the existing one. However, if you want to edit the existing definition, you must press [ENTER].

__Tip:__ To view an existing definition, press [F4] or move the cursor to the header cell and look at the entry line.

2. Type the new expression, which replaces any existing definition.

   If you used [F4] or [ENTER] in Step 1, the cursor moved to the entry line and highlighted the existing definition, if any. You can also:

   - Press [CLEAR] to clear the highlighted expression. Then type the new expression.
     — or —
   - Press ◀ or ▶ to remove the highlighting. Then edit the old expression.

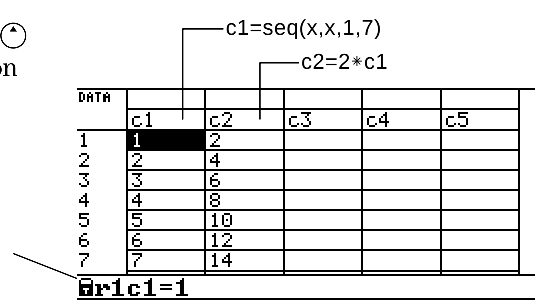
__Tip:__ To cancel any changes, press [ESC] before pressing [ENTER].

You can use an expression that:	For example:
Generates a series of numbers.	c1=seq(x^2,x,1,5) c1={1,2,3,4,5}
Refers to another column.	c2=2∗c1 c4=c1∗c2−sin(c3)

*__Note:__ The **seq** function is described in Appendix A.*

__Note:__ If you refer to an empty column, you will get an error message (unless Auto-calculate = OFF as described on page 183).

3. Press [ENTER], ▼, or ▲ to save the definition and update the columns.



__Note:__ For a data variable, header definitions are saved when you leave the Data/ Matrix Editor. For a list variable, the definitions are not saved (only their resulting cell values).

Clearing a Header Definition

1. Move the cursor to any cell in the column and press [F4].
   — or —
   Move the cursor to the header cell (c1, c2, etc.) and press [ENTER].
2. Press [CLEAR] to clear the highlighted expression.
3. Press [ENTER], ▼, or ▲.

Using an Existing List as a Column

Suppose you have one or more existing lists, and you want to use those existing lists as columns in a data variable.

Note: If you have a CBL 2/CBL or CBR, use these techniques for your collected lists.

Tip: Use [2nd] [VAR-LINK] to see existing list variables.

From the:	Do this:
Data/Matrix Editor	In the applicable column, use [F4] to define the column header. Refer to the existing list variable. For example: c1=list1
Home screen or a program	Use the **NewData** command as described in Appendix A. For example: **NewData** *datavar*, *list1* [, *list2*] [, *list3*] ... *list1* [, *list2*] [, *list3*] ... — Existing list variables to copy to columns in the data variable. *datavar* — Data variable. If this data variable already exists, it will be redefined based on the specified lists.

To Fill a Matrix with a List

You cannot use the Data/Matrix Editor to fill a matrix with a list. However, you can use the **list▸mat** command from the Home screen or a program. For information, refer to Appendix A.

The Auto-calculate Feature

For list and data variables, the Data/Matrix Editor has an Auto-calculate feature. By default, Auto-calculate = ON. Therefore, if you make a change that affects a header definition (or any column referenced in a header definition), all header definitions are recalculated automatically. For example:

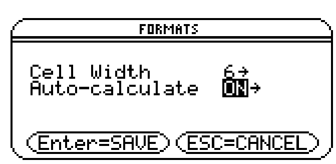
- If you change a header definition, the new definition is applied automatically.
- If column 2's header is defined as c2=2∗c1, any change you make in column 1 is automatically reflected in column 2.

Tip: You may want to set Auto-calculate = OFF to:

- *Make multiple changes without recalculating each time.*
- *Enter a definition such as c1=c2+c3 before you enter columns 2 and 3.*
- *Override any errors in a definition until you can debug the error.*

To turn Auto-calculate off and on from the Data/Matrix Editor:

1. Press [♦] F or [F1] 9.
2. Change Auto-Calculate to OFF or ON.
3. Press [ENTER] to close the dialog box.



If Auto-calculate = OFF and you make changes as described above, the header definitions are not recalculated until you set Auto-calculate = ON.

Using Shift and CumSum Functions in a Column Header

When defining a column header, you can use the **shift** and **cumSum** functions as described below. These descriptions differ slightly from Appendix A. This section describes how to use the functions in the Data/Matrix Editor. Appendix A gives a more general description for the Home screen or a program.

Using the Shift Function

The **shift** function copies a base column and shifts it up or down by a specified number of elements. Use [F4] to define a column header with the syntax:

shift (*column* [,*integer*])

Number of elements to shift (positive shifts up; negative shifts down). Default is ⁻1.

Column used as the base for the shift.

For example, for a two-element shift up and down:

c2=shift(c1,2)

c3=shift(c1,⁻2)

c1	c2	c3
1	3	undef
2	4	undef
3	5	1
4	6	2
5	undef	3
6	undef	4

Shifted columns have the same length as the base column (c1).

Last two elements of c1 shift down and out the bottom; undefined elements shift into the top.

First two elements of c1 shift up and out the top; undefined elements shift into the bottom.

Note: *To enter "shift", type it from the keyboard or select it from* [2nd] [CATALOG].

Using the CumSum Function

The **cumSum** function returns a cumulative sum of the elements in a base column. Use [F4] to define a column header with the syntax:

cumSum (*column*)

Column used as the base for the cumulative sum

For example:

c2=cumSum(c1)

c1	c2
1	1
2	3
3	6
4	10
5	15
6	21

1+2

1+2+3+4

1+2+3+4+5+6

Note: *To enter "cumSum", type it, select it from* [2nd] [CATALOG]*, or press* [2nd] [MATH] *and select it from the List submenu.*

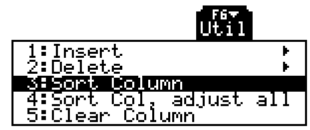
Sorting Columns

After entering information in a data, list, or matrix variable, you can easily sort a specified column in numeric or alphabetical order. You can also sort all columns as a whole, based on a "key" column.

Sorting a Single Column

In the Data/Matrix Editor:

1. Move the cursor to any cell in the column.
2. Press [F6] and select 3:Sort Column.



Numbers are sorted in ascending order.

Character strings are sorted in alphabetical order.

c1		c1
fred	→	75
sally		82
chris	→	98
jane		chris
75	→	fred
98		jane
82		sally

Sorting All Columns Based on a "Key" Column

Consider a database structure in which each column along the same row contains related information (such as a student's first name, last name, and test scores). In such a case, sorting only a single column would destroy the relationship between the columns.

In the Data/Matrix Editor:

Note: For a list variable, this is the same as sorting a single column.

1. Move the cursor to any cell in the "key" column.

   In this example, move the cursor to the second column (c2) to sort by last name.

c1	c2	c3	c4	c5
fred	stone	95	86	94
sally	ross	75	79	83
jane	smith	97	96	97
nick	castle	83	88	91
betty	brant	90	93	100
terry	miller	86	91	86
mike	reid	69	75	78

Note: This menu item is not available if any column is locked.

2. Press [F6] and select 4:Sort Col, adjust all.

c1	c2	c3	c4	c5
betty	brant	90	93	100
nick	castle	83	88	91
terry	miller	86	91	86
mike	reid	69	75	78
sally	ross	75	79	83
jane	smith	97	96	97
fred	stone	95	86	94

When using this procedure for a data variable:

- All columns must have the same length.
- None of the columns can be locked (defined by a function in the column header). When the cursor is in a locked column, 🔒 is shown at the beginning of the entry line.

Saving a Copy of a List, Data, or Matrix Variable

You can save a copy of a list, data, or matrix variable. You can also copy a list to a data variable, or you can select a column from a data variable and copy that column to a list.

Valid Copy Types

Note: A list is automatically converted to a data variable if you enter more than one column of information.

You can copy a:	To a:
List	List or data
Data	Data
Data column	List
Matrix	Matrix

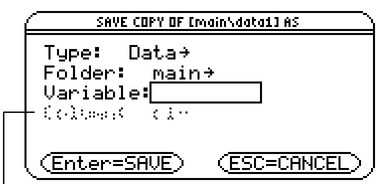
Procedure

Tip: You can press [♦] S instead of using the [F1] toolbar menu.

Note: If you type the name of an existing variable, its contents will be replaced.

From the Data/Matrix Editor:

1. Display the variable that you want to copy.
2. Press [F1] and select 2:Save Copy As.
3. In the dialog box:
   - Select the Type and Folder for the copy.
   - Type a variable name for the copy.
   - When available, select the column to copy from.



Column is dimmed unless you copy a data column to a list. The column information is not used for other types of copies.

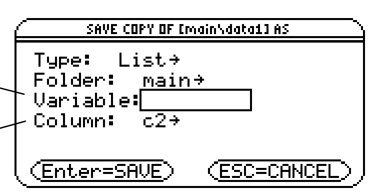
4. Press [ENTER] (after typing in an input box such as Variable, you must press [ENTER] twice).

To Copy a Data Column to a List

A data variable can have multiple columns, but a list variable can have only one column. Therefore, when copying from a data variable to a list, you must select the column that you want to copy.

List variable to copy to.

Data column that will be copied to the list. By default, this shows the column that contains the cursor.

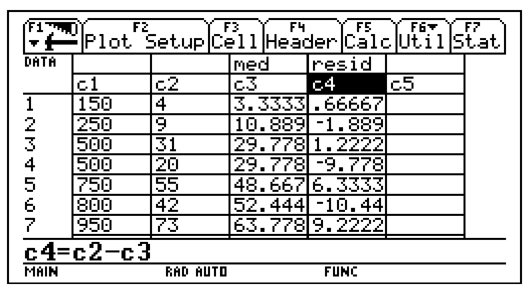


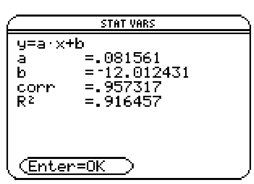
Statistics and Data Plots

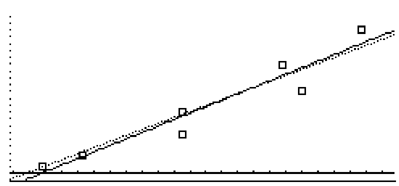
9

The Data/Matrix Editor serves two main purposes.

- As described previously in Chapter 8, the Data/Matrix Editor lets you create and maintain a list, matrix, or data variable.
- This chapter describes how to use the Data/Matrix Editor to perform statistical calculations and graph statistical plots.







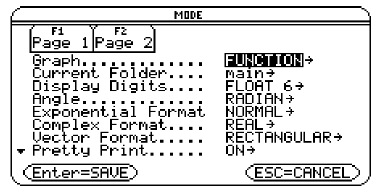
Preview of Statistics and Data Plots

Based on a sample of seven cities, enter data that relates population to the number of buildings with more than 12 stories. Using Median-Median and linear regression calculations, find and plot equations to fit the data. For each regression equation, predict how many buildings of more than 12 stories you would expect in a city of 300,000 people.

Steps	Keystrokes	Display

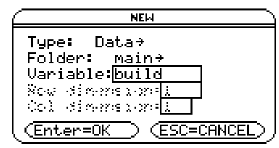
1. Display the MODE dialog box. For Graph mode, select FUNCTION.

   Keystrokes: [MODE] ⊙ 1 [ENTER]



2. Display the Data/Matrix Editor, and create a new data variable named BUILD.

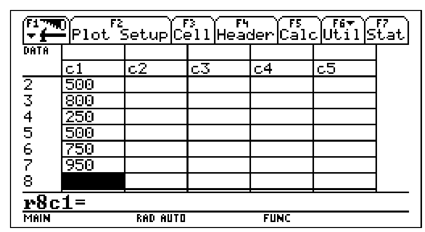
   Keystrokes: [APPS] 6 3 ⊙ ⊙ B U I L D [ENTER] [ENTER]



3. Using the sample data below, enter the population in column 1.

Pop. (in 1000s)	Bldgs > 12 stories
150	4
500	31
800	42
250	9
500	20
750	55
950	73

   Keystrokes: 1 5 0 [ENTER] 5 0 0 [ENTER] 8 0 0 [ENTER] 2 5 0 [ENTER] 5 0 0 [ENTER] 7 5 0 [ENTER] 9 5 0 [ENTER]

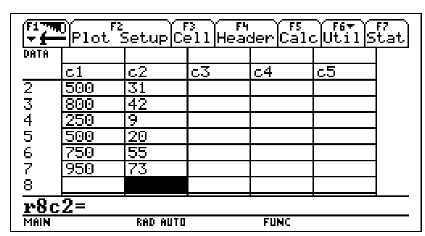


4. Move the cursor to row 1 in column 2 (r1c2). Then enter the corresponding number of buildings.

   [2nd] ⊙ moves the cursor up one page at a time.

   After typing data for a cell, you can press [ENTER] or ⊙ to enter the data and move the cursor down one cell. Pressing ⊙ enters the data and moves the cursor up one cell.

   Keystrokes: ⊙ [2nd] ⊙ 4 [ENTER] 3 1 [ENTER] 4 2 [ENTER] 9 [ENTER] 2 0 [ENTER] 5 5 [ENTER] 7 3 [ENTER]

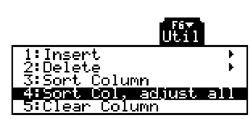


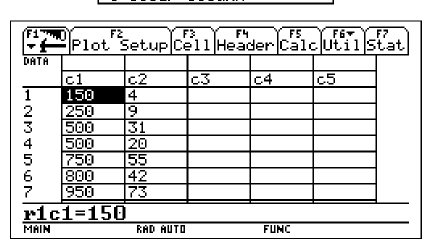
5. Move the cursor to row 1 in column 1 (r1c1). Sort the data in ascending order of population.

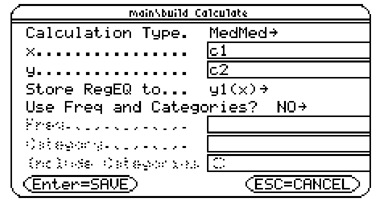
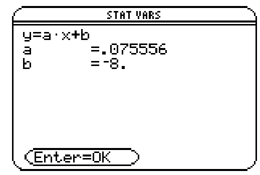
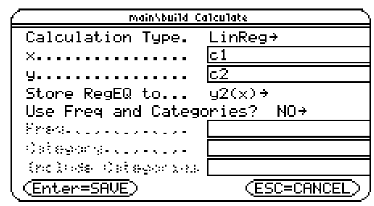
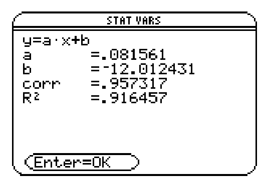
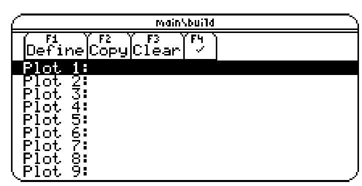
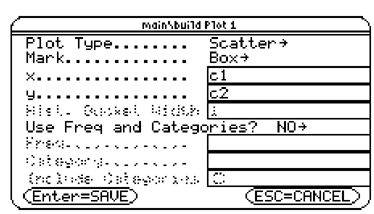
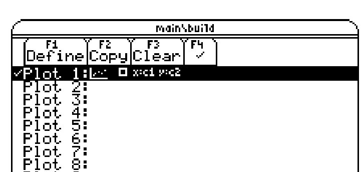
   This sorts column 1 and then adjusts all other columns so that they retain the same order as column 1. This is critical for maintaining the relationships between columns of data.

   To sort column 1, the cursor can be anywhere in column 1. This example has you press [2nd] ⊙ so that you can see all the data.

   Keystrokes: ⊙ [2nd] ⊙ [F6] 4





Steps	Keystrokes	Display
6. Display the Calculate dialog box. Set: Calculation Type = MedMed x = C1 y = C2 Store RegEQ to = y1(x)	[F5] → 7 ↓ C 1 ↓ C 2 ↓ → ↓ [ENTER]	
7. Perform the calculation to display the MedMed regression equation. *As specified on the* Calculate *dialog box, this equation is stored in y1(x).*	[ENTER]	
8. Close the STAT VARS screen.	[ENTER]	
9. Display the Calculate dialog box. Set: Calculation Type = LinReg x = C1 y = C2 Store RegEQ to = y2(x)	[F5] → 5 ↓ ↓ ↓ → ↓ [ENTER]	
10. Perform the calculation to display the LinReg regression equation. *This equation is stored in y2(x).*	[ENTER]	
11. Close the STAT VARS screen.	[ENTER]	
12. Display the Plot Setup screen. *Plot 1 is highlighted by default.*	[F2]	
13. Define Plot 1 as: Plot Type = Scatter Mark = Box x = C1 y = C2 *Notice the similarities between this and the* Calculate *dialog box.*	[F1] → 1 ↓ → 1 ↓ C 1 ↓ C 2	
14. Save the plot definition and return to the Plot Setup screen. *Notice the shorthand notation for Plot 1's definition.*	[ENTER] [ENTER]	

Steps | **Keystrokes** | **Display**

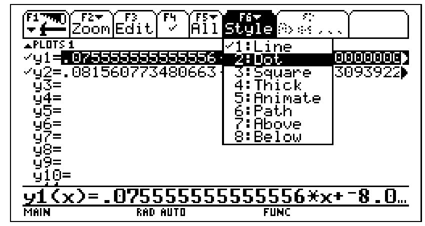
15. Display the Y= Editor. For y1(x), the MedMed regression equation, set the display style to Dot.

Note: Depending on the previous contents of your Y= Editor, you may need to move the cursor to y1.

PLOTS 1 at the top of the screen means that Plot 1 is selected.

Notice that y1(x) and y2(x) were selected when the regression equations were stored.

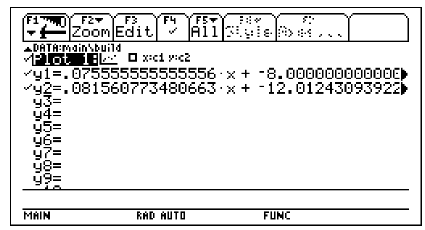
Keystrokes: [♦] [Y=] [F6] 2



16. Scroll up to highlight Plot 1.

The displayed shorthand definition is the same as on the Plot Setup screen.

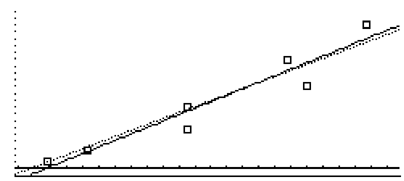
Keystrokes: [▲]



17. Use ZoomData to graph Plot 1 and the regression equations y1(x) and y2(x).

ZoomData examines the data for all selected stat plots and adjusts the viewing window to include all points.

Keystrokes: [F2] 9



18. Return to the current session of the Data/Matrix Editor.

Keystrokes: [APPS] 6 1

19. Enter a title for column 3. Define column 3's header as the values predicted by the MedMed line.

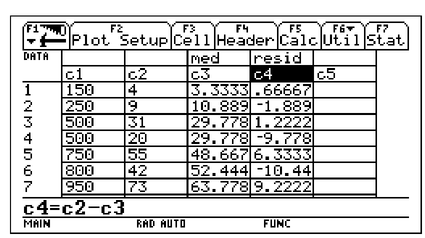
To enter a title, the cursor must highlight the title cell at the very top of the column.

[F4] lets you define a header from anywhere in a column. When the cursor is on a header cell, pressing [F4] is not required.

Keystrokes: [▶] [▶] [▲] [▲] M E D [ENTER] [F4] Y 1 [(] C 1 [)] [ENTER]

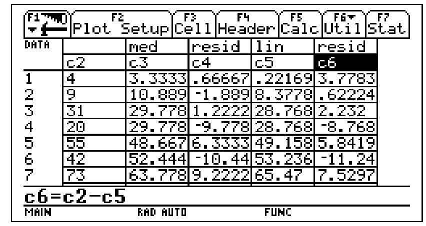
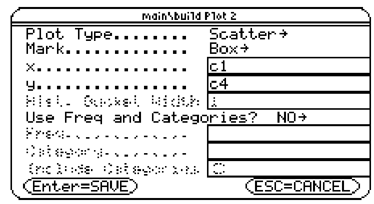
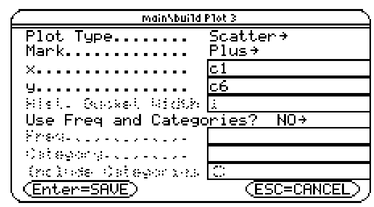
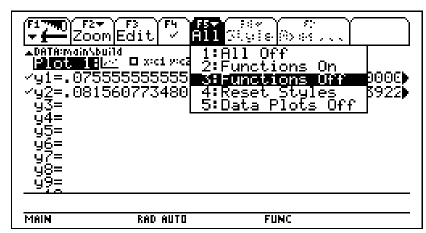
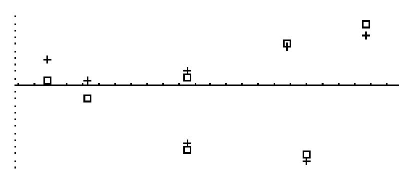
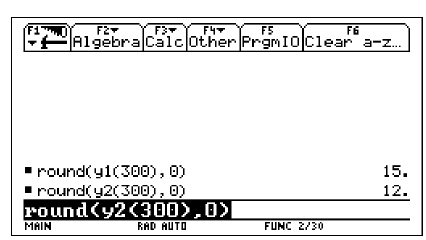
20. Enter a title for column 4. Define column 4's header as the residuals (difference between observed and predicted values) for MedMed.

Keystrokes: [▶] [▲] R E S I D [ENTER] [F4] C 2 [−] C 3 [ENTER]



21. Enter a title for column 5. Define column 5's header as the values predicted by the LinReg line.

Keystrokes: [▶] [▲] L I N [ENTER] [F4] Y 2 [(] C 1 [)] [ENTER]

Steps	Keystrokes	Display
22. Enter a title for column 6. Define column 6's header as the residuals for LinReg.	▶ ▲ R E S I D [ENTER] [F4] C 2 [−] C 5 [ENTER]	
23. Display the Plot Setup screen and deselect Plot 1.	[F2] [F4]	
24. Highlight Plot 2 and define it as: Plot Type = Scatter Mark = Box x = C1 y = C4 (MedMed residuals)	▼ [F1] ▼ ▼ C 1 ▼ C 4 [ENTER] [ENTER]	
25. Highlight Plot 3 and define it as: Plot Type = Scatter Mark = Plus x = C1 y = C6 (LinReg residuals)	▼ [F1] ▼ ▶ 3 ▼ C 1 ▼ C 6 [ENTER] [ENTER]	
26. Display the Y= Editor and turn all the y(x) functions off. *From* [F5]*, select* 3:Functions Off*, not* 1:All Off. *Plots 2 and 3 are still selected.*	[♦] [Y=] [F5] 3	
27. Use ZoomData to graph the residuals. *□ marks the MedMed residuals; + marks the LinReg residuals.*	[F2] 9	
28. Display the Home screen.	[♦] [HOME]	
29. Use the MedMed (y1(x)) and LinReg (y2(x)) regression equations to calculate values for x = 300 (300,000 population). *The* **round** *function* ([2nd] [MATH] 13) *ensures that results show an integer number of buildings.* *After calculating the first result, edit the entry line to change y1 to y2.*	[2nd] [MATH] 1 3 Y 1 [(] 3 0 0 [)] [,] 0 [)] [ENTER] ▲ ◀ ◀ ◀ ◀ ◀ ◀ ◀ ◀ [←] 2 [ENTER]	

Overview of Steps in Statistical Analysis

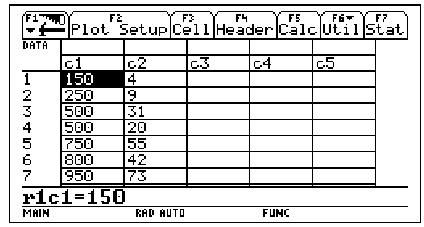
This section gives an overview of the steps used to perform a statistical calculation or graph a statistical plot. For detailed descriptions, refer to the following pages.

Calculating and Plotting Stat Data

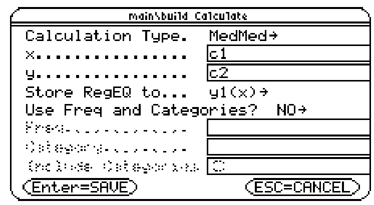
Note: Refer to Chapter 8 for details on entering data in the Data/Matrix Editor.

Set Graph mode ([MODE]) to FUNCTION.

Enter stat data in the Data/Matrix Editor ([APPS] 6).

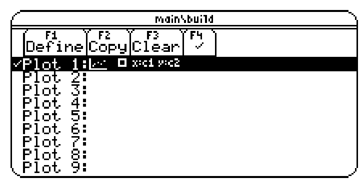


Perform stat calculations to find stat variables or fit data to a model ([F5]).



Tip: You can also use the Y= Editor to define and select stat plots and y(x) functions.

Define and select stat plots ([F2] and then [F1]).

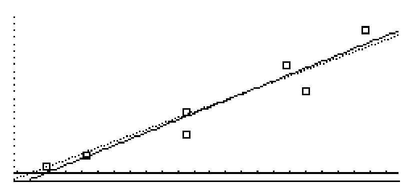


Tip: Use ZoomData to optimize the viewing window for stat plots. [F2] *Zoom is available on the Y= Editor, Window Editor, and Graph screen.*

Define the viewing window ([♦] [WINDOW]).

Change the graph format ([♦] F), if necessary.

Graph the selected stat plots and functions ([♦] [GRAPH]).



Exploring the Graphed Plots

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a plot.
- Use the [F2] Zoom toolbar menu to zoom in or out on a portion of the graph.
- Use the [F5] Math toolbar menu to analyze any function (but not plots) that may be graphed.

Performing a Statistical Calculation

From the Data/Matrix Editor, use the [F5] **Calc** toolbar menu to perform statistical calculations. You can analyze one-variable or two-variable statistics, or perform several types of regression analyses.

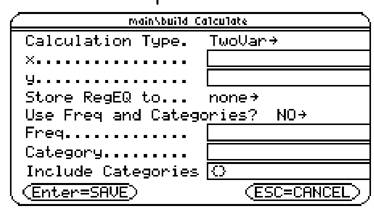
The Calculate Dialog Box

You must have a data variable opened. The Data/Matrix Editor will not perform statistical calculations with a list or matrix variable.

From the Data/Matrix Editor:

1. Press [F5] to display the Calculate dialog box.

   This example shows all items as active. On your calculator, items are active only if they are valid for the current settings of Calculation Type and Use Freq and Categories?



__Note:__ If an item is not valid for the current settings, it will appear dimmed. You cannot move the cursor to a dimmed item.

2. Specify applicable settings for the active items.

Item	Description
Calculation Type	Select the type of calculation. For descriptions, refer to page 195.
x	Type the column number in the Data/Matrix Editor (C1, C2, etc.) used for x values, the independent variable.
y	Type the column number used for y values, the dependent variable. This is required for all Calculation Types except OneVar.
Store RegEQ to	If Calculation Type is a regression analysis, you can select a function name (y1(x), y2(x), etc.). This lets you store the regression equation so that it will be displayed in the Y= Editor.
Use Freq and Categories?	Select NO or YES. Note that Freq, Category, and Include Categories are active only when Use Freq and Categories? = YES.

__Tip:__ To use an existing list variable for x, y, Freq, or Category, type the list name instead of a column number.

The Calculate Dialog Box (Continued)

Item	Description
Freq	Type the column number that contains a "weight" value for each data point. If you do not enter a column number, all data points are assumed to have the same weight (1).
Category	Type the column number that contains a category value for each data point.
Include Categories	If you specify a Category column, you can use this item to limit the calculation to specified category values. For example, if you specify {1,4}, the calculation uses only data points with a category value of 1 or 4.

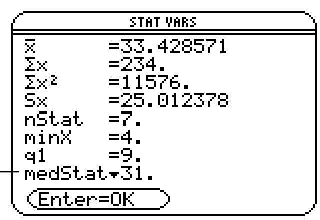
Note: *For an example of using Freq, Category, and Include Categories, refer to page 204.*

3. Press [ENTER] (after typing in an input box, press [ENTER] twice).

   The results are displayed on the STAT VARS screen. The format depends on the Calculation Type. For example:

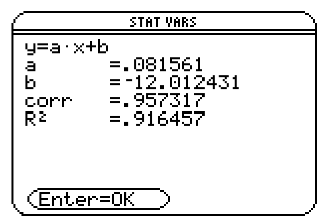
Note: *Any undefined data points (shown as* undef*) are ignored in a stat calculation.*

For Calculation Type = OneVar



When ▼ is shown instead of =, you can scroll for additional results.

For Calculation Type = LinReg



4. To close the STAT VARS screen, press [ENTER].

Redisplaying the STAT VARS Screen

The Data/Matrix Editor's [F7] Stat toolbar menu redisplays the previous calculation results (until they are cleared from memory).

Previous results are cleared when you:

- Edit the data points or change the Calculation Type.
- Open another data variable or reopen the same data variable (if the calculation referred to a column in a data variable). Results are also cleared if you leave and then reopen the Data/Matrix Editor with a data variable.
- Change the current folder (if the calculation referred to a list variable in the previous folder).

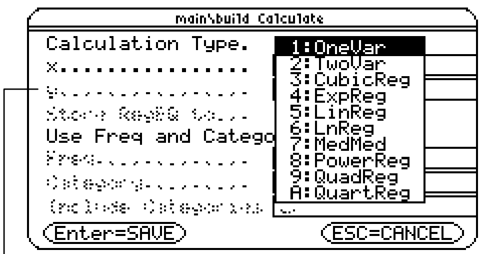
Statistical Calculation Types

As described in the previous section, the **Calculate** dialog box lets you specify the statistical calculation you want to perform. This section gives more information about the calculation types.

Selecting the Calculation Type

From the Calculate dialog box ([F5]), highlight the current setting for the Calculation Type and press ▶.

You can then select from a menu of available types.



If an item is dimmed, it is not valid for the current Calculation Type.

__Note:__ For TwoVar and all regression calculations, the columns that you specify for x and y (and optionally, Freq or Category) must have the same length.

Calc Type	Description
OneVar	One-variable statistics — Calculates the statistical variables described on page 197.
TwoVar	Two-variable statistics — Calculates the statistical variables described on page 197.
CubicReg	Cubic regression — Fits the data to the third-order polynomial $y=ax^3+bx^2+cx+d$. You must have at least four data points. • For four points, the equation is a polynomial fit. • For five or more points, it is a polynomial regression.
ExpReg	Exponential regression — Fits the data to the model equation $y=ab^x$ (where a is the y-intercept) using a least-squares fit and transformed values x and ln(y).
LinReg	Linear regression — Fits the data to the model $y=ax+b$ (where a is the slope, and b is the y-intercept) using a least-squares fit and x and y.
LnReg	Logarithmic regression — Fits the data to the model equation $y=a+b\ln(x)$ using a least-squares fit and transformed values ln(x) and y.

Selecting the Calculation Type (Continued)

Calc Type	Description
MedMed	Median-Median — Fits the data to the model y=ax+b (where a is the slope, and b is the y-intercept) using the median-median line, which is part of the resistant line technique. Summary points medx1, medy1, medx2, medy2, medx3, and medy3 are calculated and stored to variables, but they are not displayed on the STAT VARS screen.
PowerReg	Power regression — Fits the data to the model equation $y=ax^b$ using a least-squares fit and transformed values ln(x) and ln(y).
QuadReg	Quadratic regression — Fits the data to the second-order polynomial $y=ax^2+bx+c$. You must have at least three data points. • For three points, the equation is a polynomial fit. • For four or more points, it is a polynomial regression.
QuartReg	Quartic regression — Fits the data to the fourth-order polynomial $y=ax^4+bx^3+cx^2+dx+e$. You must have at least five data points. • For five points, the equation is a polynomial fit. • For six or more points, it is a polynomial regression.

From the Home Screen or a Program

Use the applicable command for the calculation that you want to perform. The commands have the same name as the corresponding Calculation Type. Refer to Appendix A for information about each command.

Important: These commands perform a stat calculation but do not automatically display the results. Use the **ShowStat** command to show the calculation results.

Statistical Variables

Statistical calculation results are stored to variables. To access these variables, type the variable name or use the VAR-LINK screen as described in Chapter 18. All statistical variables are cleared when you edit the data or change the calculation type. Other conditions that clear the variables are listed on page 194.

Calculated Variables

Stat variables are stored as system variables. However, regCoef and regeq are treated as a list and a function variable, respectively.

Tip: From the keyboard, press [2nd] G [↑] S for Σ and [2nd] G S for σ.

Tip: To type a power (such as 2 in Σx^2), $\bar{x}$, or $\bar{y}$, press [2nd] [CHAR] and select it from the Math menu.

	One Var	Two Var	Regressions
mean of x values	$\bar{x}$	$\bar{x}$	
sum of x values	Σx	Σx	
sum of x^2 values	Σx^2	Σx^2	
sample std. deviation of x	Sx	Sx	
population std. deviation of x †	σx	σx	
number of data points	nStat	nStat	
mean of y values		$\bar{y}$	
sum of y values		Σy	
sum of y^2 values		Σy^2	
sample standard deviation of y		Sy	
population std. deviation of y †		σy	
sum of x∗y values		Σxy	
minimum of x values	minX	minX	
maximum of x values	maxX	maxX	
minimum of y values		minY	
maximum of y values		maxY	
1st quartile	q1		
median	medStat		
3rd quartile	q3		
regression equation			regeq
regression coefficients (a, b, c, d, e)			regCoef
correlation coefficient ††			corr
coefficient of determination ††			R^2
summary points (MedMed only) †			medx1, medy1, medx2, medy2, medx3, medy3

Note: 1st quartile is the median of points between minX and medStat, and 3rd quartile is the median of points between medStat and maxX.

Tip: If regeq is 4x + 7, then regCoef is {4 7}. To access the "a" coefficient (the 1st element in the list), use an index such as regCoef[1].

† The indicated variables are calculated but are not shown on the STAT VARS screen.

†† corr is defined for a linear regression only; R^2 is defined for all polynomial regressions.

Defining a Statistical Plot

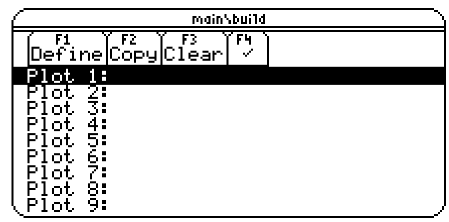
From the Data/Matrix Editor, you can use the entered data to define several types of statistical plots. You can define up to nine plots at a time.

Procedure

From the Data/Matrix Editor:

1. Press [F2] to display the Plot Setup screen.

   Initially, none of the plots are defined.



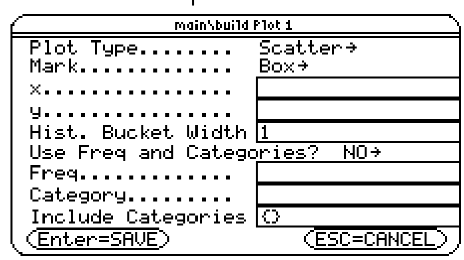
2. Move the cursor to highlight the plot number that you want to define.

Pathname of the data variable

Note: This dialog box is similar to the Calculate *dialog box.*

3. Press [F1] to define the plot.

   This example shows all items as active. On your calculator, items are active only if they are valid for the current setting of Plot Type and Use Freq and Categories?



***Note:** If an item is not valid for the current settings, it will appear dimmed. You cannot move the cursor to a dimmed item.*

4. Specify applicable settings for the active items.

Item	Description
Plot Type	Select the type of plot. For descriptions, refer to page 200.
Mark	Select the symbol used to plot the data points: Box (□), Cross (x), Plus (+), Square (■), or Dot (•).
x	Type the column number in the Data/Matrix Editor (C1, C2, etc.) used for x values, the independent variable.
y	Type the column number used for y values, the dependent variable. This is active only for Plot Type = Scatter or xyline.
Hist. Bucket Width	Specifies the width of each bar in a histogram. For more information, refer to page 201.
Use Freq and Categories?	Select NO or YES. Note that Freq, Category, and Include Categories are active only when Use Freq and Categories? = YES. (Freq is active only for Plot Type = Box Plot or Histogram.)

***Note:** Plots defined with column numbers always use the last data variable in the Data/Matrix Editor, even if that variable was not used to create the definition.*

***Tip:** To use an existing list variable for x, y, Freq, or Category, type the list name instead of the column number.*

Note: For an example of using Freq, Category, and Include Categories, refer to page 204.

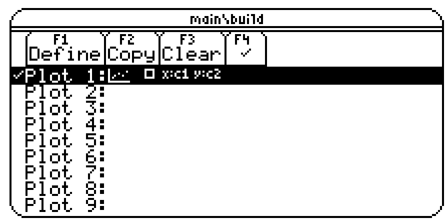
Item	Description
Freq	Type the column number that contains a “weight” value for each data point. If you do not enter a column number, all data points are assumed to have the same weight (1).
Category	Type the column number that contains a category value for each data point.
Include Categories	If you specify a Category, you can use this to limit the calculation to specified category values. For example, if you specify {1,4}, the plot uses only data points with a category value of 1 or 4.

5. Press [ENTER] (after typing in an input box, press [ENTER] twice).

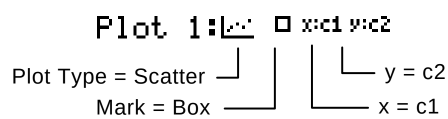
Note: Any undefined data points (shown as undef) are ignored in a stat plot.

The Plot Setup screen is redisplayed.

The plot you just defined is automatically selected for graphing.



Notice the shorthand definition for the plot.



Selecting or Deselecting a Plot

From Plot Setup, highlight the plot and press [F4] to toggle it on or off. If a stat plot is selected, it remains selected when you:

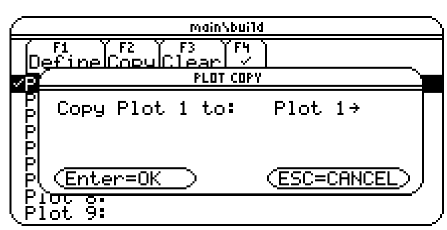
- Change the graph mode. (Stat plots are not graphed in 3D mode.)
- Execute a **Graph** command.
- Open a different variable in the Data/Matrix Editor.

Copying a Plot Definition

From Plot Setup:

1. Highlight the plot and press [F2].
2. Press ⓑ and select the plot number that you want to copy to.
3. Press [ENTER].

Note: If the original plot was selected (✓), the copy is also selected.



Clearing a Plot Definition

From Plot Setup, highlight the plot and press [F3]. To redefine an existing plot, you do not necessarily need to clear it first; you can make changes to the existing definition. To prevent a plot from graphing, you can deselect it.

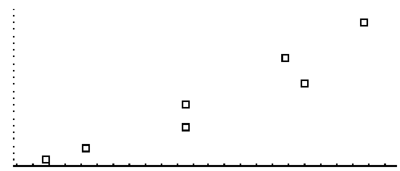
Statistical Plot Types

When you define a plot as described in the previous section, the **Plot Setup** screen lets you select the plot type. This section gives more information about the available plot types.

Scatter

Data points from x and y are plotted as coordinate pairs. Therefore, the columns or lists that you specify for x and y must be the same length.

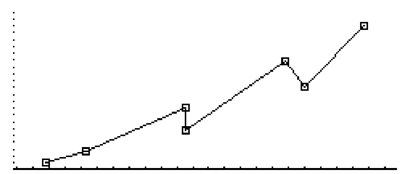
- Plotted points are shown with the symbol that you select as the Mark.
- If necessary, you can specify the same column or list for both x and y.



xyline

This is a scatter plot in which data points are plotted and connected in the order in which they appear in x and y.

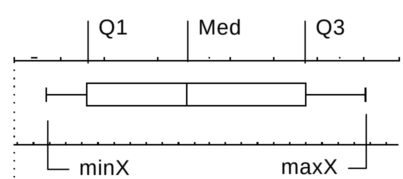
You may want to sort all the columns (F6 3 or F6 4 in the Data/Matrix Editor) before plotting.



Box Plot

This plots one-variable data with respect to the minimum and maximum data points (minX and maxX) in the set.

- A box is defined by its first quartile (Q1), median (Med), and third quartile (Q3).
- Whiskers extend from minX to Q1 and from Q3 to maxX.
- When you select multiple box plots, they are plotted one above the other in the same order as their plot numbers.

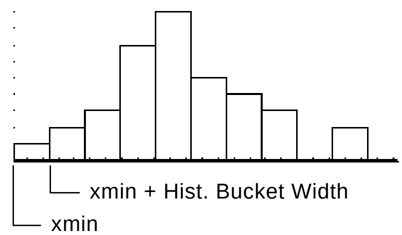


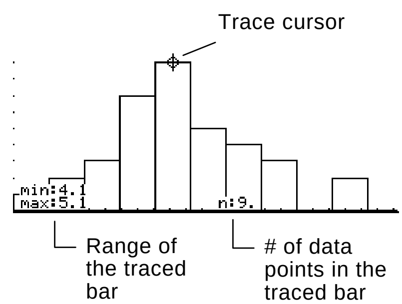
Histogram

This plots one-variable data as a histogram. The x axis is divided into equal widths called buckets or bars. The height of each bar (its y value) indicates how many data points fall within the bar's range.

$$\text{Number of bars} = \frac{\text{xmax} - \text{xmin}}{\text{Hist. Bucket Width}}$$

- When defining the plot, you can specify the Hist. Bucket Width (default is 1) to set the width of each bar.
- A data point at the edge of a bar is counted in the bar to the right.
- ZoomData ([F2] 9 from the Graph screen, Y= Editor, or Window Editor) adjusts xmin and xmax to include all data points, but it does not adjust the y axis.
  – Use [◆] [WINDOW] to set ymin = 0 and ymax = the number of data points expected in the tallest bar.
- When you trace ([F3]) a histogram, the screen shows information about the traced bar.





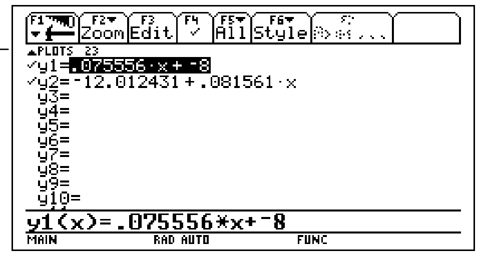
Using the Y= Editor with Stat Plots

The previous sections described how to define and select stat plots from the Data/Matrix Editor. You can also define and select stat plots from the Y= Editor.

Showing the List of Stat Plots

Press [♦] [Y=] to display the Y= Editor. Initially, the nine stat plots are located “off the top” of the screen, above the y(x) functions. However, the PLOTS indicator provides some information.

For example, PLOTS 23 means that Plots 2 & 3 are selected.

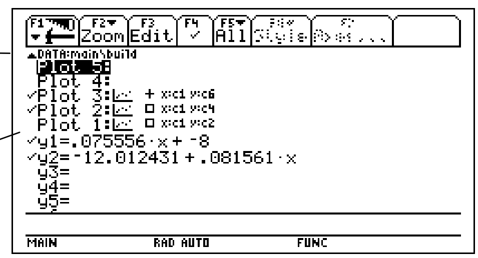


To see the list of stat plots, use ⓒ to scroll above the y(x) functions.

Note: Plots defined with column numbers always use the last data variable in the Data/Matrix Editor, even if that variable was not used to create the definition.

If a Plot is highlighted, this shows the data variable that will be used for the plots.

If a Plot is defined, it shows the same shorthand notation as the Plot Setup screen.



From the Y= Editor, you can perform most of the same operations on a stat plot as you can on any other y(x) function.

Note: You cannot use [F6] to set a plot’s display style. However, the plot definition lets you select the mark used for the plot.

To:	Do this:
Edit a plot definition	Highlight the plot and press [F3]. You will see the same definition screen that is displayed in the Data/Matrix Editor.
Select or deselect a plot	Highlight the plot and press [F4].
Turn all plots and/or functions off	Press [F5] and select the applicable item. You can also use this menu to turn all functions on.

To Graph Plots and Y= Functions

As necessary, you can select and graph stat plots and y(x) functions at the same time. The preview example at the beginning of this chapter graphs data points and their regression equations.

Graphing and Tracing a Defined Stat Plot

After entering the data points and defining the stat plots, you can graph the selected plots by using the same methods you used to graph a function from the Y= Editor (as described in Chapter 3).

Defining the Viewing Window

Stat plots are displayed on the current graph, and they use the Window variables that are defined in the Window Editor.

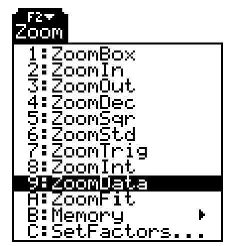
Use [♦] [WINDOW] to display the Window Editor. You can either:

- Enter appropriate values.
  — or —
- Select 9:ZoomData from the [F2] Zoom toolbar menu. (Although you can use any zoom, ZoomData is optimized for stat plots.)

__Tip:__ [F2] Zoom is available on the Y= Editor, Window Editor, and Graph screen.

ZoomData sets the viewing window to display all statistical data points.

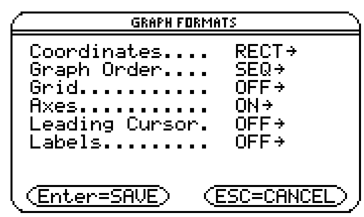
For histograms and box plots, only xmin and xmax are adjusted. If the top of a histogram is not shown, trace the histogram to find the value for ymax.



Changing the Graph Format

Press [♦] F (or [F1] 9) from the Y= Editor, Window Editor, or Graph screen.

Then change the settings as necessary.



Tracing a Stat Plot

From the Graph screen, press [F3] to trace a plot. The movement of the trace cursor depends on the Plot Type.

__Note:__ When a stat plot is displayed, the Graph screen does not automatically pan if you trace off the left or right side of the screen. However, you can still press [ENTER] to center the screen on the trace cursor.

Plot Type	Description
Scatter or xyline	Tracing begins at the first data point.
Box plot	Tracing begins at the median. Press ◀ to trace to Q1 and minX. Press ▶ to trace to Q3 and maxX.
Histogram	The cursor moves from the top center of each bar, starting from the leftmost bar.

When you press ▲ or ▼ to move to another plot or y(x) function, tracing moves to the current or beginning point on that plot (not to the nearest pixel).

Using Frequencies and Categories

To manipulate the way in which data points are analyzed, you can use frequency values and/or category values. Frequency values let you “weight” particular data points. Category values let you analyze a subset of the data points.

Example of a Frequency Column

In a data variable, you can use any column in the Data/Matrix Editor to specify a frequency value (or weight) for the data points on each row. A frequency value must be an integer ≥ 0 if Calculation Type = OneVar or MedMed or if Plot Type = Box Plot. For other stat calculations or plots, the frequency value can be any number ≥ 0.

For example, suppose you enter a student’s test scores, where:

- The mid-semester exam is weighted twice as much as other tests.
- The final exam is weighted three times as much.

In the Data/Matrix Editor, you can enter the test scores and frequency values in two columns.

Tip: A frequency value of 0 effectively removes the data point from analysis.

Test scores (c1) — Frequency values (c2)

c1	c2
85	1
97	1
92	2
89	1
91	1
95	3

These weighted scores are equivalent to the single column of scores listed to the right.

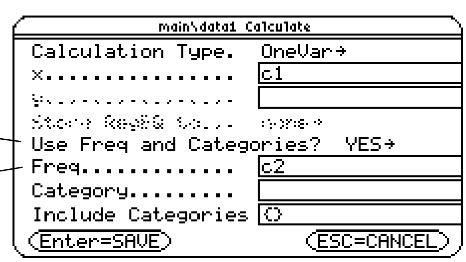
c1	
85	
97	
92	Frequency of 2
92	
89	
91	
95	Frequency of 3
95	
95	

Note: You can also use frequency values from a list variable instead of a column.

To use frequency values, specify the frequency column when you perform a stat calculation or define a stat plot. For example:

Set this to YES.

Type the column number (or list name) that contains the frequency values.



Example of a Category Column

In a data variable, you can use any column to specify a category (or subset) value for the data points on each row. A category value can be any number.

Suppose you enter the test scores from a class that has 10th and 11th grade students. You want to analyze the scores for the whole class, but you also want to analyze categories such as 10th grade girls, 10th grade boys, 10th grade girls and boys, etc.

First, determine the category values you want to use.

***Note:** You do not need a category value for the whole class. Also, you do not need category values for all 10th graders or all 11th graders since they are combinations of other categories.*

Category Value	Used to indicate:
1	10th grade girl
2	10th grade boy
3	11th grade girl
4	11th grade boy

In the Data/Matrix Editor, you can enter the scores and the category values in two columns.

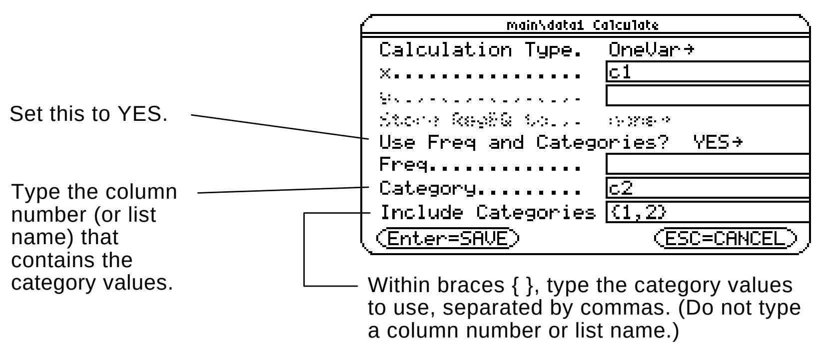
Test scores

Category values

c1	c2
85	1
97	3
92	2
88	3
90	2
95	1
79	4
68	2
92	4
84	3
82	1

***Note:** You can also use category values from a list variable instead of a column.*

To use category values, specify the category column and the category values to include in the analysis when you perform a stat calculation or define a stat plot.



***Note:** To analyze the whole class, leave the Category input box blank. Any category values are ignored.*

To analyze:	Include Categories:
10th grade girls	{1}
10th grade boys	{2}
10th grade girls and boys	{1,2}
11th grade girls	{3}
11th grade boys	{4}
11th grade girls and boys	{3,4}
all girls (10th and 11th)	{1,3}
all boys (10th and 11th)	{2,4}

If You Have a CBL 2/CBL or CBR

The Calculator-Based Laboratory™ System (CBL 2™, CBL™) and Calculator-Based Ranger™ System (CBR™) are optional accessories, available separately, that let you collect data from a variety of real-world experiments.

How CBL 2/CBL Data Is Stored

When you collect data with the CBL 2/CBL, that data is initially stored in the CBL 2/CBL unit itself. You must then retrieve the data (transfer it to the TI-92) by using the **Get** command, which is described in Appendix A.

Although each set of retrieved data can be stored in several variable types (list, real, matrix, pic), using list variables makes it easier to perform stat calculations.

Note: *For specifics about using the CBL 2/CBL and retrieving data to the TI-92, refer to the guidebook that comes with the CBL 2/CBL unit.*

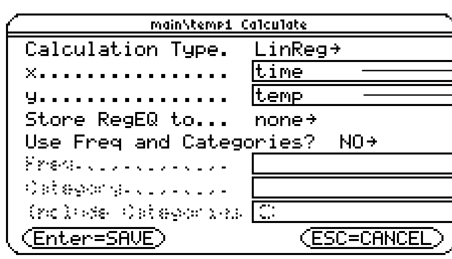
When you transfer the collected information to the TI-92, you can specify the list variable names that you want to use. For example, you can use the CBL 2/CBL to collect temperature data over a period of time. When you transfer the data, suppose you store:

- Temperature data in a list variable called temp.
- Time data in a list variable called time.

After you store the CBL 2/CBL information on the TI-92, there are two ways to use the CBL 2/CBL list variables.

Referring to the CBL 2/CBL Lists

When you perform a stat calculation or define a plot, you can refer explicitly to the CBL 2/CBL list variables. For example:



Type the CBL list variable name instead of a column number.

Creating a Data Variable with the CBL 2/CBL Lists

You can create a new data variable that consists of the necessary CBL 2/CBL list variables.

- From the Home screen or a program, use the **NewData** command.

  NewData *dataVar*, *list1* [,*list2*] [,*list3*] ...

  CBL list variable names. In the new data variable, list1 will be copied to column 1, list 2 to column 2, etc.

  Name of the new data variable that you want to create.

  For example:

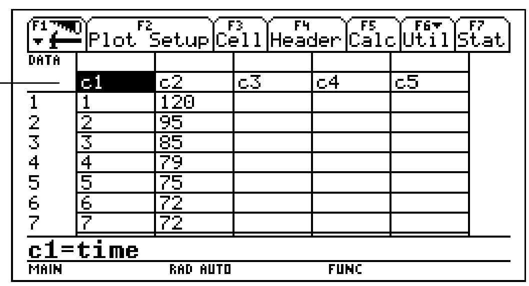
  NewData temp1, time, temp

  creates a data variable called temp1 in which time is in column 1 and temp is in column 2.

Tip: *To define or clear a column header, use* [F4]*. For more information, refer to Chapter 8.*

- From the Data/Matrix Editor, create a new, empty data variable with the applicable name. For each CBL 2/CBL list that you want to include, define a column header as that list name.

For example, define column 1 as time, column 2 as temp.



At this point, the columns are linked to the CBL 2/CBL lists. If the lists are changed, the columns will be updated automatically. However, if the lists are deleted, the data will be lost.

To make the data variable independent of the CBL 2/CBL lists, clear the column header for each column. The information remains in the column, but the column is no longer linked to the CBL list.

CBR

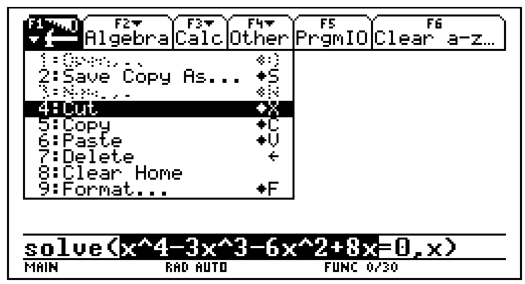
See *Getting Started with CBR*™ for more information.

Additional Home Screen Topics

10

To help you get started using the TI-92 as quickly as possible, Chapter 2 described the basic operations of the Home screen.

This chapter describes additional operations that can help you use the Home screen more effectively.



Because this chapter consists of various stand-alone topics, it does not start with a "preview" example.

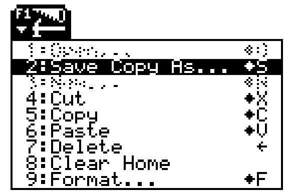
Saving the Home Screen Entries as a Text Editor Script

To save all the entries in the history area, you can save the Home screen to a text variable. When you want to reexecute those entries, use the Text Editor to open the variable as a command script.

Saving the Entries in the History Area

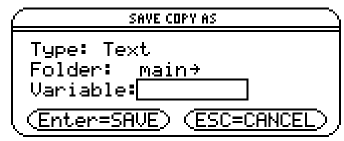
From the Home screen:

1. Press [F1] and select 2:Save Copy As. (You can press [♦] S instead of using [F1].)



Note: Only the entries are saved, not the answers.

2. Specify a folder and text variable that you want to use to store the entries.



Note: For information about folders, refer to page 216.

Item	Description
Type	Automatically set as Text and cannot be changed.
Folder	Shows the folder in which the text variable will be stored. To use a different folder, press ⊙ to display a menu of existing folders. Then select a folder.
Variable	Type a valid, unused variable name.

3. Press [ENTER] (after typing in an input box such as Variable, press [ENTER] twice).

Restoring the Saved Entries

Because the entries are stored in a script format, you cannot restore them from the Home screen. (On the Home screen's [F1] toolbar menu, 1:Open is not available.) Instead:

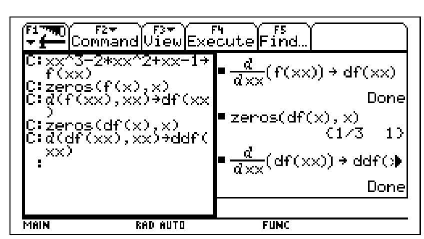
Note: For complete information on using the Text Editor and executing a command script, refer to Chapter 16.

1. Use the Text Editor to open the variable containing the saved Home screen entries.

   The saved entries are listed as a series of command lines that you can execute individually, in any order.

2. Starting with the cursor on the first line of the script, press [F4] repeatedly to execute the commands line by line.

3. Display the restored Home screen.



This split screen shows the Text Editor (with the command line script) and the restored Home screen.

Cutting, Copying, and Pasting Information

Cut, copy, and paste operations let you move or copy information within the same application or between different applications. These operations use the TI-92's clipboard, which is an area in memory that serves as a temporary storage location.

Auto-paste vs. Cut/Copy/Paste

Auto-paste, described in Chapter 2, is a quick way to copy an entry or answer in the history area and paste it to the entry line.

1. Use ⓒ and ⓓ to highlight the item in the history area.
2. Press [ENTER] to auto-paste that item to the entry line.

To copy or move information in the entry line, you must use a cut, copy, or paste operation. (You can perform a copy operation in the history area, but not a cut or paste.)

Cutting or Copying Information to the Clipboard

When you cut or copy information, that information is placed in the clipboard. However, cutting deletes the information from its current location (used to move information) and copying leaves the information.

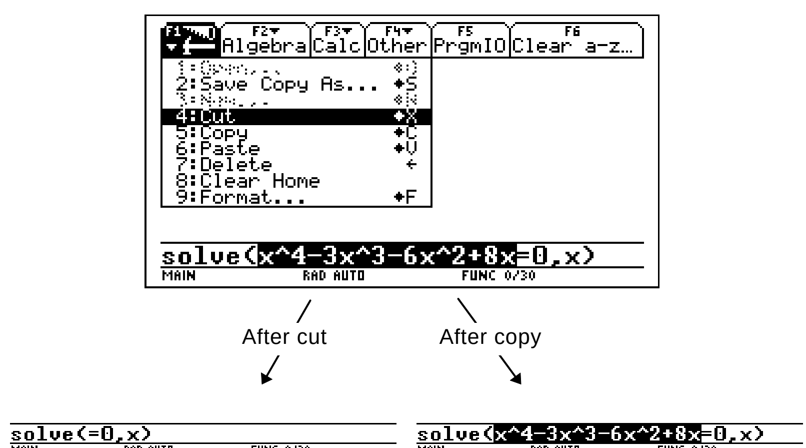
1. Highlight the characters that you want to cut or copy.

   In the entry line, move the cursor to either side of the characters. Hold [⇑] and press ⓐ or ⓑ to highlight characters to the left or right of the cursor, respectively.

Tip: You can press [♦] X, [♦] C, *or* [♦] V *to cut, copy or paste, respectively, without having to use the* [F1] *toolbar menu.*

2. Press [F1] and select 4:Cut or 5:Copy.

Clipboard = (empty or the previous contents)



Clipboard = x^4−3x^3−6x^2+8x   Clipboard = x^4−3x^3−6x^2+8x

Note: When you cut or copy information, it replaces the clipboard's previous contents, if any.

Cutting is not the same as deleting. When you delete information, it is not placed in the clipboard and cannot be retrieved.

Pasting Information from the Clipboard

A paste operation inserts the contents of the clipboard at the current cursor location on the entry line. This does not change the contents of the clipboard.

1. Position the cursor where you want to paste the information.
2. Press [F1] and select 6:Paste (or use the [♦] V shortcut).

Example: Copying and Pasting

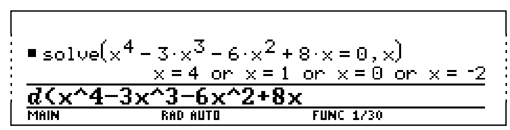
Suppose you want to reuse an expression without retyping it each time.

1. Copy the applicable information.
   a. Use [↑] ▷ or [↑] ◁ to highlight the expression.

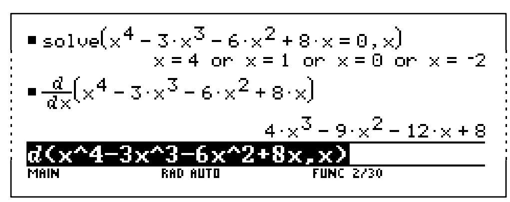
   b. Press [♦] C.
   c. For this example, press [ENTER] to evaluate the entry.

Tip: You can also reuse an expression by creating a user-defined function. Refer to page 213.

2. Paste the copied information into a new entry.
   a. Press [F3] 1 to select the *d* differentiate function.
   b. Press [♦] V to paste the copied expression.

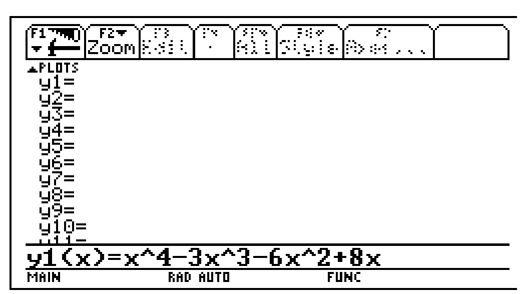


   c. Complete the new entry, and press [ENTER].



Tip: By copying and pasting, you can easily transfer information from one application to another.

3. Paste the copied information into a different application.
   a. Press [♦] [Y=] to display the Y= Editor.
   b. Press [ENTER] to define y1(x).
   c. Press [♦] V to paste.
   d. Press [ENTER] to save the new definition.



Creating and Evaluating User-Defined Functions

User-defined functions can be a great time-saver when you need to repeat the same expression (but with different values) multiple times. User-defined functions can also extend your TI-92's capabilities beyond the built-in functions.

Format of a Function

The following examples show user-defined functions with one argument and two arguments. You can use as many arguments as necessary. In these examples, the definition consists of a single expression (or statement).

Note: Function names follow the same rules as variable names. Refer to "Storing and Recalling Variable Values" in Chapter 2.

```
cube(x) = x³                    xroot(x,y) = y^(1/x)
│    │    └ Definition          │     │      └ Definition
│    └ Argument list            │     └ Argument list
└ Function name                 └ Function name
```

Tip: Use two or more character argument names (xx,yy,xtemp,...) to define function or program arguments to prevent circular definitions when calling the function or program.

When defining functions and programs, use unique names for arguments that will not be used in the arguments for a subsequent function or program call.

In the argument list, be sure to use the same arguments that are used in the definition. For example, cube(n) = x^3 gives unexpected results when you evaluate the function.

Arguments (x and y in these examples) are placeholders that represent whatever values you pass to the function. They do not represent the variables x and y unless you specifically pass x and y as the arguments when you evaluate the function.

Creating a User-Defined Function

Use one of the following methods.

Method	Description
[STO▶]	Store an expression to a function name (including the argument list). 
Define command	Define a function name (including the argument list) as an expression. 
Program Editor	Refer to Chapter 17 for information on creating a user-defined function.

Creating a Multi-Statement Function

Note: *For information about similarities and differences between functions and programs, refer to Chapter 17.*

You can also create a user-defined function whose definition consists of multiple statements. The definition can include many of the control and decision-making structures (**If**, **ElseIf**, **Return**, etc.) used in programming.

For example, suppose you want to create a function that sums a series of reciprocals based on an entered integer (n):

$$\frac{1}{n}+\frac{1}{n-1}+\ldots+\frac{1}{1}$$

When creating the definition of a multi-statement function, it may be helpful to visualize it first in a block form.

```
Func
Local temp,i
If fPart(nn)≠0 or nn≤0
  Return “bad argument”
0→temp
For i,nn,1,-1
  approx(temp+1/i)→temp
EndFor
Return temp
EndFunc
```

- Variables not in the argument list must be declared as local. (Local temp,i)
- Returns a message if nn is not an integer or if nn≤0. (Return “bad argument”)
- Sums the reciprocals. (approx(temp+1/i)→temp)
- Returns the sum. (Return temp)
- **Func** and **EndFunc** must begin and end the function.
- For information about the individual statements, refer to Appendix A.

When entering a multi-statement function on the Home screen, you must enter the entire function on a single line. Use the **Define** command just as you would for a single-statement function.

Use argument names that will never be used when calling the function or program.

Use a colon to separate each statement.

```
Define sumrecip(nn)=Func:Local temp,i: ... :EndFunc
```

Tip: *It's easier to create a complicated multi-statement function in the Program Editor than on the Home screen. Refer to Chapter 17.*

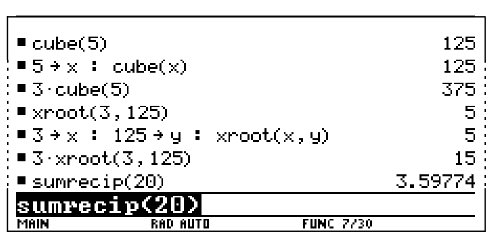
On the Home screen:

Multi-statement functions show as “Func”.

Enter a multi-statement function on one line. Be sure to include colons.

Evaluating a Function

You can use a user-defined function just as you would any other function. Evaluate it by itself or include it in another expression.



Displaying and Editing a Function Definition

To:	Do this:
Display a list of all user-defined functions	Press [2nd] [VAR-LINK] to display the VAR-LINK screen. (Refer to Chapter 18.) You may need to use the [F2] View toolbar menu to specify the Function variable type.
Display the definition of a user-defined function	From the VAR-LINK screen, highlight the function and press [F6] Contents. — or — From the Home screen, press [2nd] [RCL]. Type the function name but not the argument list (such as xroot), and press [ENTER] twice. — or — From the Program Editor, open the function. (Refer to Chapter 17.)
Edit the definition	From the Home screen, use [2nd] [RCL] to display the definition. Edit the definition as necessary. Then use [STO▸] or **Define** to save the new definition. — or — From the Program Editor, open the function, edit it, and save your changes. (Refer to Chapter 17.)

Using Folders to Store Independent Sets of Variables

The TI-92 has one built-in folder named MAIN, and all variables are stored in that folder. By creating additional folders, you can store independent sets of user-defined variables (including user-defined functions).

Folders and Variables

Folders give you a convenient way to manage variables by organizing them into related groups. For example, you can create separate folders for different TI-92 applications (Geometry, Text Editor, etc.) or classes.

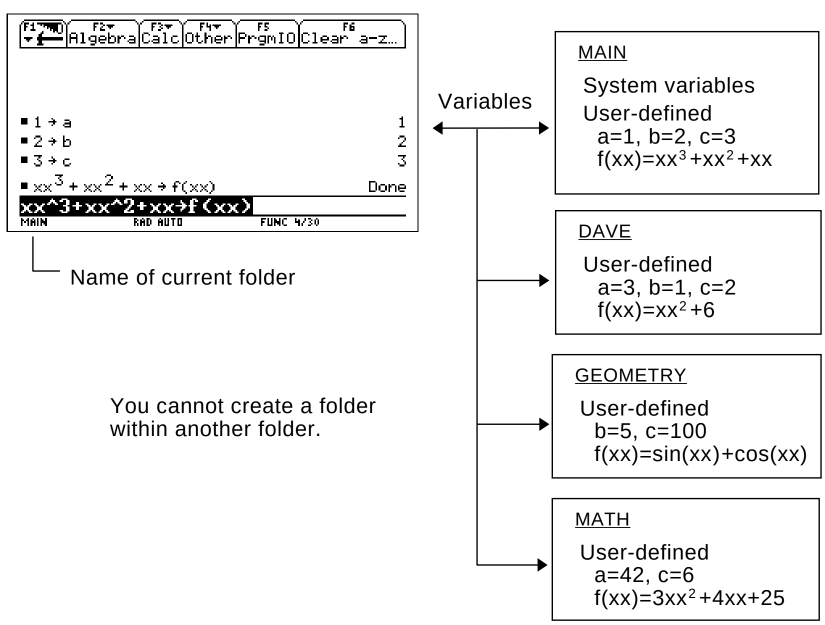
- You can store a user-defined variable in any existing folder.
- A system variable or a variable with a reserved name, however, can be stored in the MAIN folder only.

Example of variables that can be stored in MAIN only
Window variables (xmin, xmax, etc.)
Table setup variables (TblStart, ∆Tbl, etc.)
Y= Editor functions (y1(x), etc.)

The user-defined variables in one folder are independent of the variables in any other folder.

Therefore, folders can store separate sets of variables with the same names but different values.

Note: User-defined variables are stored in the "current folder" unless you specify otherwise. Refer to "Using Variables in Different Folders" on page 218.



The system variables in the MAIN folder are always directly accessible, regardless of the current folder.

Creating a Folder from the Home Screen

Enter the **NewFold** command.

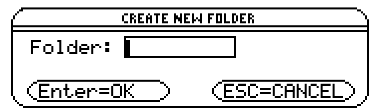
NewFold *folderName*

└── Folder name to create. This new folder is set automatically as the current folder.

Creating a Folder from the VAR-LINK Screen

The VAR-LINK screen, which is described in Chapter 18, lists the existing variables and folders.

1. Press [2nd] [VAR-LINK].
2. Press [F1] Manage and select 5:Create Folder.
3. Type a unique folder name, and press [ENTER] twice.



After you create a new folder from VAR-LINK, that folder is *not* automatically set as the current folder.

Setting the Current Folder from the Home Screen

Enter the **setFold** function.

setFold (*folderName)*

└── **setFold** is a function, which requires you to enclose the folder name in parentheses.

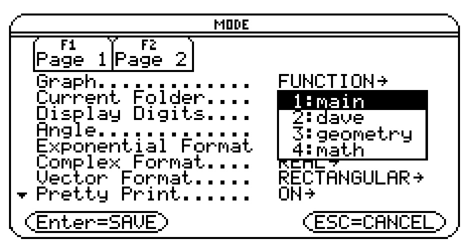
When you execute **setFold**, it returns the name of the folder that was previously set as the current folder.

Setting the Current Folder from the MODE Dialog Box

__Tip:__ To cancel the menu or exit the dialog box without saving any changes, press [ESC].

To use the MODE dialog box:

1. Press [MODE].
2. Highlight the Current Folder setting.
3. Press ⊙ to display a menu of existing folders.
4. Select the applicable folder. Either:



   - Highlight the folder name and press [ENTER].
     — or —
   - Press the corresponding number or letter for that folder.
5. Press [ENTER] to save your changes and close the dialog box.

Using Variables in Different Folders

You can access a user-defined variable or function that is not in the current folder. Specify the complete *pathname* instead of only the variable name.

A pathname has the form:

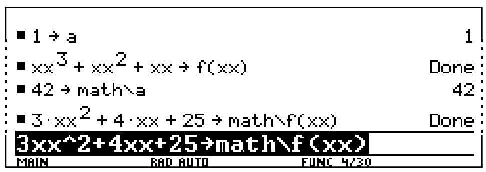
Tip: For "\", press [2nd] [\] (2nd function of [÷]).

folderName\variableName
— or —
folderName\functionName

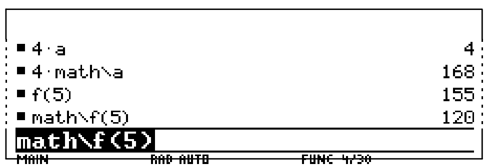
For example:

If Current Folder = MAIN | **Folders**

Note: This example assumes that you have already created a folder named MATH.



MAIN
a=1
f(xx)=xx³+xx²+xx



MATH
a=42
f(xx)=3xx²+4xx+25

Note: For information about the VAR-LINK screen, refer to Chapter 18.

To see a list of existing folders and variables, press [2nd] [VAR-LINK]. On the VAR-LINK screen, you can highlight a variable and press [ENTER] to paste that variable name to the Home screen's entry line. If you paste a variable name that is not in the current folder, the pathname (*folderName\variableName*) is pasted.

Deleting a Folder from the Home Screen

Before deleting a folder, you must delete all the variables stored in that folder.

- To delete a variable, enter the **DelVar** command.

  DelVar *var1* [, *var2*] [, *var3*] ...

Note: You cannot delete the MAIN folder.

- To delete an empty folder, enter the **DelFold** command.

  DelFold *folder1* [, *folder2*] [, *folder3*] ...

Deleting a Folder from the VAR-LINK Screen

VAR-LINK lets you delete a folder and its variables at the same time. Refer to Chapter 18.

1. Press [2nd] [VAR-LINK].
2. Select the item(s) to delete and press [F1] 1 or [←]. (If you use [F4] to select a folder, its variables are selected automatically.)
3. Press [ENTER] to confirm the deletion.

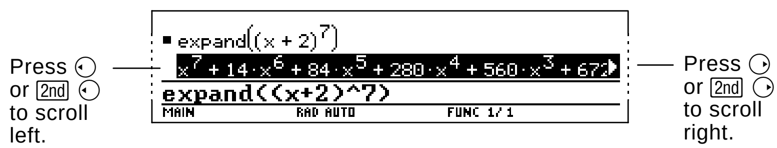
If an Entry or Answer Is “Too Big”

In some cases, an entry or answer may be “too long” and/or “too tall” to be displayed completely in the history area. In other cases, the TI-92 may not be able to display an answer because there is not enough free memory.

If an Entry or Answer Is “Too Long”

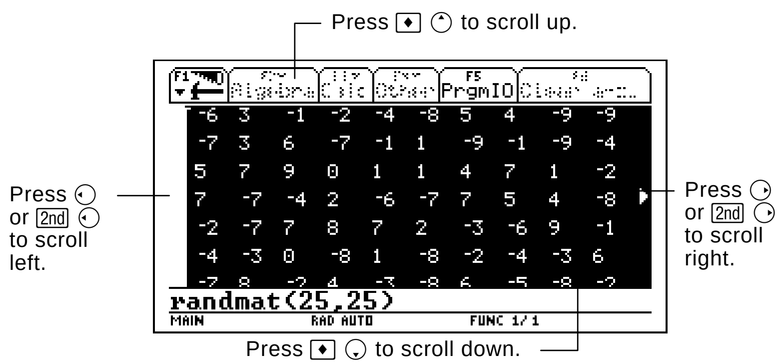
Move the cursor into the history area, and highlight the entry or answer. Then use the cursor pad to scroll. For example:

- The following shows an answer that is too long for one line.



- The following shows an answer that is both too long and too tall to be displayed on the screen.

Note: *This example uses the* **randMat** *function to generate a 25 x 25 matrix.*



If There Is not Enough Memory

A <<...>> symbol is displayed when the TI-92 does not have enough free memory to display the answer.

For example:

Note: *This example uses the* **seq** *function to generate a sequential list of integers from 1 to 2500.*

When you see the <<...>> symbol, the answer cannot be displayed even if you highlight it and try to scroll.

In general, you can try to:

- Free up additional memory by deleting unneeded variables. Use 2nd [VAR-LINK] as described in Chapter 18.
- If possible, break the problem into smaller parts that can be calculated and displayed with less memory.

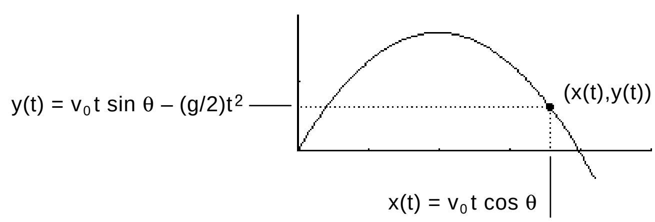
Parametric Graphing

11

This chapter describes how to graph parametric equations on the TI-92. Before using this chapter, you should be familiar with Chapter 3: Basic Function Graphing.

Parametric equations consist of both an x and y component, each expressed as a function of the same independent variable t.

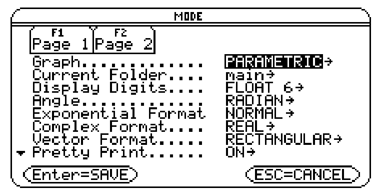
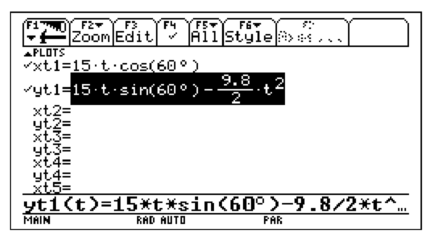
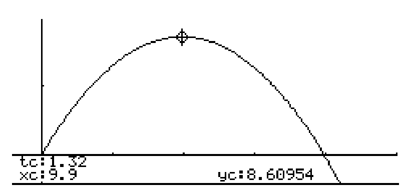
You can use parametric equations to model projectile motion. The position of a moving projectile has a horizontal (x) and vertical (y) component expressed as a function of time (t). For example:



The graph shows the path of the projectile over time, assuming that only uniform gravity (no drag forces, etc.) is acting on the projectile.

Preview of Parametric Graphing

Graph the parametric equations describing the path of a ball kicked at an angle (θ) of 60° with an initial velocity (v_0) of 15 meters/sec. The gravity constant g = 9.8 meters/sec^2. Ignoring air resistance and other drag forces, what is the maximum height of the ball and when does it hit the ground?

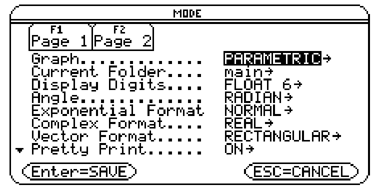
Steps	Keystrokes	Display
1. Display the MODE dialog box. For Graph mode, select PARAMETRIC.	[MODE] → 2 [ENTER]	
2. Display and clear the Y= Editor. Then define the horizontal component xt1(t) = v_0t cos θ. *Enter values for v_0 and θ.* *Type* T [×] [COS], *not* T [COS]. *Enter a ° symbol by typing either* [2nd] D *or* [2nd] [MATH] 2 1. *This ensures a number is interpreted as degrees, regardless of the angle mode.*	[◆] [Y=] [F1] 8 [ENTER] [ENTER] 1 5 T [×] [COS] 6 0 [2nd] D [)] [ENTER]	xt1(t)=15t*cos(60°)
3. Define the vertical component yt1(t) = v_0t sin θ – (g/2)t^2. *Enter values for v_0,θ, and g.*	[ENTER] 1 5 T [×] [SIN] 6 0 [2nd] D [)] [−] [(] 9 . 8 [÷] 2 [)] T [^] 2 [ENTER]	
4. Display the Window Editor. Enter Window variables appropriate for this example. *You can press either* ↓ *or* [ENTER] *to enter a value and move to the next variable.*	[◆] [WINDOW] 0 ↓ 3 ↓ . 0 2 ↓ [(-)] 2 ↓ 2 5 ↓ 5 ↓ [(-)] 2 ↓ 1 0 ↓ 5	tmin=0. tmax=3. tstep=.02 xmin=-2. xmax=25. xscl=5. ymin=-2. ymax=10. yscl=5.
5. Graph the parametric equations to model the path of the ball.	[◆] [GRAPH]	
6. Select Trace. Then move the cursor along the path to find the: • y value at maximum height. • t value where the ball hits the ground.	[F3] → or ← as necessary	

Overview of Steps in Graphing Parametric Equations

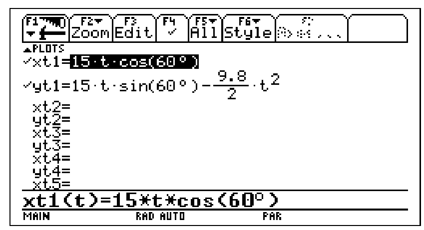
To graph parametric equations, use the same general steps used for y(x) functions as described in Chapter 3: Basic Function Graphing. Any differences that apply to parametric equations are described on the following pages.

