



Voyage™ 200 Graphing Calculator

Important Information

Texas Instruments makes no warranty, either express 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 product. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

USA 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 to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, 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.

© 2005 Texas Instruments Incorporated

Windows and Macintosh are trademarks of their respective owners.

Getting Started

Initial start-up

Installing the AAA Batteries

The Voyage™ 200 uses four AAA alkaline batteries and a back-up lithium battery (CR1616 or CR1620). The backup battery is already installed, and the AAA batteries are provided with the product.

1. Remove the battery cover from the back of the calculator.
2. Unwrap the four AAA batteries provided with your product and insert them in the battery compartment. Arrange the batteries according to the polarity (+ and -) diagram in the battery compartment.
3. Replace the battery cover on the calculator. The cover should snap into place.

Turning on your Voyage™ 200 for the first time





After installing the batteries included with the calculator, press **ON**. The Apps desktop appears.

Note: If your calculator initializes the preinstalled Apps, a progress bar will appear with the message “Installation in progress . . . Do not interrupt!” instead of the Apps desktop. To avoid losing Apps, do not remove the batteries during initialization. (You can re-install Apps from either the Product CD-ROM or education.ti.com.)

Progress bar



Adjusting the contrast

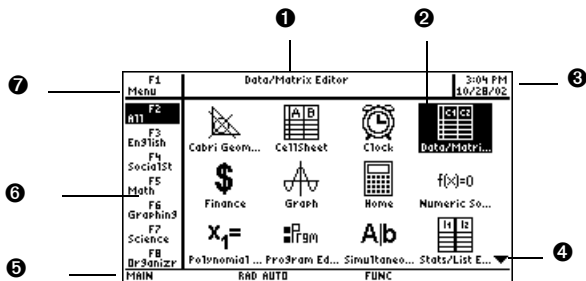
- To lighten the display, press and hold  and tap .
- To darken the display, press and hold  and tap .



The Apps desktop

The Apps desktop is the starting point for operating your Voyage™ 200. Your installed Apps appear on the Apps desktop as icons organized in categories for easy access. From the Apps desktop, you can:

- Open Apps.
- Select and edit categories of Apps.
- View all of the Apps installed on your calculator.
- View the full name of the highlighted App.
- View and edit the time and date.
- Check status line information.
- View split-screen mode information.



Voyage™ 200 Apps desktop

- ❶ View full name of highlighted App.
- ❷ Press **[ENTER]** to open highlighted App.
- ❸ View time and date.
- ❹ Scroll down to view additional Apps.
- ❺ Check status line information.
- ❻ Select category of Apps.
- ❼ Edit categories.

To return to the Apps desktop at any time, press **[APPS]**. The last category selected appears with the last open App highlighted.

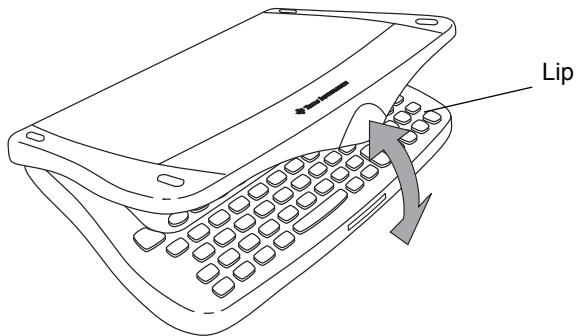
Removing and replacing the cover (Voyage™ 200)

To remove the cover:

1. Hold the unit with one hand.
2. Grasp the cover by the lip with the other hand.

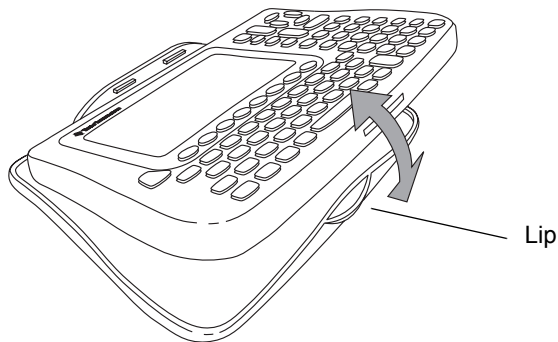
3. Lift the cover by the lip.

To replace the cover, position it over the calculator with the lip in front and snap in place.



Stowing the cover

To stow the cover, position it upside-down beneath the calculator with the lip in front and snap in place.



Turning off the calculator

Press **2nd** [OFF]. The next time you turn on the calculator, the Apps desktop appears with the same settings and memory contents retained. (If you turned off the Apps desktop, the calculator Home screen appears.)

You can use either of the following keys to turn off the Voyage™ 200.

Press:	Description
2nd [OFF] (press 2nd and then press [OFF])	Settings and memory contents are retained by the Constant Memory™ feature. <ul style="list-style-type: none">You cannot, however, use 2nd [OFF] if an error message is displayed.When you turn the Voyage™ 200 on again, it displays either the Home screen or the Apps desktop (regardless of the last application you used).

Press:	Description
\blacklozenge [OFF] (press \blacklozenge and then press [OFF])	Similar to 2nd [OFF] except: <ul style="list-style-type: none">• You can use \blacklozenge [OFF] if an error message is displayed.• When you turn the Voyage™ 200 on again, it will be exactly as you left it.

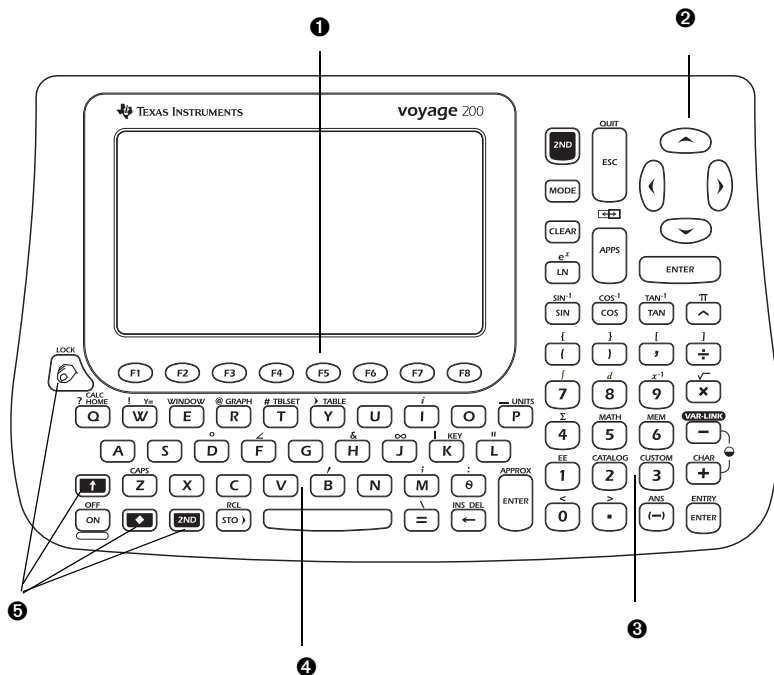
Note: [OFF] is the second function of the ON key.

The calculator's Automatic Power Down™ (APD™) feature prolongs battery life by turning the calculator off automatically following several minutes of inactivity. When you turn on the calculator after APD:

- The display, cursor, and any error conditions are exactly the same as before APD.
- All settings and memory contents are retained.

Note: APD does not occur if a calculation or program is in progress, unless the program is paused. If a program is running but waiting for a key press, APD will occur after several minutes of inactivity.

Voyage™ 200 keys



Voyage™ 200 keys

1 Function keys (**F1**–**F8**) open toolbar menus, access Apps, and edit categories of Apps.

- ② Cursor keys (⬅️, ⬇️, ⬅️, ⬆️) move the cursor.
- ③ Numeric keypad performs math and scientific functions.
- ④ QWERTY keyboard is similar to a computer keyboard.
- ⑤ Modifier keys (2nd, ⬅️, ⬆️, ⬅️) add features by increasing the number of key commands.

QWERTY keyboard

If you're familiar with typing on a computer, you're on your way to using the Voyage™ 200 QWERTY keyboard, with these differences:

- To type a single uppercase alphabetic character, press ⬆️ and the letter key.
- To turn on uppercase lock, press 2nd [CAPS]. To turn off uppercase lock, press 2nd [CAPS] again.

Entering special characters



Use the CHAR (Character) menu and key commands to enter special characters. The CHAR menu lets you access Greek, math, international, and other special characters. An on-screen keyboard map shows the locations of shortcuts used to enter other commonly used characters.

To select characters from the CHAR menu:

1. Press 2nd [CHAR]. The CHAR menu appears.

2. Use the cursor keys to select a category. A submenu lists the characters in that category.
3. Use the cursor keys to select a character, and press **[ENTER]**.

Example: Enter the right arrow symbol (→) in the Text Editor.

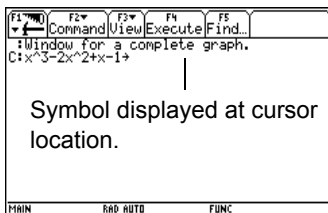
Press	Result
[2nd] [CHAR]	
4	 <p>Scroll down for more characters.</p>

Press**Result**

9

– or –

Press \leftarrow repeatedly to
select **9**: \rightarrow
and press **ENTER**

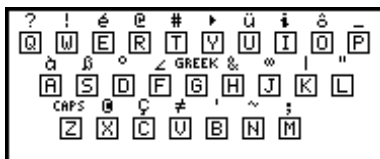


To open the keyboard map, press \blacklozenge [KEY]. The keyboard map appears.

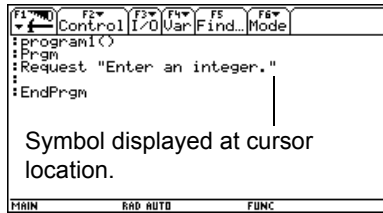
To type most characters, press \blacklozenge and the corresponding key. Press **ESC** to close the map.

Example: Use the keyboard map to find the quotation symbol (") shortcut and enter the symbol in the Program Editor.

Press**Result**

 \blacklozenge [KEY]


Press**Result**

2nd L


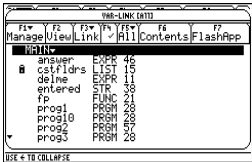
Modifier keys

Modifier keys add features by increasing the number of keyboard operations at your fingertips. To access a modifier function, press a modifier key and then press the key for the corresponding operation.

Keys	Description
2nd (Second)	Accesses Apps, menu options, and other operations. Second functions are printed above their corresponding keys in the same color as the 2nd key.
◆ (Diamond)	Accesses Apps, menu options, and other operations. Diamond functions are printed above their corresponding keys in the same color as the ◆ key.
↑ (Shift)	Types an uppercase character for the next letter key you press. Also used with ↓ and → to highlight characters when editing.

Keys	Description
 (Hand)	Lets you use the cursor keys to manipulate geometric objects. Also used when drawing on a graph.

Example: Access the VAR-LINK [All] screen, where you can manage variables and Apps.

Press	Result
 [VAR-LINK]	

Function keys

Use the function keys to perform the following operations:

- On the Apps desktop, open Apps and select or edit Apps categories.
- On the calculator Home screen, open toolbar menus to select math-related operations.
- Within Apps, open toolbar menus to select App options.

Cursor keys

Pressing \leftarrow , \rightarrow , \uparrow , or \downarrow moves the cursor in the corresponding direction. Depending on the App, and depending on whether the $\boxed{2\text{nd}}$ or $\boxed{\blacklozenge}$ modifier key is used, the cursor keys move the cursor in a variety of ways.

- \uparrow or \downarrow moves the cursor up or down one line at a time.
- $\boxed{2\text{nd}} \leftarrow$ or $\boxed{2\text{nd}} \rightarrow$ moves the cursor to the beginning or end of a line.
- $\boxed{2\text{nd}} \uparrow$ or $\boxed{2\text{nd}} \downarrow$ moves the cursor up or down one page at a time.
- $\boxed{\blacklozenge} \uparrow$ or $\boxed{\blacklozenge} \downarrow$ moves the cursor to the top or bottom of a page.
- \uparrow and \rightarrow , \uparrow and \downarrow , \downarrow and \leftarrow , or \downarrow and \rightarrow moves the cursor diagonally. (Press each pair of indicated cursor keys simultaneously.)

Numeric keypad

The numeric keypad lets you enter positive and negative numbers.

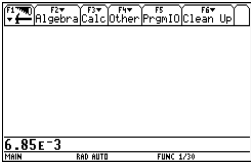
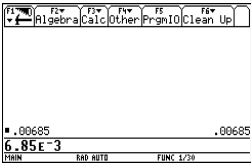
To enter a negative number, press $\boxed{(-)}$ before typing the number.

Note: Don't confuse the negation key ($\boxed{(-)}$) with the subtraction key ($\boxed{-}$).

To enter a number in scientific notation:

1. Type the numbers that precede the exponent. (This value can be an expression.)
2. Press $\boxed{2\text{nd}} \boxed{[EE]}$ The exponent symbol (E) follows the numbers you entered.
3. Type the exponent as an integer with up to three digits. (As the following example shows, you can use a negative exponent.)










Example: On the calculator Home screen, enter 0.00685 using scientific notation.

Press	Result
6 . 8 5	
2nd [EE]	
(-) 3	
ENTER	

Other Important keys

Key Command	Description
◆ [Y=]	Displays the Y= Editor.
◆ [WINDOW]	Displays the Window Editor.
◆ [GRAPH]	Displays the Graph screen.
◆ [TBLSET]	Sets parameters for the Table screen.
◆ [TABLE]	Displays the Table screen.

Key Command	Description
<input type="checkbox"/> X (cut) <input type="checkbox"/> C (copy) <input type="checkbox"/> V (paste)	These keys let you edit entered information by performing a cut, copy, or paste operation.
<input type="checkbox"/> S	Displays the SAVE COPY AS dialog box, prompting you to select a folder and type a variable name to which data entered on the screen is saved.
<input type="checkbox"/> N	Creates new file.
<input type="checkbox"/> O	Opens an existing file you specify.
<input type="checkbox"/> F	Displays the FORMATS or GRAPH FORMATS dialog box, where you enter format information for the active App.
<input type="checkbox"/> APPS	Displays the Apps desktop.
<input type="checkbox"/> <input type="checkbox"/> APPS	With the Apps desktop off, displays the FLASH APPLICATIONS menu.
<input type="checkbox"/> 2nd] <input type="checkbox"/> <input type="checkbox"/>	Switches between the last two chosen Apps.
<input type="checkbox"/> 2nd] <input type="checkbox"/> [CUSTOM]	Turns the custom menu on and off.
<input type="checkbox"/> 2nd] <input type="checkbox"/> <input type="checkbox"/>	Converts measurement units.
<input type="checkbox"/> 2nd] <input type="checkbox"/> [-]	Designates a measurement unit.

Key Command	Description
	Deletes the character to the left of the cursor (backspace).
 [DEL]	Deletes the character to the right of the cursor.
 [INS]	Switches between insert and overwrite modes.
 [MEM]	Displays the MEMORY screen.
 [CATALOG]	Displays a list of commands.
 [RCL]	Recalls the contents of a variable.
	Stores a value to a variable.
 [CHAR]	Displays the CHAR menu, which lets you select Greek letters, international accented characters, and other special characters..
 [QUIT]	<ul style="list-style-type: none"> • In full-screen mode, displays the Apps desktop. • In split-screen mode, displays the full-screen view of the active App. • With the Apps desktop off, displays the calculator Home screen.

Mode settings

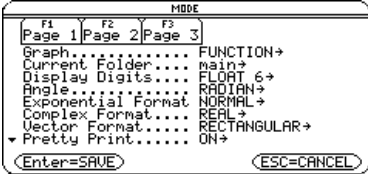
Modes control how the Voyage™ 200 displays and interprets information. All numbers, including elements of matrices and lists, are displayed according to the current mode settings. When the Voyage™ 200 is turned off, the Constant Memory™ feature retains all of the mode settings you have selected.

To view the Voyage™ 200 mode settings:

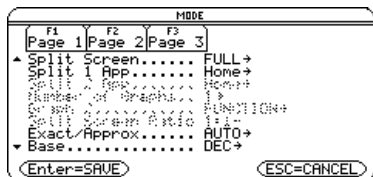
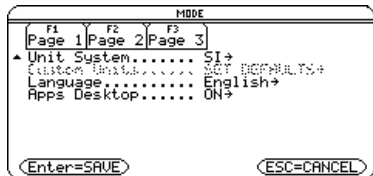
1. Press **[MODE]**. Page 1 of the MODE dialog box appears.
2. Press **[F2]** or **[F3]** to display the modes listed on Page 2 or Page 3.

Note: Modes that are grayed out are available only if other required mode settings are selected. For example, the Custom Units mode listed on Page 3 is available only if the Unit System mode is set to CUSTOM.

Viewing mode settings

Press	Result
[MODE]	 The screenshot shows a dialog box titled "MODE" with three tabs: "Page 1", "Page 2", and "Page 3". The "Page 1" tab is selected. The list of modes includes: Graph (FUNCTION+), Current Folder (MAIN+), Display Digits (FLOAT 6+), Angle (RADIAN+), Exponential Format (NORMAL+), Complex Format (REAL+), Vector Format (RECTANGULAR+), and Pretty Print (ON+). At the bottom, there are two buttons: "Enter=SAVE" and "ESC=CANCEL".

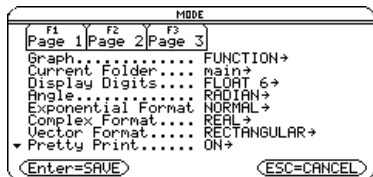
Press**Result**

F2**F3**

Changing mode settings

Example: Change the Language mode setting to Spanish (*Español*).

Press**Result**

MODE

Press

Result

F3



Scroll down to the Language field.



Press ⏴

and then press ⏵ until
3:Español is highlighted.

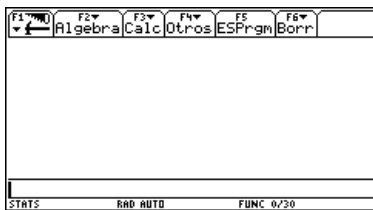
Note: Your menu list might vary, depending on the languages installed.



ENTER



Press**Result**

ENTER

Note: The previous open App appears (in this example, the calculator Home screen).

To return the Language mode setting to English, repeat the steps, selecting **1:English** in the Language field.

Using the Catalog to access commands

Use the Catalog to access a list of Voyage™ 200 commands, including functions, instructions, and user-defined programs. Commands are listed alphabetically. Commands not beginning with a letter are found at the end of the list (&, /, +, -, etc.).

The Catalog Help App includes details about each command.

Options not currently valid are grayed out. For example, the Flash Apps (**F3**) menu option is grayed out if no Flash applications are installed on your Voyage™ 200; the User-Defined (**F4**) menu option is grayed out if you have not created a function or program.

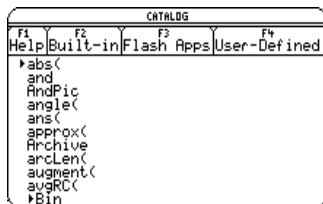
Note: Typing a letter takes you to the first command in the list starting with the same letter.

Press

[2nd] [CATALOG]

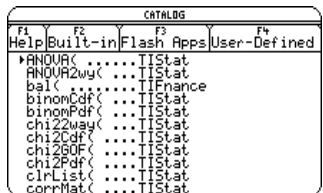
(displays Built-in commands)

Result



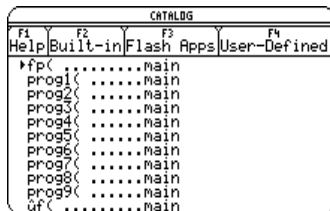
[F3]

(displays Flash Apps commands, if any)



[F4]

(displays User-Defined commands, if any)

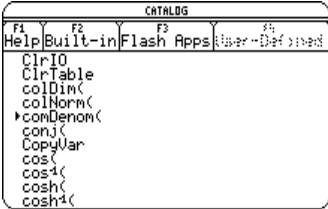
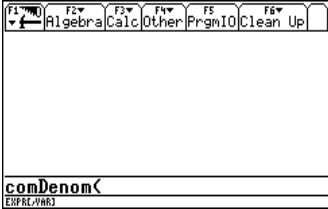


Select commands from the Catalog and insert them onto the calculator Home screen entry line or paste them to other Apps, such as the Y= Editor, Text Editor, or CellSheet™ Apps.

Example: Insert the **comDenom(** command on the calculator Home screen entry line.

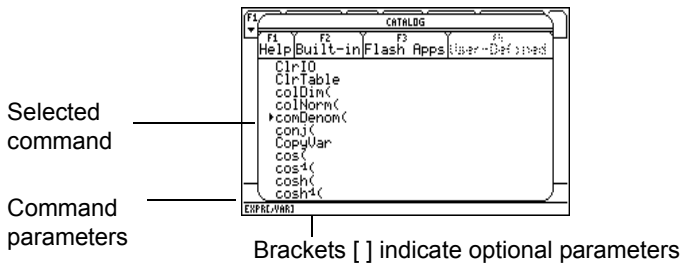
Note: Before selecting a command, position the cursor where you want the command to appear.

Pressing **2nd** \odot advances the Catalog list one page at a time.

Press	Result
2nd [CATALOG] C 2nd \odot	
Then press \odot until the pointer is at the comDenom(function.	
ENTER	

The status line displays any required and optional parameters for the selected command. Optional parameters appear in square brackets.

Note: Pressing **F1** will also display the parameters for the selected command.



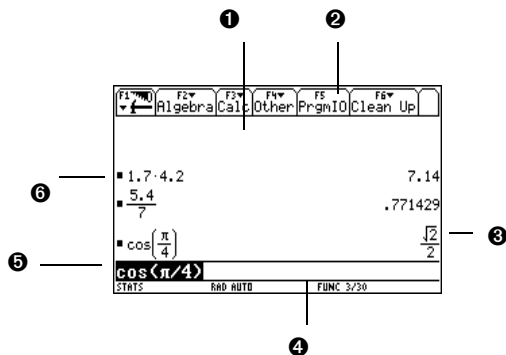
To exit the Catalog without selecting a command, press **[ESC]**.

Calculator Home screen

The calculator Home screen is the starting point for math operations, including executing instructions, evaluating expressions, and viewing results.

To display the calculator Home screen, press: **[♦] [CALC HOME]**.

You can also display the calculator Home screen from the Apps desktop by highlighting the Home icon and pressing **[ENTER]**.



❶ History area lists the entry/answer pairs entered.

❷ Tabs display menus for selecting lists of operations. Press $\boxed{F1}$, $\boxed{F2}$, and so on to display menus.

❸ Result of last entry is displayed here. (Note that results are not displayed on the entry line.)

❹ Status line shows the current state of the calculator.

❺ Entry line displays your current entry.

❻ Your previous entry is displayed here.

To return to the Apps desktop from the calculator Home screen, press $\boxed{\text{APPS}}$.

About the history area

The history area displays up to eight entry/answer pairs, depending on the complexity and height of the expressions. When the display is filled, information scrolls off the top of the screen. Use the history area to:

- Review previous entries and answers. Use the cursor keys to view entries and answers that have scrolled off the screen.
- Recall or auto-paste a previous entry or answer onto the entry line to reuse or edit. (For more information, see the electronic *Operating the Calculator* chapter.)

The cursor, which normally rests on the entry line, can be moved into the history area. The following table shows you how to move the cursor around in the history area.

To	Do this
View entries/answers scrolled off the screen	From the entry line, press \leftarrow to highlight the last answer. Continue using \leftarrow to move the cursor from answer to entry through the history area.
Go to the oldest or newest entry/answer pair	If the cursor is in the history area, press \blacklozenge \leftarrow or \blacklozenge \rightarrow .
View an entry or answer too long for one line ([DEL] is displayed at the end of the line)	Move the cursor to the entry or answer. Use \leftarrow or \rightarrow to scroll left or right and $\boxed{2nd}$ \leftarrow or $\boxed{2nd}$ \rightarrow to go to the beginning or end.
Return cursor to the entry line	Press \boxed{ESC} , or press \downarrow until the cursor is back on the entry line.

Interpreting history information on the status line

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

If the cursor is on the entry line:

Total number of pairs currently saved 8/30 Maximum number of pairs that can be saved

If the cursor is in the history area:

Pair number of the highlighted entry/answer 8/30 Total number of pairs currently saved

Modifying the history area


To change the number of pairs that can be saved:

1. From the calculator Home screen, press **[F1]** and select **9:Format**.
2. Press **⏴** and use **⏪** or **⏩** to highlight the new number.
3. Press **[ENTER]** **[ENTER]**.

To clear the history area and delete all saved pairs:

- From the calculator Home screen, press **[F1]** and select **8:Clear Home**.
– or –

- Enter **ClrHome** on the calculator Home screen entry line.

To delete an entry/answer pair, move the cursor to either the entry or answer, and press  or **CLEAR**.

Working with Apps

The Voyage™ 200 organizes Apps by category on the Apps desktop. To select a category, press a function key (**F2** through **F8**). The App icons for the selected category appear on the Apps desktop.

Note: If the name under an Apps desktop icon is truncated, use the cursor keys to highlight the icon. Now view the full name at the top of the Apps desktop.

Opening Apps

Use the cursor keys or press the first letter of the App name to highlight the Apps icon on the Apps desktop and press **ENTER**. The App either opens directly or displays a dialog box. The most common dialog box lists these options for the App:


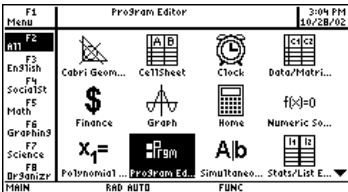





Note: The Voyage™ 200 uses the general term *variable* to refer to the App data files that you create.

Option	Description
Current	Returns the screen displayed when you last viewed the App. If no current App variable exists, the New dialog box appears.
Open	Lets you open an existing file.

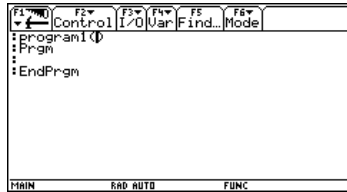
Option	Description
New	Creates a new file with the name typed in the field.

Select an option, enter any required information, and press **[ENTER]**. The App appears.

Example: Create a new program using the Program Editor.

Press	Result
Use cursor keys to highlight  Program Ed...	
[ENTER]	
3	
[ENTER]	
 program 1	

Press**Result**

ENTER **ENTER**

The newly created program variable, *program1*, is saved to the Main folder.

Returning to the Apps desktop from within an App

Press **[APPS]**. The icons for the last Apps category selected appear on the Apps desktop with the icon for the last App opened highlighted.

You can also return to the Apps desktop by pressing **[2nd] [QUIT]** in full-screen mode. In split-screen mode, press **[2nd] [QUIT]** twice.

To return to the last open App from the Apps desktop, press **[2nd] [⇐]**.

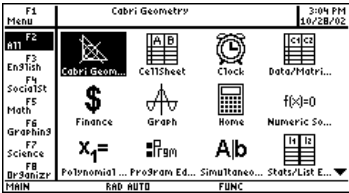
Selecting an Apps category

On the Voyage™ 200, the Apps category names appear along the left side of the Apps desktop. To select an Apps category, press the corresponding function key (displayed above the category name on the Apps desktop).

The App icons for the selected category appear on the Apps desktop.

Key	Description
F2 All	Icons for all installed Apps displayed. Not customizable.
F3 English	Customizable category. English is the default.
F4 SocialSt	Customizable category. SocialSt (social studies) is the default.
F5 Math	Customizable category. Math is the default.
F6 Graphing	Customizable category. Graphing is the default.
F7 Science	Customizable category. Science is the default.
F8 Organizr	Customizable category. Organizr (organizer) is the default.

Example: Select the All category.

Press	Result
F2	

If you select an Apps category containing no Apps, a message appears to confirm that the category is empty and point you to the **F1** 1:Edit Categories menu, where you can add App shortcuts to the category. (See [“Customizing the Apps categories”](#) on page 34.)

Press **[ENTER]** or **[ESC]** to clear the message and return to the Apps desktop.

Customizing the Apps categories

The Voyage™ 200 organizes your Apps into seven categories, six of which you can customize to fit your individual needs. (The All category contains every installed App and cannot be edited.)

To customize the **[F3]** through **[F8]** Apps categories:

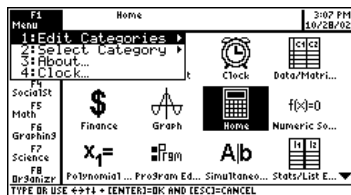
1. Select **[F1] 1:Edit Categories**. A submenu displays the six customizable Apps category names. (The All category is not listed.)
2. Highlight an Apps category and press **[ENTER]**. The Edit Categories dialog box appears with a list of installed Apps and a text box with the category name highlighted.
3. To change the Apps category name, type the desired name.
Note: Enter a name of up to eight characters, including letters with or without capitalization, numbers, punctuation, and accented characters.
4. To add or remove an App shortcut from the category, press **☞** as required to highlight the box next to the App, then press **⬇** to add or remove the check mark (✓).
5. To save the changes and return to the Apps desktop, press **[ENTER]**.