Graphing Parametric Equations

Set Graph mode ([MODE]) to PARAMETRIC. Also set Angle mode, if necessary.



Define x and y components on Y= Editor ([◆] [Y=]).

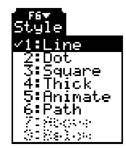


Tip: To turn off any stat data plots (Chapter 9), press [F5] 5 or use [F4] to deselect them.

Select ([F4]) which defined equations to graph. Select the x or y component, or both.

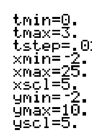
Tip: This is optional. For multiple equations, this helps visually distinguish one from another.

Set the display style ([F6]) for an equation. You can set either the x or y component.

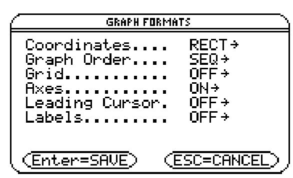


Tip: [F2] Zoom also changes the viewing window.

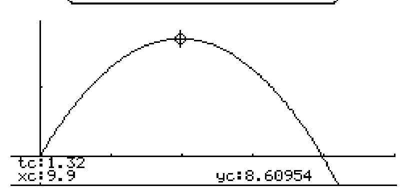
Define the viewing window ([◆] [WINDOW]).



Change the graph format ([◆] F or [F1] 9), if necessary.



Graph the selected equations ([◆] [GRAPH]).



Exploring the Graph

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a parametric equation.
- Use the [F2] Zoom toolbar menu to zoom in or out on a portion of the graph.
- Use the [F5] Math toolbar menu to find derivatives, tangents, etc. Some menu items are not available for parametric graphs.

Differences in Parametric and Function Graphing

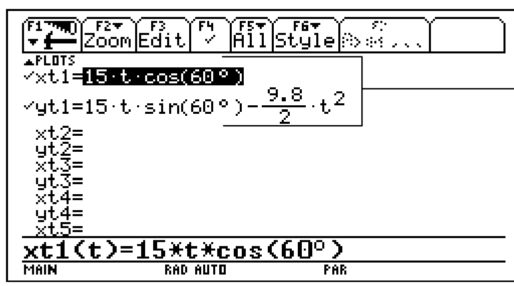
This chapter assumes that you already know how to graph y(x) functions as described in Chapter 3: Basic Function Graphing. This section describes the differences that apply to parametric equations.

Setting the Graph Mode

Use [MODE] to set Graph = PARAMETRIC before you define equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the *current* Graph mode setting only.

Defining Parametric Equations on the Y= Editor

To graph a parametric equation, you must define both its x and y components. If you define only one component, the equation cannot be graphed. (However, you can use single components to generate an automatic table as described in Chapter 4.)



Enter x and y components on separate lines.

You can define xt1(t) through xt99(t) and yt1(t) through yt99(t).

Be careful when using implied multiplication with t. For example:

Note: ***Note:*** *When using t, be sure implied multiplication is valid for your situation.*

Enter:	Instead of:	Because:
t∗cos(60)	tcos(60)	tcos is interpreted as a user-defined function called **tcos**, not as implied multiplication. In most cases, this refers to a nonexistent function. So the TI-92 simply returns the function name, not a number.

Tip: *You can use the* ***Define*** *command from the Home screen (see Appendix A) to define functions and equations for any graphing mode, regardless of the current mode.*

The Y= Editor maintains an independent function list for each Graph mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of y(x) functions. You change to PARAMETRIC graphing mode and define a set of x and y components.
- When you return to FUNCTION graphing mode, your y(x) functions are still defined in the Y= Editor. When you return to PARAMETRIC graphing mode, your x and y components are still defined.

Selecting Parametric Equations

To graph a parametric equation, select *either* its x or y component or *both*. When you enter or edit a component, it is selected automatically.

Selecting x and y components separately can be useful for tables as described in Chapter 4. With multiple parametric equations, you can select and compare all the x components or all the y components.

Selecting the Display Style

You can set the style for either the x or y component. For example, if you set the x component to Dot, the TI-92 automatically sets the y component to Dot.

Tip: *Use the Animate and Path styles for interesting projectile-motion effects.*

The Above and Below styles are not available for parametric equations and are dimmed on the Y= Editor's [F6] Style toolbar menu.

Window Variables

The Window Editor maintains an independent set of Window variables for each Graph mode setting (just as the Y= Editor maintains independent function lists). Parametric graphs use the following Window variables.

Note: *You can use a negative tstep. If so, tmin must be greater than tmax.*

Variable	Description
tmin, tmax	Smallest and largest t values to evaluate.
tstep	Increment for the t value. Parametric equations are evaluated at: x(tmin) y(tmin) x(tmin+tstep) y(tmin+tstep) x(tmin+2(tstep)) y(tmin+2(tstep)) ... not to exceed ... ... not to exceed ... x(tmax) y(tmax)
xmin, xmax, ymin, ymax	Boundaries of the viewing window.
xscl, yscl	Distance between tick marks on the x and y axes.

Standard values (set when you select 6:ZoomStd from the [F2] Zoom toolbar menu) are:

tmin = 0.		xmin = ⁻10.	ymin = ⁻10.
tmax = 2π	(6.2831853... radians or 360 degrees)	xmax = 10.	ymax = 10.
tstep =π/24	(.1308996... radians or 7.5 degrees)	xscl = 1.	yscl = 1.

You may need to change the standard values for the t variables (tmin, tmax, tstep) to ensure that enough points are plotted.

Exploring a Graph

As in function graphing, you can explore a graph by using the following tools.

Tip: During a trace, you can also evaluate x(t) and y(t) by typing the t value and pressing [ENTER].

Tip: You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.

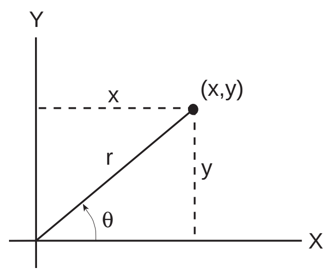
Tool	For Parametric Graphs:
Free-Moving Cursor	Works just as it does for function graphs.
[F2] Zoom	Works just as it does for function graphs, with the following exceptions: • Only x (xmin, xmax, xscl) and y (ymin, ymax, yscl) Window variables are affected. • The t Window variables (tmin, tmax, tstep) are not affected unless you select 6:ZoomStd (which sets tmin = 0, tmax = 2π, and tstep = π/24).
[F3] Trace	Lets you move the cursor along a graph one tstep at a time. • When you begin a trace, the cursor is on the first selected parametric equation at tmin. • QuickCenter applies to all directions. If you move the cursor off the screen (top or bottom, left or right), press [ENTER] to center the viewing window on the cursor location. • Automatic panning is not available. If you move the cursor off the left or right side of the screen, the TI-92 will not automatically pan the viewing window. However, you can use QuickCenter.
[F5] Math	Only 1:Value, 6:Derivatives, 9:Distance, A:Tangent, and B:Arc are available for parametric graphs. These tools are based on t values. For example: • 1:Value displays x and y values for a specified t value. • 6:Derivatives finds dy/dx, dy/dt, or dx/dt at a point defined for a specified t value.

Polar Graphing

12

This chapter describes how to graph polar equations on the TI-92. Before using this chapter, you should be familiar with Chapter 3: Basic Function Graphing.

Consider a point (x,y) as shown below. In a polar equation, the point's distance (r) from the origin is a function of its angle (θ) from the positive x axis. Polar equations are expressed as r = f(θ).



To convert between rectangular (x,y) and polar coordinates (r,θ):

$x = r\cos\theta$ $\qquad r^2 = x^2 + y^2$

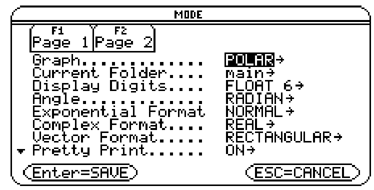
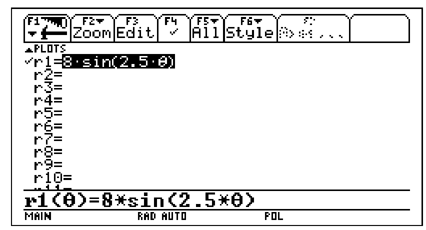
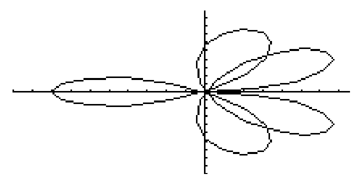
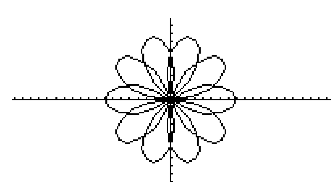
$y = r\sin\theta$ $\qquad \theta = -\tan^{-1}\frac{x}{y} + \frac{\text{sign}(y)\cdot\pi}{2}$

Note: To find θ, use the TI-92 function angle(x+*i*y), which automatically performs the calculation shown above.

You can view the coordinates of any point in either polar (r,θ) or rectangular (x,y) form.

Preview of Polar Graphing

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

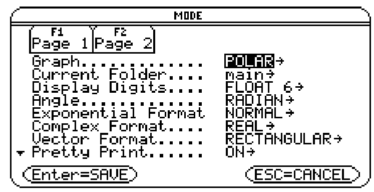
Steps	Keystrokes	Display
1. Display the MODE dialog box. For Graph mode, select POLAR. For Angle mode, select RADIAN.	[MODE] ▶ 3 ▼ ▼ ▼ ▶ 1 [ENTER]	
2. Display and clear the Y= Editor. Then define the polar equation r1(θ) = A sin Bθ. *Enter 8 and 2.5 for A and B, respectively.*	[♦] [Y=] [F1] 8 [ENTER] [ENTER] 8 [SIN] 2 . 5 [θ] [)] [ENTER]	
3. Select the ZoomStd viewing window, which graphs the equation. • *The graph shows only five rose petals.* – *In the standard viewing window, the Window variable θmax = 2π. The remaining petals have θ values greater than 2π.* • *The rose does not appear symmetrical.* – *Both the x and y axes range from −10 to 10. However, this range is spread over a longer distance along the x axis than the y axis.*	[F2] 6	
4. Display the Window Editor, and change θmax to 4π. *4π will be evaluated to a number when you leave the Window Editor.*	[♦] [WINDOW] ▼ 4 [2nd] [π]	θmin=0. θmax=4π θstep=.13089969389957 xmin=-10. xmax=10. xscl=1. ymin=-10. ymax=10. yscl=1.
5. Select ZoomSqr, which regraphs the equation. *ZoomSqr increases the range along the x axis so that the graph is shown in correct proportion.*	[F2] 5	
6. You can change values for A and B as necessary and regraph the equation.		

Overview of Steps in Graphing Polar Equations

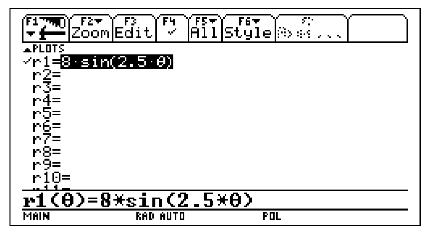
To graph polar equations, use the same general steps used for y(x) functions as described in Chapter 3: Basic Function Graphing. Any differences that apply to polar equations are described on the following pages.

Graphing Polar Equations

Set Graph mode ([MODE]) to POLAR. Also set Angle mode, if necessary.



Define polar equations on Y= Editor ([♦] [Y=]).

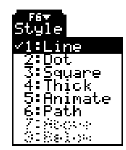


Tip: To turn off any stat data plots (Chapter 9), press [F5] 5 *or use* [F4] *to deselect them.*

Select ([F4]) which defined equations to graph.

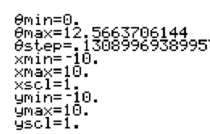
Tip: This is optional. For multiple equations, this helps visually distinguish one from another.

Set the display style ([F6]) for an equation.



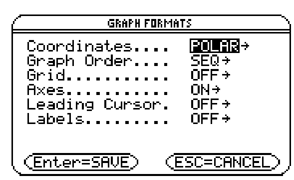
Tip: [F2] Zoom *also changes the viewing window.*

Define the viewing window ([♦] [WINDOW]).

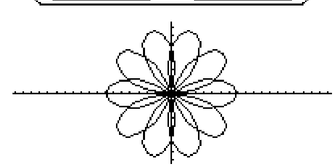


Tip: To display r and θ, set Coordinates = POLAR.

Change the graph format ([♦] F or [F1] 9), if necessary.



Graph the selected equations ([♦] [GRAPH]).



Exploring the Graph

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a polar equation.
- Use the [F2] Zoom toolbar menu to zoom in or out on a portion of the graph.
- Use the [F5] Math toolbar menu to find derivatives, tangents, etc. Some menu items are not available for polar graphs.

Differences in Polar and Function Graphing

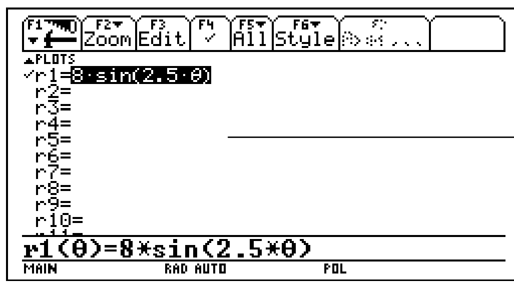
This chapter assumes that you already know how to graph y(x) functions as described in Chapter 3: Basic Function Graphing. This section describes the differences that apply to polar equations.

Setting the Graph Mode

Use [MODE] to set Graph = POLAR before you define equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the *current* Graph mode setting only.

You should also set the Angle mode to the units (RADIAN or DEGREE) you want to use for θ.

Defining Polar Equations on the Y= Editor



You can define polar equations for r1(θ) through r99(θ).

__Tip:__ You can use the __Define__ command from the Home screen (see Appendix A) to define functions and equations for any graphing mode, regardless of the current mode.

The Y= Editor maintains an independent function list for each Graph mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of y(x) functions. You change to POLAR graphing mode and define a set of r(θ) equations.
- When you return to FUNCTION graphing mode, your y(x) functions are still defined in the Y= Editor. When you return to POLAR graphing mode, your r(θ) equations are still defined.

Selecting the Display Style

The Above and Below styles are not available for polar equations and are dimmed on the Y= Editor's [F6] Style toolbar menu.

Window Variables

The Window Editor maintains an independent set of Window variables for each Graph mode setting (just as the Y= Editor maintains independent function lists). Polar graphs use the following Window variables.

Note: You can use a negative θstep. If so, θmin must be greater than θmax.

Variable	Description
θmin, θmax	Smallest and largest θ values to evaluate.
θstep	Increment for the θ value. Polar equations are evaluated at: r(θmin) r(θmin+θstep) r(θmin+2(θstep)) ... not to exceed ... r(θmax)
xmin, xmax, ymin, ymax	Boundaries of the viewing window.
xscl, yscl	Distance between tick marks on the x and y axes.

Standard values (set when you select 6:ZoomStd from the [F2] Zoom toolbar menu) are:

θmin = 0.		xmin = ⁻10.	ymin = ⁻10.
θmax = 2π	(6.2831853... radians or 360 degrees)	xmax = 10.	ymax = 10.
θstep = π/24	(.1308996... radians or 7.5 degrees)	xscl = 1.	yscl = 1.

You may need to change the standard values for the θ variables (θmin, θmax, θstep) to ensure that enough points are plotted.

Setting the Graph Format

To display coordinates as r and θ values, use [♦] F or [F1] 9 to set Coordinates = POLAR. If Coordinates = RECT, the polar equations will be graphed properly, but coordinates will be displayed as x and y.

When you trace a polar equation, the θ coordinate is shown even if Coordinates = RECT.

Exploring a Graph

As in function graphing, you can explore a graph by using the following tools. Any displayed coordinates are shown in polar or rectangular form as set in the graph format.

Tip: During a trace, you can also evaluate r(θ) by typing the θ value and pressing [ENTER].

Tip: You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.

Tool	For Polar Graphs:
Free-Moving Cursor	Works just as it does for function graphs.
[F2] Zoom	Works just as it does for function graphs. • Only x (xmin, xmax, xscl) and y (ymin, ymax, yscl) Window variables are affected. • The θ Window variables (θmin, θmax, θstep) are not affected unless you select 6:ZoomStd (which sets θmin = 0, θmax = 2π, and θstep = π/24).
[F3] Trace	Lets you move the cursor along a graph one θstep at a time. • When you begin a trace, the cursor is on the first selected equation at θmin. • QuickCenter applies to all directions. If you move the cursor off the screen (top or bottom, left or right), press [ENTER] to center the viewing window on the cursor location. • Automatic panning is not available. If you move the cursor off the left or right side of the screen, the TI-92 will not automatically pan the viewing window. However, you can use QuickCenter.
[F5] Math	Only 1:Value, 6:Derivatives, 9:Distance, A:Tangent, and B:Arc are available for polar graphs. These tools are based on θ values. For example: • 1:Value displays an r value (or x and y, depending on the graph format) for a specified θ value. • 6:Derivatives finds dy/dx or dr/dθ at a point defined for a specified θ value.

Sequence Graphing

13

This chapter describes how to graph sequences on the TI-92. Before using this chapter, you should be familiar with Chapter 3: Basic Function Graphing.

Sequences are evaluated only at consecutive integer values. The two general types of sequences are:

- **Nonrecursive** — The nth term in the sequence is a function of the independent variable n.

  Each term is independent of any other terms. In the following example sequence, you can calculate u(5) directly, without first calculating u(1) or any other previous term.

  u(n) = 2 ∗ n for n = 1, 2, 3, ...

  n is always a series of consecutive integers, starting at any positive integer or zero.

  u(n) = 2 ∗ n gives the sequence 2, 4, 6, 8, 10, ...

__Note:__ A recursive sequence can reference another sequence. For example, $u2(n) = n^2+u1(n-1)$.

- **Recursive** — The nth term is defined in relation to one or more previous terms, represented by u(n−1), u(n−2), etc. In addition to previous terms, a recursive sequence may also be defined in relation to n (such as u(n) = u(n−1) + n).

  In the following example sequence, you cannot calculate u(5) without first calculating u(1), u(2), u(3), and u(4).

  u(n) = 2 ∗ u(n−1) for n = 1, 2, 3, ...

  The first term is undefined since it has no previous term. So you must specify an initial value to use for the first term.

  Using an initial value of 1:

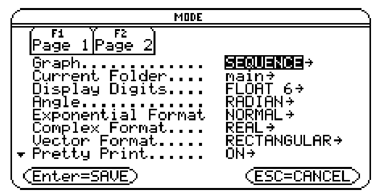
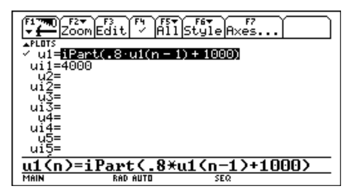
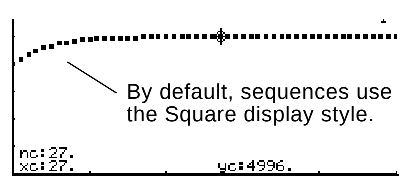
  u(n) = 2 ∗ u(n−1) gives the sequence 1, 2, 4, 8, 16, ...

  The number of initial values you need to specify depends on how deep the recursion goes. For example, if each term is defined in relation to the previous two terms, you must specify initial values for the first two terms.

Preview of Sequence Graphing

A small forest contains 4000 trees. Each year, 20% of the trees will be harvested (with 80% remaining) and 1000 new trees will be planted. Using a sequence, calculate the number of trees in the forest at the end of each year. Does it stabilize at a certain number?

Initially	After 1 Year	After 2 Years	After 3 Years	. . .
4000	.8 x 4000 + 1000	.8 x (.8 x 4000 + 1000) + 1000	.8 x (.8 x (.8 x 4000 + 1000) + 1000) + 1000	. . .

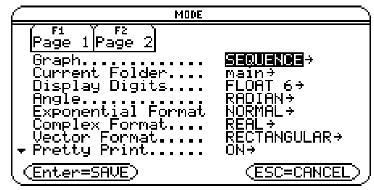
Steps	Keystrokes	Display
1. Display the MODE dialog box. For Graph mode, select SEQUENCE.	[MODE] [▶] 4 [ENTER]	
2. Display and clear the Y= Editor. Then define the sequence as u1(n) = iPart(.8∗u1(n−1)+1000). *Use* **iPart** *to take the integer part of the result. No fractional trees are harvested.* *To access* **iPart(**, *you can use* [2nd] [MATH], [2nd] [CATALOG], *or simply type it.*	[♦] [Y=] [F1] 8 [ENTER] [ENTER] [2nd] [MATH] 1 4 . 8 U 1 [(] N [−] 1 [)] [+] 1 0 0 0 [)] [ENTER]	
3. Define ui1 as the initial value that will be used as the first term.	[ENTER] 4 0 0 0 [ENTER]	
4. Display the Window Editor. Set the n and plot Window variables. *nmin=0 and nmax=50 evaluate the size of the forest over 50 years.*	[♦] [WINDOW] 0 [▼] 5 0 [▼] 1 [▼] 1 [▼]	nmin=0. nmax=50. plotstrt=1. plotstep=1. xmin=0. xmax=50. xscl=10. ymin=0. ymax=6000. yscl=1000.
5. Set the x and y Window variables to appropriate values for this example.	0 [▼] 5 0 [▼] 1 0 [▼] 0 [▼] 6 0 0 0 [▼] 1 0 0 0	
6. Display the Graph screen.	[♦] [GRAPH]	
7. Select Trace. Move the cursor to trace year by year. How many years (nc) does it take the number of trees (yc) to stabilize? *Trace begins at nc=0.* *nc is the number of years.* *xc = nc since n is plotted on the x axis.* *yc = u1(n), the number of trees at year n.*	[F3] [▶] and [◀] as necessary	

Overview of Steps in Graphing Sequences

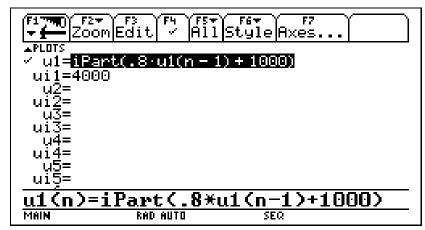
To graph sequences, use the same general steps used for y(x) functions as described in Chapter 3: Basic Function Graphing. Any differences are described on the following pages.

Graphing Sequences

Set Graph mode ([MODE]) to SEQUENCE. Also set Angle mode, if necessary.



Define sequences and, if needed, initial values on Y= Editor ([♦] [Y=]).

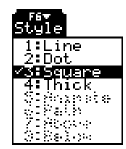


Tip: To turn off any stat data plots (Chapter 9), press [F5] 5 *or use* [F4] *to deselect them.*

Select ([F4]) which defined sequences to graph. Do not select initial values.

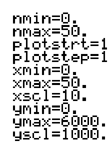
Note: For sequences, the default style is Square.

Set the display style ([F6]) for a sequence.

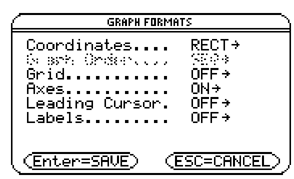


Tip: [F2] Zoom *also changes the viewing window.*

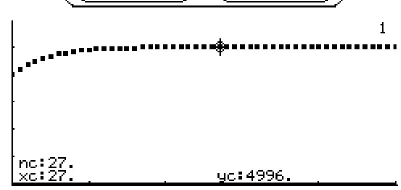
Define the viewing window ([♦] [WINDOW]).



Change the graph format ([♦] F or [F1] 9), if necessary.



Graph the selected sequences ([♦] [GRAPH]).



Exploring the Graph

From the Graph screen, you can:

- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a sequence.
- Use the [F2] Zoom toolbar menu to zoom in or out on a portion of the graph.
- Use the [F5] Math toolbar menu to evaluate a sequence. Only 1:Value is available for sequences.
- Plot sequences on Time (the default), Web, or Custom axes.

Tip: You can also evaluate a sequence while tracing. Simply enter the n value directly from the keyboard.

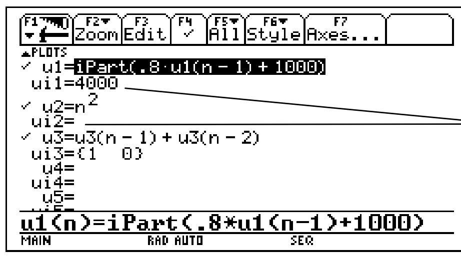
Differences in Sequence and Function Graphing

This chapter assumes that you already know how to graph y(x) functions as described in Chapter 3: Basic Function Graphing. This section describes the differences that apply to sequences.

Setting the Graph Mode

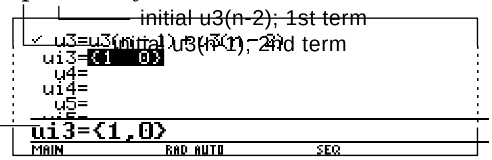
Use [MODE] to set Graph = SEQUENCE before you define sequences or set Window variables. The Y= Editor and the Window Editor let you enter information for the *current* Graph mode setting only.

Defining Sequences on the Y= Editor



Note: *You must use a list to enter two or more initial values.*

If a sequence requires more than one initial value, enter them as a list enclosed in braces { } and separated by commas.



If a sequence requires an initial value but you do not enter one, you will get an error when graphing.

Note: *Optionally, for sequences only, you can select different axes for the graph. TIME is the default.*

On the Y= Editor, [F7] Axes lets you select the axes that are used to graph the sequences. For more detailed information, refer to page 240.

Axes	Description
TIME	Plots n on the x axis and u(n) on the y axis.
WEB	Plots u(n-1) on the x axis and u(n) on the y axis.
CUSTOM	Lets you select the x and y axes.

Tip: *You can use the **Define** command from the Home screen (see Appendix A) to define functions and equations for any graphing mode, regardless of the current mode.*

The Y= Editor maintains an independent function list for each Graph mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of y(x) functions. You change to SEQUENCE graphing mode and define a set of u(n) sequences.
- When you return to FUNCTION graphing mode, your y(x) functions are still defined in the Y= Editor. When you return to SEQUENCE graphing mode, your u(n) sequences are still defined.

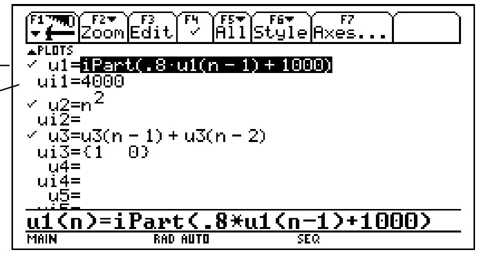
Selecting Sequences

With TIME and WEB axes, the TI-92 graphs only the selected sequences. If you entered any sequences that require an initial value, you must enter the corresponding ui value.

Note: *With TIME and CUSTOM axes, all defined sequences are evaluated even if they are not plotted.*

You can select a sequence.

You cannot select its initial value.



With CUSTOM axes, when you specify a sequence in the custom settings, it is graphed regardless of whether it is selected.

Selecting the Display Style

Only the Line, Dot, Square, and Thick styles are available for sequence graphs. Dot and Square mark only those discrete integer values (in plotstep increments) at which a sequence is plotted.

Window Variables

The Window Editor maintains an independent set of Window variables for each Graph mode setting (just as the Y= Editor maintains independent function lists). Sequence graphs use the following Window variables.

Note: *Both nmin and nmax must be positive integers, although nmin can be zero.*

Note: *nmin, nmax, plotstrt and plotstep must be integers ≥ 1. If you do not enter integers, they will be rounded to integers.*

Variable	Description
nmin, nmax	Smallest and largest n values to evaluate. Sequences are evaluated at: u(nmin) u(nmin+1) u(nmin+2) ... not to exceed ... u(nmax)
plotstrt	The term number that will be the first one plotted (depending on plotstep). For example, to begin plotting with the 2nd term in the sequence, set plotstrt = 2. The first term will be evaluated at nmin but not plotted.
plotstep	Incremental n value *for graphing only*. This does not affect how the sequence is evaluated, only which points are plotted. For example, suppose plotstep = 2. The sequence is evaluated at each consecutive integer but is plotted at only every other integer.
xmin, xmax, ymin, ymax	Boundaries of the viewing window.
xscl, yscl	Distance between tick marks on the x and y axes.

Window Variables (Continued)

Standard values (set when you select 6:ZoomStd from the F2 Zoom toolbar menu) are:

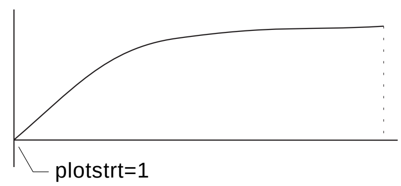
nmin = 1.	xmin = -10.	ymin = -10.
nmax = 10.	xmax = 10.	ymax = 10.
plotstrt = 1.	xscl = 1.	yscl = 1.
plotstep = 1.		

You may need to change the standard values for the n and plot variables to ensure that sufficient points are plotted.

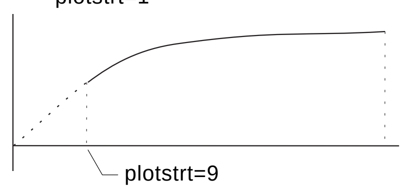
To see how plotstrt affects a graph, look at the following examples of a recursive sequence.

Note: Both of these graphs use the same Window variables, except for plotstrt.

This graph is plotted beginning with the 1st term.



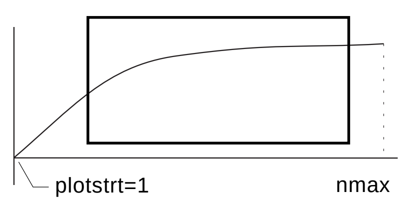
This graph is plotted beginning with the 9th term.



With TIME axes (from F7 Axes on the Y= Editor), you can set plotstrt = 1 and still graph only a selected part of the sequence. Simply define a viewing window that shows only the area of the coordinate plane you want to view.

You could set:

- xmin = first n value to graph
- xmax = nmax (although you can use other values)
- ymin and ymax = expected values for the sequence



Changing the Graph Format

The Graph Order format is not available.

- With TIME or CUSTOM axes, multiple sequences are always plotted simultaneously.
- With WEB axes, multiple sequences are always plotted sequentially.

Exploring a Graph

As in function graphing, you can explore a graph by using the following tools. Any displayed coordinates are shown in rectangular or polar form as set in the graph format.

Tip: During a trace, you can evaluate a sequence by typing a value for n and pressing [ENTER].

Tip: You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.

Tool	For Sequence Graphs:
Free-Moving Cursor	Works just as it does for function graphs.
[F2] Zoom	Works just as it does for function graphs. • Only x (xmin, xmax, xscl) and y (ymin, ymax, yscl) Window variables are affected. • The n and plot Window variables (nmin, nmax, plotstrt, plotstep) are not affected unless you select 6:ZoomStd (which sets all Window variables to their standard values).
[F3] Trace	Depending on whether you use TIME, CUSTOM, or WEB axes, Trace operates very differently. • With TIME or CUSTOM axes, you move the cursor along the sequence one plotstep at a time. To move approximately ten plotted points at a time, press [2nd] ▶ or [2nd] ◀. – When you begin a trace, the cursor is on the first selected sequence at the term number specified by plotstrt, even if it is outside the viewing window. – QuickCenter applies to all directions. If you move the cursor off the screen (top or bottom, left or right), press [ENTER] to center the viewing window on the cursor location. • With WEB axes, the trace cursor follows the web, not the sequence. Refer to page 241.
[F5] Math	Only 1:Value is available for sequence graphs. • With TIME and WEB axes, the u(n) value (represented by yc) is displayed for a specified n value. • With CUSTOM axes, the values that correspond to x and y depend on the axes you choose.

Setting Axes for Time, Web, or Custom Plots

For sequences only, you can select different types of axes for the graph. Examples of the different types are given later in this chapter.

Displaying the AXES Dialog Box

From the Y= Editor, press [F7].

- Depending on the current Axes setting, some items may be dimmed.
- To exit without making any changes, press [ESC].

Item	Description
Axes	TIME — Plots u(n) on the y axis and n on the x axis.
	WEB — Plots u(n) on the y axis and u(n-1) on the x axis.
	CUSTOM — Lets you select the x and y axes.
Build Web	Active only when Axes = WEB, this specifies whether a web is drawn manually (TRACE) or automatically (AUTO).
	Refer to page 241 for more information.
X Axis and Y Axis	Active only when Axes = CUSTOM, these let you select the value or sequence to plot on the x and y axes.
	Refer to page 244 for more information.

To change any of these settings, use the same procedure that you use to change other types of dialog boxes, such as the MODE dialog box.

Using Web Plots

A web plot graphs u(n) vs. u(n−1), which lets you study the long-term behavior of a recursive sequence. The examples in this section also show how the initial value can affect a sequence's behavior.

Valid Functions for Web Plots

A sequence must meet the following criteria; otherwise, it will not be graphed properly on WEB axes. The sequence:

- Must be recursive with only one recursion level; u(n−1) but not u(n−2).
- Cannot reference n directly.
- Cannot reference any other defined sequence except itself.

When You Display the Graph Screen

After you select WEB axes and display the Graph screen, the TI-92:

- Draws a y=x reference line.
- Plots the selected sequence definitions as functions, with u(n−1) as the independent variable. This effectively converts a recursive sequence into a nonrecursive form for graphing.

For example, consider the sequence $u1(n) = \sqrt{5-u1(n-1)}$. The TI-92 draws the y=x reference line and then plots $y = \sqrt{5-x}$.

Drawing the Web

After the sequence is plotted, the web may be displayed manually or automatically, depending on how you set Build Web on the AXES dialog box.

If Build Web =	The web is:
TRACE	Not drawn until you press [F3]. The web is then drawn step-by-step as you move the trace cursor. **Note:** With WEB axes, you cannot trace along the sequence itself as you do in other graphing modes.
AUTO	Drawn automatically. You can then press [F3] to trace the web and display its coordinates.

__Note:__ The web starts at plotstrt. The value of n is incremented by 1 each time the web moves to the sequence (plotstep is ignored).

The web:

1. Starts on the x axis at the initial value ui (when plotstrt = 1).
2. Moves vertically (either up or down) to the sequence.
3. Moves horizontally to the y=x reference line.
4. Repeats this vertical and horizontal movement until n=nmax.

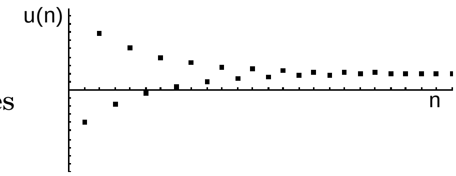
Example: Convergence

1. On the Y= Editor ([♦] [Y=]), define u1(n) = ⁻.8u1(n−1) + 3.6. Set initial value ui1 = ⁻4.
2. Press [F7]. Set Axes = TIME.
3. On the Window Editor ([♦] [WINDOW]), set the Window variables.

nmin=1.	xmin=0.	ymin=⁻10.
nmax=25.	xmax=25.	ymax=10.
plotstrt=1.	xscl=1.	yscl=1.
plotstep=1.		

4. Graph the sequence ([♦] [GRAPH]).

   By default, a sequence uses the Square display style.



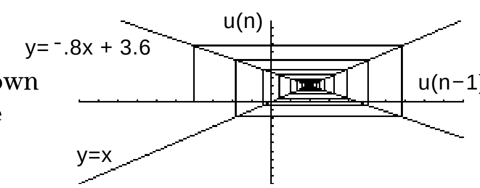
5. On the Y= Editor, press [F7]. Set Axes = WEB and Build Web = AUTO.
6. On the Window Editor, change the Window variables.

nmin=1.	xmin=⁻10.	ymin=⁻10.
nmax=25.	xmax=10.	ymax=10.
plotstrt=1.	xscl=1.	yscl=1.
plotstep=1.		

Tip: *During a trace, you can move the cursor to a specified n value by typing the value and pressing [ENTER].*

7. Regraph the sequence.

   Web plots are always shown as lines, regardless of the selected display style.



Tip: *When the nc value changes, the cursor is on the sequence. The next time you press [▶], nc stays the same but the cursor is now on the y=x reference line.*

8. Press [F3]. As you press [▶], the trace cursor follows the web. The screen displays the cursor coordinates nc, xc, and yc (where xc and yc represent u(n−1) and u(n), respectively).

As you trace to larger values of nc, you can see xc and yc approach the convergence point.

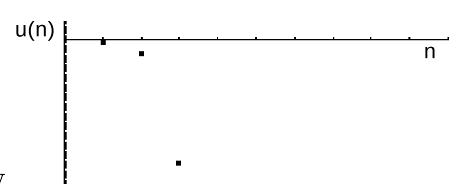
Example: Divergence

1. On the Y= Editor ([♦] [Y=]), define u1(n) = 3.2u1(n−1) − .8(u1(n−1))^2. Set initial value ui1 = 4.45.
2. Press [F7]. Set Axes = TIME.
3. On the Window Editor ([♦] [WINDOW]), set the Window variables.

nmin=0.	xmin=0.	ymin=⁻75.
nmax=10.	xmax=10.	ymax=10.
plotstrt=1.	xscl=1.	yscl=1.
plotstep=1.		

4. Graph the sequence ([♦] [GRAPH]).

   Because the sequence quickly diverges to large negative values, only a few points are plotted.



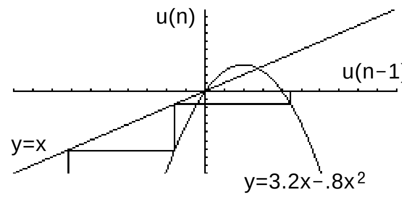
5. On the Y= Editor, press [F7]. Set Axes = WEB and Build Web = AUTO.

6. On the Window Editor, change the Window variables.

nmin=0.	xmin=-10.	ymin=-10.
nmax=10.	xmax=10.	ymax=10.
plotstrt=1.	xscl=1.	yscl=1.
plotstep=1.		

7. Regraph the sequence.

   The web plot shows how quickly the sequence diverges to large negative values.



Example: Oscillation

This example shows how the initial value can affect a sequence.

1. On the Y= Editor ([♦] [Y=]), use the same sequence defined in the divergence example: $u1(n) = 3.2u1(n-1) - .8(u1(n-1))^2$. Set initial value ui1 = 0.5.

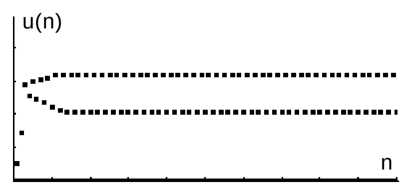
2. Press [F7]. Set Axes = TIME.

3. On the Window Editor ([♦] [WINDOW]), set the Window variables.

nmin=1.	xmin=0.	ymin=0.
nmax=100.	xmax=100.	ymax=5.
plotstrt=1.	xscl=10.	yscl=1.
plotstep=1.		

__Note:__ Compare this graph with the divergence example. This is the same sequence with a different initial value.

4. Graph the sequence ([♦] [GRAPH]).



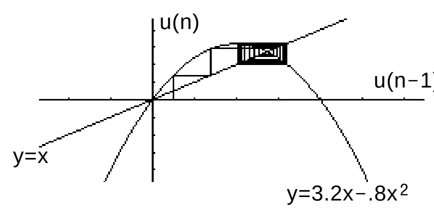
5. On the Y= Editor, press [F7]. Set Axes = WEB and Build Web = AUTO.

6. On the Window Editor, change the Window variables.

nmin=1.	xmin=-2.68	ymin=-4.7
nmax=100.	xmax=6.47	ymax=4.7
plotstrt=1.	xscl=1.	yscl=1.
plotstep=1.		

__Note:__ The web moves to an orbit oscillating between two stable points.

7. Regraph the sequence.

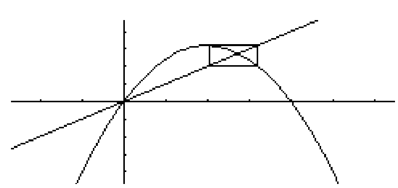


8. Press [F3]. Then use ▶ to trace the web.

   As you trace to larger values of nc, notice that xc and yc oscillate between 2.05218 and 3.19782.

__Note:__ By starting the web plot at a later term, the stable oscillation orbit is shown more clearly.

9. On the Window Editor, set plotstrt=50. Then regraph the sequence.



CUSTOM axes give you great flexibility in graphing sequences. As shown in the following example, CUSTOM axes are particularly effective for showing relationships between one sequence and another.

Example: Predator-Prey Model

Using the predator-prey model in biology, determine the numbers of rabbits and wolves that maintain population equilibrium in a certain region.

R = Number of rabbits
M = Growth rate of rabbits if there are no wolves (use .05)
K = Rate at which wolves can kill rabbits (use .001)
W = Number of wolves
G = Growth rate of wolves if there are rabbits (use .0002)
D = Death rate of wolves if there are no rabbits (use .03)

$R_n = R_{n-1} (1 + M - K W_{n-1})$
$W_n = W_{n-1} (1 + G R_{n-1} - D)$

1. On the Y= Editor ([◆] [Y=]), define the sequences and initial values for R_n and W_n.

   Note: *Assume there are initially 200 rabbits and 50 wolves.*

   u1(n) = u1(n−1) ∗ (1 + .05 − .001 ∗ u2(n−1))
   ui1 = 200
   u2(n) = u2(n−1) ∗ (1 + .0002 ∗ u1(n−1) − .03)
   ui2 = 50

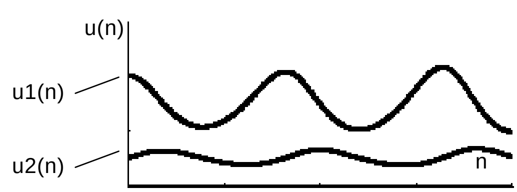
2. Press [F7]. Set Axes = TIME.

3. On the Window Editor ([◆] [WINDOW]), set the Window variables.

   nmin=0. nmax=400. plotstrt=1. plotstep=1.
   xmin=0. xmax=400. xscl=100.
   ymin=0. ymax=300. yscl=100.

4. Graph the sequence ([◆] [GRAPH]).

   Note: *Use [F3] to individually trace the number of rabbits u1(n) and wolves u2(n) over time (n).*



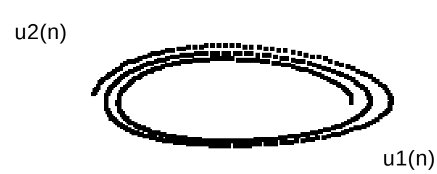
5. On the Y= Editor, press [F7]. Set Axes = CUSTOM, X Axis = u1, and Y Axis = u2.

6. On the Window Editor, change the Window variables.

   nmin=0. nmax=400. plotstrt=1. plotstep=1.
   xmin=84. xmax=237. xscl=50.
   ymin=25. ymax=75. yscl=10.

7. Regraph the sequence.

   Note: *Use [F3] to trace both the number of rabbits (xc) and wolves (yc) over the cycle of 400 generations.*



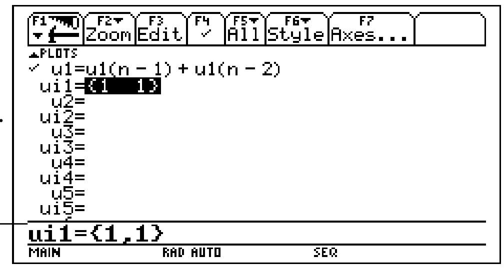
Using a Sequence to Generate a Table

Previous sections described how to graph a sequence. You can also use a sequence to generate a table. Refer to Chapter 4 for detailed information about tables.

Example: Fibonacci Sequence

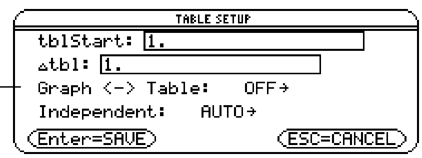
In a Fibonacci sequence, the first two terms are 1 and 1. Each succeeding term is the sum of the two immediately preceding terms.

1. On the Y= Editor ([◆] [Y=]), define the sequence and set the initial values as shown.



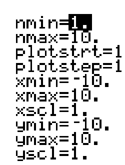
You must enter {1,1}, although {1 1} is shown in the sequence list.

2. Set table parameters ([◆] [TblSet]) to:
   tblStart = 1
   Δtbl = 1
   Independent = AUTO

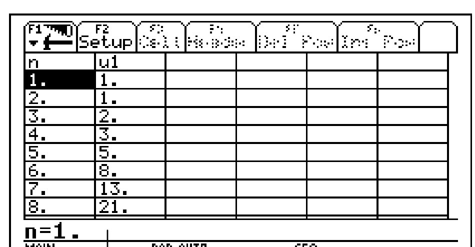


This item is dimmed if you are not using TIME axes (set by [F7] on the Y= Editor).

3. Set Window variables ([◆] [WINDOW]) so that nmin has the same value as tblStart.



4. Display the table ([◆] [TABLE]).



Fibonacci sequence is in column 2.

5. Scroll down the table (⊙ or [2nd] ⊙) to see more of the sequence.

Comparison of TI-92 and TI-82 Sequence Functions

Use the following table if you are familiar with sequences on the TI-82. The table lists sequences and sequence-related Window variables on the TI-92 and shows their counterpart on the TI-82.

Comparison Table

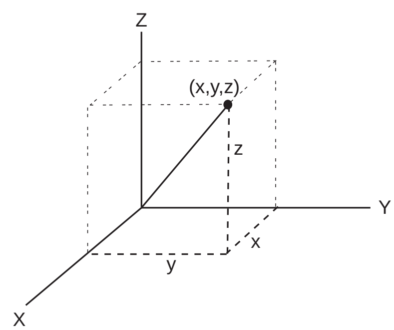
TI-92	TI-82
On the Y= Editor:	
u1(n)	Un
ui1	UnStart (Window variable on TI-82)
u2(n)	Vn
ui2	VnStart (Window variable on TI-82)
u3(n) through u99(n)	not available
ui3 through ui99	not available
On the Window Editor:	
nmin	nStart
nmax	nMax
plotstrt	nMin
plotstep	not available

3D Graphing

14

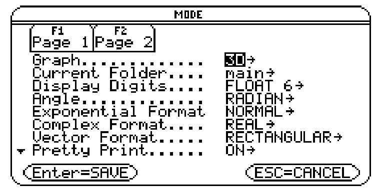
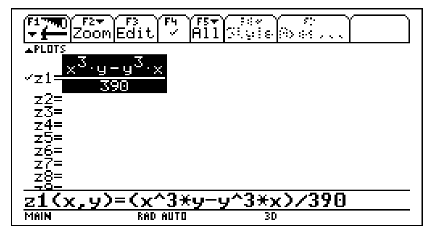
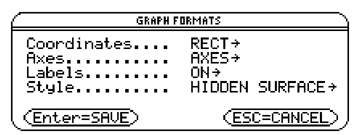
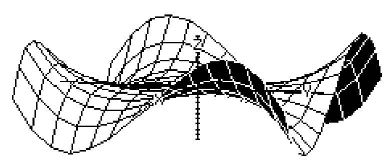
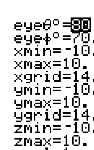
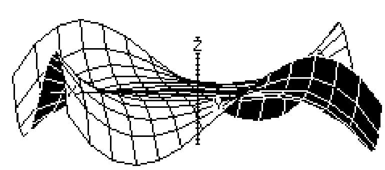
This chapter describes how to graph 3D equations on the TI-92. Before using this chapter, you should be familiar with Chapter 3: Basic Function Graphing.

In a 3D graph of an equation for z(x,y), a point's location is defined as shown below.



Preview of 3D Graphing

Graph the 3D equation z(x,y) = $(x^3y - y^3x) / 390$. Then rotate your viewing angle around the Z axis.

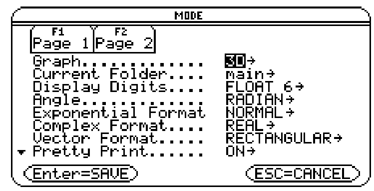
Steps	Keystrokes	Display
1. Display the MODE dialog box. For Graph mode, select 3D.	[MODE] ▸ 5 [ENTER]	
2. Display and clear the Y= Editor. Then define the 3D equation z1(x,y) = $(x^3y - y^3x) / 390$. *Notice how implied multiplication is used in the keystrokes.*	[♦] [Y=] [F1] 8 [ENTER] [ENTER] [(] X [^] 3 Y [−] Y [^] 3 X [)] [÷] 3 9 0 [ENTER]	
3. Change the graph format to display and label the axes.	[♦] F ▾ ▸ 2 ▾ ▸ 2 [ENTER]	
4. Select the ZoomStd viewing cube. This automatically graphs the equation. *As the TI-92 evaluates the equation (before displaying a graph), the "percent evaluated" is shown in the upper-left corner of the screen.*	[F2] 6	
5. Display the Window Editor, and change eyeθ° from 20 to 80. *This rotates the viewing angle by an additional 60° around the Z axis.*	[♦] [WINDOW] 8 0	
6. Regraph the equation and notice the rotation.	[♦] [GRAPH]	

Overview of Steps in Graphing 3D Equations

To graph 3D equations, use the same general steps used for y(x) functions as described in Chapter 3: Basic Function Graphing. Any differences that apply to 3D equations are described on the following pages.

Graphing 3D Equations

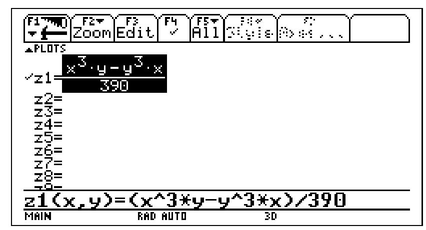
Set Graph mode ([MODE]) to 3D. Also set Angle mode, if necessary.



Define 3D equations on Y= Editor ([♦] [Y=]).

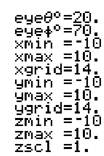
Tip: To turn off any stat data plots (Chapter 9), press [F5] 5 *or use* [F4] *to deselect them.*

Select ([F4]) which equation to graph. You can select only one 3D equation.



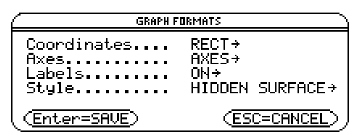
Note: For 3D graphs, the viewing window is called the viewing cube. [F2] Zoom *also changes the viewing cube.*

Define the viewing cube ([♦] [WINDOW]).



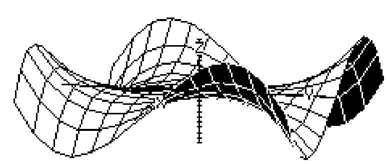
Tip: To help you see the orientation of 3D graphs, turn on Axes and Labels.

Change the graph format ([♦] F or [F1] 9), if necessary.



Note: Before displaying the graph, the screen shows the "percent evaluated."

Graph the selected equation ([♦] [GRAPH]).



Exploring the Graph

From the Graph screen, you can:

- Trace the equation.
- Use the [F2] Zoom toolbar menu to zoom in or out on a portion of the graph. Some of the menu items are dimmed because they are not available for 3D graphs.
- Use the [F5] Math toolbar menu to evaluate the equation at a specified point. Only 1:Value is available for 3D graphs.

Tip: You can also evaluate z(x,y) while tracing. Type the x value and press [ENTER]; *then type the y value and press* [ENTER].

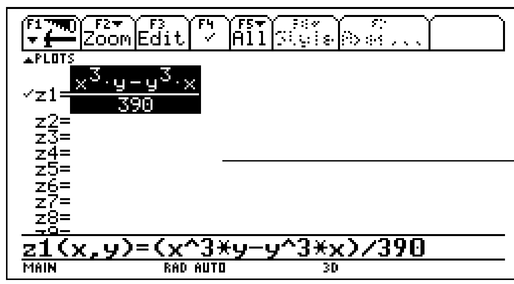
Differences in 3D and Function Graphing

This chapter assumes that you already know how to graph y(x) functions as described in Chapter 3: Basic Function Graphing. This section describes the differences that apply to 3D equations.

Setting the Graph Mode

Use [MODE] to set Graph = 3D before you define equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the *current* Graph mode setting only.

Defining 3D Equations on the Y= Editor



You can define 3D equations for z1(x,y) through z99(x,y).

__Tip:__ You can use the __Define__ command from the Home screen (see Appendix A) to define functions and equations for any graphing mode, regardless of the current mode.

The Y= Editor maintains an independent function list for each Graph mode setting. For example, suppose:

- In FUNCTION graphing mode, you define a set of y(x) functions. You change to 3D graphing mode and define a set of z(x,y) equations.
- When you return to FUNCTION graphing mode, your y(x) functions are still defined in the Y= Editor. When you return to 3D graphing mode, your z(x,y) equations are still defined.

Selecting the Display Style

Because you can graph only one 3D equation at a time, display styles are not available. On the Y= Editor, the [F6] Style toolbar menu is dimmed.

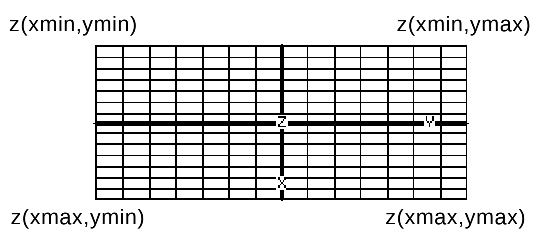
For 3D equations, however, you can use [♦] F or [F1] 9 to set the Style format to WIRE FRAME or HIDDEN SURFACE. Refer to “Changing the Axes and Style Formats” on page 257.

Window Variables

The Window Editor maintains an independent set of Window variables for each Graph mode setting (just as the Y= Editor maintains independent function lists). 3D graphs use the following Window variables.

Note: If you enter a fractional number for xgrid *or* ygrid*, it is rounded to the nearest whole number ≥ 1.*

Note: You cannot display tick marks on the X and Y axes.

Variable	Description
eyeθ°, eyeϕ°	Angles (always in degrees) used to view the graph. Refer to “Rotating and/or Elevating the Viewing Angle” on page 255.
xmin, xmax, ymin, ymax, zmin, zmax	Boundaries of the viewing cube.
xgrid, ygrid	The distance between xmin and xmax and between ymin and ymax is divided into the specified number of grids. The z(x,y) equation is evaluated at each grid point where the grid lines (or grid wires) intersect. The incremental value along x and y is calculated as: $\text{x increment} = \frac{\text{xmax} - \text{xmin}}{\text{xgrid}}$ $\quad \text{y increment} = \frac{\text{ymax} - \text{ymin}}{\text{ygrid}}$ The number of grid wires is xgrid + 1 and ygrid + 1. For example, when xgrid = 14 and ygrid = 14, the XY grid consists of 225 (15 × 15) grid points. 
zscl	Distance between tick marks on the Z axis.

Standard values (set when you select 6:ZoomStd from the F2 Zoom toolbar menu) are:

eyeθ° = 20.	xmin = ⁻10.	ymin = ⁻10.	zmin = ⁻10.
eyeϕ° = 70.	xmax = 10.	ymax = 10.	zmax = 10.
	xgrid = 14.	ygrid = 14.	zscl = 1.

Note: Increasing the grid variables decreases the graphing speed.

You may need to increase the standard values for the grid variables (xgrid, ygrid) to ensure that enough points are plotted.

Setting the Graph Format

The Axes and Style formats are specific to the 3D graphing mode. Refer to “Changing the Axes and Style Formats” on page 257.

Exploring a Graph

As in function graphing, you can explore a graph by using the following tools. Any displayed coordinates are shown in rectangular or cylindrical form as set in the graph format. (In 3D graphing, cylindrical coordinates are shown when you use [♦] F to set Coordinates = POLAR.)

Tool	For 3D Graphs:
Free-Moving Cursor	The free-moving cursor is not available.
[F2] Zoom	Works essentially the same as it does for function graphs, but remember that you are now using three dimensions instead of two. • Only the following zooms are available: 2:ZoomIn 5:ZoomSqr A:ZoomFit 3:ZoomOut 6:ZoomStd B:Memory C:SetFactors • Only x (xmin, xmax), y (ymin, ymax), and z (zmin, zmax, zscl) Window variables are affected. • The grid (xgrid, ygrid) and eye (eyeθ°, eyeφ°) Window variables are not affected unless you select 6:ZoomStd (which resets these variables to their standard values).
[F3] Trace	Lets you move the cursor along a grid wire from one grid point to the next on the 3D surface. • When you begin a trace, the cursor appears at the midpoint of the XY grid. • QuickCenter is available. At any time during a trace, regardless of the cursor’s location, you can press [ENTER] to center the viewing cube on the cursor. • Cursor movement is restricted in the x and y directions. You cannot move the cursor beyond the viewing cube boundaries set by xmin, xmax, ymin, and ymax.
[F5] Math	Only 1:Value is available for 3D graphs. This tool displays the z value for a specified x and y value. After selecting 1:Value, type the x value and press [ENTER]. Then type the y value and press [ENTER].

Tip: *Refer to “Moving the Cursor in 3D” on page 253.*

Tip: *During a trace, you can also evaluate z(x,y). Type the x value and press [ENTER]; then type the y value and press [ENTER].*

Moving the Cursor in 3D

When you move the cursor along a 3D surface, it may not be obvious why the cursor moves as it does. 3D graphs have two independent variables (x, y) instead of one, and the X and Y axes have a different orientation than other graphing modes.

How to Move the Cursor

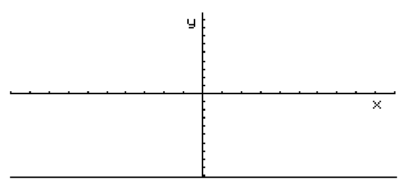
__Note:__ You can move the cursor only within the x and y boundaries set by Window variables xmin, xmax, ymin, and ymax.

On a 3D surface, the cursor always follows along a grid wire.

Cursor Key	Moves the cursor to the next grid point in the:
→	Positive x direction
←	Negative x direction
↑	Positive y direction
↓	Negative y direction

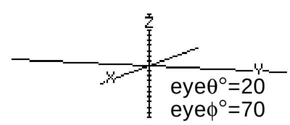
Although the rules are straightforward, the actual cursor movement can be confusing unless you know the orientation of the axes.

In 2D graphing, the X and Y axes always have the same orientation relative to the Graph screen.



__Tip:__ From the Y= Editor, Window Editor, or Graph screen, use [◆] F to show the axes and their labels.

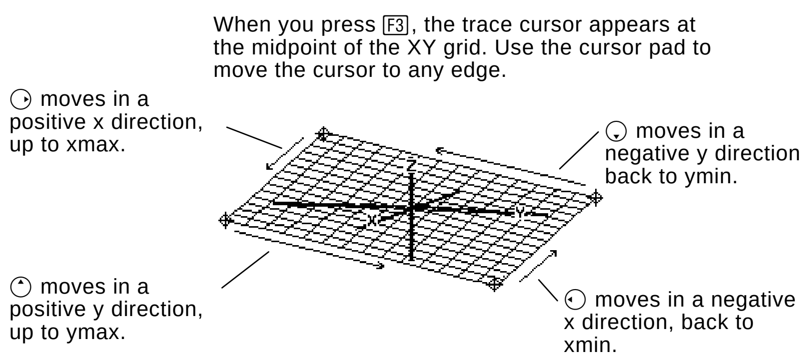
In 3D graphing, X and Y have a different orientation relative to the Graph screen. Also, you can rotate and/or elevate the viewing angle.



Simple Example of Moving the Cursor

__Tip:__ By displaying and labeling the axes, you can more easily see the pattern in the cursor movement.

The following graph shows a sloped plane that has the equation z1(x,y) = ⁻(x + y) / 2. Suppose you want to trace around the displayed boundary.



__Tip:__ To move grid points closer together, you can increase Window variables xgrid and ygrid.

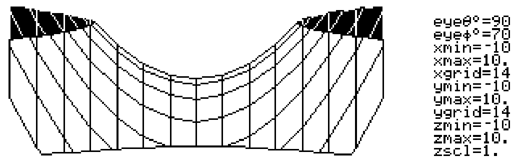
When the trace cursor is on an interior point in the displayed plane, the cursor moves from one grid point to the next along one of the grid wires. You cannot move diagonally across the grid.

Notice that the grid wires may not appear parallel to the axes.

Example of the Cursor on a Hidden Surface

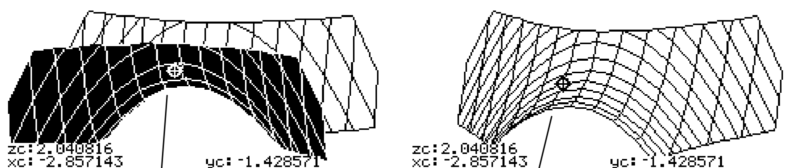
On more complex shapes, the cursor may appear as if it is not on a grid point. This is an optical illusion caused when the cursor is on a hidden surface.

For example, consider a saddle shape z1(x,y) = (x^2 – y^2) / 3. The following graph shows the view looking down the Y axis.



Now look at the same shape at 10° from the X axis (eyeθ° = 10).

__Tip:__ To cut away the front of the saddle in this example, set xmax=0 to show only negative x values.



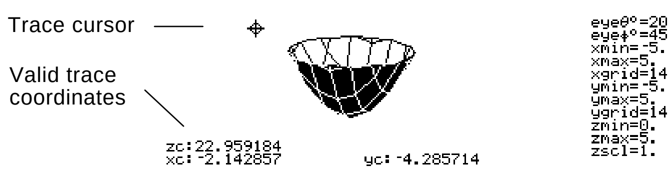
You can move the cursor so that it does not appear to be on a grid point.

If you cut away the front side, you can see the cursor is actually on a grid point on the hidden back side.

Example of an "Off the Curve" Cursor

Although the cursor can move only along a grid wire, you will see many cases where the cursor does not appear to be on the 3D surface at all. This occurs when the Z axis is too short to show z(x,y) for the corresponding x and y values.

For example, suppose you trace the paraboloid z(x,y) = x^2 + y^2 graphed with the indicated Window variables. You can easily move the cursor to a position such as:



__Tip:__ QuickCenter lets you center the viewing cube on the cursor's location. Simply press [ENTER].

Although the cursor is actually tracing the paraboloid, it appears off the curve because the trace coordinates:

- xc and yc are within the viewing cube.
  — but —
- zc is outside the viewing cube.

When zc is outside the z boundary of the viewing cube, the cursor is physically displayed at zmin or zmax (although the screen shows the correct trace coordinates).

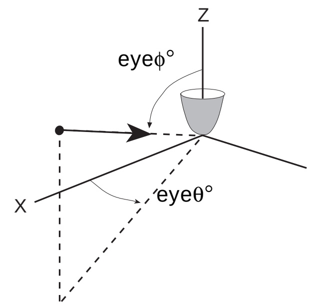
Rotating and/or Elevating the Viewing Angle

The Window variables eyeθ° and eyeϕ° let you view a 3D graph from any angle. These variables do not affect the graph's orientation along the axes; they affect only the angle used to view the graph.

How the Viewing Angle Is Measured

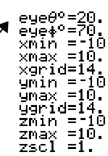
The viewing angle has two components:

- eyeθ° — angle in degrees from the positive X axis (rotation).
- eyeϕ° — angle in degrees from the positive Z axis (elevation).



You can enter negative angles as necessary. The default values are eyeθ° = 20 and eyeϕ° = 70.

On the Window Editor, always enter eyeθ° and eyeϕ° in degrees, regardless of the current angle mode.

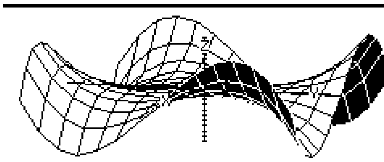
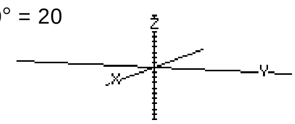
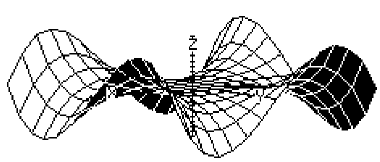
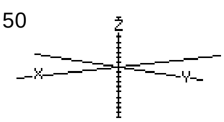
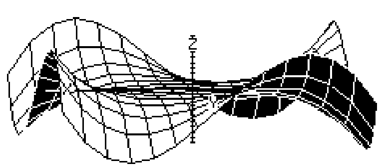
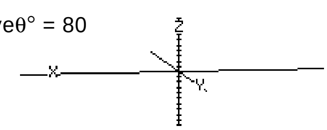


Do not enter a ° symbol. For example, type 20 and 70, not 20° and 70°.

Effect of Changing eyeθ°

The view on the Graph screen is always oriented along the viewing angle. From this point of view, you can change eyeθ° to rotate the viewing angle around the Z axis.

Note: *This example increments* eyeθ° *by* 30.

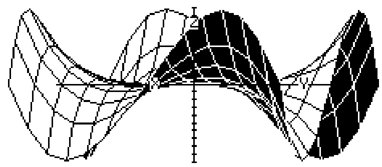
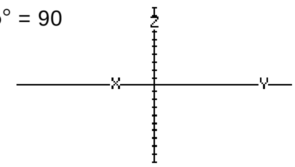
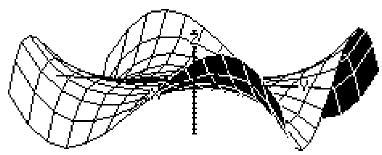
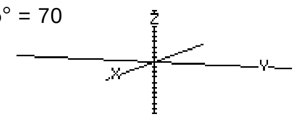
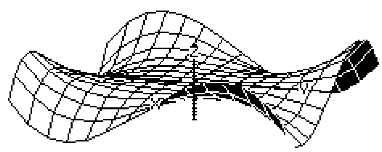
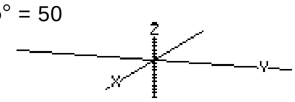
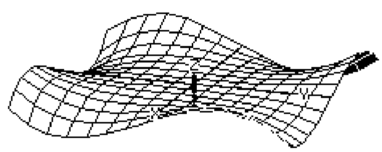
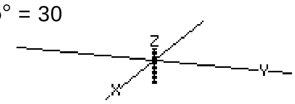
$z1(x,y) = (x^3y - y^3x) / 390$	In this example, eyeϕ° = 70
	eyeθ° = 20 
	eyeθ° = 50 
	eyeθ° = 80 

Effect of Changing eyeϕ°

By changing eyeϕ°, you can elevate your viewing angle above the XY plane.

If 90 < eyeϕ° < 270, the viewing angle is below the XY plane.

*Note: This example starts on the XY plane (*eyeϕ° = 90*) and decrements* eyeϕ° *by* 20 *to elevate the viewing angle.*

$z1(x,y) = (x^3y - y^3x) / 390$	In this example, eyeθ° = 20
	eyeϕ° = 90 
	eyeϕ° = 70 
	eyeϕ° = 50 
	eyeϕ° = 30 

From the Home Screen or a Program

The values used for eyeθ° and eyeϕ° are stored in the system variables eyeθ and eyeϕ (without the ° symbol). You can access or store to these variables as necessary.

To type ϕ (in eyeϕ), press [2nd] G F or press [2nd] [CHAR] and use the Greek menu.

Changing the Axes and Style Formats

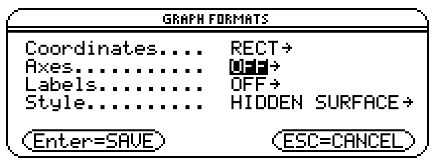
With its default settings, the TI-92 displays hidden surfaces on a 3D graph but does not display the axes. However, you can change the graph format at any time.

Displaying the GRAPH FORMATS Dialog Box

From the Y= Editor, Window Editor, or Graph screen:

- Press [F1] and select 9:Format.

  — or —
- Press [♦] F.



- The dialog box shows the current graph format settings.
- To exit without making a change, press [ESC].

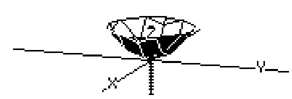
To change any of these settings, use the same procedure that you use to change other types of dialog boxes, such as the MODE dialog box.

Examples of Axes Settings

Tip: Setting Labels = ON is helpful when you display either type of 3D axes.

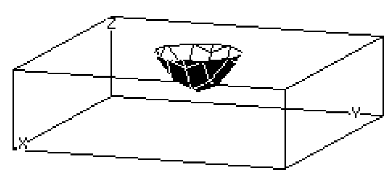
To display the valid Axes settings, highlight the current setting and press ►.

- AXES — Shows standard XYZ axes.



- BOX — Shows 3-dimensional box axes.

  The edges of the box are determined by the Window variables xmin, xmax, etc.



  In many cases, the origin (0,0,0) is inside the box, not at a corner.

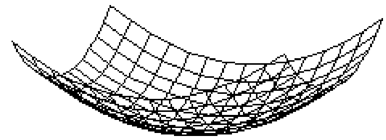
  For example, if xmin = ymin = zmin = ⁻10 and xmax = ymax = zmax = 10, the origin is at the center of the box.

Examples of Style Settings

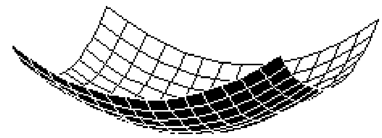
Tip: *WIRE FRAME is faster to graph and may be more convenient when you're experimenting with different shapes.*

To display the valid Style settings, highlight the current setting and press ⊙.

- WIRE FRAME — Shows the 3D shape as a transparent wire frame.



- HIDDEN SURFACES — Uses shading to differentiate the two sides of the 3D shape.



Be Aware of Possible Optical Illusions

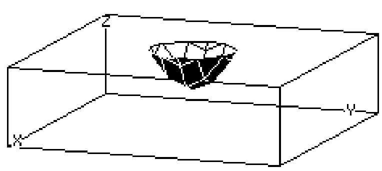
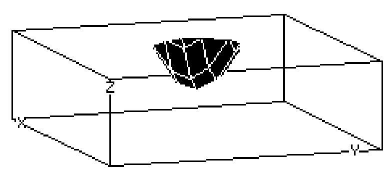
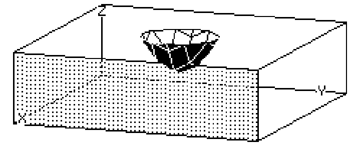
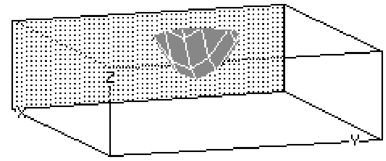
The eye angles used to view a graph (eyeθ° and eyeϕ° Window variables) can result in optical illusions that cause you to lose perspective on a graph.

Typically, most optical illusions occur when the eye angles are in a negative quadrant of the coordinate system.

Optical illusions may be more noticeable with box axes. For example, it may not be immediately obvious which is the "front" of the box.

Note: *These examples show the graphs as displayed on the screen.*

Note: *These examples use artificial shading (which is not displayed on the screen) to show the front of the box.*

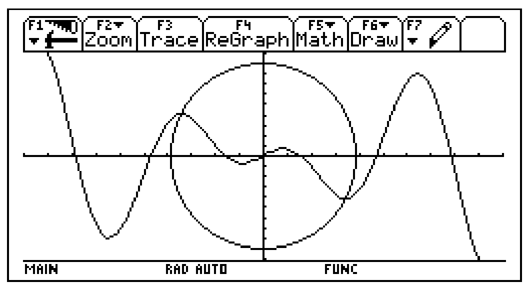
Looking down from above the XY plane	Looking up from below the XY plane
	
eyeθ° = 20 eyeϕ° = 70	eyeθ° = 20 eyeϕ° = 120
	

To minimize the effect of optical illusions, use the GRAPH FORMATS dialog box to set Style = HIDDEN SURFACE.

Additional Graphing Topics

15

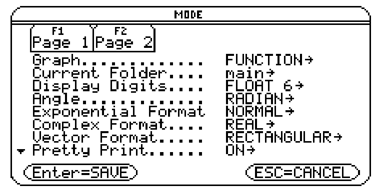
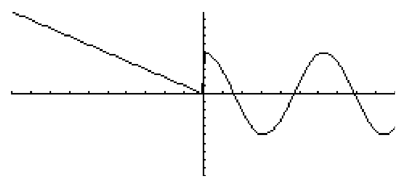
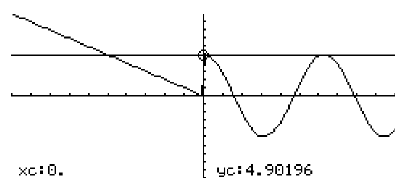
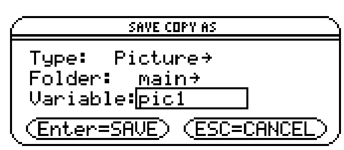
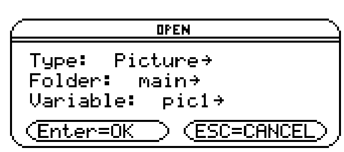
This chapter describes additional features that you can use to create graphs on the TI-92. This information generally applies to all Graph mode settings.



This chapter assumes that you already know the fundamental procedures for defining and selecting functions, setting Window variables, and displaying graphs as described in Chapter 3: Basic Function Graphing.

Preview of Additional Graphing Topics

From the Home screen, graph the piecewise defined function: y = −x when x < 0 and y = 5 cos(x) when x ≥ 0. Draw a horizontal line across the top of the cosine curve. Then save a picture of the displayed graph.

Steps	Keystrokes	Display
1. Display the MODE dialog box. For Graph mode, select FUNCTION. For Angle mode, select RADIAN.	[MODE] ▷ 1 ▽ ▽ ▽ ▷ 1 [ENTER]	
2. Display the Home screen. Use the **Graph** command and the **when** function to specify the piecewise defined function. *[F4] 2 selects* **Graph** *from the* Other *toolbar menu and automatically adds a space.*	[♦] [HOME] [F4] 2 W H E N [(] X [2nd] [<] 0 [,] [(-)] X [,] 5 [×] [COS] X [)] [)]	`Graph when(x<0,⁻x,5*cos(x))`
3. Execute the **Graph** command, which automatically displays the Graph screen. *The graph uses the current Window variables, which are assumed to be their standard values ([F2] 6) for this example.*	[ENTER]	
4. Draw a horizontal line across the top of the cosine curve. *After you press [F7] 5, the TI-92 remains in “line” mode until you select a different operation or press [ESC].*	[F7] 5 △ (until the line is positioned) [ENTER]	
5. Save a picture of the graph. Use PIC1 as the variable name for the picture. *Be sure to set Type = Picture. By default, it is set to GDB.*	[F1] 2 ▷ 2 ▽ ▽ P I C 1 [ENTER] [ENTER]	
6. Clear the drawn horizontal line. *You can also press [F4] to regraph.*	[F6] 1	
7. Open the saved picture variable to redisplay the graph with the line. *Be sure to set Type = Picture. By default, it is set to GDB.*	[F1] 1 ▷ 2 (if not already shown, also set Variable = pic1) [ENTER]	

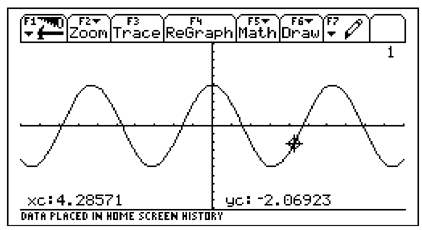
Collecting Data Points from a Graph

From the Graph screen, you can store sets of coordinate values and/or math results for later analysis. You can store the information as a single-row matrix (vector) on the Home screen or as data points in a system data variable that can be opened in the Data/Matrix Editor.

Collecting the Points

1. Display the graph. (This example shows y1(x)=5*cos(x).)
2. Display the coordinates or math results you want to collect.
3. Press [◆] H or [◆] D to save the information to the Home screen or the sysData variable, respectively.
4. Repeat the process as necessary.

Tip: To display coordinates or math results, trace a function with [F3] *or perform an* [F5] Math *operation (such as Minimum or Maximum). You can also use the free-moving cursor.*



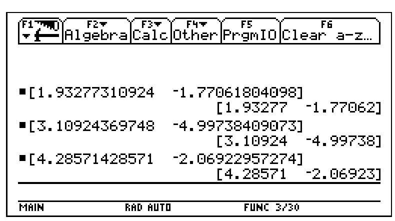
[◆] H

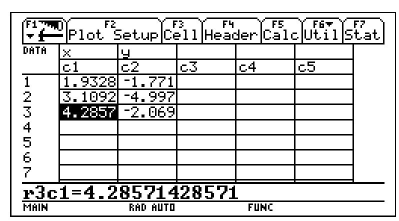
Displayed coordinates are added to he Home screen's history area (but not the entry line) as a ingle-row matrix or vector.

[◆] D

Displayed coordinates are stored in data variable named sysData, hich you can open in the ata/Matrix Editor.

Tip: Use a split screen to show a graph and the Home screen or Data/Matrix Editor at the same time.





Notes about SysData Variable

- When you press [◆] D:
  - If sysData does not exist, it is created in the MAIN folder.
  - If sysData already exists, new data is appended to the end of any existing data. Existing titles or column headers (for the affected columns) are cleared; titles are replaced with the applicable titles for the new data.
- The sysData variable can be cleared, deleted, etc., just as any other data variable. However, it cannot be locked.
- If the Graph screen contains a function or stat plot that references the current contents of sysData, [◆] D will not operate.

Graphing a Function Defined on the Home Screen

In many cases, you may create a function or expression on the Home screen and then decide to graph it. You can copy an expression to the Y= Editor, or graph it directly from the Home screen without using the Y= Editor.

What Is the "Native" Independent Variable?

On the Y= Editor, all functions must be defined in terms of the current graph mode's "native" independent variable.

Graph Mode	Native Independent Variable
Function	x
Parametric	t
Polar	θ
Sequence	n
3D	x, y

Copying from the Home Screen to the Y= Editor

If you have an expression on the Home screen, you can use any of the following methods to copy it to the Y= Editor.

Tip: Use [♦] C *or* [♦] V *to copy or paste, respectively, instead of* [F1] 5 *or* [F1] 6.

Tip: To copy an expression from the Home screen's history area to the entry line, use the auto-paste feature or copy and paste.

*Tip: **Define** is available from the Home screen's* [F4] *toolbar menu.*

Tip: [2nd] [RCL] *is useful if an expression is stored to a variable or function that does not correspond to the Y= Editor, such as* a1 *or* f1(x).