Example: Replace the Social Studies category with the Business category and add the CellSheet™ and Finance App shortcuts.

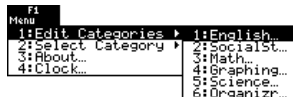
Press

F1

Result



↓



2

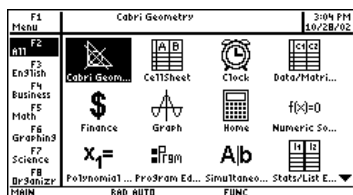
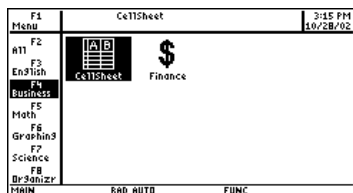
– OR –

⏏ ENTER



⏏ **Business**



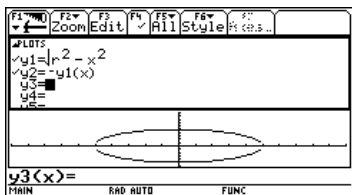
Press**Result****ENTER****F4**

Open Apps and split-screen status

Your Voyage™ 200 lets you split the screen to view two Apps simultaneously. For example, view the Y= Editor and Graph screens simultaneously to see the list of functions and how they are graphed.

Select the Split Screen mode from Page 2 of the MODE screen. The Voyage™ 200 displays the selected Apps in the split-screen view as shown. Split the screen horizontally (top-bottom) or vertically (left-right).

Top-bottom split screen

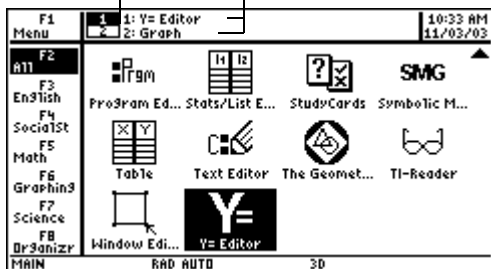


To return to the Apps desktop, press **[APPS]**. The split-screen status appears at the top of the Apps desktop with the names of the open Apps and the portions of the screen in which each is displayed. The highlighted numeral indicates the split-screen portion where the next App you open will appear.

Note: The Apps desktop always appears in the full-screen view.

Split-screen status (highlight indicates the portion where the next App selected will open.)

Names of open Apps







More information is available about using split screens. (For more information, see the electronic *Split Screens* chapter.)

Checking status information

Look to the status line, located at the bottom of the screen, for information about the current state of your Voyage™ 200.

MAIN	2ND	DRAD	AUTO	GR#1	FUNC	22/30	BATT	99/99	
1	2	3	4	5	6	7	8	9	10

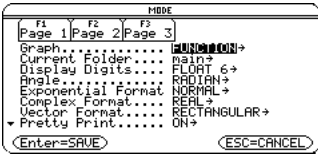
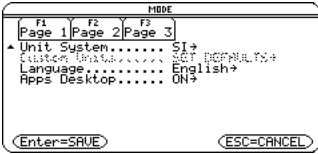
Indicator	Meaning
1	Current folder Name of the selected folder (MAIN is the default folder.)
2	Modifier key Selected modifier key (2nd,  , ) , if any.
3	Hand key The  modifier key has been selected.
4	Angle mode Selected units in which angle values are displayed and interpreted (RAD, DEG, GRAD)
5	Exact/Approx mode Mode in which answers are calculated and displayed (AUTO, EXACT, APPROX)
6	Graph number Active of two independent graphs in split-screen mode (GR#1, GR#2)
7	Graph mode Selected type of graph that can be plotted (FUNC, PAR, POL, SEQ, 3D, DE)
8	Entry/Answer pairs 22/30—Number of entry/answer pairs (default is 30, maximum is 99) in the history area of the calculator Home screen.
9	Replace batteries Displayed when batteries are low (BATT). If BATT is highlighted with a black background, change the batteries as soon as possible  .

Indicator	Meaning
Ⓢ Busy/Pause, Locked/Archived variable	BUSY—Calculation or graph is in progress PAUSE—You paused a graph or program 🔒—Variable opened in the current editor is locked or archived and cannot be modified

Turning off the Apps desktop

You can turn off the Apps desktop from the MODE dialog box. If you do, open Apps from the APPLICATIONS menu. To open the APPLICATIONS menu, press **[APPS]**.

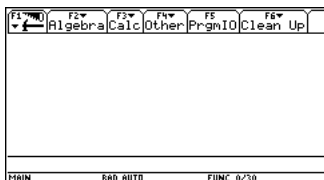
Example: Turn off the Apps desktop.

Press	Result
[MODE]	 <p>The screenshot shows the MODE dialog box with the Applications menu open. The menu items are: Graph, Current Folder, Display Digits, Angle, Exponential Format, Complex Format, Vector Format, and Pretty Print. The Applications menu is open, showing: ON, main, FLOAT 6, RADIAN, NORMAL, REAL, RECTANGULAR, and ON.</p>
[F3]	 <p>The screenshot shows the MODE dialog box with the Apps Desktop option selected. The menu items are: Unit System, Custom Input, Language, and Apps Desktop. The Apps Desktop menu is open, showing: SI, SET DEFAULTS, English, and ON.</p>

Press**Result**



Note: The previous open App appears (in this example, the calculator Home screen).



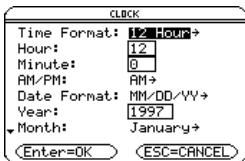
To turn on the Apps desktop, repeat the procedure, selecting ON in the Apps Desktop mode field. To return to the Apps desktop from the calculator Home screen, press **[APPS]**.

Using the clock

Use the **CLOCK** dialog box to set the time and date, select the clock display format, and turn the clock off and on.

The clock is turned on by default. If you turn off the clock, all Clock dialog box options except Clock ON/OFF are grayed out.

▼ indicates you can scroll down for more options)



Displaying the CLOCK dialog box

1. Use the cursor keys to highlight the Clock icon on the Apps desktop.
2. Press **[ENTER]**. The CLOCK dialog box appears with the Time Format field highlighted.

Note: Because the CLOCK dialog box displays the settings current at the time you open the dialog box, you might need to update the time before exiting.

Setting the time

1. Press **[↓]** to open the list of time formats.
2. Press **[←]** or **[→]** to highlight an option, then press **[ENTER]**. The selected format appears in the Time Format field.
3. Press **[↓]** to highlight the Hour field.
4. Type the hour, then press **[↓]** to highlight the Minute field.
5. Type the minute(s).





- If the time format is 24 hours, proceed to step 9.
— or —
If the time format is 12 hours, press \odot to highlight the AM/PM field.
- Press \odot to open the list of AM/PM options.
- Press \odot or \odot to highlight an AM/PM option, then press **ENTER**. The selected AM/PM option appears.
- Set the date (for procedures, see *Setting the date*).
— or —
To save your settings and exit, press **ENTER**. The time is updated in the top right corner of the Apps desktop.

Setting the date

- Press \odot or \odot as required to highlight the Date Format field.
- Press \odot to open the list of date formats.
- Press \odot or \odot to highlight an option, then press **ENTER**. The selected format appears in the Date Format field.
- Press \odot to highlight the Year field.
- Type the year, then press \odot to highlight the Month field.
- Press \odot to open the list of months.
- Press \odot or \odot to highlight an option, then press **ENTER**. The selected month appears in the Month field.
- Press \odot to highlight the Day field.

9. Type the day, then press **ENTER** **ENTER** to save your settings and exit. The date is updated in the top right corner of the Apps desktop.

Example: Set the time and date to 19/10/02 (October 19, 2002) at 1:30 p.m.

Press	Result
Use cursor keys to highlight  Clock	Time and date 
ENTER	
1	

Press

Result

30 ↵

CLOCK

Time Format: 12 Hour→
Hour: 1
Minute: 30
AM/PM: AM→
Date Format: MM/DD/YY→
Year: 1997
Month: January→

Enter=OK ESC=CANCEL

▶ ↵

CLOCK

Time Format: 12 Hour→
Hour: 1
Minute: 30
AM/PM: 1:AM
Date Format: 2:PM→
Year: 1997
Month: January→

Enter=OK ESC=CANCEL

ENTER ↵

CLOCK

Time Format: 12 Hour→
Hour: 1
Minute: 30
AM/PM: PM→
Date Format: MM/DD/YY→
Year: 1997
Month: January→

Enter=OK ESC=CANCEL

▶ ↵

CLOCK

Time Format: 12 Hour→
Hour: 1
Minute: 30
AM/PM: 1:MM-DD-YY
Date Format: 2:MM-DD-YY
Year: 3:MM-DD-YY
Month: 4:DD-MM-YY
5:YY-MM-DD
6:MM-DD-YY
7:DD-MM-YY
8:YY-MM-DD

Enter=OK

Press

Result

ENTER ▾

CLOCK

Time Format: 12 Hour→
Hour: 1
Minute: 30
AM/PM: PM→
Date Format: DD/MM/VY→
Year: 1997
▼ Month: January→

Enter=OK ESC=CANCEL

2002

CLOCK

Time Format: 12 Hour→
Hour: 1
Minute: 30
AM/PM: PM→
Date Format: DD/MM/VY→
Year: 2002
▼ Month: January→

Enter=OK ESC=CANCEL

⬇ ⬆

CLOCK

Time Format: 12 Hour→
Hour: 1
▼ Month: 1: January
2: February
3: March
4: April
5: May
6: June
7: July
8: August
9: September
A: October
B: November
C: December

Enter=OK

Scroll down to October
and press ENTER

CLOCK

Time Format: 12 Hour→
Hour: 1
Minute: 30
AM/PM: PM→
Date Format: DD/MM/VY→
Year: 2002
▼ Month: October→

Enter=OK ESC=CANCEL

Press**Result**

⏪ 19

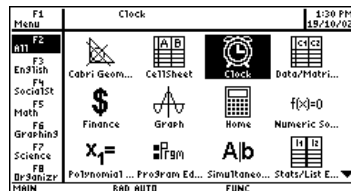
CLOCK

Hour: 1
Minute: 30
AM/PM: PM
Date Format: DD/MM/VV
Year: 2002
Month: October
Day: 19

Enter=OK ESC=CANCEL

ENTER **ENTER**

Revised time and date



Turning off the clock

From the Apps desktop, open the CLOCK dialog box and select OFF in the Clock field.

Example: Turn off the clock.

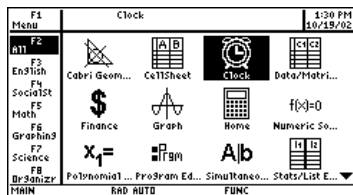
Press

Result

Use cursor keys to highlight



Click on



ENTER

Scroll down to the Clock field.



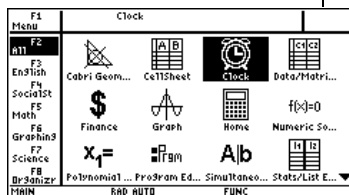
ENTER



Press**Result**

ENTER

Clock off



To turn on the clock, repeat the procedure, selecting ON in the Clock field. Remember to reset the time and date.

Using menus

To select most Voyage™ 200 menus, press the function keys corresponding to the toolbars at the top of the calculator Home screen and most App screens. Select other menus using key commands.

Toolbar menus

The starting point for Voyage™ 200 math operations, the calculator Home screen displays toolbar menus that let you choose math-related options.

Toolbar menus also appear at the top of most App screens. These menus list common functions of the active App.

Other menus

Use key commands to select the following menus. These menus contain the same options regardless of the screen displayed or the active App.


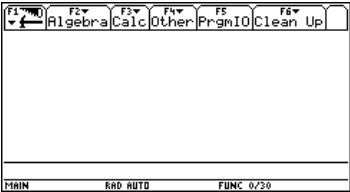
Press	To display
2nd [CHAR]	CHAR menu. Lists characters not available on the keyboard; characters are organized by category (Greek, math, punctuation, special, and international).
2nd [MATH]	MATH menu. Lists math operations by category.
APPS	APPLICATIONS menu. Lists the installed Apps. (Menu is available only when the Apps desktop is turned off; Apps are normally accessed from the Apps desktop.)
◆ APPS	FLASH APPLICATIONS menu. Lists the installed Flash Apps. (Menu is available only when Apps desktop is turned off; Flash Apps are normally accessed from the Apps desktop.)


Selecting menu options

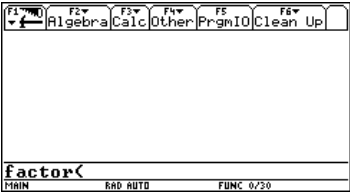
- Press the number or letter to the left of the option you want to select.
— or —
- Press **⤴** or **⤵** to select the option, and press **[ENTER]**.

Note: If the first menu option is selected, press **⤴** to select the last option on the menu. If the last menu option is selected, press **⤵** to select the first option on the menu.

Example: Select **factor()** from the Algebra menu on the calculator Home screen.

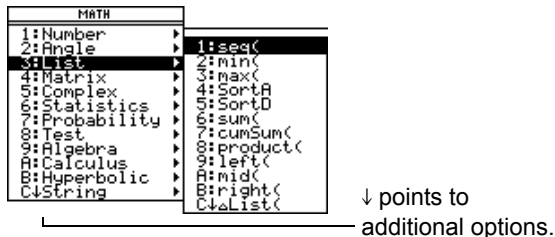
Press	Result
Press: ◊ [CALC HOME] – or – From the Apps desktop, use the cursor keys to highlight  Home and press [ENTER]	

[F2]		▼ indicates Algebra menu will open when you press [F2].
------	---	--

2 – or – ◀ [ENTER]	
------------------------------	---


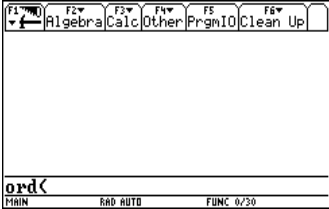
Selecting submenu options

A small arrow symbol (▶) to the right of a menu option indicates that selecting the option will open a submenu.



Example: Select `ord()` from the MATH menu on the calculator Home screen.

Press	Result
<code>[2nd] [MATH]</code>	

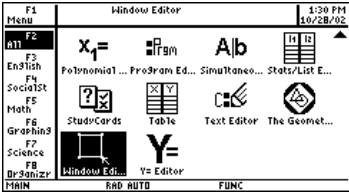

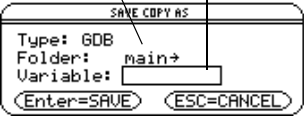
Press	Result
D – or – ⬅ ➡ ⬇	
B – or – ⬅ ENTER	

Using dialog boxes

An ellipsis (...) at the end of a menu option indicates that choosing the option will open a dialog box. Select the option and press **ENTER**.



Example: Open the **SAVE COPY AS** dialog box from the Window Editor.

Press	Result
<p>Use the cursor keys to highlight Window Edi... and press ENTER</p>	
<p>F1</p>	
<p>2 – or – ENTER</p>	<p>Press ↓ to display a list of folders. Type the name of the variable.</p> 
	<p>Press ENTER twice to save and close the dialog box.</p>

Note: Pressing the **◆ S** key shortcut also opens the **SAVE COPY AS** dialog box in most Apps.

Canceling a menu

To cancel a menu without making a selection, press **[ESC]**.

Moving among toolbar menus

To move among the toolbar menus without selecting a menu option:

- Press the function key (**[F1]** through **[F8]**) of a toolbar menu.
- Press a function key, then press **▶** or **◀** to move from one toolbar menu to the next. Press **▶** from the last menu to move to the first menu. Press **◀** to move from the first menu to the last menu.

Note: If you press **▶** when a menu option with a submenu is selected, the submenu will appear instead of the next toolbar menu. Press **▶** again to move to the next menu.

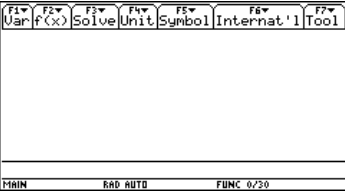
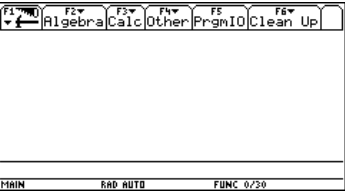
More information is available about menus. (See the electronic *Operating the Calculator* chapter.)

Custom menu

The custom menu provides quick access to your most commonly used options. Use the default custom menu or create your own using the Program Editor. You can include any available Voyage™ 200 command or character.

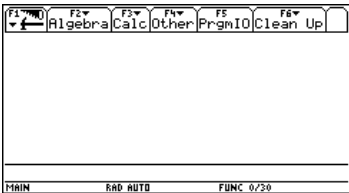

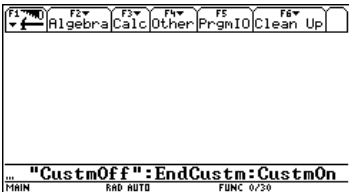
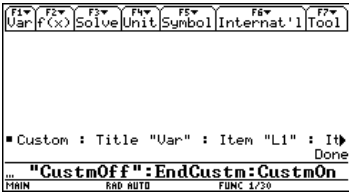
The custom menu replaces the standard toolbar menu on the calculator Home screen. (For details on creating a custom menu, see the electronic *Programming* chapter.) More information is available about custom menus. (See the electronic *Operating the Calculator* chapter.)

Example: Turn on and turn off the custom menu from the calculator Home screen.

Press	Result
$\boxed{2\text{nd}} \text{ [CUSTOM]}$	<p data-bbox="472 215 702 238">Default custom menu</p> 
$\boxed{2\text{nd}} \text{ [CUSTOM]}$	<p data-bbox="472 479 702 502">Normal toolbar menu</p> 

Example: Restore the default custom menu.

Note: Restoring the default custom menu erases the previous custom menu. If you created the previous custom menu with a program, you can run the program again to reuse the menu.


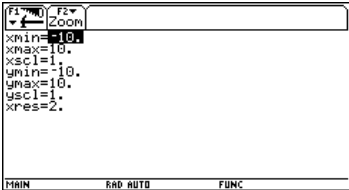
Press	Result
<p>2nd [CUSTOM] (to turn off the custom menu and turn on the standard toolbar menu)</p>	
<p>F6</p>	
<p>3 – or – ⏏ ⏏ [ENTER]</p>	
<p>[ENTER]</p>	

Opening Apps with the Apps desktop turned off

If you turn off the Apps desktop, use the APPLICATIONS menu to open Apps. To open the APPLICATIONS menu with the Apps desktop off, press **[APPS]**.

Note: If you press **[APPS]** with the Apps desktop turned on, the Apps desktop will appear instead of the APPLICATIONS menu.

Example: With the Apps desktop turned off, open the Window Editor from the APPLICATIONS menu.

Press	Result
[APPS]	
3 – or – [LEFT] [RIGHT] [ENTER]	

To access Apps not listed on the APPLICATIONS menu, select **1:FlashApps**.

Using split screens

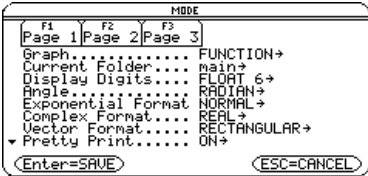
The Voyage™ 200 lets you split the screen to show two Apps at the same time. For example, display both the Y= Editor and Graph screens to compare the list of functions and how they are graphed.

Setting split-screen mode

You can split the screen either top to bottom or left to right from the MODE dialog box. The split-screen setting stays in effect until you change it.

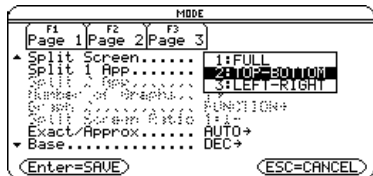
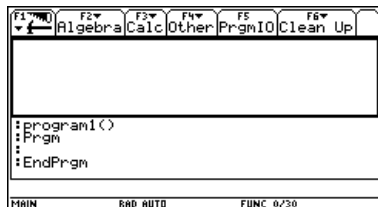
1. Press **[MODE]** to display the MODE dialog box.
2. Press **[F2]** to display the Split Screen mode setting.
3. Press **[↓]** to open the Split Screen mode menu.
4. Press **[→]** as required to highlight either TOP-BOTTOM or LEFT-RIGHT.
5. Press **[ENTER]**. The Split Screen mode setting displays the option you selected.
6. Press **[ENTER]** again to save this change and display the split screen.

Example: Set split-screen mode to TOP-BOTTOM.

Press	Result
[MODE]	

Press**Result****F2**

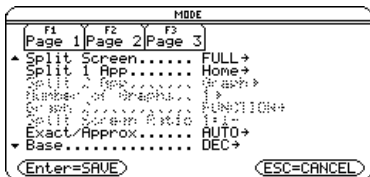
⏪ ⏩

**ENTER****ENTER**

Setting the Initial Apps for split screen

After you select either TOP-BOTTOM or LEFT-RIGHT split-screen mode, additional mode settings become available.

Full-screen mode



Split-screen mode



Mode	Description
Split 2 App	Lets you specify the App displayed in the bottom or right portion of the split screen. Works together with Split 1 App, which lets you specify the App displayed in the top or left portion of the split screen.
Number of Graphs	Lets you set up and display two independent graphs.
Split Screen Ratio	Lets you change the ratio of the split-screen views.

To set the initial App for each split-screen portion:

1. Select the Split 1 App mode setting and press \blacktriangleright to display a menu of available Apps. (See “[Setting split-screen mode](#)” on page 59.)
2. Press \blacktriangleleft or \blacktriangleright to highlight the App and press **[ENTER]**.
3. Repeat steps 1 and 2 for the Split 2 App mode setting.

Example: Display the Y= Editor in the top screen and the Graph App in the bottom screen.

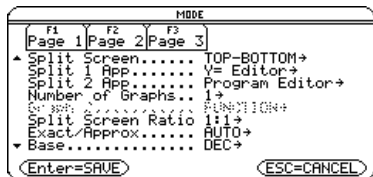
Press



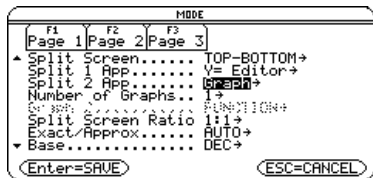
Result



2

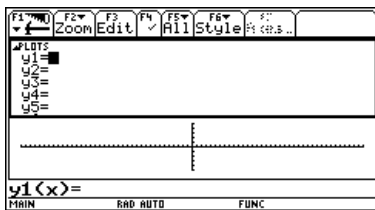


4



Press**Result**

[ENTER]



If you set Split 1 App and Split 2 App to the same nongraphing App or to the same graphing App with Number of Graphs set to 1, the Voyage™ 200 exits split-screen mode and displays the App in full-screen mode.

Selecting the active App

In split-screen mode, only one App can be active at a time.

- To switch between active Apps, press [2nd] [⇄].
- To open a third App, press [APPS] and select the App. This App replaces the active split-screen App.

Exiting split-screen mode

Exit split-screen mode in any of the following ways:

- Press [2nd] [QUIT] to close the active App and display the full-screen view of the other open App.

- If the Apps desktop is turned off, pressing **[2nd] [QUIT]** replaces the active split-screen App with the calculator Home screen. Pressing **[2nd] [QUIT]** again turns off the split-screen mode and displays the calculator Home screen in full-screen mode.
- Select Split Screen on Page 2 of the MODE dialog box, set split-screen mode to FULL, and press **[ENTER]**.
- Press **[2nd] [QUIT]** twice to display the Apps desktop

More information is available about using split screens. (See the electronic *Split Screens* chapter.)

Managing Apps and operating system (OS) versions

Using the Voyage™ 200 connectivity features, you can download Apps from:

- The TI Educational & Productivity Solutions (E&PS) Web site at: education.ti.com/latest
- The CD-ROM included with your Voyage™ 200.
- A compatible graphing calculator.

Adding Apps to your Voyage™ 200 is like loading software on a computer. All you need is TI Connect™ software and the the USB Silver Edition cable that came with your Voyage™ 200.

For system requirements and instructions to link to compatible calculators and download TI Connect software, Apps, and OS versions, see the TI E&PS Web site.

Before downloading Apps to your Voyage™ 200, please read the license agreement on the CD-ROM or TI Web site.

Finding the OS version and Identification (ID) numbers

If you purchase software from the TI E&PS Web site or call the customer support number, you will be asked to provide information about your Voyage™ 200. You will find this information on the ABOUT screen.

To display the ABOUT screen, press **[F1] 3:About** from the Apps desktop. The ABOUT screen displays the following information about your Voyage™ 200:



1 OS version

2 Hardware version

3 Unit ID (required to obtain certificates for installing purchased Apps). Similar to a serial number. Write this number down and keep it in a safe place in case the calculator is ever lost or stolen.

4 Apps certificate revision number (Cert. Rev.)

5 Product identifier (Product ID). Similar to a model number.

Note that your screen will be different than the one shown above.

Deleting an Application

Deleting an application removes it from the Voyage™ 200 and increases space for other applications. Before deleting an application, consider storing it on a computer for reinstallation later.

1. Quit the application.
2. Press **[2nd]** **[VAR-LINK]** to display the VAR-LINK (All) screen.
3. Press **[F7]** to display the list of installed applications.
4. Select the application you want to delete by pressing **[F4]**. (Press **[F4]** again to deselect.)
5. Press **[F1]** **1:Delete**. The VAR-LINK delete confirmation dialog box displays.
6. Press **[ENTER]** to delete the application.

Note: Only Flash Apps can be deleted.

Connecting your Voyage™ 200 to other devices

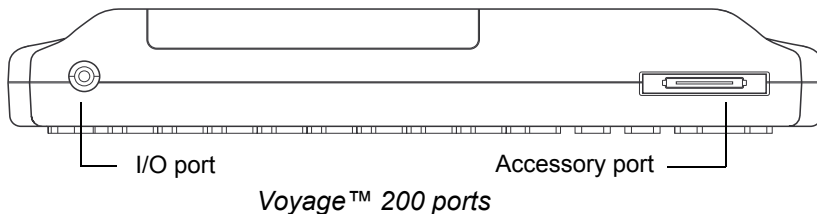
The Voyage™ 200 includes a standard I/O port. Ports are used to link two compatible graphing calculators or connect to a computer or peripheral device.

In addition, any Voyage™ 200 calculator includes an accessory port. This port is used to output visual data so that a classroom can view the calculator's display on a video device or overhead screen.

To connect your calculator to a computer – Connect your Voyage™ 200 using the I/O port and the included USB Silver Edition cable.

To connect your calculator to another calculator – Use the an I/O unit-to-unit cable to connect the Voyage™ 200 to a compatible graphing calculator or peripheral device, such as a TI-89 or TI-92 Plus graphing calculator or the CBL 2™ and CBR™ systems.

To show your calculator's display to the classroom – Use the accessory port to connect the TI-Presenter™ video adapter to the Voyage™ 200. The TI-Presenter video adapter provides a video interface between the calculator and video display or recording devices. Or use the accessory port to connect the TI ViewScreen™ overhead panel to your calculator. The TI ViewScreen overhead panel enlarges and projects the display so an entire class can view it. For more information about the TI-Presenter video adapter and TI ViewScreen panel, see the TI E&PS Web site at education.ti.com.



Batteries

The Voyage™ 200 use four AAA alkaline batteries and a back-up lithium battery (CR1616 or CR1620). The backup battery is already installed, and the AAA batteries are provided with your product.

Important OS download information

New batteries should be installed before beginning an OS download.

When in OS download mode, the APD™ feature does not function. If you leave your calculator in download mode for an extended time before you actually start the download, your batteries may become depleted. You will then need to replace the depleted batteries with new batteries before downloading.

You can also transfer the OS to another Voyage™ 200 using an I/O unit-to-unit cable. If you accidentally interrupt the transfer before it is complete, you will need to reinstall the OS via a computer. Again, remember to install new batteries before downloading.

Please contact Texas Instruments as described in Service & Support Information, if you experience a problem.

Battery Precautions

Take these precautions when replacing batteries:

- Do not leave batteries within the reach of children.
- Do not mix new and used batteries. Do not mix brands (or types within brands) of batteries.
- Do not mix rechargeable and non-rechargeable batteries.
- Install batteries according to polarity (+ and –) diagrams.
- Do not place non-rechargeable batteries in a battery recharger.
- Properly dispose of used batteries immediately.
- Do not incinerate or dismantle batteries.