Method	Description
Copy and paste	1. Highlight the expression on the Home screen. Press [F1] and select 5:Copy. 2. Display the Y= Editor, highlight the desired function, and press [ENTER]. 3. Press [F1] and select 6:Paste. Then press [ENTER].
[STO▸]	Store the expression to a Y= function name. `2x^3+3x^2-4x+12→y1(x)` Use the complete function name: y1(x), not just y1.
Define command	Define the expression as a user-defined Y= function. `Define y1(x)=2x^3+3x^2-4x+12`
[2nd] [RCL]	If the expression is already stored to a variable: 1. Display the Y= Editor, highlight the desired function, and press [ENTER]. 2. Press [2nd] [RCL]. Type the variable name that contains the expression, and press [ENTER] twice. **Important:** To recall a function variable such as f1(x), type only f1, not the full function name. 3. Press [ENTER] to save the recalled expression in the Y= Editor's function list.

Graphing Directly from the Home Screen

The **Graph** command lets you graph an expression from the Home screen without using the Y= Editor. Unlike the Y= Editor, **Graph** lets you specify an expression in terms of any independent variable, regardless of the current graphing mode.

*Tip: **Graph** is available from the Home screen's [F4] toolbar menu.*

*Note: **Graph** uses the current Window variable settings.*

If the expression is in terms of:	Use the Graph command as shown in this example:
The native independent variable	`graph 1.25x*cos(x)` — For function graphing, x is the native variable.
A non-native independent variable	`graph 1.25a*cos(a),a` — Specify the independent variable; otherwise, you may get an error.

*Tip: To create a table from the Home screen, use the **Table** command. It is similar to **Graph**. Both share the same expressions.*

Graph does not work with sequence graphs. For parametric, polar, and 3D graphs, use the following variations.

In PARAMETRIC graphing mode:	**Graph** *xExpr*, *yExpr*, *t*
In POLAR graphing mode:	**Graph** *expr*, θ
In 3D graphing mode:	**Graph** *expr*, *x*, *y*

Graph does not copy the expression to the Y= Editor. Instead, it temporarily suspends any functions selected on the Y= Editor. You can trace, zoom, or show **Graph** expressions on the Table screen, just the same as Y= Editor functions.

Clearing the Graph Screen

Each time you execute **Graph**, the new expression is added to the existing ones. To clear the graphs:

- Execute the **ClrGraph** command (available from the Home screen's [F4] Other toolbar menu).
  — or —
- Display the Y= Editor. The next time you display the Graph screen, it will use the functions selected on the Y= Editor.

Extra Benefits of User-Defined Functions

You can define a user-defined function in terms of any independent variable. When you call that function, you should refer to it by using a different variable. For example:

Note: Use two or more character argument names (xx,yy,xtemp,...) to define function arguments to minimize the chance of a circular definition error when calling the function with common arguments (x,y,z,a,b,c,...)

Defined in terms of "aa".

```
define f1(aa)=1.25aa*cos(aa)
graph f1(x)
```

Refers to the function by using the native independent variable.

and:

```
define f1(aa)=1.25aa*cos(aa)
f1(x)→y1(x)
```

Graphing a Piecewise Defined Function

To graph a piecewise function, you must first define the function by specifying boundaries and expressions for each piece. The **when** function is extremely useful for two-piece functions. For three or more pieces, it may be easier to create a multi-statement, user-defined function.

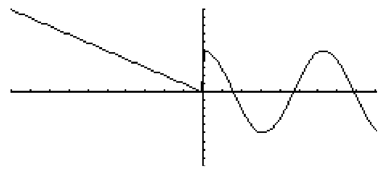
Using the When Function

To define a two-piece function, use the syntax:

when(*condition*, *trueExpression*, *falseExpression*)

For example, suppose you want to graph a function with two pieces.

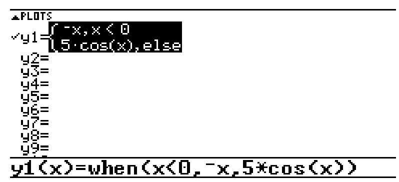
When:	Use expression:
$x < 0$	^{-}x
$x \ge 0$	$5\cos(x)$



In the Y= Editor:

The function is pretty printed in this form.

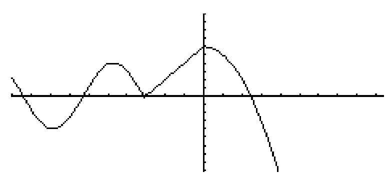
Enter the function in this form.



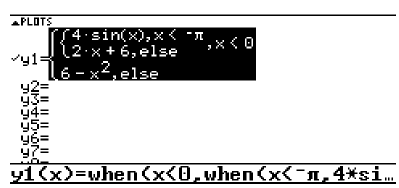
Tip: To enter **when**, *type it or use* [2nd] [CATALOG].

For three or more pieces, you can use nested **when** functions.

When:	Use expression:
$x < -\pi$	$4\sin(x)$
$x \ge -\pi$ and $x < 0$	$2x + 6$
$x \ge 0$	$6 - x^2$



In the Y= Editor:



where:

```
y1(x)=when(x<0,when(x<-π,4*sin(x),2x+6),6-x^2)
```

This nested function is in effect when x<0.

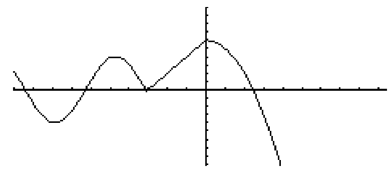
Nested functions quickly become complex and difficult to visualize.

Using a Multi-Statement, User-Defined Function

For three or more pieces, you may want to create a multi-statement, user-defined function.

For example, consider the previous three-piece function.

When:	Use expression:
$x < -\pi$	4 sin(x)
$x \geq -\pi$ and $x < 0$	2x + 6
$x \geq 0$	$6 - x^2$



Note: *For information about similarities and differences between functions and programs, refer to Chapter 17.*

A multi-statement, user-defined function can have many of the control and decision-making structures (**If**, **ElseIf**, **Return**, etc.) used in programming. When creating the structure of a function, it may be helpful to visualize it first in a block form.

```
Func
  If x<-π Then
  Return 4*sin(x)
ElseIf x>=-π and x<0 Then
  Return 2x+6
Else
  Return 6-x^2
EndIf
EndFunc
```

Func and **EndFunc** must begin and end the function.

For information about the individual statements, refer to Appendix A.

When entering a multi-statement function on the Y= Editor or Home screen, you must enter the entire function on a single line.

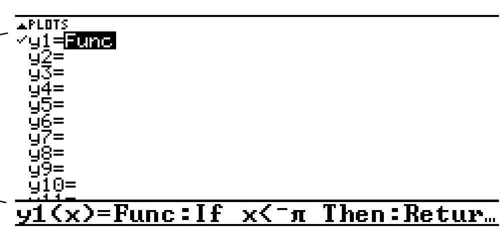
Use a colon (:) to separate each statement.

```
Func:If x<-π Then:Return 4*sin(x): ... :EndIf:EndFunc
```

On the Y= Editor:

Only “Func” is shown for a multi-statement function.

Enter a multi-statement function on one line. Be sure to include colons.



From the Home Screen or a Program

From the Home screen, you can also use the **Define** command to create a multi-statement, user-defined function. Refer to page 262 for other information on copying a function from the Home screen to the Y= Editor.

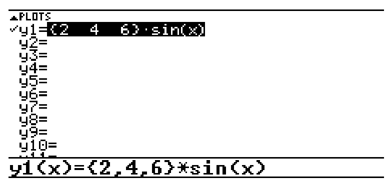
From the Program Editor (Chapter 17), you can create a user-defined function. For example, use the Program Editor to create a function named f1(xx). In the Y= Editor, set y1(x) = f1(x).

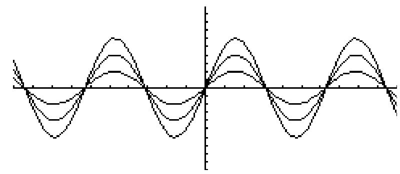
Graphing a Family of Curves

By entering a list in an expression, you can plot a separate function for each value in the list. (You cannot graph a family of curves in SEQUENCE or 3D graphing mode.)

Examples Using the Y= Editor

Enter the expression {2,4,6} sin(x) and graph the functions.

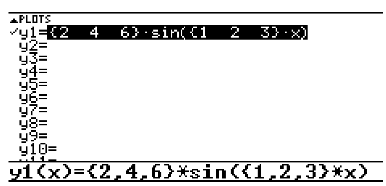


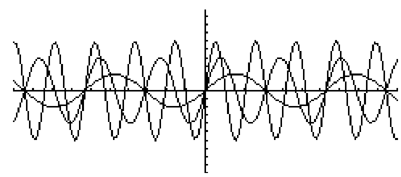


Graphs three functions:
2 sin(x), 4 sin(x), 6 sin(x)

Tip: Enclose list elements in braces ([2nd] [{] and [2nd] [}]) and separate them with commas.

Enter the expression {2,4,6} sin({1,2,3} x) and graph the functions.





Graphs three functions:
2 sin(x), 4 sin(2x), 6 sin(3x)

Note: The commas are shown in the entry line but not in the function list.

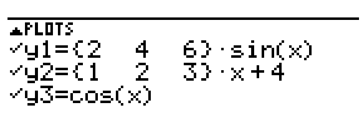
Example Using the Graph Command

Similarly, you can use the **Graph** command from the Home screen or a program as described on page 263.

```
graph {2,4,6}sin(x)
graph {2,4,6}sin({1,2,3}x)
```

Simultaneous Graphs with Lists

When the graph format is set for Graph Order = SIMUL, the functions are graphed in groups according to the element number in the list.



For these example functions, the TI-92 graphs three groups.

- 2 sin(x), x+4, cos(x)
- 4 sin(x), 2x+4
- 6 sin(x), 3x+4

The functions within each group are graphed simultaneously, but the groups are graphed sequentially.

Tip: To set graph formats, press [♦] F from the Y= Editor, Window Editor, or Graph screen.

When Tracing a Family of Curves

Pressing ⓓ or ⓤ moves the trace cursor to the next or previous curve in the same family before moving to the next or previous selected function.

Using the Two-Graph Mode

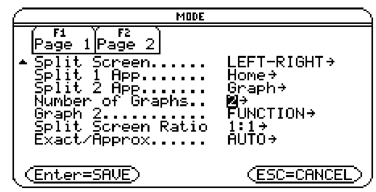
In two-graph mode, the TI-92's graph-related features are duplicated, giving you two independent graphing calculators. The two-graph mode is only available in split screen mode. For more information about split screens, refer to Chapter 5.

Setting the Mode

Several mode settings affect the two-graph mode, but only two settings are required. Both are on Page 2 of the MODE dialog box.

1. Press [MODE]. Then press [F2] to display Page 2.

2. Set the following required modes.

   - Split Screen = TOP-BOTTOM or LEFT-RIGHT
   - Number of Graphs = 2

3. Optionally, you can set the following modes.

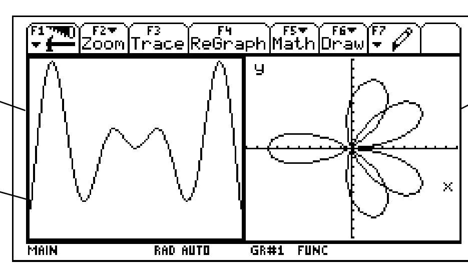
   Page 1: • Graph = Graph mode for top or left side of the split

   Page 2: • Split 1 App = application for top or left side
   - Split 2 App = application for bottom or right side
   - Graph 2 = Graph mode for bottom or right side
   - Split Screen Ratio = relative sizes of the two sides

4. Press [ENTER] to close the dialog box.

The Two-Graph Screen

A two-graph screen is similar to a regular split screen.



Independent Graph-Related Features

Both Graph 1 and Graph 2 have independent:

- Graph modes (FUNCTION, POLAR, etc.). Other modes such as Angle, Display Digits, etc., are shared and affect both graphs.
- Window Editor variables.
- Table setup parameters and Table screens.
- Graph formats (◆ F) such as Coordinates, Axes, etc.
- Graph screens.
- Y= Editors. However, both graphs share common function and stat plot definitions.

__Note:__ The Y= Editor is completely independent only when the two sides use different graphing modes (as described below).

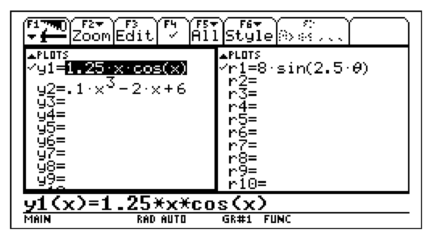
Independent graph-related applications (Y= Editor, Graph screen, etc.) can be displayed on both sides of the screen at the same time.

Non-graph-related applications (Home screen, Data/Matrix Editor, etc.) are shared and can be displayed on only one side at a time.

The Y= Editor in Two-Graph Mode

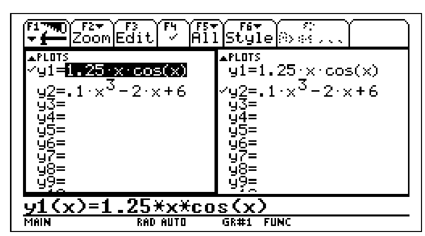
Even in two-graph mode, there is actually only one Y= Editor, which maintains a single function list for each Graph mode setting. However, if both sides use the same graphing mode, each side can select different functions from that single list.

- When both sides use different graphing modes, each side shows a different function list.



- When both sides use the same graphing mode, each side shows the same function list.
  - You can use F4 to select different functions and stat plots (indicated by ✓) for each side.
  - If you set a display style (F6) for a function, that style is used by both sides.

__Note:__ If you make a change on the active Y= Editor (redefine a function, change a style, etc.), that change is not reflected on the inactive side until you switch to it.



Suppose Graph 1 and Graph 2 are set for function graphing. Although both sides show the same function list, you can select (✓) different functions for graphing.

Review of Using a Split Screen

Note: *You can display non-graph-related applications (such as the Home screen) on only one side at a time.*

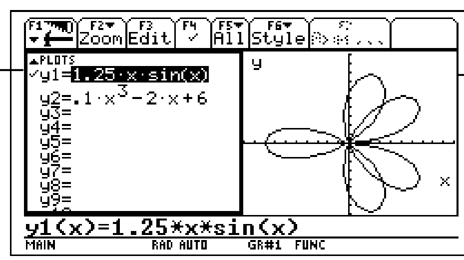
For more complete information about split screens, refer to Chapter 5.

- To switch from one graph side to the other, press [2nd] [⊞] (second function of [APPS]).
- To display different applications:
  - Switch to the applicable graph side and display the application as you normally would.
    — or —
  - Use [MODE] to change Split 1 App and/or Split 2 App.
- To exit two-graph mode:
  - Use [MODE] to set Number of Graphs = 1, or exit the split screen by setting Split Screen = FULL.
    — or —
  - Press [2nd] [QUIT] twice. This always exits a split screen and returns to a full-sized Home screen.

Remember that the Two Sides Are Independent

In two-graph mode, the two sides may appear to be related when, in fact, they are not. For example:

For Graph 1, the Y= Editor lists y(x) functions.



For Graph 2, the polar graph uses r(θ) equations that are not shown.

From the Home Screen or a Program

After the two-graph mode is set up, graph-related operations refer to the active graph side. For example:

```
10→xmax
```

affects either Graph 1 or Graph 2, depending on which is active when you execute the command.

To switch the active sides, press [2nd] [⊞] or use the **switch** function, **switch**(1) or **switch**(2).

Drawing a Function or Inverse on a Graph

For comparison purposes, you may want to draw a function over your current graph. Typically, the drawn function is some variation of the graph. You can also draw the inverse of a function. (These operations are not available for 3D graphs.)

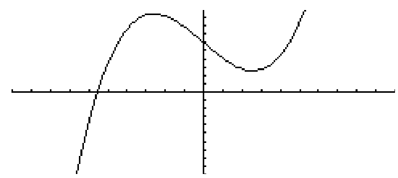
Drawing a Function, Parametric, or Polar Equation

Execute **DrawFunc**, **DrawParm**, or **DrawPol** from the Home screen or a program. You cannot draw a function or equation interactively from the Graph screen.

DrawFunc *expression*
DrawParm *expression1, expression2* [,*tmin*] [,*tmax*] [,*tstep*]
DrawPol *expression* [,*θmin*] [,*θmax*] [,*θstep*]

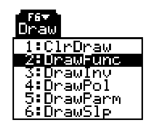
For example:

1. Define y1(x)=.1x^3−2x+6 on the Y= Editor, and graph the function.



Note: F6 2 *displays the Home screen and puts* **DrawFunc** *in the entry line.*

2. On the Graph screen, press F6 and select 2:DrawFunc.



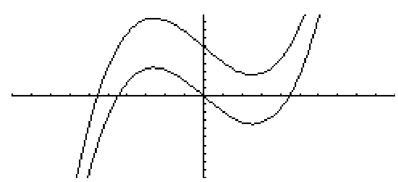
3. On the Home screen, specify the function to draw.

```
DrawFunc y1(x)-6
```

Tip: *To clear the drawn function, press* F4 *or press* F6 *and select* 1:ClrDraw.

4. Press ENTER to draw the function on the Graph screen.

   You cannot trace, zoom, or perform a math operation on a drawn function.



Drawing the Inverse of a Function

Execute **DrawInv** from the Home screen or a program. You cannot draw an inverse function interactively from the Graph screen.

DrawInv *expression*

For example, use the graph of y1(x)=.1x^3−2x+6 as shown above.

Note: F6 3 *displays the Home screen and puts* **DrawInv** *in the entry line.*

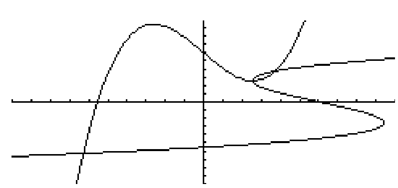
1. On the Graph screen, press F6 and select 3:DrawInv.
2. On the Home screen, specify the inverse function.

```
DrawInv y1(x)
```

Tip: *To clear the drawn inverse from the Graph screen, press* F4 *or press* F6 *and select* 1:ClrDraw.

3. Press ENTER.

   The inverse is plotted as (y,x) instead of (x,y).



Drawing a Line, Circle, or Text Label on a Graph

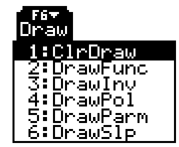
You can draw one or more objects on the Graph screen, usually for comparisons. For example, draw a horizontal line to show that two parts of a graph have the same y value. (Some objects are not available for 3D graphs.)

Clearing All Drawings

A drawn object is not part of the graph itself. It is drawn “on top of” the graph and remains on the screen until you clear it.

Tip: You can also enter **ClrDraw** *on the Home screen’s entry line.*

From the Graph screen:



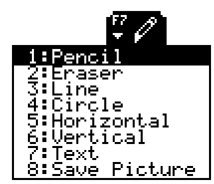
- Press [F6] and select 1:ClrDraw.

  — or —
- Press [F4] to regraph.

You can also do anything that causes the Smart Graph feature to redraw the graph (such as change the Window variables or deselect a function on the Y= Editor).

Drawing a Point or a Freehand Line

From the Graph screen:



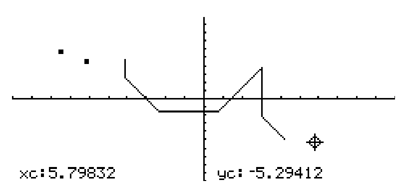
1. Press [F7] and select 1:Pencil.
2. Move the cursor to the applicable location.

Tip: When drawing a freehand line, you can move the cursor diagonally.

To draw a:	Do this:
Point (pixel-sized)	Press [ENTER].
Freehand line	Press and hold [hand key], and move the cursor to draw the line. To quit drawing the line, release [hand key].

***Note:** If you start drawing on a white pixel, the pencil draws a black point or line. If you start on a black pixel, the pencil draws a white point or line (which can act as an eraser).*

After drawing the point or line, you are still in “pencil” mode.



- To continue drawing, move the cursor to another point.
- To quit, press [ESC].

Erasing Individual Parts of a Drawing Object

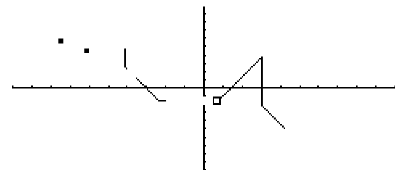
Note: These techniques also erase parts of graphed functions.

From the Graph screen:

1. Press [F7] and select 2:Eraser. The cursor is shown as a small box.
2. Move the cursor to the applicable location.

To erase:	Do this:
Area under the box	Press [ENTER].
Along a freehand line	Press and hold [hand], and move the cursor. To quit, release [hand].

After erasing, you are still in "eraser" mode.



- To continue erasing, move the box cursor to another location.
- To quit, press [ESC].

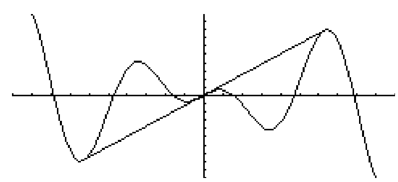
Drawing a Line Between Two Points

Tip: Use [2nd] to move the cursor in larger increments; [2nd] [right], etc.

From the Graph screen:

1. Press [F7] and select 3:Line.
2. Move the cursor to the 1st point, and press [ENTER].
3. Move to the 2nd point, and press [ENTER]. (As you move, a line extends from the 1st point to the cursor.)

After drawing the line, you are still in "line" mode.



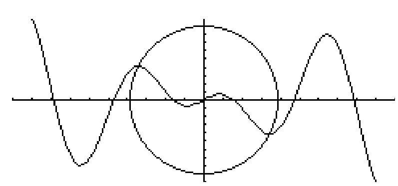
- To continue drawing another line, move the cursor to a new 1st point.
- To quit, press [ESC].

Drawing a Circle

Tip: Use [2nd] to move the cursor in larger increments; [2nd] [right], etc.

From the Graph screen:

1. Press [F7] and select 4:Circle.
2. Move the cursor to the center of the circle, and press [ENTER].
3. Move the cursor to set the radius, and press [ENTER].



Drawing a Horizontal or Vertical Line

From the Graph screen:

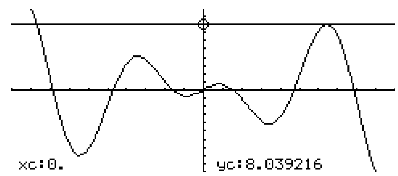
1. Press [F7] and select 5:Horizontal or 6:Vertical. A horizontal or vertical line and a flashing cursor are displayed on the screen.

   If the line is initially displayed on an axis, it may be difficult to see. However, you can easily see the flashing cursor.

Tip: Use [2nd] to move the cursor in larger increments; [2nd] [→], etc.

2. Use the cursor pad to move the line to the appropriate position. Then press [ENTER].

After drawing the line, you are still in "line" mode.



- To continue, move the cursor to another location.
- To quit, press [ESC].

Drawing a Tangent Line

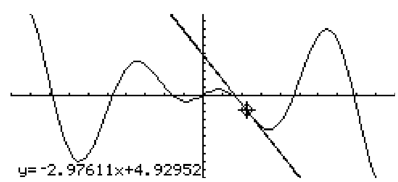
To draw a tangent line, use the [F5] Math toolbar menu instead of [F6] or [F7]. From the Graph screen:

1. Press [F5] and select A:Tangent.
2. As necessary, use [↓] and [↑] to select the applicable function.

Tip: To set the tangent point, you can also type its x value and press [ENTER].

3. Move the cursor to the tangent point, and press [ENTER].

   The tangent line is drawn, and its equation is displayed.



Drawing a Line Based on a Point and a Slope

To draw a line through a specified point with a specified slope, execute the **DrawSlp** command from the Home screen or a program. Use the syntax:

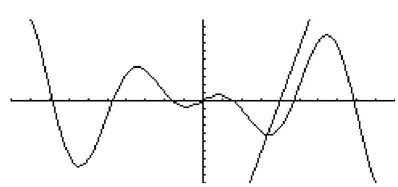
DrawSlp *x*, *y*, *slope*

You can also access **DrawSlp** from the Graph screen.

1. Press [F6] and select 6:DrawSlp. This switches to the Home screen and puts **DrawSlp** in the entry line.
2. Complete the command, and press [ENTER].

```
DrawSlp 4,0,6.37
```

   The TI-92 automatically switches to the Graph screen and draws the line.



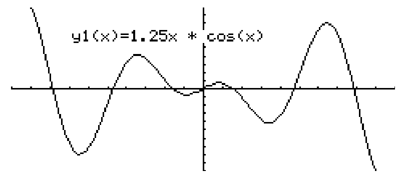
Typing Text Labels

Tip: The text cursor indicates the upper-left corner of the next character you type.

From the Graph screen:

1. Press [F7] and select 7:Text.
2. Move the text cursor to the location where you want to begin typing.
3. Type the text label.

After typing the text, you are still in "text" mode.



- To continue, move the cursor to another location.
- To quit, press [ENTER] or [ESC].

From the Home Screen or a Program

Commands are available for drawing any of the objects described in this section. There are also commands (such as **PxlOn**, **PxlLine**, etc.) that let you draw objects by specifying exact pixel locations on the screen.

For a list of the available drawing commands, refer to "Drawing on the Graph Screen" in Chapter 17.

Saving and Opening a Picture of a Graph

You can save an image of the current Graph screen in a PICTURE (or PIC) variable. Then, at a later time, you can open that variable and display the image. This saves the image only, not the graph settings used to produce it.

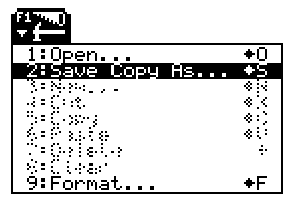
Saving a Picture of the Whole Graph Screen

A picture includes any plotted functions, axes, tick marks, and drawn objects. The picture does not include lower and upper bound indicators, prompts, or cursor coordinates.

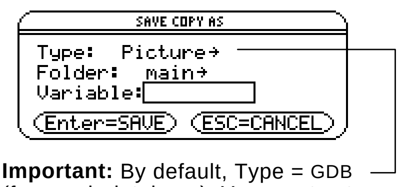
Display the Graph screen as you want to save it. Then:

Tip: You can press [♦] S instead of [F1] 2.

1. Press [F1] and select 2:Save Copy As.



2. Specify the type (Picture), folder, and a unique variable name.
3. Press [ENTER]. After typing in an input box such as Variable, you must press [ENTER] twice.



Important: By default, Type = GDB (for graph database). You must set Type = Picture.

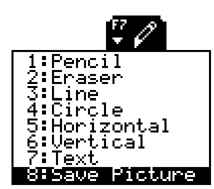
Saving a Portion of the Graph Screen

You can define a rectangular box that encloses only the portion of the Graph screen that you want to save.

1. Press [F7] and select 8:Save Picture.

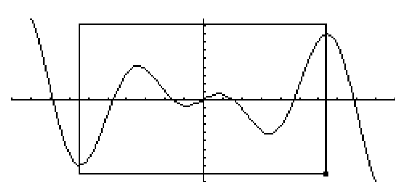
   A box is shown around the outer edge of the screen.

Note: You cannot save a portion of a 3D graph.

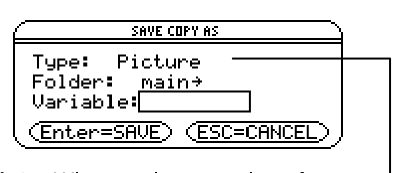


Tip: Use ↓ and ↑ to move the top or bottom, and use → and ← to move the sides.

2. Set the 1st corner of the box by moving its top and left sides. Then press [ENTER].
3. Set the 2nd corner by moving the bottom and right sides. Then press [ENTER].



4. Specify the folder and a unique variable name.
5. Press [ENTER]. After typing in an input box such as Variable, you must press [ENTER] twice.



Note: When saving a portion of a graph, Type is automatically fixed as Picture.

Opening a Graph Picture

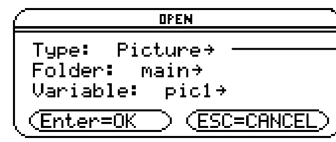
When you open a graph picture, it is superimposed over the current Graph screen. To display only the picture, use the Y= Editor to deselect any other functions before opening the graph picture.

From the Graph screen:

Tip: You can press [♦] O instead of [F1] 1.

1. Press [F1] and select 1:Open.
2. Select the type (Picture), folder, and variable that contain the graph picture you want to open.
3. Press [ENTER].

Note: If a variable name is not shown on the dialog box, there are no graph pictures in the folder.



Important: By default, Type = GDB (for graph database). Be sure to set Type = Picture.

A graph picture is a drawing object. You cannot trace any curve on a picture.

For Pictures Saved from a Portion of the Graph Screen

When you press [F1] and select 1:Open, the picture is superimposed starting at the upper-left corner of the Graph screen. If the picture was saved from a portion of the Graph screen (page 275), it may appear shifted from the underlying graph.

To specify which screen pixel to use as the upper-left corner, you can use the commands listed in "From a Program or the Home Screen" below.

Deleting a Graph Picture

Unwanted Picture variables take up calculator memory. To delete a variable, use the VAR-LINK screen ([2nd] [VAR-LINK]) as described in Chapter 18.

From a Program or the Home Screen

To save (store) and open (recall) a graph picture, use the **StoPic**, **RclPic**, **AndPic**, **XorPic**, and **RplcPic** commands as described in Appendix A.

To display a series of graph pictures as an animation, use the **CyclePic** command. For an example, refer to page 277.

Animating a Series of Graph Pictures

As described earlier in this chapter, you can save a picture of a graph. By using the **CyclePic** command, you can flip through a series of graph pictures to create an animation.

CyclePic Command

Before using **CyclePic**, you must have a series of graph pictures that have the same base name and are sequentially numbered starting with 1 (such as pic1, pic2, pic3, . . .).

To cycle the pictures, use the syntax:

CyclePic *picNameString*, *n* [,*wait*] [,*cycles*] [,*direction*]

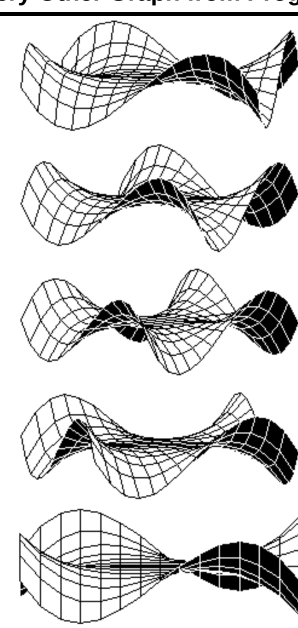
- *direction*: 1 = forward/circular cycle; ⁻1= forward/backward
- *cycles*: # of times to repeat cycle
- *wait*: seconds between pictures
- *n*: # of pictures to cycle
- *picNameString*: base name of pictures in quotes, such as "pic"

Example

This example program (named **cyc**) generates 10 views of a 3D graph, with each view rotated 10° further around the Z axis. For information about each command, refer to Appendix A. For information about using the Program Editor, refer to Chapter 17.

Program Listing | **Every Other Graph from Program**

```
:cyc()
:Prgm
:local i
:©Set mode and Window variables
:setMode("graph","3d")
:70→eyeφ
:⁻10→xmin
:10→xmax
:14→xgrid
:⁻10→ymin
:10→ymax
:14→ygrid
:⁻10→zmin
:10→zmax
:1→zscl
:©Define the function
:(x^3*y-y^3*x)/390→z1(x,y)
:©Generate pics and rotate
:For i,1,10,1
:  i*10→eyeθ
:  DispG
:  StoPic #("pic" & string(i))
:EndFor
:©Display animation
:CyclePic "pic",10,.5,5,⁻1
:EndPrgm
```



Comments start with ©. For ©, press [2nd] X.

For φ, press [2nd] G F (or press [2nd] [CHAR] and use the Greek menu).

For #, press [2nd] T; for &, press [2nd] H.

***Note:** Due to its complexity, this program takes several minutes to run.*

After entering this program on the Program Editor, go to the Home screen and enter cyc().

Saving and Opening a Graph Database

A graph database is the set of all elements that define a particular graph. By saving a graph database as a GDB variable, you can recreate that graph at a later time by opening its stored database variable.

Elements in a Graph Database

Note: In two-graph mode, the elements for both graphs are saved in a single database.

A graph database consists of:

- Mode settings ([MODE]) for Graph, Angle, Complex Format, and Split Screen (only if you are using the two-graph mode).
- All functions in the Y= Editor ([♦] [Y=]), including display styles and which functions are selected.
- Table parameters ([♦] [TblSet]), Window variables ([♦] [WINDOW]), and graph formats ([♦] F or [F1] 9).

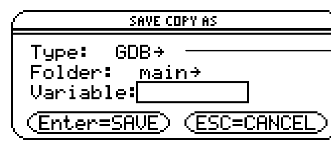
A graph database does not include drawn objects or stat plots.

Saving the Current Graph Database

Tip: You can press [♦] S instead of [F1] 2.

From the Y= Editor, Window Editor, Table screen, or Graph screen:

1. Press [F1] and select 2:Save Copy As.
2. Specify the folder and a unique variable name.
3. Press [ENTER]. After typing in an input box such as Variable, you must press [ENTER] twice.



Note: If you start from the Graph screen, be sure to use Type=GDB.

Opening a Graph Database

Tip: You can press [♦] O instead of [F1] 1.

Caution: When you open a graph database, all information in the current database is replaced. You may want to store the current graph database before opening a stored database.

From the Y= Editor, Window Editor, Table screen, or Graph screen:

1. Press [F1] and select 1:Open.
2. Select the folder and variable that contain the graph database you want to open.
3. Press [ENTER].

Note: If you start from the Graph screen, be sure to use Type=GDB.

Deleting a Graph Database

Unused GDB variables take up calculator memory. To delete them, use the VAR-LINK screen ([2nd] [VAR-LINK]) described in Chapter 18.

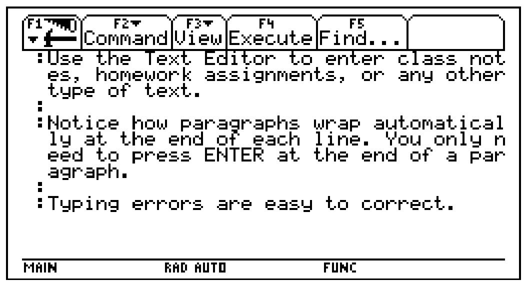
From a Program or the Home Screen

You can save (store) and open (recall) a graph database by using the **StoGDB** and **RclGDB** commands as described in Appendix A.

Text Editor

16

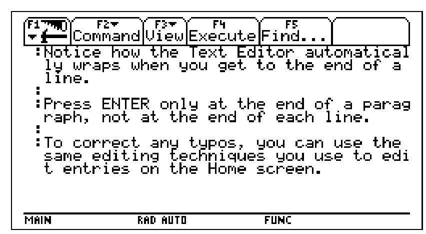
This chapter shows you how to use the Text Editor to enter and edit text. Entering text is simple; just begin typing. To edit text, you can use the same techniques that you use to edit information on the Home screen.



Each time you start a new text session, you must specify the name of a text variable. After you begin a session, any text that you type is stored automatically in the associated text variable. You do not need to save a session manually before leaving the Text Editor.

Preview of Text Operations

Start a new Text Editor session. Then practice using the Text Editor by typing whatever text you want. As you type, practice moving the text cursor and correcting any typos you may enter.

Steps	Keystrokes	Display
1. Start a new session on the Text Editor.	[APPS] 9 3	
2. Create a text variable called TEST, which will automatically store any text you enter in the new session. *Use the* MAIN *folder, shown as the default on the* NEW *dialog box.* *After typing in an input box such as* Variable, *you must press* [ENTER] *twice.*	↓ T E S T [ENTER] [ENTER]	
3. Type some sample text. *Practice editing your text by using:* • *The cursor pad to move the text cursor.* • [←] *or* [♦] [←] *to delete the character to the left or right of the cursor, respectively.*	type anything you want	
4. Leave the Text Editor and display the Home screen. *Your text session was stored automatically as you typed. Therefore, you do not need to save the session manually before exiting the Text Editor.*	[♦] [HOME]	
5. Return to the current session on the Text Editor.	[APPS] 9 1	
6. Notice that the displayed session is exactly the same as you left it.		

Starting a Text Editor Session

Each time you start the Text Editor, you can start a new text session, resume the current session (the session that was displayed the last time you used the Text Editor), or open a previous session.

Starting a New Session

1. Press [APPS] and then select 9:Text Editor.

2. Select 3:New.

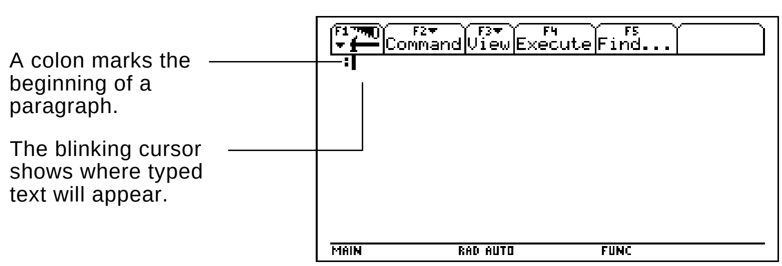
   The NEW dialog box is displayed.

3. Specify a folder and text variable that you want to use to store the new session.

Item	Description
Type	Automatically set as Text and cannot be changed.
Folder	Shows the folder in which the text variable will be stored. For information about folders, refer to Chapter 10. To use a different folder, press ⊙ to display a menu of existing folders. Then select a folder.
Variable	Type a variable name. If you specify a variable that already exists, an error message will be displayed when you press [ENTER]. When you press [ESC] or [ENTER] to acknowledge the error, the NEW dialog box is redisplayed.

4. Press [ENTER] (after typing in an input box such as Variable, you must press [ENTER] twice) to display an empty Text Editor screen.

__Note:__ Your session is saved automatically as you type. You do not need to save a session manually before leaving the Text Editor, starting a new session, or opening a previous one.



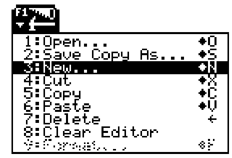
You can now use the Text Editor as described in the remaining sections of this chapter.

Resuming the Current Session

You can leave the Text Editor and go to another application at any time. To return to the session that was displayed when you left the Text Editor, press [APPS] 9 and select 1:Current.

Starting a New Session from the Text Editor

To leave the current Text Editor session and start a new one:



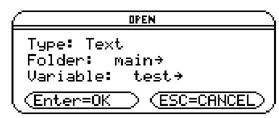
1. Press [F1] and select 3:New. (You can press [♦] N instead of using the [F1] toolbar menu.)
2. Specify a folder and text variable for the new session.
3. Press [ENTER] twice.

Opening a Previous Session

You can open a previous Text Editor session at any time.

1. From within the Text Editor, press [F1] and select 1:Open. (You can press [♦] O instead of using the [F1] toolbar menu.)
   — or —
   From any application, press [APPS] 9 and select 2:Open.

__Note:__ By default, Variable *shows the first existing text variable in alphabetic order.*



2. Select the applicable folder and text variable.
3. Press [ENTER].

Copying a Session

In some cases, you may want to copy a session so that you can edit the copy while retaining the original.

1. Display the session you want to copy.
2. Press [F1] and select 2:Save Copy As. (You can press [♦] S instead of using the [F1] toolbar menu.)
3. Specify the folder and text variable for the copied session.
4. Press [ENTER] twice.

Note about Deleting a Session

Because all Text Editor sessions are saved automatically, you can accumulate quite a few previous sessions, which take up memory storage space.

To delete a session, use the VAR-LINK screen ([2nd] [VAR-LINK]) to delete that session's text variable. For information about VAR-LINK, refer to Chapter 18.

Entering and Editing Text

After beginning a Text Editor session, you can enter and edit text. In general, use the same techniques that you have already used to enter and edit information on the Home screen's entry line.

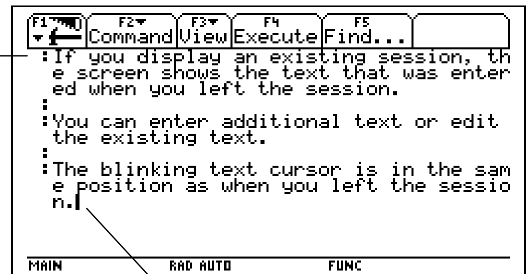
Typing Text

When you create a new Text Editor session, you see an empty screen. When you open a previous session or return to the current session, you see the existing text for that session.

Note: Use the cursor pad to scroll through a session or position the text cursor for entering or editing text.

All text paragraphs begin with a space and a colon.

The beginning space is used in command scripts and lab reports.



Blinking text cursor

Type your text just as you would in a word processor.

- You do not need to press ENTER at the end of each line. When you reach the end of a line, the next character you type automatically wraps to the next line.
- Press ENTER only when you want to start a new paragraph.

As you reach the bottom of the screen, previous lines scroll off the top of the screen.

Typing Uppercase Letters with Shift (↑) or Caps Lock

To:	Press:
Type a single uppercase letter	↑ and then the letter
Turn Caps Lock on or off	2nd [CAPS]

Deleting Characters

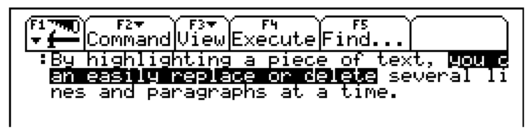
To delete:	Press:
The character to the left of the cursor	← or F1 7
The character to the right of the cursor	♦ ←
All characters to the right of the cursor through the end of the paragraph	CLEAR
All characters in the paragraph (regardless of the cursor's position in that paragraph)	CLEAR CLEAR

Note: *If there are no characters to the right of the cursor,* CLEAR *erases the entire paragraph.*

Replacing or Deleting Highlighted Text

Tip: To remove highlighting without replacing or deleting, move the cursor.

To:	Do this:
Highlight text	1. Move the cursor to the beginning or end of the text. 2. Hold [↑] and press: • ◀ or ▶ to highlight characters to the left or right of the cursor, respectively. • ▼ or ▲ to highlight all characters up to the cursor position on the next or previous line, respectively.



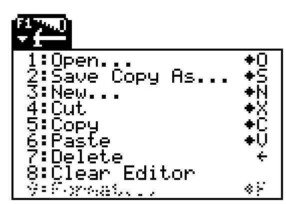
To:	Do this:
Replace highlighted text	Type the new text.
Delete highlighted text	Press [←].

Cutting, Copying, and Pasting Text

Tip: You can press [◆] X, [◆] C, and [◆] V to cut, copy, and paste without having to use the [F1] toolbar menu.

Cutting and copying both place highlighted text into the TI-92's clipboard. Cutting deletes the text from its current location (used to move text) and copying leaves the text.

1. Highlight the text you want to move or copy.
2. Press [F1].
3. Select the applicable menu item.
   - To move the text, select 4:Cut.
     — or —
   - To copy the text, select 5:Copy.



4. Move the text cursor to the location where you want to insert the text.
5. Press [F1] and then select 6:Paste.

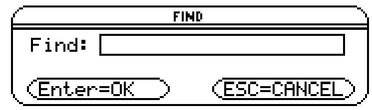
You can use this general procedure to cut , copy, and paste text:

- Within the same text session.
- From one text session to another. After cutting or copying text in one session, open the other session and then paste the text.
- From a text session to a different application. For example, you can paste the text into the Home screen's entry line.

Finding Text

From the Text Editor:

1. Place the text cursor at any location preceding the text you want to search for. All searches start at the current cursor location.
2. Press [F5].



3. Type the search text.

   The search is not case sensitive. For example: CASE, case, and Case have the same effect.
4. Press [ENTER] twice.

Tip: The FIND dialog box retains the last search text you entered. You can type over it or edit it.

If the search text is:	The cursor:
Found	Moves to beginning of the search text.
Not found	Does not move.

Inserting or Overtyping a Character

By default, the TI-92 is in insert mode. To toggle between insert and overtype mode, press [2nd] [INS].

Tip: Look at the shape of the cursor to see if you're in insert or overtype mode.

If the TI-92 is in:	The next character you type:
Insert mode Thin cursor between characters	Will be inserted at the cursor.
Overtype mode Cursor highlights a character	Will replace the highlighted character.

Clearing the Text Editor

To erase all existing paragraphs and display an empty text screen, press [F1] and then select 8:Clear Editor.

Entering Special Characters

You can use the **CHAR** menu to select any special character from a list. You can also type certain commonly used special characters as second functions of the QWERTY keyboard. To see which special characters are available from the keyboard, you can display a map that shows the characters and their corresponding keys.

Using the CHAR Menu

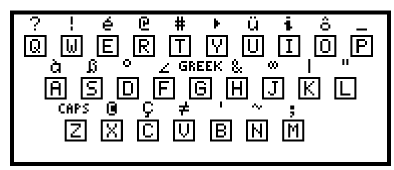
1. Press [2nd] [CHAR].
2. Select the applicable category.

   A menu lists the characters in that category.
3. Select a character. You may need to scroll through the menu.

↓ indicates that you can scroll.

Displaying the QWERTY Keyboard Map

Press [♦] K to display the map.



These characters are second functions of the QWERTY keyboard. Some are marked on the keyboard, but most are not.

The map shows:

- Special symbols — ?, !, #, &, etc.
- Accent marks — é, ü, ô, à, ç, and ~
- Greek letters — accessed by pressing [2nd] G (as described later in this section)

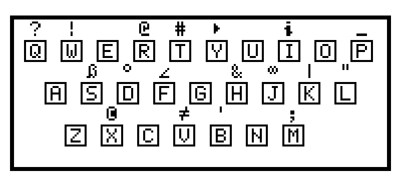
The map also shows [2nd] [CAPS], which turns Caps Lock on and off.

Typing Special Symbols from the Keyboard

Press [2nd] and then the key for the symbol.

For example:
[2nd] T displays #.

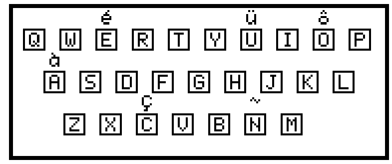
These special symbols are not affected by whether Caps Lock is on or off.



Note: To help you find the applicable keys, this map shows only the special symbols.

Typing Accent Marks from the Keyboard

Pressing an accent mark key does not display an accented letter. The accent mark will be added to the *next* letter you press.



Note: To help you find the applicable keys, this map shows only the accent mark keys.

1. Press [2nd] and then the key for the accent mark.
2. Press the key for the letter you want to accent.
   - You can accent lowercase and uppercase letters.
   - An accent mark can be added to only those letters that are valid for that mark.

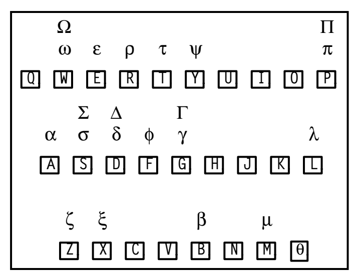
Accent Mark	Valid Letters (lowercase or uppercase)	Examples
´	A, E, I, O, U, Y	é, É
¨	A, E, I, O, U, y (but not Y)	ü, Ü
^	A, E, I, O, U	ô, Ô
`	A, E, I, O, U	à, À
Ç	C	ç, Ç
~	A, O, N	ñ, Ñ

Typing Greek Letters from the Keyboard

Tip: For θ or π, press [θ] or [2nd] [π].

1. Press [2nd] G to access the Greek character set.
2. Press the key for the applicable Greek letter.
   - Several keys let you access lowercase and uppercase Greek letters. For example:

     [2nd] G W displays ω.
     [2nd] G [↑] W displays Ω.
   - If you press a key combination that does not access a Greek letter, you get the normal letter for that key.



Note: The TI-92 does not display this map of Greek letters. A map is shown in this guidebook for reference purposes only.

For a List of All Special Characters

For a list of all special characters, refer to Appendix B.

Entering and Executing a Command Script

By using a command script, you can use the Text Editor to type a series of command lines that can be executed at any time on the Home screen. This lets you create interactive example scripts in which you predefine a series of commands and then execute them individually.

Inserting a Command Mark

__Note:__ This does not insert a new line for the command, it simply marks an existing line as a command line.

In the Text Editor:

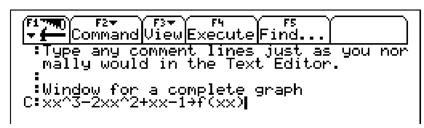
1. Place the cursor on the line for the command.
2. Press [F2] to display the Command toolbar menu.

3. Select 1:Command.

   "C" is displayed at the beginning of the text line (to the left of the colon).

__Tip:__ You can mark a line as a command either before or after typing the command on that line.

4. Type a command just as you would on the Home screen.



   The line can contain only the command, with no additional text.

   You can type multiple commands on the same line if you type a colon to separate the commands.

Deleting a Command Mark

This deletes only the "C" mark; it does not delete the command text itself.

1. Place the cursor anywhere on the marked line.
2. Press [F2] and select 4:Clear command.

Executing a Command

To execute a command, you must first mark the line with a "C". If you execute a line that is not marked with "C", it will be ignored.

1. Place the cursor anywhere on the command line.
2. Press [F4].

__Tip:__ To examine the result on the Home screen, press [♦] [HOME] or use a split screen.

The command is copied to the entry line on the Home screen and executed. The Home screen is displayed temporarily during execution, and then the Text Editor is redisplayed.

After execution, the cursor moves to the next line in the script so that you can continue to execute a series of commands.

Splitting the Text Editor/ Home Screen

With a split screen, you can view your command script and see the result of an executed command at the same time.

To:	Press:
Split the screen	[F3] and select 1:Script view.
Return to a full screen Text Editor	[F3] and select 2:Clear split.

You can also use [MODE] to set up a split screen manually. However, [F3] sets up a Text Editor/Home screen split much easier than [MODE].

- The active application is indicated by a thick border. (By default, the Text Editor is the active application.)
- To switch between the Text Editor and the Home screen, press [2nd] [⇔] (second function of [APPS]).

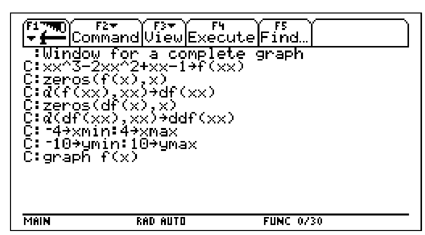
Creating a Script from Your Home Screen Entries

From the Home screen, you can save all the entries in the history area to a text variable. The entries are automatically saved in a script format so that you can open the text variable in the Text Editor and execute the entries as commands.

For information, refer to "Saving the Home Screen Entries as a Text Editor Script" in Chapter 10.

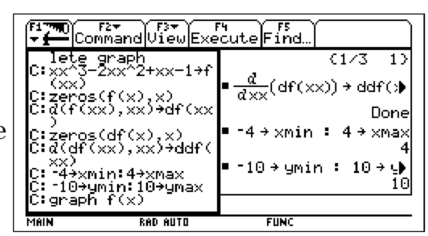
Example

1. Type your script. Press [F2] and select 1:Command to mark the command lines.
2. Press [F3] and select 1:Script view.



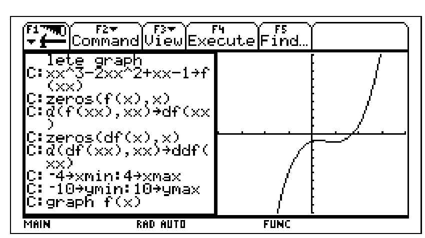
Note: Some commands take longer to execute. Wait until the Busy indicator disappears before pressing [F4] again.

3. Move the cursor to the first command line. Then press [F4] to execute the command.
4. Continue using [F4] to execute each command, but stop just before executing the **Graph** command.



*Note: In this example, the **Graph** command displays the Graph screen in place of the Home screen.*

5. Execute the **Graph** command.
6. Press [F3] and select 2:Clear split to return to a full screen Text Editor.



Creating a Lab Report

If you have a TI-GRAPH LINK™, an optional accessory that lets the TI-92 communicate with a personal computer, you can create lab reports. Use the Text Editor to write a report, which can include print objects. Then use the TI-GRAPH LINK to print the report on the printer attached to the computer.

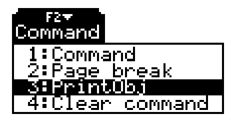
Print Objects

In the Text Editor, you can specify a variable name as a print object. When you print the report by using the TI-GRAPH LINK, the TI-92 substitutes the contents of the variable (an expression, picture, list, etc.) in place of the variable name.

Inserting a Print Object Mark

In the Text Editor:

1. Place the cursor on the line for the print object.
2. Press [F2] to display the Command toolbar menu.



***Note:** This does not insert a new line for the print object, it simply marks an existing line as a print object.*

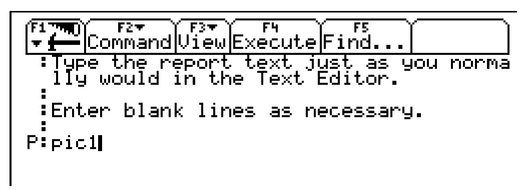
3. Select 3:PrintObj.

   "P" is displayed at the beginning of the text line (to the left of the colon).

***Tip:** You can mark a line as a print object either before or after typing a variable name on that line.*

4. Type the name of the variable that contains the print object.

   The line can contain only the variable name, with no additional text.



Inserting a Page Break Mark

When you print a lab report, page breaks occur automatically at the bottom of each printed page. However, you can manually force a page break at any line.

1. Place the cursor on the line that you want to print on the top of the next page. (The line can be blank or you can enter text on it.)
2. Press [F2] and select 2:Page break.

A "⤒" is displayed at the beginning of the line (to the left of the colon).

Deleting a Print Object or Page Break Mark

This deletes only the "P" or "⤒" mark; it does not delete any text that is on the line.

1. Place the cursor anywhere on the marked line.
2. Press [F2] and select 4:Clear command.

Printing the Report

General Steps	For Detailed Information
1. Connect the TI-92 to your computer via the TI-GRAPH LINK.	Refer to the manual that came with your TI-GRAPH LINK.
2. Use the TI-92's VAR-LINK screen to send the text variable that contains your lab report.	Refer to Chapter 18 of this guidebook.

Example

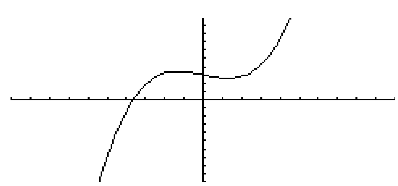
Assume you have stored:

- A function as y1(x) (specify y1, not y1(x)).
- A graph picture as pic1.
- Applicable information in variables der and sol.

When you print the lab report, the contents of the print objects are printed in place of their variable names.

```
 :My homework assignment was to study th
  e function:
 :
P:y1
 :
 :There were three parts to the assignme
  nt.
 :
 :1. Graph the function.
P:pic1
 :
 :2. Find its derivative.
P:der
 :
 :3. Look for critical points.
 :
P:sol
```

```
My homework assignment was to study the function:
.1*x^3-.5*x+3
There were three parts to the assignment.
1. Graph the function.
```



```
2. Find its derivative.
.3*x^2-.5
3. Look for critical points.
x=1.29099 or x=-1.29099
```

Note: *To store the derivative to variable der, enter:* ***d***(y1(x),x)→der

Note: *To store the derivative's critical points to variable sol, enter:* **solve**(der=0,x)→sol

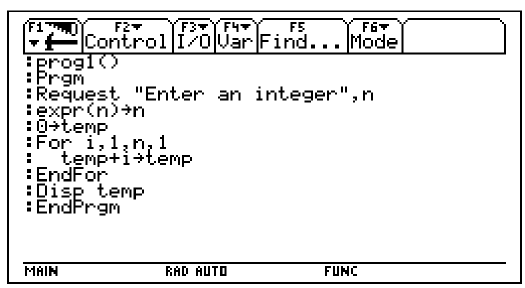
In cases where a graph picture cannot fit on the current page, the entire picture is shifted to the top of the next page.

Programming

17

***Note:** For details and examples of any TI-92 program command mentioned in this chapter, refer to Appendix A.*

This chapter describes how to use the TI-92's Program Editor to create your own programs or functions.

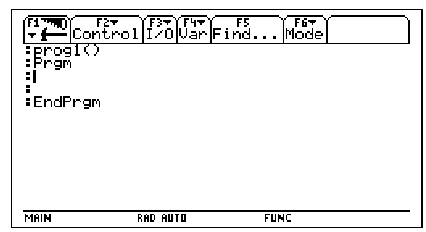
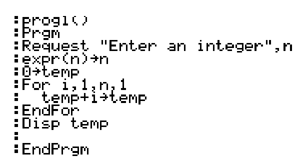


The chapter includes:

- Specific instructions on using the Program Editor itself and running an existing program.
- An overview of fundamental programming techniques such as **If..EndIf** structures and various kinds of loops.
- Reference information that categorizes the available program commands.

Preview of Programming

Write a program that prompts the user to enter an integer, sums all integers from 1 to the entered integer, and displays the result.

Steps	Keystrokes	Display
1. Start a new program on the Program Editor.	[APPS] 7 3	
2. Type PROG1 (with no spaces) as the name of the new program variable.	⊙ ⊙ P R O G 1	
3. Display the “template” for a new program. The program name, **Prgm**, and **EndPrgm** are shown automatically. *After typing in an input box such as Variable, you must press [ENTER] twice.* *The cursor is automatically positioned on the first line after* **Prgm**.	[ENTER] [ENTER]	
4. Type the following program lines.	Type the program lines as shown. Press [ENTER] at the end of each line.	

`Request "Enter an integer",n`

Displays a dialog box that prompts “Enter an integer”, waits for the user to enter a value, and stores it (as a string) to variable n.

`expr(n)→n`

Converts the string to a numeric expression.

`0→temp`

Creates a variable named temp and initializes it to 0.

`For i,1,n,1`

Starts a For loop based on variable i. First time through the loop, i = 1. At end of loop, i is incremented by 1. Loop continues until i > n.

`temp+i→temp`

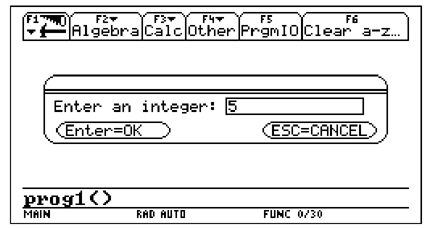
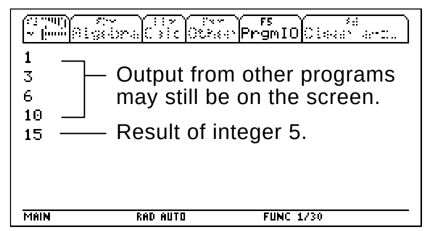
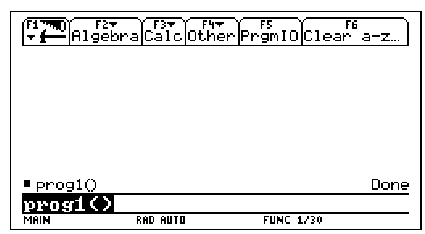
Adds current value of i to temp.

`EndFor`

Marks the end of the For loop.

`Disp temp`

Displays the final value of temp.

Steps	Keystrokes	Display
5. Go to the Home screen. Enter the program name, followed by a set of parentheses. *You must include ( ) even when there are no arguments for the program.* *The program displays a dialog box with the prompt specified in the program.*	[♦] [HOME] P R O G 1 [(] [)] [ENTER]	prog1()
6. Type 5 in the displayed dialog box.	5	
7. Continue with the program. The **Disp** command displays the result on the Program I/O screen. *The result is the sum of the integers from 1 through 5.* *Although the Program I/O screen looks similar to the Home screen, it is for program input and output only. You cannot perform calculations on the Program I/O screen.*	[ENTER] [ENTER]	
8. Leave the Program I/O screen and return to the Home screen. *You can also press [ESC], [2nd] [QUIT], or [♦] [HOME] to return to the Home screen.*	[F5]	

Running an Existing Program

After a program is created (as described in the remaining sections of this chapter), you can run it from the Home screen. The program's output, if any, is displayed on the Program I/O screen, in a dialog box, or on the Graph screen.

Running a Program

__Tip:__ Use [2nd] [VAR-LINK] to list existing PRGM variables. Highlight a variable and press [ENTER] to paste its name to the entry line.

__Note:__ Arguments specify initial values for a program. Refer to page 301.

__Note:__ The TI-92 also checks for run-time errors that are found within the program itself. Refer to page 324.

On the Home screen:

1. Type the name of the program.
2. You must *always* type a set of parentheses after the name.

   `prog1()`

   If arguments are not required

   Some programs require you to pass an argument to the program.

   `prog1(x,y)`

   If arguments are required

3. Press [ENTER].

When you run a program, the TI-92 automatically checks for errors. For example, the following message is displayed if you:

- Do not enter ( ) after the program name.
- Do not enter enough arguments, if required.

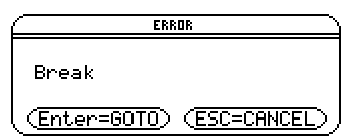
To cancel program execution if an error occurs, press [ESC]. You can then correct any problems and run the program again.

"Breaking" a Program

When a program is running, the BUSY indicator is displayed in the status line.

Press [ON] to stop program execution. A message is then displayed.

- To display the program in the Program Editor, press [ENTER]. The cursor appears at the command where the break occurred.
- To cancel program execution, press [ESC].



Where Is the Output Displayed?

Depending on the commands in the program, the TI-92 automatically displays information on the applicable screen.

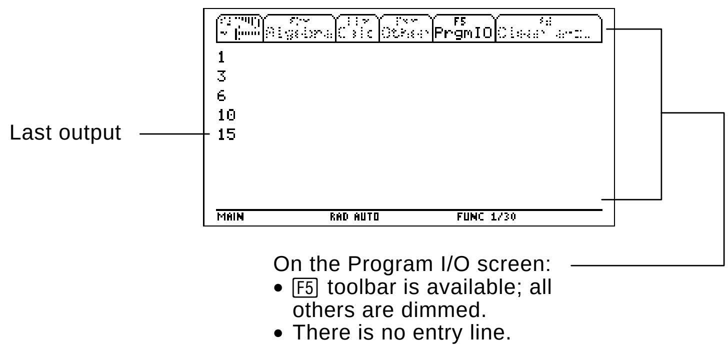
- Most output and input commands use the Program I/O screen. (Input commands prompt the user to enter information.)
- Graph-related commands typically use the Graph screen.

After the program stops, the TI-92 shows the last screen that was displayed.

The Program I/O Screen

On the Program I/O screen, new output is displayed below any previous output (which may have been displayed earlier in the same program or a different program). After a full page of output, the previous output scrolls off the top of the screen.

Tip: *To clear any previous output, enter the* **ClrIO** *command in your program. You can also execute* **ClrIO** *from the Home screen.*



Tip: *If Home screen calculations don't work after you run a program, you may be on the Program I/O screen.*

When a program stops on the Program I/O screen, you need to recognize that it is *not* the Home screen (although the two screens are similar). The Program I/O screen is used only to display output or to prompt the user for input. You cannot perform calculations on this screen.

Leaving the Program I/O Screen

From the Program I/O screen:

- Press [F5] to display the Home screen. ([F5] toggles between the Home screen and the Program I/O screen.)

  — or —
- Press [ESC] or [2nd] [QUIT] to display the Home screen.

  — or —
- Display any other application screen (with [APPS], [♦] [HOME], [♦] [Y=], etc.).

Starting a Program Editor Session

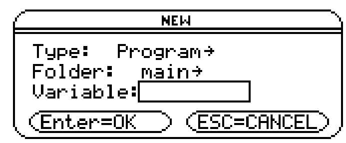
Each time you start the Program Editor, you can resume the current program or function (that was displayed the last time you used the Program Editor), open an existing program or function, or start a new program or function.

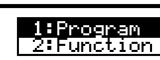
Starting a New Program or Function

1. Press [APPS] and then select 7:Program Editor.

2. Select 3:New.

3. Specify the applicable information for the new program or function.

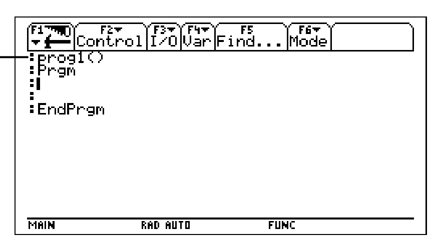


Item	Lets you:
Type	Select whether to create a new program or function. 
Folder	Select the folder in which the new program or function will be stored. For information about folders, refer to Chapter 10.
Variable	Type a variable name for the program or function. If you specify a variable that already exists, an error message will be displayed when you press [ENTER]. When you press [ESC] or [ENTER] to acknowledge the error, the NEW dialog box is redisplayed.

4. Press [ENTER] (after typing in an input box such as Variable, you must press [ENTER] twice) to display an empty "template."

Note: A program (or function) is saved automatically as you type. You do not need to save it manually before leaving the Program Editor, starting a new program, or opening a previous one.

This is the template for a program. Functions have a similar template.



You can now use the Program Editor as described in the remaining sections of this chapter.

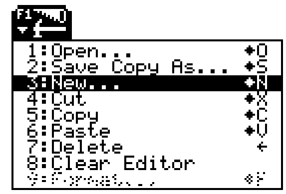
Resuming the Current Program

You can leave the Program Editor and go to another application at any time. To return to the program or function that was displayed when you left the Program Editor, press [APPS] 7 and select 1:Current.

Starting a New Program from the Program Editor

To leave the current program or function and start a new one:

1. Press [F1] and select 3:New. (You can press [♦] N instead of using the [F1] toolbar menu.)
2. Specify the type, folder, and variable for the new program or function.
3. Press [ENTER] twice.

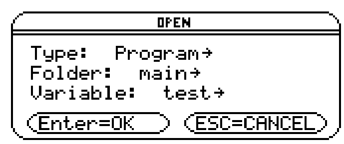


Opening a Previous Program

You can open a previously created program or function at any time.

1. From within the Program Editor, press [F1] and select 1:Open. You can press [♦] O instead of using the [F1] toolbar menu.)
   — or —
   From another application, press [APPS] 7 and select 2:Open.
2. Select the applicable type, folder, and variable.
3. Press [ENTER].

__Note:__ By default, Variable shows the first existing program or function in alphabetical order.



Copying a Program

In some cases, you may want to copy a program or function so that you can edit the copy while retaining the original.

1. Display the program or function you want to copy.
2. Press [F1] and select 2:Save Copy As. (You can press [♦] S instead of using the [F1] toolbar menu.)
3. Specify the folder and variable for the copy.
4. Press [ENTER] twice.

Note about Deleting a Program

Because all Program Editor sessions are saved automatically, you can accumulate quite a few previous programs and functions, which take up memory storage space.

To delete programs and functions, use the VAR-LINK screen ([2nd] [VAR-LINK]). For information about VAR-LINK, refer to Chapter 18.

Overview of Entering a Program

A program is a series of commands executed in sequential order (although some commands alter the program flow). In general, anything that can be executed from the Home screen can be included in a program. Program execution continues until it reaches the end of the program or a **Stop** command.

Entering and Editing Program Lines

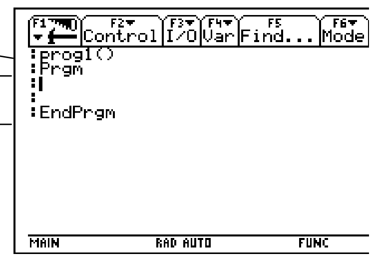
On a blank template, you can begin entering commands for your new program.

Note: Use the cursor pad to scroll through the program for entering or editing commands.

Program name, which you specify when you create a new program.

Enter your program commands between **Prgm** and **EndPrgm**.

All program lines begin with a colon.



You enter and edit program commands in the Program Editor by using the same techniques used to enter and edit text in the Text Editor. Refer to “Entering and Editing Text” in Chapter 16.

Note: *Entering a command does not execute that command. It is not executed until you run the program.*

After typing each program line, press [ENTER]. This inserts a new blank line and lets you continue entering another line. A program line can be longer than one line on the screen; if so, it will wrap to the next screen line automatically.

Entering Multi-Command Lines

To enter more than one command on the same line, separate them with a colon by pressing [2nd] [:].

Entering Comments

A comment symbol (©) lets you enter a remark in a program. When you run the program, all characters to the right of © are ignored.

Tip: *Use comments to enter information that is useful to someone reading the program code.*

Description of the program. — :©Displays sum of 1 thru n

Description of **expr**. — :expr(n)→n:©Convert to numeric expression

```
:prog1()
:Prgm
:©Displays sum of 1 thru n
:Request "Enter an integer",n
:expr(n)→n:©Convert to numeric expression
:------
```

To enter the comment symbol:

- Press [2nd] X.

  — or —
- Press [F2] and select 9:©.

Controlling the Flow of a Program

Tip: For information, refer to pages 311 and 313.

When you run a program, the program lines are executed in sequential order. However, some commands alter the program flow. For example:

- Control structures such as **If...EndIf** commands use a conditional test to decide which part of a program to execute.
- Loops commands such as **For...EndFor** repeat a group of commands.

Using Indentation

For more complex programs that use **If...EndIf** and loop structures such as **For...EndFor**, you can make the programs easier to read and understand by using indentation.

```
:If x>5 Then
:  Disp "x is > 5"
:Else
:  Disp "x is < or = 5"
:EndIf
```

Displaying Calculated Results

In a program, calculated results are not displayed unless you use an output command. This is an important difference between performing a calculation on the Home screen and in a program.

These calculations will not display a result in a program (although they will on the Home screen).

```
:12*6
:cos(π/4)
:solve(x^2-x-2=0,x)
```

Tip: For a list of available output commands, refer to page 318.

Output commands such as **Disp** will display a result in a program.

```
:Disp 12*6
:Disp cos(π/4)
:Disp solve(x^2-x-2=0,x)
```

Displaying a calculation result does not store that result. If you need to refer to a result later, store it to a variable.

```
:cos(π/4)→max
:Disp max
```

Getting Values into a Program

Tip: For a list of available input commands, refer to page 317.

To input values into a program, you can:

- Require the users to store a value (with [STO▸]) to the necessary variables before running the program. The program can then refer to these variables.
- Enter the values directly into the program itself.

```
:Disp 12*6
:cos(π/4)→max
```

- Include input commands that prompt the users to enter the necessary values when they run the program.

```
:Input "Enter a value",i
:Request "Enter an integer",n
```

- Require the users to pass one or more values to the program when they run it.

```
prog1(3,5)
```

Example of Passing Values to a Program

The following program draws a circle on the Graph screen and then draws a horizontal line across the top of the circle. Three values must be passed to the program: x and y coordinates for the circle's center and the radius r.

Note: In this example, you cannot use **circle** *as the program name because it conflicts with a command name.*

- When you write the program in the Program Editor:

  In the ( ) beside the program name, specify the variables that will be used to store the passed values.

  Notice that the program also contains commands that set up the Graph screen.

```
:circ(xx,yy,rr)
:Prgm
:FnOff
:ZoomStd
:ZoomSqr
:Circle xx,yy,rr
:LineHorz yy+rr
:EndPrgm
```

Only **circ( )** is initially displayed on the blank template; be sure to edit this line.

Before drawing the circle, the program turns off any selected Y= Editor functions, displays a standard viewing window, and "squares" the window.

Note: This example assumes that the user enters values that can be displayed by the viewing window set up by ZoomStd and ZoomSqr.

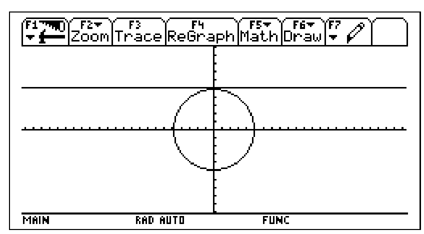
- To run the program from the Home screen:

  The user must specify the applicable values as arguments within the ( ).

  The arguments, in order, are passed to the program.

```
circ(0,0,5)
```

Passed to rr.
Passed to yy.
Passed to xx.



Overview of Entering a Function

A function created in the Program Editor is very similar to the functions and instructions that you typically use from the Home screen.

Why Create a User-Defined Function?

Note: *You can create a function from the Home screen (see Chapter 10), but the Program Editor is more convenient for complex, multi-line functions.*

Functions (as well as programs) are ideal for repetitive calculations or tasks. You only need to write the function once. Then you can reuse it as many times as necessary. Functions, however, have some advantages over programs.

- You can create functions that expand on the TI-92's built-in functions. You can then use the new functions the same as any other function.
- Functions return values that can be graphed or entered in a table. Programs cannot.
- You can use a function (but not a program) within an expression. For example: 3*func1(3) is valid, but not 3*prog1(3).
- Because you pass arguments to a function, you can write generic functions that are not tied to specific variable names.

Differences Between Functions and Programs

This guidebook sometimes use the word *command* as a generic reference to instructions and functions. When writing a function, however, you must differentiate between instructions and functions.

A user-defined function:

- Can use the following instructions only. Any others are invalid.

Cycle	Define	Exit
For...EndFor	Goto	If...EndIf (all forms)
Lbl	Local	Loop...EndLoop
Return	While...EndWhile	→ ([STO▸] key)

- Can use all built-in TI-92 functions except:

setFold	setGraph	setMode
setTable	switch	

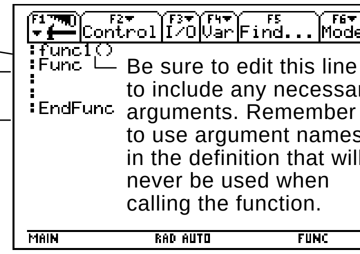
Tip: *For information about local variables, refer to pages 306 and 307.*

- Can refer to any variable; however, it can store a value to a local variable only.
  – The arguments used to pass values to a function are treated as local variables automatically. If you store to any other variables, you *must* declare them as local from within the function.
- Cannot call a program as a subroutine, but it can call another user-defined function.
- Cannot define a program.
- Cannot define a global function, but it can define a local function.

Entering a Function

Note: Use the cursor pad to scroll through the function for entering or editing commands.

When you create a new function in the Program Editor, the TI-92 displays a blank "template".



If the function requires input, one or more values must be passed to the function. (A user-defined function can store to local variables only, and it cannot use instructions that prompt the user for input.)

How to Return a Value from a Function

Note: This example calculates the cube if xx≥0; otherwise, it returns a 0.

There are two ways to return a value from a function:

- As the last line in the function (before **EndFunc**), calculate the value to be returned.

```
:cube(xx)
:Func
:xx^3
:EndFunc
```

- Use **Return**. This is useful for exiting a function and returning a value at some point other than the end of the function.

```
:cube(xx)
:Func
:If xx<0
:  Return 0
:xx^3
:EndFunc
```

The argument xx is automatically treated as a local variable. However, if the example needed another variable, the function would need to declare it as local by using the **Local** command (pages 306 and 307).

There is an implied **Return** at the end of the function. If the last line is not an expression, an error occurs.

Example of a Function

Note: Because xx and yy in the function are local, they are not affected by any existing xx or yy variable.

The following function returns the xth root of a value y ($\sqrt[x]{y}$). Two values must be passed to the function: x and y.

Function as called from the Home Screen	Function as defined in the Program Editor
`4*xroot(3,125)` → 20	`:xroot(xx,yy)`
3→xx; 125→yy	`:Func`
5	`:yy^(1/xx)`
	`:EndFunc`

Calling One Program from Another

One program can call another program as a subroutine. The subroutine can be external (a separate program) or internal (included in the main program). Subroutines are useful when a program needs to repeat the same group of commands at several different places.

Calling a Separate Program

To call a separate program, use the same syntax used to run the program from the Home screen.

```
:subtest1()
:Prgm
:For i,1,4,1
:  subtest2(i,i*1000)
:EndFor
:EndPrgm
```

```
:subtest2(xx,yy)
:Prgm
:  Disp xx,yy
:EndPrgm
```

Calling an Internal Subroutine

To define an internal subroutine, use the **Define** command with **Prgm...EndPrgm**. Because a subroutine must be defined before it can be called, it is a good practice to define subroutines at the beginning of the main program.

An internal subroutine is called and executed in the same way as a separate program.

Tip: *Use the Program Editor's* [F4] Var *toolbar menu to enter the* **Define** *and* **Prgm...EndPrgm** *commands.*

```
:subtest1()
:Prgm
:local subtest2
:Define subtest2(xx,yy)=Prgm
:  Disp xx,yy
:EndPrgm
:©Beginning of main program
:For i,1,4,1
:  subtest2(i,i*1000)
:EndFor
:EndPrgm
```

Declares the subroutine as a local variable. (`:local subtest2`)

Defines the subroutine. (`:Define subtest2(xx,yy)=Prgm` … `:EndPrgm`)

Calls the subroutine. (`:  subtest2(i,i*1000)`)

Notes about Using Subroutines

At the end of a subroutine, execution returns to the calling program. To exit a subroutine at any other time, use the **Return** command.

A subroutine cannot access local variables declared in the calling program. Likewise, the calling program cannot access local variables declared in a subroutine.

Lbl commands are local to the programs in which they are located. Therefore, a **Goto** command in the calling program cannot branch to a label in a subroutine or vice versa.

Using Variables in a Program

Programs use variables in the same general way that you use them from the Home screen. However, the “scope” of the variables affects how they are stored and accessed.

Scope of Variables

Note: *For information about folders, refer to Chapter 10.*

Note: *If a program has local variables, a graphed function cannot access them. For example:*

```
Local aa
5→aa
Graph aa∗cos(x)
```

may display an error or an unexpected result (if aa is an existing variable in the current folder).

<table>
<tr><th>Scope</th><th>Description</th></tr>
<tr><td>System (Global) Variables</td><td>Variables with reserved names that are created automatically to store data about the state of the TI-92. For example, Window variables (xmin, xmax, ymin, ymax, etc.) are globally available from any folder.
You can always refer to these variables by using the variable name only, regardless of the current folder.
A program cannot create system variables, but it can use the values and (in most cases) store new values.</td></tr>
<tr><td>Folder Variables</td><td>Variables that are stored in a particular folder.
If you store to a variable name only, it is stored in the current folder. For example:
5→start
If you refer to a variable name only, that variable must be in the current folder. Otherwise, it cannot be found (even if the variable exists in a different folder).
To store or refer to a variable in another folder, you must specify a pathname. For example:
5→class\start
(class = Folder name; start = Variable name)
After the program stops, any folder variables created by the program still exist and still take up memory.</td></tr>
<tr><td>Local Variables</td><td>Temporary variables that exist only while a program is running. When the program stops, local variables are deleted automatically.
To create a local variable in a program, use the Local command to declare the variable.
A local variable is treated as unique even if there is an existing folder variable with the same name.
Local variables are ideal for temporarily storing values that you do not want to save.</td></tr>
</table>

Variable-Related Commands

Note: The **Define**, **DelVar**, *and* **Local** *commands are available from the Program Editor's* F4 Var *toolbar menu.*

Command	Description
STO▸ key	Stores a value to a variable. As on the Home screen, pressing STO▸ enters a → symbol.
CopyVar	Copies the contents of a variable.
Define	Defines a program (subroutine) or function variable within a program.
DelFold	Deletes a folder. All variables in that folder must be deleted first.
DelVar	Deletes a variable.
getFold	Returns the name of the current folder.
getType	Returns a string that indicates the data type (EXPR, LIST, etc.) of a variable.
Local	Declares one or more variables as local variables.
Lock	Locks a variable so that it cannot be accidentally changed or deleted without first being unlocked.
MoveVar	Moves a variable from one folder to another.
NewData	Creates a data variable whose columns consist of a series of specified lists.
NewFold	Creates a new folder.
NewPic	Creates a picture variable based on a matrix.
Rename	Renames a variable.
Unlock	Unlocks a locked variable.