Installing the AAA Batteries

1. Remove the battery cover from the back of the calculator.
2. Unwrap the four AAA batteries provided with your product and insert them in the battery compartment. Arrange the batteries according to the polarity (+ and -) diagram in the battery compartment.
3. Replace the battery cover on the calculator. The cover should snap into place.

Replacing the AAA (alkaline) batteries

As the batteries lose power, the display begins to dim, especially during calculations. If you find yourself increasing the contrast frequently, replace the AAA alkaline batteries.

The status line also gives battery information.

Indicator	Meaning
BATT	Batteries are low.
BATT	Replace batteries as soon as possible.

Before replacing the batteries, turn off the Voyage™ 200 by pressing **2nd** **[OFF]** to avoid losing information stored in memory. Do not remove both the back-up battery and the AAA alkaline batteries at the same time.

Previews

Performing Computations

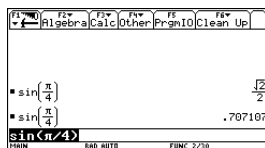
This section provides several examples for you to perform from the Calculator Home screen that demonstrate some of the computational features of the Voyage™ 200. The history area in each screen was cleared by pressing **[F1]** and selecting **8:Clear Home**, before performing each example, to illustrate only the results of the example's keystrokes.

Showing Computations

Steps and keystrokes

Compute $\sin(\pi/4)$ and display the result in symbolic and numeric format. To clear the history area of previous calculations, press **[F1]** and select **8:Clear Home**.

Display



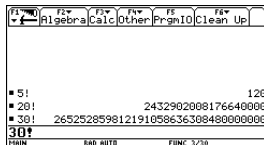
Press **[SIN]** **[2nd]** **[π]** **[÷]** **4** **[)]** **[ENTER]** **[♦]** **[≈]**

Finding the Factorial of Numbers

Steps and keystrokes

Compute the factorial of several numbers to see how the Voyage™ 200 handles very large integers. To get the factorial operator (!), press **2nd** **[MATH]**, select **7:Probability**, and then select **1:!**.

Display



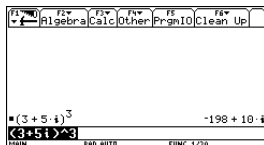
Press 5 **2nd** **[!]** **ENTER** 20 **2nd** **[!]** **ENTER** 30 **2nd** **[!]** **ENTER**

Expanding Complex Numbers

Steps and keystrokes

Compute $(3+5i)^3$ to see how the Voyage™ 200 handles computations involving complex numbers.

Display



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

Finding Prime Factors

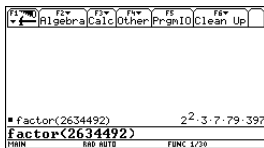
Steps and keystrokes

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 $\boxed{F2}$ and selecting **2:factor(**.

Press $\boxed{F2}$ 2 2634492 $\boxed{\rightarrow}$ \boxed{ENTER}

(Optional) Enter other numbers on your own.

Display



Finding Roots

Steps and keystrokes

Find the root of the expression (x,y). You can enter “root” on the entry line by typing **ROOT** on the keyboard, or by pressing $\boxed{\blacklozenge}$ 9.

This example illustrates using the root function and how the expression is displayed in “pretty print” in the history area.

Press $\boxed{\blacklozenge}$ 9 X $\boxed{\rightarrow}$ Y $\boxed{\rightarrow}$ \boxed{ENTER}

Display



Expanding Expressions

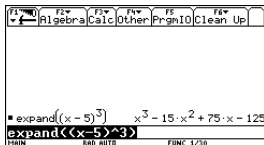
Steps and keystrokes

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(**.

Press **F2** 3 **(** X **)** 5 **)** **^** 3 **)** **ENTER**

(Optional) Enter other expressions on your own.

Display



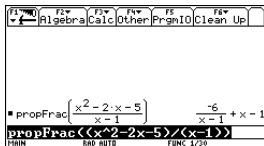
Reducing Expressions

Steps and keystrokes

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(**.

Press **F2** 7 **(** X **^** 2 **-** 2 X **-** 5 **)** **÷** **(** X **-** 1 **)** **ENTER**

Display



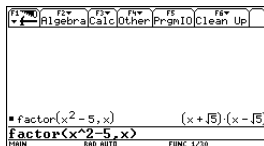
Factoring Polynomials

Steps and keystrokes

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()**.

Press **F2** **2** **X** **^** **2** **-** **5** **,** **X** **)** **ENTER**

Display



Solving Equations

Steps and keystrokes

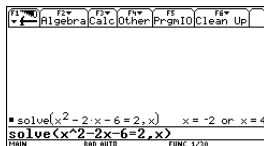
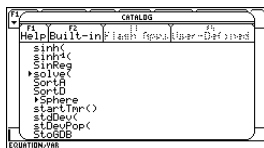
Solve the equation $x^2-2x-6=2$ with respect to x .

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.

Press **F2** **1** **X** **^** **2** **-** **2** **X** **-** **6** **=** **2** **,** **X** **)** **ENTER**

Display



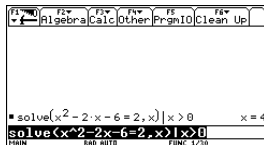
Solving Equations with a Domain Constraint

Steps and keystrokes

Solve the equation $x^2 - 2x - 6 = 2$ with respect to x where x is greater than zero. The “with” (!) operator provides domain constraint.

Press $\boxed{\text{F2}}$ $\boxed{1}$ \times $\boxed{\wedge}$ $\boxed{2}$ $\boxed{-}$ $\boxed{2}$ \times $\boxed{-}$ $\boxed{6}$ $\boxed{=}$ $\boxed{2}$ $\boxed{,}$ \times $\boxed{!}$ $\boxed{\text{2nd}}$
 $\boxed{[1]}$ \times $\boxed{\text{2nd}}$ $\boxed{[>]}$ $\boxed{0}$ $\boxed{\text{ENTER}}$

Display



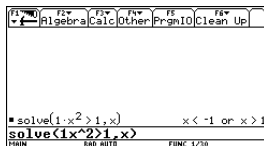
Solving Inequalities

Steps and keystrokes

Solve the inequality $(x^2 > 1, x)$ with respect to x .

Press $\boxed{\text{F2}}$ $\boxed{1}$ \times $\boxed{\wedge}$ $\boxed{2}$ $\boxed{\text{2nd}}$ $\boxed{[>]}$ $\boxed{1}$ $\boxed{,}$ \times $\boxed{!}$ $\boxed{\text{ENTER}}$

Display



Finding the Derivative of Functions

Steps and keystrokes

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.

Press $\boxed{2nd} \boxed{[d]} \boxed{(} \boxed{X} \boxed{-} \boxed{Y} \boxed{)} \boxed{\wedge} \boxed{3} \boxed{\div} \boxed{(} \boxed{X} \boxed{+} \boxed{Y} \boxed{)} \boxed{\wedge} \boxed{2} \boxed{,} \boxed{X} \boxed{)} \boxed{ENTER}$

Display

The calculator display shows the function $\frac{d}{dx} \left(\frac{(x-y)^3}{(x+y)^2} \right)$ in 'pretty print' mode. Below the function, the result is shown as $\frac{d}{dx} \left(\frac{(x-y)^3}{(x+y)^2} \right) = \frac{3(x-y)^2 \cdot (x+y)^2 - (x-y)^3 \cdot 2(x+y)}{(x+y)^4}$. The history area at the bottom shows the input $d((x-y)^3/(x+y)^2,x)$.

Finding Implicit Derivatives

Steps and keystrokes

Compute implicit derivatives for equations in two variables in which one variable is defined implicitly in terms of another.

This example illustrates using the calculus implicit derivative function.

Press $\boxed{F3} \boxed{D} \boxed{X} \boxed{\wedge} \boxed{2} \boxed{+} \boxed{Y} \boxed{\wedge} \boxed{2} \boxed{=} \boxed{100} \boxed{,} \boxed{X} \boxed{,} \boxed{Y} \boxed{)} \boxed{ENTER}$

Display

The calculator display shows the equation $\text{impDif}(x^2 + y^2 = 100, x, y)$ in 'pretty print' mode. Below the equation, the result is shown as $\text{impDif}(x^2 + y^2 = 100, x, y) = -\frac{x}{y}$. The history area at the bottom shows the input $\text{impDif}(x^2+y^2=100,x,y)$.

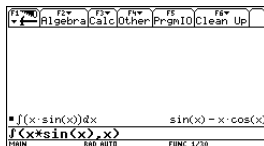
Finding the Integral of Functions

Steps and keystrokes

Find the integral of $x \cdot \sin(x)$ with respect to x .

This example illustrates using the calculus integration function.

Display



Press $\boxed{2nd} \boxed{[]} X \boxed{\times} \boxed{SIN} X \boxed{[]} \boxed{,} X \boxed{[]} \boxed{ENTER}$

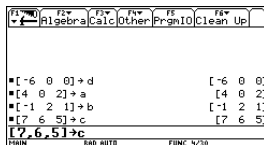
Solving Problems Involving Vectors

Steps and keystrokes

1. Input a row or column of vectors.

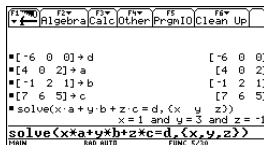
Press $\boxed{2nd} \boxed{[]} \boxed{(-)} \boxed{6} \boxed{,} \boxed{0} \boxed{,} \boxed{0} \boxed{2nd} \boxed{[]} \boxed{STO} \boxed{>} \boxed{d}$
 $\boxed{ENTER} \boxed{2nd} \boxed{[]} \boxed{4} \boxed{,} \boxed{0} \boxed{,} \boxed{2} \boxed{2nd} \boxed{[]} \boxed{STO} \boxed{>}$
a $\boxed{ENTER} \boxed{2nd} \boxed{[]} \boxed{(-)} \boxed{1} \boxed{,} \boxed{2} \boxed{,} \boxed{1} \boxed{2nd} \boxed{[]} \boxed{STO} \boxed{>}$
b $\boxed{ENTER} \boxed{2nd} \boxed{[]} \boxed{7} \boxed{,} \boxed{6} \boxed{,} \boxed{5} \boxed{2nd} \boxed{[]} \boxed{STO} \boxed{>}$
c \boxed{ENTER}

Display



2. Solve $(x \cdot a + y \cdot b + z \cdot c = d \{x, y, z\})$

Press $\boxed{F2} \boxed{1} X \boxed{\times} \boxed{a} \boxed{+} \boxed{y} \boxed{\times} \boxed{b} \boxed{+} \boxed{z} \boxed{\times} \boxed{c} \boxed{=}$
 $\boxed{d} \boxed{,} \boxed{2nd} \boxed{[]} X \boxed{,} Y \boxed{,} Z \boxed{2nd} \boxed{[]} \boxed{)} \boxed{ENTER}$



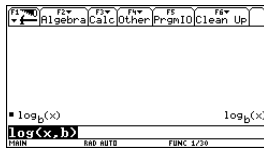
Log to Any Base

Steps and keystrokes

Find $\log(x, b)$. You can enter “log” on the entry line by typing **LOG** on the keyboard, or by pressing \blacklozenge 7.

Press \blacklozenge 7 X , b \square **ENTER**

Display



Converting Angle Measures

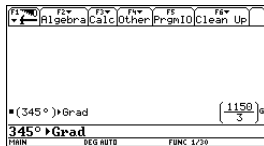
Steps and keystrokes

1. Display the **MODE** dialog box. For **Angle** mode select **DEGREE**. Convert 345 degrees to Gradian angle measure.

You can enter “**►Grad**” on the entry line by selecting “**►Grad**” from the Catalog menu, or from the Math menu by pressing \square 2 **nd** **[MATH]** and selecting **2:angle, A:►Grad**.

Press **MODE** \downarrow \downarrow \downarrow \downarrow 2 **ENTER** 345 \square 2 **nd** \square **[MATH]** 2 **A** **ENTER**

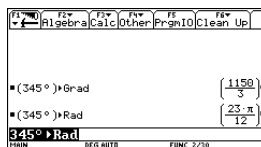
Display



2. Convert 345 degrees to Radian angle measure.

You can enter “►Rad” on the entry line by selecting “►Rad” from the Catalog menu, or from the Math menu by pressing $\boxed{2\text{nd}} \boxed{[\text{MATH}]}$ and selecting **2:angle**, **B:►Rad**.

Note: You can also use $^{\circ}$, r , or G to override the angle mode setting temporarily.



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 .

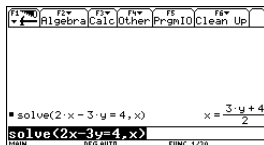
Steps and keystrokes

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 or select it from the **Catalog**.

Press **[♦]** **[CALC HOME]** **[CLEAR]** **[CLEAR]** **[F2]** 1 2
X **[=]** 3 **Y** **[=]** 4 **[,]** **X** **[)]** **[ENTER]**

Display



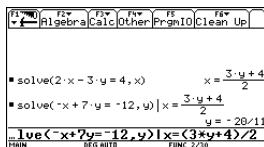
2. Begin to solve the equation $-x + 7y = -12$ for y , but do not press **[ENTER]** yet.

Press **[F2]** 1 **[(-)]** **X** **[+]** 7 **Y** **[=]** **[(-)]** 12 **[,]** **Y** **[)]**

3. Use the “with” operator 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.



Press **[2nd]** **[1]** **[>]** **[ENTER]** **[ENTER]**

4. Highlight the equation for x in the history area.

Press \leftarrow \rightarrow \rightarrow

TI-84 Plus calculator display showing the solution of a system of equations. The first equation is $\text{solve}(2 \cdot x - 3 \cdot y = 4, x)$ and the second is $\text{solve}(-x + 7 \cdot y = -12, y)$. The solution for x is $x = \frac{3 \cdot y + 4}{2}$. The second equation is highlighted.

5. Auto-paste the highlighted expression to the entry line. Then substitute the value of y that was calculated from the second equation.

Press $\boxed{\text{ENTER}}$ $\boxed{2\text{nd}}$ $\boxed{[1]}$ \rightarrow $\boxed{\text{ENTER}}$ $\boxed{\text{ENTER}}$

The solution is:

$$x = -8/11 \text{ and } y = -20/11$$

TI-84 Plus calculator display showing the same system of equations. The solution for x is $x = \frac{3 \cdot y + 4}{2}$. The second equation is $\text{solve}(-x + 7 \cdot y = -12, y)$ and the solution for y is $y = -20/11$. The expression $x = \frac{3 \cdot y + 4}{2}$ is highlighted.

This example is a demonstration of symbolic manipulation. A one-step function is available for solving systems of equations.

Constants and Measurement Units

Using the equation $f = m \cdot a$, calculate the force when $m = 5$ kilograms and $a = 20$ meters/second². What is the force when $a = 9.8$ meters/second². (This is the

acceleration due to gravity, which is a constant named `_g`). Convert the result from newtons to kilograms of force.

Steps and keystrokes

1. Display the **MODE** dialog box, Page 3. For **Unit System** mode, select **SI** for the metric system of measurements.

Results are displayed according to these default units.

Press `MODE` `F3` `↓` `1` `ENTER`

Display

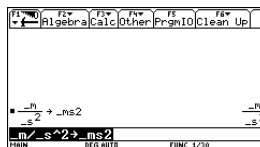
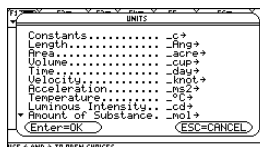


2. Create an acceleration unit for meters/second² named `_ms2`.

The **UNITS** dialog box lets you select units from an alphabetical list of categories.

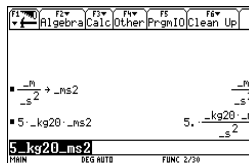
You can use `[2nd]` `[→]` and `[2nd]` `[←]` to scroll one page at a time through the categories.

If you use the **UNITS** dialog box to select a unit, the `_` is entered automatically. Now, instead of re-entering `_m/_s2` each time you need it, you can use `_ms2`. Also, you can now use the **UNITS** dialog box to select `_ms2` from the Acceleration category.



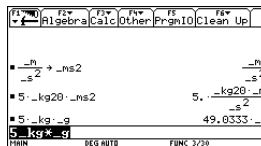
Press `[♦]` `[UNITS]` `[←]` `[↓]` `M` `[ENTER]` `[÷]` `[♦]` `[UNITS]`
`[←]` `[←]` `[←]` `[←]` `[↓]` `S` `[ENTER]` `[^]` `2` `[STO→]` `[2nd]` `[_]`
`MS2` `[ENTER]`

3. Calculate the force when
 $m = 5$ kilograms ($_kg$) and
 $a = 20$ meters/second² ($_ms2$).
 If you know the abbreviation for a unit,
 you can type it from the keyboard.



Press 5 **[2nd]** **[_]** KG **[x]** 20 **[2nd]** **[_]** MS2
[ENTER]

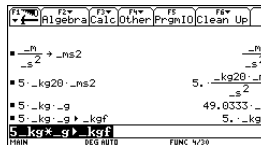
4. Using the same m , calculate the force for
 an acceleration due to gravity
 (the constant $_g$).
 For $_g$, you can use the pre-defined
 constant available from the **UNITS** dialog
 box or you can type $_g$.



Press 5 **[2nd]** **[_]** KG **[x]** **[♦]** **[UNITS]** **[↓]** G
[ENTER] **[ENTER]**

5. Convert to kilograms of force ($_kgf$).
[2nd] **[▶]** displays the **▶** conversion operator.

Press **[♦]** **[2nd]** **[▶]** **[2nd]** **[_]** KGF **[ENTER]**



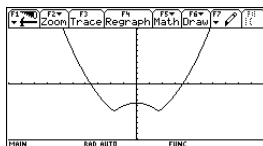
Steps and keystrokes

Display

3. Display the graph of the function.

Select **6:ZoomStd** by pressing **6** or by moving the cursor to **6:ZoomStd** and pressing **[ENTER]**.

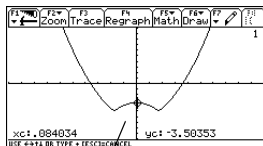
Press **[F2]** **6**



4. Turn on **Trace**.

The tracing cursor, and the x and y coordinates are displayed.

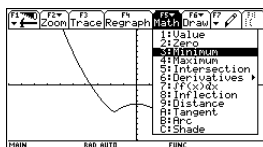
Press **[F3]**



tracing cursor

5. Open the **MATH** menu and select **3:Minimum**.

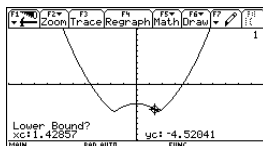
Press **[F5]** **⏪** **⏪** **[ENTER]**



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.

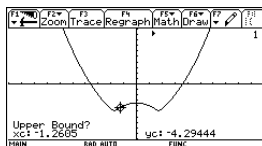
Press **⏩** ... **⏩** **[ENTER]**



7. Set the upper bound.

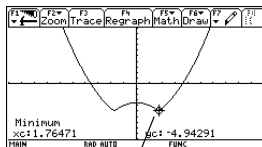
Press \rightarrow (right cursor) to move the tracing cursor until the upper bound for x is just to the right of the minimum node.

Press \rightarrow ... \rightarrow



8. Find the minimum point on the graph between the lower and upper bounds.

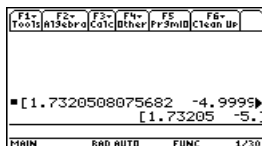
Press $\boxed{\text{ENTER}}$



minimum point
minimum coordinates

9. Transfer the result to the Home screen, and then display the Home screen.

Press \blacklozenge H \blacklozenge [CALC HOME]



Basic Function Graphing II

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 and keystrokes

Display

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

Press **MODE** **1** **ENTER**



2. Display the Home screen. Then store the radius, 5, in variable r.

5 **→r**

Press **◀** **[CALC HOME]** **5** **[STO▶]** **R** **ENTER**

3. Display and clear the **Y= Editor**. Then define $y_1(x) = \sqrt{r^2 - x^2}$, the top half of a circle.

In function graphing, you must define separate functions for the top and bottom halves of a circle.

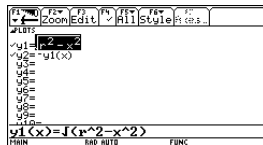
Press **◀** **[Y=]** **F1** **8** **ENTER** **ENTER** **2nd** **[√]** **R**
^ **2** **-** **X** **^** **2** **]** **ENTER**

4. Define $y_2(x) = -\sqrt{r^2 - x^2}$, the function for the bottom half of the circle.

The bottom half is the negative of the top half, so you can define $y_2(x) = -y_1(x)$.

Use the full function name $y_1(x)$, not simply y_1 .

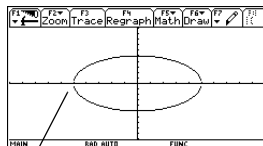
Press **[ENTER]** **(-)** **Y 1** **()** **X** **()** **[ENTER]**



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.

Press **[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.

Press **[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.

Basic Function Graphing III

Use the “Detect Discontinuities” graph format to eliminate faux asymptotes and connections in a jump discontinuity.

Steps and keystrokes

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

Press **[MODE]** **[▶]** **[◂]** **[◃]** **[◄]** **[▶]** **[1]** **[ENTER]**

Display



2. Open the **Y=** Editor and enter $y_1(x)=1/(x-1)$.

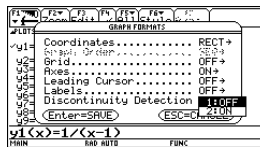
Press **[◀]** **[Y=]** **[1]** **[÷]** **[(]** **[X]** **[-]** **[1]** **[)]** **[ENTER]**



3. Display the **Graph Formats** dialog box and set “Detect Discontinuities” to **OFF**

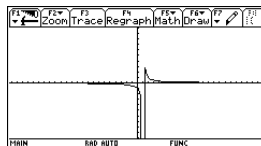
Note: The second item on the Graph Format dialog is not greyed out, which means it can be set to sequential “Seq” or simultaneous “Simul”.

Press **[◀]** **[F]** **[◂]** **[◃]** **[◄]** **[▶]** **[1]** **[ENTER]**



4. Execute the **Graph** command, which automatically displays the Graph screen. Observe the “faux” asymptotes contained in the graph.

Press \blacklozenge [GRAPH]



5. Display the Graph Formats dialog box and set “Detect Discontinuities” to ON.

Note: The second item on the Graph Format dialog is greyed out, which means the graph order is set to sequential “Seq”.

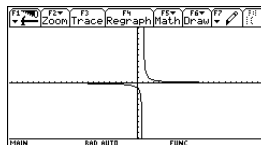
Press \blacklozenge F \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow \leftarrow 2 [ENTER]



6. Execute the **Graph** command, which automatically displays the Graph screen. No “faux” asymptotes are present on the graph.

Note: Graphing speed may slow considerably when “Detect Discontinuities” is set to ON.

Press \blacklozenge [GRAPH]



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/sec². Ignoring air resistance and other drag forces, what is the maximum height of the ball and when does it hit the ground?

Steps and keystrokes

Display

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

Press **[MODE]** **[2]** **[ENTER]**



2. Display and clear the **Y= Editor**. Then define the horizontal component $xt1(t) = v_0 t \cos \theta$.

$$xt1(t) = 15t * \cos(60^\circ)$$

Enter values for v_0 and θ .

Press **[♦]** **[Y=]** **[F1]** **[8]** **[ENTER]** **[ENTER]** **15T** **[X]**
[COS] **60** **[2nd]** **[°]** **[)]** **[ENTER]**

Type **T** **[X]** **[COS]**, not **T** **[COS]**.

Enter a $^\circ$ symbol by typing either **[2nd]** **[°]** or **[2nd]** **[MATH]** **2** **1**. This ensures a number is interpreted as degrees, regardless of the angle mode.

Polar Graphing

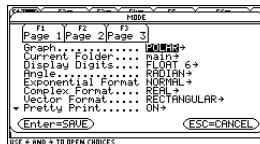
The graph of the polar equation $r_1(\theta) = A \sin B\theta$ forms the shape of a rose. Graph the rose for $A=8$ and $B=2.5$. Then explore the appearance of the rose for other values of A and B .

Steps and keystrokes

1. Display the **MODE** dialog box. For **Graph** mode, select **POLAR**. For **Angle** mode, select **RADIAN**.

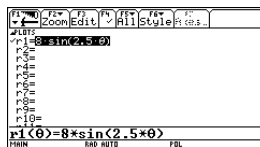
Press **MODE** **3** **3** **3** **1** **ENTER**

Display



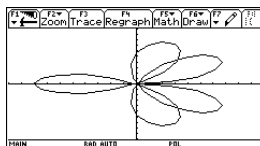
2. Display and clear the **Y= Editor**. Then define the polar equation $r_1(\theta) = A \sin B\theta$. Enter 8 and 2.5 for A and B , respectively.

Press **2ND** **[Y=]** **F1** **8** **ENTER** **ENTER** **8** **SIN** **2.5** **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 $\theta_{\max} = 2\pi$. 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.

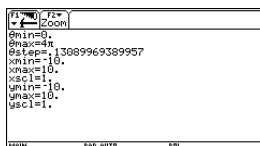


Press **[F2]** 6

4. Display the **Window Editor**, and change θ_{\max} to 4π .

4π will be evaluated to a number when you leave the **Window Editor**.

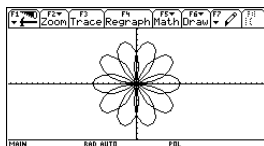
Press **[♦]** **[WINDOW]** **[◀]** 4 **[2nd]** **[π]**



5. Select **ZoomSqr**, which regraphs the equation.

ZoomSqr increases the range along the x axis so that the graph is shown in correct proportion.

Press $\boxed{F2}$ 5



You can change values for A and B as necessary and regraph the equation.

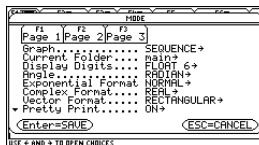
Sequence Graphing

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	$.8 \times 4000$ $+ 1000$	$.8 \times (.8 \times 4000 +$ $1000) + 1000$	$.8 \times (.8 \times (.8 \times$ $4000 + 1000) +$ $1000) + 1000$...

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

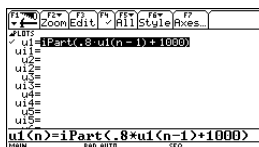
Press **[MODE]** **[4]** **[ENTER]**



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]**, simply type it, or select it from the **CATALOG**.



Press **[↓]** **[Y=]** **[F1]** **[8]** **[ENTER]** **[ENTER]** **[2nd]** **[MATH]**
[1] **[.]** **[8]** **[U1]** **[(]** **[N]** **[=]** **[1]** **)** **[+]** **[1000]** **)** **[ENTER]**

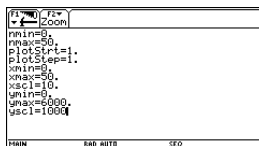
3. Define **u1** as the initial value that will be used as the first term.

Press **[ENTER]** **[4000]** **[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.

Press **[↓]** **[WINDOW]** **[0]** **[↓]** **[50]** **[↓]** **[1]** **[↓]** **[1]** **[↓]**

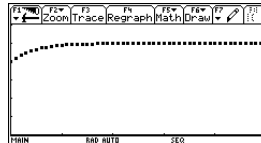


5. Set the x and y Window variables to appropriate values for this example.

Press $0 \downarrow 50 \downarrow 10 \downarrow 0 \downarrow 6000 \downarrow 1000$

6. Display the Graph screen.

Press \blacklozenge [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?

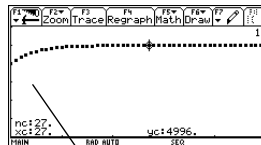
Trace begins at $nc=0$.

nc is the number of years.

$xc = nc$ since n is plotted on the x axis.

$yc = u_1(n)$, the number of trees at year n .

Press $F3 \downarrow$ and \updownarrow as necessary



By default, sequences use the Square display style.

3D Graphing

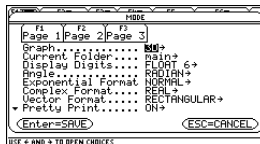
Graph the 3D equation $z(x,y) = (x^3y - y^3x) / 390$. Animate the graph by using the cursor to interactively change the eye Window variable values that control your viewing angle. Then view the graph in different graph format styles.

Steps and keystrokes

Display

1. Display the **MODE** dialog box. For **Graph** mode, select **3D**.

Press **MODE** **5** **ENTER**

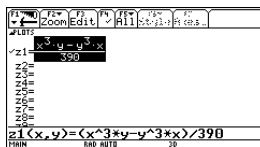


2. Display and clear the **Y= Editor**. Then define the 3D equation

$$z1(x,y) = (x^3y - y^3x) / 390.$$

Notice that implied multiplication is used in the keystrokes.