Example of a Local Variable

Tip: As often as possible, use local variables for any variable that is used only within a program and does not need to be stored after the program stops.

The following program segment shows a **For...EndFor** loop (which is discussed later in this chapter). The variable i is the loop counter. In most cases, the variable i is used only while the program is running.

Declares variable i as local.

```
:Local i
:For i,0,5,1
:  Disp i
:EndFor
:Disp i
```

If you declare variable i as local, it is deleted automatically when the program stops so that it does not use up memory.

String Operations

Strings are used to enter and display text characters. You can type a string directly, or you can store a string to a variable.

How Strings Are Used

A string is a sequence of characters enclosed in "quotes". In programming, strings allow the program to display information or prompt the user to perform some action. For example:

```
Disp "The result is",answer
```

— or —

```
Input "Enter the angle in degrees",ang1
```

— or —

```
"Enter the angle in degrees"→str1
Input str1,ang1
```

Some input commands (such as **InputStr**) automatically store user input as a string and do not require the user to enter quotation marks.

A string cannot be evaluated mathematically, even if it appears to be a numeric expression. For example, the string "61" represents the characters "6" and "1", not the number 61.

Although you cannot use a string such as "61" or "2x+4" in a calculation, you can convert a string into a numeric expression by using the **expr** command.

String Commands

Command	Description
#	Converts a string into a variable name. This is called indirection.
&	Appends (concatenates) two strings into one string.
char	Returns the character that corresponds to a specified character code. This is the opposite of the **ord** command.
dim	Returns the number of characters in a string.
expr	Converts a string into an expression and executes that expression. This is the opposite of the **string** command. **Important:** Some user input commands store the entered value as a string. Before you can perform a mathematical operation on that value, you must convert it to a numeric expression.
inString	Searches a string to see if it contains a specified substring. If so, **inString** returns the character position where the first occurrence of the substring begins.
left	Returns a specified number of characters from the left side (beginning) of a string.
mid	Returns a specified number of characters from any position within a string.
ord	Returns the character code of the first character within a string. This is the opposite of the **char** command.
right	Returns a specified number of characters from the right side (end) of a string.
string	Converts a numeric expression into a string. This is the opposite of the **expr** command.

Conditional Tests

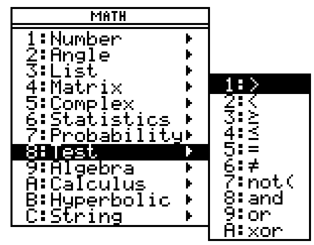
Conditional tests let programs make decisions. For example, depending on whether a test is true or false, a program can decide which of two actions to perform. Conditional tests are used with control structures such as **If...EndIf** and loops such as **While...EndWhile** (described later in this chapter).

Entering a Test Operator

- Type the operator directly from the keyboard.

  — or —
- Press [2nd] [MATH] and select 8:Test. Then select the operator from the menu.

  — or —
- Press [2nd] [CATALOG]. The test operators are listed near the bottom of the CATALOG menu.



Relational Tests

Relational operators let you define a conditional test that compares two values. The values can be numbers, expressions, lists, or matrices (but they must match in type and dimension).

__Tip:__ From the keyboard, you can type:
>= for ≥
<= for ≤
/= for ≠
(To get the / character, press [÷].)

Operator	True if:	Example
>	Greater than	a>8
<	Less than	a<0
≥	Greater than or equal to	a+b≥100
≤	Less than or equal to	a+6≤b+1
=	Equal	list1=list2
≠	Not equal to	mat1≠mat2

Boolean Tests

Boolean operators let you combine the results of two separate tests.

Operator	True if:	Example
and	Both tests are true	a>0 and a≤10
or	At least one test is true	a≤0 or b+c>10
xor	One test is true and the other is false	a+6<b+1 xor c<d

The Not Function

The **not** function changes the result of a test from true to false and vice versa. For example:

not(x>2) is true if x≤2
false if x>2

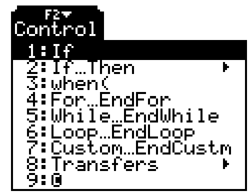
Note: If you use **not** from the Home screen, it is shown as **~** in the history area. For example, not(x>2) is shown as ~(x>2).

Using If, Lbl, and Goto to Control Program Flow

An **If...EndIf** structure uses a conditional test to decide whether or not to execute one or more commands. **Lbl** (label) and **Goto** commands can also be used to branch (or jump) from one place to another in a program.

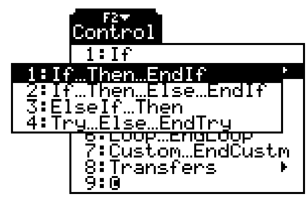
[F2] Control Toolbar Menu

To enter **If...EndIf** structures, use the Program Editor's [F2] Control toolbar menu.



The **If** command is available directly from the [F2] menu.

To see a submenu that lists other **If** structures, select 2:If...Then.



When you select a structure such as **If...Then...EndIf**, a template is inserted at the cursor location.

```
:If | Then

:EndIf
```

The cursor is positioned so that you can enter a conditional test.

If Command

To execute only one command if a conditional test is true, use the general form:

Tip: Use indentation to make your programs easier to read and understand.

```
:If x>5
:   Disp "x is greater than 5"
:Disp x
```

Executed only if x>5; otherwise, skipped. (`Disp "x is greater than 5"`)

Always displays the value of x. (`:Disp x`)

In this example, you must store a value to x before executing the **If** command.

If...Then...EndIf Structures

To execute multiple commands if a conditional test is true, use the structure:

***Note:* EndIf** *marks the end of the* **Then** *block that is executed if the condition is true.*

```
:If x>5 Then
:  Disp "x is greater than 5"
:  2*x→x
:EndIf
:Disp x
```

Executed only if x>5. (the two indented lines)

Displays value of: (`:Disp x`)
- 2x if x>5.
- x if x≤5.

If...Then...Else... EndIf Structures

To execute one group of commands if a conditional test is true and a different group if the condition is false, use this structure:

Executed only if x>5.

Executed only if x≤5.

Displays value of:
- 2x if x>5.
- 5x if x≤5.

```
:If x>5 Then
:  Disp "x is greater than 5"
:  2*x→x
:Else
:  Disp "x is less than or equal to 5"
:  5*x→x
:EndIf
:Disp x
```

If...Then...ElseIf... EndIf Structures

A more complex form of the **If** command lets you test a series of conditions. Suppose your program prompts the user for a number that corresponds to one of four options. To test for each option (If Choice=1, If Choice = 2, etc.), use the **If...Then...ElseIf...EndIf** structure.

Refer to Appendix A for more information and an example.

Lbl and Goto Commands

You can also control the flow of your program by using **Lbl** (label) and **Goto** commands.

Use the **Lbl** command to label (assign a name to) a particular location in the program.

Lbl *labelName*

labelName — name to assign to this location (use the same naming convention as a variable name)

You can then use the **Goto** command at any point in the program to branch to the location that corresponds to the specified label.

Goto *labelName*

labelName — specifies which **Lbl** command to branch to

Because a **Goto** command is unconditional (it always branches to the specified label), it is often used with an **If** command so that you can specify a conditional test. For example:

If x>5, branches directly to label GT5.

For this example, the program must include commands (such as **Stop**) that prevent Lbl GT5 from being executed if x≤5.

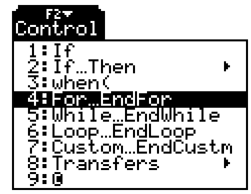
```
:If x>5
:  Goto GT5
:Disp x
:--------
:--------
:Lbl GT5
:Disp "The number was > 5"
```

Using Loops to Repeat a Group of Commands

To repeat the same group of commands successively, use a loop. Several types of loops are available. Each type gives you a different way to exit the loop, based on a conditional test.

[F2] Control Toolbar Menu

To enter most of the loop-related commands, use the Program Editor's [F2] Control toolbar menu.



Note: *A loop command marks the start of the loop. The corresponding* **End** *command marks the end of the loop.*

When you select a loop, the loop command and its corresponding **End** command are inserted at the cursor location.

```
:For |
:EndFor
```

If the loop requires arguments, the cursor is positioned after the command.

You can then begin entering the commands that will be executed in the loop.

For...EndFor Loops

A **For...EndFor** loop uses a counter to control the number of times the loop is repeated. The syntax of the **For** command is:

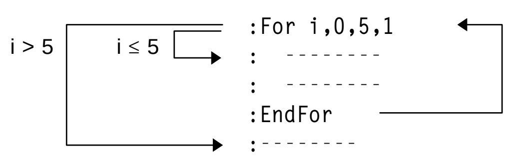
For(*variable*, *begin*, *end* [, *increment*])

- *increment* — added to the counter each subsequent time **For** is executed (If this optional value is omitted, the increment is 1.)
- *end* — exits the loop when *variable* exceeds this value
- *begin* — counter value used the first time **For** is executed
- *variable* — variable used as a counter

Note: *The ending value can be less than the beginning value, but the increment must be negative.*

When **For** is executed, the *variable* value is compared to the *end* value. If *variable* does not exceed *end*, the loop is executed; otherwise, program control jumps to the command following **EndFor**.

Note: *The* **For** *command automatically increments the counter variable so that the program can exit the loop after a certain number of repetitions.*



At the end of the loop (**EndFor**), program control jumps back to the **For** command, where *variable* is incremented and compared to *end*.

For example:

Tip: *You can declare the counter variable as local (pages 306 and 307) if it does not need to be saved after the program stops.*

```
:For i,0,5,1
:  Disp i
:EndFor
:Disp i
```

- `:  Disp i` — Displays 0, 1, 2, 3, 4, and 5.
- `:Disp i` — Displays 6. When *variable* increments to 6, the loop is not executed.

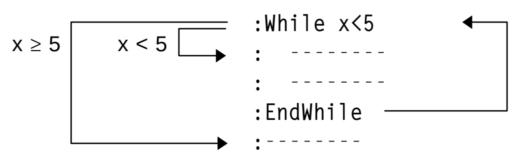
While...EndWhile Loops

A **While...EndWhile** loop repeats a block of commands as long as a specified condition is true. The syntax of the **While** command is:

While *condition*

When **While** is executed, the condition is evaluated. If *condition* is true, the loop is executed; otherwise, program control jumps to the command following **EndWhile**.

Note: *The* **While** *command does not automatically change the condition. You must include commands that allow the program to exit the loop.*



At the end of the loop (**EndWhile**), program control jumps back to the **While** command, where *condition* is re-evaluated.

To execute the loop the first time, the *condition* must initially be true.

- Any variables referenced in the *condition* must be set before the **While** command. (You can build the values into the program or prompt the user to enter the values.)
- The loop must contain commands that change the values in the *condition*, eventually causing it to be false. Otherwise, the *condition* is always true and the program cannot exit the loop (called an infinite loop).

For example:

```
:0→x
:While x<5
:  Disp x
:  x+1→x
:EndWhile
:Disp x
```

- `:0→x` — Initially sets x.
- `:  Disp x` — Displays 0, 1, 2, 3, and 4.
- `:  x+1→x` — Increments x.
- `:Disp x` — Displays 5. When x increments to 5, the loop is not executed.

Loop...EndLoop Loops

A **Loop...EndLoop** creates an infinite loop, which is repeated endlessly. The **Loop** command does not have any arguments.

```
:Loop        ◄──┐
:  --------     │
:  --------     │
:EndLoop     ───┘
:--------
```

Typically, the loop contains commands that let the program exit from the loop. Commonly used commands are: **If**, **Exit**, **Goto**, and **Lbl** (label). For example:

Note: The **Exit** *command exits from the current loop.*

```
:0→x
:Loop
:  Disp x
:  x+1→x
:  If x>5
:    Exit
:EndLoop
:Disp x
```

An **If** command checks the condition. (points to `: If x>5`)

Exits the loop and jumps to here when x increments to 6. (points to `:Disp x`)

In this example, the **If** command can be anywhere in the loop.

When the If command is:	The loop is:
At the beginning of the loop	Executed only if the condition is true.
At the end of the loop	Executed at least once and repeated only if the condition is true.

The **If** command could also use a **Goto** command to transfer program control to a specified **Lbl** (label) command.

Repeating a Loop Immediately

The **Cycle** command immediately transfers program control to the next iteration of a loop (before the current iteration is complete). This command works with **For...EndFor**, **While...EndWhile**, and **Loop...EndLoop**.

Lbl and Goto Loops

Although the **Lbl** (label) and **Goto** commands are not strictly loop commands, they can be used to create an infinite loop. For example:

```
:Lbl START     ◄──┐
:  --------       │
:  --------       │
:Goto START    ───┘
:--------
```

As with **Loop...EndLoop**, the loop should contain commands that let the program exit from the loop.

Configuring the TI-92

Programs can contain commands that change the TI-92's configuration. Because mode changes are particularly useful, the Program Editor's [F6] **Mode** toolbar menu makes it easy to enter the correct syntax for the **setMode** command.

Configuration Commands

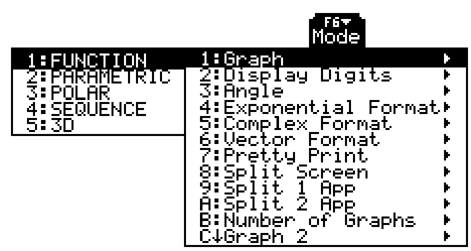
Command	Description
getFold	Returns the name of the current folder.
getMode	Returns the current setting for a specified mode.
setFold	Sets the current folder.
setGraph	Sets a specified graph format (Coordinates, Graph Order, etc.).
setMode	Sets any mode except Current Folder.
setTable	Sets a specified table setup parameter (tblStart, ∆tbl, etc.)
switch	Sets the active window in a split screen, or returns the number of the active window.

Entering the SetMode Command

Note: [F6] *does not let you set the* Current Folder *mode. To set this mode, use the* **setFold** *command.*

In the Program Editor:

1. Position the cursor where you want to insert the **setMode** command.
2. Press [F6] to display a list of modes.
3. Select a mode to display a menu of its valid settings.
4. Select a setting.



The correct syntax is inserted into your program.

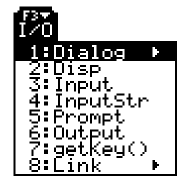
```
:setMode("Graph","FUNCTION")
```

Getting Input from the User and Displaying Output

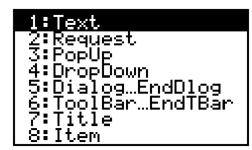
Although values can be built into a program (or stored to variables in advance), a program can prompt the user to enter information while the program is running. Likewise, a program can display information such as the result of a calculation.

[F3] I/O Toolbar Menu

To enter most of the commonly used input/output commands, use the Program Editor's [F3] I/O toolbar menu.



To see a submenu that lists additional commands, select 1:Dialog.



Input Commands

__Tip:__ String input cannot be used in a calculation. To convert a string to a numeric expression, use the __expr__ command.

Command	Description
getKey	Returns the key code of the next key pressed.
Input	Prompts the user to enter an expression. The expression is treated according to how it is entered. For example: • A numeric expression is treated as an expression. • An expression enclosed in "quotes" is treated as a string. **Input** can also display the Graph screen and let the user update the variables xc and yc (rc and θc in polar mode) by positioning the graph cursor.
InputStr	Prompts the user to enter an expression. The expression is always treated as a string; the user does not need to enclose the expression in "quotes".
PopUp	Displays a pop-up menu box and lets the user select an item.
Prompt	Prompts the user to enter a series of expressions. As with **Input**, each expression is treated according to how it is entered.
Request	Displays a dialog box that prompts the user to enter an expression. **Request** always treats the entered expression as a string.

Output Commands

Note: In a program, simply performing a calculation does not display the result. You must use an output command.

*Tip: After **Disp** and **Output**, the program immediately continues. You may want to add a **Pause** command.*

Command	Description
ClrIO	Clears the Program I/O screen.
Disp	Displays an expression or string on the Program I/O screen. **Disp** can also display the current contents of the Program I/O screen without displaying additional information.
DispG	Displays the current contents of the Graph screen.
DispTbl	Displays the current contents of the Table screen.
Output	Displays an expression or string starting at specified coordinates on the Program I/O screen.
Format	Formats the way in which numeric information is displayed.
Pause	Suspends program execution until the user presses [ENTER]. Optionally, you can display an expression during the pause. A pause lets users read your output and decide when they are ready to continue.
Text	Displays a dialog box that contains a specified character string.

Graphical User Interface Commands

Tip: When you run a program that sets up a custom toolbar, that toolbar is still available even after the program has stopped.

*Note: **Request** and **Text** are stand-alone commands that can also be used outside of a dialog box or toolbar program block.*

Command	Description
Dialog... endDlog	Defines a program block (consisting of **Title**, **Request**, etc., commands) that displays a dialog box.
Toolbar... EndTbar	Defines a program block (consisting of **Title**, **Item**, etc., commands) that replaces the toolbar menus. The redefined toolbar is in effect only while the program is running and only until the user selects an item. Then the original toolbar is redisplayed.
Custom... EndCustm	Defines a program block that displays a custom toolbar when the user presses [2nd] [CUSTOM]. That toolbar remains in effect until the user presses [2nd] [CUSTOM] again or changes applications.
DropDown	Displays a drop-down menu within a dialog box.
Item	Displays a menu item for a redefined toolbar.
Request	Creates an input box within a dialog box.
Text	Displays a character string within a dialog box.
Title	Displays the title of a dialog box or a menu title within a toolbar.

Creating a Table or Graph

To create a table or a graph based on one or more functions or equations, use the commands listed in this section.

Table Commands

Command	Description
DispTbl	Displays the current contents of the Table screen.
setTable	Sets the Graph <–> Table or Independent table parameters. (To set the other two table parameters, you can store the applicable values to the tblStart and ∆tbl system variables.)
Table	Builds and displays a table based on one or more expressions or functions.

Graphing Commands

__Note:__ For more information about using **setMode**, *refer to page 316.*

Command	Description
ClrGraph	Erases any functions or expressions that were graphed with the **Graph** command.
Define	Creates a user-defined function.
DispG	Displays the current contents of the Graph screen.
FnOff	Deselects all (or only specified) Y= functions.
FnOn	Selects all (or only specified) Y= functions.
Graph	Graphs one or more specified expressions, using the current graphing mode.
Input	Displays the Graph screen and lets the user update the variables xc and yc (rc and θc in polar mode) by positioning the graph cursor.
NewPlot	Creates a new stat plot definition.
PlotsOff	Deselects all (or only specified) stat data plots.
PlotsOn	Selects all (or only specified) stat data plots.
setGraph	Changes settings for the various graph formats (Coordinates, Graph Order, etc.).
setMode	Sets the Graph mode, as well as other modes.
Style	Sets the display style for a function.
Trace	Lets a program trace a graph.
ZoomBox – to – ZoomTrig	Perform all of the Zoom operations that are available from the [F2] toolbar menu on the Y= Editor, Window Editor, and Graph screen.

Graph Picture and Database Commands

Note: *For information about graph pictures and databases, also refer to Chapter 15.*

Command	Description
AndPic	Displays the Graph screen and superimposes a stored graph picture by using AND logic.
CyclePic	Animates a series of stored graph pictures.
NewPic	Creates a graph picture variable based on a matrix.
RclGDB	Restores all settings stored in a graph database.
RclPic	Displays the Graph screen and superimposes a stored graph picture by using OR logic.
RplcPic	Clears the Graph screen and displays a stored graph picture.
StoGDB	Stores the current graph settings to a graph database variable.
StoPic	Copies the Graph screen (or a specified rectangular portion) to a graph picture variable.
XorPic	Displays the Graph screen and superimposes a stored graph picture by using XOR logic.

Drawing on the Graph Screen

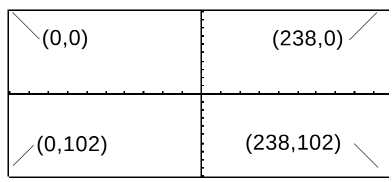
To create a drawing object on the Graph screen, use the commands listed in this section.

Pixel vs. Point Coordinates

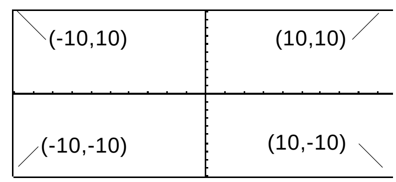
When drawing an object, you can use either of two coordinate systems to specify a location on the screen.

- **Pixel coordinates** — Refer to the pixels that physically make up the screen. These are independent of the viewing window because the screen is always 239 (0 to 238) pixels wide and 103 (0 to 102) pixels tall.
- **Point coordinates** — Refer to the coordinates in effect for the current viewing window (as defined in the Window Editor).

***Tip:** For information about pixel coordinates in split screens, refer to Chapter 5.*



Pixel coordinates (independent of viewing window)



Point coordinates (for standard viewing window)

***Note:** Pixel commands start with* Pxl*, such as* **PxlChg**.

Many drawing commands have two forms: one for pixel coordinates and one for point coordinates.

Erasing Drawn Objects

Command	Description
ClrDraw	Erases all drawn objects from the Graph screen.

Drawing a Point or Pixel

Command	Description
PtChg or PxlChg	Toggles (inverts) a pixel at the specified coordinates. **PtChg**, which uses point coordinates, affects the pixel closest to the specified point. If the pixel is off, it is turned on. If the pixel is on, it is turned off.
PtOff or PxlOff	Turns off (erases) a pixel at the specified coordinates. **PtOff**, which uses point coordinates, affects the pixel closest to the specified point.
PtOn or PxlOn	Turns on (displays) a pixel at the specified coordinates. **PtOn**, which uses point coordinates, affects the pixel closest to the specified point.
PtTest or PxlTest	Returns true or false to indicate if the specified coordinate is on or off, respectively.
PtText or PxlText	Displays a character string at the specified coordinates.

Drawing Lines and Circles

Command	Description
Circle or PxlCrcl	Draws, erases, or inverts a circle with a specified center and radius.
DrawSlp	Draws a line with a specified slope through a specified point.
Line or PxlLine	Draws, erases, or inverts a line between two sets of coordinates.
LineHorz or PxlHorz	Draws, erases, or inverts a horizontal line at a specified row coordinate.
LineTan	Draws a tangent line for a specified expression at a specified point. (This draws the tangent line only, not the expression.)
LineVert or PxlVert	Draws, erases, or inverts a vertical line at a specified column coordinate.

Drawing Expressions

Command	Description
DrawFunc	Draws a specified expression.
DrawInv	Draws the inverse of a specified expression.
DrawParm	Draws a parametric equation using specified expressions as its x and y components.
DrawPol	Draws a specified polar expression.
Shade	Draws two expressions and shades the areas where *expression1* < *expression2*.

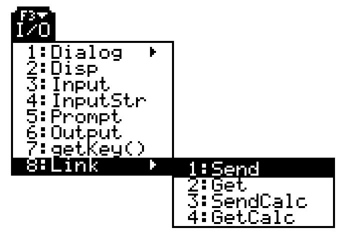
Accessing Another TI-92, a CBL 2/CBL, or a CBR

If you link two TI-92s (described in Chapter 18), programs on both units can transmit variables between them. If you link a TI-92 to a CBL 2/CBL or a CBR, a program on the TI-92 can access the CBL 2/CBL or CBR.

[F3] I/O Toolbar Menu

Use the Program Editor's [F3] I/O toolbar menu to enter the commands in this section.

1. Press [F3] and select 8:Link.
2. Select a command.



Accessing Another TI-92

When two TI-92s are linked, one acts as a receiving unit and the other as a sending unit.

*Note: For a sample program that synchronizes the receiving and sending units so that **GetCalc** and **SendCalc** are executed in the proper sequence, refer to "Transmitting Variables under Program Control" in Chapter 18.*

Command	Description
GetCalc	Executed on the receiving unit. Sets up the unit to receive a variable via the I/O port. • After the receiving unit executes **GetCalc**, the sending unit must execute **SendCalc**. • After the sending unit executes **SendCalc**, the sent variable is stored on the receiving unit (in the variable name specified by **GetCalc**).
SendCalc	Executed on the sending unit. Sends a variable to the receiving unit via the I/O port. • Before the sending unit executes **SendCalc**, the receiving unit must execute **GetCalc**.

Accessing a CBL 2/CBL or CBR

For additional information, refer to the manual that comes with the CBL 2/CBL or CBR unit.

Command	Description
Get	Gets a variable from an attached CBL 2/CBL or CBR and stores it in the TI-92.
Send	Sends a list variable from the TI-92 to the CBL 2/CBL or CBR.

Debugging Programs and Handling Errors

After you write a program, you can use several techniques to find and correct errors. You can also build an error-handling command into the program itself.

Run-Time Errors

The first step in debugging your program is to run it. The TI-92 automatically checks each executed command for syntax errors. If there is an error, a message indicates the nature of the error.

- To display the program in the Program Editor, press [ENTER]. The cursor appears in the approximate area of the error.

- To cancel program execution and return to the Home screen, press [ESC].

If your program allows the user to select from several options, be sure to run the program and test each option.

Debugging Techniques

Run-time error messages can locate syntax errors but not errors in program logic. The following techniques may be useful.

- During testing, do not use local variables so that you can check the variable values after the program stops. When the program is debugged, declare the applicable variables as local.
- Within a program, temporarily insert **Disp** and **Pause** commands to display the values of critical variables.
  - **Disp** and **Pause** cannot be used in a user-defined function. To temporarily change the function into a program, change **Func** and **EndFunc** to **Prgm** and **EndPrgm**. Use **Disp** and **Pause** to debug the program. Then remove **Disp** and **Pause** and change the program back into a function.
- To confirm that a loop is executed the correct number of times, display the counter variable or the values in the conditional test.
- To confirm that a subroutine is executed, display messages such as "Entering subroutine" and "Exiting subroutine" at the beginning and end of the subroutine.

Error-Handling Commands

Command	Description
Try...EndTry	Defines a program block that lets the program execute a command and, if necessary, recover from an error generated by that command.
ClrErr	Clears the error status and sets the error number in system variable Errornum to zero.
PassErr	Passes an error to the next level of the **Try...EndTry** block.

Example: Using Alternative Approaches

The preview at the beginning of this chapter shows a program that prompts the user to enter an integer, sums all integers from 1 to the entered integer, and displays the result. This section gives several approaches that you can use to achieve the same goal.

Example 1

This example is the program given in the preview at the beginning of the chapter. Refer to the preview for detailed information.

```
:prog1()
:Prgm
:Request "Enter an integer",n      Prompts for input in a dialog box.
:expr(n)→n                         Converts string entered with Request to an expression.
:0→temp
:For i,1,n,1                       ┐
:  temp+i→temp                     ├ Loop calculation.
:EndFor                            ┘
:Disp temp                         Displays output on Program I/O screen.
:EndPrgm
```

Example 2

This example uses **InputStr** for input, a **While...EndWhile** loop to calculate the result, and **Text** to display the result.

Tip: For ≤, type <=. For &, press [2nd] H.

```
:prog2()
:Prgm
:InputStr "Enter an integer",n     Prompts for input on Program I/O screen.
:expr(n)→n                         Converts string entered with InputStr to an expression.
:0→temp:1→i
:While i≤n                         ┐
:  temp+i→temp                     │ Loop calculation.
:  i+1→i                           │
:EndWhile                          ┘
:Text "The answer is "&string(temp)   Displays output in a dialog box.
:EndPrgm
```

Example 3

This example uses **Prompt** for input, **Lbl** and **Goto** to create a loop, and **Disp** to display the result.

***Note:** Because* **Prompt** *returns n as a number, you do not need to use* **expr** *to convert n.*

***Tip:** For ≤, type <=.*

```
:prog3()
:Prgm
:Prompt n                          Prompts for input on Program I/O screen.
:0→temp:1→i
:Lbl top                           ┐
:  temp+i→temp                     │
:  i+1→i                           │ Loop calculation.
:  If i≤n                          │
:    Goto top                      ┘
:Disp temp                         Displays output on Program I/O screen.
:EndPrgm
```

Example 4

This example uses **Dialog...EndDlog** to create dialog boxes for input and output. It uses **Loop...EndLoop** to calculate the result.

Defines a dialog box for input.

Converts string entered with **Request** to an expression.

Loop calculation.

Defines a dialog box for output.

```
:prog4()
:Prgm
:Dialog
:  Title "Enter an integer"
:  Request "Integer",n
:EndDlog
:expr(n)→n
:0→temp:0→i
:Loop
:  temp+i→temp
:  i+1→i
:  If i>n
:    Exit
:EndLoop
:Dialog
:  Title "The answer is"
:  Text string(temp)
:EndDlog
:EndPrgm
```

Example 5

This example uses the TI-92's built-in functions to calculate the result without using a loop.

Note: *Because* **Input** *returns n as a number, you do not need to use* **expr** *to convert n.*

Prompts for input on Program I/O screen.

Calculates sum.

Displays output on Program I/O screen.

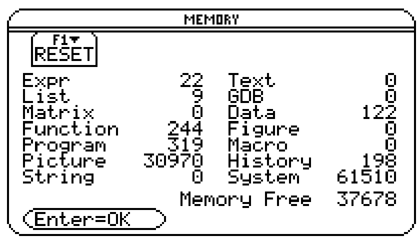
```
:prog5()
:Prgm
:Input "Enter an integer",n
:sum(seq(i,i,1,n))→temp
:Disp temp
:EndPrgm
```

Function	Used in this example to:
seq	Generate the sequence of integers from 1 to n. **seq**(*expression*, *var*, *low*, *high* [,*step*]) *step*: increment for *var*; if omitted, uses 1. *low*, *high*: initial and final values of *var* *var*: variable that will be incremented *expression*: expression used to generate the sequence
sum	Sum the integers in the list generated by **seq**.

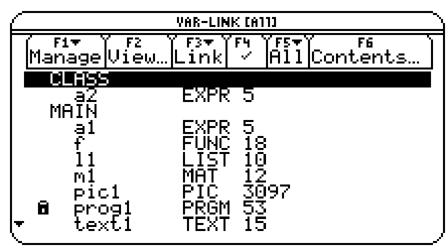
Memory and Variable Management

18

This chapter describes how you can manage the TI-92's memory, including the variables stored in memory, by using the MEMORY screen and the VAR-LINK screen.



The MEMORY screen shows how the memory is currently being used.



The VAR-LINK screen displays a list of defined variables and folders.

Note: *For information about using folders, refer to Chapter 10.*

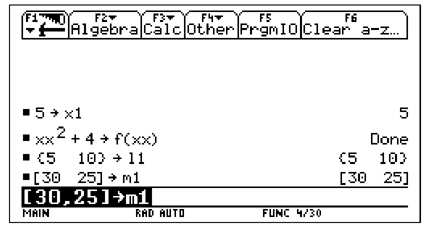
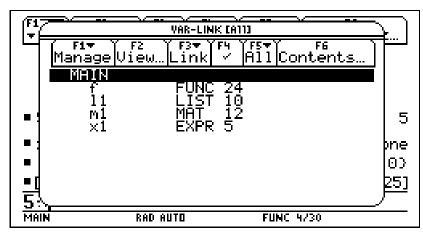
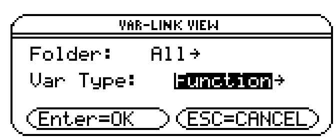
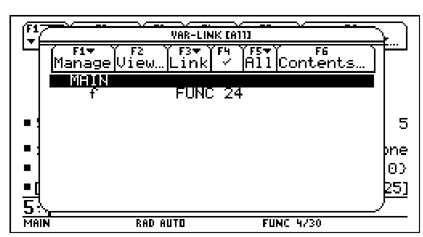
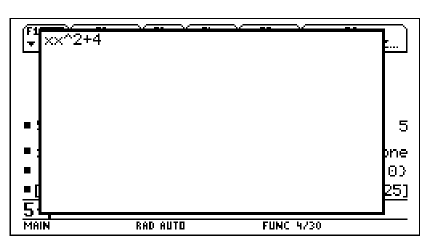
Note: *To communicate with a PC or Macintosh, you must use the TI-GRAPH LINK, an optional accessory.*

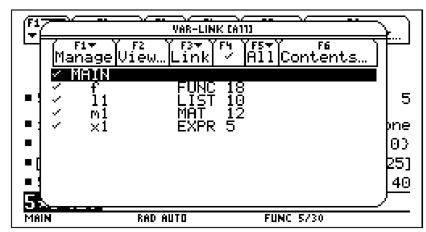
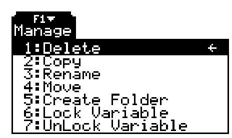
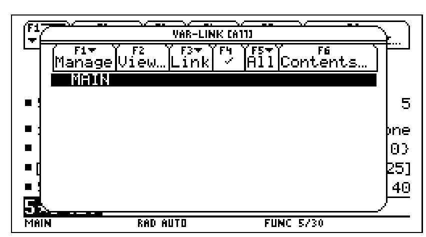
You can also use VAR-LINK to send/receive variables between two TI-92s or between the TI-92 and a personal computer. For information about:

- Linking two TI-92s, refer to the applicable section at the end of this chapter.
- Using the optional TI-GRAPH LINK™ to communicate with a PC or Macintosh, refer to the manual that comes with the TI-GRAPH LINK.

Preview of Memory and Variable Management

Assign values to different variables using a variety of data types. Use the VAR-LINK screen to view a list of the defined variables. Then delete the unused variables so that they will not take up memory.

Steps	Keystrokes	Display
1. From the Home screen, assign variables with the following variable types. Expression: 5 → x1 Function: xx²+4 → f(xx) List: {5,10} → l1 Matrix: [30,25] → m1	[♦] [HOME] 5 [STO▸] X 1 [ENTER] X X [^] 2 [+] 4 [STO▸] F [(] X X [)] [ENTER] [2nd] [{] 5 [,] 1 0 [2nd] [}] [STO▸] L 1 [ENTER] [2nd] [[] 3 0 [,] 2 5 [2nd] []] [STO▸] M 1 [ENTER]	
2. Suppose you start to perform an operation using a function variable but can't remember its name.	5 [×]	5∗
3. Display the VAR-LINK screen. By default, this screen lists all defined variables. *This example assumes that the variables assigned above are the only ones defined.*	[2nd] [VAR-LINK]	
4. Change the screen's view to show only function variables. *Although this may not seem particularly useful in an example with four variables, consider how useful it could be if there were many variables of all different types.*	[F2] ⊙ ⊙ 5 [ENTER]	 
5. Highlight the f function variable, and view its contents. *Notice that the function was assigned using f(xx) but is listed as f on the screen.*	⊙ [F6]	

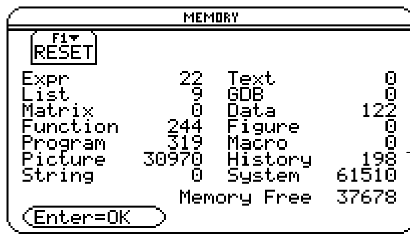
Steps	Keystrokes	Display
6. Close the Contents window.	[ESC]	
7. With the f variable still highlighted, close the VAR-LINK screen and paste the variable name to the entry line.	[ENTER]	5∗f(Notice that “ (” is pasted.
8. Complete the operation.	2 [)] [ENTER]	5∗f(2) 40
9. Redisplay the VAR-LINK screen. *The previous change in view is no longer in effect. The screen lists all defined variables.*	[2nd] [VAR-LINK]	
10. Use the [F5] All toolbar menu to select all variables. *A ✓ mark indicates items that are selected. Notice that this also selected the MAIN folder (see Step 13).* ***Note:** Instead of using [F5] (if you don’t want to delete all your variables), you can select individual variables. Highlight each item to delete and press [F4].*	[F5] 1	 
11. Use the [F1] Manage toolbar menu to delete.	[F1] 1	
12. Confirm the deletion.	[ENTER]	
13. Because [F5] 1 also selected the MAIN folder, an error message states that you cannot delete the MAIN folder. Acknowledge the message. *When VAR-LINK is redisplayed, the deleted variables are not listed.*	[ENTER]	
14. Close the VAR-LINK screen and return to the current application (Home screen in this example). *When you use [ESC] (instead of [ENTER]) to close VAR-LINK, the highlighted name is not pasted to the entry line.*	[ESC]	

Checking and Resetting Memory

The MEMORY screen shows how your TI-92's memory is being used. The screen shows the amount of memory (in bytes) used by each variable type and the amount of free memory. You can also use this screen to reset the memory.

Displaying the MEMORY Screen

1. Press [2nd] [MEM].



Size of history pairs saved in the Home screen's history area

2. To close the screen, press [ENTER].
   — or —
   To reset the memory, use the following procedure.

Resetting the Memory

From the MEMORY screen:

1. Press [F1].
2. Select the applicable item.

Note: Selecting 1:All *resets the display contrast to its factory setting. To adjust the contrast, use* [♦] [+] *and* [♦] [−].

Item	Description
All	Deletes all user-defined variables, functions, and folders; resets all system variables and modes to their original factory settings.
Memory	Deletes all user-defined variables, functions, and folders. This does not affect system variables (xmin, ymin, etc.) or mode settings.
Default	Resets all system variables and modes to their original factory settings. This does not affect any user-defined variables, functions, or folders.

Tip: *To cancel the reset, press* [ESC] *instead of* [ENTER].

3. When prompted for confirmation, press [ENTER].

   The TI-92 displays a message when the reset is complete.

4. Press [ENTER] to acknowledge the message.

To Delete Individual Variables

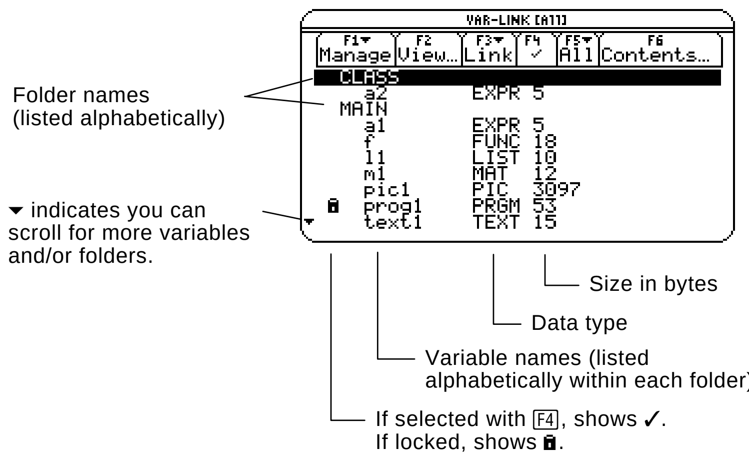
The MEMORY screen lets you delete *all* user-defined variables. To delete individual variables only, use the VAR-LINK screen ([2nd] [VAR-LINK]). Refer to page 333.

Displaying the VAR-LINK Screen

The VAR-LINK screen lists the variables and folders that are currently defined. After displaying the screen, you can manipulate the variables and/or folders as described in the remaining sections of this chapter.

Displaying the VAR-LINK Screen

Press [2nd] [VAR-LINK]. By default, the VAR-LINK screen lists all user-defined variables in all folders and with all data types.



Note: For information about using folders, refer to Chapter 10.

To scroll through the list:

- Press ↓ or ↑. (Use [2nd] ↓ or [2nd] ↑ to scroll one page at a time.)

  — or —
- Type a letter. If there are any variable names that start with that letter, the cursor moves to highlight the first of those variable names.

***Tip:** Type a letter repeatedly to cycle through the names that start with that letter.*

Variable Types as Listed on VAR-LINK

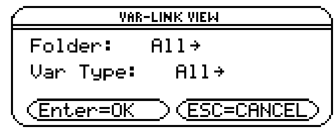
Type	Description
DATA	Data
EXPR	Expression (includes numeric values)
FIG	Geometry session
FUNC	Function
GDB	Graph database
LIST	List
MAC	Macro for a geometry session
MAT	Matrix
PIC	Picture of a graph
PRGM	Program
STR	String
TEXT	Text Editor session

Listing Only a Specified Folder and/or Variable Type

If you have a lot of variables and/or folders, it may be difficult to locate a particular variable. By changing VAR-LINK's view, you can specify the information you want to see.

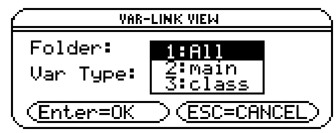
From the VAR-LINK screen:

1. Press F2 View.
2. Highlight the setting you want to change, and press ▶. This displays a menu of valid choices.



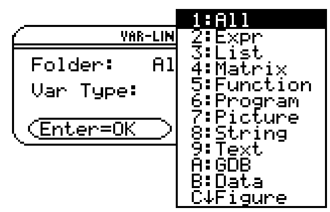
Tip: To cancel a menu, press ESC.

Folder — Always lists 1:All and 2:main, but lists other folders only if you have created them.



Tip: To list system variables (Y= Editor functions, window variables, etc.), select E:System, *the last item in the Var Type menu.*

Var Type — Lists the valid variable types.



↓ indicates that you can scroll for additional variable types.

3. Select the new setting.
4. When you are back on the VAR-LINK VIEW screen, press ENTER.

The VAR-LINK screen is updated to show only the specified folder and/or variable type.

Closing the VAR-LINK Screen

To close the VAR-LINK screen and return to the current application, use ENTER or ESC as described below.

Press:	To:
ENTER	Paste the highlighted variable or folder name to the cursor location in the current application.
ESC	Return to the current application without pasting the highlighted name.

Tip: For more information on using the ENTER paste feature, refer to page 335.

Manipulating Variables and Folders with VAR-LINK

On the VAR-LINK screen, you can show the contents of a variable. You can also select one or more listed items and manipulate them by using the operations in this section.

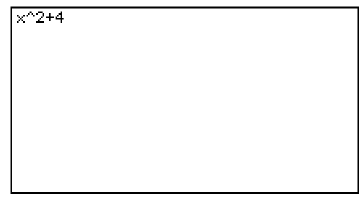
Showing the Contents of a Variable

Note: You cannot edit the contents from this screen.

You can show all variable types except DATA, FIG, GDB, and MAC. For example, you must open a FIG variable as a geometry session.

1. On VAR-LINK, move the cursor to highlight the variable.
2. Press [F6] Contents.

   If you highlight a folder, the screen shows the number of variables in that folder.
3. To return to VAR-LINK, press any key.

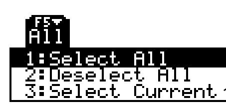


Selecting Items from the List

Note: If you use [F4] to ✓ one or more items and then highlight a different item, the following operations affect only the ✓'ed items.

For other operations, select one or more variables and/or folders.

To select:	Do this:
A single variable or folder	Move the cursor to highlight the item.
A group of variables or folders	Highlight each item and press [F4]. A ✓ is displayed to the left of each selected item. (If you select a folder, all variables in that folder are selected.) Use [F4] to select or deselect an item.
All folders and all variables	Press [F5] All and select 1:Select All.



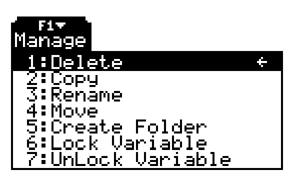
Selects the last set of items transmitted to your unit. Refer to page 336.

Deleting Variables or Folders

Tip: When you use [F4] to select a folder, its variables are selected automatically so that you can delete the folder and its variables at the same time.

To delete a folder, you must delete all of the variables in that folder. However, you cannot delete the MAIN folder even if it is empty.

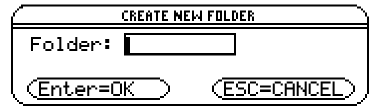
1. On VAR-LINK, select the variables and/or folders.
2. Press [F1] Manage and select 1:Delete. (You can press [←] instead of [F1] 1.)
3. To confirm the deletion, press [ENTER].



Creating a New Folder

For information about using folders, refer to Chapter 10.

1. On VAR-LINK, press [F1] Manage and select 5:Create Folder.
2. Type a unique name, and press [ENTER] twice.

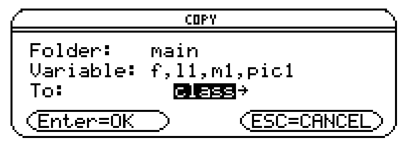


Copying or Moving Variables from One Folder to Another

You must have at least one folder other than MAIN. You cannot use VAR-LINK to copy variables within the same folder.

1. On VAR-LINK, select the variables.
2. Press [F1] Manage and select 2:Copy or 4:Move.
3. Select the destination folder.
4. Press [ENTER].

   The copied or moved variables retain their original names.



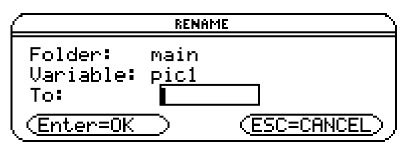
Tip: *To copy a variable to a different name in the same folder, use* [STO▸] *(such as* a1→a2) *or the* **CopyVar** *command from the Home screen.*

Renaming Variables or Folders

Remember, if you use [F4] to select a folder, the variables in that folder are selected automatically. As necessary, use [F4] to deselect individual variables.

1. On VAR-LINK, select the variables and/or folders.
2. Press [F1] Manage and select 3:Rename.
3. Type a unique name, and press [ENTER] twice.

   If you selected multiple items, you are prompted to enter a new name for each one.

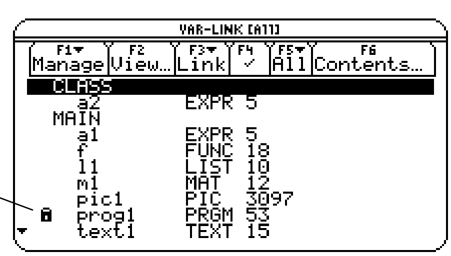


Locking or Unlocking Variables or Folders

When a variable is locked, you cannot delete, rename, or store to it. However, you can copy, move, or display its contents. When a folder is locked, you can manipulate the variables in the folder (assuming the variables are not locked), but you cannot delete the folder.

1. On VAR-LINK, select the variables and/or folders.
2. Press [F1] Manage and select 6:Lock Variable or 7:UnLock Variable.

🔒 indicates a locked variable or folder.



Pasting a Variable Name to an Application

Suppose you are typing an expression on the Home screen and can't remember which variable to use. You can display the VAR-LINK screen, select a variable from the list, and paste that variable name directly onto the Home screen's entry line.

Which Applications Can You Use?

From the following applications, you can paste a variable name to the current cursor location.

- Home screen or Y= Editor — The cursor must be on the entry line.
- Text Editor or Program Editor — The cursor can be anywhere on the screen.

Procedure

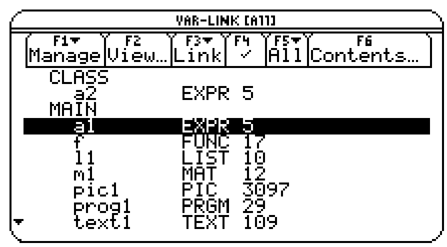
Starting from an application listed above:

1. Position the cursor where you want to insert the variable name.

```
sin(|
```

2. Press [2nd] [VAR-LINK].
3. Highlight the applicable variable.

Note: You can also highlight and paste folder names.



4. Press [ENTER] to paste the variable name.

```
sin(a1|
```

5. Finish typing the expression.

```
sin(a1)|
```

Note: *This pastes the variable's name, not its contents. (Use [2nd] [RCL], instead of [2nd] [VAR-LINK], to recall a variable's contents.)*

If you paste a variable name that is not in the current folder, the variable's pathname is pasted.

```
sin(class\a2|
```

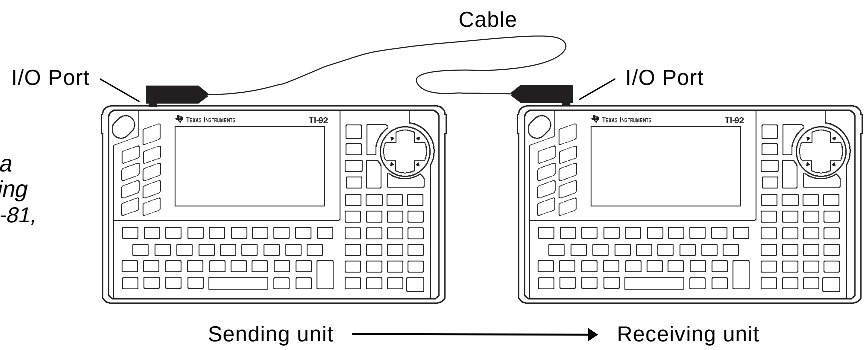
Assuming that CLASS is *not* the current folder, this is pasted if you highlight the a2 variable in CLASS.

Transmitting Variables between Two TI-92s

By linking two TI-92s, you can transmit variables and folders from one unit to the other. This is a convenient way to share any variable listed on the VAR-LINK screen — functions, text sessions, programs, etc.

Linking Two TI-92s

Your TI-92 comes with a cable that lets you link two units. Using firm pressure, insert each end of the cable into the I/O port of a TI-92. It doesn't matter which end of the cable goes into which unit.



Note: *You cannot link a TI-92 to another graphing calculator such as a TI-81, TI-82, or TI-85.*

One TI-92 acts as the sending unit; the other acts as the receiving unit. Either unit can send or receive, depending on how you set them up from the VAR-LINK screen.

Transmitting Variables

After linking the two units, use the following procedure to set up the receiving unit first. Then set up the sending unit.

Note: *If you set up the sending unit first, it may display an error message or it may remain BUSY until you cancel the transmission.*

On the:	Do this:
Receiving unit	1. Display the VAR-LINK screen ([2nd] [VAR-LINK]). 2. Press [F3] Link and select 2:Receive.  The message VAR-LINK: WAITING TO RECEIVE and the BUSY indicator are displayed in the status line.
Sending unit	1. Display the VAR-LINK screen ([2nd] [VAR-LINK]). 2. Select the variables to send, as described earlier in this chapter. 3. Press [F3] Link and select 1:Send. This starts the transmission.

Note: *Depending on transmission speed and variable sizes, messages in the status line may be displayed only briefly.*

- During transmission, messages are displayed in the status line of both units to show the name of each transmitted item.
- When transmission is complete, the VAR-LINK screen is updated on the receiving unit.

Rules for Transmitting Variables or Folders

If you select a:	What happens:
Variable (but not the folder it is in)	The variable is transmitted to the current folder on the receiving unit.
Folder	The folder and its contents are transmitted to the receiving unit. **Note:** If you use [F4] to select a folder, all variables in that folder are selected automatically. Use [F4] to deselect any variables that you do not want to transmit.

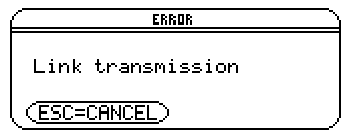
Canceling a Transmission

From either the sending or receiving unit:

1. Press [ON].

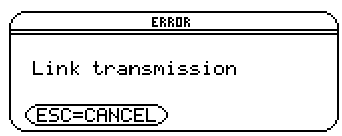
   An error message is displayed.

2. Press [ESC] or [ENTER].



Common Error and Notification Messages

__Note:__ The sending unit may not always display this message. Instead, it may remain BUSY until you cancel the transmission.

<table>
<tr><th>Shown on:</th><th>Message and Description</th></tr>
<tr><td>Sending unit</td><td>

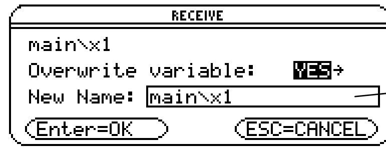
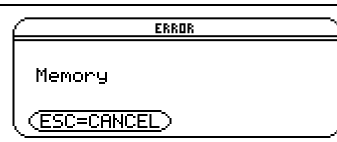
This is displayed after several seconds if:
• A cable is not attached to the sending unit’s I/O port.
— or —
• A receiving unit is not attached to the other end of the cable.
— or —
• The receiving unit is not set up to receive.

Press [ESC] or [ENTER] to cancel the transmission.</td></tr>
</table>

Common Error and Notification Messages (Continued)

Tip: *In the New Name input box, you can keep the same variable name and specify a different folder. For example:*

main\x1
└─ Variable name (x1)
└─ Folder name (main)

Shown on:	Message and Description
Receiving unit	 The receiving unit has a variable with the same name as the specified variable being sent. • To overwrite the existing variable, press [ENTER]. (By default, Overwrite variable = YES.) • To store the variable to a different name, set Overwrite variable = NO. In the New Name input box, type a variable name that does not exist in the receiving unit. Then press [ENTER] twice. • To skip this variable and continue with the next one, set Overwrite variable = SKIP and press [ENTER]. • To cancel the transmission, press [ESC].
Receiving unit	 The receiving unit does not have enough memory for the variable being sent. Press [ESC] or [ENTER] to cancel the transmission.

Transmitting Variables under Program Control

In a program, you can use the **GetCalc** and **SendCalc** instructions to transmit a variable between two linked TI-92s. However, the programs on the two units must be synchronized so that the receiving unit executes **GetCalc** before the sending unit executes **SendCalc**.

The "Chat" Program

The following program illustrates how to use **GetCalc** and **SendCalc**. The program sets up two loops (one for each of the linked TI-92s) so that the units can take turns sending and receiving/displaying a variable named msg. The **InputStr** instruction lets each user enter a message in the msg variable.

```
:Chat()
:Prgm
:ClrIO
:Disp "On first unit to send, enter 1;"
:InputStr "On first unit to receive, enter 0",msg
:If msg="0" Then
:  While true
:    GetCalc msg
:    Disp msg
:    InputStr msg
:    SendCalc msg
:  EndWhile
:Else
:  While true
:    InputStr msg
:    SendCalc msg
:    GetCalc msg
:    Disp msg
:  EndWhile
:EndIf
:EndPrgm
```

First loop (While true … EndWhile after If msg="0" Then): Loop executed by the unit that receives the first message.

- GetCalc msg / Disp msg: Sets up this unit to receive and display the variable msg.
- InputStr msg / SendCalc msg: Then lets this user enter a message in msg and send it.

Second loop (While true … EndWhile after Else): Loop executed by the unit that sends the first message.

- InputStr msg / SendCalc msg: Lets this user enter a message in msg and send it.
- GetCalc msg / Disp msg: Then sets up this unit to receive and display msg.

To synchronize **GetCalc** and **SendCalc**, the loops are arranged so that the receiving unit executes **GetCalc** while the sending unit is waiting for the user to enter a message.

Transmitting Variables under Program Control (Continued)

Running the Program

This procedure assumes that:

- Two TI-92s are linked with the connecting cable as described on page 336.
- The Chat program is loaded on both TI-92s.
  - Use each unit's Program Editor to enter the program.
    — or —
  - Enter the program on one unit and then use the VAR-LINK screen to transmit the program variable to the other unit, as described in the previous section.

Note: For information about using the Program Editor, refer to Chapter 17.

To run the program on both units:

1. On the Home screen of each unit, enter:

   chat()

2. When each unit displays its initial prompt, respond as shown below.

On the:	Type:
Unit that will send the first message	1 and press [ENTER].
Unit that will receive the first message.	0 and press [ENTER].

3. Take turns typing a message and pressing [ENTER] to send the variable msg to the other unit.

Stopping the Program

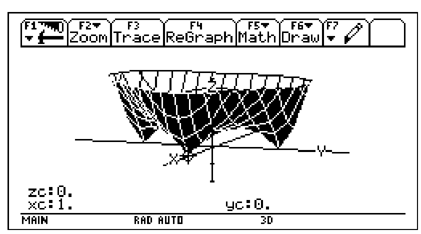
Because the Chat program sets up an infinite loop on both units, press [ON] (on both units) to break the program.

The program stops on the Program I/O screen. Press [F5] or [ESC] to return to the Home screen.

Applications

19

This chapter contains applications that show how the TI-92 can be used to solve, analyze, and visualize actual mathematical problems.

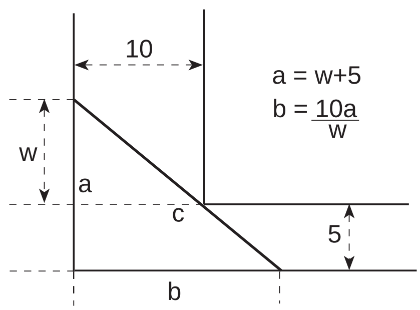


A ten-foot-wide hallway meets a five-foot-wide hallway in the corner of a building. Find the maximum length pole that can be moved around the corner without tilting the pole.

Maximum Length of Pole in Hallway

The maximum length of a pole c is the shortest line segment touching the interior corner and opposite sides of the two hallways as shown in the diagram below.

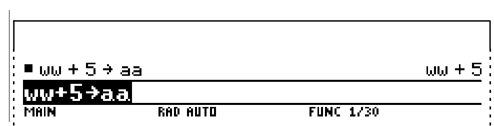
Hint: Use proportional sides and the Pythagorean theorem to find the length c with respect to w. Then find the zeros of the first derivative of c(w). The minimum value of c(w) is the maximum length of the pole.



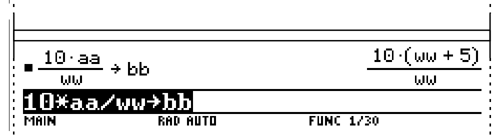
Tip: When you want to define a function, use multiple character names as you build the definition. (See page 213.)

Note: The maximum length of the pole is the minimum value of c(w).

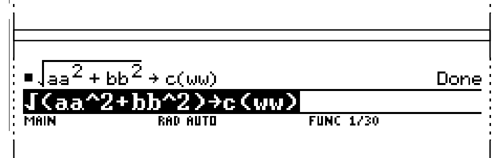
1. Enter the expression for side a in terms of ww and store it in aa.



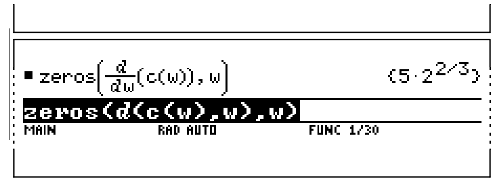
2. Enter the expression for side b in terms of ww and store it in bb.



3. Use the store (→) command to express the length of side c as a function of ww.

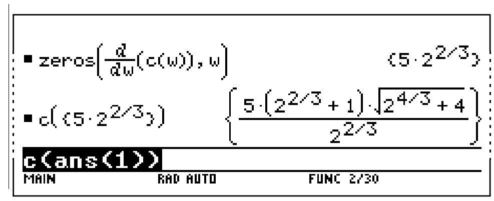


4. Use the **zeros()** command to compute the zeros of the first derivative of c(w) to find the minimum value of c(w).



5. Compute the exact maximum length of the pole.

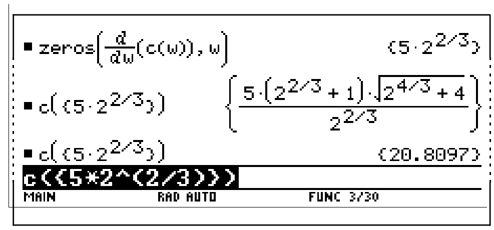
   Enter: c([2nd] [ANS])



__Hint:__ Use the auto-paste feature (page 42) to copy the result from step 4 to the entry line inside the parentheses of c( ) *and press* [♦] [ENTER].

6. Compute the approximate maximum length of the pole.

   Result: Approximately 20.8097 feet.



App. 2: Deriving the Quadratic Formula

This application shows you how to derive the quadratic formula:

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

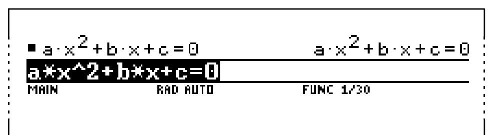
Detailed information about using the commands in this example can be found in Chapter 6: Symbolic Manipulation.

Performing Computations to Derive the Quadratic Formula

Perform the following steps to derive the quadratic formula by completing the square of the generalized quadratic equation.

Clear all one-character variables in the current folder by pressing [F6] [ENTER].

On the Home screen, enter the generalized quadratic equation: $ax^2+bx+c=0$.

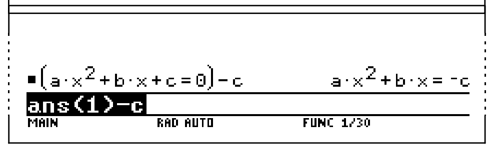


Note: This example uses the result of the last answer to perform computations on the TI-92. This feature reduces keystroking and chances for error.

Tip: *Continue to use the last answer ([2nd] [ANS]) as in step 3 in steps 4 through 9.*

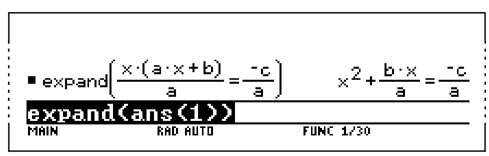
Subtract c from both sides of the equation.

Enter: [2nd] [ANS] - c

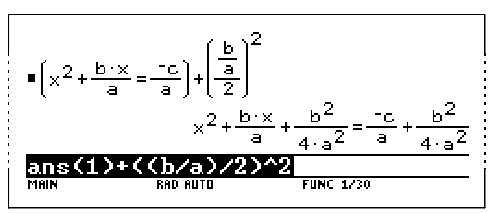


Divide both sides of the equation by the leading coefficient a.

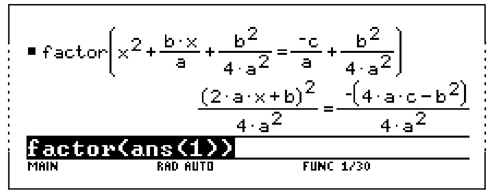
Use the **expand()** command to expand the result of the last answer.



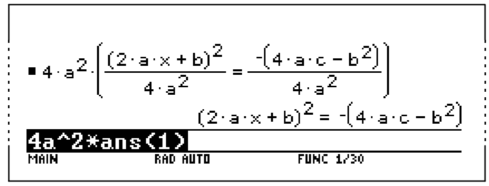
Complete the square by adding $((b/a)/2)^2$ to both sides of the equation.



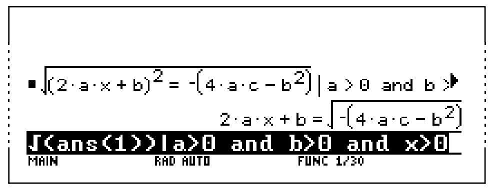
Factor the result using the **factor()** command.



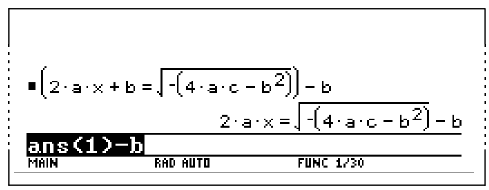
Multiply both sides of the equation by $4a^2$.



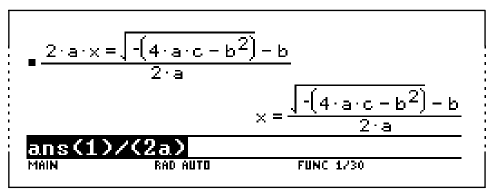
Take the square root of both sides of the equation with the constraint that a>0 and b>0 and x>0.



10. Solve for x by subtracting b from both sides and then dividing by 2a.



***Note:** This is only one of the two general quadratic solutions due to the constraint in step 9.*

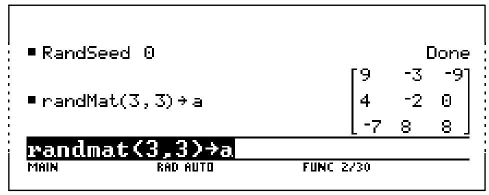


This application shows you how to perform several matrix operations.

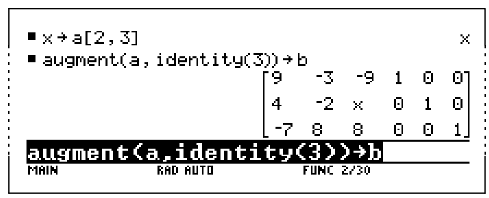
Exploring a 3x3 Matrix

Perform these steps to generate a random matrix, augment and find the identity matrix, and then solve to find an invalid value of the inverse.

On the Home screen, use **RandSeed** to set the random number generator seed to the factory default, and then use **randMat()** to create a random 3x3 matrix and store it in a.



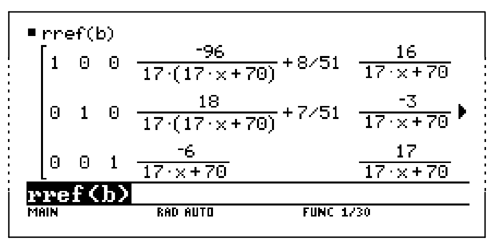
Replace the [2,3] element of the matrix with the variable x, and then use the **augment()** command, to augment the 3x3 identity to a and store the result in b.



Tip: *Use the cursor in the history area to scroll the result.*

Use **rref()** to "row reduce" matrix b:

The result will have the identity matrix in the first three columns and a^-1 in the last three columns.

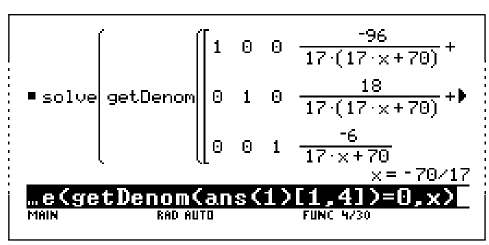


Tip: *Use the cursor in the history area to scroll the result.*

Solve for the value of x that will cause the inverse of the matrix to be invalid.

Enter: solve(getDenom([2nd] [ANS][1,4])=0,x)

Result: x= -70/17



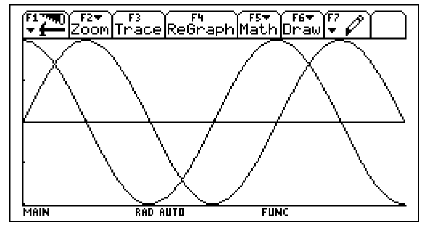
App. 4: Exploring cos(x) = sin(x)

This application uses two methods to find where cos(x) = sin(x) for the values of x between 0 and 3π.

Method 1: Graph Plot

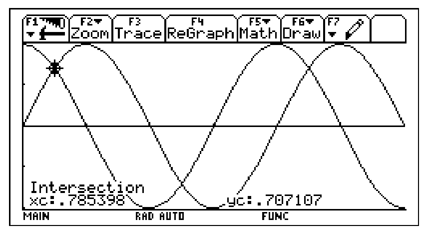
Perform the following steps to observe where the graphs of the functions y1(x)=cos(x) and y2(x)=sin(x) intersect.

1. In the Y= Editor, set y1(x)=cos(x) and y2(x)=sin(x).
2. In the Window Editor, set xmin=0 and xmax=3π.
3. Press [F2] and select A:ZoomFit.



Hint: ***Press*** *[F5] and select* 5:Intersection. *Respond to the screen prompts to select the two curves, and the lower and upper bounds for intersection* A.

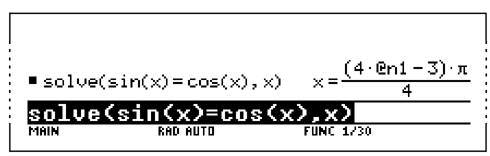
4. Find the intersection points of the two functions.
5. Note the x and y coordinates. (Repeat steps 4 and 5 to find the other intersections.)



Method 2: Symbolic Manipulation

Perform the following steps to solve the equation sin(x)=cos(x) with respect to x.

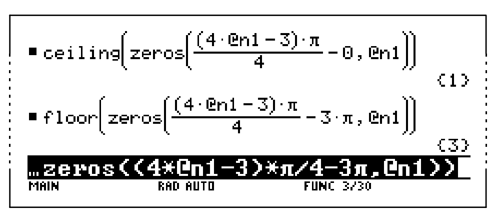
1. On the Home screen, enter solve(sin(x)= cos(x),x).



The solutions for x is where @n1 is any integer.

Hint: *Move the cursor into the history area to highlight the last answer. Press [ENTER] to copy the result of the general solution.*

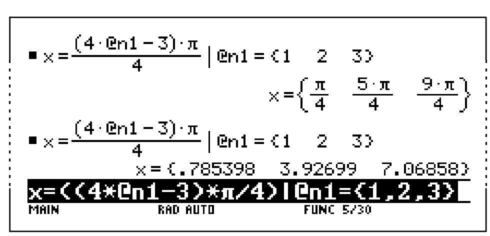
2. Using the ceiling and floor commands, find the ceiling and floor values for the intersection points as shown.



Hint*: Press [2nd] K to get the "with" [|] operator.*

3 Enter the general solution for x and apply the constraint for @n1 as shown.

Compare the result with Method 1.

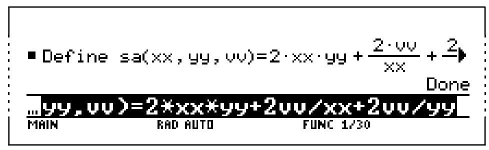


This application shows you how to find the minimum surface area of a parallelepiped having a constant volume V. Detailed information about the steps used in this example can be found in Chapter 6: Symbolic Manipulation and Chapter 14: 3D Graphing.

Exploring a 3D Graph of the Surface Area of a Parallelepiped

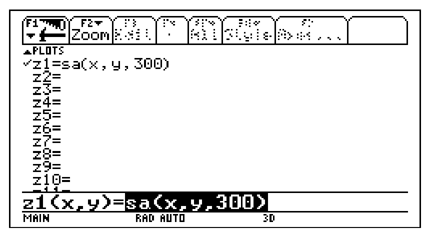
Perform the following steps to define a function for the surface area of a parallelepiped, draw a 3D graph, and use the **Trace** tool to find a point close to the minimum surface area.

1. On the Home screen, define the function sa(xx,yy,vv) for the surface area of a parallelepiped.



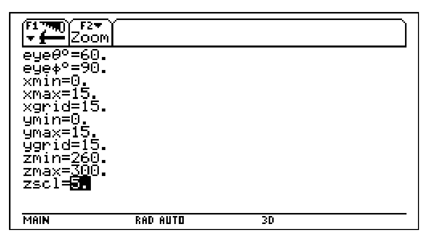
Enter: define
sa(xx,yy,vv)=2∗xx∗yy+
2vv/xx+2vv/yy

2. Select the 3D Graph mode. Then enter the function for z1(x,y) as shown in this example with volume v=300.

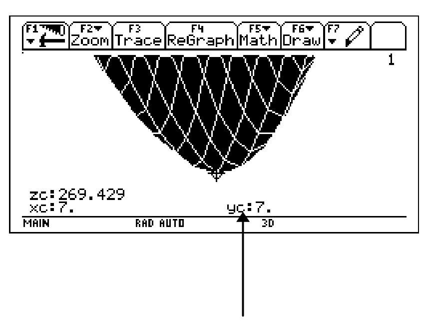


3. Set the Window variables to:

eye=	[60,90]
x=	[0,15,15]
y=	[0,15,15]
z=	[260,300,5]



4. Graph the function and use **Trace** to go to the point close to the minimum value of the surface area function.



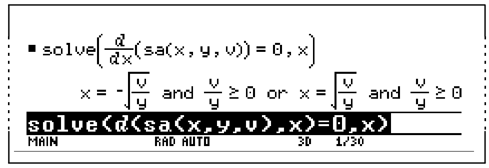
The Trace cursor is here.

Finding the Minimum Surface Area Analytically

Perform the following steps to solve the problem analytically on the Home screen.

1. Solve for x in terms of v and y.

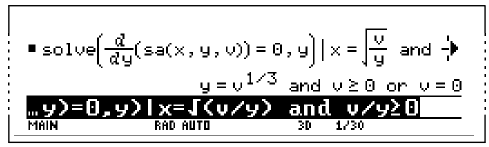
   Enter: solve(d(sa(x,y,v), x)=0,x)



Hint: Copy and paste the result from step 1 after the "with" symbol (|). Then edit to delete the negative solution.

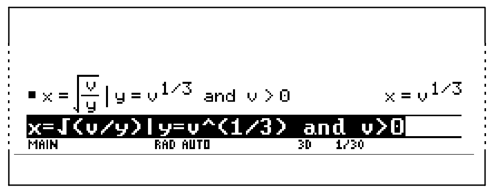
2. Solve for y in terms of v and x.

   Enter: solve(d(sa(x,y,v), y)=0,y)|x= (see Hint).



3. Evaluate for x in terms of v by substituting the y result into the result from step 1.

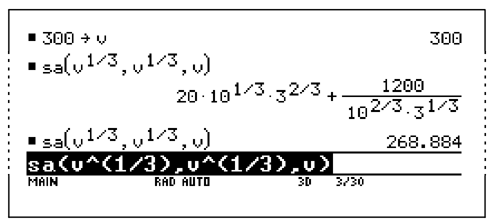
   Enter: x=√(v/y)| y=v^(1/3) and v>0



Hint: Press [ENTER] to obtain the exact result in symbolic form. Press [♦] [ENTER] to obtain the approximate result in decimal form.

4. Find the minimum surface area when the value of v equals 300.

   Enter: 300→v
   Enter: sa(v^(1/3), v^(1/3),v)



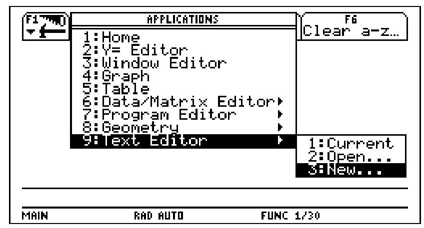
App. 6: Running a Tutorial Script Using the Text Editor

This application shows you how to use the Text Editor to run a tutorial script. Detailed information about text operations can be found in Chapter 16: Text Editor.

Running a Tutorial Script

Perform the following steps to write a script using the Text Editor, test each line, and observe the results in the history area on the Home screen.

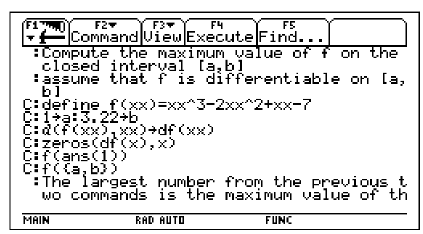
1. Open the Text Editor, and create a new variable named demo1.



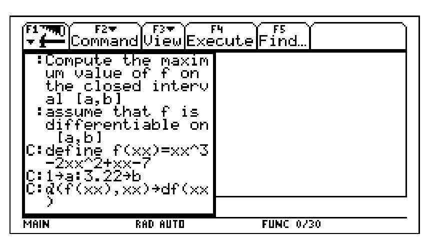
***Note:** The command symbol* "C" *is accessed from the* F2 1:Command *toolbar menu.*

2. Type the following lines into the Text Editor.

   : Compute the maximum value of f on the closed interval [a,b]
   : assume that f is differentiable on [a,b]
C : define f(xx)=xx^3−2xx^2+xx−7
C : 1→a:3.22→b
C : *d*(f(xx),xx)→df(xx)
C : zeros(df(x),x)
C : f(ans(1))
C : f({a,b})
   : The largest number from the previous two commands is the maximum value of the function. The smallest number is the minimum value.

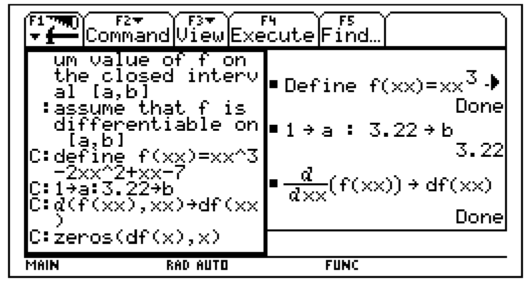


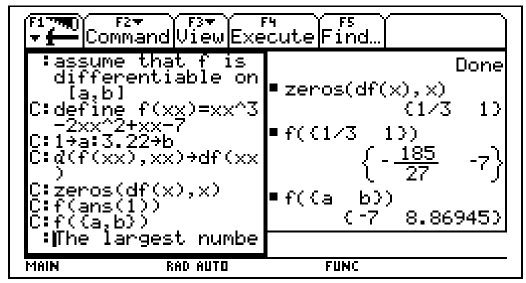
3. Press F3 and select 1:Script view to show the Text Editor and the Home screen on a split-screen. Move the cursor to the first line in the Text Editor.



Note: ***Press*** *[F3] and select* 2:Clear split *to go back to a full-sized Text Editor screen.*

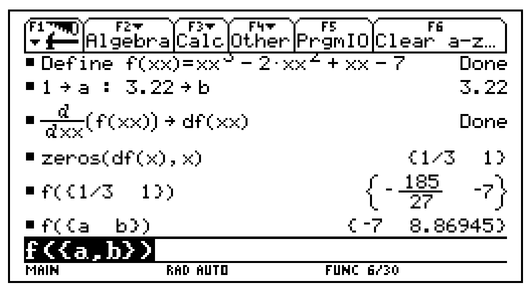
4. Press [F4] repeatedly to execute each line in the script one at a time.





Tip: ***Press*** *[2nd] [QUIT] twice to display the Home screen.*

5. To see the results of the script on a full-sized screen, go to the Home screen.



App. 7: Decomposing a Rational Function

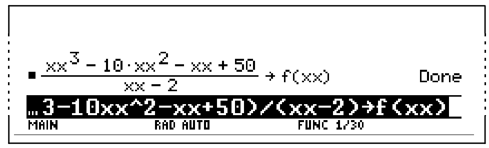
This application examines what happens when a rational function is decomposed into a quotient and remainder. Detailed information about the steps used in this example can be found in Chapter 3: Basic Function Graphing and Chapter 6: Symbolic Manipulation.