Press **Y=** **F1** **8** **ENTER** **ENTER** **X** **^** **3**
Y **-** **Y** **^** **3** **X** **)** **÷** **390** **ENTER**



3. Change the graph format to display and label the axes. Also set **Style = WIRE FRAME**.

You can animate any graph format style, but **WIRE FRAME** is fastest.

Press **F** **2** **2** **1** **ENTER**



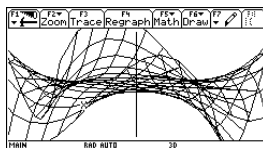
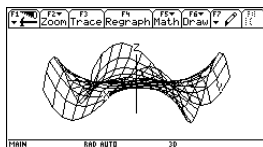
4. Select the **ZoomStd** viewing cube, which automatically graphs the equation.

As the equation is evaluated (before it is graphed), “evaluation percentages” are shown in the upper-left part of the screen.

Press **[F2]** 6

Note: If you have already used 3D graphing, the graph may be shown in expanded view. When you animate the graph, the screen returns to normal view automatically. (Except for animation, you can do the same things in normal and expanded view.)

Press **[X]** (press **[X]** to switch between expanded and normal view)

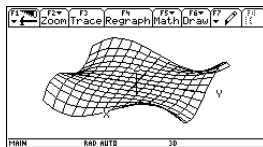


5. Animate the graph by decreasing the eye ϕ Window variable value.

[Left Arrow] or **[Right Arrow]** may affect eye θ and eye ψ , but to a lesser extent than eye ϕ .

To animate the graph continuously, press and hold the cursor for about 1 second and then release it. To stop, press **[ENTER]**.

Press **[Left Arrow]** eight times



Steps and keystrokes**Display**

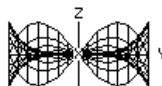
6. Return the graph to its initial orientation.
Then move the viewing angle along the
“viewing orbit” around the graph.



Press 0 (zero, not the letter O) ⓪ ⓪ ⓪

7. View the graph along the x axis, the
y axis, and then the z axis.

Press X

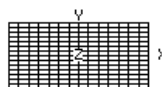


This graph has the same shape along the
y axis and x axis.

Press Y



Press Z



-
8. Return to the initial orientation.

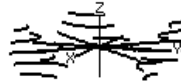
Press 0 (zero)

9. Display the graph in different graph format styles.

Press F (press F to switch from each style to the next)



HIDDEN SURFACE



CONTOUR LEVELS
(may require extra
time to calculate
contours)



WIRE AND
CONTOUR



WIRE FRAME

Note: You can also display the graph as an implicit plot by using the **GRAPH FORMATS** dialog box (\blacktriangledown **F**). If you press **F** to switch between styles, the implicit plot is not displayed.

Differential Equation Graphing

Graph the solution to the logistic 1st-order differential equation $y' = .001y*(100-y)$. Start by drawing only the slope field. Then enter initial conditions in the **Y= Editor** and interactively from the Graph screen.

Steps and keystrokes

1. Display the **MODE** dialog box. For Graph mode, select **DIFF EQUATIONS**.

Press **MODE** \blacktriangleright 6 **ENTER**

Display



2. Display and clear the **Y= Editor**. Then define the 1st-order differential equation:

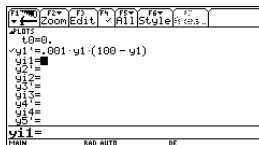
$$y1'(t)=.001y1*(100-y1)$$

Press $\boxed{\times}$ to enter the * shown above. Do not use implied multiplication between the variable and parentheses. If you do, it is treated as a function call.

Leave the initial condition **y11** blank.

Note: With $y1'$ selected, the device will graph the **y1** solution curve, not the derivative $y1'$.

Press $\boxed{\blacklozenge}$ $\boxed{[Y=]}$ $\boxed{F1}$ $\boxed{8}$ \boxed{ENTER} \boxed{ENTER} $\boxed{.001}$ $\boxed{Y1}$
 $\boxed{\times}$ $\boxed{(}$ $\boxed{100}$ $\boxed{)}$ $\boxed{Y1}$ $\boxed{)}$ \boxed{ENTER}



3. Display the **GRAPH FORMATS** dialog box. Then set **Axes = ON**, **Labels = ON**, **Solution Method = RK**, and **Fields = SLPFLD**.

Note: To graph one differential equation, **Fields** must be set to **SLPFLD** or **FLDOFF**. If **Fields=DIRFLD**, an error occurs when you graph.

Press $\boxed{\blacklozenge}$ \boxed{F} $\boxed{\downarrow}$ $\boxed{\downarrow}$ $\boxed{2}$ $\boxed{\downarrow}$ $\boxed{\downarrow}$ $\boxed{2}$ $\boxed{\downarrow}$ $\boxed{1}$ $\boxed{\downarrow}$
 $\boxed{1}$ \boxed{ENTER}



Steps and keystrokes**Display**

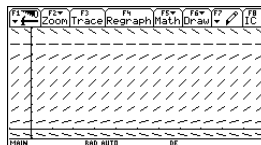
4. Display the **Window Editor**, and set the Window variables as shown to the right.

Press \blacklozenge [WINDOW] 0 \odot 10 \odot .1 \odot 0 \odot
 \ominus 10 \odot 110 \odot 10 \odot \ominus 10 \odot 120 \odot 10
 \odot 0 \odot .001 \odot 20

```
t0=0.
tmax=10.
tstep=.1
tplot=0.
xmin=-10.
xmax=110.
xsc1=10.
ymin=-10.
ymax=120.
ysc1=10.
ncurves=0.
difftol=.001
fldres=20.
```

5. Display the Graph screen.

Because you did not specify an initial condition, only the slope field is drawn (as specified by **Fields=SLPFLD** in the **GRAPH FORMATS** dialog box).

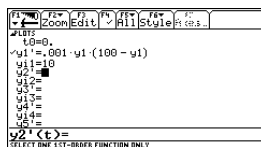


Press \blacklozenge [GRAPH]

6. Return to the **Y= Editor** and enter an initial condition:

yi1=10

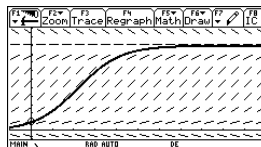
Press \blacklozenge [Y=] [ENTER] 10 [ENTER]



7. Return to the Graph screen.

Initial conditions entered in the **Y= Editor** always occur at t_0 . The graph begins at the initial condition and plots to the right. Then it plots to the left.

Press \blacklozenge [GRAPH]

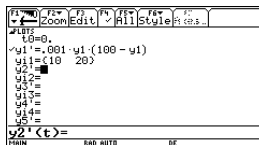


The initial condition is marked with a circle.

8. Return to the Y= Editor and change y_1 to enter two initial conditions as a list:

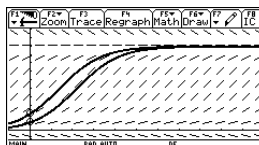
$y_1 = \{10, 20\}$

Press \blacklozenge [Y=] \blacktriangleleft [ENTER] [2nd] [{] 10 [.] 20
[2nd] [}] [ENTER]



9. Return to the Graph screen.

Press \blacklozenge [GRAPH]



10. To select an initial condition interactively, press:

F8

When prompted, enter $t=40$ and $y1=45$.

When selecting an initial condition interactively, you can specify a value for t other than the t_0 value entered in the

Y= Editor or **Window Editor**.

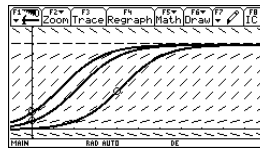
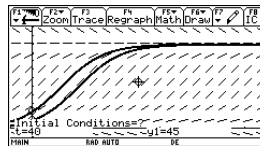
Instead of entering t and $y1$ after pressing

F8

you can move the cursor to a point on the screen and then press **ENTER**.

You can use **F3** to trace curves for initial conditions specified in the **Y= Editor**.

However, you cannot trace the curve for an initial condition selected interactively.



Press **F8** 40 **ENTER** 45 **ENTER**

Additional Graphing Topics

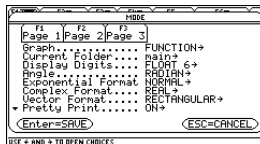
From the Home screen, graph the piecewise defined function: $y = -x$ when $x < 0$ and $y = 5 \cos(x)$ when $x \geq 0$. Draw a horizontal line across the top of the cosine curve. Then save a picture of the displayed graph.

Steps and keystrokes

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

Press **[MODE]** **[1]** **[2]** **[1]** **[ENTER]**

Display



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.

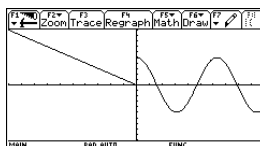
Graph when($x<0,-x,$
 $5*\cos(x)$)

Press **[♦]** **[CALC HOME]** **[F4]** **2** **WHEN** **[X]**
[2nd] **[<]** **0** **[,]** **(-)** **X** **[,]** **5** **[X]** **[COS]** **X** **[)]** **[)]**

3. Execute the **Graph** command, which automatically displays the Graph screen.

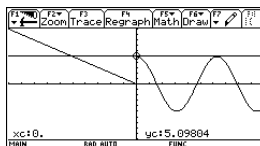
The graph uses the current Window variables, which are assumed to be their standard values (**[F2]** **6**) for this example.

Press **[ENTER]**



4. Draw a horizontal line across the top of the cosine curve.

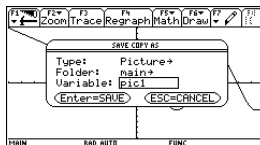
The calculator remains in “horizontal” mode until you select a different operation or press **[ESC]**.



Press **[F7]** 5 \ominus (until the line is positioned)
[ENTER]

5. Save a picture of the graph. Use **PIC1** as the variable name for the picture.

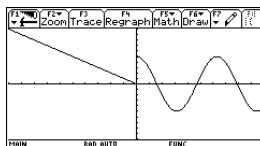
Be sure to set **Type = Picture**. By default, it is set to **GDB**.



Press **[F1]** 2 \Downarrow 2 \Downarrow **PIC1** **[ENTER]** **[ENTER]**

6. Clear the drawn horizontal line.
You can also press **[F4]** to regraph.

Press **[F6]** 1



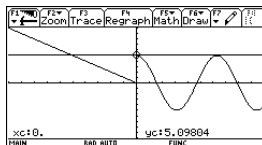
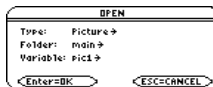
Steps and keystrokes

Display

- 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**.

Press $\boxed{F1}$ 1 \downarrow 2 (if not already shown, also set Variable = pic1) \boxed{ENTER}



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 and keystrokes

Display

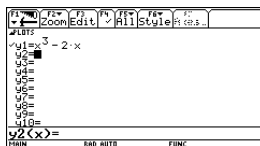
- Display the **MODE** dialog box. For the **Graph mode**, select **FUNCTION**.

Press \boxed{MODE} \downarrow 1 \boxed{ENTER}



2. Display and clear the **Y= Editor**. Then define $y_1(x) = x^3 - 2x$.

Press \blacklozenge [Y=] [F1] 8 [ENTER] [ENTER] X \wedge 3 \ominus
2 X [ENTER]



3. Set the table parameters to:

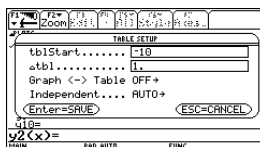
tblStart = -10

Δ tbl = 1

Graph <-> Table = OFF

Independent = AUTO

Press \blacklozenge [TBLSET] \leftarrow 10 \downarrow 1 \downarrow 1 \downarrow 1
[ENTER]



4. Display the Table screen.

Press \blacklozenge [TABLE]

x	y1	y2	y3	y4	y5	y6
-10	-980					
-9	-711					
-8	-496					
-7	-329					
-6	-204					
-5	-119					
-4	-56					
-3	-21					

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

To scroll one page at a time, use $\boxed{2nd}$ \downarrow
and $\boxed{2nd}$ \uparrow .

Press \downarrow and \uparrow as necessary

x	y1	y2	y3	y4	y5	y6
-4	-56					
-3	-21					
-2	-4					
-1	1					
0	0					
1	1					
2	4					

Steps and keystrokes

Display

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

tblStart = -2

Δ tbl = .1

Press **[F2]** **[(-)]** **2** **[∇]** **.1** **[ENTER]** **[ENTER]**

TblStart	Δtbl	Y1	Y2	X	Y
-2	.1			-2	-4
				-1.9	-3.9505
				-1.8	-3.9200
				-1.7	-3.8505
				-1.6	-3.7500
				-1.5	-3.6005
				-1.4	-3.4000
				-1.3	-3.1505

Split Screens

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

Steps and keystrokes

Display

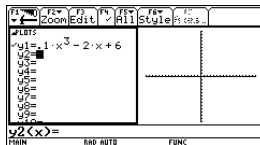
1. Display the **MODE** dialog box.
For **Graph**, select **FUNCTION**.
For **Split Screen**, select **LEFT-RIGHT**.
For **Split 1 App**, select **Y= Editor**.
For **Split 2 App**, select **Graph**.

Press **[MODE]** **[∇]** **1** **[F2]** **[∇]** **3** **[∇]** **2** **[∇]** **4**
[ENTER]



2. Clear the **Y= Editor** and turn off any stat data plots. Define $y_1(x) = .1x^3 - 2x + 6$.

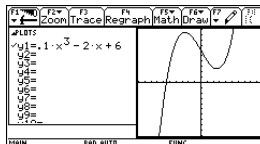
A thick border around the **Y= Editor** indicates it is active. When active, its entry line goes all the way across the display.



Press **[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.

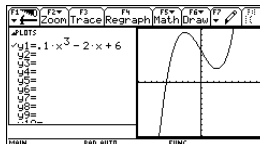


Press **[F2]** **6**

4. Switch to the **Y= Editor** and edit $y_1(x)$ to change $.1x^3$ to $.5x^3$.

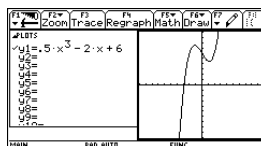
[2nd] **[+]** is the second function of **[APPS]**. The thick border is around the **Y= Editor**.