Decomposing a Rational Function

Note: ***Actual entries are displayed in reverse type in the example screens.***

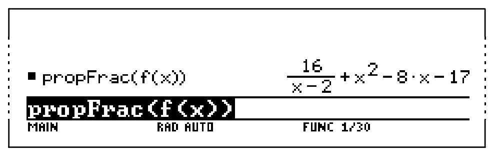
To examine the decomposition of the rational function f(x)=(x3−10x2−x+50)/(x−2) on a graph:

1. On the Home screen, enter the rational function as shown below and store it in a function f(xx).



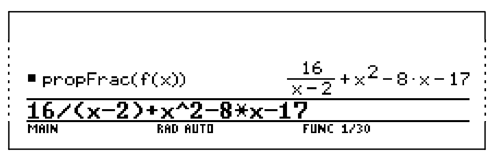
   Enter: (xx^3−10xx^2−xx+50)/(xx−2)→f(xx)

2. Use the proper fraction command (**propFrac**) to split the function into a quotient and remainder.

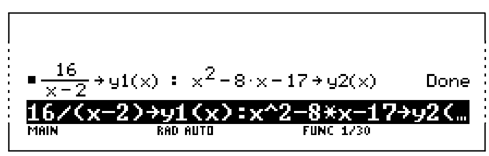


Hint: Move the cursor into the history area to highlight the last answer. Press [ENTER] *to copy it to the entry line .*

3. Copy the last answer to the entry line.
   —or—
   Enter: 16/(x−2)+x^2−8∗x−17

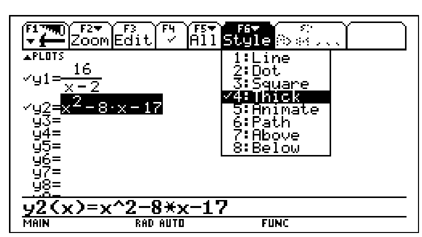


4. Edit the last answer in the entry line. Store the remainder to y1(x) and the quotient to y2(x) as shown.

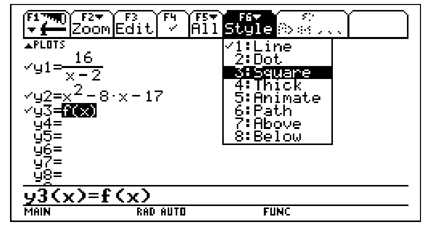


   Enter: 16/ (x−2)→y1(x): x^2−8∗x−17→y2(x)

5. In the Y= Editor, select the thick graphing style for y2(x).

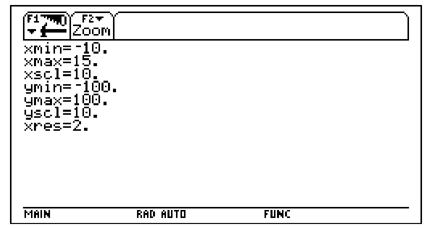


6. Add the original function f(x) to y3(x) and select the square graphing style.



7. In the Window Editor, set the window variables to:

   x= [⁻10,15,10]
   y= [⁻100,100,10]

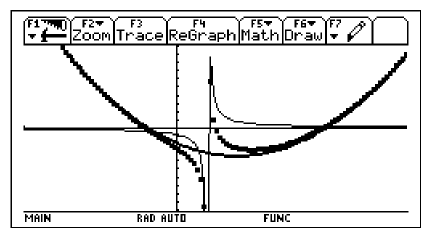


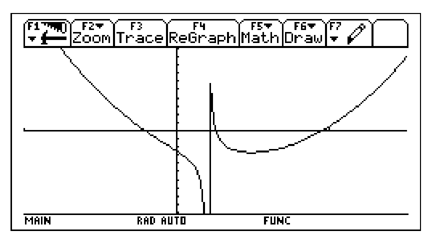
Note: Be sure the Graph mode is set to Function.

8. Draw the graph.

Observe that the global behavior of the f(x) function is basically represented by the quadratic quotient y2(x). The rational expression is basically a quadratic function as x gets very large in both the positive and negative directions.

The lower graph is y3(x)=f(x) graphed separately using the line style.





App. 8: Studying Statistics: Filtering Data by Categories

This application provides a statistical study of the weights of high school students using categories to filter the data. Detailed information about using the commands in this example can be found in Chapter 8: Data/Matrix Editor, and Chapter 9: Statistics and Data Plots.

Filtering Data by Categories

Each student is placed into one of eight categories depending on the student's sex and academic year (freshman, sophomore, junior, or senior). The data (weight in pounds) and respective categories are entered in the Data/Matrix Editor.

Table 1: Category vs. Description

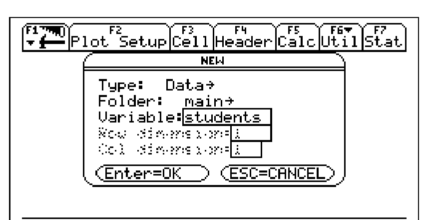
Category (C2)	Academic Year and Sex
1	Freshman boys
2	Freshman girls
3	Sophomore boys
4	Sophomore girls
5	Junior boys
6	Junior girls
7	Senior boys
8	Senior girls

Table 2: C1 (weight of each student in pounds) vs. C2 (category)

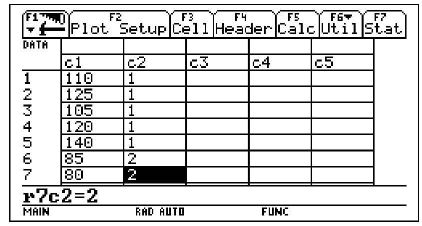
C1	C2	C1	C2	C1	C2	C1	C2
110	1	115	3	130	5	145	7
125	1	135	3	145	5	160	7
105	1	110	3	140	5	165	7
120	1	130	3	145	5	170	7
140	1	150	3	165	5	190	7
85	2	90	4	100	6	110	8
80	2	95	4	105	6	115	8
90	2	85	4	115	6	125	8
80	2	100	4	110	6	120	8
95	2	95	4	120	6	125	8

Perform the following steps to compare the weight of high school students to their year in school.

1. Start the Data/Matrix Editor, and create a new Data variable named students.

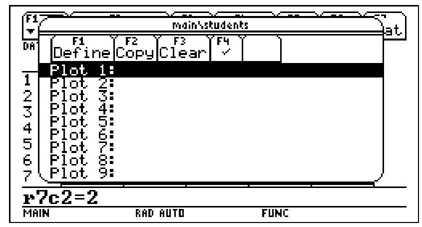


2. Enter the data and categories from Table 2 into columns c1 and c2, respectively.

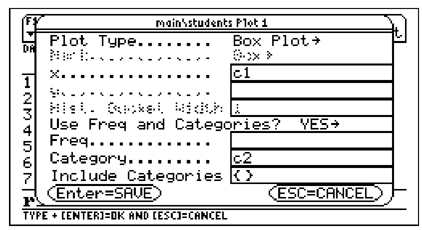


Note: Set up several box plots to compare different subsets of the entire data set.

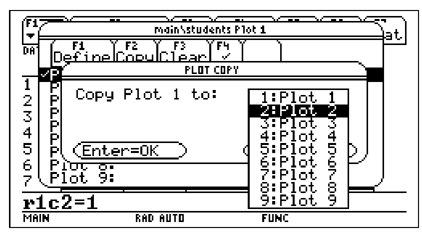
3. Open the [F2] Plot Setup toolbar menu.



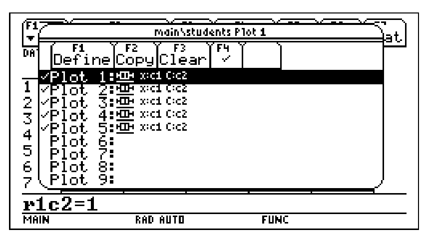
4. Define the plot and filter parameters for Plot 1 as shown in this screen.



5. Copy Plot 1 to Plot 2.

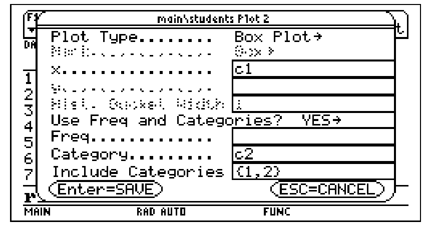


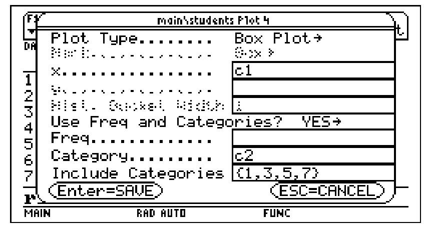
6. Repeat step 5 and copy Plot 1 to Plot 3, Plot 4, and Plot 5.



7. Press [F1], and modify the Include Categories item for Plot 2 through Plot 5 to the following:

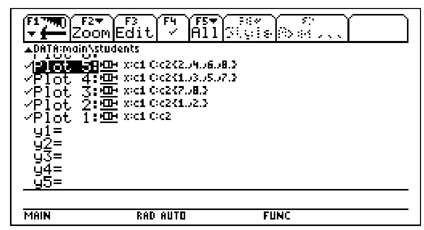
   Plot 2: {1,2}
   (freshman boys, girls)
   Plot 3: {7,8}
   (senior boys, girls)
   Plot 4: {1,3,5,7}
   (all boys)
   Plot 5: {2,4,6,8}
   (all girls)



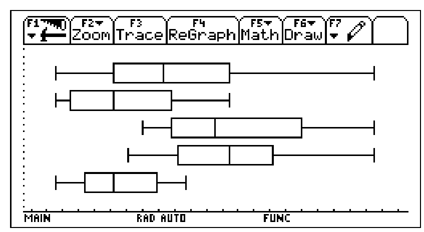


*Note: **Only Plot 1 through Plot 5 should be selected.***

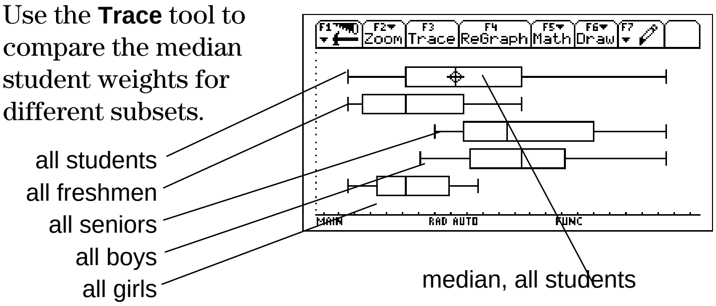
8. In the Y= Editor, deselect any functions that may be selected from a previous application.



9. Display the plots by pressing [F2] and selecting 9:Zoomdata.



10. Use the **Trace** tool to compare the median student weights for different subsets.



App. 9: CBL 2/CBL Program for the TI-92

This application provides a program that can be used when the TI-92 is connected to a Calculator-Based Laboratory™ (CBL 2™, CBL™)) unit. This program works with the "Newton's Law of Cooling" experiment and, with minor changes, the "Coffee To Go" experiment in the *CBL System Experiment Workbook*.

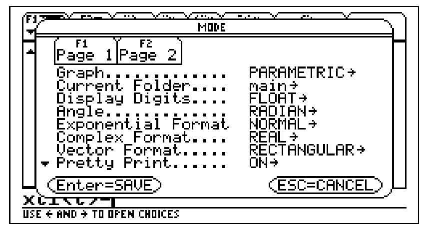
Program Instruction	Description
`:cooltemp()`	Program name
`:Prgm`	
`:Local i`	Declare local variable; exists only at run time.
`:setMode("Graph","FUNCTION")`	Set up the TI-92 for function graphing.
`:PlotsOff`	Turn off any previous plots.
`:FnOff`	Turn off any previous functions.
`:ClrDraw`	Clear any items previously drawn on graph screens.
`:ClrGraph`	Clear any previous graphs.
`:ClrIO`	Clear the TI-92 Program IO (input/output) screen.
`:-10→xmin`	Set up the Window variables.
`:99→xmax`	
`:10→xscl`	
`:-20→ymin`	
`:100→ymax`	
`:10→yscl`	
`:`	
`:{0}→data`	Create and/or clear a list named data.
`:{0}→time`	Create and/or clear a list named time.
`:Send{1,0}`	Send a command to clear the CBL 2/CBL unit.
`:Send{1,2,1}`	Set up Chan. 2 of the CBL 2/CBL to AutoID to record temperature.
`:Disp "Press ENTER to start graphing"`	Prompt the user to press ENTER.
`:Disp "Temperature."`	
`:Pause`	Wait until the user is ready to start.
`:PtText "TEMP(C)",2,99`	Label the y axis of the graph.
`:PtText "T(S)",80,-5`	Label the x axis of the graph.
`:Send{3,1,-1,0}`	Send the **Trigger** command to the CBL 2/CBL; collect data in real-time.
`:`	
`:For i,1,99`	Repeat next two instructions for 99 temperature readings.
`:Get data[i]`	Get a temperature from the CBL 2/CBL and store it in a list.
`:PtOn i,data[i]`	Plot the temperature data on a graph.
`:EndFor`	
`:seq(i,i,1,99,1)→time`	Create a list to represent time or data sample number.
`:NewPlot 1,1,time,data,,,,4`	Plot time and data using NewPlot and the **Trace** tool.
`:DispG`	Display the graph.
`:PtText "TEMP(C)",2,99`	Re-label the axes.
`:PtText "T(S)",80,-5`	
`:EndPrgm`	Stop the program.

This application uses the split screen settings to show a parametric graph and a table at the same time to study the flight of a hit baseball.

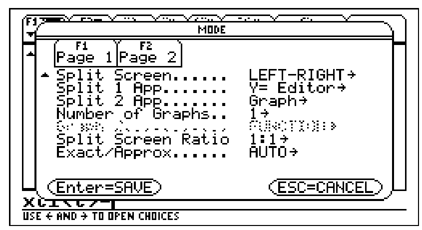
Setting Up a Parametric Graph and Table

Perform the following steps to study the flight of a hit baseball that has an initial velocity of 95 feet per second and an initial angle of 32 degrees.

1. Set the modes for Page 1 as shown in this screen.

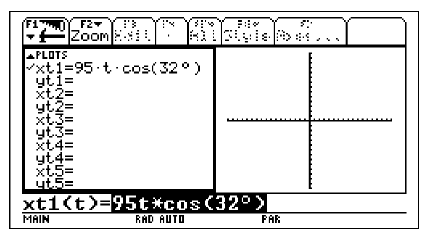


2. Set the modes for Page 2 as shown in this screen.

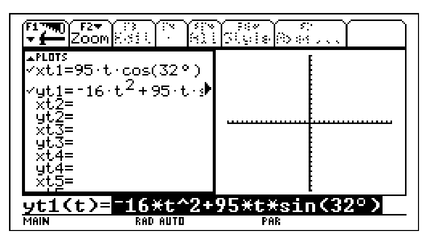


Hint: Press 2nd D to obtain the degree symbol.

3. In the Y= Editor on the left side, enter the equation for the distance of the ball at time t for xt1(t).

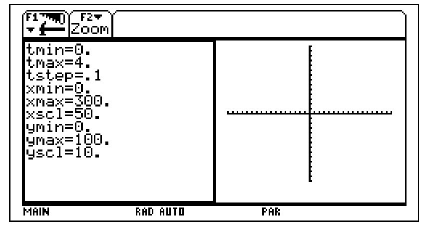


4. In the Y= Editor, enter the equation for the height of the ball at time t for yt1(t).



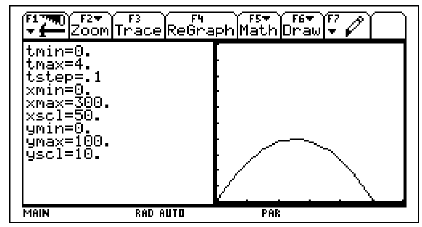
5. Set the Window variables to:

t values=	[0,4,.1]
x values=	[0,300,50]
y values=	[0,100,10]



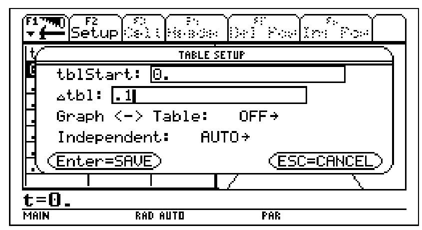
Hint: *Press* [2nd] [APPS].

6. Switch to the right side and display the graph.



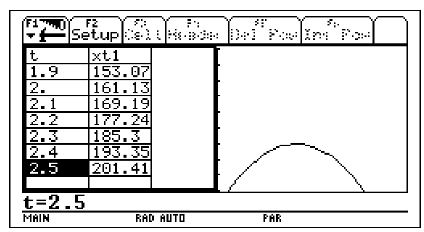
Hint: *Press* [♦] [TblSet] .

7. Display the TABLE SETUP dialog box, and change tblStart to 0 and Δtbl to 0.1.



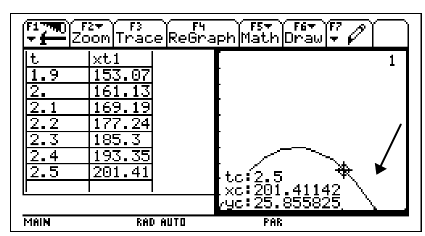
Hint: *Press* [♦] [TABLE].

8. Display the table in the left side and press ⓓ to highlight t=2.5.



Note: *As you move the trace cursor from* tc=0.0 *to* tc=3.1, *you will see the position of the ball at time* tc.

9. Switch to the right side. Press [F3], and trace the graph to show the values of xc and yc when tc=2.5.



Optional Exercise

Assuming the same initial velocity of 95 feet per second, find the angle that the ball should be hit to achieve the greatest distance.

App. 11: Visualizing Complex Zeros of a Cubic Polynomial

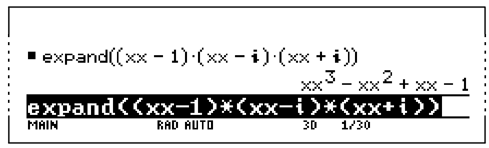
This application describes graphing the complex zeros of a cubic polynomial. Detailed information about the steps used in this example can be found in Chapter 6: Symbolic Manipulation and Chapter 14: 3D Graphing.

Visualizing Complex Roots

Perform the following steps to expand the cubic polynomial (x−1)(x−*i*)(x+*i*), find the absolute value of the function, graph the modulus surface, and use the **Trace** tool to explore the modulus surface.

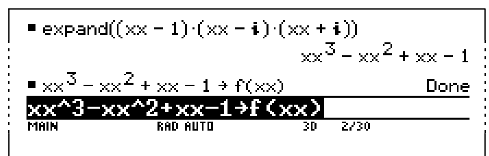
__Note:__ Actual entries are displayed in reverse type in the example screens.

1. On the Home screen, use the **expand** command to expand the cubic expression (xx−1)(xx−*i*) (xx+*i*) and see the first polynomial.



__Hint:__ Move the cursor into the history area to highlight the last answer and press [ENTER]*, or press* [♦] C *to copy and* [♦] V *to paste.*

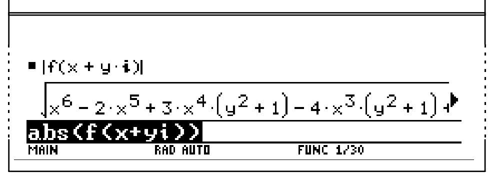
2. Copy and paste the last answer to the entry line and store it in the function f(xx).



__Note:__ The absolute value of a function forces any roots to visually just touch rather than cross the x axis. Likewise, the absolute value of a function of two variables will force any roots to visually just touch the xy plane.

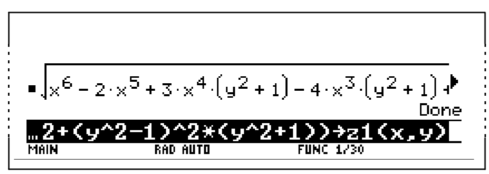
3. Use the **abs** command to find the absolute value of f(x+y*i*).

   (This calculation may take about 2 minutes.)



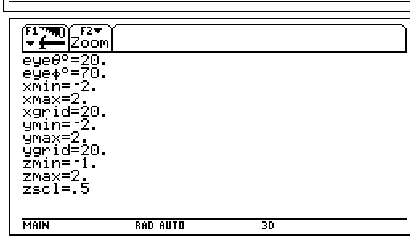
__Note:__ The graph of z1(x,y) *will be the modulus surface.*

4. Copy and paste the last answer to the entry line and store it in the function z1(x,y).



5. Set the unit to 3D graph mode, turn on the axes for graph format, and set the Window variables to:

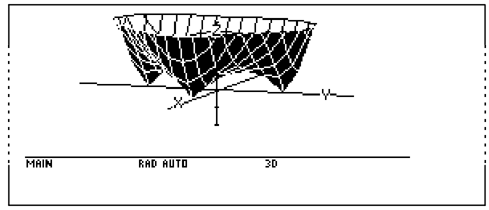
eye=	[20,70]
x=	[⁻2,2,20]
y=	[⁻2,2,20]
z=	[⁻1,2,.5]



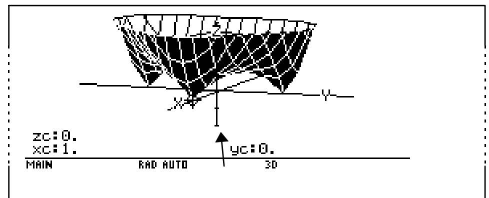
Note: Calculating and drawing the graph takes about three minutes.

6. Graph the modulus surface.

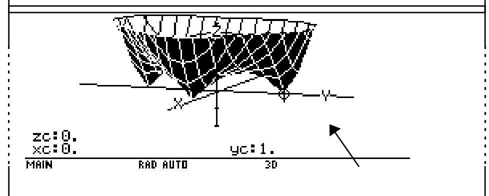
   The 3D graph is used to visually display a picture of the roots where the surface touches the xy plane.



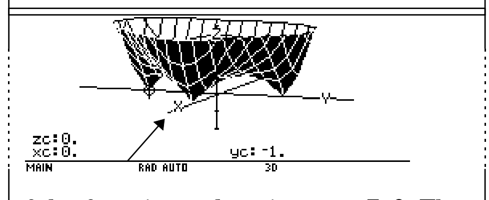
7. Use the **Trace** tool to explore the function values at x=1 and y=0.



8. Use the **Trace** tool to explore the function values at x=0 and y=1.



9. Use the **Trace** tool to explore the function values at x=0 and y=-1.



Summary

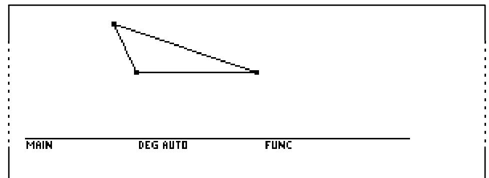
Note that zc is zero for each of the function values in steps 7–9. Thus, the complex zeros 1, $-i$, i of the polynomial x^3-x^2+x-1 can be visualized with the three points where the graph of the modulus surface touches the xy plane.

This application investigates the reflections of a point on the circumcircle of a triangle and the orthocenter.

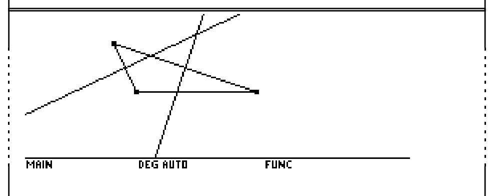
Creating the Construction

Perform the following steps to create the reflected points of a circle with respect to an inscribed triangle and the altitudes of the triangle.

1. Create a triangle that looks like the one shown to the right.



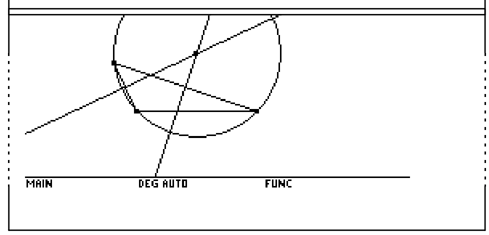
2. Construct perpendicular bisectors for two sides of the triangle.



Hint: ***The circle passes through each vertex of the triangle and its center point is the intersection of the perpendicular bisectors.***

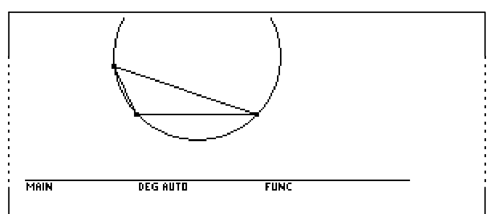
3. Create a circle to circumscribe the triangle.

3a. *(Optional)* Drag the triangle around to verify that the geometric constraints are correctly defined.

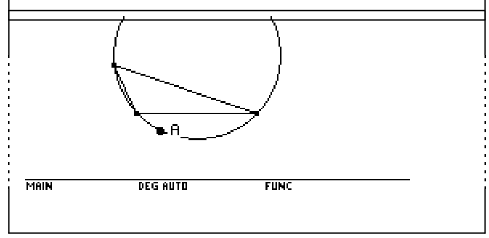


Hint: *Press* F7 *and select* 1:Hide/Show.

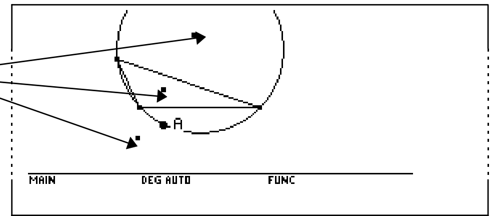
4. Hide the extraneous objects (two lines and center point of the circle).



5. Place and label a point anywhere on the circle as shown.

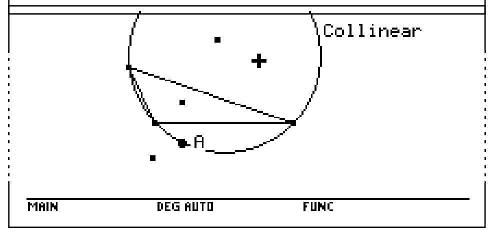


6. Create the reflections of point A with respect to each side of the triangle.



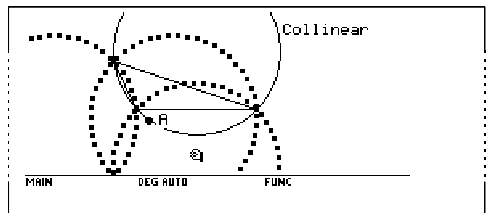
Hint: **Press** *F6* *and select* 8:Check Property.

7. Verify if the three points are collinear.
8. Drag point A around the circle while observing the three reflected points.



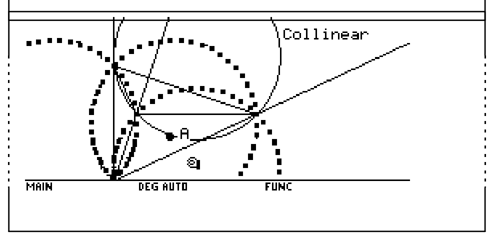
Hint: *Press F7 for both.*

9. Select each of the three reflected points for tracing, and then animate point A.



Hint: *Press ENTER to pause the animation. Press ENTER again to resume. Press ON to stop the animation.*

10. Pause or stop the animation, and draw the altitudes of the original triangle to construct the orthocenter.



Exploring Reflections and Orthocenters

In step 8, what do you notice about the three reflected points?

In step 9, what do you notice about the traces of the reflected points? Are the reflected points always collinear?

In step 10, what can you conclude about the intersection of the loci of the three reflected points and the intersection of the altitudes (orthocenter).

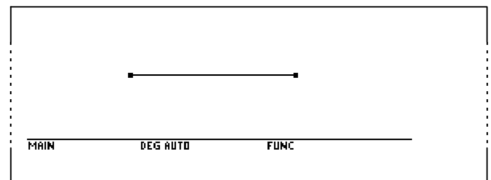
App. 13: Creating a Trisection Macro in Geometry

This application shows you how to create a macro in Geometry that can be used to trisect any segment or the side of any polygon.

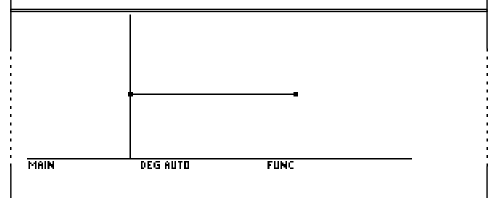
Trisecting a Segment

Although the TI-92 does not have a trisection tool, you can create a macro for one by first creating a trisection construction.

1. Create a segment.

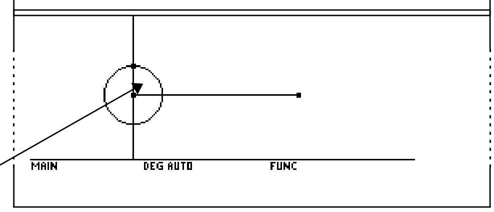


2. Construct a perpendicular line to the segment that passes through one of its endpoints.



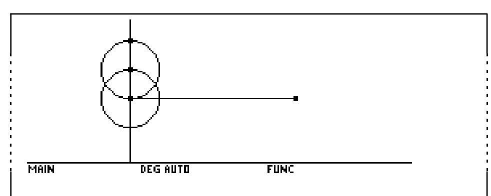
Note: *Create three circles that are on and attached to the perpendicular line such that the radius of each circle passes through the center point of the previous circle.*

3. Create a circle with its center point at the intersection of the endpoint of the segment and the perpendicular line (attach the circle to the perpendicular line).

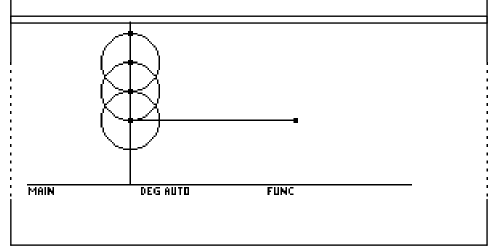


Note: *Attach the second and third circles to the perpendicular line.*

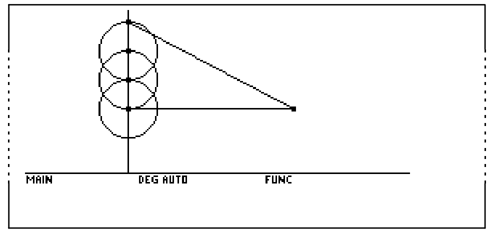
4. Create the second circle as shown.



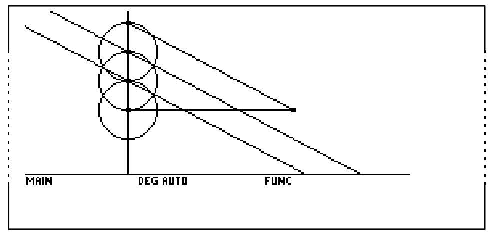
5. Create the third circle as shown.



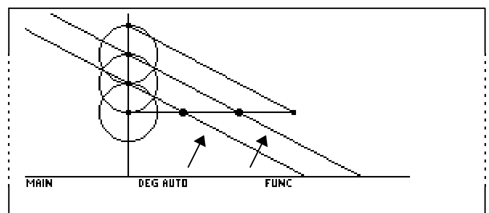
6. Create a second segment from the intersection of the top circle and the perpendicular line to the other endpoint of the first segment.



7. Create two lines both of which are parallel to the second segment and pass through the intersections of the circles on the perpendicular line.

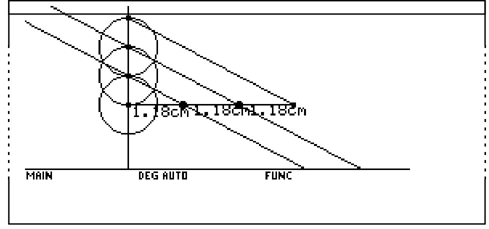


8. Create the intersection points where the two parallel lines intersect the first segment.



__Hint:__ You can verify your construction by dragging the endpoint of the first segment while observing the changes in the measured distance between the three sections.

9. (*Optional*) Measure the distance between the three sections of the first segment.

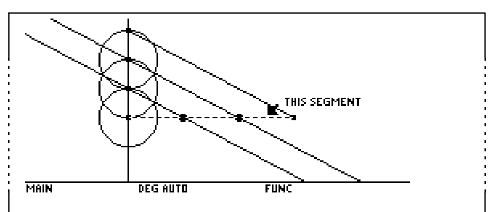


Creating the Trisection Macro

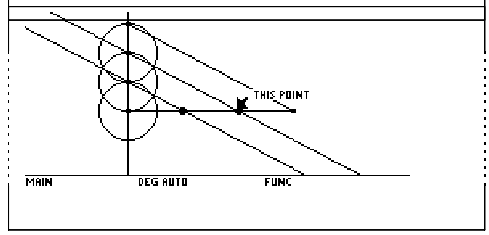
__Hint:__ Press [F4] *and select* 6:Macro Construction *before selecting* 2:Initial Objects *and* 3:Final Objects.

Perform the following steps to create a trisection macro.

1. Select the Initial Objects menu item, and then select the first segment.

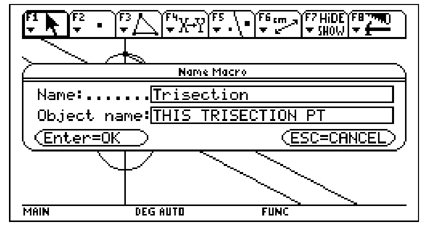


2. Select the Final Objects menu item, and then select the two trisection points.

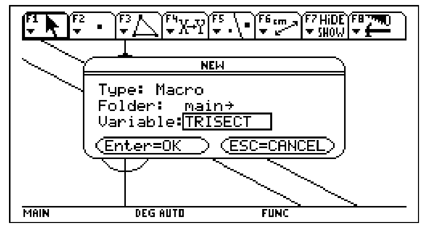


Creating the Trisection Macro (Continued)

3. Select the Define Macro menu item to enter the macro name and object name as shown.



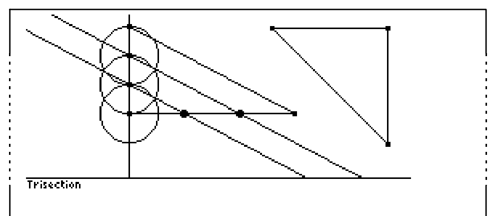
4. Select a folder and enter the name of the variable in which to save the macro.



Using the Trisection Macro

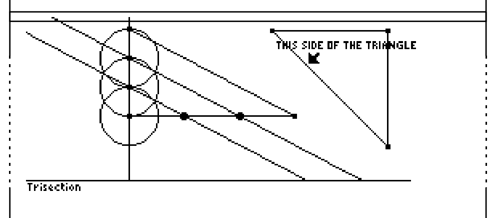
Perform the following steps to apply the Trisection macro to a segment or side of a triangle.

1. Create a triangle in your construction as shown.

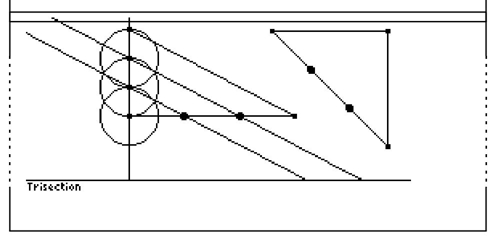


Hint: ***Press*** F4 6 *to open the* Macro Construction *menu and select* 1:Execute Macro.

2. Execute the Trisection macro, and then point to a side of the triangle.



3. When you press ENTER to apply the macro, the selected side is trisected.



Hint: *To open the macro, press* ◆ O*, select Type= Macro, and then select Variable= Trisect.*

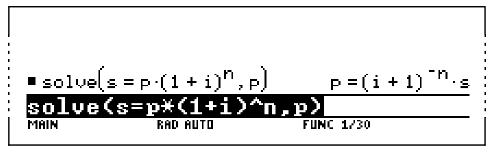
You can use the Trisection macro in other constructions by first opening the macro, and then selecting 1:Execute Macro from the Macro Construction dialog box.

This application can be used to find the interest rate, starting principal, number of compounding periods, and future value of an annuity.

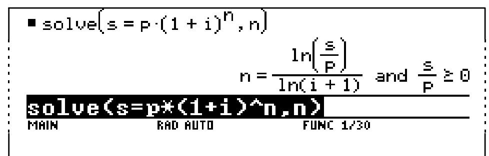
Finding the Interest Rate of an Annuity

Perform the following steps to find the interest rate (i) of an annuity where the starting principal (p) is 1,000, number of compounding periods (n) is 6, and the future value (s) is 2,000.

1. On the Home screen, enter the equation to solve for p.



2. Enter the equation to solve for n.



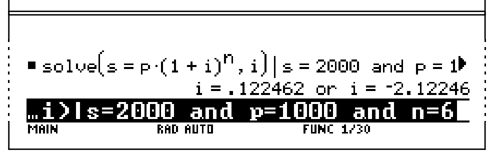
Tip: Press [2nd] K to enter the "with" (|) operator.

Tip: Press [♦] [ENTER] to obtain a floating-point result.

3 Enter the equation to solve for i using the "with" operator.

solve(s=p∗(1+i)^n,i) | s=2000 and p=1000 and n=6

Result: The interest rate is 12.246%.



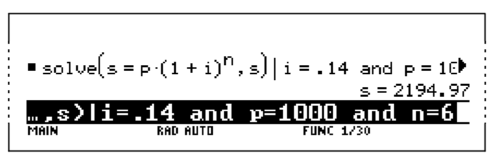
Finding the Future Value of an Annuity

Find the future value of an annuity using the values from the previous example where the interest rate is 14%.

Enter the equation to solve for s.

solve(s=p∗(1+i)^n,s)| i=.14 and p=1000 and n=6

Result: The future value at 14% interest is 2,194.97.



App. 15: Computing the Time-Value-of-Money

This application creates a function that can be used to find the cost of financing an item. Detailed information about the steps used in this example can be found in Chapter 17: Programming.

Time-Value-of-Money Function

In the Program Editor, define the following Time-Value-of-Money (tvm) function where temp1= number of payments, temp2= annual interest rate, temp3= present value, temp4= monthly payment, temp5=future value, and temp6=begin- or end-of-payment period (1=begining of month, 0=end of month).

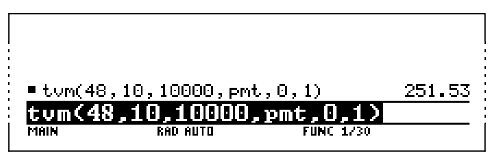
```
:tvm(temp1,temp2,temp3,temp4,temp5,temp6)
:Func
:Local tempi,tempfunc,tempstr1
: -temp3+(1+temp2/1200*temp6)*temp4*((1-(1+temp2/1200)^
      (-temp1))/(temp2/1200))-temp5*(1+temp2/1200)^(-temp1)
      →tempfunc
:For tempi,1,5,1
:"temp"&exact(string(tempi))→tempstr1
:If when(#tempstr1=0,false,false,true) Then
:If tempi=2
:Return approx(nsolve(tempfunc=0,#tempstr1) | #tempstr1>0 and
      #tempstr1<100)
:Return approx(nsolve(tempfunc=0,#tempstr1))
:EndIf
:EndFor
:Return "parameter error"
:EndFunc
```

Finding the Monthly Payment

Find the monthly payment on 10,000 if you make 48 payments at 10% interest per year.

On the Home screen, enter the tvm values to find pmt.

Result: The monthly payment is 251.53.

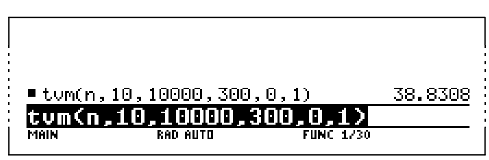


Finding the Number of Payments

Find the number of payments it will take to pay off the loan if you could make a 300 payment each month.

On the Home screen, enter the tvm values to find n.

Result: The number of payments is 38.8308.



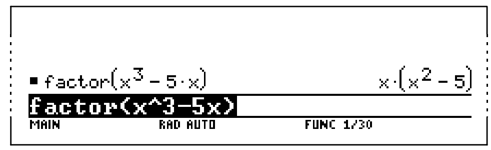
App. 16: Finding Rational, Real, and Complex Factors

This application shows how to find rational, real, or complex factors of expressions. Detailed information about the steps used in this example can be found in Chapter 6: Symbolic Manipulation.

Finding Factors

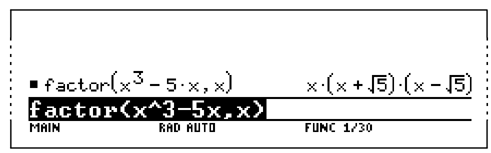
Enter the expressions shown below on the Home screen.

1. factor(x^3−5x) [ENTER] displays a rational result.

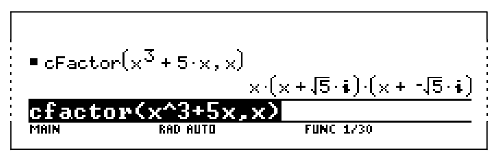


2. factor(x^3+5x) [ENTER] displays a rational result.

3. factor(x^3−5x,x) [ENTER] displays a real result.



4. cfactor(x^3+5x,x) [ENTER] displays a complex result.



App. 17: A Simple Function for Finding Eigenvalues

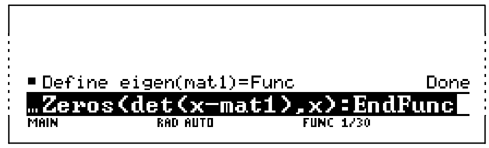
This application shows how to define a function to find the eigenvalues of a matrix.

Finding Eigenvalues

Perform the following steps to define a function to calculate eigenvalues.

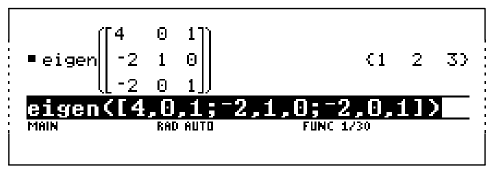
1. On the Home screen, enter the following function:

   define eigen(mat1)= func:Local x:Return cZeros (det(x−mat1), x):EndFunc



Note: The matrix must be of equal dimensions.

2. To find the eigenvalues of a matrix, substitute your values for those shown in the entry line. For example, enter: eigen([4,0,1;-2,1, 0;-2,0,1])



App. 18: Simulation of Sampling without Replacement

This application simulates drawing different colored balls from an urn without replacing them. Detailed information about the steps used in this example can be found in Chapter 17: Programming.

Sampling-without-Replacement Function

In the Program Editor, define drawball() as a function that can be called with two parameters. The first parameter is a list where each element is the number of balls of a certain color. The second parameter is the number of balls to select. This function returns a list where each element is the number of balls of each color that were selected.

```
:drawball(urnlist,drawnum)
:Func
:Local templist,drawlist,colordim,
   numballs,i,pick,urncum,j
:If drawnum>sum(urnlist)
:Return “too few balls”
:dim(urnlist)→colordim
:urnlist→templist
:newlist(colordim)→drawlist
:For i,1,drawnum,1
:sum(templist)→numballs
:rand(numballs)→pick
:For j,1,colordim,1
:cumSum(templist)→urncum
```

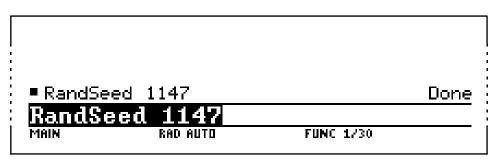
(continued in next column)

```
:If pick ≤ urncum[j] Then
:drawlist[j]+1→drawlist[j]
:templist[j]−1→templist[j]
:Exit
:EndIf
:EndFor
:EndFor
:Return drawlist
:EndFunc
```

Sampling without Replacement

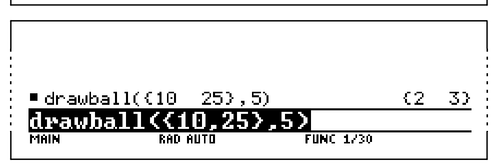
Suppose an urn contains *n1* balls of a color, *n2* balls of a second color, *n3* balls of a third color, etc. Simulate drawing balls without replacing them.

1. Enter a random seed using the **RandSeed** command.



2. Assuming the urn contains 10 red balls and 25 white balls, simulate picking 5 balls at random from the urn without replacement. Enter drawball({10,25},5).

   Result: 2 red balls and 3 white balls.



TI-92 Functions and Instructions

A

This appendix describes the syntax and the action of each TI-92 function and instruction.

Name of the function or instruction.

Key or menu for entering the name.
You can also type the name.

Example

Circle **CATALOG**

Circle *x*, *y*, *r* [, *drawMode*]

Draws a circle with its center at window coordinates (*x*, *y*) and with a radius of *r*.

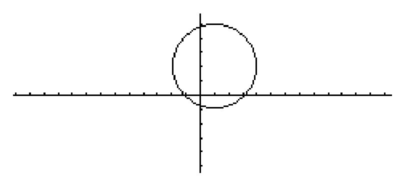
x, *y*, and *r* must be real values.

If *drawMode* = 1, draws the circle (default).
If *drawMode* = 0, turns off the circle.
If *drawMode* = -1, inverts pixels along the circle.

Note: Regraphing erases all drawn items.

In a ZoomSqr viewing window:

`ZoomSqr:Circle 1,2,3` ENTER



Arguments are shown in *italics*.
Arguments in [ ] brackets are optional.
Do not type the brackets.

Explanation of the function or instruction.

Syntax line shows the order and the type of arguments that you supply. Be sure to separate multiple arguments with a comma (**,**).

Quick-Find Locator

This section lists the TI-92 functions and instructions in functional groups along with the page numbers where they are described in this appendix.

Math

Matrices

Quick-Find Locator (Continued)

Programming

Statistics

Strings

Operations whose names are not alphabetic (such as +, !, and >) are listed at the end of this appendix, starting on page 458. Unless otherwise specified, all examples in this section were performed in the default reset mode, and all variables are assumed to be undefined. Additionally, due to formatting restraints, approximate results are truncated at three decimal places (3.14159265359 is shown as 3.141...).

abs() MATH/Number menu

abs(*expression1***)** ⇒ *expression*
abs(*list1***)** ⇒ *list*
abs(*matrix1***)** ⇒ *matrix*

Returns the absolute value of the argument.

If the argument is a complex number, returns the number's modulus.

Note: All undefined variables are treated as real variables.

```
abs({π/2, ⁻π/3}) [ENTER]          {π/2  π/3}
abs(2-3i) [ENTER]                 √13
abs(z) [ENTER]                    |z|
abs(x+yi) [ENTER]                 √(x²+y²)
```

and MATH/Test menu

Boolean expression1 **and** *expression2* ⇒ *Boolean expression*
Boolean list1 **and** *list2* ⇒ *Boolean list*
Boolean matrix1 **and** *matrix2* ⇒ *Boolean matrix*

Returns true or false or a simplified form of the original entry.

```
x≥3 and x≥4 [ENTER]                        x≥4
{x≥3,x≤0} and {x≥4,x≤⁻2} [ENTER]
                                 {x ≥ 4  x ≤ ⁻2}
```

AndPic CATALOG

AndPic *picVar*[, *row*, *column*]

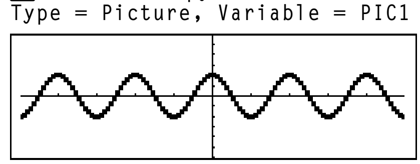
Displays the Graph screen and logically "ANDS" the picture stored in *picVar* and the current graph screen at pixel coordinates *(row, column)*.

picVar must be a picture type.

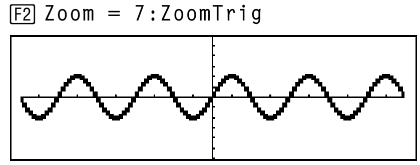
Default coordinates are (0,0), which is the upper left corner of the screen.

In function graphing mode and Y= Editor:

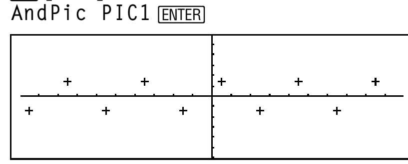
```
y1(x) = cos(x)
[F6] Style = 3:Square
[F2] Zoom = 7:ZoomTrig
[F1] = 2:Save Copy As...
Type = Picture, Variable = PIC1
```



```
y2(x) = sin(x)
[F6] Style = 3:Square
y1 = no checkmark (F4 to deselect)
[F2] Zoom = 7:ZoomTrig
```



```
[◆] [HOME]
AndPic PIC1 [ENTER]                     Done
```



angle() MATH/Complex menu

angle(*expression1***)** ⇒ *expression*

Returns the angle of *expression1*, interpreting *expression1* as a complex number.

Note: All undefined variables are treated as real variables.

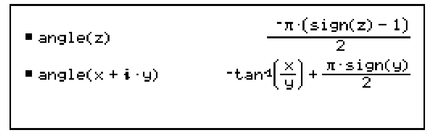
In Degree angle mode:

`angle(0+2i)` [ENTER] 90

In Radian angle mode:

`angle(1+i)` [ENTER] $\frac{\pi}{4}$

`angle(z)` [ENTER]
`angle(x+iy)` [ENTER]



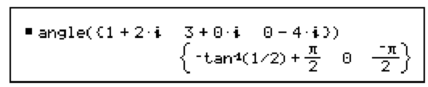
---

angle(*list1***)** ⇒ *list*
angle(*matrix1***)** ⇒ *matrix*

Returns a list or matrix of angles of the elements in *list1* or *matrix1*, interpreting each element as a complex number that represents a two-dimensional rectangular coordinate point.

In Radian angle mode:

`angle({1+2i,3+0i,0-4i})` [ENTER]



ans() [2nd] [ANS] key

ans() ⇒ *value*
ans(*integer***)** ⇒ *value*

Returns a previous answer from the Home screen history area.

integer, if included, specifies which previous answer to recall. Valid range for *integer* is from 1 to 99 and cannot be an expression. Default is 1, the most recent answer.

To use **ans()** to generate the Fibonacci sequence on the Home screen, press:

1 [ENTER]	1
1 [ENTER]	1
[2nd] [ANS] [+] [2nd] [ANS] [(] [←] 2 [ENTER]	2
[ENTER]	3
[ENTER]	5

approx() MATH/Algebra menu

approx(*expression***)** ⇒ *value*

Returns the evaluation of *expression* as a decimal value, when possible, regardless of the current Exact/Approx mode.

This is equivalent to entering *expression* and pressing [◆] [ENTER] on the Home screen.

`approx(π)` [ENTER] 3.141...

---

approx(*list1***)** ⇒ *list*
approx(*matrix1***)** ⇒ *matrix*

Returns a list or matrix where each element has been evaluated to a decimal value, when possible.

`approx({sin(π),cos(π)})` [ENTER]
{0. -1.}

`approx([√(2),√(3)])` [ENTER]
[1.414... 1.732...]

arcLen() MATH/Calculus menu

arcLen(*expression1*,*var*,*start*,*end***)** ⇒ *expression*

Returns the arc length of *expression1* from *start* to *end* with respect to variable *var*.

Regardless of the graphing mode, arc length is calculated as an integral assuming a function mode definition.

```
arcLen(cos(x),x,0,π) [ENTER]       3.820...
arcLen(f(x),x,a,b) [ENTER]
```

$$\int_a^b \sqrt{\left(\frac{d}{dx}(f(x))\right)^2+1}\,dx$$

---

arcLen(*list1*,*var*,*start*,*end***)** ⇒ *list*

Returns a list of the arc lengths of each element of *list1* from *start* to *end* with respect to *var*.

```
arcLen({sin(x),cos(x)},x,0,π)
                  {3.820...  3.820...}
```

augment() MATH/Matrix menu

augment(*list1*, *list2***)** ⇒ *list*

Returns a new list that is *list2* appended to the end of *list1*.

```
augment({1,⁻3,2},{5,4}) [ENTER]
                       {1 ⁻3 2 5 4}
```

---

augment(*matrix1*, *matrix2***)** ⇒ *matrix*

Returns a new matrix by appending *matrix2* to *matrix1* as new columns. Does not alter *matrix1* or *matrix2*.

Both arguments must have equal row dimensions.

```
[1,2;3,4]→M1 [ENTER]
```

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$

```
[5;6]→M2 [ENTER]
```

$\begin{bmatrix}5\\ 6\end{bmatrix}$

```
augment(M1,M2) [ENTER]
```

$\begin{bmatrix}1 & 2 & 5\\ 3 & 4 & 6\end{bmatrix}$

avgRC() CATALOG

avgRC(*expression1*, *var* [, *h*]**)** ⇒ *expression*

Returns the forward-difference quotient (average rate of change).

expression1 can be a user-defined function name (see **Func**, page 403).

h is the step value. If *h* is omitted, it defaults to 0.001.

Note that the similar function **nDeriv()** uses the central-difference quotient.

```
avgRC(f(x),x,h) [ENTER]
```

$\frac{f(x+h) - f(x)}{h}$

```
avgRC(sin(x),x,h)|x=2 [ENTER]
```

$\frac{\sin(h+2) - \sin(2)}{h}$

```
avgRC(x^2-x+2,x) [ENTER]     2.·(x-.4995)
avgRC(x^2-x+2,x,.1) [ENTER]
                             2.·(x-.45)
avgRC(x^2-x+2,x,3) [ENTER]       2·(x+1)
```

ceiling() MATH/Number menu

ceiling(*expression1***)** ⇒ *integer*

Returns the nearest integer that is ≥ the argument.

The argument can be a real or a complex number.

Note: See also **floor()** (page 400).

```
ceiling(0.456) [ENTER]                1.
```

---

ceiling(*list1***)** ⇒ *list*
ceiling(*matrix1***)** ⇒ *matrix*

Returns a list or matrix of the ceiling of each element.

```
ceiling({⁻3.1,1,2.5}) [ENTER]
                             {⁻3. 1 3.}
ceiling([0,⁻3.2i;1.3,4]) [ENTER]
```

$\begin{bmatrix}0 & -3.\cdot i\\ 2. & 4\end{bmatrix}$

cFactor() MATH/Algebra/Complex menu

cFactor(*expression1*[, *var*]**)** ⇒ *expression*
cFactor(*list1*[,*var*]**)** ⇒ *list*
cFactor(*matrix1*[,*var*]**)** ⇒ *matrix*

cFactor(*expression1***)** returns *expression1* factored with respect to all of its variables over a common denominator.

expression1 is factored as much as possible toward linear rational factors even if this introduces new non-real numbers. This alternative is appropriate if you want factorization with respect to more than one variable.

```
cFactor(a^3*x^2+a*x^2+a^3+a) [ENTER]
     a·(a + ⁻i)·(a + i)·(x + ⁻i)·(x + i)

cFactor(x^2+4/9) [ENTER]
     (3·x + ⁻2·i)·(3·x + 2·i)
     ------------------------
                9

cFactor(x^2+3) [ENTER]          x² + 3

cFactor(x^2+a) [ENTER]          x² + a
```

---

cFactor(*expression1,var***)** returns *expression1* factored with respect to variable *var*.

expression1 is factored as much as possible toward factors that are linear in *var*, with perhaps non-real constants, even if it introduces irrational constants or subexpressions that are irrational in other variables.

```
cFactor(a^3*x^2+a*x^2+a^3+a,x) [ENTER]
     a·(a² + 1)·(x + ⁻i)·(x + i)

cFactor(x^2+3,x) [ENTER]
     (x + √3·i)·(x + ⁻√3·i)

cFactor(x^2+a,x) [ENTER]
     (x + √a·⁻i)·(x + √a·i)
```

The factors and their terms are sorted with *var* as the main variable. Similar powers of *var* are collected in each factor. Include *var* if factorization is needed with respect to only that variable and you are willing to accept irrational expressions in any other variables to increase factorization with respect to *var*. There might be some incidental factoring with respect to other variables.

For the AUTO setting of the Exact/Approx mode, including *var* also permits approximation with floating-point coefficients where irrational coefficients cannot be explicitly expressed concisely in terms of the built-in functions. Even when there is only one variable, including *var* might yield more complete factorization.

```
cFactor(x^5+4x^4+5x^3-6x-3) [ENTER]
     x⁵ + 4·x⁴ + 5·x³ -6·x-3

cFactor(ans(1),x) [ENTER]
     (x -.965)·(x +.612)·(x + 2.13)·
     (x + 1.11 - 1.07·i)·(x + 1.11 + 1.07·i)
```

Note: See also **factor()** (page 399).

char() MATH/String menu

char(*integer***)** ⇒ *character*

Returns a character string containing the character numbered *integer* from the TI-92 character set. See Appendix B for a complete listing of TI-92 characters and their codes.

The valid range for *integer* is 0–255.

```
char(38) [ENTER]          "&"

char(65) [ENTER]          "A"
```

Circle CATALOG

Circle *x*, *y*, *r* [, *drawMode*]

Draws a circle with its center at window coordinates (*x*, *y*) and with a radius of *r*.

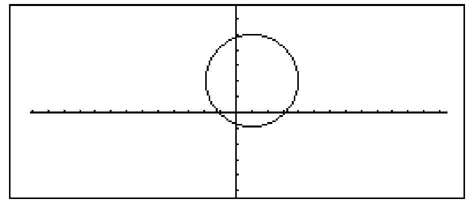
x, *y*, and *r* must be real values.

If *drawMode* = 1, draws the circle (default).
If *drawMode* = 0, turns off the circle.
If *drawMode* = -1, inverts pixels along the circle.

Note: Regraphing erases all drawn items. See also **PxlCrcl** (page 428).

In a ZoomSqr viewing window:

```
ZoomSqr:Circle 1,2,3 [ENTER]
```



ClrDraw CATALOG

ClrDraw

Clears the Graph screen and resets the Smart Graph feature so that the next time the Graph screen is displayed, the graph will be redrawn.

While viewing the Graph screen, you can clear all drawn items (such as lines and points) by pressing [F4] (ReGraph) or pressing [F6] and selecting 1:ClrDraw.

ClrErr CATALOG

ClrErr

Clears the error status. It sets errornum to zero and clears the internal error context variables.

The **Else** clause of the **Try...EndTry** in the program should use **ClrErr** or **PassErr**. If the error is to be processed or ignored, use **ClrErr**. If what to do with the error is not known, use **PassErr** to send it to the next error handler. If there are no more pending **Try...EndTry** error handlers, the error dialog box will be displayed as normal.

Note: See also **PassErr** (page 424) and **Try** (page 450).

Program listing:

```
:clearerr()
:Prgm
:PlotsOff:FnOff:ZoomStd
:For i,0,238
:Δx*i+xmin→xcord
: Try
:  PtOn xcord,ln(xcord)
: Else
:  If errornum=800 Then
:   ClrErr © clear the error
:  Else
:   PassErr © pass on any other
      error
:  EndIf
: EndTry
:EndFor
:EndPrgm
```

ClrGraph CATALOG

ClrGraph

Clears any functions or expressions that were graphed with the **Graph** command or were created with the **Table** command. (See **Graph** on page 406 or **Table** on page 447.)

Any previously selected Y= functions will be graphed the next time that the graph is displayed.

ClrHome CATALOG

ClrHome

Clears all items stored in the **entry()** and **ans()** Home screen history area.

Does not clear the current entry line.

While viewing the Home screen, you can clear the history area by pressing [F1] and selecting 8:Clear Home.

ClrIO CATALOG

ClrIO

Clears the Program I/O screen.

ClrTable CATALOG

ClrTable

Clears all table values. Applies only to the ASK setting on the Table Setup dialog box.

While viewing the Table screen in Ask mode, you can clear the values by pressing [F1] and selecting 8:Clear Table.

colDim() MATH/Matrix/Dimensions menu

colDim(*matrix***)** ⇒ *expression*

Returns the number of columns contained in *matrix*.

Note: See also **rowDim()** (page 435).

```
colDim([0,1,2;3,4,5]) [ENTER]          3
```

colNorm() MATH/Matrix/Norms menu

colNorm(*matrix***)** ⇒ *expression*

Returns the maximum of the sums of the absolute values of the elements in the columns in *matrix*.

Note: Undefined matrix elements are not allowed. See also **rowNorm()** (page 435).

```
[1,⁻2,3;4,5,⁻6]→mat [ENTER]
                              [1  ⁻2   3]
                              [4   5  ⁻6]
colNorm(mat) [ENTER]                   9
```

comDenom() MATH/Algebra menu

comDenom(*expression1*[,*var*]**)** ⇒ *expression*
comDenom(*list1*[,*var*]**)** ⇒ *list*
comDenom(*matrix1*[,*var*]**)** ⇒ *matrix*

comDenom(*expression1***)** returns a reduced ratio of a fully expanded numerator over a fully expanded denominator.

```
comDenom((y^2+y)/(x+1)^2+y^2+y) [ENTER]
```

$$\blacksquare\ \text{comDenom}\left(\frac{y^2+y}{(x+1)^2}+y^2+y\right) \quad \frac{x^2\cdot y^2+x^2\cdot y+2\cdot x\cdot y^2+2\cdot x\cdot y+2\cdot y^2+2\cdot y}{x^2+2\cdot x+1}$$

---

comDenom(*expression1*,*var***)** returns a reduced ratio of numerator and denominator expanded with respect to *var*. The terms and their factors are sorted with *var* as the main variable. Similar powers of *var* are collected. There might be some incidental factoring of the collected coefficients. Compared to omitting *var*, this often saves time, memory, and screen space, while making the expression more comprehensible. It also makes subsequent operations on the result faster and less likely to exhaust memory.

```
comDenom((y^2+y)/(x+1)^2+y^2+y,x) [ENTER]
```

$$\blacksquare\ \text{comDenom}\left(\frac{y^2+y}{(x+1)^2}+y^2+y,x\right) \quad \frac{x^2\cdot y\cdot(y+1)+2\cdot x\cdot y\cdot(y+1)+2\cdot y\cdot(y+1)}{x^2+2\cdot x+1}$$

```
comDenom((y^2+y)/(x+1)^2+y^2+y,y) [ENTER]
```

$$\blacksquare\ \text{comDenom}\left(\frac{y^2+y}{(x+1)^2}+y^2+y,y\right) \quad \frac{y^2\cdot(x^2+2\cdot x+2)+y\cdot(x^2+2\cdot x+2)}{x^2+2\cdot x+1}$$

If *var* does not occur in *expression1*, **comDenom(***expression1*,*var***)** returns a reduced ratio of an unexpanded numerator over an unexpanded denominator. Such results usually save even more time, memory, and screen space. Such partially factored results also make subsequent operations on the result much faster and much less likely to exhaust memory.

```
comDenom(exprn,abc)→comden(exprn)
[ENTER]                         Done
comden((y^2+y)/(x+1)^2+y^2+y)
[ENTER]
```

$$\blacksquare\ \text{comden}\left(\frac{y^2+y}{(x+1)^2}+y^2+y\right) \quad \frac{(x^2+2\cdot x+2)\cdot y\cdot(y+1)}{(x+1)^2}$$

Even when there is no denominator, the **comden** function is often a fast way to achieve partial factorization if **factor()** is too slow or if it exhausts memory.

```
comden(1234x^2*(y^3-y)+2468x*
(y^2-1)) [ENTER]
                 1234·x·(x·y + 2)·(y² -1)
```

Hint: Enter this **comden()** function definition and routinely try it as an alternative to **comDenom()** and **factor()**.

conj() MATH/Complex menu

conj(*expression1***)** ⇒ *expression*
conj(*list1***)** ⇒ *list*
conj(*matrix1***)** ⇒ *matrix*

Returns the complex conjugate of the argument.

Note: All undefined variables are treated as real variables.

```
conj(1+2i) [ENTER]                    1 - 2·i

conj([2,1-3i;-i,-7]) [ENTER]   [2   1+3·i]
                               [i      -7]

conj(z)                               z

conj(x+iy)                      x + -i·y
```

CopyVar CATALOG

CopyVar *var1*, *var2*

Copies the contents of variable *var1* to *var2*. If *var2* does not exist, **CopyVar** creates it.

Note: CopyVar is similar to the store instruction (→) when you are copying an expression, list, matrix, or character string except that no simplification takes place when using **CopyVar**. You must use **CopyVar** with non-algebraic variable types such as Pic and GDB variables.

```
x+y→a [ENTER]              x + y
10→x [ENTER]                  10
CopyVar a,b [ENTER]         Done
a→c [ENTER]               y + 10
DelVar x [ENTER]            Done
b [ENTER]                  x + y
c [ENTER]                 y + 10
```

cos() [COS] key

cos(*expression1***)** ⇒ *expression*
cos(*list1***)** ⇒ *list*

cos(*expression1***)** returns the cosine of the argument as an expression.

cos(*list1***)** returns a list of the cosines of all elements in *list1*.

Note: The argument is interpreted as either a degree or radian angle, according to the current angle mode setting. You can use ° (page 467) or ʳ (page 467) to override the angle mode temporarily.

In Degree angle mode:

cos((π/4)ʳ) [ENTER] $\frac{\sqrt{2}}{2}$

cos(45) [ENTER] $\frac{\sqrt{2}}{2}$

cos({0,60,90}) [ENTER] {1 1/2 0}

In Radian angle mode:

cos(π/4) [ENTER] $\frac{\sqrt{2}}{2}$

cos(45°) [ENTER] $\frac{\sqrt{2}}{2}$

cos⁻¹() [2nd] [COS⁻¹] key

cos⁻¹(*expression1***)** ⇒ *expression*
cos⁻¹(*list1***)** ⇒ *list*

cos⁻¹ (*expression1***)** returns the angle whose cosine is *expression1* as an expression.

cos⁻¹ (*list1***)** returns a list of the inverse cosines of each element of *list1*.

Note: The result is returned as either a degree or radian angle, according to the current angle mode setting.

In Degree angle mode:

cos⁻¹(1) [ENTER] 0

In Radian angle mode:

cos⁻¹({0,.2,.5}) [ENTER]
{$\frac{\pi}{2}$ 1.369... 1.047...}

cosh() MATH/Hyperbolic menu

cosh(*expression1***)** ⇒ *expression*
cosh(*list1***)** ⇒ *list*

cosh (*expression1***)** returns the hyperbolic cosine of the argument as an expression.

cosh (*list***)** returns a list of the hyperbolic cosines of each element of *list1*.

cosh(1.2) [ENTER] 1.810...

cosh({0,1.2}) [ENTER] {1 1.810...}

cosh⁻¹() MATH/Hyperbolic menu

cosh⁻¹(*expression1***)** ⇒ *expression*
cosh⁻¹(*list1***)** ⇒ *list*

cosh⁻¹ (*expression1***)** returns the inverse hyperbolic cosine of the argument as an expression.

cosh⁻¹ (*list1***)** returns a list of the inverse hyperbolic cosines of each element of *list1*.

cosh⁻¹(1) [ENTER] 0

cosh⁻¹({1,2.1,3}) [ENTER]
{0 1.372... cosh⁻¹(3)}

crossP() MATH/Matrix/Vector ops menu

crossP(*list1*, *list2***)** ⇒ *list*

Returns the cross product of *list1* and *list2* as a list.

list1 and *list2* must have equal dimension, and the dimension must be either 2 or 3.

```
crossP({a1,b1},{a2,b2}) [ENTER]
                    {0 0 a1•b2−a2•b1}

crossP({0.1,2.2,⁻5},{1,⁻.5,0}) [ENTER]
                    {⁻2.5 ⁻5. ⁻2.25}
```

---

crossP(*vector1*, *vector2***)** ⇒ *vector*

Returns a row or column vector (depending on the arguments) that is the cross product of *vector1* and *vector2*.

Both *vector1* and *vector2* must be row vectors, or both must be column vectors. Both vectors must have equal dimension, and the dimension must be either 2 or 3.

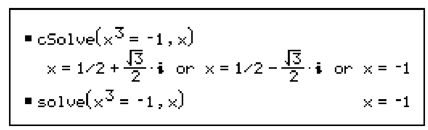
```
crossP([1,2,3],[4,5,6]) [ENTER]
                    [⁻3 6 ⁻3]

crossP([1,2],[3,4]) [ENTER]
                    [0 0 ⁻2]
```

cSolve() MATH/Algebra/Complex menu

cSolve(*equation*, *var***)** ⇒ *Boolean expression*

Returns candidate complex solutions of an equation for *var*. The goal is to produce candidates for all real and non-real solutions. Even if *equation* is real**, cSolve()** allows non-real results in real mode.

```
cSolve(x^3=⁻1,x) [ENTER]
solve(x^3=⁻1,x) [ENTER]
```



Although the TI-92 processes all undefined variables as if they were real, **cSolve()** can solve polynomial equations for complex solutions. (See also “Using Undefined or Defined Variables” in Chapter 6: Symbolic Manipulation.)

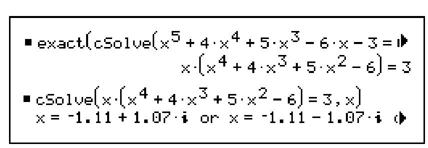
cSolve() temporarily sets the domain to complex during the solution even if the current domain is real. In the complex domain, fractional powers having odd denominators use the principal rather than the real branch. Consequently, solutions from **solve()** to equations involving such fractional powers are not necessarily a subset of those from **cSolve()**.

```
cSolve(x^(1/3)=⁻1,x) [ENTER]      false

solve(x^(1/3)=⁻1,x) [ENTER]       x = ⁻1
```

cSolve() starts with exact symbolic methods. Except in EXACT mode, **cSolve()** also uses iterative approximate complex polynomial factoring, if necessary.

Display Digits mode in Fix 2:

```
exact(cSolve(x^5+4x^4+5x^3-6x-3=0,
  x)) [ENTER]
cSolve(ans(1),x) [ENTER]
```



Note: See also **cZeros()** (page 387), **solve()** (page 442), and **zeros()** (page 453).

CubicReg MATH/Statistics/Regressions menu

CubicReg *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

Calculates the cubic polynomial regression and updates all the statistics variables.

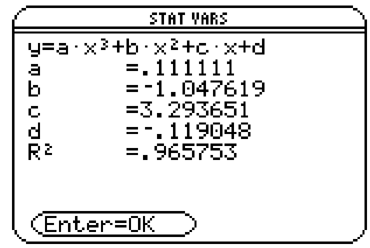
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

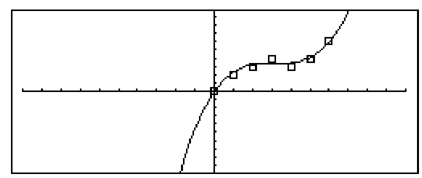
In function graphing mode.

```
{0,1,2,3,4,5,6}→L1 [ENTER]      {0 1 2 …}
{0,2,3,4,3,4,6}→L2 [ENTER]      {0 2 3 …}
CubicReg L1,L2 [ENTER]               Done
ShowStat [ENTER]
```




```
[ENTER]
regeq(x)→y1(x) [ENTER]               Done
NewPlot 1,1,L1,L2 [ENTER]            Done
```

[◆] [GRAPH]



cumSum() MATH/List menu

cumSum(*list1***)** ⇒ *list*

Returns a list of the cumulative sums of the elements in *list1*, starting at element 1.

```
cumSum({1,2,3,4}) [ENTER]     {1 3 6 10}
```

---

cumSum(*matrix1***)** ⇒ *matrix*

Returns a matrix of the cumulative sums of the elements in *matrix1*. Each element is the cumulative sum of the column from top to bottom.

[1,2;3,4;5,6]→m1 [ENTER] $\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}$

cumSum(m1) [ENTER] $\begin{bmatrix}1 & 2\\4 & 6\\9 & 12\end{bmatrix}$

Custom [2nd] [CUSTOM] key

Custom
    block
EndCustm

Sets up a toolbar that is activated when you press [2nd] [CUSTOM]. It is very similar to the **ToolBar** instruction (page 450) except that Title and Item statements cannot have labels.

block can be either a single statement or a series of statements separated with the ":" character.

Note: [2nd] [CUSTOM] acts as a toggle. The first instance invokes the menu, and the second instance removes the menu. The menu is removed also when you change applications.

Program listing:

```
:Test()
:Prgm
:Custom
:Title       "Lists"
:Item        "List1"
:Item        "Scores"
:Item        "L3"
:Title       "Fractions"
:Item        "f(x)"
:Item        "h(x)"
:Title       "Graph"
:EndCustm
:EndPrgm
```

Cycle — CATALOG

Cycle

Transfers program control immediately to the next iteration of the current loop (**For**, **While**, or **Loop**).

Cycle is not allowed outside the three looping structures (**For**, **While**, or **Loop**).

Program listing:

```
:© Sum the integers from 1 to 100
  skipping 50.
:0→temp
:For i,1,100,1
:If i=50
:Cycle
:temp+i→temp
:EndFor
:Disp temp
```

Contents of `temp` after execution: `5000`

CyclePic — CATALOG

CyclePic *picNameString*, *n* [, [*wait*] , [*cycles*], [*direction*]]

Displays all the PIC variables specified and at the specified interval. The user has optional control over the time between pictures, the number of times to cycle through the pictures, and the direction to go, circular or forward and backwards.

direction is 1 for circular or ⁻1 for forward and backwards. Default = 1.

1. Save three pics named pic1, pic2, and pic3.
2. Enter: `CyclePic "pic",3,.5,4,⁻1`
3. The three pictures (3) will be displayed automatically—one-half second (.5) between pictures, for four cycles (4), and forward and backwards (⁻1).

▶Cylind — MATH/Matrix/Vector ops menu

vector **▶Cylind**

Displays the row or column vector in cylindrical form [r∠θ, z].

vector must have exactly three elements. It can be either a row or a column.

`[2,2,3] ▶Cylind` ENTER $[2\cdot\sqrt{2}\ \angle\frac{\pi}{4}\ \ 3]$

cZeros() — MATH/Algebra/Complex menu

cZeros(*expression*, *var***)** ⇒ *list*

Returns a list of candidate real and non-real values of *var* that make *expression*=0. **cZeros()** does this by computing **exp▶list(cSolve(***expression*=0,*var***),***var***)**. Otherwise, **cZeros()** is similar to **zeros()**.

Note: See also **cSolve()** (page 385), **solve()** (page 442), and **zeros()** (page 453).

Display Digits mode in Fix 3:

`cZeros(x^5+4x^4+5x^3-6x-3,x)` ENTER
{⁻2.125 ⁻.612 .965 ⁻1.114 − 1.073·*i* ⁻1.114 + 1.073·*i*}

d() — [2nd] [*d*] key or MATH/Calculus menu

d(*expression1*, *var* [,*order*]) ⇒ *expression*
d(*list1*,*var* [,*order*]) ⇒ *list*
d(*matrix1*,*var* [,*order*]) ⇒ *matrix*

Returns the first derivative of *expression1* with respect to variable *var*. *expression1* can be a list or a matrix.

order, if included, must be an integer. If the order is less than zero, the result will be an anti-derivative.

***d*()** does not follow the normal evaluation mechanism of fully simplifying its arguments and then applying the function definition to these fully simplified arguments. Instead, ***d*()** performs the following steps:

1. Simplify the second argument only to the extent that it does not lead to a non-variable.
2. Simplify the first argument only to the extent that it does recall any stored value for the variable determined by step 1.
3. Determine the symbolic derivative of the result of step 2 with respect to the variable from step 1.
4. If the variable from step 1 has a stored value or a value specified by a "with" (|) operator, substitute that value into the result from step 3.

```
d(3x^3-x+7,x) [ENTER]                 9x²-1
d(3x^3-x+7,x,2) [ENTER]               18·x
d(f(x)*g(x),x) [ENTER]
```

$$\frac{d}{dx}(f(x))\cdot g(x) + \frac{d}{dx}(g(x))\cdot f(x)$$

```
d(sin(f(x)),x) [ENTER]
```

$$\cos(f(x))\frac{d}{dx}(f(x))$$

```
d(x^3,x)|x=5 [ENTER]                  75
d(d(x^2*y^3,x),y) [ENTER]             6·y²·x
d(x^2,x,⁻1) [ENTER]                   x³/3
d({x^2,x^3,x^4},x) [ENTER]
                          {2·x   3·x²   4·x³}
```

▶DD — MATH/Angle menu

number **▶DD** ⇒ *value*
list1 **▶DD** ⇒ *list*
matrix1 **▶DD** ⇒ *matrix*

Returns the decimal equivalent of the argument. The argument is a number, list, or matrix that is interpreted by the Mode setting in radians or degrees.

Note: ▶DD can also accept input in radians.

In Degree angle mode:

```
1.5°▶DD [ENTER]                       1.5°
45°22'14.3"▶DD [ENTER]                45.370...°
{45°22'14.3",60°0'0"}▶DD [ENTER]
                          {45.370...   60}°
```

In Radian angle mode:

```
1.5▶DD [ENTER]                        85.9°
```

Define CATALOG

Define *funcName*(*arg1Name*, *arg2Name*, ...) **=** *expression*

Creates *funcName* as a user-defined function. You then can use *funcName*(), just as you use built-in functions. The function evaluates *expression* using the supplied arguments and returns the result.

funcName cannot be the name of a system variable or built-in function.

The argument names are placeholders; you should not use those same names as arguments when you use the function.

Note: This form of **Define** is equivalent to executing the expression: *expression*→ *funcName*(*arg1Name*,*arg2Name*). This command also can be used to define simple variables ; for example, Define a=3.

```
Define g(xx,yy)=2xx-3yy [ENTER]      Done
g(1,2) [ENTER]                        ⁻4
1→a:2→b:g(a,b) [ENTER]                ⁻4

Define h(xx)=when(xx<2,2xx-3,
⁻2xx+3) [ENTER]                      Done

h(⁻3) [ENTER]                         ⁻9
h(4) [ENTER]                          ⁻5

Define eigenvl(aa)=
  cZeros(det(identity(dim(aa)
  [1])-x*aa),x) [ENTER]              Done
eigenvl([⁻1,2;4,3]) [ENTER]
```

$$\left\{\frac{2\cdot\sqrt{3}-1}{11} \quad \frac{^-(2\cdot\sqrt{3}+1)}{11}\right\}$$

Define *funcName*(*arg1Name*, *arg2Name*, ...) **= Func**
block
EndFunc

Is identical to the previous form of **Define**, except that in this form, the user-defined function *funcName*() can execute a block of multiple statements.

block can be either a single statement or a series of statements separated with the ":" character. *block* also can include expressions and instructions (such as **If**, **Then**, **Else**, and **For**). This allows the function *funcName*() to use the **Return** instruction to return a specific result.

Note: It is usually easier to author and edit this form of Function in the program editor rather than on the entry line. (See Chapter 17: Programming.)

```
Define g(xx,yy)=func:If xx>yy Then
:Return xx:Else:Return yy:EndIf
:EndFunc [ENTER]                     Done

g(3,⁻7) [ENTER]                         3
```

Define *progName*(*arg1Name*, *arg2Name*, ...) **= Prgm**
block
EndPrgm

Creates *progName* as a program or subprogram, but cannot return a result using **Return**. Can execute a block of multiple statements.

block can be either a single statement or a series of statements separated with the ":" character. *block* also can include expressions and instructions (such as **If**, **Then**, **Else**, and **For**) without restrictions.

Note: It is usually easier to author and edit a program block in the Program Editor rather than on the entry line. (See Chapter 17: Programming.)

```
Define listinpt()=prgm:Local
  n,i,str1,num:InputStr "Enter
  name of list",str1:Input "No. of
  elements",n:For i,1,n,1:Input
  "element "&string(i),num:
  num→#str1[i]:EndFor:EndPrgm [ENTER]
                                     Done

listinpt() [ENTER]      Enter name of list
```

DelFold CATALOG

DelFold *folderName1*[, *folderName2*] [, *folderName3*] ...

Deletes user-defined folders with the names *folderName1*, *folderName2*, etc. An error message is displayed if the folders contain any variables.

Note: You cannot delete the main folder.

```
NewFold games [ENTER]                Done
```
(creates the folder games)

```
DelFold games [ENTER]                Done
```
(deletes the folder games)

DelVar CATALOG

DelVar *var1*[, *var2*] [, *var3*] ...

Deletes the specified variables from memory.

```
2→a [ENTER]                          2
(a+2)^2 [ENTER]                      16
DelVar a [ENTER]                     Done
(a+2)^2 [ENTER]                      (a + 2)²
```

det() MATH/Matrix menu

det(*squareMatrix***)** ⇒ *expression*

Returns the determinant of *squareMatrix*.

squareMatrix must be square.

```
det([a,b;c,d]) [ENTER]               a•d - b•c
det([1,2;3,4]) [ENTER]               ⁻2
det(identity(3) - x*[1,⁻2,3;
⁻2,4,1;⁻6,⁻2,7]) [ENTER]
                     ⁻(98•x³ - 55•x² + 12•x - 1)
```

diag() MATH/Matrix menu

diag(*list***)** ⇒ *matrix*
diag(*rowMatrix***)** ⇒ *matrix*
diag(*columnMatrix***)** ⇒ *matrix*

Returns a matrix with the values in the argument list or matrix in its main diagonal.

```
diag({2,4,6}) [ENTER]                [2 0 0]
                                     [0 4 0]
                                     [0 0 6]
```

---

diag(*squareMatrix***)** ⇒ *rowMatrix*

Returns a row matrix containing the elements from the main diagonal of *squareMatrix*.

squareMatrix must be square.

```
[4,6,8;1,2,3;5,7,9] [ENTER]          [4 6 8]
                                     [1 2 3]
                                     [5 7 9]
diag(ans(1)) [ENTER]                 [4 2 9]
```

Dialog CATALOG

Dialog
  block
EndDlog

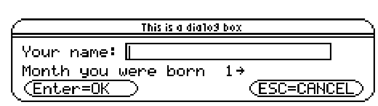
Generates a dialog box when the program is executed.

block can be either a single statement or a series of statements separated with the ":" character. Valid *block* options in the [F3] I/O, 1:Dialog menu item in the Program Editor are 1:Text, 2:Request, 4:DropDown, and 7:Title.

The variables in a dialog box can be given values that will be displayed as the default (or initial) value. If [ENTER] is pressed, the variables are updated from the dialog box and variable ok is set to 1. If [ESC] is pressed, its variables are not updated, and system variable ok is set to zero.

Program listing:

```
:Dlogtest()
:Prgm
:Dialog
:Title     "This is a dialog box"
:Request   "Your name",Str1
:Dropdown  "Month you were born",
  seq(string(i),i,1,12),Var1
:EndDlog
:EndPrgm
```



dim() MATH/Matrix/Dimensions menu

dim(*list***)** ⇒ *integer*

Returns the dimension of *list*.

```
dim({0,1,2}) ENTER                                   3
```

---

dim(*matrix***)** ⇒ *list*

Returns the dimensions of *matrix* as a two-element list {rows, columns}.

```
dim([1,‾1,2;‾2,3,5]) ENTER                      {2 3}
```

---

dim(*string***)** ⇒ *integer*

Returns the number of characters contained in character string *string*.

```
dim("Hello") ENTER                                   5
dim("Hello"&" there") ENTER                         11
```

Disp CATALOG

Disp

Displays the current contents of the Program I/O screen.

Disp [*exprOrString1*] [, *exprOrString2*] ...

Displays each expression or character string on a separate line of the Program I/O screen.

If Pretty Print = ON, expressions are displayed in pretty print.

```
Disp "Hello" ENTER                               Hello
Disp cos(2.3) ENTER                             ‾.666…
{1,2,3,4}→L1 ENTER
Disp L1 ENTER                              {1  2  3  4}
```

DispG CATALOG

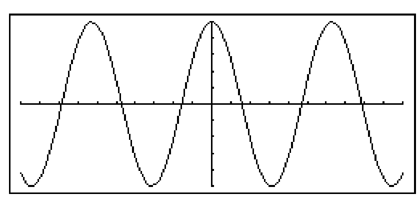
DispG

Displays the current contents of the Graph screen.

In function graphing mode:

Program segment:

```
   :
:5*cos(x)→y1(x)
:‾10→xmin
:10→xmax
:‾5→ymin
:5→ymax
:DispG
   :
```



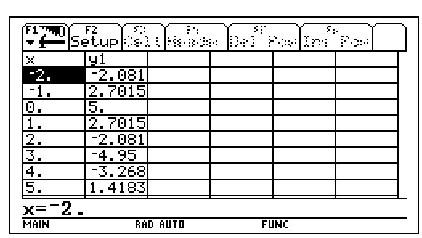
DispTbl CATALOG

DispTbl

Displays the current contents of the Table screen.

Note: The cursor pad is active for scrolling. Press ESC or ENTER to resume execution if in a program.

```
5*cos(x)→y1(x) ENTER
DispTbl ENTER
```



▶DMS — MATH/Angle menu

expression **▶DMS**
list **▶DMS**
matrix **▶DMS**

Interprets the argument as an angle and displays the equivalent DMS (*DDDDDD°MM′SS.ss″*) number. See °, ', " on page 467 for DMS (degree, minutes, seconds) format.

Note: **▶DMS** will convert from radians to degrees when used in radian mode. If the input is followed by a degree symbol (°), no conversion will occur. You can use **▶DMS** only at the end of an entry line.

In Degree angle mode:

```
45.371 ▶DMS [ENTER]                45°22'15.6"

{45.371,60} ▶DMS [ENTER]
                              {45°22'15.6"  60°}
```

dotP() — MATH/Matrix/Vector ops menu

dotP(*list1***,** *list2***)** ⇒ *expression*

Returns the "dot" product of two lists.

```
dotP({a,b,c},{d,e,f}) [ENTER]
                                   a·d + b·e + c·f
dotP({1,2},{5,6}) [ENTER]                       17
```

---

dotP(*vector1***,** *vector2***)** ⇒ *expression*

Returns the "dot" product of two vectors.

Both must be row vectors, or both must be column vectors.

```
dotP([a,b,c],[d,e,f]) [ENTER]
                                   a·d + b·e + c·f
dotP([1,2,3],[4,5,6]) [ENTER]                   32
```

DrawFunc — CATALOG

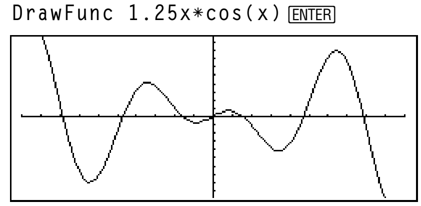
DrawFunc *expression*

Draws *expression* as a function, using x as the independent variable.

Note: Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

```
DrawFunc 1.25x*cos(x) [ENTER]
```



DrawInv — CATALOG

DrawInv *expression*

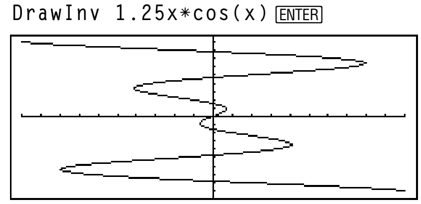
Draws the inverse of *expression* by plotting x values on the y axis and y values on the x axis.

x is the independent variable.

Note: Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

```
DrawInv 1.25x*cos(x) [ENTER]
```



DrawParm CATALOG

DrawParm *expression1*, *expression2*
[, *tmin*] [, *tmax*] [, *tstep*]

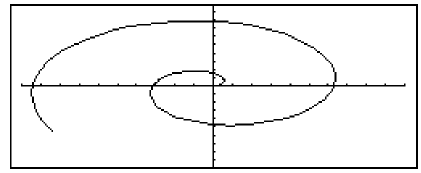
Draws the parametric equations *expression1* and *expression2*, using t as the independent variable.

Defaults for *tmin*, *tmax*, and *tstep* are the current settings for the Window variables tmin, tmax, and tstep. Specifying values does not alter the window settings. If the current graphing mode is not parametric, these three arguments are required.

Note: Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

```
DrawParm t*cos(t),t*sin(t),0,10,.1
```
[ENTER]



DrawPol CATALOG

DrawPol *expression*[, θ*min*] [, θ*max*] [, θ*step*]

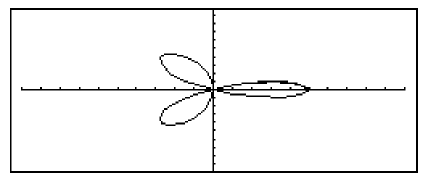
Draws the polar graph of *expression*, using θ as the independent variable.

Defaults for θ*min*, θ*max*, and θ*step* are the current settings for the Window variables θmin, θmax, and θstep. Specifying values does not alter the window settings. If the current graphing mode is not polar, these three arguments are required.

Note: Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

```
DrawPol 5*cos(3*θ),0,3.5,.1
```
[ENTER]



DrawSlp CATALOG

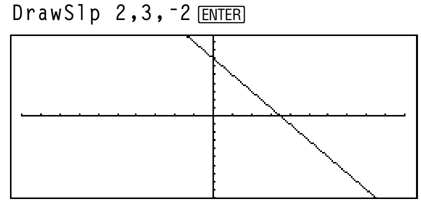
DrawSlp *x1*, *y1*, *slope*

Displays the graph and draws a line using the formula y−y1=slope•(x−x1).

Note: Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

```
DrawSlp 2,3,-2
```
[ENTER]



DropDown CATALOG

DropDown *titleString*, **{***item1String*, *item2String*, **...},** *varName*

Displays a drop-down menu with the name *titleString* and containing the items **1:***item1String*, **2:***item2String*, and so forth. **DropDown** must be within a **Dialog...EndDlog** block.

If *varName* already exists and has a value within the range of items, the referenced item is displayed as the default selection. Otherwise, the menu's first item is the default selection.

When you select an item from the menu, the corresponding number of the item is stored in the variable *varName*. (If necessary, **DropDown** creates *varName*.)

See **Dialog** program listing example on page 390.

E [2nd][EE] key

*mantissa***E***exponent*

Enters a number in scientific notation. The number is interpreted as *mantissa* × $10^{exponent}$.

Hint: If you want to enter a power of 10 without causing a decimal value result, use 10^*integer*.

```
2.3E4 [ENTER]                 23000.
2.3E9+4.1E15 [ENTER]          4.1E15
3*10^4 [ENTER]                30000
```

e^() [2nd] [eˣ] key

e*^(expression1***)** ⇒ *expression*

Returns *e* raised to the *expression1* power.

Note: Pressing [2nd] [eˣ] to display *e*^(is different from accessing the character e from the QWERTY keyboard.

```
e^(1) [ENTER]                 e
e^(1.) [ENTER]                2.718...
```

---

e* ^(list1***)** ⇒ *list*

Returns *e* raised to the power of each element in *list1*.

```
e^({1,1.,0,.5}) [ENTER]
        {e   2.718...   1   1.648...}
```

Else

See **If**, page 407.

ElseIf

CATALOG See also **If**, page 407.

If *Boolean expression1* **Then**
block1
ElseIf *Boolean expression2* **Then**
block2
⋮
ElseIf *Boolean expressionN* **Then**
blockN
EndIf
⋮

ElseIf can be used as a program instruction for program branching.

Program segment:

```
   ⋮
:If choice=1 Then
:  Goto option1
:  ElseIf choice=2 Then
:  Goto option2
:  ElseIf choice=3 Then
:  Goto option3
:  ElseIf choice=4 Then
:  Disp "Exiting Program"
:  Return
:EndIf
   ⋮
```

EndCustm

See **Custom**, page 386.

EndDlog

See **Dialog**, page 390.

EndFor

See **For**, page 402.

EndFunc

See **Func**, page 403.

EndIf

See **If**, page 407.

EndLoop

See **Loop**, page 415.

EndPrgm

See **Prgm**, page 426.

EndTBar

See **ToolBar**, page 450.

EndTry

See **Try**, page 450.

EndWhile

See **While**, page 452.

entry() CATALOG

entry() ⇒ *expression*
entry(*integer***)** ⇒ *expression*

Returns a previous entry-line entry from the Home screen history area.

integer, if included, specifies which entry expression in the history area. The default is 1, the most recently evaluated entry. Valid range is from 1 to 99 and cannot be an expression.

Note: If the last entry is still highlighted on the Home screen, pressing [ENTER] is equivalent to executing **entry(1)**.

On the Home screen:

`1+1/x` [ENTER] $\frac{1}{x}+1$

`1+1/entry(1)` [ENTER] $\frac{-1}{x+1}+2$

[ENTER] $\frac{1}{2\cdot(2\cdot x+1)}+3/2$

[ENTER] $\frac{-1}{3\cdot(3\cdot x+2)}+5/3$

`entry(4)` [ENTER] $\frac{1}{x}+1$

exact() MATH/Number menu

exact(*expression1* [, *tol*]**)** ⇒ *expression*
exact(*list1* [, *tol*]**)** ⇒ *list*
exact(*matrix1* [, *tol*]**)** ⇒ *matrix*

Uses Exact mode arithmetic regardless of the Exact/Approx mode setting to return, when possible, the rational-number equivalent of the argument.

tol specifies the tolerance for the conversion; the default is 0 (zero).

`exact(.25)` [ENTER] 1/4

`exact(.333333)` [ENTER] $\frac{333333}{1000000}$

`exact(.33333,.001)` 1/3

`exact(3.5x+y)` [ENTER] $\frac{7\cdot x}{2}+y$

`exact({.2,.33,4.125})` [ENTER]
$\{1/5 \;\; \frac{33}{100} \;\; 33/8\}$

Exit CATALOG

Exit

Exits the current **For**, **While**, or **Loop** block.

Exit is not allowed outside the three looping structures (**For**, **While**, or **Loop**).

Program listing:

```
:0→temp
:For i,1,100,1
:  temp+i→temp
:  If temp>20
:  Exit
:EndFor
:Disp temp
```

Contents of `temp` after execution: 21

exp▶list() CATALOG

exp▶list(*expression*,*var***)** ⇒ *list*

Examines *expression* for equations that are separated by the word "or," and returns a list containing the right-hand sides of the equations of the form *var=expression*. This gives you an easy way to extract some solution values embedded in the results of the **solve()**, **cSolve()**, **fMin()**, and **fMax()** functions.

Note: exp▶list() is not necessary with the **zeros** and **cZeros()** functions because they return a list of solution values directly.

`solve(x^2-x-2=0,x)` [ENTER] x=2 or x=-1

`exp▶list(solve(x^2-x-2=0,x),x)` [ENTER]
{-1 2}

expand() MATH/Algebra menu

expand(*expression1* [**,** *var*]**)** ⇒ *expression*
expand(*list1* [,*var*]**)** ⇒ *list*
expand(*matrix1* [,*var*]**)** ⇒ *matrix*

expand(*expression1***)** returns *expression1* expanded with respect to all its variables. The expansion is polynomial expansion for polynomials and partial fraction expansion for rational expressions.

The goal of **expand()** is to transform *expression1* into a sum and/or difference of simple terms. In contrast, the goal of **factor()** is to transform *expression1* into a product and/or quotient of simple factors.

```
expand((x+y+1)^2) [ENTER]
```
$x^2 + 2\cdot x\cdot y + 2\cdot x + y^2 + 2\cdot y + 1$

```
expand((x^2-x+y^2-y)/(x^2*y^2-x^2
  *y-x*y^2+x*y)) [ENTER]
```

$$\blacksquare\ \text{expand}\left(\frac{x^2 - x + y^2 - y}{x^2\cdot y^2 - x^2\cdot y - x\cdot y^2 + x\cdot y}\right) \qquad \frac{1}{x-1} - \frac{1}{x} + \frac{1}{y-1} - \frac{1}{y}$$

---

expand(*expression1*,*var***)** returns *expression* expanded with respect to *var*. Similar powers of *var* are collected. The terms and their factors are sorted with *var* as the main variable. There might be some incidental factoring or expansion of the collected coefficients. Compared to omitting *var*, this often saves time, memory, and screen space, while making the expression more comprehensible.

```
expand((x+y+1)^2,y) [ENTER]
```
$y^2 + 2\cdot y\cdot (x + 1) + (x + 1)^2$

```
expand((x+y+1)^2,x) [ENTER]
```
$x^2 + 2\cdot x\cdot (y + 1) + (y + 1)^2$

```
expand((x^2-x+y^2-y)/(x^2*y^2-x^2
  *y-x*y^2+x*y),y) [ENTER]
```

$$\blacksquare\ \text{expand}\left(\frac{x^2 - x + y^2 - y}{x^2\cdot y^2 - x^2\cdot y - x\cdot y^2 + x\cdot y}, y\right) \qquad \frac{1}{y-1} - \frac{1}{y} + \frac{1}{x\cdot(x-1)}$$

```
expand(ans(1),x) [ENTER]
```

$$\blacksquare\ \text{expand}\left(\frac{1}{y-1} - \frac{1}{y} + \frac{1}{x\cdot(x-1)}, x\right) \qquad \frac{1}{x-1} - \frac{1}{x} + \frac{1}{y\cdot(y-1)}$$

Even when there is only one variable, using *var* might make the denominator factorization used for partial fraction expansion more complete.

```
expand((x^3+x^2-2)/(x^2-2)) [ENTER]
```
$$\frac{2\cdot x}{x^2-2} + x+1$$

```
expand(ans(1),x) [ENTER]
```
$$\frac{1}{x-\sqrt{2}} + \frac{1}{x+\sqrt{2}} + x+1$$

Hint: For rational expressions, **propFrac()** (page 427) is a faster but less extreme alternative to **expand()**.

Note: See also **comDenom()** (page 383) for an expanded numerator over an expanded denominator.

expand(*expression1*,[*var*]**)** also distributes logarithms and fractional powers regardless of *var*. For increased distribution of logarithms and fractional powers, inequality constraints might be necessary to guarantee that some factors are nonnegative.

```
ln(2x*y)+√(2x*y) [ENTER]
```
$\ln(2\cdot x\cdot y) + \sqrt{(2\cdot x\cdot y)}$

```
expand(ans(1)) [ENTER]
```
$\ln(x\cdot y) + \sqrt{2}\cdot\sqrt{(x\cdot y)} + \ln(2)$

```
expand(ans(1))|y>=0 [ENTER]
```
$\ln(x) + \sqrt{2}\cdot\sqrt{x}\cdot\sqrt{y} + \ln(y) + \ln(2)$

expand(*expression1*, [*var*]**)** also distributes absolute values, **sign()**, and exponentials, regardless of *var*.

```
sign(x*y)+abs(x*y)+ e^(2x+y) [ENTER]
```
$e^{2x+y} + \text{sign}(x\cdot y) + |x\cdot y|$

```
expand(ans(1)) [ENTER]
```
$(e^x)^2\cdot e^y + \text{sign}(x)\cdot\text{sign}(y) + |x|\cdot|y|$

Note: See also **tExpand()** (page 449) for trigonometric angle-sum and multiple-angle expansion.

expr() MATH/String menu

expr(*string***)** ⇒ *expression*

Returns the character string contained in *string* as an expression and immediately executes it.

```
expr("1+2+x^2+x") [ENTER]             x^2 + x + 3
expr("expand((1+x)^2)") [ENTER]
                                      x^2 + 2·x + 1
"Define cube(xx)=xx^3"→funcstr [ENTER]
                          "Define cube(xx)=xx^3"
expr(funcstr) [ENTER]                 Done
cube(2) [ENTER]                       8
```

ExpReg MATH/Statistics/Regressions menu

ExpReg *list1*, *list2* [, [*list3*] [, *list4*, *list5*]]

Calculates the exponential regression and updates all the system statistics variables.

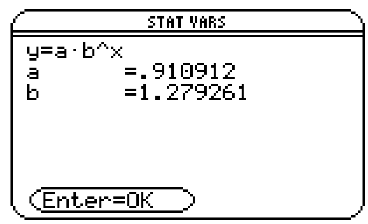
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

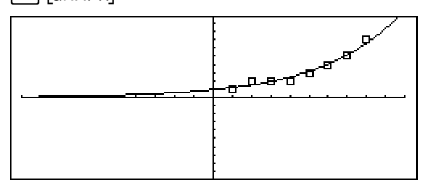
In function graphing mode:

```
{1,2,3,4,5,6,7,8}→L1 [ENTER]     {1 2 ...}
{1,2,2,2,3,4,5,7}→L2 [ENTER]     {1 2 ...}
ExpReg L1,L2 [ENTER]             Done
ShowStat [ENTER]
```




```
[ENTER]
Regeq(x)→y1(x) [ENTER]           Done
NewPlot 1,1,L1,L2 [ENTER]        Done
```

[♦] [GRAPH]



factor() MATH/Algebra menu

factor(*expression1*[, *var*]**)** ⇒ *expression*
factor(*list1*[,*var*]**)** ⇒ *list*
factor(*matrix1*[,*var*]**)** ⇒ *matrix*

factor(*expression1*) returns *expression1* factored with respect to all of its variables over a common denominator.

expression1 is factored as much as possible toward linear rational factors without introducing new non-real subexpressions. This alternative is appropriate if you want factorization with respect to more than one variable.

```
factor(a^3*x^2-a*x^2-a^3+a) [ENTER]
        a·(a − 1)·(a + 1)·(x − 1)·(x + 1)
```

factor(x^2+1) [ENTER] $x^2 + 1$

factor(x^2-4) [ENTER] $(x - 2)\cdot(x + 2)$

factor(x^2-3) [ENTER] $x^2 - 3$

factor(x^2-a) [ENTER] $x^2 - a$

---

factor(*expression1*,*var*) returns *expression1* factored with respect to variable *var*.

expression1 is factored as much as possible toward real factors that are linear in *var*, even if it introduces irrational constants or subexpressions that are irrational in other variables.

The factors and their terms are sorted with *var* as the main variable. Similar powers of *var* are collected in each factor. Include *var* if factorization is needed with respect to only that variable and you are willing to accept irrational expressions in any other variables to increase factorization with respect to *var*. There might be some incidental factoring with respect to other variables.

factor(a^3*x^2-a*x^2-a^3+a,x) [ENTER]
$a\cdot(a^2 - 1)\cdot(x - 1)\cdot(x + 1)$

factor(x^2-3,x) [ENTER] $(x + \sqrt{3})\cdot(x - \sqrt{3})$

factor(x^2-a,x) [ENTER] $(x + \sqrt{a})\cdot(x - \sqrt{a})$

For the AUTO setting of the Exact/Approx mode, including *var* permits approximation with floating-point coefficients where irrational coefficients cannot be explicitly expressed concisely in terms of the built-in functions. Even when there is only one variable, including *var* might yield more complete factorization.

factor(x^5+4x^4+5x^3-6x-3) [ENTER]
$x^5 + 4\cdot x^4 + 5\cdot x^3 - 6\cdot x - 3$

factor(ans(1),x) [ENTER]
$(x - .965)\cdot(x + .612)\cdot(x + 2.13)\cdot(x^2 + 2.23\cdot x + 2.39)$

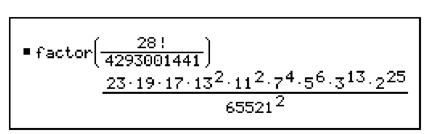
Note: See also **comDenom()** (page 383)for a fast way to achieve partial factoring when **factor()** is not fast enough or if it exhausts memory.

Note: See also **cFactor()** (page 380) for factoring all the way to complex coefficients in pursuit of linear factors.

---

factor(*rational_number*) returns the rational number factored into primes and a residual having prime factors that exceed 65521.

factor(28!/4293001441) [ENTER]



Fill — MATH/Matrix menu

Fill *expression*, *matrixVar* ⇒ *matrix*

Replaces each element in variable *matrixVar* with *expression*.

matrixVar must already exist.

```
[1,2;3,4]→amatrx ENTER        [1 2; 3 4]
Fill 1.01,amatrx ENTER        Done
amatrx ENTER                  [1.01 1.01; 1.01 1.01]
```

Fill *expression*, *listVar* ⇒ *list*

Replaces each element in variable *listVar* with *expression*.

listVar must already exist.

```
{1,2,3,4,5}→alist ENTER    {1 2 3 4 5}
Fill 1.01,alist ENTER      Done
alist ENTER
           {1.01 1.01 1.01 1.01 1.01}
```

floor() — MATH/Number menu

floor(*expression***)** ⇒ *integer*

Returns the greatest integer that is ≤ the argument. This function is identical to **int()**.

The argument can be a real or a complex number.

```
floor(⁻2.14) ENTER          ⁻3.
```

---

floor(*list1***)** ⇒ *list*
floor(*matrix1***)** ⇒ *matrix*

Returns a list or matrix of the floor of each element.

Note: See also **ceiling()** (page 379) and **int()** (page 409).

```
floor({3/2,0,⁻5.3}) ENTER    {1 0 ⁻6.}
floor([1.2,3.4;2.5,4.8]) ENTER
                             [1. 3.; 2. 4.]
```

fMax() — MATH/Calculus menu

fMax(*expression*, *var***)** ⇒ *Boolean expression*

Returns a Boolean expression specifying candidate values of *var* that maximize *expression* or locate its least upper bound.

Use the "|" operator to restrict the solution interval and/or specify the sign of other undefined variables.

For the APPROX setting of the Exact/Approx mode, **fMax()** iteratively searches for one approximate local maximum. This is often faster, particularly if you use the "|" operator to constrain the search to a relatively small interval that contains exactly one local maximum.

Note: See also **fMin()** (page 401) and **max()** (page 415).

```
fMax(1-(x-a)^2-(x-b)^2,x) ENTER
                               x = (a+b)/2
fMax(.5x^3-x-2,x) ENTER        x = ∞
fMax(.5x^3-x-2,x)|x≤1 ENTER    x = ⁻.816...
fMax(a*x^2,x) ENTER
        x = ∞ or x = ⁻∞ or x = 0 or a = 0
fMax(a*x^2,x)|a<0 ENTER        x = 0
```

fMin() MATH/Calculus menu

fMin(*expression*, *var***)** ⇒ *Boolean expression*

Returns a Boolean expression specifying candidate values of *var* that minimize *expression* or locate its greatest lower bound.

Use the "|" operator to restrict the solution interval and/or specify the sign of other undefined variables.

For the APPROX setting of the Exact/Approx mode, **fMin()** iteratively searches for one approximate local minimum. This is often faster, particularly if you use the "|" operator to constrain the search to a relatively small interval that contains exactly one local minimum.

Note: See also **fMax()** (page 400) and **min()** (page 417).

```
fMin(1-(x-a)^2-(x-b)^2,x) [ENTER]
                                x = ∞ or x = ⁻∞

fMin(.5x^3-x-2,x)|x≥1 [ENTER]          x = 1

fMin(a*x^2,x) [ENTER]
              x = ∞ or x = ⁻∞ or x = 0 or a = 0

fMin(a*x^2,x)|a>0 and x>1 [ENTER]     x = 1.

fMin(a*x^2,x)|a>0 [ENTER]             x = 0
```

FnOff CATALOG

FnOff

Deselects all Y= functions for the current graphing mode.

In split-screen, two-graph mode, **FnOff** only applies to the active graph.

FnOff [1] [, 2] ... [,99]

Deselects the specified Y= functions for the current graphing mode.

In function graphing mode:
FnOff 1,3 [ENTER] deselects y1(x) and y3(x).

In parametric graphing mode:
FnOff 1,3 [ENTER] deselects xt1(t), yt1(t), xt3(t), and yt3(t).

FnOn CATALOG

FnOn

Selects all Y= functions that are defined for the current graphing mode.

In split-screen, two-graph mode, **FnOn** only applies to the active graph.

FnOn [1] [, 2] ... [,99]

Selects the specified Y= functions for the current graphing mode.

Note: In 3D graphing mode, only one function at a time can be selected. FnOn 2 selects z2(x,y) and deselects any previously selected function. In the other graph modes, previously selected functions are not affected.

For CATALOG

For *var*, *low*, *high* [, *step*]
block
EndFor

Executes the statements in *block* iteratively for each value of *var*, from *low* to *high*, in increments of *step*.

var must not be a system variable.

step can be positive or negative. The default value is 1.

block can be either a single statement or a series of statements separated with the ":" character.

Program segment:

```
   :
:0→tempsum : 1→step
:For i,1,100,step
:  tempsum+i→tempsum
:EndFor
:Disp tempsum
   :
```

Contents of `tempsum` after execution: `5050`

Contents of `tempsum` when `step` is changed to 2: `2500`

format() MATH/String menu

format(*expression*[, *formatString*]**)** ⇒ *string*

Returns *expression* as a character string based on the format template.

expression must simplify to a number. *formatString* is a string and must be in the form: "F[*n*]", "S[*n*]", "E[*n*]", "G[*n*][*c*]", where [ ] indicate optional portions.

F[*n*]: Fixed format. *n* is the number of digits to display after the decimal point.

S[*n*]: Scientific format. *n* is the number of digits to display after the decimal point.

E[*n*]: Engineering format. *n* is the number of digits after the first significant digit. The exponent is adjusted to a multiple of three, and the decimal point is moved to the right by zero, one, or two digits.

G[*n*][*c*]: Same as fixed format but also separates digits to the left of the radix into groups of three. *c* specifies the group separator character and defaults to a comma. If *c* is a period, the radix will be shown as a comma.

[R*c*]: Any of the above specifiers may be suffixed with the R*c* radix flag, where *c* is a single character that specifies what to substitute for the radix point.

```
format(1.234567,"f3") [ENTER]       "1.235"
format(1.234567,"s2") [ENTER]      "1.23E0"
format(1.234567,"e3") [ENTER]     "1.235E0"
format(1.234567,"g3") [ENTER]       "1.235"
format(1234.567, "g3") [ENTER]
                                "1,234.567"
format(1.234567,"g3,r:") [ENTER]
                                    "1:235"
```

fpart() MATH/Number menu

fpart(*expression1***)** ⇒ *expression*
fpart(*list1***)** ⇒ *list*
fpart(*matrix1***)** ⇒ *matrix*

Returns the fractional part of the argument.

For a list or matrix, returns the fractional parts of the elements.

The argument can be a real or a complex number.

```
fpart(⁻1.234) [ENTER]                 ⁻.234
fpart({1, ⁻2.3, 7.003}) [ENTER]
                            {0  ⁻.3  .003}
```

Func — CATALOG

Func
  block
EndFunc

Required as the first statement in a multi-statement function definition.

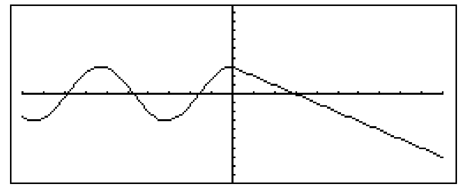
block can be either a single statement or a series of statements separated with the ":" character.

Note: when() (page 452) also can be used to define and graph piecewise-defined functions.

In function graphing mode, define a piecewise function:

```
Define g(xx)=Func:If xx<0 Then
  :Return 3*cos(xx):Else:Return
  3-xx:EndIf:EndFunc [ENTER]        Done
```

```
Graph g(x) [ENTER]
```



gcd() — MATH/Number menu

gcd(*number1*, *number2***)** ⇒ *expression*

Returns the greatest common divisor of the two arguments. The **gcd** of two fractions is the **gcd** of their numerators divided by the **lcm** of their denominators.

The **gcd** of fractional floating-point numbers is 1.0.

```
gcd(18,33) [ENTER]                        3
```

---

gcd(*list1*, *list2***)** ⇒ *list*

Returns the greatest common divisors of the corresponding elements in *list1* and *list2*.

```
gcd({12,14,16},{9,7,5}) [ENTER] {3 7 1}
```

---

gcd(*matrix1*, *matrix2***)** ⇒ *matrix*

Returns the greatest common divisors of the corresponding elements in *matrix1* and *matrix2*.

```
gcd([2,4;6,8],[4,8;12,16]) [ENTER]
                                  [2  4]
                                  [6  8]
```

Get — CATALOG

Get *var*

Retrieves a CBL 2/CBL (Calculator-Based Laboratory) or CBR (Calculator-Based Ranger) value from the link port and stores it in variable *var*.

Program segment:

```
   ⋮
:Send {3,1,-1,0}
:For i,1,99
:  Get data[i]
:  PtOn i,data[i]
:EndFor
   ⋮
```

GetCalc — CATALOG

GetCalc *var*

Retrieves a value from the link port and stores it in variable *var*. This is for unit-to-unit linking.

Note: To get a variable to the link port from another unit, use [2nd] [VAR-LINK] on the other unit to select and send a variable, or do a **SendCalc** on the other unit.

Program segment:

```
   ⋮
:Disp "Press Enter when ready"
:Pause
:GetCalc L1
:Disp "List L1 received"
   ⋮
```

getDenom() MATH/Algebra/Extract menu

getDenom(*expression1***)** ⇒ *expression*

Transforms *expression1* into one having a reduced common denominator, and then returns its denominator.

```
getDenom((x+2)/(y-3)) [ENTER]        y - 3
getDenom(2/7) [ENTER]                    7
getDenom(1/x+(y^2+y)/y^2) [ENTER]    x·y
```

getFold() CATALOG

getFold() ⇒ *nameString*

Returns the name of the current folder as a string.

```
getFold() [ENTER]                 "main"
getFold()→oldfoldr [ENTER]        "main"
oldfoldr [ENTER]                  "main"
```

getKey() CATALOG

getKey() ⇒ *integer*

Returns the key code of the key pressed. Returns 0 if no key is pressed.

The prefix keys (shift [↑], second function [2nd], option [♦], and drag [✋]) are not recognized by themselves; however, they modify the keycodes of the key that follows them. For example: [♦]K ≠ K ≠ [2nd]K.

For a listing of key codes, see Appendix B.

Program listing:

```
:Disp
:Loop
:  getKey()→key
:  while key=0
:    getKey()→key
:  EndWhile
:  Disp key
:  If key = ord("a")
:  Stop
:EndLoop
```

getMode() CATALOG

getMode(*modeNameString***)** ⇒ *string*
getMode("ALL") ⇒ *ListStringPairs*

If the argument is a specific mode name, returns a string containing the current setting for that mode.

If the argument is **"ALL"**, returns a list of string pairs containing the settings of all the modes. If you want to restore the mode settings later, you must store the **getMode("ALL")** result in a variable, and then use **setMode** to restore the modes.

For a listing of mode names and possible settings, see **setMode** on page 438.

```
getMode("angle") [ENTER]        "RADIAN"
getMode("graph") [ENTER]      "FUNCTION"
getMode("all") [ENTER]
 {"Graph" "FUNCTION" "Display Digits"
           "FLOAT 6" "Angle" "RADIAN"
        "Exponential Format" "NORMAL"
      "Complex Format" "REAL" "Vector
 Format" "RECTANGULAR" "Pretty Print"
  "ON" "Split Screen" "FULL" "Split 1
    App" "Home" "Split 2 App" "Graph"
     "Number of Graphs" "1" "Graph 2"
      "FUNCTION" "Split Screen Ratio"
         "1:1" "Exact/Approx" "AUTO"}
```

Note: Your screen may display different mode settings.

getNum() MATH/Algebra/Extract menu

getNum(*expression1***)** ⇒ *expression*

Transforms *expression1* into one having a reduced common denominator, and then returns its numerator.

```
getNum((x+2)/(y-3)) [ENTER]        x + 2
getNum(2/7) [ENTER]                    2
getNum(1/x+1/y) [ENTER]            x + y
```

getType() CATALOG

getType(*var***)** ⇒ *string*

Returns a string indicating the TI-92 data type of variable *var*.

If *var* has not been defined, returns the string "NONE."

```
{1,2,3}→temp [ENTER]                  {1 2 3}
getType(temp) [ENTER]                  "LIST"

2+3i→temp [ENTER]                       2 + 3i
getType(temp) [ENTER]                  "EXPR"

DelVar temp [ENTER]                      Done
getType(temp) [ENTER]                  "NONE"
```

Data Type	Variable Contents
"DATA"	Data type
"EXPR"	Expression (includes complex/arbitrary/undefined, ∞, ⁻∞, TRUE, FALSE, pi, *e*)
"FIG"	Geometry figure
"FUNC"	Function
"GDB"	Graph Data Base
"LIST"	List
"MAC"	Geometry macro
"MAT"	Matrix
"NONE"	Variable does not exist
"NUM"	Real number
"PIC"	Picture
"PRGM"	Program
"STR"	String
"TEXT"	Text type
"VAR"	Name of another variable

Goto CATALOG

Goto *labelName*

Transfers program control to the label *labelName*.

labelName must be defined in the same program using a **Lbl** instruction. (See page 410.)

Program segment:

```
   :
:0→temp
:1→i
:Lbl TOP
:  temp+i→temp
:  If i<10 Then
:   i+1→i
:   Goto TOP
:  EndIf
:Disp temp
   :
```

Graph — CATALOG

Graph *expression1*[, *expression2*] [, *var1*] [, *var2*]

The Smart Graph feature graphs the requested expressions/ functions using the current graphing mode.

Expressions entered using the **Graph** or **Table** (page 447) commands are assigned increasing function numbers starting with 1. They can be modified or individually deleted using the edit functions available when the table is displayed by pressing [F4] Header. The currently selected Y= functions are ignored.

If you omit an optional *var* argument, **Graph** uses the independent variable of the current graphing mode.

Note: Not all optional arguments are valid in all modes because you can never have all four arguments at the same time.

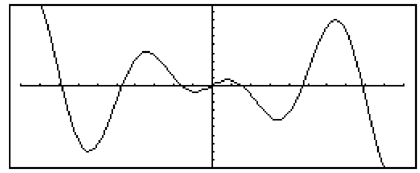
Some valid variations of this instruction are:

Function graphing	**Graph** *expr*, *x*
Parametric graphing	**Graph** *xExpr*, *yExpr*, *t*
Polar graphing	**Graph** *expr*, θ
Sequence graphing	Not allowed.
3D graphing	**Graph** *expr*, *x*, *y*

Note: Use **ClrGraph** (page 381) to clear these functions, or go to the Y= Editor to re-enable the system Y= functions.

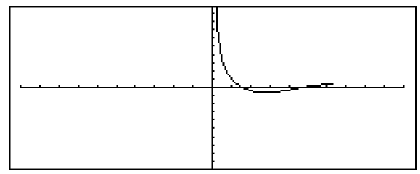
In function graphing mode and ZoomStd window:

```
Graph 1.25a*cos(a),a [ENTER]
```



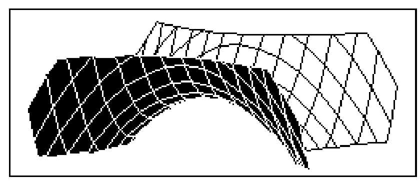
In parametric graphing mode and ZoomStd window:

```
Graph time,2cos(time)/time,time
[ENTER]
```



In 3D graphing mode:

```
Graph (v^2 − w^2)/4,v,w [ENTER]
```



identity() — MATH/Matrix menu

identity(*expression***)** ⇒ *matrix*

Returns the identity matrix with a dimension of *expression*.

expression must evaluate to a positive integer.

```
identity(4) [ENTER]
```

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

If CATALOG

If *Boolean expression*
statement

If *Boolean expression* **Then**
block
EndIf

If *Boolean expression* evaluates to true, executes the single statement *statement* or the block of statements *block* before continuing execution.

If *Boolean expression* evaluates to false, continues execution without executing the statement or block of statements.

block can be either a single statement or a sequence of statements separated with the ":" character.

Program segment:

```
    :
:If x<0
:Disp "x is negative"
    :
```

–or–

```
    :
:If x<0 Then
:  Disp "x is negative"
:  abs(x)→x
:EndIf
    :
```

If *Boolean expression* **Then**
block1
Else
block2
EndIf

If *Boolean expression* evaluates to true, executes *block1* and then skips *block2*.

If *Boolean expression* evaluates to false, skips *block1* but executes *block2*.

block1 and *block2* can be a single statement.

Program segment:

```
    :
:If x<0 Then
:  Disp "x is negative"
:    Else
:  Disp "x is positive or zero"
:EndIf
    :
```

If *Boolean expression1* **Then**
block1
ElseIf *Boolean expression2* **Then**
block2
⋮
ElseIf *Boolean expressionN* **Then**
blockN
EndIf

Allows for program branching. If *Boolean expression1* evaluates to true, executes *block1*. If *Boolean expression1* evaluates to false, evaluates *Boolean expression2*, etc.

Program segment:

```
    :
:If choice=1 Then
:  Goto option1
:  ElseIf choice=2 Then
:    Goto option2
:  ElseIf choice=3 Then
:    Goto option3
:  ElseIf choice=4 Then
:    Disp "Exiting Program"
:    Return
:EndIf
    :
```

imag() MATH/Complex menu

imag(*expression1***)** ⇒ *expression*

imag(*expression1***)** returns the imaginary part of the argument.

Note: All undefined variables are treated as real variables. See also **real()** (page 432).

```
imag(1+2i) [ENTER]        2
imag(z) [ENTER]           0
imag(x+iy) [ENTER]        y
```

---

imag(*list1***)** ⇒ *list*

Returns a list of the imaginary parts of the elements.

```
imag({-3,4-i,i}) [ENTER]      {0 -1 1}
```

---

imag(*matrix1***)** ⇒ *matrix*

Returns a matrix of the imaginary parts of the elements.

imag([a,b;ic,id]) [ENTER] $\begin{bmatrix} 0 & 0 \\ c & d \end{bmatrix}$

Input CATALOG

Input

Pauses the program, displays the current Graph screen, and lets you update variables *xc* and *yc* (also *rc* and θ*c* for polar coordinate mode) by positioning the graph cursor.

When you press [ENTER], the program resumes.

Program segment:

```
   :
:© Get 10 points from the Graph
   Screen
:For i,1,10
:  Input
:  xc→XLIST[i]
:  yc→YLIST[i]
:EndFor
   :
```

Input [*promptString*,] *var*

Input [*promptString*], *var* pauses the program, displays *promptString* on the Program I/O screen, waits for you to enter an expression, and stores the expression in variable *var*.

If you omit *promptString*, "?" is displayed as a prompt.

Program segment:

```
   :
:For i,1,9,1
:  "Enter x" & string(i)→str1
:  Input str1,#(right(str1,2))
:EndFor
   :
```

InputStr CATALOG

InputStr [*promptString*,] *var*

Pauses the program, displays *promptString* on the Program I/O screen, waits for you to enter a response, and stores your response as a string in variable *var*.

If you omit *promptString*, "?" is displayed as a prompt.

Note: The difference between **Input** and **InputStr** is that **InputStr** always stores the result as a string so that " " are not required.

Program segment:

```
   :
:InputStr "Enter Your Name",str1
   :
```

inString() MATH/String menu

inString(*srcString*, *subString*[, *start*]**)** ⇒ *integer*

Returns the character position in string *srcString* at which the first occurrence of string *subString* begins.

start, if included, specifies the character position within *srcString* where the search begins. Default = 1 (the first character of *srcString*).

If *srcString* does not contain *subString* or *start* is > the length of *srcString*, returns zero.

```
inString("Hello there","the")
[ENTER]                                  7

"ABCEFG"→s1:If inString(s1,
"D")=0:Disp "D not found." [ENTER]
                            D not found.
```

int() CATALOG

int(*expression***)** ⇒ *integer*
int(*list1***)** ⇒ *list*
int(*matrix1***)** ⇒ *matrix*

Returns the greatest integer that is less than or equal to the argument. This function is identical to **floor()**.

The argument can be a real or a complex number.

For a list or matrix, returns the greatest integer of each of the elements.

```
int(⁻2.5) [ENTER]                    ⁻3.
int([-1.234,0,0.37]) [ENTER]
                               [-2. 0 0.]
```

intDiv() CATALOG

intDiv(*number1***,** *number2***)** ⇒ *integer*
intDiv(*list1***,** *list2***)** ⇒ *list*
intDiv(*matrix1***,** *matrix2***)** ⇒ *matrix*

Returns the signed integer part of argument 1 divided by argument 2.

For lists and matrices returns the signed integer part of argument 1 divided by argument 2 for each element pair.

```
intDiv(⁻7,2) [ENTER]                  ⁻3
intDiv(4,5) [ENTER]                    0
intDiv({12,⁻14,⁻16},{5,4,⁻3}) [ENTER]
                               {2 ⁻3 5}
```

integrate See ∫, page 464.

iPart() MATH/Number menu

iPart(*number***)** ⇒ *integer*
iPart(*list1***)** ⇒ *list*
iPart(*matrix1***)** ⇒ *matrix*

Returns the integer part of the argument.

For lists and matrices, returns the integer part of each element.

The argument can be a real or a complex number.

```
iPart(⁻1.234) [ENTER]                 ⁻1.
iPart({3/2,⁻2.3,7.003}) [ENTER]
                               {1 ⁻2. 7.}
```

Item CATALOG

Item *itemNameString*
Item *itemNameString***,** *label*

Valid only within a **Custom...EndCustm** or **ToolBar...EndTBar** block. Sets up a drop-down menu element to let you paste text to the cursor position (**Custom**) or branch to a label (**ToolBar**).

Note: Branching to a label is not allowed within a **Custom** block (page 386).

See **Custom** example on page 386.

Lbl — CATALOG

Lbl *labelName*

Defines a label with the name *labelName* in the program.

You can use a **Goto** *labelName* instruction to transfer program control to the instruction immediately following the label.

labelName must meet the same naming requirements as a variable name.

Program segment:

```
  :
:Lbl lbl1
:InputStr "Enter password", str1
:If str1≠password
:  Goto lbl1
:Disp "Welcome to ..."
  :
```

lcm() — MATH/Number menu

lcm(*number1*, *number2***)** ⇒ *expression*
lcm(*list1*, *list2***)** ⇒ *list*
lcm(*matrix1*, *matrix2***)** ⇒ *matrix*

Returns the least common multiple of the two arguments. The **lcm** of two fractions is the **lcm** of their numerators divided by the **gcd** of their denominators. The **lcm** of fractional floating-point numbers is their product.

For two lists or matrices, returns the least common multiples of the corresponding elements.

```
lcm(6,9) [ENTER]                    18
lcm({1/3,-14,16},{2/15,7,5}) [ENTER]
                           {2/3 14 80}
```

left() — MATH/String menu

left(*sourceString*[, *num*]**)** ⇒ *string*

Returns the leftmost *num* characters contained in character string *sourceString*.

If you omit *num*, returns all of *sourceString*.

```
left("Hello",2) [ENTER]             "He"
```

---

left(*list1*[, *num*]**)** ⇒ *list*

Returns the leftmost *num* elements contained in *list1*.

If you omit *num*, returns all of *list1*.

```
left({1,3,-2,4},3) [ENTER]    {1 3 -2}
```

---

left(*comparison***)** ⇒ *expression*

Returns the left-hand side of an equation or inequality.

```
left(x<3) [ENTER]                    x
```

limit()   MATH/Calculus menu

limit(*expression1*, *var*, *point*[, *direction*]**)** ⇒ *expression*
limit(*list1*, *var*, *point*[, *direction*]**)** ⇒ *list*
limit(*matrix1*, *var*, *point*[, *direction*]**)** ⇒ *matrix*

Returns the limit requested.

direction: negative=from left, positive=from right, otherwise=both. (If omitted, *direction* defaults to both.)

Limits at positive ∞ and at negative ∞ are always converted to one-sided limits from the finite side.

Depending on the circumstances, **limit()** returns itself or undef when it cannot determine a unique limit. This does not necessarily mean that a unique limit does not exist. undef means that the result is either an unknown number with finite or infinite magnitude, or it is the entire set of such numbers.

```
limit(2x+3,x,5) [ENTER]                    13
limit(1/x,x,0,1) [ENTER]                   ∞
limit(sin(x)/x,x,0) [ENTER]                1
limit((sin(x+h)-sin(x))/h,h,0) [ENTER]
                                       cos(x)
limit((1+1/n)^n,n,∞) [ENTER]               e
```

limit() uses methods such as L'Hopital's rule, so there are unique limits that it cannot determine. If *expression1* contains undefined variables other than *var*, you might have to constrain them to obtain a more concise result.

```
limit(a^x,x,∞) [ENTER]                     undef
limit(a^x,x,∞)|a>1 [ENTER]                 ∞
limit(a^x,x,∞)|a>0 and a<1 [ENTER]         0
```

Limits can be very sensitive to rounding error. When possible, avoid the APPROX setting of the Exact/Approx mode and approximate numbers when computing limits. Otherwise, limits that should be zero or have infinite magnitude probably will not, and limits that should have finite non-zero magnitude might not.

Line   CATALOG

Line *xStart*, *yStart*, *xEnd*, *yEnd*[, *drawMode*]

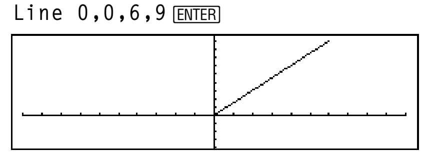
Displays the Graph screen and draws, erases, or inverts a line segment between the window coordinates (*xStart*, *yStart*) and (*xEnd*, *yEnd*), including both endpoints.

If *drawMode* = 1, draws the line (default).
If *drawMode* = 0, turns off the line.
If *drawMode* = ⁻1, turns a line that is on to off or off to on (inverts pixels along the line).

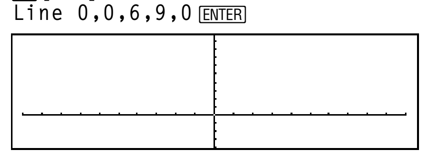
Note: Regraphing erases all drawn items. See also **PxlLine** (page 428).

In the ZoomStd window, draw a line and then erase it.

```
Line 0,0,6,9 [ENTER]
```



```
[♦] [HOME]
Line 0,0,6,9,0 [ENTER]
```



LineHorz CATALOG

LineHorz *y* [, *drawMode*]

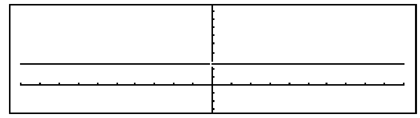
Displays the Graph screen and draws, erases, or inverts a horizontal line at window position *y*.

If *drawMode* = 1, draws the line (default).
If *drawMode* = 0, turns off the line.
If *drawMode* = ⁻1, turns a line that is on to off or off to on (inverts pixels along the line).

Note: Regraphing erases all drawn items. See also **PxlHorz** (page 428).

In a ZoomStd window:

`LineHorz 2.5` [ENTER]



LineTan CATALOG

LineTan *expression1*, *expression2*

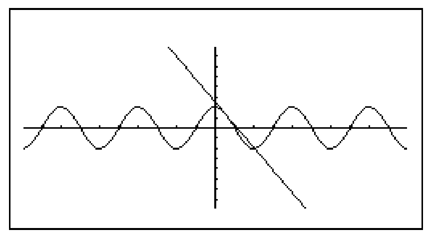
Displays the Graph screen and draws a line tangent to *expression1* at the point specified.

expression1 is an expression or the name of a function, where x is assumed to be the independent variable, and *expression2* is the x value of the point that is tangent.

Note: In the example shown, *expression1* is graphed separately. **LineTan** does not graph *expression1*.

In function graphing mode and a ZoomTrig window:

`Graph cos(x)`
[♦] [HOME]
`LineTan cos(x),π/4` [ENTER]



LineVert CATALOG

LineVert *x* [, *drawMode*]

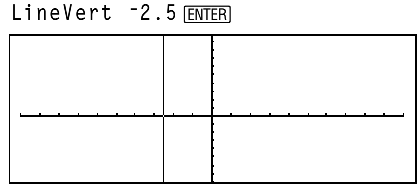
Displays the Graph screen and draws, erases, or inverts a vertical line at window position *x*.

If *drawMode* = 1, draws the line (default).
If *drawMode* = 0, turns off the line.
If *drawMode* = ⁻1, turns a line that is on to off or off to on (inverts pixels along the line).

Note: Regraphing erases all drawn items. See also **PxlVert** (page 429).

In a ZoomStd window:

`LineVert ⁻2.5` [ENTER]



LinReg MATH/Statistics/Regressions menu

LinReg *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

Calculates the linear regression and updates all the system statistics variables.

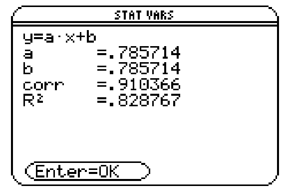
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

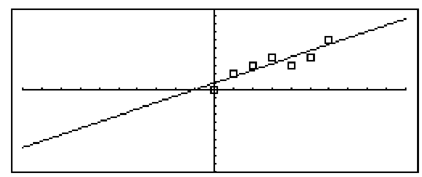
In function graphing mode:

```
{0,1,2,3,4,5,6}→L1 [ENTER]      {0 1 2 ...}
{0,2,3,4,3,4,6}→L2 [ENTER]      {0 2 3 ...}
LinReg L1,L2 [ENTER]                   Done
ShowStat [ENTER]
```




```
[ENTER]
Regeq(x)→y1(x) [ENTER]                 Done
NewPlot 1,1,L1,L2 [ENTER]              Done
```

[♦] [GRAPH]



list▶mat() MATH/List menu

list▶mat(*list* [, *elementsPerRow*]**)** ⇒ *matrix*

Returns a matrix filled row-by-row with the elements from *list*.

elementsPerRow, if included, specifies the number of elements per row. Default is the number of elements in *list* (one row).

If *list* does not fill the resulting matrix, zeros are added.

```
list▶mat({1,2,3}) [ENTER]            [1 2 3]
list▶mat({1,2,3,4,5},2) [ENTER]
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix}$$

ln() [LN] key

ln(*expression1***)** ⇒ *expression*
ln(*list1***)** ⇒ *list*

Returns the natural logarithm of the argument.

For a list, returns the natural logarithms of the elements.

```
ln(2.0) [ENTER]                          .693...
```

If complex format mode is REAL:

```
ln({-3,1.2,5}) [ENTER]
                    Error: Non-real result
```

If complex format mode is RECTANGULAR:

```
ln({-3,1.2,5}) [ENTER]
           {ln(3) + π·i   .182...   ln(5)}
```

LnReg MATH/Statistics/Regressions menu

LnReg *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

Calculates the logarithmic regression and updates all the system statistics variables.

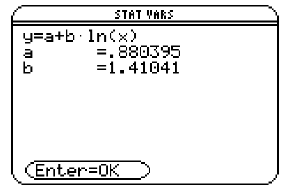
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

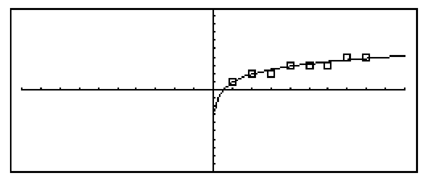
In function graphing mode:

```
{1,2,3,4,5,6,7,8}→L1 [ENTER]   {1 2 3 ...}
{1,2,2,3,3,3,4,4}→L2 [ENTER]   {1 2 2 ...}
LnReg L1,L2 [ENTER]                   Done
ShowStat [ENTER]
```




```
[ENTER]
Regeq(x)→y1(x) [ENTER]              Done
NewPlot 1,1,L1,L2 [ENTER]           Done
```

[◆] [GRAPH]



Local CATALOG

Local *var1*[, *var2*] [, *var3*] ...

Declares the specified *vars* as local variables. Those variables exist only during evaluation of a program or function and are deleted when the program or function finishes execution.

Note: Local variables save memory because they only exist temporarily. Also, they do not disturb any existing global variable values. Local variables must be used for **For** loops and for temporarily saving values in a multi-line function since modifications on global variables are not allowed in a function.

Program listing:

```
:prgmname()
:Prgm
:Local x,y
:Input "Enter x",x
:Input "Enter y",y
:Disp x*y
:EndPrgm
```

Note: *x* and *y* do not exist after the program executes.

Lock CATALOG

Lock *var1*[, *var2*] ...

Locks the specified variables. This prevents you from accidentally deleting or changing the variable without first using the unlock instruction on that variable.

In the example to the right, the variable L1 is locked and cannot be deleted or modified.

Note: The variables can be unlocked using the **unlock** command (page 451).

```
{1,2,3,4}→L1 [ENTER]              {1,2,3,4}
Lock L1 [ENTER]                        Done
DelVar L1 [ENTER]
  Error: Variable is locked or protected
```

log() CATALOG

log(*expression1***)** ⇒ *expression*
log(*list1***)** ⇒ *list*

Returns the base-10 logarithm of the argument.

For a list, returns the base-10 logs of the elements.

```
log(2.0) [ENTER]                    .301...
```

If complex format mode is REAL:

```
log({⁻3,1.2,5}) [ENTER]
              Error: Non-real result
```

If complex format mode is RECTANGULAR:

```
log({⁻3,1.2,5}) [ENTER]
```

$\{\frac{\ln(3)}{\ln(10)} + \frac{\pi}{\ln(10)} \cdot i \quad .079\ldots \quad \frac{\ln(5)}{\ln(10)}\}$

Loop CATALOG

Loop
  block
EndLoop

Repeatedly executes the statements in *block*. Note that the loop will be executed endlessly, unless a **Goto** or **Exit** instruction is executed within *block*.

block is a sequence of statements separated with the ":" character.

Program segment:

```
  :
:1→i
:Loop
:  Rand(6)→die1
:  Rand(6)→die2
:  If die1=6 and die2=6
:    Goto End
:  i+1→i
:EndLoop
:Lbl End
:Disp "The number of rolls is", i
  :
```

mat▸list() MATH/List menu

mat▸list(*matrix***)** ⇒ *list*

Returns a list filled with the elements in *matrix*. The elements are copied from *matrix* row by row.

```
mat▸list([1,2,3]) [ENTER]           {1 2 3}
[1,2,3;4,5,6]→M1 [ENTER]
```

$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$

```
mat▸list(M1) [ENTER]          {1 2 3 4 5 6}
```

max() MATH/List menu

max(*expression1***,** *expression2***)** ⇒ *expression*
max(*list1***,** *list2***)** ⇒ *list*
max(*matrix1***,** *matrix2***)** ⇒ *matrix*

Returns the maximum of the two arguments. If the arguments are two lists or matrices, returns a list or matrix containing the maximum value of each pair of corresponding elements.

```
max(2.3,1.4) [ENTER]                    2.3
max({1,2},{⁻4,3}) [ENTER]             {1 3}
```

---

max(*list***)** ⇒ *expression*

Returns the maximum element in *list*.

```
max({0,1,⁻7,1.3,.5}) [ENTER]            1.3
```

---

max(*matrix1***)** ⇒ *matrix*

Returns a row vector containing the maximum element of each column in *matrix1*.

Note: See also **fMax()** (page 400) and **min()** (page 417).

```
max([1,⁻3,7;⁻4,0,.3]) [ENTER]     [1 0 7]
```

mean() MATH/Statistics menu

mean(*list***)** ⇒ *expression*

Returns the mean of the elements in *list*.

```
mean({.2,0,1,⁻.3,.4}) [ENTER]          .26
```

---

mean(*matrix1***)** ⇒ *matrix*

Returns a row vector of the means of all the columns in *matrix1*.

In vector format rectangular mode:

```
mean([.2,0;⁻1,3;.4,⁻.5]) [ENTER]
                      [⁻.133... .833...]

mean([1/5,0;⁻1,3;2/5,⁻1/2]) [ENTER]
                           [⁻2/15  5/6]
```

median() MATH/Statistics menu

median(*list***)** ⇒ *expression*

Returns the median of the elements in *list1*.

```
median({.2,0,1,⁻.3,.4}) [ENTER]        .2
```

---

median(*matrix1***)** ⇒ *matrix*

Returns a row vector containing the medians of the columns in *matrix1*.

Note: All entries in the list or matrix must simplify to numbers.

```
median([.2,0;1,⁻.3;.4,⁻.5]) [ENTER]
                               [.4 ⁻.3]
```

MedMed MATH/Statistics/Regressions menu

MedMed *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

Calculates the median-median line and updates all the system statistics variables.

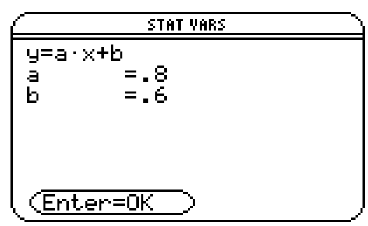
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

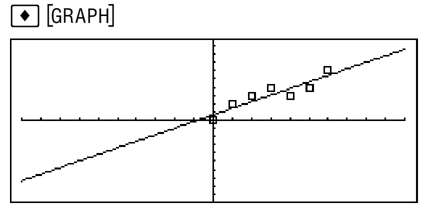
In function graphing mode:

```
{0,1,2,3,4,5,6}→L1 [ENTER]     {0 1 2 ...}
{0,2,3,4,3,4,6}→L2 [ENTER]     {0 2 3 ...}
MedMed L1,L2 [ENTER]                  Done
ShowStat [ENTER]
```




```
[ENTER]
Regeq(x)→y1(x) [ENTER]                Done
NewPlot 1,1,L1,L2 [ENTER]             Done
```

[♦] [GRAPH]



mid() MATH/String menu

mid(_sourceString_, _start_[, _count_]**)** ⇒ _string_

Returns _count_ characters from character string _sourceString_, beginning with character number _start_.

If _count_ is omitted or is greater than the dimension of _sourceString_, returns all characters from _sourceString_, beginning with character number _start_.

count must be ≥ 0. If _count_ = 0, returns an empty string.

```
mid("Hello there",2) [ENTER]
                          "ello there"
mid("Hello there",7,3) [ENTER]    "the"
mid("Hello there",1,5) [ENTER]  "Hello"
mid("Hello there",1,0) [ENTER]       ""
```

---

mid(_sourceList_, _start_ [, _count_]**)** ⇒ _list_

Returns _count_ elements from _sourceList_, beginning with element number _start_.

If _count_ is omitted or is greater than the dimension of _sourceList_, returns all elements from _sourceList_, beginning with element number _start_.

count must be ≥ 0. If count = 0, returns an empty list.

```
mid({9,8,7,6},3) [ENTER]          {7 6}
mid({9,8,7,6},2,2) [ENTER]        {8 7}
mid({9,8,7,6},1,2) [ENTER]        {9 8}
mid({9,8,7,6},1,0) [ENTER]           {}
```

---

mid(_sourceStringList_, _start_[, _count_]**)** ⇒ _list_

Returns _count_ strings from the list of strings _sourceStringList_, beginning with element number _start_.

```
mid({"A","B","C","D"},2,2) [ENTER]
                            {"B" "C"}
```

min() MATH/List menu

min(_expression1_, _expression2_**)** ⇒ _expression_
min(_list1_, _list2_**)** ⇒ _list_
min(_matrix1_, _matrix2_**)** ⇒ _matrix_

Returns the minimum of the two arguments. If the arguments are two lists or matrices, returns a list or matrix containing the minimum value of each pair of corresponding elements.

```
min(2.3,1.4) [ENTER]                1.4
min({1,2},{⁻4,3}) [ENTER]        {⁻4 2}
```

---

min(_list_**)** ⇒ _expression_

Returns the minimum element of _list_.

```
min({0,1,⁻7,1.3,.5}) [ENTER]         ⁻7
```

---

min(_matrix1_**)** ⇒ _matrix_

Returns a row vector containing the minimum element of each column in _matrix1_.

Note: See also **fMin()** (page 401) and **max()** 415).

```
min([1,⁻3,7;⁻4,0,.3]) [ENTER]
                          [⁻4 ⁻3 .3]
```

mod() MATH/Number menu

mod(*expression1*, *expression2***)** ⇒ *expression*
mod(*list1*, *list2***)** ⇒ *list*
mod(*matrix1*, *matrix2***)** ⇒ *matrix*

Returns the first argument modulo the second argument as defined by the identities:

mod(x,0) ≡ x
mod(x,y) ≡ x−y floor(x/y)

When the second argument is non-zero, the result is periodic in that argument. The result is either zero or has the same sign as the second argument.

If the arguments are two lists or two matrices, returns a list or matrix containing the modulo of each pair of corresponding elements.

Note: See also **remain()** (page 433).

```
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
mod({12,⁻14,16},{9,7,⁻5}) [ENTER]
                            {3 0 ⁻4}
```

MoveVar CATALOG

MoveVar *var*, *oldFolder*, *newFolder*

Moves variable *var* from *oldFolder* to *newFolder*. If *newFolder* does not exist, **MoveVar** creates it.

```
{1,2,3,4}→L1 [ENTER]          {1 2 3 4}
MoveVar L1,Main,Games [ENTER]      Done
```

mRow() MATH/Matrix/Row ops menu

mRow(*expression*, *matrix1*, *index***)** ⇒ *matrix*

Returns a copy of *matrix1* with each element in row *index* of *matrix1* multiplied by *expression*.

```
mRow(⁻1/3,[1,2;3,4],2) [ENTER]
                         [ 1    2  ]
                         [-1  -4/3 ]
```

mRowAdd() MATH/Matrix/Row ops menu

mRowAdd(*expression*, *matrix1*, *index1*, *index2***)** ⇒ *matrix*

Returns a copy of *matrix1* with each element in row *index2* of *matrix1* replaced with:

expression × row *index1* + row *index2*

```
mRowAdd(⁻3,[1,2;3,4],1,2) [ENTER]
                          [1   2]
                          [0  -2]
mRowAdd(n,[a,b;c,d],1,2) [ENTER]
                    [a        b    ]
                    [a·n+c    b·n+d]
```

nCr() MATH/Probability menu

nCr(*expression1*, *expression2***)** ⇒ *expression*

For integer *expression1* and *expression2* with *expression1* ≥ *expression2* ≥ 0, **nCr()** is the number of combinations of *expression1* things taken *expression2* at a time. (This is also known as a binomial coefficient.)

Both arguments can be integers or symbolic expressions.

nCr(*expression*, 0**)** ⇒ 1

nCr(*expression*, *negInteger***)** ⇒ 0

nCr(*expression*, *posInteger***)** ⇒ *expression* • (*expression*−1)... (*expression*−*posInteger*+1)/ *posInteger*!

nCr(*expression*, *nonInteger***)** ⇒ *expression*!/ ((*expression*−*nonInteger*)! • *nonInteger*!)

```
nCr(z,3)                 z•(z-2)•(z-1)/6
ans(1)|z=5               10
nCr(z,c)                 z!/(c!(z-c)!)
ans(1)/nPr(z,c)          1/c!
```

---

nCr(*list1*, *list2***)** ⇒ *list*

Returns a list of combinations based on the corresponding element pairs in the two lists.

The arguments must be the same size list.

```
nCr({5,4,3},{2,4,2}) [ENTER] {10 1 3}
```

---

nCr(*matrix1*, *matrix2***)** ⇒ *matrix*

Returns a matrix of combinations based on the corresponding element pairs in the two matrices.

The arguments must be the same size matrix.

```
nCr([6,5;4,3],[2,2;2,2]) [ENTER]
                         [15 10]
                         [6   3]
```

nDeriv() MATH/Calculus menu

nDeriv(*expression1*, *var*[, *h*]**)** ⇒ *expression*

Returns the numerical derivative as an expression. Uses the central difference quotient formula.

h is the step value. If *h* is omitted, it defaults to 0.001.

Note: See also **avgRC()** (page 379) and ***d*()** (page 388).

```
nDeriv(cos(x),x,h) [ENTER]
                     ⁻(cos(x-h)-cos(x+h))/(2•h)
limit(nDeriv(cos(x),x,h),h,0) [ENTER]
                     ⁻sin(x)
nDeriv(x^3,x,0.01) [ENTER]
                     3.•(x²+.000033)
nDeriv(cos(x),x)|x=π/2 [ENTER]
                     ⁻1.
```

NewData CATALOG

NewData *dataVar*, *list1*[, *list2*] [, *list3*]**...**

Creates data variable *dataVar*, where the columns are the lists in order.

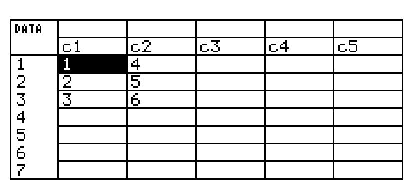
Must have at least one list.

list1, *list2*, ..., *listn* can be lists as shown, expressions that resolve to lists, or list variable names.

NewData makes the new variable current in the Data/Matrix Editor.

```
NewData mydata,{1,2,3},{4,5,6} [ENTER]
                     Done
```

(Go to the Data/Matrix Editor and open the *var* `mydata` to display the data variable below.)



NewFold CATALOG

NewFold *folderName*

Creates a user-defined folder with the name *folderName*, and then sets the current folder to that folder. After you execute this instruction, you are in the new folder.

```
NewFold games [ENTER]                Done
```

newList() CATALOG

newList(*numElements***)** ⇒ *list*

Returns a list with a dimension of *numElements*. Each element is zero.

```
newList(4) [ENTER]              {0 0 0 0}
```

newMat() CATALOG

newMat(*numRows*, *numColumns***)** ⇒ *matrix*

Returns a matrix of zeros with the dimension *numRows* by *numColumns*.

newMat(2,3) [ENTER] $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

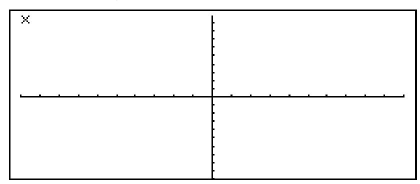
NewPic CATALOG

NewPic *matrix*, *picVar* [, *maxRow*][, *maxCol*]

Creates a pic variable *picVar* based on *matrix*. *matrix* must be an *n*×*2* matrix in which each row represents a pixel. Pixel coordinates start at 0,0. If *picVar* already exists, **NewPic** replaces it.

The default for *picVar* is the minimum area required for the matrix values. The optional arguments, *maxRow* and *maxCol*, determine the maximum boundary limits for *picVar*.

```
NewPic [1,1;2,2;3,3;4,4;5,5;
  5,1;4,2;2,4;1,5],xpic [ENTER]      Done
RclPic xpic [ENTER]
```



NewPlot CATALOG

NewPlot *n*, *type*, *xList* [,[*yList*], [*frqList*], [*catList*], [*includeCatList*], [*mark*] [, *bucketSize*]]

Creates a new plot definition for plot number *n*.

type specifies the type of the graph plot.
1 = scatter plot
2 = xyline plot
3 = box plot
4 = histogram

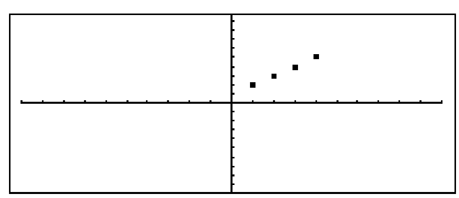
mark specifies the display type of the mark.
1 = ▫ (box)
2 = × (cross)
3 = + (plus)
4 = ▪ (square)
5 = · (dot)

bucketSize is the width of each histogram "bucket" (*type* = 4), and will vary based on the window variables xmin and xmax. *bucketSize* must be >0. Default = 1.

Note: *n* can be 1–9. Lists must be variable names or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor), except for *includeCatList*, which does not have to be a variable name and cannot be c1–c99.

```
FnOff [ENTER]                        Done
PlotsOff [ENTER]                     Done
{1,2,3,4}→L1 [ENTER]            {1 2 3 4}
{2,3,4,5}→L2 [ENTER]            {2 3 4 5}
NewPlot 1,1,L1,L2,,,,4 [ENTER]       Done
```

Press [◆] [GRAPH] to display:



nInt()   MATH/Calculus menu

nInt(*expression1*, *var*, *lower*, *upper***)** ⇒ *expression*

```
nInt(e^(-x^2),x,-1,1) ENTER                 1.493...
```

If the integrand *expression1* contains no variable other than *var*, and if *lower* and *upper* are constants, positive ∞, or negative ∞, then **nInt()** returns an approximation of **∫(***expression1*, *var*, *lower*, *upper***)**. This approximation is a weighted average of some sample values of the integrand in the interval *lower<var<upper*.

The goal is six significant digits. The adaptive algorithm terminates when it seems likely that the goal has been achieved, or when it seems unlikely that additional samples will yield a worthwhile improvement.

```
nInt(cos(x),x,-π,π+1E-12) ENTER
                                        -1.041...E-12
```

```
∫(cos(x),x,-π,π+10^(-12)) ENTER
```

$$^-\sin\left(\frac{1}{1000000000000}\right)$$

A warning is displayed ("Questionable accuracy") when it seems that the goal has not been achieved.

```
ans(1)◆ ENTER                               -1.E-12
```

Nest **nInt()** to do multiple numeric integration. Integration limits can depend on integration variables outside them.

```
nInt(nInt(e^(-x*y)/√(x^2-y^2),
  y,-x,x),x,0,1) ENTER                      3.304...
```

Note: See also **∫()** (page 464).

norm()   MATH/Matrix/Norms menu

norm(*matrix***)** ⇒ *expression*

Returns the Frobenius norm.

```
norm([a,b;c,d]) ENTER           √(a²+b²+c²+d²)
norm([1,2;3,4]) ENTER                      √30
```

not()   MATH/Test menu

not(*Boolean expression1***)** ⇒ *Boolean expression*

Returns true, false, or a simplified *Boolean expression1*.

```
not(2>=3)                                 true
not(x<2) ENTER                           x ≥ 2
not(not(innocent)) ENTER              innocent
```

nPr() MATH/Probability menu

nPr(*expression1*, *expression2***)** ⇒ *expression*

For integer *expression1* and *expression2* with *expression1* ≥ *expression2* ≥ 0, **nPr()** is the number of permutations of *expression1* things taken *expression2* at a time.

Both arguments can be integers or symbolic expressions.

nPr(*expression*, 0**)** ⇒ 1

nPr(*expression*, *negInteger***)** ⇒ 1/((*expression*+1)· (*expression*+2)... (*expression*−*negInteger*))

nPr(*expression*, *posInteger***)** ⇒ *expression*· (*expression*−1)... (*expression*−*posInteger*+1)

nPr(*expression*, *nonInteger***)** ⇒ *expression*!/ (*expression*−*nonInteger*)!

```
nPr(z,3) [ENTER]                z·(z-2)·(z-1)
ans(1)|z=5 [ENTER]                         60
```

$$nPr(z,^-3) \; [ENTER] \qquad \frac{1}{(z+1)\cdot(z+2)\cdot(z+3)}$$

$$nPr(z,c) \; [ENTER] \qquad \frac{z!}{(z-c)!}$$

```
ans(1)*nPr(z-c,-c) [ENTER]                  1
```

---

nPr(*list1*, *list2***)** ⇒ *list*

Returns a list of permutations based on the corresponding element pairs in the two lists.

The arguments must be the same size list.

```
nPr({5,4,3},{2,4,2}) [ENTER]      {20 24 6}
```

---

nPr(*matrix1*, *matrix2***)** ⇒ *matrix*

Returns a matrix of permutations based on the corresponding element pairs in the two matrices.

The arguments must be the same size matrix.

```
nPr([6,5;4,3],[2,2;2,2]) [ENTER]
```

$$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$$

nSolve() MATH/Algebra menu

nSolve(*equation*, *var***)** ⇒ *number or error_string*

Iteratively searches for one approximate real numeric solution to *equation* for its one variable *var*.

nSolve() is often much faster than **solve()** or **zeros()**, particularly if the “|” operator is used to constrain the search to a relatively small interval that contains exactly one simple solution.

nSolve() attempts to determine either one point where the residual is zero or two relatively close points where the residual has opposite signs and the magnitude of the residual is not excessive. If it cannot achieve this using a modest number of sample points, it returns the string “no solution found.”

Therefore, if you use **nSolve()** in a program, you can use **getType()** (page 405), to check for a numeric result before using the result in an algebraic expression.

Note: See also **cSolve()** (page 385), **cZeros()** (page 387), **solve()** (page 442), and **zeros()** (page 453).

```
nSolve(x^2+5x-25=9,x) [ENTER]
                                     3.844...
nSolve(x^2+5x-25=9,x)|x<0 [ENTER]
                                    -8.844...
nSolve(((1+r)^24-1)/r=26,r)|r>0
  and r<.25 [ENTER]                   .0068...

nSolve(x^2=-1,x) [ENTER]
                          "no solution found"
```

OneVar — MATH/Statistics menu

OneVar *list1* [[, *list2*] [, *list3*] [, *list4*]]

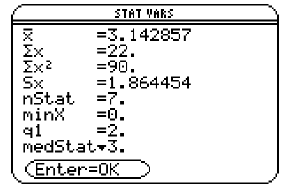
Calculates 1-variable statistics and updates all the system statistics variables.

All the lists must have equal dimensions except for *list4*.

list1 represents xlist.
list2 represents frequency.
list3 represents category codes.
list4 represents category include list.

Note: *list1* through *list3* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list4* does not have to be a variable name and cannot be c1–c99.

```
{0,2,3,4,3,4,6}→L1 [ENTER]
OneVar L1 [ENTER]                  Done
ShowStat [ENTER]
```



or — MATH/Test menu

Boolean expression1 **or** *Boolean expression2* ⇒ *Boolean expression*

Returns true or false or a simplified form of the original entry.

Returns true if either or both expressions simplify to true. Returns false only if both expressions evaluate to false.

Note: See **xor** (page 453).

```
x≥3 or x≥4 [ENTER]                 x ≥ 3
```

Program segment:

```
   ⋮
If x<0 or x≥5
  Goto END
   ⋮
If choice=1 or choice=2
  Disp "Wrong choice"
   ⋮
```

ord() — MATH/String menu

ord(*string***)** ⇒ *integer*
ord(*list1***)** ⇒ *list*

Returns the numeric code of the first character in character string *string*, or a list of the first characters of each list element.

See Appendix B for a complete listing of TI-92 characters and their codes.

```
ord("hello") [ENTER]                104
char(104) [ENTER]                   "h"
ord(char(24)) [ENTER]                24
ord({"alpha","beta"}) [ENTER]   {97  98}
```

Output — CATALOG

Output *row*, *column*, *exprOrString*

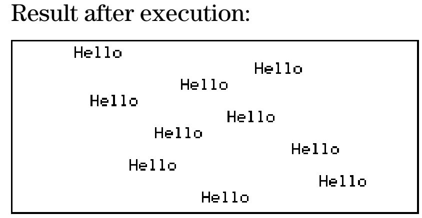
Displays *exprOrString* (an expression or character string) on the Program I/O screen at the text coordinates (*row*, *column*).

If Pretty Print = ON, *exprOrString* is “pretty printed.”

Program segment:

```
   ⋮
:randseed(1147)
:ClrIO
:For i,1,100,10
:  Output i, rand(200),"Hello"
:EndFor
   ⋮
```

Result after execution:



P▸Rx() MATH/Angle menu

P▸Rx(*rExpression*, θ*Expression***)** ⇒ *expression*
P▸Rx(*rList*, θ*List***)** ⇒ *list*
P▸Rx(*rMatrix*, θ*Matrix***)** ⇒ *matrix*

Returns the equivalent x-coordinate of the (r, θ) pair.

Note: The θ argument is interpreted as either a degree or radian angle, according to the current angle mode. If the argument is an expression, you can use ° (page 467) or ^{r} (page 467) to override the angle mode setting temporarily.

In Radian angle mode:

```
P▸Rx(r,θ) [ENTER]                        cos(θ)·r
P▸Rx(4,60°) [ENTER]                             2
P▸Rx({-3,10,1.3},{π/3,-π/4,0}) [ENTER]
                            {-3/2  5·√2  1.3}
```

P▸Ry() MATH/Angle menu

P▸Ry(*rExpression*, θ*Expression***)** ⇒ *expression*
P▸Ry(*rList*, θ*List***)** ⇒ *list*
P▸Ry(*rMatrix*, θ*Matrix***)** ⇒ *matrix*

Returns the equivalent y-coordinate of the (r, θ) pair.

Note: The θ argument is interpreted as either a degree or radian angle, according to the current angle mode. If the argument is an expression, you can use ° (page 467) or ^{r} (page 467) to override the angle mode setting temporarily.

In Radian angle mode:

```
P▸Ry(r,θ) [ENTER]                        sin(θ)·r
P▸Ry(4,60°) [ENTER]                         2·√3
P▸Ry({-3,10,1.3},{π/3,-π/4,0}) [ENTER]
```

$$\left\{ \frac{-3\cdot\sqrt{3}}{2} \quad -5\cdot\sqrt{2} \quad 0. \right\}$$

PassErr CATALOG

PassErr

Passes an error to the next level.

If “errornum” is zero, **PassErr** does not do anything.

The **Else** clause in the program should use **ClrErr** or **PassErr**. If the error is to be processed or ignored, use **ClrErr**. If what to do with the error is not known, use **PassErr** to send it to the next error handler. (See also **ClrErr**.)

Program listing:
(See **ClrErr** on page 381.)

Pause CATALOG

Pause [*expression*]

Suspends program execution.

If you include *expression*, displays *expression* on the Program I/O screen.

Program execution resumes when you press [ENTER].

Program segment:

```
   :
:DelVar temp
:1→temp[1]
:1→temp[2]
:Disp temp[2]
:© Guess the Pattern
:For i,3,20
:  temp[i-2]+temp[i-1] →temp[i]
:  Disp temp[i]
:  Disp temp, "Can you guess the
     next number?"
:  Pause
:EndFor
   :
```

PlotsOff CATALOG

PlotsOff [**1**] [**, 2**] [**, 3**] **...** [**, 9**]

Turns off the specified plots for graphing. When in 2-graph mode, only affects the active graph.

If no parameters, then turns off all plots.

```
PlotsOff 1,2,5 [ENTER]          Done
PlotsOff [ENTER]                Done
```

PlotsOn CATALOG

PlotsOn [**1**] [**, 2**] [**, 3**] **...** [**, 9**]

Turns on the specified plots for graphing. When in 2-graph mode, only affects the active graph.

If you do not include any arguments, turns on all plots.

```
PlotsOn 2,4,5 [ENTER]           Done
PlotsOn [ENTER]                 Done
```

▶Polar MATH/Matrix/Vector ops menu

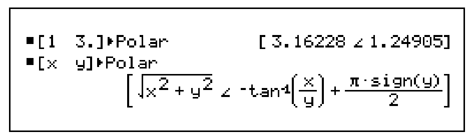
vector **▶Polar**

Displays *vector* in polar form [r ∠θ]. The vector must be of dimension 2 and can be a row or a column.

Note: **▶Polar** is a display-format instruction, not a conversion function. You can use it only at the end of an entry line, and it does not update ans.

Note: See also **▶Rect** (page 433).

```
[1,3.]▶Polar [ENTER]
[x,y]▶Polar [ENTER]
```



polyEval() MATH/List menu

polyEval(*list1***,** *expression1***)** ⇒ *expression*
polyEval(*list1***,** *list2***)** ⇒ *expression*

Interprets the first argument as the coefficients of a descending-degree polynomial, and returns the polynomial evaluated for the value of the second argument.

```
polyEval({a,b,c},x) [ENTER]   a·x²+b·x+c
polyEval({1,2,3,4},2) [ENTER]         26
polyEval({1,2,3,4},{2,⁻7})
[ENTER]                       {26 ⁻262}
```

PopUp CATALOG

PopUp *itemList*, *var*

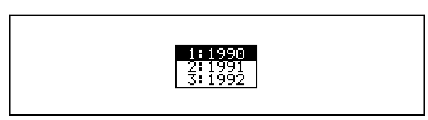
Displays a pop-up menu containing the character strings from *itemList*, waits for you to select an item, and stores the number of your selection in *var*.

The elements of *itemList* must be character strings: {*item1String*, *item2String*, *item3String*, ...}

If *var* already exists and has a valid item number, that item is displayed as the default choice.

itemList must contain at least one choice.

```
PopUp {"1990","1991","1992"},var1
[ENTER]
```



PowerReg MATH/Statistics/Regressions menu

PowerReg *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

Calculates the power regression and updates all the system statistics variables.

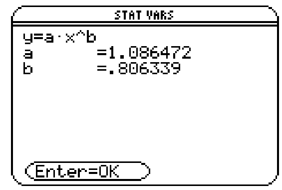
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

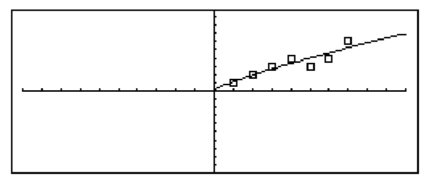
In function graphing mode:

```
{1,2,3,4,5,6,7}→L1 [ENTER]      {1 2 3 ...}
{1,2,3,4,3,4,6}→L2 [ENTER]      {1 2 3 ...}
PowerReg L1,L2 [ENTER]                 Done
ShowStat [ENTER]
```




```
[ENTER]
Regeq(x)→y1(x) [ENTER]                 Done
NewPlot 1,1,L1,L2 [ENTER]              Done
```

[♦] [GRAPH]



Prgm CATALOG

Prgm
⋮
EndPrgm

Required instruction that identifies the beginning of a program. Last line of program must be **EndPrgm**.

Program segment:

```
:prgmname()
:Prgm
:
:
:EndPrgm
```

product() MATH/List menu

product(*list***)** ⇒ *expression*

Returns the product of the elements contained in *list*.

```
product({1,2,3,4}) [ENTER]               24
product({2,x,y}) [ENTER]              2·x·y
```

---

product(*matrix1***)** ⇒ *matrix*

Returns a row vector containing the products of the elements in the columns of *matrix1*.

```
product([1,2,3;4,5,6;7,8,9]) [ENTER]
                             [28 80 162]
```

Prompt CATALOG

Prompt *var1*[, *var2*] [, *var3*] **...**

Displays a prompt on the Program I/O screen for each variable in the argument list, using the prompt var1?. Stores the entered expression in the corresponding variable.

Prompt must have at least one argument.

Program segment:

```
   ⋮
Prompt A,B,C
   ⋮
EndPrgm
```

propFrac() MATH/Algebra menu

propFrac(_expression1_[, _var_]**)** ⇒ _expression_

propFrac(_rational_number_**)** returns _rational_number_ as the sum of an integer and a fraction having the same sign and a greater denominator magnitude than numerator magnitude.

propFrac(_rational_expression,var_**)** returns the sum of proper ratios and a polynomial with respect to _var_. The degree of _var_ in the denominator exceeds the degree of _var_ in the numerator in each proper ratio. Similar powers of _var_ are collected. The terms and their factors are sorted with _var_ as the main variable.

If _var_ is omitted, a proper fraction expansion is done with respect to the most main variable. The coefficients of the polynomial part are then made proper with respect to their most main variable first and so on.

For rational expressions, **propFrac()** is a faster but less extreme alternative to **expand()** (page 397).

```
propFrac(4/3) [ENTER]                 1 + 1/3
propFrac(-4/3) [ENTER]                 -1-1/3
propFrac((x^2+x+1)/(x+1)+
  (y^2+y+1)/(y+1),x) [ENTER]
```

$$\blacksquare\ \text{propFrac}\left(\frac{x^2+x+1}{x+1}+\frac{y^2+y+1}{y+1},x\right) \quad \frac{1}{x+1}+x+\frac{y^2+y+1}{y+1}$$

```
propFrac(ans(1))
```

$$\blacksquare\ \text{propFrac}\left(\frac{1}{x+1}+x+\frac{y^2+y+1}{y+1}\right) \quad \frac{1}{x+1}+x+\frac{1}{y+1}+y$$

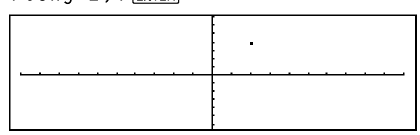
PtChg CATALOG

PtChg _x_, _y_
PtChg _xList_, _yList_

Displays the Graph screen and reverses the screen pixel nearest to window coordinates (_x_, _y_).

Note: PtChg through **PtText** show continuing similar examples.

```
PtChg 2,4 [ENTER]
```

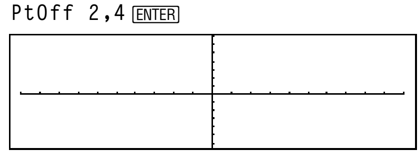


PtOff CATALOG

PtOff _x_, _y_
PtOff _xList_, _yList_

Displays the Graph screen and turns off the screen pixel nearest to window coordinates (_x_, _y_).

```
PtOff 2,4 [ENTER]
```

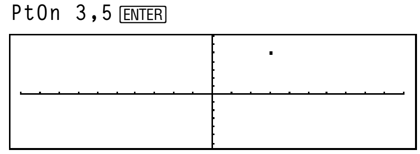


PtOn CATALOG

PtOn _x_, _y_
PtOn _xList_, _yList_

Displays the Graph screen and turns on the screen pixel nearest to window coordinates (_x_, _y_).

```
PtOn 3,5 [ENTER]
```



ptTest() CATALOG

ptTest (_x_, _y_**)** ⇒ _Boolean constant expression_
ptTest (_xList_, _yList_**)** ⇒ _Boolean constant expression_

Returns true or false. Returns true only if the screen pixel nearest to window coordinates (_x_, _y_) is on.

```
ptTest(3,5) [ENTER]                      true
```

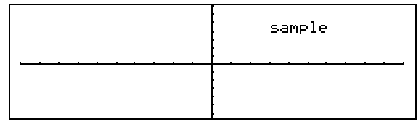
PtText CATALOG

PtText *string*, *x*, *y*

Displays the Graph screen and places the character string *string* on the screen at the pixel nearest the specified (*x*, *y*) window coordinates.

string is positioned with the upper-left corner of its first character at the coordinates.

`PtText "sample",3,5` [ENTER]



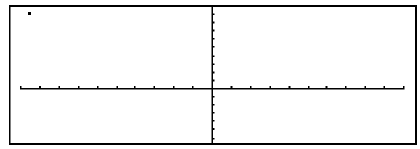
PxlChg CATALOG

PxlChg *row*, *col*
PxlChg *rowList*, *colList*

Displays the Graph screen and reverses the pixel at pixel coordinates (*row*, *col*).

Note: Regraphing erases all drawn items.

`PxlChg 2,4` [ENTER]



PxlCrcl CATALOG

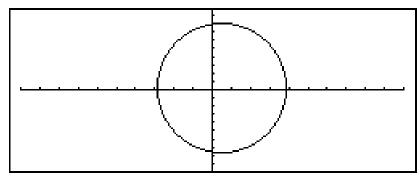
PxlCrcl *row*, *col*, *r* [, *drawMode*]

Displays the Graph screen and draws a circle centered at pixel coordinates (*row*, *col*) with a radius of *r* pixels.

If *drawMode* = 1, draws the circle (default).
If *drawMode* = 0, turns off the circle.
If *drawMode* = -1, inverts pixels along the circle.

Note: Regraphing erases all drawn items. See also **Circle** (page 381).

`PxlCrcl 50,125,40,1` [ENTER]



PxlHorz CATALOG

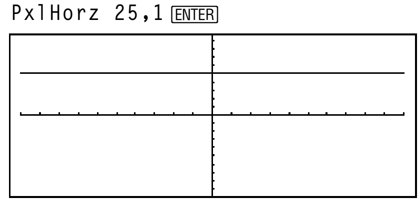
PxlHorz *row* [, *drawMode*]

Displays the Graph screen and draws a horizontal line at pixel position *row*.

If *drawMode* = 1, draws the line (default).
If *drawMode* = 0, turns off the line.
If *drawMode* = -1, turns a line that is on to off or off to on (inverts pixels along the line).

Note: Regraphing erases all drawn items. See also **LineHorz** (page 412).

`PxlHorz 25,1` [ENTER]



PxlLine CATALOG

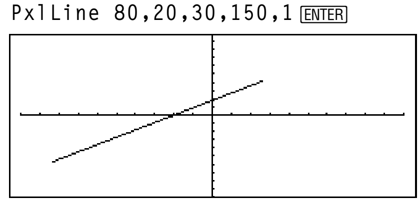
PxlLine *rowStart*, *colStart*, *rowEnd*, *colEnd* [, *drawMode*]

Displays the Graph screen and draws a line between pixel coordinates (*rowStart*, *colStart*) and (*rowEnd*, *colEnd*), including both endpoints.

If *drawMode* = 1, draws the line (default).
If *drawMode* = 0, turns off the line.
If *drawMode* = -1, turns a line that is on to off or off to on (inverts pixels along the line).

Note: Regraphing erases all drawn items. See also **Line** (page 411)

`PxlLine 80,20,30,150,1` [ENTER]



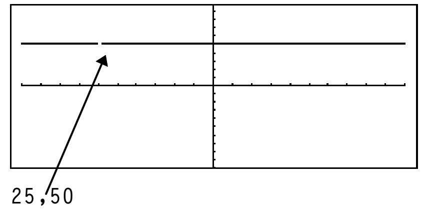
PxlOff CATALOG

PxlOff *row*, *col*
PxlOff *rowList*, *colList*

Displays the Graph screen and turns off the pixel at pixel coordinates (*row*, *col*).

Note: Regraphing erases all drawn items.

```
PxlHorz 25,1 [ENTER]
PxlOff 25,50 [ENTER]
```



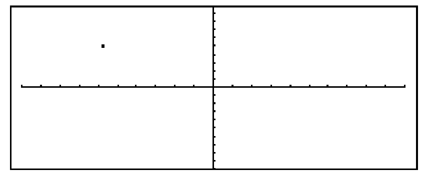
PxlOn CATALOG

PxlOn *row*, *col*
PxlOn *rowList*, *colList*

Displays the Graph screen and turns on the pixel at pixel coordinates (*row*, *col*).

Note: Regraphing erases all drawn items.

```
PxlOn 25,50 [ENTER]
```



pxlTest() CATALOG

pxlTest (*row*, *col***)** ⇒ *Boolean expression*
pxlTest (*rowList*, *colList***)** ⇒ *Boolean expression*

Returns true if the pixel at pixel coordinates (*row*, *col*) is on. Returns false if the pixel is off.

Note: Regraphing erases all drawn items.

```
PxlOn 25,50 [ENTER]
[◆][HOME]
PxlTest(25,50) [ENTER]          true

PxlOff 25,50 [ENTER]
[◆][HOME]
PxlTest(25,50) [ENTER]          false
```

PxlText CATALOG

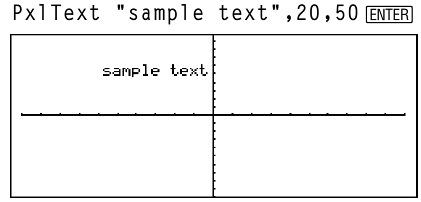
PxlText *string*, *row*, *col*

Displays the Graph screen and places character string *string* on the screen, starting at pixel coordinates (*row*, *col*).

string is positioned with the upper-left corner of its first character at the coordinates.

Note: Regraphing erases all drawn items.

```
PxlText "sample text",20,50 [ENTER]
```



PxlVert CATALOG

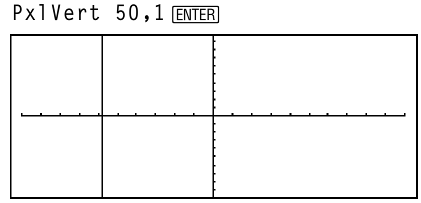
PxlVert *col* [, *drawMode*]

Draws a vertical line down the screen at pixel position *col*.

If *drawMode* = 1, draws the line (default).
If *drawMode* = 0, turns off the line.
If *drawMode* = -1, turns a line that is on to off or off to on (inverts pixels along the line).

Note: Regraphing erases all drawn items. See also **LineVert** (page 412).

```
PxlVert 50,1 [ENTER]
```



QuadReg MATH/Statistics/Regressions menu

QuadReg *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

Calculates the quadratic polynomial regression and updates the system statistics variables.

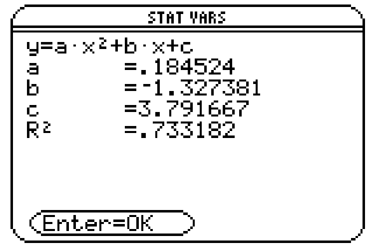
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

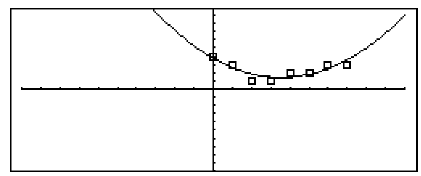
In function graphing mode:

```
{0,1,2,3,4,5,6,7}→L1 [ENTER]  {1 2 3 ...}
{4,3,1,1,2,2,3,3}→L2 [ENTER]  {4 3 1 ...}
QuadReg L1,L2 [ENTER]              Done
ShowStat [ENTER]
```




```
[ENTER]
Regeq(x)→y1(x) [ENTER]             Done
NewPlot 1,1,L1,L2 [ENTER]          Done
```

◆ [GRAPH]



QuartReg MATH/Statistics/Regressions menu

QuartReg *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

Calculates the quartic polynomial regression and updates the system statistics variables.

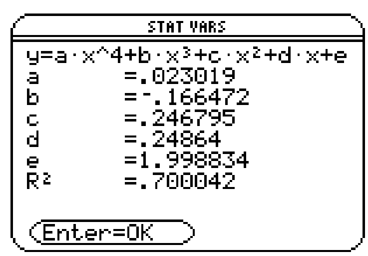
All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

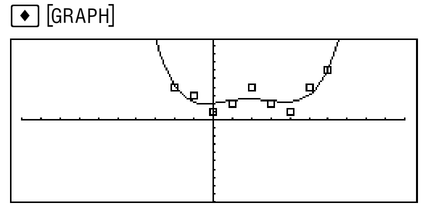
In function graphing mode:

```
{-2,-1,0,1,2,3,4,5,6}→L1 [ENTER]
                           {-2 -1 0 ...}
{4,3,1,2,4,2,1,4,6}→L2 [ENTER]
                            {4 3 1 ...}
QuartReg L1,L2 [ENTER]             Done
ShowStat [ENTER]
```




```
[ENTER]
Regeq(x)→y1(x) [ENTER]             Done
NewPlot 1,1,L1,L2 [ENTER]          Done
```

◆ [GRAPH]



R▸Pθ() MATH/Angle menu

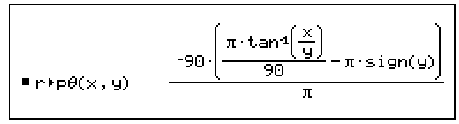
R▸Pθ (*xExpression*, *yExpression*) ⇒ *expression*
R▸Pθ (*xList*, *yList*) ⇒ *list*
R▸Pθ (*xMatrix*, *yMatrix*) ⇒ *matrix*

Returns the equivalent θ-coordinate of the (x, y) pair arguments.

Note: The result is returned as either a degree or radian angle, according to the current angle mode.

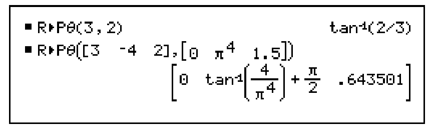
In Degree angle mode:

`R▸Pθ(x,y)` ENTER



In Radian angle mode:

`R▸Pθ(3,2)` ENTER
`R▸Pθ([3,-4,2],[0,π/4,1.5])` ENTER



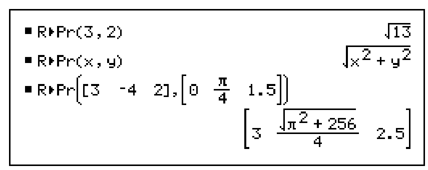
R▸Pr() MATH/Angle menu

R▸Pr (*xExpression*, *yExpression*) ⇒ *expression*
R▸Pr (*xList*, *yList*) ⇒ *list*
R▸Pr (*xMatrix*, *yMatrix*) ⇒ *matrix*

Returns the equivalent r-coordinate of the (*x*,*y*) pair arguments.

In Radian angle mode:

`R▸Pr(3,2)` ENTER
`R▸Pr(x,y)` ENTER
`R▸Pr([3,-4,2],[0,π/4,1.5])` ENTER



rand() MATH/Probability menu

rand(*n*) ⇒ *expression*

n is an integer ≠ zero.

With no parameter, returns the next random number between 0 and 1 in the sequence. When an argument is positive, returns a random integer in the interval [1, *n*]. When an argument is negative, returns a random integer in the interval [⁻*n*, ⁻1].

```
RandSeed 1147 ENTER          Done
```

(Sets the random-number seed.)

```
rand() ENTER                0.158...
rand(6) ENTER                  5
rand(⁻100) ENTER             ⁻49
```

randMat() MATH/Probability menu

randMat(*numRows*, *numColumns*) ⇒ *matrix*

Returns a matrix of integers between -9 and 9 of the specified dimension.

Both arguments must simplify to integers.

```
RandSeed 1147 ENTER          Done
randMat(3,3) ENTER
```

$$\begin{bmatrix} 8 & ^-3 & 6 \\ ^-2 & 3 & ^-6 \\ 0 & 4 & ^-6 \end{bmatrix}$$

(**Note:** The values in this matrix will change each time you press ENTER.)

randNorm() MATH/Probability menu

randNorm(*mean*, *sd*) ⇒ *expression*

Returns a decimal number from the specific normal distribution. It could be any real number but will be heavily concentrated in the interval [*mean*-3*sd, *mean*+3*sd].

```
RandSeed 1147 ENTER          Done
randNorm(0,1) ENTER          0.492...
randNorm(3,4.5) ENTER       ⁻3.543...
```

randPoly() MATH/Probability menu

randPoly(*var*, *order***)** ⇒ *expression*

Returns a polynomial in *var* of the specified order. The coefficients are random integers in the range ⁻9 through 9. The leading coefficient will not be zero.

order must be 0–99.

```
RandSeed 1147 [ENTER]                    Done
randPoly(x,5) [ENTER]
              ⁻2•x^5+3•x^4−6•x^3+4•x−6
```

RandSeed MATH/Probability menu

RandSeed *number*

If *number* = 0, sets the seeds to the factory defaults for the random-number generator. If *number* ≠ 0, it is used to generate two seeds, which are stored in system variables seed1 and seed2.

```
RandSeed 1147 [ENTER]                    Done
rand() [ENTER]                          0.158...
```

RclGDB CATALOG

RclGDB *GDBvar*

Restores all the settings stored in the Graph database variable GDBvar.

For a listing of the settings, see **StoGDB** on page 444.

```
RclGDB GDBvar [ENTER]                    Done
```

RclPic CATALOG

RclPic *picVar* [, *row*, *column*]

Displays the Graph screen and adds the picture stored in *picVar* at the upper left-hand corner pixel coordinates (*row*, *column*) using OR logic.

picVar must be a picture data type.

Default coordinates are (0, 0).

real() MATH/Complex menu

real(*expression1***)** ⇒ *expression*

Returns the real part of the argument.

Note: All undefined variables are treated as real variables. See also **imag()** page (407).

```
real(2+3i) [ENTER]                          2
real(z) [ENTER]                             z
real(x+iy) [ENTER]                          x
```

---

real(*list1***)** ⇒ *list*

Returns the real parts of all elements.

```
real({a+i*b,3,i}) [ENTER]           {a  3  0}
```

---

real(*matrix1***)** ⇒ *matrix*

Returns the real parts of all elements.

```
real([a+i*b,3;c,i]) [ENTER]         [a  3]
                                    [c  0]
```

▶Rect — MATH/Matrix/Vector ops menu

vector **▶Rect**

Displays *vector* in rectangular form [x, y, z]. The vector must be of dimension 2 or 3 and can be a row or a column.

Note: **▶Rect** is a display-format instruction, not a conversion function. You can use it only at the end of an entry line, and it does not update ans.

Note: See also ▶Polar (page 425).

```
[3,∠π/4,∠π/6]▶Rect [ENTER]
```

$$\left[\frac{3\cdot\sqrt{2}}{4}\quad \frac{3\cdot\sqrt{2}}{4}\quad \frac{3\cdot\sqrt{3}}{2}\right]$$

```
[a,∠b,∠c] [ENTER]      [a·cos(b)·sin(c)
              a·sin(b)·sin(c)   a·cos(c)]
```

ref() — MATH/Matrix menu

ref(*matrix1***)** ⇒ *matrix*

Returns the row echelon form of *matrix1*.

Note: See also **rref()** (page 435).

```
ref([-2,-2,0,-6;1,-1,9,-9;-5,
2,4,-4]) [ENTER]
```

$$\begin{bmatrix} 1 & -2/5 & -4/5 & 4/5 \\ 0 & 1 & 4/7 & 11/7 \\ 0 & 0 & 1 & -62/71 \end{bmatrix}$$

remain() — MATH/Number menu

remain(*expression1*, *expression2***)** ⇒ *expression*
remain(*list1*, *list2***)** ⇒ *list*
remain(*matrix1*, *matrix2***)** ⇒ *matrix*

Returns the remainder of the first argument with respect to the second argument as defined by the identities:

remain(x,0) ≡ x
remain(x,y) ≡ x−y intDiv(x/y)

As a consequence, note that **remain(**−x,y**)** ≡ −**remain(**x,y**)**. The result is either zero or it has the same sign as the first argument.

Note: See also **mod()** (page 418).

```
remain(7,0) [ENTER]                       7
remain(7,3) [ENTER]                       1
remain(-7,3) [ENTER]                     -1
remain(7,-3) [ENTER]                      1
remain(-7,-3) [ENTER]                    -1
remain({12,-14,16},{9,7,-5}) [ENTER]
                                  {3  0  1}
remain([9,-7;6,4],[4,3;4,-3]) [ENTER]
```

$$\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

Rename — CATALOG

Rename *oldVarName*, *newVarName*

Renames the variable *oldVarName* as *newVarName*.

```
{1,2,3,4}→L1 [ENTER]             {1,2,3,4}
Rename L1, list1 [ENTER]              Done
list1 [ENTER]                    {1,2,3,4}
```

Request — CATALOG

Request *promptString*, *var*

If **Request** is inside a **Dialog**...**EndDlog** construct, it creates an input box for the user to type in data. If it is a stand-alone instruction, it creates a dialog box for this input. In either case, if *var* contains a string, it is displayed and highlighted in the input box as a default choice. *promptString* must be ≤ 20 characters.

This instruction can be stand-alone or part of a dialog construct.

```
Request "Enter Your Name",str1 [ENTER]
```

Return CATALOG

Return [*expression*]

Returns *expression* as the result of the function. Use within a **Func**:**EndFunc** block, or **Prgm**...**EndPrgm** block.

Note: Use **Return** without an argument to exit a program.

```
Define factoral(nn)=Func
  :local answer,count:1→answer
  :For count,1,nn
  :answer*count→answer:EndFor
  :Return answer:EndFunc [ENTER]    Done
factoral(3) [ENTER]                    6
```

right() MATH/List menu

right(*list1*[, *num*]**)** ⇒ *list*

Returns the rightmost *num* elements contained in *list1*.

If you omit *num*, returns all of *list1*.

```
right({1,3,-2,4},3) [ENTER]     {3 -2 4}
```

---

right(*sourceString*[, *num*]**)** ⇒ *string*

Returns the rightmost *num* characters contained in character string *sourceString*.

If you omit *num*, returns all of *sourceString*.

```
right("Hello",2) [ENTER]            "lo"
```

---

right(*comparison***)** ⇒ *expression*

Returns the right side of an equation or inequality.

```
right(x<3) [ENTER]                     3
```

round() MATH/Number menu

round(*expression1*[, *digits*]**)** ⇒ *expression*

Returns the argument rounded to the specified number of digits after the decimal point.

digits must be an integer in the range 0–12. If *digits* is not included, returns the argument rounded to 12 significant digits.

Note: Display digits mode may still affect how this is displayed.

```
round(1.234567,3) [ENTER]          1.235
```

---

round(*list1*[, *digits*]**)** ⇒ *list*

Returns a list of the elements rounded to the specified number of digits.

```
round({π,√(2),ln(2)},4) [ENTER]
                    {3.1416 1.4142 .6931}
```

---

round(*matrix1*[, *digits*]**)** ⇒ *matrix*

Returns a matrix of the elements rounded to the specified number of digits.

```
round([ln(5),ln(3);π,e^(1)],1) [ENTER]
                          [1.6  1.1]
                          [3.1  2.7]
```

rowAdd() MATH/Matrix/Row ops menu

rowAdd(*matrix1*, *rIndex1*, *rIndex2***)** ⇒ *matrix*

Returns a copy of *matrix1* with row *rIndex2* replaced by the sum of rows *rIndex1* and *rIndex2*.

```
rowAdd([3,4;-3,-2],1,2) [ENTER]
                              [3 4]
                              [0 2]

rowAdd([a,b;c,d],1,2) [ENTER]
                          [a    b  ]
                          [a+c  b+d]
```

rowDim() MATH/Matrix/Dimensions menu

rowDim(*matrix***)** ⇒ *expression*

Returns the number of rows in *matrix*.

Note: See also **colDim()** (page 382).

```
[1,2;3,4;5,6]→M1 [ENTER]        [1 2]
                                [3 4]
                                [5 6]
rowdim(M1) [ENTER]                  3
```

rowNorm() MATH/Matrix/Norms menu

rowNorm(*matrix***)** ⇒ *expression*

Returns the maximum of the sums of the absolute values of the elements in the rows in *matrix*.

Note: All matrix elements must simplify to numbers. See also **colNorm()** (page 382).

```
rowNorm([-5,6,-7;3,4,9;9,-9,-7])
[ENTER]                            25
```

rowSwap() MATH/Matrix/Row ops menu

rowSwap(*matrix1*, *rIndex1*, *rIndex2***)** ⇒ *matrix*

Returns *matrix1* with rows *rIndex1* and *rIndex2* exchanged.

```
[1,2;3,4;5,6]→Mat [ENTER]
                                [1 2]
                                [3 4]
                                [5 6]
rowSwap(Mat,1,3) [ENTER]
                                [5 6]
                                [3 4]
                                [1 2]
```

RplcPic CATALOG

RplcPic *picVar*[, *row*][, *column*]

Clears the Graph screen and places picture *picVar* at pixel coordinates (*row*, *column*). If you do not want to clear the screen, use **RclPic**.

picVar must be a picture data type variable. *row* and *column*, if included, specify the pixel coordinates of the upper left corner of the picture. Default coordinates are (0, 0).

Note: For less than full-screen pictures, only the area affected by the new picture is cleared.

rref() MATH/Matrix menu

rref(*matrix1***)** ⇒ *matrix*

Returns the reduced row echelon form of the matrix.

Note: See also **ref()** (page 433).

```
rref([-2,-2,0,-6;1,-1,9,-9;
  -5,2,4,-4]) [ENTER]
```

$$\begin{bmatrix} 1 & 0 & 0 & 66/71 \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & -62/71 \end{bmatrix}$$

```
rref([a,b,x;c,d,y]) [ENTER]
```

$$\begin{bmatrix} 1 & 0 & \frac{d \cdot x - b \cdot y}{a \cdot d - b \cdot c} \\ 0 & 1 & \frac{-(c \cdot x - a \cdot y)}{a \cdot d - b \cdot c} \end{bmatrix}$$

Send CATALOG

Send *list*

CBL 2/CBL (Calculator-Based Laboratory) or CBR (Calculator-Based Ranger) instruction. Sends *list* to the link port.

Program segment:

```
   ⋮
:Send {1,0}
:Send {1,2,1}
   ⋮
```

SendCalc CATALOG

SendCalc *var*

Sends variable *var* to the link port. This is for unit-to-unit linking.

Program segment:

```
   ⋮
:a+b→x
:SendCalc x
   ⋮
```

seq() MATH/List menu

seq(*expression*, *var*, *low*, *high*[, *step*]**)** ⇒ *list*

Increments *var* from *low* through *high* by an increment of *step*, evaluates *expression*, and returns the results as a list. The original contents of *var* are still there after **seq()** is completed.

var cannot be a system variable.

The default value for *step* = 1.

```
seq(n^2,n,1,6) [ENTER]   {1 4 9 16 25 36}
seq(1/n,n,1,10,2) [ENTER]
                          {1 1/3 1/5 1/7 1/9}
sum(seq(1/n^2,n,1,10,1)) [ENTER]   196.../127...
```

or press [♦] [ENTER] to get: 1.549...

setFold() CATALOG

setFold(*newfolderName***)** ⇒ *oldfolderString*

Returns the name of the current folder as a string and sets *newfolderName* as the current folder.

The folder *newfolderName* must exist.

```
newFold chris [ENTER]              Done
setFold(main) [ENTER]              "chris"
setFold(chris)→oldfoldr [ENTER]    "main"
1→a [ENTER]                        1
setFold(#oldfoldr) [ENTER]         "chris"
a [ENTER]                          a
chris\a [ENTER]                    1
```

setGraph() CATALOG

setGraph(*modeNameString*, *settingString***)** ⇒ *string*

Sets the Graph mode *modeNameString* to *settingString*, and returns the previous setting of the mode. Storing the previous setting lets you restore it later.

modeNameString is a character string that specifies which mode you want to set. It must be one of the mode names from the table below.

settingString is a character string that specifies the new setting for the mode. It must be one of the settings listed below for the specific mode you are setting.

```
setGraph("Graph Order","Seq")
[ENTER]                          "SEQ"

setGraph("Coordinates","Off")
[ENTER]                          "RECT"
```

Note: Capitalization and blank spaces are optional when entering mode names.

Mode Name	Settings	
"Coordinates"	"Rect", "Polar", "Off"	
"Graph Order"	"Seq", "Simul" [1]	
"Grid"	"Off", "On" [2]	
"Axes"	"Off", "On" "Box", "Axes", "Off"	(not 3D graph mode) (3D graph mode)
"Leading Cursor"	"Off", "On" [2]	
"Labels"	"Off", "On"	
"Style"	"Wire Frame", "Hidden Surface" [3]	
"Seq Axes"	"Time", "Web", "U1-vs-U2" [4]	

[1]Not available in Sequence or 3D graph mode.
[2]Not available in 3D graph mode.
[3]Applies only to 3D graph mode.
[4]Applies only to Sequence graph mode.

setMode() CATALOG

setMode(*modeNameString*, *settingString***)** ⇒ *string*
setMode(*list***)** ⇒ *stringList*

Sets mode *modeNameString* to the new setting *settingString*, and returns the current setting of that mode.

modeNameString is a character string that specifies which mode you want to set. It must be one of the mode names from the table below.

settingString is a character string that specifies the new setting for the mode. It must be one of the settings listed below for the specific mode you are setting.

list contains pairs of keyword strings and will set them all at once. This is recommended for multiple-mode changes. The example shown may not work if each of the pairs is entered with a separate **setMode()** in the order shown.

Use **setMode(***var***)** to restore settings saved with **getMode("ALL")→***var*.

Note: See **getMode** (page 404).

```
setMode("Angle","Degree")
[ENTER]                          "RADIAN"
sin(45) [ENTER]                     √2/2

setMode("Angle","Radian")
[ENTER]                          "DEGREE"
sin(π/4) [ENTER]                    √2/2

setMode("Display Digits",
  "Fix 2") [ENTER]                "FLOAT"
π [◆] [ENTER]                        3.14

setMode ("Display Digits",
  "Float") [ENTER]                "FIX 2"
π [◆] [ENTER]                     3.141...

setMode ({"Split Screen",
  "Left-Right","Split 1 App",
  "Graph","Split 2 App","Table"})
  [ENTER]
              {"Split 2 App" "Graph"
                "Split 1 App" "Home"
              "Split Screen" "FULL"}
```

Note: Capitalization and blank spaces are optional when entering mode names. Also, the results in these examples may be different on your TI-92.

Mode Name	Settings
"Graph"	"Function", "Parametric", "Polar", "Sequence", "3D"
"Display Digits"	"Fix 0", "Fix 1", ..., "Fix 12", "Float", "Float 1", ..., "Float 12"
"Angle"	"Radian", "Degree"
"Exponential Format"	"Normal", "Scientific", "Engineering"
"Complex Format"	"Real", "Rectangular", "Polar"
"Vector Format"	"Rectangular", "Cylindrical", "Spherical"
"Pretty Print"	"Off", "On"
"Split Screen"	"Full", "Top-Bottom", "Left-Right"
"Split 1 App"	"Home", "Y= Editor", "Window Editor", "Graph", "Table", "Data/Matrix Editor", "Program Editor", "Geometry", "Text Editor"
"Split 2 App"	"Home", "Y= Editor", "Window Editor", "Graph", "Table", "Data/Matrix Editor", "Program Editor", "Geometry", "Text Editor"
"Number of Graphs"	"1", "2"
"Graph2"	"Function", "Parametric", "Polar", "Sequence", "3D"
"Split Screen Ratio"	"1:1", "1:2", "2:1"
"Exact/Approx"	"Auto", "Exact", "Approximate"

setTable() CATALOG

setTable(*modenameString*, *settingString***)** ⇒ *string*

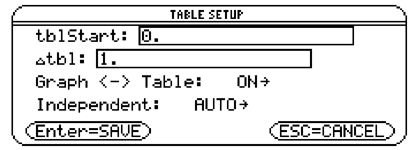
Sets the table parameter *modeNameString* to *settingString*, and returns the previous setting of the parameter. Storing the previous setting lets you restore it later.

modeNameString is a character string that specifies which parameter you want to set. It must be one of the parameters from the table below.

settingString is a character string that specifies the new setting for the parameter. It must be one of the settings listed below for the specific parameter you are setting.

```
setTable("Graph <-> Table","ON")
[ENTER]                                "OFF"

setTable("Independent","AUTO")
[ENTER]                                "ASK"
```

[♦] [TblSet]



Note: Capitalization and blank spaces are optional when entering parameters.

Parameter Name	**Settings**
“Graph <-> Table”	“Off”, On”
“Independent”	“Auto”, “Ask”

Shade CATALOG

Shade *expr1*, *expr2*, [*xlow*], [*xhigh*], [*pattern*], [*patRes*]

Displays the Graph screen, graphs *expr1* and *expr2*, and shades areas in which *expr1* is less than *expr2*. (*expr1* and *expr2* must be expressions that use x as the independent variable.)

xlow and *xhigh*, if included, specify left and right boundaries for the shading. Valid inputs are between xmin and xmax. Defaults are xmin and xmax.

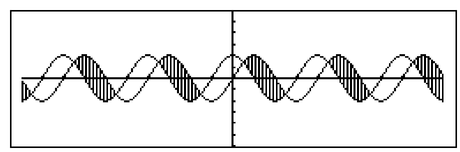
pattern specifies one of four shading patterns:
1 = vertical (default)
2 = horizontal
3 = negative-slope 45°
4 = positive-slope 45°

patRes specifies the resolution of the shading patterns:
1= solid shading
2= 1 pixel spacing (default)
3= 2 pixels spacing
⋮
10= 9 pixels spacing

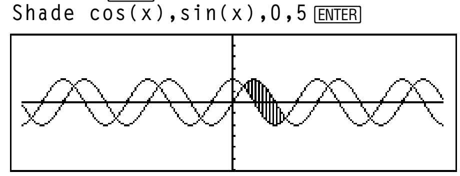
Note: Interactive shading is available on the Graph screen through the **Shade** instruction. Automatic shading of a specific function is available through the **Style** instruction (page 445). **Shade** is not valid in 3D graphing mode.

In the ZoomTrig viewing window:

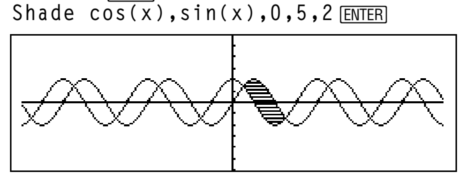
```
Shade cos(x),sin(x) [ENTER]
```



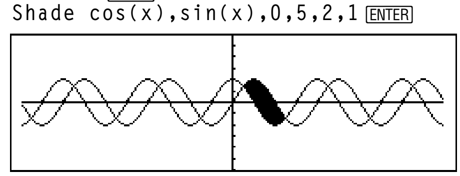
```
[♦][HOME]
ClrDraw [ENTER]                        Done
Shade cos(x),sin(x),0,5 [ENTER]
```



```
[♦][HOME]
ClrDraw [ENTER]                        Done
Shade cos(x),sin(x),0,5,2 [ENTER]
```



```
[♦][HOME]
ClrDraw [ENTER]                        Done
Shade cos(x),sin(x),0,5,2,1 [ENTER]
```



shift() CATALOG

shift(*list1*[, *integer*]**)** ⇒ *list*

Returns a copy of *list1* shifted right or left by *integer* elements. Does not alter *list1*.

If *integer* is positive, the shift is to the left. If *integer* is negative, the shift is to the right. The default for *integer* is ⁻1 (shift right one element).

Elements introduced at the beginning or end of *list* by the shift are set to the symbol "undef."

```
shift({1,2,3,4},1) [ENTER]
                        {2 3 4 undef}

shift({1,2,3,4},2) [ENTER]
                    {3 4 undef undef}
```

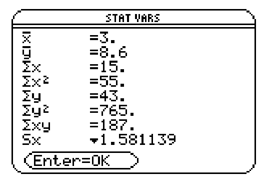
ShowStat CATALOG

ShowStat

Displays a dialog box containing the last computed statistics results if they are still valid. Statistics results are cleared automatically if the data to compute them has changed.

Use this instruction after a statistics calculation, such as **LinReg**.

```
{1,2,3,4,5}→L1 [ENTER]      {1 2 3 4 5}
{0,2,6,10,25}→L2 [ENTER]  {0 2 6 10 25}
TwoVar L1,L2 [ENTER]
ShowStat [ENTER]
```



sign() MATH/Number menu

sign(*expression1***)** ⇒ *expression*
sign(*list1***)** ⇒ *list*
sign(*matrix1***)** ⇒ *matrix*

For real and complex *expression1*, returns *expression1*/**abs(***expression1***)** when *expression1*≠ 0.

Returns 1 if *expression1* is positive.
Returns ⁻1 if *expression1* is negative.
sign(0**)** returns itself as the result.
sign(0**)** represents ±1 in the real domain.
sign(0**)** represents the unit circle in the complex domain.

For a list or matrix, returns the signs of all the elements.

```
sign(⁻3.2) [ENTER]                      ⁻1.
sign({2,3,4,⁻5}) [ENTER]      {1  1  1 ⁻1}
sign([⁻3,0,3]) [ENTER]     [⁻1 sign(0) 1]
sign(1+abs(x)) [ENTER]                   1
```

simult() MATH/Matrix menu

simult(*matrixExpr*, *vectorExpr***)** ⇒ *matrix*

Returns a column vector that contains the solutions to a system of linear equations.

matrixExpr must be a square matrix and consists of the coefficients of the equation.

vectorExpr must have the same number of rows (same dimension) as *matrixExpr* and contain the constants.

```
simult([1,2;3,4],[1;-1]) [ENTER]
```

$$\begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

```
[a,b;c,d]→matx1 [ENTER]
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
simult(matx1,[1;2]) [ENTER]
```

$$\begin{bmatrix} \frac{-(2\cdot b-d)}{a\cdot d-b\cdot c} \\ \frac{2\cdot a-c}{a\cdot d-b\cdot c} \end{bmatrix}$$

sin() — [SIN] key

sin(*expression1***)** ⇒ *expression*
sin(*list1***)** ⇒ *list*

sin(*expression1***)** returns the sine of the argument as an expression.

sin(*list1***)** returns a list of the sines of all elements in *list1*.

Note: The argument is interpreted as either a degree or radian angle, according to the current angle mode. You can use ° (page 467) or ʳ (page 467) to override the angle mode setting temporarily.

In Degree angle mode:

`sin((π/4)ʳ)` [ENTER] $\frac{\sqrt{2}}{2}$

`sin(45)` [ENTER] $\frac{\sqrt{2}}{2}$

`sin({0,60,90})` [ENTER] $\{0 \quad \frac{\sqrt{3}}{2} \quad 1\}$

In Radian angle mode:

`sin(π/4)` [ENTER] $\frac{\sqrt{2}}{2}$

`sin(45°)` [ENTER] $\frac{\sqrt{2}}{2}$

sin⁻¹() — [2nd] [SIN⁻¹] key

sin⁻¹(*expression1***)** ⇒ *expression*
sin⁻¹(*list1***)** ⇒ *list*

sin⁻¹ (*expression1***)** returns the angle whose sine is *expression1* as an expression.

sin⁻¹ (*list1***)** returns a list of the inverse sines of each element of *list1*.

Note: The result is returned as either a degree or radian angle, according to the current angle mode setting.

In Degree angle mode:

`sin⁻¹(1)` [ENTER] `90`

In Radian angle mode:

`sin⁻¹({0,.2,.5})` [ENTER]
`{0 .201... .523...}`

sinh() — MATH/Hyperbolic menu

sinh(*expression1***)** ⇒ *expression*
sinh(*list1***)** ⇒ *list*

sinh (*expression1***)** returns the hyperbolic sine of the argument as an expression.

sinh (*list***)** returns a list of the hyperbolic sines of each element of *list1*.

`sinh(1.2)` [ENTER] `1.509...`

`sinh({0,1.2,3.})` [ENTER]
`{0 1.509... 10.017...}`

sinh⁻¹() — MATH/Hyperbolic menu

sinh⁻¹(*expression1***)** ⇒ *expression*
sinh⁻¹(*list1***)** ⇒ *list*

sinh⁻¹ (*expression1***)** returns the inverse hyperbolic sine of the argument as an expression.

sinh⁻¹ (*list1***)** returns a list of the inverse hyperbolic sines of each element of *list1*.

`sinh⁻¹(0)` [ENTER] `0`

`sinh⁻¹({0,2.1,3})` [ENTER]
`{0 1.487... sinh⁻¹(3)}`

solve() MATH/Algebra menu

solve(*equation*, *var***)** ⇒ *Boolean expression*
solve(*inequality*, *var***)** ⇒ *Boolean expression*

Returns candidate real solutions of an equation or an inequality for *var*. The goal is to return candidates for all solutions. However, there might be equations or inequalities for which the number of solutions is infinite.

```
solve(a*x^2+b*x+c=0,x) [ENTER]
```

$$x = \frac{\sqrt{-(4\cdot a\cdot c-b^2)}-b}{2\cdot a}$$

$$\text{or } x = \frac{-(\sqrt{-(4\cdot a\cdot c-b^2)}+b)}{2\cdot a}$$

Solution candidates might not be real finite solutions for some combinations of values for undefined variables.

```
ans(1)| a=1 and b=1 and c=1 [ENTER]
                    Error: Non-real result
```

For the AUTO setting of the Exact/Approx mode, the goal is to produce exact solutions when they are concise, and supplemented by iterative searches with approximate arithmetic when exact solutions are impractical.

```
solve((x-a)e^(x)=-x*(x-a),x) [ENTER]
                         x = a or x = -.567…
```

Due to default cancellation of the greatest common divisor from the numerator and denominator of ratios, solutions might be solutions only in the limit from one or both sides.

```
(x+1)(x-1)/(x-1)+x-3 [ENTER]        2·x-2
solve(entry(1)=0,x) [ENTER]          x = 1
entry(2)|ans(1) [ENTER]              undef
limit(entry(3),x,1) [ENTER]              0
```

For inequalities of types ≥, ≤, <, or >, explicit solutions are unlikely unless the inequality is linear and contains only *var*.

```
solve(5x-2 ≥ 2x,x) [ENTER]          x ≥ 2/3
```

For the EXACT setting of the Exact/Approx mode, portions that cannot be solved are returned as an implicit equation or inequality.

```
exact(solve((x-a)e^(x)=-x*
  (x-a),x)) [ENTER]
                         e^x + x = 0 or x = a
```

Use the "|" operator to restrict the solution interval and/or other variables that occur in the equation or inequality. When you find a solution in one interval, you can use the inequality operators to exclude that interval from subsequent searches.

In Radian angle mode:

```
solve(tan(x)=1/x,x)|x>0 and x<1
[ENTER]                          x =.860…
```

false is returned when no real solutions are found. true is returned if **solve()** can determine that any finite real value of *var* satisfies the equation or inequality.

```
solve(x=x+1,x) [ENTER]              false
solve(x=x,x) [ENTER]                 true
```

Since **solve()** always returns a Boolean result, you can use "and," "or," and "not" to combine results from **solve()** with each other or with other Boolean expressions.

```
2x-1≤1 and solve(x^2≠9,x) [ENTER]
                       x ≤ 1 and x ≠ -3
```

Solutions might contain a unique new undefined variable of the form @nj with j being an integer in the interval 1–255. Such variables designate an arbitrary integer.

In Radian angle mode:

```
solve(sin(x)=0,x) [ENTER]         x = @n1·π
```

In real mode, fractional powers having odd denominators denote only the real branch. Otherwise, multiple branched expressions such as fractional powers, logarithms, and inverse trigonometric functions denote only the principal branch. Consequently, **solve()** produces only solutions corresponding to that one real or principal branch.

```
solve(x^(1/3)=-1,x) [ENTER]          x = -1
solve(√(x)=-2,x) [ENTER]             false
solve(-√(x)=-2,x) [ENTER]             x = 4
```

Note: See also **cSolve()** (page 385), **cZeros()** (page 387), **nSolve()** (page 422), and **zeros()** (page 453).

SortA — MATH/List menu

SortA *listName1*[, *listName2*] [, *listName3*] ...
SortA *vectorName1*[, *vectorName2*] [, *vectorName3*] ...

Sorts the elements of the first argument in ascending order.

If you include additional arguments, sorts the elements of each so that their new positions match the new positions of the elements in the first argument.

All arguments must be names of lists or vectors. All arguments must have equal dimensions.

```
{2,1,4,3}→list1 [ENTER]          {2,1,4,3}
SortA list1 [ENTER]                   Done
list1 [ENTER]                    {1 2 3 4}
{4,3,2,1}→list2 [ENTER]          {4 3 2 1}
SortA list2,list1 [ENTER]             Done
list2 [ENTER]                    {1 2 3 4}
list1 [ENTER]                    {4 3 2 1}
```

SortD — MATH/List menu

SortD *listName1*[, *listName2*] [, *listName3*] ...
SortD *vectorName1*[,*vectorName 2*] [,*vectorName 3*] ...

Identical to **SortA**, except **SortD** sorts the elements in descending order.

```
{2,1,4,3}→list1 [ENTER]          {2 1 4 3}
{1,2,3,4}→list2 [ENTER]          {1 2 3 4}
SortD list1,list2 [ENTER]             Done
list1 [ENTER]                    {4 3 2 1}
list2 [ENTER]                    {3 4 1 2}
```

▶Sphere — MATH/Matrix/Vector ops menu

vector **▶Sphere**

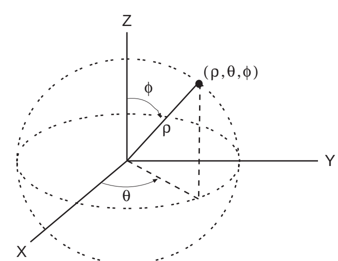
Displays the row or column vector in spherical form [ρ ∠θ ∠φ].

vector must be of dimension 3 and can be either a row or a column vector.

Note: **▶Sphere** is a display-format instruction, not a conversion function. You can use it only at the end of an entry line.

```
[1,2,3]▶Sphere
[◆] [ENTER]      [3.741... ∠1.107... ∠.640...]

[2,∠π/4,3]▶Sphere
[◆] [ENTER]      [3.605... ∠.785... ∠.588...]
```

[ENTER] $\left[\sqrt{13}\quad \angle\frac{\pi}{4}\quad \angle{-\sin^{-1}}\left(\frac{3\cdot\sqrt{13}}{13}\right)+\frac{\pi}{2}\right]$



stdDev() — MATH/Statistics menu

stdDev(*list***)** ⇒ *expression*

Returns the standard deviation of the elements in *list*.

Note: *list* must have at least two elements.

```
stdDev({a,b,c}) [ENTER]
stdDev({1,2,5,-6,3,-2}) [ENTER]
```

■ stdDev({a b c})
$\frac{\sqrt{a^2-a\cdot(b+c)+b^2-b\cdot c+c^2}}{\sqrt{3}}$

■ stdDev({1 2 5 -6 3 -2}) $\frac{\sqrt{62}}{2}$

---

stdDev(*matrix1***)** ⇒ *matrix*

Returns a row vector of the standard deviations of the columns in *matrix1*.

Note: *matrix1* must have at least two rows.

```
stdDev([1,2,5;-3,0,1;.5,.7,3]) [ENTER]
                     [2.179... 1.014... 2]
```

StoGDB CATALOG

StoGDB *GDBvar*

Creates a Graph database (GDB) variable that contains the current:

* Graphing mode
* Y= functions
* Window variables
* Graph format settings
  1- or 2-Graph setting (split screen and ratio settings if 2-Graph mode)
  Angle mode
  Real/complex mode
* Initial conditions if Sequence mode
* Table flags
* tblStart, Δtbl, tblInput

You can use **RclGDB** *GDBvar* to restore the graph environment.

***Note:** These items are saved for both graphs in 2-Graph mode.

Stop CATALOG

Stop

Used as a program instruction to stop program execution.

Program segment:

```
   :
For i,1,10,1
  If i=5
  Stop
EndFor
   :
```

StoPic CATALOG

StoPic *picVar* [, *pxlRow*, *pxlCol*] [, *width*, *height*]

Displays the graph screen and copies a rectangular area of the display to the variable *picVar*.

pxlRow and *pxlCol*, if included, specify the upper-left corner of the area to copy (defaults are 0, 0).

width and *height*, if included, specify the dimensions, in pixels, of the area. Defaults are the width and height, in pixels, of the current graph screen.

Store See →, page 469.

string() MATH/String menu

string(*expression***)** ⇒ *string*

Simplifies *expression* and returns the result as a character string.

```
string(1.2345) [ENTER]          "1.2345"
string(1+2) [ENTER]                  "3"
string(cos(x)+√(3)) [ENTER]
                            "cos(x) + √(3)"
```

Style CATALOG

Style *equanum*, *stylePropertyString*

Sets the system graphing function *equanum* in the current graph mode to use the graphing property *stylePropertyString*.

equanum must be an integer from 1–99 and must already exist.

stylePropertyString must be one of: "Line," "Dot," "Square," "Thick," "Animate," "Path," "Above," or "Below."

Note that in parametric graphing, only the *xt* half of the pair contains the style information.

Valid style names vs. graphing mode:

Function:	all styles
Parametric/Polar:	line, dot, square, thick, animate, path
Sequence:	line, dot, square, thick
3D:	none

Note: Capitalization and blank spaces are optional when entering *stylePropertyString* names.

```
Style 1,"thick" [ENTER]            Done
Style 10,"path" [ENTER]            Done
```

Note: In function graphing mode, these examples set the style of y1(x) to "Thick" and y10(x) to "Path."

subMat() CATALOG

subMat(*matrix1*[, *startRow*] [, *startCol*] [, *endRow*] [, *endCol*]**)** ⇒ *matrix*

Returns the specified submatrix of *matrix1*.

Defaults: *startRow*=1, *startCol*=1, *endRow*=last row, *endCol*=last column.

```
[1,2,3;4,5,6;7,8,9]→m1 [ENTER]
                                  [1 2 3]
                                  [4 5 6]
                                  [7 8 9]
subMat(m1,2,1,3,2) [ENTER]
                                  [4 5]
                                  [7 8]
subMat(m1,2,2) [ENTER]
                                  [5 6]
                                  [8 9]
```

sum() MATH/List menu

sum(*list***)** ⇒ *expression*

Returns the sum of the elements in *list*.

```
sum({1,2,3,4,5}) [ENTER]           15
sum({a,2a,3a}) [ENTER]             6·a
sum(seq(n,n,1,10)) [ENTER]         55
```

---

sum(*matrix1***)** ⇒ *matrix*

Returns a row vector containing the sums of the elements in the columns in *matrix1*.

```
sum([1,2,3;4,5,6]) [ENTER]     [5 7 9]
sum([1,2,3;4,5,6;7,8,9]) [ENTER]
                               [12 15 18]
```

switch() CATALOG

switch([*integer1*]**)** ⇒ *integer*

Returns the number of the active window. Also can set the active window.

Note: Window 1 is left or top; Window 2 is right or bottom.

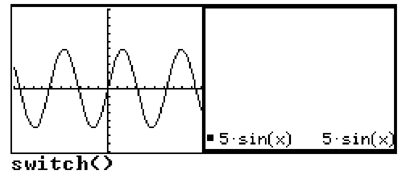
If *integer1* = 0, returns the active window number.

If *integer1* = 1, activates window 1 and returns the previously active window number.

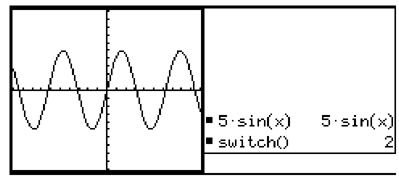
If *integer1* = 2, activates window 2 and returns the previously active window number.

If *integer1* is omitted, switches windows and returns the previously active window number.

integer1 is ignored if the TI-92 is not displaying a split screen.



`switch` [ENTER]



T (transpose) MATH/Matrix menu

matrix1^T ⇒ *matrix*

Returns the complex conjugate transpose of *matrix1*.

`[1,2,3;4,5,6;7,8,9]→mat1` [ENTER]

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$$

`mat1`^{T} [ENTER]

$$\begin{bmatrix}1 & 4 & 7\\2 & 5 & 8\\3 & 6 & 9\end{bmatrix}$$

`[a,b;c,d]→mat2` [ENTER] $\begin{bmatrix}a & b\\c & d\end{bmatrix}$

`mat2`^{T} [ENTER] $\begin{bmatrix}a & c\\b & d\end{bmatrix}$

`[1+`i`,2+`i`;3+`i`,4+`i`]→mat3` [ENTER]

$$\begin{bmatrix}1+i & 2+i\\3+i & 4+i\end{bmatrix}$$

`mat3`^{T} [ENTER] $\begin{bmatrix}1-i & 3-i\\2-i & 4-i\end{bmatrix}$

Table — CATALOG

Table *expression1*[, *expression2*] [, *var1*]

Builds a table of the specified expressions or functions.

The expressions in the table can also be graphed. Expressions entered using the **Table** or **Graph** (page 406) commands are assigned increasing function numbers starting with 1. The expressions can be modified or individually deleted using the edit functions available when the table is displayed by pressing [F4] Header. The currently selected functions in the Y= Editor are temporarily ignored.

To clear the functions created by **Table** or **Graph**, execute the **ClrGraph** command or display the Y= Editor.

If the *var* parameter is omitted, the current graph-mode independent variable is assumed. Some valid variations of this instruction are:

Function graphing: **Table** *expr*, *x*

Parametric graphing: **Table** *xExpr*, *yExpr*, *t*

Polar graphing: **Table** *expr*, θ

Note: The **Table** command is not valid for 3D or sequence graphing.

In function graphing mode.

`Table 1.25x*cos(x)` [ENTER]

x	1				
0.	0.				
1.	.67538				
2.	-1.04				
3.	-3.712				
4.	-3.268				
5.	1.7729				
6.	7.2013				
7.	6.5966				

`Table cos(time),time` [ENTER]

x	1	2			
0.	0.	1.			
1.	.67538	.5403			
2.	-1.04	-.4161			
3.	-3.712	-.99			
4.	-3.268	-.6536			
5.	1.7729	.28366			
6.	7.2013	.96017			
7.	6.5966	.7539			

tan() — [TAN] key

tan(*expression1***)** ⇒ *expression*
tan(*list1***)** ⇒ *list*

tan(*expression1***)** returns the tangent of the argument as an expression.

tan(*list1***)** returns a list of the tangents of all elements in *list1*.

Note: The argument is interpreted as either a degree or radian angle, according to the current angle mode. You can use ° (page 467) or ʳ (page 467) to override the angle mode temporarily.

In Degree angle mode:

`tan((π/4)ʳ)` [ENTER] 1

`tan(45)` [ENTER] 1

`tan({0,60,90})` [ENTER] {0 √3 undef}

In Radian angle mode:

`tan(π/4)` [ENTER] 1

`tan(45°)` [ENTER] 1

`tan({π,π/3,-π,π/4})` [ENTER] {0 √3 0 1}

tan^{-1}() — [2nd] [TAN⁻¹] key

tan^{-1}(*expression1***)** ⇒ *expression*
tan^{-1}(*list1***)** ⇒ *list*

tan^{-1} (*expression1***)** returns the angle whose tangent is *expression1* as an expression.

tan^{-1} (*list1***)** returns a list of the inverse tangents of each element of *list1*.

Note: The result is returned as either a degree or radian angle, according to the current angle mode setting.

In Degree angle mode:

`tan⁻¹(1)` [ENTER] 45

In Radian angle mode:

`tan⁻¹({0,.2,.5})` [ENTER]
{0 .197... .463...}

tanh() MATH/Hyperbolic menu

tanh(*expression1***)** ⇒ *expression*
tanh(*list1***)** ⇒ *list*

tanh(*expression1***)** returns the hyperbolic tangent of the argument as an expression.

tanh(*list***)** returns a list of the hyperbolic tangents of each element of *list1*.

```
tanh(1.2) [ENTER]                .833...
```

tanh({0,1}) [ENTER] $\left\{0 \quad \frac{e^2-1}{e^2+1}\right\}$

tanh⁻¹() MATH/Hyperbolic menu

tanh⁻¹(*expression1***)** ⇒ *expression*
tanh⁻¹(*list1***)** ⇒ *list*

tanh⁻¹(*expression1***)** returns the inverse hyperbolic tangent of the argument as an expression.

tanh⁻¹(*list1***)** returns a list of the inverse hyperbolic tangents of each element of *list1*.

In rectangular complex format mode:

```
tanh⁻¹(0) [ENTER]                0
tanh⁻¹({1,2.1,3}) [ENTER]
     {∞   .518... -1.570...•i  tanh⁻¹(3)}
```

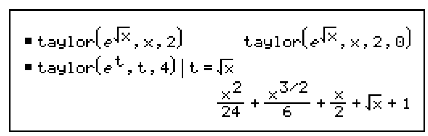
taylor() MATH/Calculus menu

taylor(*expression1*, *var*, *order*[, *point*]**)** ⇒ *expression*

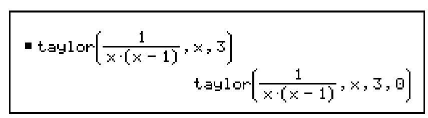
Returns the requested Taylor polynomial. The polynomial includes non-zero terms of integer degrees from zero through *order* in (*var* minus *point*). **taylor()** returns itself if there is no truncated power series of this order, or if it would require negative or fractional exponents. Use substitution and/or temporary multiplication by a power of (*var* minus *point*) to determine more general power series.

point defaults to zero and is the expansion point.

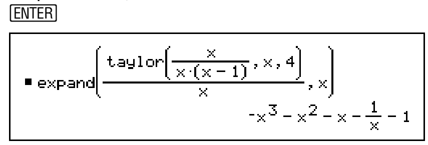
```
taylor(e^(√(x)),x,2) [ENTER]
taylor(e^(t),t,4)|t=√(x) [ENTER]
```



```
taylor(1/(x*(x-1)),x,3) [ENTER]
```



```
expand(taylor(x/(x*(x-1)),x,4)/x,x)
[ENTER]
```



tCollect() MATH\Algebra\Trig menu

tCollect(*expression1***)** ⇒ *expression*

Returns an expression in which products and integer powers of sines and cosines are converted to a linear combination of sines and cosines of multiple angles, angle sums, and angle differences. The transformation converts trigonometric polynomials into a linear combination of their harmonics.

Sometimes **tCollect()** will accomplish your goals when the default trigonometric simplification does not. **tCollect()** tends to reverse transformations done by **tExpand()**. Sometimes applying **tExpand()** to a result from **tCollect()**, or vice versa, in two separate steps simplifies an expression.

tCollect((cos(α))^2) [ENTER] $\frac{\cos(2\cdot\alpha)+1}{2}$

tCollect(sin(α)cos(β)) [ENTER] $\frac{\sin(\alpha-\beta)+\sin(\alpha+\beta)}{2}$

tExpand() MATH\Algebra\Trig menu

tExpand(*expression1***)** ⇒ *expression*

Returns an expression in which sines and cosines of integer-multiple angles, angle sums, and angle differences are expanded. Because of the identity $(\sin(x))^2+(\cos(x))^2=1$, there are many possible equivalent results. Consequently, a result might differ from a result shown in other publications.

Sometimes **tExpand()** will accomplish your goals when the default trigonometric simplification does not. **tExpand()** tends to reverse transformations done by **tCollect()**. Sometimes applying **tCollect()** to a result from **tExpand()**, or vice versa, in two separate steps simplifies an expression.

Note: Degree-mode scaling by π/180 interferes with the ability of **tExpand()** to recognize expandable forms. For best results, **tExpand()** should be used in Radian mode.

```
tExpand(sin(3φ)) [ENTER]
          4·sin(φ)·(cos(φ))²−sin(φ)

tExpand(cos(α−β)) [ENTER]
        cos(α)·cos(β)+sin(α)·sin(β)
```

Text CATALOG

Text *promptString*

Displays the character string *promptString* dialog box.

If used as part of a **Dialog:...EndDlog block**, *promptString* is displayed inside that dialog box. If used as a standalone instruction, **Text** creates a dialog box to display the string.

```
Text "Have a nice day." [ENTER]     Done
```

Then See **If,** page 407.

Title CATALOG

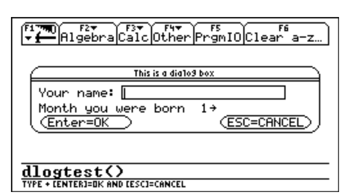
Title *titleString*, [*Lbl*]

Creates the title of a pull-down menu or dialog box when used inside a **Toolbar** or **Custom** construct, or a **Dialog...EndDlog** block.

Note: *Lbl* is only valid in the **Toolbar** construct. When present, it allows the menu choice to branch to a specified label inside the program.

Program segment:

```
  :
:Dialog
:Title      "This is a dialog box"
:Request    "Your name",Str1
:Dropdown   "Month you were born",
  seq(string(i),i,1,12),Var1
:EndDlog
  :
```



Toolbar CATALOG

Toolbar
block
EndTBar

Creates a toolbar menu.

block can be either a single statement or a sequence of statements separated with the ":" character. The statements can be either Title or Item.

Items must have labels. A Title must also have a label if it does not have an item.

Program segment:

```
   :
:Toolbar
:  Title "Examples"
:  Item "Trig", t
:  Item "Calc", c
:  Item "Stop", Pexit
:EndTbar
   :
```

Note: When run in a program, this segment creates a menu with three choices that branch to three places in the program.

Trace CATALOG

Trace

Draws a Smart Graph and places the trace cursor on the first defined Y= function at the previously defined cursor position, or at the reset position if regraphing was necessary.

Allows operation of the cursor and most keys when editing coordinate values. Several keys, such as the function keys, [APPS], and [MODE], are not activated during trace.

Note: Press [ENTER] to resume operation.

Try CATALOG

Try
block1
Else
block2
EndTry

Executes *block1* unless an error occurs. Program execution transfers to *block2* if an error occurs in *block1*. Variable errornum contains the error number to allow the program to perform error recovery.

block1 and *block2* can be either a single statement or a series of statements separated with the ":" character.

Program segment:

```
   :
:Try
:  NewFold(temp)
:  Else
:  ©Already exists
:  ClrErr
:EndTry
   :
```

Note: See **ClrErr** (page 381) and **PassErr** (page 424).

TwoVar — MATH/Statistics menu

TwoVar *list1*, *list2*[, [*list3*] [, *list4*, *list5*]]

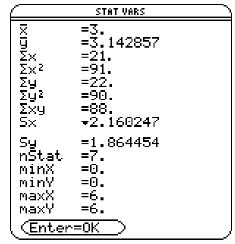
Calculates the **TwoVar** statistics and updates all the system statistics variables.

All the lists must have equal dimensions except for *list5*.

list1 represents xlist.
list2 represents ylist.
list3 represents frequency.
list4 represents category codes.
list5 represents category include list.

Note: *list1* through *list4* must be a variable name or c1–c99 (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be c1–c99.

```
{0,1,2,3,4,5,6}→L1 [ENTER]     {0 1 2 …}
{0,2,3,4,3,4,6}→L2 [ENTER]     {0 2 3 …}
TwoVar L1,L2 [ENTER]                Done
ShowStat [ENTER]
```



unitV() — MATH/Matrix/Vector ops menu

unitV(*vector1***)** ⇒ *vector*

Returns either a row- or column-unit vector, depending on the form of *vector1*.

vector1 must be either a single-row matrix or a single-column matrix.

unitV([a,b,c]) [ENTER]

$$\left[\frac{a}{\sqrt{a^2+b^2+c^2}} \quad \frac{b}{\sqrt{a^2+b^2+c^2}} \quad \frac{c}{\sqrt{a^2+b^2+c^2}}\right]$$

unitV([1,2,1]) [ENTER]

$$\left[\frac{\sqrt{6}}{6} \quad \frac{\sqrt{6}}{3} \quad \frac{\sqrt{6}}{6}\right]$$

unitV([1;2;3]) [ENTER]

$$\begin{bmatrix}\frac{\sqrt{14}}{14}\\ \frac{\sqrt{14}}{7}\\ \frac{3\cdot\sqrt{14}}{14}\end{bmatrix}$$

Unlock — CATALOG

Unlock *var1*[, *var2*][, *var3*]...

Unlocks the specified variables.

Note: The variables can be locked using the **Lock** command (page 414).

variance() — MATH/Statistics menu

variance(*list***)** ⇒ *expression*

Returns the variance of *list*.

Note: *list* must contain at least two elements.

variance({a,b,c}) [ENTER]

$$\frac{a^2-a\cdot(b+c)+b^2-b\cdot c+c^2}{3}$$

variance({1,2,5,⁻6,3,⁻2}) [ENTER] 31/2

---

variance(*matrix1***)** ⇒ *matrix*

Returns a row vector containing the variance of each column in *matrix1*.

Note: *matrix1* must contain at least two rows.

```
variance([1,2,5;⁻3,0,1;.5,.7,3])
[ENTER]                  [4.75  1.03  4]
```

when() CATALOG

when(_condition_, _trueResult_ [, _falseResult_] [, _unknownResult_]**)** ⇒ _expression_

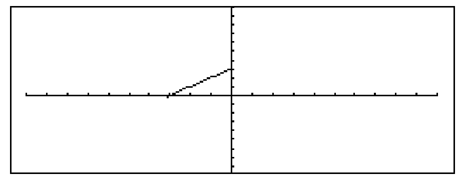
Returns _trueResult_, _falseResult_, or _unknownResult_, depending on whether _condition_ is true, false, or unknown. Returns the input if there are too few arguments to specify the appropriate result.

Omit both _falseResult_ and _unknownResult_ to make an expression defined only in the region where _condition_ is true.

```
when(x<0,x+3)|x=5 [ENTER]
                       when(x<0,3+x)
```

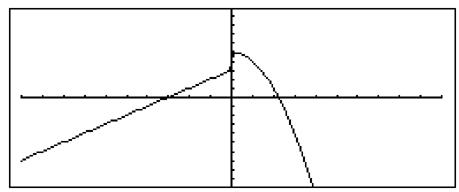
Use an undef _falseResult_ to define an expression that graphs only on an interval.

```
ClrGraph [ENTER]
Graph when(x≥⁻π and x<0,x+3,undef)
[ENTER]
```



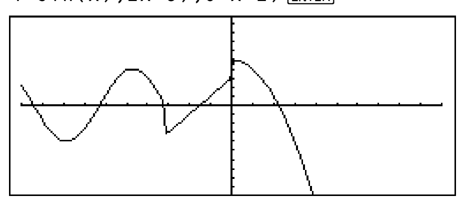
Omit only the _unknownResult_ to define a two-piece expression.

```
Graph when(x<0,x+3,5-x^2) [ENTER]
```



Nest **when()** to define expressions that have more than two pieces.

```
[♦] [HOME]
ClrGraph [ENTER]                    Done
Graph when(x<0,when(x<⁻π,
4*sin(x),2x+3),5-x^2) [ENTER]
```



when() is helpful for defining recursive functions.

```
when(n>0,n*factoral(n-1),1)
→factoral(n) [ENTER]                Done
factoral(3) [ENTER]                    6
3! [ENTER]                             6
```

While CATALOG

While _condition_
block
EndWhile

Executes the statements in _block_ as long as _condition_ is true.

block can be either a single statement or a sequence of statements separated with the ":" character.

Program segment:

```
   :
:1→i
:0→temp
:While i<=20
:  temp+1/i→temp
:  i+1→i
:EndWhile
:Disp "sum of reciprocals up to
20",temp
   :
```

"With" — See |, page 468.

xor — MATH/Test menu

Boolean expression1 **xor** *Boolean expression2* ⇒ *Boolean expression*

Returns true if *Boolean expression1* is **true** and *Boolean expression2* is false, or vice versa. Returns false if *Boolean expression1* and *Boolean expression2* are both true or both false. Returns a simplified Boolean expression if either of the original Boolean expressions cannot be resolved to true or false.

Note: See **or** (page 423).

```
true xor true [ENTER]                false
(5>3) xor (3>5) [ENTER]              true
```

XorPic — CATALOG

XorPic *picVar*[, *row*] [, *column*]

Displays the picture stored in *picVar* on the current Graph screen.

Uses XOR logic for each pixel. Only those pixel positions that are exclusive to either the screen or the picture are turned on. This instruction turns off pixels that are turned on in both images.

picVar must contain a pic data type.

row and *column*, if included, specify the pixel coordinates for the upper left corner of the picture. Defaults are (0, 0).

zeros() — MATH/Algebra menu

zeros(*expression*, *var***)** ⇒ *list*

Returns a list of candidate real values of *var* that make *expression*=0. **zeros()** does this by computing **exp▶list(solve(***expression*=0,*var***))**.

```
zeros(a*x^2+b*x+c,x) [ENTER]
```

$$\left\{\frac{-(\sqrt{-(4\cdot a\cdot c-b^2)}+b)}{2\cdot a}\ \frac{\sqrt{-(4\cdot a\cdot c-b^2)}-b}{2\cdot a}\right\}$$

```
a*x^2+b*x+c|x=ans(1)[2] [ENTER]      0
```

For some purposes, the result form for **zeros()** is more convenient than that of **solve()**. However, the result form of **zeros()** cannot express implicit solutions, solutions that require inequalities, or solutions that do not involve *var*.

```
exact(zeros(a*(e^(x)+x)(sign
  (x)-1),x)) [ENTER]                 {}

exact(solve(a*(e^(x)+x)(sign
  (x)-1)=0,x)) [ENTER]
             e^x + x = 0 or x>0 or a = 0
```

Note: See also **cSolve()** (page 385), **cZeros()** (page 387), and **solve()** (page 442).

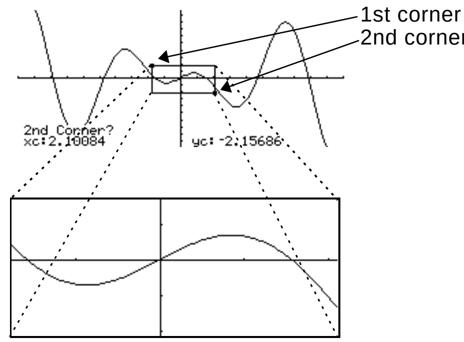
ZoomBox CATALOG

ZoomBox

Displays the Graph screen, lets you draw a box that defines a new viewing window, and updates the window.

In function graphing mode:

```
1.25x*cos(x)→y1(x) [ENTER]                Done
ZoomStd:ZoomBox [ENTER]
```



The display after defining ZoomBox by pressing [ENTER] the second time.

ZoomData CATALOG

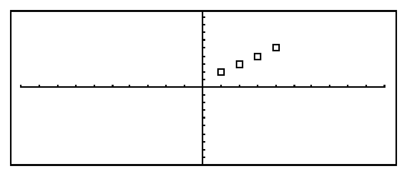
ZoomData

Adjusts the window settings based on the currently defined plots (and data) so that all statistical data points will be sampled, and displays the Graph screen.

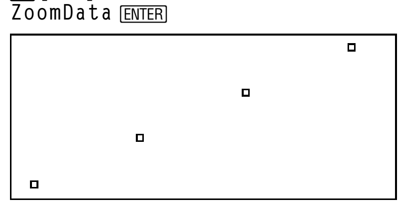
Note: Does not adjust ymin and ymax for histograms.

In function graphing mode:

```
{1,2,3,4}→L1 [ENTER]               {1 2 3 4}
{2,3,4,5}→L2 [ENTER]               {2 3 4 5}
newPlot 1,1,L1,L2 [ENTER]               Done
ZoomStd [ENTER]
```



[♦] [HOME]
ZoomData [ENTER]



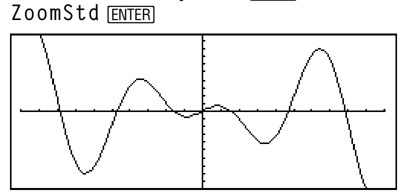
ZoomDec CATALOG

ZoomDec

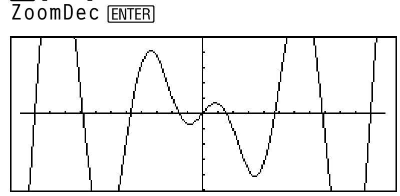
Adjusts the viewing window so that Δx and Δy = 0.1 displays the Graph screen with the origin centered on the screen.

In function graphing mode:

```
1.25x*cos(x)→y1(x) [ENTER]                Done
ZoomStd [ENTER]
```



[♦] [HOME]
ZoomDec [ENTER]



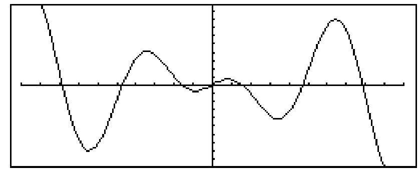
ZoomFit CATALOG

ZoomFit

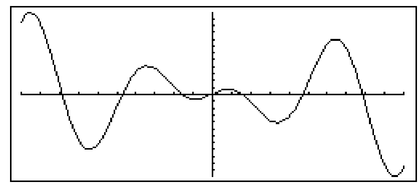
Displays the Graph screen, and calculates the necessary window dimensions for the dependent variables to view all the picture for the current independent variable settings.

In function graphing mode:

1.25x*cos(x)→y1(x) [ENTER] Done
ZoomStd [ENTER]



[◆] [HOME]
ZoomFit [ENTER]



ZoomIn CATALOG

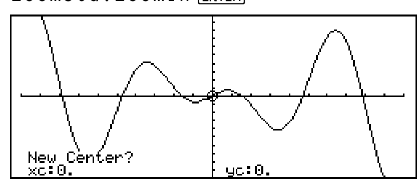
ZoomIn

Displays the Graph screen, lets you set a center point for a zoom in, and updates the viewing window.

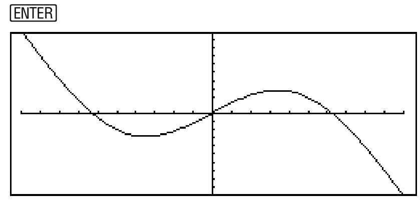
The magnitude of the zoom is dependent on the Zoom factors xFact and yFact. In 3D Graph mode, the magnitude is dependent on xFact, yFact, and zFact.

In function graphing mode:

1.25x*cos(x)→y1(x) [ENTER] Done
ZoomStd:ZoomIn [ENTER]



[ENTER]



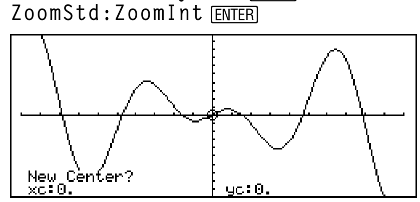
ZoomInt CATALOG

ZoomInt

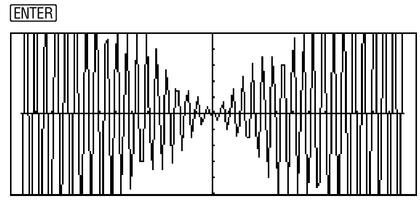
Displays the Graph screen, lets you set a center point for the zoom, and adjusts the window settings so that each pixel is an integer in all directions.

In function graphing mode:

1.25x*cos(x)→y1(x) [ENTER] Done
ZoomStd:ZoomInt [ENTER]



[ENTER]



ZoomOut CATALOG

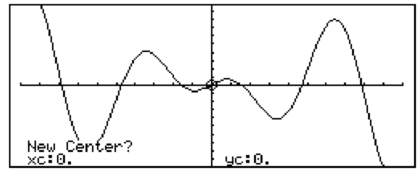
ZoomOut

Displays the Graph screen, lets you set a center point for a zoom out, and updates the viewing window.

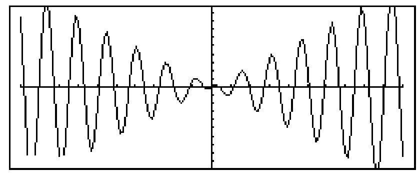
The magnitude of the zoom is dependent on the Zoom factors xFact and yFact. In 3D Graph mode, the magnitude is dependent on xFact, yFact, and zFact.

In function graphing mode:

```
1.25x*cos(x)→y1(x) [ENTER]          Done
ZoomStd:ZoomOut [ENTER]
```



[ENTER]



ZoomPrev CATALOG

ZoomPrev

Displays the Graph screen, and updates the viewing window with the settings in use before the last zoom.

ZoomRcl CATALOG

ZoomRcl

Displays the Graph screen, and updates the viewing window using the settings stored with the **ZoomSto** instruction.

ZoomSqr CATALOG

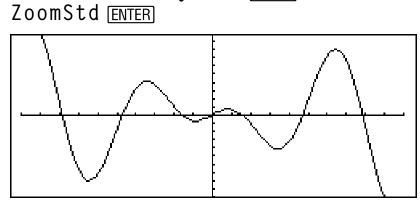
ZoomSqr

Displays the Graph screen, adjusts the x or y window settings so that each pixel represents an equal width and height in the coordinate system, and updates the viewing window.

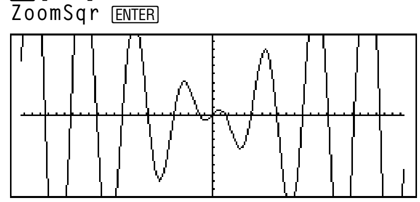
In 3D Graph mode, **ZoomSqr** lengthens the shortest two axes to be the same as the longest axis.

In function graphing mode:

```
1.25x*cos(x)→y1(x) [ENTER]          Done
ZoomStd [ENTER]
```



```
[♦] [HOME]
ZoomSqr [ENTER]
```



ZoomStd   CATALOG

ZoomStd

Sets the window variables to the following standard values, and then updates the viewing window.

Function graphing:
x: [-10, 10, 1], y: [-10, 10, 1] and xres=2

Parametric graphing:
t: [0, 2π, π/24], x:[-10,10,1], y:[-10,10,1]

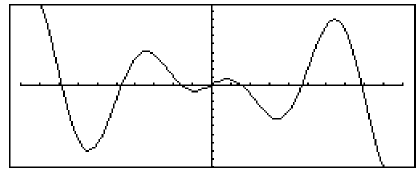
Polar graphing:
θ: [0, 2π, π/24], x:[-10,10,1], y:[-10,10,1]

Sequence graphing:
nmin=1, nmax=10, plotstrt=1, plotstep=1,
x: [-10,10,1], y:[-10,10,1]

3D graphing:
x: [-10, 10, 14], y: [-10,10,14],
z: [-10,10,1], eyeθ°=20, eyeφ°=70

In function graphing mode:

```
1.25x*cos(x)→y1(x) [ENTER]        Done
ZoomStd [ENTER]
```



ZoomSto   CATALOG

ZoomSto

Stores the current Window settings in the Zoom memory. You can use **ZoomRcl** to restore the settings.

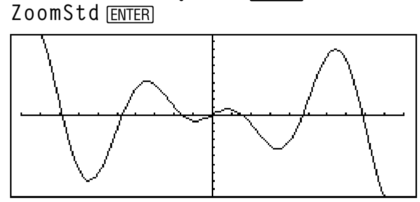
ZoomTrig   CATALOG

ZoomTrig

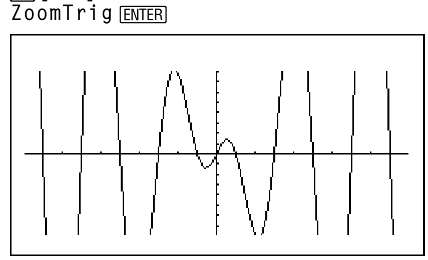
Displays the Graph screen, sets Δx to π/24, and xscl to π/2, centers the origin, sets the y settings to [-4, 4, .5], and updates the viewing window.

In function graphing mode:

```
1.25x*cos(x)→y1(x) [ENTER]        Done
ZoomStd [ENTER]
```



```
[♦] [HOME]
ZoomTrig [ENTER]
```



+ (add) [+] key

expression1 + *expression2* ⇒ *expression*

Returns the sum of *expression1* and *expression2*.

```
56 [ENTER]                56
ans(1)+4 [ENTER]          60
ans(1)+4 [ENTER]          64
ans(1)+4 [ENTER]          68
ans(1)+4 [ENTER]          72
```

---

list1 + *list2* ⇒ *list*
matrix1 + *matrix2* ⇒ *matrix*

Returns a list (or matrix) containing the sums of corresponding elements in *list1* and *list2* (or *matrix1* and *matrix2*).

Dimensions of the arguments must be equal.

```
{22,π,π/2}→L1 [ENTER]            {22 π π/2}
{10,5,π/2}→L2 [ENTER]            {10 5 π/2}
L1+L2 [ENTER]                    {32 π+5 π}
ans(1)+{π,⁻5,⁻π} [ENTER]         {π+32 π 0}
[a,b;c,d]+[1,0;0,1] [ENTER]      [a+1  b  ]
                                 [c    d+1]
```

---

expression + *list1* ⇒ *list*
list1 + *expression* ⇒ *list*

Returns a list containing the sums of *expression* and each element in *list1*.

```
15+{10,15,20} [ENTER]            {25 30 35}
{10,15,20}+15 [ENTER]            {25 30 35}
```

---

expression + *matrix1* ⇒ *matrix*
matrix1 + *expression* ⇒ *matrix*

Returns a matrix with *expression* added to each element on the diagonal of *matrix1*. *matrix1* must be square.

Note: Use **.+** (dot plus) to add an expression to each element.

```
20+[1,2;3,4] [ENTER]
                                 [21  2 ]
                                 [3   24]
```

− (subtract) [−] key

expression1 − *expression2* ⇒ *expression*

Returns *expression1* minus *expression2*.

```
6−2 [ENTER]                      4
π−π/6 [ENTER]                    5•π/6
```

---

list1 − *list2* ⇒ *list*
matrix1 − *matrix2* ⇒ *matrix*

Subtracts each element in *list2* (or *matrix2*) from the corresponding element in *list1* (or *matrix1*), and returns the results.

Dimensions of the arguments must be equal.

```
{22,π,π/2}−{10,5,π/2} [ENTER]    {12 π−5 0}
[3,4]−[1,2] [ENTER]              [2 2]
```

---

expression − *list1* ⇒ *list*
list1 − *expression* ⇒ *list*

Subtracts each *list1* element from *expression* or subtracts *expression* from each *list1* element, and returns a list of the results.

```
15−{10,15,20} [ENTER]            {5 0 ⁻5}
{10,15,20}−15 [ENTER]            {⁻5 0 5}
```

---

expression − *matrix1* ⇒ *matrix*
matrix1 − *expression* ⇒ *matrix*

expression − *matrix1* returns a matrix of *expression* times the identity matrix minus *matrix1*. *matrix1* must be square.

matrix1 − *expression* returns a matrix of *expression* times the identity matrix subtracted from *matrix1*. *matrix1* must be square.

Note: Use **.-** (dot minus) to subtract an expression from each element.

```
20−[1,2;3,4] [ENTER]
                                 [19  ⁻2]
                                 [⁻3  16]
```

* (multiply) [×] key

expression1 * *expression2* ⇒ *expression*

Returns the product of *expression1* and *expression2*.

```
2*3.45 [ENTER]                6.9
x*y*x [ENTER]                 x²·y
```

---

list1 * *list2* ⇒ *list*

Returns a list containing the products of the corresponding elements in *list1* and *list2*.

Dimensions of the lists must be equal.

```
{1.0,2,3}*{4,5,6} [ENTER]     {4. 10 18}
{2/a,3/2}*{a²,b/3} [ENTER]    {2·a  b/2}
```

---

matrix1 * *matrix2* ⇒ *matrix*

Returns the matrix product of *matrix1* and *matrix2*.

The number of rows in *matrix1* must equal the number of columns in *matrix2*.

```
[1,2,3;4,5,6]*[a,d;b,e;c,f] [ENTER]
```

$$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}\cdot\begin{bmatrix}a&d\\b&e\\c&f\end{bmatrix} \quad \begin{bmatrix}a+2\cdot b+3\cdot c & d+2\cdot e+3\cdot f\\4\cdot a+5\cdot b+6\cdot c & 4\cdot d+5\cdot e+6\cdot f\end{bmatrix}$$

---

expression * *list1* ⇒ *list*
list1 * *expression* ⇒ *list*

Returns a list containing the products of *expression* and each element in *list1*.

```
π*{4,5,6} [ENTER]             {4·π 5·π 6·π}
```

---

expression * *matrix1* ⇒ *matrix*
matrix1 * *expression* ⇒ *matrix*

Returns a matrix containing the products of *expression* and each element in *matrix1*.

Note: Use .* (dot multiply) to multiply an expression by each element.

```
[1,2;3,4]*.01 [ENTER]
```

$$\begin{bmatrix}.01&.02\\.03&.04\end{bmatrix}$$

```
λ*identity(3) [ENTER]
```

$$\begin{bmatrix}\lambda&0&0\\0&\lambda&0\\0&0&\lambda\end{bmatrix}$$

/ (divide) [÷] key

expression1 / *expression2* ⇒ *expression*

Returns the quotient of *expression1* divided by *expression2*.

```
2/3.45 [ENTER]                .57971
x^3/x [ENTER]                 x²
```

---

list1 / *list2* ⇒ *list*

Returns a list containing the quotients of *list1* divided by *list2*.

Dimensions of the lists must be equal.

```
{1.0,2,3}/{4,5,6} [ENTER]
                              {.25 2/5 1/2}
```

---

expression / *list1* ⇒ *list*
list1 / *expression* ⇒ *list*

Returns a list containing the quotients of *expression* divided by *list1* or *list1* divided by *expression*.

```
a/{3,a,√(a)} [ENTER]
```

$$\{\frac{a}{3}\ \ 1\ \ \sqrt{a}\}$$

```
{a,b,c}/(a*b*c) [ENTER]
```

$$\{\frac{1}{b\cdot c}\ \ \frac{1}{a\cdot c}\ \ \frac{1}{a\cdot b}\}$$

---

matrix1 / *expression* ⇒ *matrix*

Returns a matrix containing the quotients of *matrix1*/*expression*.

Note: Use ./ (dot divide) to divide an expression by each element.

```
[a,b,c]/(a*b*c) [ENTER]
```

$$\begin{bmatrix}\frac{1}{b\cdot c} & \frac{1}{a\cdot c} & \frac{1}{a\cdot b}\end{bmatrix}$$

⁻ (negate) [(-)] key

⁻*expression1* ⇒ *expression*
⁻*list1* ⇒ *list*
⁻*matrix1* ⇒ *matrix*

Returns the negation of the argument.

For a list or matrix, returns all the elements negated.

```
⁻2.43 [ENTER]                    ⁻2.43

⁻{⁻1,0.4,1.2E19} [ENTER]
                    {1 ⁻.4 ⁻1.2E19}

⁻a*⁻b [ENTER]                     a•b
```

% CHAR/Punctuation menu

expression1 **%** ⇒ *expression*
list1 **%** ⇒ *list*
matrix1 **%** ⇒ *matrix*

Returns $\frac{argument}{100}$.

For a list or matrix, returns a list or matrix with each element divided by 100.

```
13% [♦] [ENTER]                     .13

{1, 10, 100}% [♦] [ENTER]   {.01 .1  1.}
```

= [=] key

expression1 **=** *expression2* ⇒ *Boolean expression*
list1 **=** *list2* ⇒ *Boolean list*
matrix1 **=** *matrix2* ⇒ *Boolean matrix*

Returns true if *expression1* is determined to be equal to *expression2*.

Returns false if *expression1* is determined to not be equal to *expression2*.

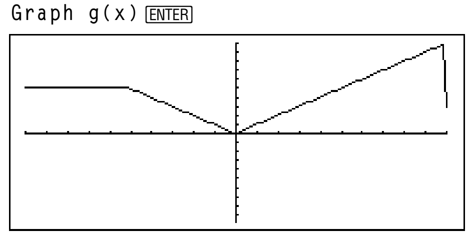
Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

Example function listing using math test symbols: =, ≠, <, ≤, >, ≥

```
:g(xx)
:Func
:If xx≤⁻5 Then
:  Return 5
:  ElseIf xx>⁻5 and xx<0 Then
:  Return ⁻xx
:  ElseIf xx≥0 and xx≠10 Then
:  Return xx
:  ElseIf xx=10 Then
:  Return 3
:EndIf
:EndFunc
```

```
Graph g(x) [ENTER]
```



/= (not equal) [2nd] [V] key or [÷] [=] keys

expression1 **/=** *expression2* ⇒ *Boolean expression*
list1 **/=** *list2* ⇒ *Boolean list*
matrix1 **/=** *matrix2* ⇒ *Boolean matrix*

Returns true if *expression1* is determined to be not equal to *expression2*.

Returns false if *expression1* is determined to be equal to *expression2*.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

See "=" example above.

< 2nd [<] key

expression1 < *expression2* ⇒ *Boolean expression*
list1 < *list2* ⇒ *Boolean list*
matrix1 < *matrix2* ⇒ *Boolean matrix*

Returns true if *expression1* is determined to be less than *expression2*.

Returns false if *expression1* is determined to be greater than or equal to *expression2*.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

See “=” example on previous page.

<= 2nd [<] = keys

expression1 <= *expression2* ⇒ *Boolean expression*
list1 <= *list2* ⇒ *Boolean list*
matrix1 <= *matrix2* ⇒ *Boolean matrix*

Returns true if *expression1* is determined to be less than or equal to *expression2*.

Returns false if *expression1* is determined to be greater than *expression2*.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

See “=” example on previous page.

> 2nd [>] key

expression1 > *expression2* ⇒ *Boolean expression*
list1 > *list2* ⇒ *Boolean list*
matrix1 > *matrix2* ⇒ *Boolean matrix*

Returns true if *expression1* is determined to be greater than *expression2*.

Returns false if *expression1* is determined to be less than or equal to *expression2*.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

See “=” example on previous page.

>=   [2nd] [>] [=] keys

expression1 >= *expression2* ⇒ *Boolean expression*
list1 >= *list2* ⇒ *Boolean list*
matrix1 >= *matrix2* ⇒ *Boolean matrix*

Returns true if *expression1* is determined to be greater than or equal to *expression2*.

Returns false if *expression1* is determined to be less than *expression2*.

Anything else returns a simplified form of the equation.

For lists and matrices, returns comparisons element by element.

See “=” example on page 460.

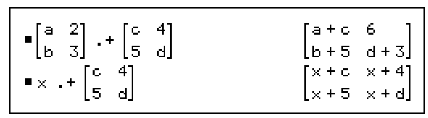
.+ (dot add)   [.] [+] keys

matrix1 **.+** *matrix2* ⇒ *matrix*
expression **.+** *matrix1* ⇒ *matrix*

matrix1 **.+** *matrix2* returns a matrix that is the sum of each pair of corresponding elements in *matrix1* and *matrix2*.

expression **.+** *matrix1* returns a matrix that is the sum of *expression* and each element in *matrix1*.

```
[a,2;b,3].+[c,4;5,d] [ENTER]
x.+[c,4;5,d] [ENTER]
```



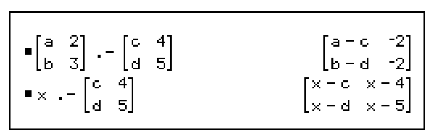
.– (dot subt.)   [.] [–] keys

matrix1 **.–** *matrix2* ⇒ *matrix*
expression **.–** *matrix1* ⇒ *matrix*

matrix1 **.–** *matrix2* returns a matrix that is the difference between each pair of corresponding elements in *matrix1* and *matrix2*.

expression **.–** *matrix1* returns a matrix that is the difference of *expression* and each element in *matrix1*.

```
[a,2;b,3].-[c,4;d,5] [ENTER]
x.-[c,4;d,5] [ENTER]
```



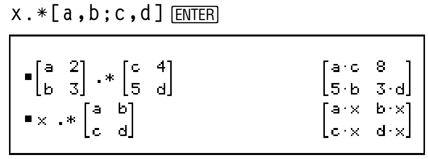
.* (dot mult.)   [.] [×] keys

matrix1 **.*** *matrix2* ⇒ *matrix*
expression **.*** *matrix1* ⇒ *matrix*

matrix1 **.*** *matrix2* returns a matrix that is the product of each pair of corresponding elements in *matrix1* and *matrix2*.

expression **.*** *matrix1* returns a matrix containing the products of *expression* and each element in *matrix1*.

```
[a,2;b,3].*[c,4;5,d] [ENTER]
x.*[a,b;c,d] [ENTER]
```



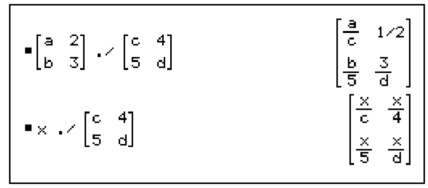
./ (dot divide) [.] [÷] keys

matrix1 ./ *matrix2* ⇒ *matrix*
expression ./ *matrix1* ⇒ *matrix*

matrix1 ./ *matrix2* returns a matrix that is the quotient of each pair of corresponding elements in *matrix1* and *matrix2*.

expression ./ *matrix1* returns a matrix that is the quotient of *expression* and each element in *matrix1*.

```
[a,2;b,3]./[c,4;5,d] [ENTER]
x./[c,4;5,d] [ENTER]
```



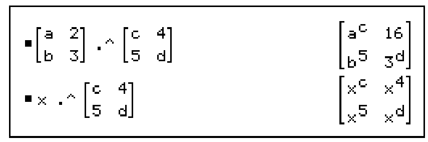
.^ (dot power) [.] [^] keys

matrix1 .^ *matrix2* ⇒ *matrix*
expression .^ *matrix1* ⇒ *matrix*

matrix1 .^ *matrix2* returns a matrix where each element in *matrix2* is the exponent for the corresponding element in *matrix1*.

expression . ^ *matrix1* returns a matrix where each element in *matrix1* is the exponent for *expression*.

```
[a,2;b,3].^[c,4;5,d] [ENTER]
x.^[c,4;5,d] [ENTER]
```



! (factorial) [2nd] [W] key

expression1! ⇒ *expression*
list1! ⇒ *list*
matrix1! ⇒ *matrix*

Returns the factorial of the argument.

For a list or matrix, returns a list or matrix of factorials of the elements.

The TI-92 computes a numeric value for only non-negative whole-number values.

```
5! [ENTER]                    120
{5,4,3}! [ENTER]      {120  24  6}
[1,2;3,4]! [ENTER]       [1   2]
                         [6  24]
```

& (append) [2nd] [H] key

string1 & *string2* ⇒ *string*

Returns a text string that is *string2* appended to *string1*.

```
"Hello " & "Nick" [ENTER]   "Hello Nick"
```

∫() (integrate) [2nd] [∫] **key**

∫(*expression1*, *var*[, *lower*] [,*upper*]**)** ⇒ *expression*

Returns the integral of *expression1* with respect to the variable *var* from *lower* to *upper*.

```
∫(x^2,x,a,b) [ENTER]
```
$\frac{-a^3}{3}+\frac{b^3}{3}$

Returns an anti-derivative if *lower* and *upper* are omitted. A symbolic constant of integration such as C is omitted.

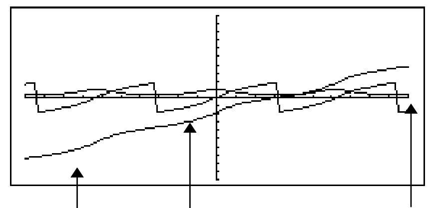
```
∫(x^2,x) [ENTER]
```
$\frac{x^3}{3}$

However, *lower* is added as a constant of integration if only *upper* is omitted.

```
∫(a*x^2,x,c) [ENTER]
```
$\frac{a\cdot x^3}{3}+c$

Equally valid anti-derivatives might differ by a numeric constant. Such a constant might be disguised—particularly when an anti-derivative contains logarithms or inverse trigonometric functions. Moreover, piecewise constant expressions are sometimes added to make an anti-derivative valid over a larger interval than the usual formula.

```
∫(1/(2-cos(x)),x)→tmp [ENTER]
ClrGraph:Graph tmp:Graph
  1/(2-cos(x)):Graph √(3)
  (2tan-1(√(3)(tan(x/2)))/3) [ENTER]
```



$\int\left(\frac{1}{2-\cos(x)}\right)dx$

$\sqrt{3}\cdot\frac{2\cdot\tan^{-1}\left(\sqrt{3}\cdot\tan\left(\frac{x}{2}\right)\right)}{3}$

∫() returns itself for pieces of *expression1* that it cannot determine as an explicit finite combination of its built-in functions and operators.

```
∫(b*e^(-x^2)+a/(x^2+a^2),x) [ENTER]
```

■ $\int\left(b\cdot e^{-x^2}+\frac{a}{x^2+a^2}\right)dx$
$b\cdot\int\left(e^{-x^2}\right)dx+\tan^{-1}\left(\frac{x}{a}\right)$

When *lower* and *upper* are both present, an attempt is made to locate any discontinuities or discontinuous derivatives in the interval *lower* < *var* < *upper* and to subdivide the interval at those places.

For the AUTO setting of the Exact/Approx mode, numerical integration is used where applicable when an anti-derivative or a limit cannot be determined.

For the APPROX setting, numerical integration is tried first, if applicable. Anti-derivatives are sought only where such numerical integration is inapplicable or fails.

```
∫(e^(-x^2),x,-1,1)[♦] [ENTER]      1.493...
```

∫() can be nested to do multiple integrals. Integration limits can depend on integration variables outside them.

```
∫(∫(ln(x+y),y,0,x),x,0,a) [ENTER]
```

■ $\int_0^a\int_0^x \ln(x+y)\,dy\,dx$
$\frac{a^2\cdot\ln(a)}{2}+a^2\cdot(\ln(2)-3/4)$

Note: See also **nInt()** (page 421).

√() (sqr. root) [2nd] [√] **key**

√ **(***expression1***)** ⇒ *expression*
√ **(***list1***)** ⇒ *list*

Returns the square root of the argument.

For a list, returns the square roots of all the elements in *list1*.

```
√(4) ENTER                    2
√({9,a,4}) ENTER      {3  √a  2}
```

Π() (product) **MATH/Calculus menu**

Π(*expression1*, *var*, *low*, *high***)** ⇒ *expression*

Evaluates *expression1* for each value of *var* from *low* to *high*, and returns the product of the results.

Π(1/n,n,1,5) [ENTER] $\frac{1}{120}$

Π(k^2,k,1,n) [ENTER] $(n!)^2$

Π({1/n,n,2},n,1,5) [ENTER]
$\left\{\frac{1}{120}\ \ 120\ \ 32\right\}$

Π(*expression1*, *var*, *low*, *low*−1**)** ⇒ 1

Π(k,k,4,3) [ENTER] 1

Π(*expression1*, *var*, *low*, *high***)** ⇒ 1/Π(*expression1*, *var*, *high*+1, *low*−1) if *high* < *low*−1

Π(1/k,k,4,1) [ENTER] 6

Π(1/k,k,4,1)∗Π(1/k,k,2,4) [ENTER] 1/4

Σ() (sum) [2nd] [Σ] **key**

Σ(*expression1*, *var*, *low*, *high***)** ⇒ *expression*

Evaluates *expression1* for each value of *var* from *low* to *high*, and returns the sum of the results.

Σ(1/n,n,1,5) [ENTER] $\frac{137}{60}$

Σ(k^2,k,1,n) [ENTER] $\frac{n\cdot(n+1)\cdot(2\cdot n+1)}{6}$

Σ(1/n^2,n,1,∞) [ENTER] $\frac{\pi^2}{6}$

Σ(*expression1*, *var*, *low*, *low*−1**)** ⇒ 0

Σ(k,k,4,3) [ENTER] 0

Σ(*expression1*, *var*, *low*, *high***)** ⇒ ⁻ Σ**(***expression1*, *var*, *high*+1, *low*−1) if *high* < *low*−1

Σ(k,k,4,1) [ENTER] ⁻5

Σ(k,k,4,1)+Σ(k,k,2,4) [ENTER] 4

^ (power) [^] key

expression1 **^** *expression2* ⇒ *expression*
list1 **^** *list2* ⇒ *list*

Returns the first argument raised to the power of the second argument.

For a list, returns the elements in *list1* raised to the power of the corresponding elements in *list2*.

In the real domain, fractional powers that have reduced exponents with odd denominators use the real branch versus the principal branch for complex mode.

```
4^2 [ENTER]                                16
{a,2,c}^{1,b,3} [ENTER]         {a  2^b  c^3}
```

---

expression **^** *list1* ⇒ *list*

Returns *expression* raised to the power of the elements in *list1*.

```
p^{a,2,-3} [ENTER]
```

$\{p^a \quad p^2 \quad \frac{1}{p^3}\}$

---

list1 **^** *expression* ⇒ *list*

Returns the elements in *list1* raised to the power of *expression*.

```
{1,2,3,4}^-2 [ENTER]
                          {1  1/4  1/9  1/16}
```

---

squareMatrix1 **^** *integer* ⇒ *matrix*

Returns *squareMatrix1* raised to the *integer* power.

squareMatrix1 must be a square matrix.

If *integer* = ⁻1, computes the inverse matrix.
If *integer* < ⁻1, computes the inverse matrix to an appropriate positive power.

```
[1,2;3,4]^2 [ENTER]
[1,2;3,4]^-1 [ENTER]
[1,2;3,4]^-2 [ENTER]
```

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^2 \quad \begin{bmatrix}7 & 10\\15 & 22\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{-1} \quad \begin{bmatrix}-2 & 1\\3/2 & -1/2\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{-2} \quad \begin{bmatrix}11/2 & -5/2\\-15/4 & 7/4\end{bmatrix}$$

10^() CATALOG

10^ (*expression1***)** ⇒ *expression*
10^ (*list1***)** ⇒ *list*

Returns 10 raised to the power of the argument.

For a list, returns 10 raised to the power of the elements in *list1*.

```
10^1.5 [ENTER]                        31.622...
10^{0,-2,2,a} [ENTER]
```

$\{1 \quad \frac{1}{100} \quad 100 \quad 10^a\}$

(indirection) [2nd] [T] key

*varNameString*

Refers to the variable whose name is *varNameString*. This lets you create and modify variables from a program using strings.

Program segment:

```
   :
:Request "Enter Your Name",str1
:NewFold #str1
   :

   :
:For i,1,5,1
:  ClrGraph
:  Graph i*x
:  StoPic #("pic" & string(i))
:EndFor
   :
```

^{r} (radian) — MATH/Angle menu

expression1^r ⇒ *expression*
list1^r ⇒ *list*
matrix1^r ⇒ *matrix*

In Degree angle mode, multiplies *expression1* by 180/π. In Radian angle mode, returns *expression1* unchanged.

This function gives you a way to use a radian angle while in Degree mode. (In Degree angle mode, **sin()**, **cos()**, **tan()**, and polar-to-rectangular conversions expect the angle argument to be in degrees.)

Hint: Use ^{r} if you want to force radians in a function or program definition regardless of the mode that prevails when the function or program is used.

In Degree or Radian angle mode:

cos((π/4)^r) [ENTER] $\frac{\sqrt{2}}{2}$

cos({0^r,(π/12)^r,-π^r}) [ENTER]
$\{1 \quad \frac{(\sqrt{3}+1)\cdot\sqrt{2}}{4} \quad -1\}$

° (degree) — [2nd] [D] key

expression° ⇒ *value*
list1° ⇒ *list*
matrix1° ⇒ *matrix*

In Radian angle mode, multiplies *expression* by π/180. In Degree angle mode, returns *expression* unchanged.

This function gives you a way to use a degree angle while in Radian mode. (In Radian angle mode, **sin()**, **cos()**, **tan()**, and polar-to-rectangular conversions expect the angle argument to be in radians.)

In Radian angle mode:

cos(45°) [ENTER] $\frac{\sqrt{2}}{2}$

cos({0,π/4,90°,30.12°}) [♦] [ENTER]
{1 .707... 0 .864...}

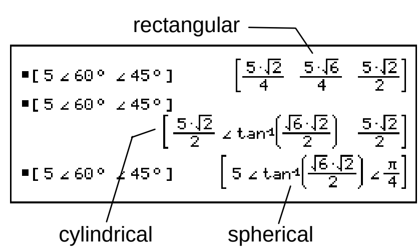
∠ (angle) — [2nd] [F] key

[*radius*,∠θ_*angle*] ⇒ *vector* (polar input)
[*radius*,∠θ_*angle*,*Z_coordinate*] ⇒ *vector* (cylindrical input)
[*radius*,∠θ_*angle*,∠φ_*angle*] ⇒ *vector* (spherical input)

Returns coordinates as a vector depending on the Vector Format mode setting: rectangular, cylindrical, or spherical.

[5,∠60°,∠45°] [ENTER]

In Radian mode and vector format set to:



°, ', " — [2nd] [D] key (°), [2nd] [B] key ('), [2nd] [L] key (")

dd°*mm*'*ss.ss*" ⇒ *expression*

dd A positive or negative number
mm A non-negative number
ss.ss A non-negative number

Returns *dd*+(*mm*/60)+(*ss.ss*/3600).

This base-60 entry format lets you:

- Enter an angle in degrees/minutes/seconds without regard to the current angle mode.
- Enter time as hours/minutes/seconds.

In Degree angle mode:

25°13'17.5" [ENTER] 25.221...

25°30' [ENTER] 51/2

x^{-1} — [2nd] [x^{-1}] key

expression1 x^{-1} ⇒ *expression*
list1 x^{-1} ⇒ *list*

Returns the reciprocal of the argument.

For a list, returns the reciprocals of the elements in *list1*.

```
3.1^-1 [ENTER]                      .322581
{a,4,-.1,x-2}^-1 [ENTER]
```

$$\left\{\frac{1}{a}\quad \frac{1}{4}\quad {}^{-}10\quad \frac{1}{x-2}\right\}$$

---

squareMatrix1 x^{-1} ⇒ *squareMatrix*

Returns the inverse of *squareMatrix1*.

squareMatrix1 must be a non-singular square matrix.

```
[1,2;3,4]^-1 [ENTER]
[1,2;a,4]^-1 [ENTER]
```

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \quad \begin{bmatrix}{}^{-}2 & 1\\ 3/2 & {}^{-}1/2\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ a & 4\end{bmatrix}^{-1} \quad \begin{bmatrix}\frac{{}^{-}2}{a-2} & \frac{1}{a-2}\\ \frac{a}{2\cdot(a-2)} & \frac{{}^{-}1}{2\cdot(a-2)}\end{bmatrix}$$

| ("with") — [2nd] [K] key

expression **|** *Boolean expression1* [*and Boolean expression2*]...[*and Boolean expressionN*]

The "with" (|) symbol serves as a binary operator. The operand to the left of | is an expression. The operand to the right of | specifies one or more relations that are intended to affect the simplification of the expression. Multiple relations after | must be joined by a logical "and".

```
x+1| x=3 [ENTER]                    4
x+y| x=sin(y) [ENTER]               sin(y) + y
x+y| sin(y)=x [ENTER]               x + y
```

The "with" operator provides three basic types of functionality: substitutions, interval constraints, and exclusions.

Substitutions are in the form of an equality, such as x=3 or y=sin(x). To be most effective, the left side should be a simple variable. *expression* | *variable* = *value* will substitute *value* for every occurrence of *variable* in *expression*.

```
xx^3-2xx+7→f(xx) [ENTER]            Done
f(x)| x=√(3) [ENTER]                3^(3/2) - 2·√3 + 7
(sin(x))^2+2sin(x)-6| sin(x)=d [ENTER]
                                    d²+2d-6
```

Interval constraints take the form of one or more inequalities joined by logical "and" operators. Interval constraints also permit simplification that otherwise might be invalid or not computable.

```
solve(x^2-1=0,x)|x>0 and x<2 [ENTER]
                                    x = 1
√(x)*√(1/x)|x>0 [ENTER]             1
√(x)*√(1/x) [ENTER]                 √(1/x) · √x
```

Exclusions use the "not equals" (/= or ≠) relational operator to exclude a specific value from consideration. They are used primarily to exclude an exact solution when using **cSolve()**, **cZeros()**, **fMax()**, **fMin()**, **solve()**, **zeros()**, etc.

```
solve(x^2-1=0,x)| x≠1 [ENTER]       x = -1
```

→ (store) STO► key

expression → *var*
list → *var*
matrix → *var*
expression → *fun_name(parameter1,...)*
list → *fun_name(parameter1,...)*
matrix → *fun_name(parameter1,...)*

If variable *var* does not exist, creates *var* and initializes it to *expression*, *list*, or *matrix*.

If *var* already exists and if it is not locked or protected, replaces its contents with *expression*, *list*, or *matrix*.

Hint: If you plan to do symbolic computations using undefined variables, avoid storing anything into commonly used, one-letter variables such as a, b, c, x, y, z, etc.

```
π/4→myvar ENTER                    π/4
2cos(x)→Y1(x) ENTER               Done
{1,2,3,4}→Lst5 ENTER          {1 2 3 4}
[1,2,3;4,5,6]→MatG ENTER      [1 2 3]
                              [4 5 6]
"Hello"→str1 ENTER             "Hello"
```

© (comment) 2nd [X] key or Program Editor/Control menu

© [*text*]

© processes *text* as a comment line, which can be used to annotate program instructions.

© can be at the beginning or anywhere in the line. Everything to the right of ©, to the end of the line, is the comment.

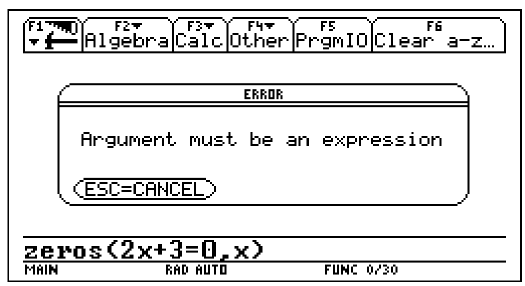
Program segment:

```
  :
:© Get 10 points from the Graph
   screen
:For i,1,10 © This loops 10 times
  :
```

Reference Information

B

This appendix contains reference information that includes a comprehensive list of error messages, TI-92 modes of operation, character codes, key maps, system variables and reserved names, and the EOS™ hierarchy.



Relevant messages are displayed to help you find and correct errors in your entries.

TI-92 Error Messages

The table below lists error messages that may be displayed when input or internal errors are encountered. The number to the left of each error message represents an internal error number that is not displayed. If the error occurs inside a Try...EndTry block, the error number is stored in system variable errornum. Many of the error messages are self-explanatory and do not require descriptive information. However, additional information has been added for some error messages on a selective basis.

Error Number	Description
10	**A function did not return a value**
20	**A test did not resolve to TRUE or FALSE** Generally, undefined variables cannot be compared. For example, the test If a<b will cause this error if either a or b is undefined when the If statement is executed.
30	**Argument cannot be a folder name**
40	**Argument error**
50	**Argument mismatch** Two or more arguments must be of the same type. For example, **PtOn** *expression1*,*expression2* and **PtOn** *list1*,*list2* are both valid, but **PtOn** *expression*,*list* is a mismatch.
60	**Argument must be a Boolean expression**
70	**Argument must be a decimal number**
80	**Argument must be a label name**
90	**Argument must be a list**
100	**Argument must be a matrix**
110	**Argument must be a Pic**
120	**Argument must be a Pic or string**
130	**Argument must be a string**
140	**Argument must be a variable name** For example, DelVar 12 is invalid because a number cannot be a variable name.
150	**Argument must be an empty folder name**

Error Number	Description
160	**Argument must be an expression** For example, zeros(2x+3=0,x) is invalid because the first argument is an equation.
170	**Bound** For the interactive graph math functions like 2:Zero, the lower bound must be less than the upper bound to define the search interval.
180	**Break** The [ON] key was pressed during a long calculation or during program execution.
190	**Circular definition** This message is displayed to avoid running out of memory during infinite replacement of variable values during simplification. For example, a+1→a, where a is an undefined variable, will cause this error.
200	**Constraint expression invalid** For example, solve(3x^2−4=0, x) | x<0 or x>5 would produce this error message because the constraint is separated by “or” and not “and.”
210	**Data type** An argument is of the wrong data type. For example, **sin(***expression***)** is valid, but **sin(***matrix***)** is not valid because the matrix data type is not supported by the **sin()** function.
220	**Dependent Limit** A limit of integration is dependent on the integration variable. For example, ∫(x^2,x,1,x) is not allowed.
230	**Dimension** A list or matrix index is not valid. For example, if the list {1,2,3,4} is stored in L1, then L1[5] is a dimension error because L1 only contains four elements.
240	**Dimension mismatch** Two or more arguments must be of the same dimension. For example, [1,2]+[1,2,3] is a dimension mismatch because the matrices contain a different number of elements.
250	**Divide by zero**
260	**Domain error** An argument must be in a specified domain. For example, ans(100) is not valid because the argument for **ans()** must be in the range 1–99.
270	**Duplicate variable name**

Error Number	Description
280	**Else and ElseIf invalid outside of If..EndIf block**
290	**EndTry is missing the matching Else statement**
300	**Expected 2 or 3-element list or matrix**
310	**First argument of nSolve must be a univariate equation** The first argument must be an equation, and the equation cannot contain a non-valued variable other than the variable of interest. For example, nSolve(3x^2−4=0, x) is a valid equation; however, nSolve(3x^2−4, x) is not an equation, and nSolve(3x^2−y=0,x) is not a univariate equation because y has no value in this example.
320	**First argument of solve or cSolve must be an equation or inequality** For example, solve(3x^2−4, x) is invalid because the first argument is not an equation.
330	**Folder** An attempt was made in the VAR-LINK menu to store a variable in a folder that does not exist.
340	**Incomplete initial object list** There are too few initial objects chosen to define the macro's final object.
350	**Index out of range**
360	**Indirection string is not a valid variable name**
370	**Initial and final are same object** The initial and final objects chosen for the geometry macro are the same object.
380	**Invalid ans()**
390	**Invalid assignment**
400	**Invalid assignment value**
410	**Invalid command**
420	**Invalid folder name**
430	**Invalid for the current mode settings**
440	**Invalid implied multiply** For example, x(x+1) is invalid; whereas, x*(x+1) is the correct syntax. This is to avoid confusion between implied multiplication and function calls.

Error Number	Description
450	**Invalid in a function or current expression** Only certain commands are valid in a user-defined function. Entries that are made in the Window Editor, Table Editor, Data/Matrix Editor, and Geometry, as well as system prompts such as Lower Bound cannot contain any commands or a colon (:). See also “Creating and Evaluating User-Defined Functions” in Chapter 10.
460	**Invalid in Custom..EndCustm block**
470	**Invalid in Dialog..EndDlog block**
480	**Invalid in Toolbar..EndTBar block**
490	**Invalid in Try..EndTry block**
500	**Invalid label** Label names must follow the same rules used for naming variables.
510	**Invalid list or matrix** For example, a list inside a list such as {2,{3,4}} is not valid.
520	**Invalid outside Custom..EndCustm or ToolBar..EndTbar blocks** For example, an **Item** command is attempted outside a **Custom** or **ToolBar** structure.
530	**Invalid outside Dialog..EndDlog, Custom..EndCustm, or ToolBar..EndTBar blocks** For example, a Title command is attempted outside a **Dialog**, **Custom**, or **ToolBar** structure.
540	**Invalid outside Dialog..EndDlog block** For example, the **DropDown** command is attempted outside a **Dialog** structure.
550	**Invalid outside function or program** A number of commands are not valid outside a program or a function. For example, Local cannot be used unless it is in a program or function.
560	**Invalid outside Loop..EndLoop, For..EndFor, or While..EndWhile blocks** For example, the **Exit** command is valid only inside these loop blocks.
570	**Invalid pathname** For example, \\var is invalid.
580	**Invalid program reference** Programs cannot be referenced within functions or expressions such as 1+p(x) where p is a program.

Error Number	Description
590	**Invalid syntax block** A **Dialog..EndDlog** block is empty or has more than one title. A **Custom..EndCustm** block cannot contain PIC variables, and items must be preceded by a title. A **Toolbar..EndTBar** block must have a second argument if no items follow; or items must have a second argument and must be preceded by a title.
600	**Invalid table**
610	**Invalid variable name in a Local statement**
620	**Invalid variable or function name**
630	**Invalid variable reference**
640	**Invalid vector syntax**
650	**Link transmission** A transmission between two units was not completed. Verify that the connecting cable is connected firmly to both units.
660	**Macro objects cannot be redefined** An object in Geometry that was created by a macro cannot be redefined with Redefine Point.
670 **673**	**Memory** The calculation required more memory than was available at that time.
680	**Missing (**
690	**Missing)**
700	**Missing "**
710	**Missing]**
720	**Missing }**
730	**Missing start or end of block syntax**
740	**Missing Then in the If..EndIf block**
750	**Name is not a function or program**
760	**No final object** No final objects were selected for a macro definition in Geometry.
770	**No initial object** No initial objects were selected for a macro definition in Geometry.

Error Number	Description
780	**No solution** Using the interactive math features (F5:Math) in the Graph application can give this error. For example, if you attempt to find an inflection point of the parabola $y1(x)=x^2$, which does not exist, this error will be displayed.
790	**Non-algebraic variable in expression** If a is the name of a PIC, GDB, MAC, FIG, etc., a+1 is invalid. Use a different variable name in the expression or delete the variable.
800	**Non-real result** For example, if the unit is in the REAL setting of the Complex Format mode, ln(⁻2) is invalid.
810	**Not enough memory to save current variable. Please delete unneeded variables on the Var-Link screen and re-open editor as current OR re-open editor and use F1 8 to clear editor.** This error message is caused by very low memory conditions inside the Data/Matrix Editor.
820	**Objects are unrelated** A macro cannot be defined because the initial and final objects selected are geometrically unrelated.
830	**Overflow**
840	**Plot setup**
850	**Program not found** A program reference inside another program could not be found in the provided path during execution.
860	**Recursion is limited to 255 calls deep**
870	**Reserved name or system variable**
880	**Sequence setup**
890	**Singular matrix**
900	**Stat**
910	**Syntax** The structure of the entry is incorrect. For example, x+−y (x plus minus y) is invalid; whereas, x+⁻y (x plus negative y) is correct.
920	**The point does not lie on a path**

Error Number	Description
930	**Too few arguments** The expression or equation is missing one or more arguments. For example, *d*(f(x)) is invalid; whereas, *d*(f(x),x) is the correct syntax.
940	**Too many arguments** The expression or equation contains an excessive number of arguments and cannot be evaluated.
950	**Too many subscripts**
960	**Undefined variable**
970	**Variable in use so references or changes are not allowed**
980	**Variable is locked or protected**
990	**Variable name is limited to 8 characters**
1000	**Window variables domain**
1010	**Zoom**
	Warning: ∞^0 or undef^0 replaced by 1
	Warning: 0^0 replaced by 1
	Warning: 1^∞ or 1^undef replaced by 1
	Warning: cSolve might specify more zeros
	Warning: Differentiating an equation may produce a false equation
	Warning: Expected finite real integrand
	Warning: Memory full, simplification might be incomplete
	Warning: Object already exists
	Warning: Operation might introduce false solutions
	Warning: Operation might lose solutions
	Warning: Overflow replaced by ∞ or ⁻∞
	Warning: Questionable accuracy
	Warning: Questionable solution
	Warning: Solve might specify more zeros
	Warning: Trig function argument too big for accurate reduction

TI-92 Modes

This section describes the modes of the TI-92 and lists the possible settings of each mode. These mode settings are displayed when you press [MODE].

Graph

Specifies the type of graphs you can plot.

1:FUNCTION	y(x) functions (Chapter 3)
2:PARAMETRIC	x(t) and y(t) parametric equations (Chapter 11)
3:POLAR	r(θ) polar equations (Chapter 12)
4:SEQUENCE	u(n) sequences (Chapter 13)
5:3D	z(x,y) 3D equations (Chapter 14)

Note: If you use a split screen with Number of Graphs = 2, Graph is for the top or left part of the screen and Graph 2 is for the bottom or right part.

Current Folder

__Note:__ For detailed information about using folders, see Chapter 10.

Specifies the current folder. You can set up multiple folders with unique configurations of variables, graph databases, programs, etc.

1:main	Default folder included with the TI-92.
2: — (custom folders)	Other folders are available only if they have been created by a user.

Display Digits

Selects the number of digits. These decimal settings affect only how results are displayed—you can enter a number in any format.

Internally, the TI-92 retains decimal numbers with 14 significant digits. For display purposes, such numbers are rounded to a maximum of 12 significant digits.

1:FIX 0 2:FIX 1 … D:FIX 12	Results are always displayed with the selected number of decimal places.
E:FLOAT	The number of decimal places varies, depending on the result.
F:FLOAT 1 G:FLOAT 2 … Q:FLOAT 12	If the integer part has more than the selected number of digits, the result is rounded and displayed in scientific notation. For example, in FLOAT 4: 12345. is shown as 1.235E4

Angle

Specifies the units in which angle values are interpreted and displayed in trig functions and polar/rectangular conversions.

1:RADIAN
2:DEGREE

Exponential Format

Specifies which notation format should be used. These formats affect only how an answer is displayed; you can enter a number in any format. Numeric answers can be displayed with up to 12 digits and a 3-digit exponent.

1:NORMAL	Expresses numbers in standard format. For example, 12345.67
2:SCIENTIFIC	Expresses numbers in two parts: • The significant digits display with one digit to the left of the decimal. • The power of 10 displays to the right of E. For example, 1.234567E4 means 1.234567×10^4
3:ENGINEERING	Similar to scientific notation. However: • The number may have one, two, or three digits before the decimal. • The power-of-10 exponent is a multiple of three. For example, 12.34567E3 means 12.34567×10^3

Note: If you select NORMAL, but the answer cannot be displayed in the number of digits selected by Display Digits, the TI-92 displays the answer in SCIENTIFIC notation. If Display Digits = FLOAT, scientific notation will be used for exponents of 12 or more and exponents of −4 or less.

Complex Format

Specifies whether complex results are displayed and, if so, their format.

1:REAL	Does not display complex results. (If a result is a complex number and the input does not contain the complex unit ***i***, an error message is displayed.)
2:RECTANGULAR	Displays complex numbers in the form: a+b***i***
3:POLAR	Displays complex numbers in the form: $re^{i\theta}$

Vector Format

Determines how 2-element and 3-element vectors are displayed. You can enter vectors in any of the coordinate systems.

1:RECTANGULAR	Coordinates are in terms of x, y, and z. For example, [3,5,2] represents x = 3, y = 5, and z = 2.
2:CYLINDRICAL	Coordinates are in terms of r, θ, and z. For example, [3,∠45,2] represents r = 3, θ = 45, and z = 2.
3:SPHERICAL	Coordinates are in terms of r, θ, and φ. For example, [3, ∠45, ∠90] represents r = 3, θ = 45, and φ = 90.

Pretty Print

Determines how results are displayed on the Home screen.

1:OFF	Results are displayed in a linear, one-dimensional form. For example, π^2, π/2, or √((x-3)/x)
2:ON	Results are displayed in conventional mathematical format. For example, π^2, $\frac{\pi}{2}$, or $\sqrt{\frac{x-3}{x}}$

Note: For a complete description of these settings, refer to “Formats of Displayed Results” in Chapter 2.

Split Screen

Lets you split the screen into two parts. For example, you can display a graph and see the Y= Editor at the same time (Chapter 5).

1:FULL	The screen is not split.
2:TOP-BOTTOM	The applications are shown in two screens that are above and below each other.
3:LEFT-RIGHT	The applications are shown in two screens that are to the left and right of each other.

To determine what and how information is displayed on a split screen, use this mode in conjunction with other modes such as Split 1 App, Split 2 App, Number of Graphs, and Split Screen Ratio.

Split 1 App and Split 2 App

Specifies which application is displayed on the screen.

- For a full screen, only Split 1 App is active.
- For a split screen, Split 1 App is the top or left part of the screen and Split 2 App is the bottom or right part.

The available application choices are those listed when you press ◯ from the Page 2 mode screen or when you press [APPS]. You must have different applications in each screen unless you are in 2-graph mode.

Number of Graphs

Specifies whether both parts of a split screen can display graphs at the same time.

1	Only one part can display graphs.
2	Both parts can display an independent graph screen (Graph or Graph 2 setting) with independent settings.

Graph 2

Specifies the type of graphs that you can plot for the second graph on a two-graph split screen. This is active only when Number of Graphs = 2. In this two-graph setting, Graph sets the type of graph for the top or left part of the split screen, and Graph 2 sets the bottom or right part. The available choices are the same as for Graph.

Split Screen Ratio

Specifies the proportional sizes of the two parts of a split screen.

1:1	The screen is split evenly.
1:2	The bottom or right part is approximately twice the size of the top or left part.
2:1	The top or left part is approximately twice the size of the bottom or right part.

Exact/Approx

Specifies how fractional and symbolic expressions are calculated and displayed. By retaining rational and symbolic forms in the EXACT setting, the TI-92 increases precision by eliminating most numeric rounding errors.

1:AUTO	Uses EXACT setting in most cases. However, uses APPROXIMATE if the entry contains a decimal point.
2:EXACT	Displays non-whole-number results in their rational or symbolic form.
3:APPROXIMATE	Displays numeric results in floating-point form.

Note: For a complete description of these settings, refer to "Formats of Displayed Results" in Chapter 2.

TI-92 Character Codes

The **char()** function lets you refer to any TI-92 character by its numeric character code. For example, to display ◆ on the Program I/O screen, use `Disp char(127)`. You can use the **ord()** function to find the numeric code of a character. For example, `ord("A")` returns the value 65.

1. SOH	41.)	81. Q	121. y	161. ¡	201. É	241. ñ
2. STX	42. *	82. R	122. z	162. ¢	202. Ê	242. ò
3. ETX	43. +	83. S	123. {	163. £	203. Ë	243. ó
4. EOT	44. ,	84. T	124. |	164. ¤	204. Ì	244. ô
5. ENQ	45. −	85. U	125. }	165. ¥	205. Í	245. õ
6. ACK	46. .	86. V	126. ~	166. ¦	206. Î	246. ö
7. BELL	47. /	87. W	127. ◆	167. §	207. Ï	247. ÷
8. BS	48. 0	88. X	128. α	168. √	208. Ð	248. ø
9. TAB	49. 1	89. Y	129. β	169. ◐	209. Ñ	249. ù
10. LF	50. 2	90. Z	130. Γ	170. ª	210. Ò	250. ú
11. ✎	51. 3	91. [	131. γ	171. «	211. Ó	251. û
12. FF	52. 4	92. \	132. Δ	172. ¬	212. Ô	252. ü
13. CR	53. 5	93.]	133. δ	173. -	213. Õ	253. ý
14. 🔒	54. 6	94. ^	134. ε	174. ®	214. Ö	254. þ
15. ✓	55. 7	95. _	135. ζ	175. ¯	215. ×	255. ÿ
16. ▪	56. 8	96. `	136. θ	176. °	216. Ø	
17. ◀	57. 9	97. a	137. λ	177. $^{+}$	217. Ù	
18. ▶	58. :	98. b	138. ξ	178. ^{2}	218. Ú	
19. ▲	59. ;	99. c	139. Π	179. ^{3}	219. Û	
20. ▼	60. <	100. d	140. π	180. $^{-1}$	220. Ü	
21. ←	61. =	101. e	141. ρ	181. µ	221. Ý	
22. →	62. >	102. f	142. Σ	182. ¶	222. Þ	
23. ↑	63. ?	103. g	143. σ	183. •	223. ß	
24. ↓	64. @	104. h	144. τ	184. ×	224. à	
25. ◀	65. A	105. i	145. φ	185. ^{1}	225. á	
26. ▶	66. B	106. j	146. ψ	186. º	226. â	
27. ↑	67. C	107. k	147. Ω	187. »	227. ã	
28. ∪	68. D	108. l	148. ω	188. *d*	228. ä	
29. ∩	69. E	109. m	149. E	189. ∫	229. å	
30. ⊂	70. F	110. n	150. *e*	190. ∞	230. æ	
31. ∈	71. G	111. o	151. ***i***	191. ¿	231. ç	
32. SPACE	72. H	112. p	152. ^{r}	192. À	232. è	
33. !	73. I	113. q	153. T	193. Á	233. é	
34. "	74. J	114. r	154. x̄	194. Â	234. ê	
35. #	75. K	115. s	155. ȳ	195. Ã	235. ë	
36. $	76. L	116. t	156. ≤	196. Ä	236. ì	
37. %	77. M	117. u	157. ≠	197. Å	237. í	
38. &	78. N	118. v	158. ≥	198. Æ	238. î	
39. '	79. O	119. w	159. ∠	199. Ç	239. ï	
40. (	80. P	120. x	160. ..	200. È	240. ð	

TI-92 Key Map

The **getKey()** function returns a number that corresponds to the last key pressed, according to the tables shown in this section. For example, if your program contains a **getKey()** function, pressing [2nd] [F1] will return a value of 268.

Table 1: Key Values for Primary Keys

Key	Modifier							
	None		↑		2nd		♦	
	Assoc.	**Value**	**Assoc.**	**Value**	**Assoc.**	**Value**	**Assoc.**	**Value**
F1	F1	268	F1	268	F1	268		8460
F2	F2	269	F2	269	F2	269		8461
F3	F3	270	F3	270	F3	270		8462
F4	F4	271	F4	271	F4	271		8463
F5	F5	272	F5	272	F5	272		8464
F6	F6	273	F6	273	F6	273		8465
F7	F7	274	F7	274	F7	274		8466
F8	F8	275	F8	275	F8	275		8467
MODE	MODE	266	MODE	266	MODE	266		8458
CLEAR	CLEAR	263	CLEAR	263	CLEAR	263		8455
LN	LN	262	LN	262	e^x	4358		8454
ESC	ESC	264	ESC	264	QUIT	4360		8456
APPS	APPS	265	APPS	265	SWITCH	4361		8457
ENTER	CR	13	CR	13	ENTRY	4109	APPROX	8205
SIN	SIN	259	SIN	259	SIN^{-1}	4355		8451
COS	COS	260	COS	260	COS^{-1}	4356		8452
TAN	TAN	261	TAN	261	TAN^{-1}	4357		8453
^	^	94	^	94	π	140		8286
(	(	40	(	40	{	123		8232
)	)	41	)	41	}	125		8233
,	,	44	,	44	[	91		8236
÷	/	47	/	47	]	93		8239
×	*	42	*	42	√	4138		8234
-	-	45	-	45	VAR-LNK	4141	*Contrast −*	
+	+	43	+	43	CHAR	4139	*Contrast +*	
STO▶	STO▶	258	STO▶	258	RCL	4354		8450
SPACE		32		32		32		8224
=	=	61	=	61	\	92		8253
←	BS	257	BS	257	INS	4353	DEL	8449
θ	θ	136	θ	136	:	58		8328
(-)	-	173	-	173	ANS	4372		8365
.	.	46	.	46	>	62		8238

Table 1: Key Values for Primary Keys (Continued)

Key	Modifier							
	None		↑		2nd		♦	
	Assoc.	Value	Assoc.	Value	Assoc.	Value	Assoc.	Value
0	0	48	0	48	<	60		8240
1	1	49	1	49	E	149		8241
2	2	50	2	50	CATLG	4146		8242
3	3	51	3	51	CUST	4147		8243
4	4	52	4	52	Σ	4148		8244
5	5	53	5	53	MATH	4149		8245
6	6	54	6	54	MEM	4150		8246
7	7	55	7	55	VAR-LNK	4151		8247
8	8	56	8	56	∫	4152		8248
9	9	57	9	57	δ	4153		8249
A	a	97	A	65	*Table 3*			8257
B	b	98	B	66	‘	39		8258
C	c	99	C	67	*Table 4*		COPY	8259
D	d	100	D	68	°	176		8260
E	e	101	E	69	*Table 5*		WINDOW	8261
F	f	102	F	70	∠	159	FORMAT	8262
G	g	103	G	71	*Table 6*			8263
H	h	104	H	72	&	38		8264
I	i	105	I	73	i	151		8265
J		106	J	74	∞	190		8266
K	k	107	K	75	|	124		8267
L	l	108	L	76	“	34		8268
M	m	109	M	77	;	59		8269
N	n	110	N	78	*Table 7*		NEW	8270
O	o	111	O	79	*Table 8*		OPEN	8271
P	p	112	P	80	_	95		8272
Q	q	113	Q	81	?	63	HOME	8273
R	r	114	R	82	@	64	GRAPH	8274
S	s	115	S	83	β	223	SAVE	8275
T	t	116	T	84	#	35	TblSet	8276
U	u	117	U	85	*Table 9*			8277
V	v	118	V	86	≠	157	PASTE	8278
W	w	119	W	87	!	33	Y=	8279
X	x	120	X	88	©	169	CUT	8280
Y	y	121	Y	89	▸	18	TABLE	8281
Z	z	122	Z	90	*Caps Lock*			8282

Table 2: Arrow Keys

Arrow Keys	Normal	⇧	2nd	◆	✋
↑	338	16722	4434	8530	33106
↗	342	16726	4438	8534	33110
→	340	16724	4436	8532	33108
↘	348	16732	4444	8540	33116
↓	344	16728	4440	8536	33112
↙	345	16729	4441	8537	33113
←	337	16721	4433	8529	33105
↖	339	16723	4435	8531	33107

Note: The Grab (✋)modifier only affects the arrow keys.

Table 3: Grave Accent Prefix (2nd A)

Key	Assoc.	Normal	⇧
A	à	224	192
E	è	232	200
I	ì	236	204
O	ò	242	210
U	ù	249	217

Table 4: Cedilla Prefix (2nd C)

Key	Assoc.	Normal	⇧
C	ç	231	199

Table 5: Acute Accent Prefix (2nd E)

Key	Assoc.	Normal	⇧
A	á	225	193
E	é	233	201
I	í	237	205
O	ó	243	211
U	ú	250	218
Y	ý	253	221

Table 6: Greek Prefix ([2nd] G)

Key	Assoc.	Normal	[↑]
A	α	128	
B	β	129	
D	δ	133	132
E	ε	134	
F	φ	145	
G	γ	131	130
L	λ	137	
M	μ	181	
P	π	140	139
R	ρ	141	
S	σ	143	142
T	τ	144	
W	ω	148	147
X	ξ	138	
Y	ψ	146	
Z	ζ	135	

Table 7: Tilde Prefix ([2nd] N)

Key	Assoc.	Normal	[↑]
N	ñ	241	209
O	õ	245	

Table 8: Caret Prefix ([2nd] O)

Key	Assoc.	Normal	[↑]
A	â	226	194
E	ê	234	202
I	î	238	206
O	ô	244	212
U	û	251	219

Table 9: Umlaut Prefix ([2nd] U)

Key	Assoc.	Normal	[↑]
A	ä	228	196
E	ë	235	203
I	ï	239	207
O	ö	246	214
U	ü	252	220
Y	ÿ	255	

Complex Numbers

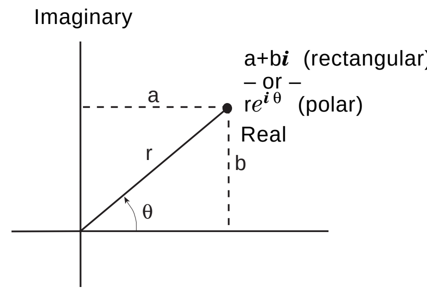
This section describes how to enter complex numbers. It also describes how the Complex Format mode setting affects the way in which complex results are displayed.

Overview of Complex Numbers

A complex number has real and imaginary components that identify a point in the complex plane. These components are measured along the real and imaginary axes, which are similar to the x and y axes in the real plane.

Notice that the point can be expressed in rectangular or polar form.

The ***i*** symbol identifies a complex number.



Important: *To get the* ***i*** *symbol, press* [2nd] [*i*] *(second function of I). Do not simply type an I.*

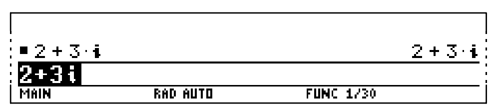
To enter the: | **Use the key sequence:**

Rectangular form
a+b***i***

Substitute the applicable values or variable names for a and b.

a [+] b [2nd] [*i*]

For example:



Important: *To get the e symbol, press* [2nd] [e^x]*. Do not simply type an E.*

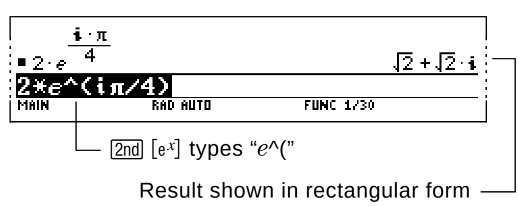
Tip: *To enter θ in degrees, type a ° symbol (such as 45°). To get the ° symbol, type* [2nd] D *or* [2nd] [MATH] 2 1.

Polar form
re^***i*** θ

Substitute the applicable values or variable names for r and θ.

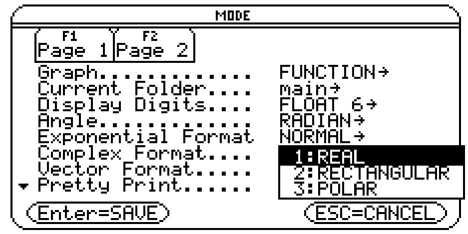
r [2nd] [e^x] [2nd] [*i*] θ [)]

For example:



Complex Format Mode

You can use [MODE] to set the Complex Format mode to one of three settings.



You can enter a complex number at any time, regardless of the Complex Format mode setting. However, the mode setting determines how results are displayed.

If Complex Format is:	The TI-92:
REAL	Will not introduce complex results unless you: Enter a complex number in a calculation. — or — Use a special complex function (**cFactor**, **cSolve**, **cZeros**).
RECTANGULAR or POLAR	Will introduce complex results in the specified form. However, you can enter complex numbers in any form (or a mixture of both forms).

To Use Complex Variables in Symbolic Calculations

Regardless of the Complex Format mode setting, all undefined variables are treated as real numbers in symbolic calculations. To perform complex symbolic analysis, you must define a complex variable. For example:

x+y***i***→z

Then you can use z as a complex variable.

Complex Numbers and Degree Mode

Degree-mode scaling by $\pi/180$ applies only to the trigonometric and inverse trigonometric functions. This scaling does not apply to the related exponential, logarithmic, hyperbolic, or inverse-hyperbolic functions. Consequently, radian-mode identities between these functions are not generally true for degree mode when the inputs or results are non-real. For example, degree-mode scaling is applied to $\cos(\theta) + i \sin(\theta)$ but not to the radian-equivalent expression $e\hat{}(i\theta)$. Radian mode is recommended for complex number calculations.

Accuracy Information

To maximize accuracy, the TI-92 carries more digits internally than it displays.

Computational Accuracy

Floating-point (decimal) values in memory are stored using up to 14 digits with a 3-digit exponent.

- For min and max Window variables (xmin, xmax, ymin, ymax, etc.), you can store values using up to 12 digits. Other Window variables use 14 digits.
- When a floating-point value is displayed, the displayed value is rounded as specified by the applicable mode settings (Display Digits, Exponential Format, etc.), with a maximum of 12 digits and a 3-digit exponent.
- RegEQ displays up to 14-digit coefficients.

Integer values in memory are stored using up to 614 digits.

Graphing Accuracy

__Note:__ For a table that lists the number of pixels in a full screen or split screen, refer to "Setting and Exiting the Split Screen Mode" in Chapter 5.

The Window variable xmin is the center of the leftmost pixel used, and xmax is the center of the rightmost pixel used. Δx is the distance between the centers of two horizontally adjacent pixels.

- Δx is calculated as (xmax − xmin) / (# of x pixels − 1).
- If Δx is entered from the Home screen or a program, xmax is calculated as xmin + Δx ∗ (# of x pixels − 1).

The Window variable ymin is the center of the bottom pixel used, and ymax is the center of the top pixel used. Δy is the distance between the centers of two vertically adjacent pixels.

- Δy is calculated as (ymax − ymin) / (# of y pixels − 1).
- If Δy is entered from the Home screen or a program, ymax is calculated as ymin + Δy ∗ (# of y pixels − 1).

Cursor coordinates are displayed as eight characters (which may include a negative sign, decimal point, and exponent). The coordinate values (xc, yc, zc, etc.) are updated with a maximum of 12-digit accuracy.

System Variables and Reserved Names

This section lists the names of system variables and reserved function names that are used by the TI-92. Only those system variables and reserved function names that are identified by an asterisk (*) can be deleted by using **DelVar** *var* on the entry line.

Graph

y1(x)–y99(x)*	r1(θ)–r99(θ)*	xt1(t)–xt99(t)*	yt1(t)– yt99(t)*
z1(x,y)–z99(x,y)*	u1(n)–u99(n)*	ui1–ui99*	xc
yc	zc	tc	rc
θc	nc	xfact	yfact
zfact	xmin	xmax	xscl
xgrid	ymin	ymax	yscl
ygrid	xres	Δx	Δy
zmin	zmax	zscl	eyeθ
eyeϕ	θmin	θmax	θstep
tmin	tmax	tstep	nmin
nmax	plotStrt	plotStep	sysMath

Graph Zoom

zxmin	zxmax	zxscl	zxgrid
zymin	zymax	zyscl	zygrid
zxres	zθmin	zθmax	zθstep
ztmin	ztmax	ztstep	zzmin
zzmax	zzscl	zeyeθ	zeyeϕ
znmin	znmax	zpltstrt	zpltstep

Statistics

$\bar{x}$	$\bar{y}$	Σx	σx
Σx^2	Σxy	Σy	σy
Σy^2	corr	maxX	maxY
medStat	medx1	medx2	medx3
medy1	medy2	medy3	minX
minY	nStat	q1	q3
regCoef*	regEq(x)*	seed1	seed2
Sx	Sy	R^2	

Table

tblStart	Δtbl	tblInput

Data/Matrix

c1–c99	sysData*

Miscellaneous

main	ok	errornum

EOS™ (Equation Operating System) Hierarchy

This section describes the Equation Operating System (EOS™) that is used by the TI-92. Numbers, variables, and functions are entered in a simple, straightforward sequence. EOS evaluates expressions and equations using parenthetical grouping and according to the priorities described below.

Order of Evaluation

Level	Operator
1	Parentheses ( ), brackets [ ], braces { }
2	Indirection (#)
3	Function calls
4	Post operators: degrees-minutes-seconds (°,','"), factorial (!), percentage (%), radian (^{r}), subscript ([ ]), transpose (^{T})
5	Exponentiation, power operator (^)
6	Negation (⁻)
7	String concatenation (&)
8	Multiplication (∗), division (/)
9	Addition (+), subtraction (−)
10	Equality relations: equal (=), not equal (≠ or /=), less than (<), less than or equal (≤ or <=), greater than (>), greater than or equal (≥ or >=)
11	Logical **not()**
12	Logical **and**
13	Logical **or**, exclusive logical **xor**
14	Constraint "with" operator (|)
15	Store (→)

Parentheses, Brackets, and Braces

All calculations inside a pair of parentheses, brackets, or braces are evaluated first. For example, in the expression 4(1+2), EOS first evaluates the portion of the expression inside the parentheses, 1+2, and then multiplies the result, 3, by 4.

The number of opening and closing parentheses, brackets, and braces must be the same within an expression or equation. If not, an error message is displayed that indicates the missing element. For example, (1+2)/(3+4 will display the error message "Missing)."

Note: Because the TI-92 allows you to define your own functions, a variable name followed by an expression in parentheses is considered a "function call" instead of implied multiplication. For example a(b+c) is the function a evaluated by b+c. To multiply the expression b+c by the variable a, use explicit multiplication: a∗(b+c).

Indirection

The indirection operator (#) converts a string to a variable or function name. For example, #("x"&"y"&"z") creates the variable name xyz. Indirection also allows the creation and modification of variables from inside a program. For example, if 10→r and "r"→s1, then #s1=10.

Post Operators

Post operators are operators that come directly after an argument, such as 5!, 25%, or 60° 15' 45". Arguments followed by a post operator are evaluated at the fourth priority level. For example, in the expression 4^3!, 3! is evaluated first. The result, 6, then becomes the exponent of 4 to yield 4096.

Exponentiation

Exponentiation (^) and element-by-element exponentiation (.^) are evaluated from right to left. For example, the expression 2^3^2 is evaluated the same as 2^(3^2) to produce 512. This is different from (2^3)^2, which is 64.

Negation

To enter a negative number, press [(-)] followed by the number. Post operations and exponentiation are performed before negation. For example, the result of $^{-}x^2$ is a negative number, and $^{-}9^2 = {}^{-}81$. Use parentheses to square a negative number such as $(^{-}9)^2$ to produce 81. Note also that negative 5 ($^{-}$5) is different from minus 5 (−5), and $^{-}$3! evaluates as $^{-}$(3!).

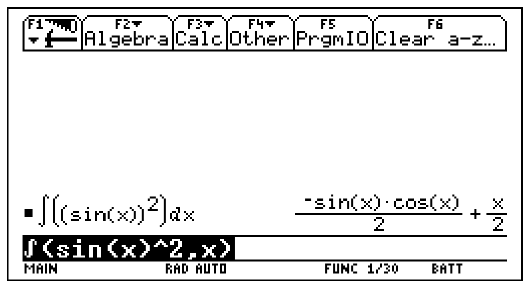
Constraint (|)

The argument following the "with" (|) operator provides a set of constraints that affect the evaluation of the argument preceding the "with" operator.

Service and Warranty Information

C

This appendix provides supplemental information that may be helpful as you use the TI-92. It includes procedures that may help you correct problems with the TI-92, and it describes the service and warranty provided by Texas Instruments.



When the BATT indicator appears in the status line, it is time to change the batteries.

Battery Information

The TI-92 uses two types of batteries: four AA alkaline batteries, and a lithium battery as a backup for retaining memory while you change the AA batteries.

When to Replace the Batteries

As the AA batteries run down, the display will begin to dim (especially during calculations). To compensate for this, you will need to adjust the contrast to a higher setting. If you find it necessary to increase the contrast setting frequently, you will need to replace the AA batteries. To assist you, a BATT indicator (BATT) will display in the status line area when the batteries have drained down to the point when you should replace them soon. When the BATT indicator is displayed in reverse video (BATT), you must replace the AA batteries immediately. You should change the lithium backup battery about once every three years.

Note: To avoid loss of information stored in memory, the TI-92 must be off; also do not remove the AA batteries and the lithium battery at the same time.

Effects of Replacing the Batteries

If you do not remove both types of batteries at the same time or allow them to run down completely, you can change either type of battery without losing anything in memory.

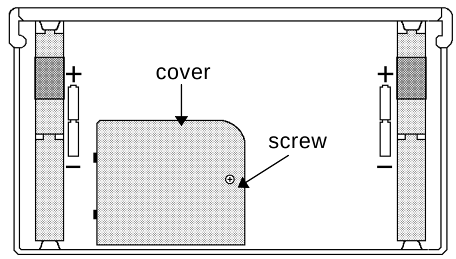
Replacing the AA Batteries

1. Turn the TI-92 off and place the TI-92 face down on a clean surface to avoid inadvertently turning the TI-92 on.
2. Holding the TI-92 unit upright, slide the latch on the top of the unit to the right unlocked position; slide the rear cover down about one-eighth inch and remove it from the main unit. (See the diagrams for installing AA batteries in Chapter 1: Getting Started, if necessary.)
3. To replace the AA alkaline batteries, remove all four discharged AA batteries and install new ones as shown on the polarity diagram located in the battery compartment. (See the opposite page for directions on replacing the lithium battery.)

   CAUTION: Dispose of used batteries properly. Do not incinerate them or leave them within reach of small children.
4. Replace the rear cover, and slide the latch on the top of the TI-92 to the locked position to lock the cover back in place.
5. Turn the TI-92 on, and adjust the display contrast, if necessary.

Replacing the Lithium Battery

1. Turn the TI-92 off and place the TI-92 face down on a clean surface to avoid inadvertently turning the TI-92 on.
2. Holding the TI-92 unit upright, slide the latch on the top of the unit to the right unlocked position; slide the rear cover down about one-eighth inch and remove it from the main unit. (See the diagrams for installing AA batteries in Chapter 1: Getting Started, if necessary.)
3. Loosen and remove the Phillips screw from the cover of the lithium battery compartment, and lift off the cover.



4. Depending on the model of the lithium battery that is in your TI-92, refer to the appropriate illustration below.
5. Loosen the screw and remove the metal clip that holds the lithium battery.

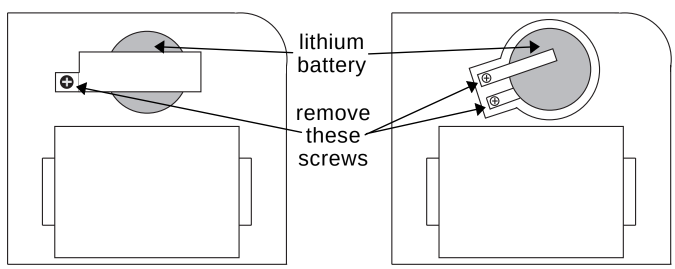


Figure A
Lithium battery: CR 2032

Figure B
See Note below.

6. Remove the old battery and install the new battery, positive (+) side up. Then replace the metal clip and screw.

   CAUTION: Dispose of used batteries properly. Do not incinerate them or leave them within reach of small children.

7. Replace the lithium battery compartment cover, and then replace the rear cover. Slide the latch on the top of the TI-92 to the locked position to lock the cover back in place.
8. Turn the TI-92 on, and adjust the display contrast, if necessary.

Note: If the lithium battery in your TI-92 resembles Figure B, please call 1-800-TI-CARES.

In Case of Difficulty

If you have difficulty operating the TI-92, the following suggestions may help you correct the problem.

Suggestions

If:	Suggested action:
You cannot see anything on the display.	Press [♦] [+] to darken or [♦] [−] to lighten the display contrast.
The BATT indicator is displayed.	Replace the batteries as described on page 496. If BATT is displayed in reverse video (BATT), replace the batteries as soon as possible.
The BUSY indicator is displayed.	A calculation is in progress. If you want to stop the calculation, press [ON].
The PAUSE indicator is displayed.	A graph or program is paused and the TI-92 is waiting for input; press [ENTER].
An error message is displayed.	Refer to Appendix B for a list of error messages. Press [ESC] to clear.
The TI-92 does not appear to be working properly.	Press [ESC] several times to exit any menu or dialog box and to return the cursor to the entry line. — or — Be sure that the batteries are installed properly and that they are fresh.
The TI-92 appears to be "locked up" and will not respond to keyboard input.	Press and hold [2nd] and [hand]. Then press and release [ON]. — or — If [2nd] [hand] and [ON] do not correct the problem: 1. Remove one of the four AA batteries. Refer to page 496. 2. Press and hold [(-)] and [)] as you reinstall the battery. 3. Continue holding [(-)] and [)] for five seconds before releasing.

***Note:** Correcting a "lock up" will reset your TI-92 and clear its memory.*

Support and Service Information

For additional information about TI support, service, and products, please see below.

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

See the information below concerning the warranty for your TI-92.

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 of 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.

General Index

This section contains an alphabetical index to help you find information in this guidebook. To help you distinguish items that refer to interactive geometry from the other TI-92 applications, there is a separate Geometry index that begins on page 516.

F

Geometry Index

This section contains an alphabetical index of only interactive geometry information. Refer to the General Index for all other TI-92 applications.