Press **[2nd]** **[+]** **[<]** **[ENTER]** **[↓]** **[↓]** **[↓]** **[←]** **5**
[ENTER]



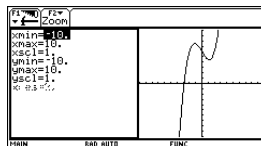
5. Switch to the Graph screen, which regraphs the edited function.
The thick border is around the Graph screen.

Press $\boxed{2nd}$ $\boxed{[⇄]}$



6. Switch to the **Y= Editor** and open the **Window Editor** in its place.

Press $\boxed{2nd}$ $\boxed{[⇄]}$ \blacklozenge $\boxed{[WINDOW]}$



7. Open the Home screen and then exit to a full-sized Home screen.

Press: $\boxed{2nd}$ $\boxed{[QUIT]}$

Data/Matrix Editor

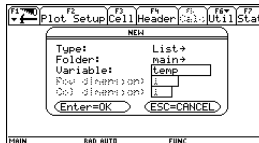
Use the **Data/Matrix Editor** to create a one-column list variable. Then add a second column of information. Notice that the list variable (which can have only one column) is automatically converted into a data variable (which can have multiple columns).

Steps and keystrokes

1. Use **[APPS]** to display the **Data/Matrix Editor**. Create a new list variable named **TEMP**.

Press 3 **[▶]** 3 **[◀]** **[◀]** **TEMP** **[ENTER]** **[ENTER]**

Display

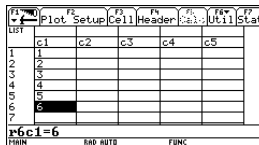


2. Enter a column of numbers. Then move the cursor up one cell (just to see that a highlighted cell's value is shown on the entry line).

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.

Press 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.

	F1	F2	F3	F4	F5	F6	F7
←	Plot	Setup	Cell	Header	Calc	Util	Stat
DATA	c1	c2	c3	c4	c5		
1	1	2					
2	2	4					
3	3	6					
4	4	8					
5	5	10					
6	6	12					

DATA

c2=2*c1

MAIN END MODE FUNC

DATA means the cell is in a defined column.

Press \downarrow $\boxed{F4}$ $\boxed{2}$ $\boxed{\times}$ $\boxed{C1}$ $\boxed{\text{ENTER}}$

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 $\boxed{F4}$ to define it. Simply begin typing the expression.

	F1	F2	F3	F4	F5	F6	F7
←	Plot	Setup	Cell	Header	Calc	Util	Stat
DATA	c1	c2	c3	c4	c5		
1	1	2					
2	2	4					
3	3	6					
4	4	8					
5	5	10					
6	6	12					

DATA

c2=2*c1

MAIN END MODE FUNC

Press $\boxed{2nd}$ $\boxed{\rightarrow}$ $\boxed{\leftarrow}$

5. Clear the contents of the variable.

Simply clearing the data does not convert the data variable back into a list variable.

Press $\boxed{F1}$ $\boxed{8}$ $\boxed{\text{ENTER}}$

	F1	F2	F3	F4	F5	F6	F7
←	Plot	Setup	Cell	Header	Calc	Util	Stat
DATA	c1	c2	c3	c4	c5		
1							
2							
3							
4							
5							
6							

DATA

c1

MAIN END MODE FUNC

Note: 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 wasting memory by creating a new variable each time.

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 and keystrokes

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

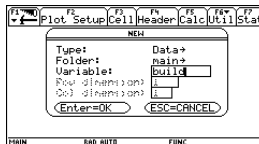
Press **MODE** **1** **ENTER**

Display



2. Use **APPS** to display the **Data/Matrix Editor**. Create a new data variable named **BUILD**.

Press **3** **2** **BUILD** **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

Press 150 **ENTER** 500 **ENTER** 800 **ENTER**
 250 **ENTER** 500 **ENTER** 750 **ENTER** 950
ENTER

Plot	Setup	Cell	Header	Calc	Util	Stat
c1	c2	c3	c4	c5		
500						
800						
250						
500						
750						
950						

r8c1 =

4. Move the cursor to row 1 in column 2 (r1c2). Then enter the corresponding number of buildings.

◀ **↶** moves the cursor to the top of the page. 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.

Press **↶** **2nd** **↶** 4 **ENTER** 31 **ENTER** 42
ENTER 9 **ENTER** 20 **ENTER** 55 **ENTER** 73
ENTER

Plot	Setup	Cell	Header	Calc	Util	Stat
c1	c2	c3	c4	c5		
500	31					
800	42					
250	9					
500	20					
750	55					
950	73					

r8c2 =

7. Perform the calculation to display the MedMed regression equation.

As specified on the **Calculate** dialog box, this equation is stored in **y1(x)**.

Press **ENTER**



8. Close the **STAT VARS** screen. The **Data/Matrix Editor** displays.

Press **ENTER**

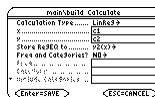
9. Display the **Calculate** dialog box. Set:
Calculation Type = LinReg

x = C1

y = C2

Store RegEQ to = y2(x)

Press **F5** **↓** **5** **↶** **↷** **↶** **↷** **ENTER**



10. Perform the calculation to display the LinReg regression equation.

This equation is stored in **y2(x)**.

Press **ENTER**



11. Close the **STAT VARS** screen. The **Data/Matrix Editor** displays.

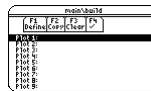
Press **ENTER**

12. Display the Plot Setup screen.

Plot 1 is highlighted by default.

[F3] lets you clear highlighted Plot settings.

Press **[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.



Press **[F1]** **▶** 1 **◀** **▶** 1 **◀** C1 **▶** C2

14. Save the plot definition and return to the Plot Setup screen.

Notice the shorthand notation for **Plot 1's** definition.

Press **[ENTER]** twice



15. Display the **Y= Editor**. For $y_1(x)$, 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 y_1 .

PLOTS 1 at the top of the screen means that **Plot 1** is selected.

Notice that $y_1(x)$ and $y_2(x)$ were selected when the regression equations were stored.

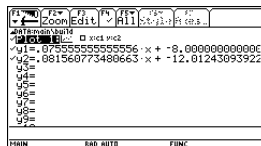
Press \blacklozenge [Y=] [F6] 2



16. Scroll up to highlight **Plot 1**.

The displayed shorthand definition is the same as on the Plot Setup screen.

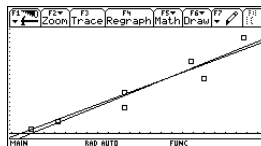
Press \leftarrow



17. Use **ZoomData** to graph **Plot 1** and the regression equations $y_1(x)$ and $y_2(x)$.

ZoomData examines the data for all selected stat plots and adjusts the viewing window to include all points.

Press [F2] 9



18. Return to the current session of the **Data/Matrix Editor**.

Press **[APPS]** **D** **[ENTER]** **[ENTER]**

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.

Press **[↓]** **[↓]** **[←]** **[←]** **MED** **[ENTER]** **[F4]** **Y1** **[]** **C1**
[] **[ENTER]**

20. Enter a title for column 4. Define column 4's header as the residuals (difference between observed and predicted values) for MedMed.

	F1	F2	F3	F4	F5	F6	F7
Plot	Setup	Cell	Header	Calc	Util	Stat	
			med	resid			
	c1	c2	c3	c4	c5		
1	150	4	3.3333	6.6667			
2	250	9	10.5556	1.8889			
3	500	51	23.7778	1.2222			
4	500	20	23.7778	-9.7778			
5	750	85	48.5556	6.3333			
6	800	42	52.4444	-10.4444			
7	950	75	63.7778	9.2222			

e4=c2-c3

FORM RAO RATE FUNC

Press **[↓]** **[←]** **RESID** **[ENTER]** **[F4]** **C2** **[]** **C3**
[ENTER]

21. Enter a title for column 5. Define column 5's header as the values predicted by the LinReg line.

Press \rightarrow \leftarrow LIN $\boxed{\text{ENTER}}$ $\boxed{\text{F4}}$ Y2 $\boxed{\text{[]}}$ C1 $\boxed{\text{[]}}$
 $\boxed{\text{ENTER}}$

22. Enter a title for column 6. Define column 6's header as the residuals for LinReg.

MEMO	PL	Y1	Y2	FN	DE	FR	TF
Plot Setup	Cell	Header	Calc	Util	Stat		
DATA	C2	C3	C4	C5	C6		
1							
4	5.3333	66567	22169	-8867			
9	10.889	1.8898	3778	-75.4			
31	29.778	1.2222	25.788	-811.8			
20	29.778	5.1222	25.788	-974.4			
55	48.557	6.3333	49.188	-2704.			
47	52.444	-10.444	59.236	-2285.			
75	63.778	2.2222	65.477	-4793.			
c6=c2*-c5							
END EDIT FUNC							

Press \rightarrow \leftarrow RESID $\boxed{\text{ENTER}}$ $\boxed{\text{F4}}$ C2 $\boxed{\text{[]}}$ C5
 $\boxed{\text{ENTER}}$

23. Display the Plot Setup screen and deselect Plot 1.

Press $\boxed{\text{F2}}$ $\boxed{\text{F4}}$

24. Highlight Plot 2 and define it as:

Plot Type = Scatter

Mark = Box

x = C1

y = C4 (MedMed residuals)

MedMed Plot 2	
Plot Type.....	Scatter+
Mark.....	Box
X.....	C1
Y.....	C4
Plot: Scatter Region	L
Use Freq and Categories?	NO+
PF6d.....	
Calculator.....	
Include Distributions	C
Enter=SRVE	
ESC=CANCEL	
TYPE < ENTER;FB; and DEGREE/CANCEL	

Press \leftarrow $\boxed{\text{F1}}$ \leftarrow \leftarrow C1 \leftarrow C4 $\boxed{\text{ENTER}}$ $\boxed{\text{ENTER}}$

25. Highlight Plot 3 and define it as:

Plot Type = Scatter

Mark = Plus

x = C1

y = C6 (LinReg residuals)



Press \odot $\boxed{F1}$ \odot \odot 3 \odot C1 \odot C6 $\boxed{\text{ENTER}}$

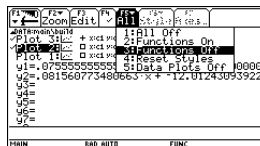
$\boxed{\text{ENTER}}$

26. Display the **Y= Editor** and turn all the **y(x)** functions off.

From $\boxed{F5}$, select **3:Functions Off**, not **1:All Off**.

Plots 2 and 3 are still selected.

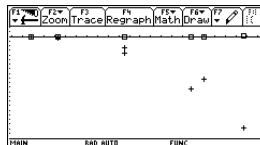
Press \blacklozenge $\boxed{Y=}$ $\boxed{F5}$ 3



27. Use **ZoomData** to graph the residuals.

- \square marks the MedMed residuals;
- $+$ marks the LinReg residuals.

Press $\boxed{F2}$ 9



28. Display the Home screen.

Press \blacklozenge $\boxed{\text{CALC HOME}}$

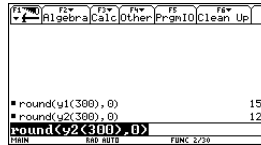
Steps and keystrokes**Display**

29. Use the MedMed ($y1(x)$) and LinReg ($y2(x)$) regression equations to calculate values for $x = 300$ (300,000 population).

The **round** function ($\boxed{2nd}$ $\boxed{[MATH]}$ $\boxed{1}$ $\boxed{3}$) ensures that results show an integer number of buildings.

After calculating the first result, edit the entry line to change $y1$ to $y2$.

Press $\boxed{2nd}$ $\boxed{[MATH]}$ $\boxed{1}$ $\boxed{3}$ $\boxed{Y1}$ $\boxed{[]}$ $\boxed{300}$ $\boxed{[]}$ $\boxed{,}$ $\boxed{0}$ $\boxed{[]}$
 \boxed{ENTER} $\boxed{\downarrow}$ $\boxed{\downarrow}$ (eight times) $\boxed{\leftarrow}$ $\boxed{2}$ \boxed{ENTER}



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 and keystrokes**Display**

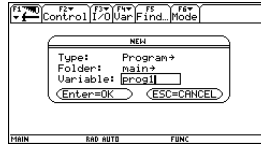
1. Use \boxed{APPS} to display the **Program Editor**.
Create a new program.

Press 3




2. Type **PROG1** (with no spaces) as the name of the new program variable.

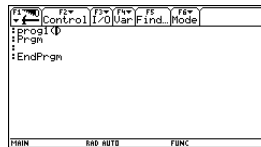
Press   PROG 1



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  twice.

Press  twice



4. Type the following program lines.

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.

```
:prog1()
:Prgh
:Request "Enter an integer
",n
:expr(n)→n
:0→temp
:For i,1,n,1
: temp+i→temp
:EndFor
:Disp temp
:
:EndPrgh
```

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.

Type the program lines as shown.

Press **ENTER** at the end of each line.

5. Go to the Home screen. Enter the program name, followed by a set of parentheses.



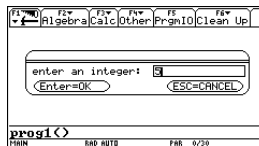
You must include () even when there are no arguments for the program.

The program displays a dialog box with the prompt specified in the program.

Press  [CALC HOME] PROG1   

6. Type 5 in the displayed dialog box.

Press 5

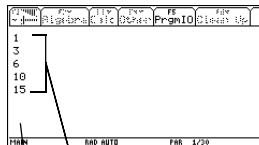


7. Continue with the program. The **Disp** command displays the result on the Program I/O screen.

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.

Press  twice



Output from other programs may still be on the screen.

Result of integer 5

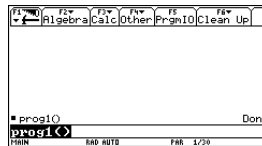
Steps and keystrokes

Display

8. Leave the Program I/O screen and return to the Home screen.

You can also press **[ESC]**, **[2nd] [QUIT]**, or **[F5] [CALC HOME]** to return to the Home screen.

Press **[F5]**



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 and keystrokes

Display

1. Start a new session of the **Text Editor**.

Press **3**

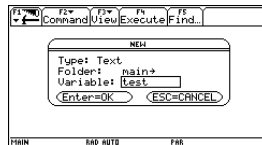


2. Create a text variable called **TEST**, which will automatically store any text you enter in the new session.


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.



Press **[F5] TEST [ENTER] [ENTER]**

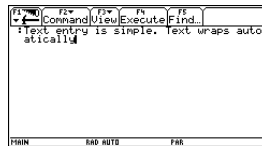


3. Type some sample text.

- To type a single uppercase letter, press  and then the letter.

Practice editing your text by using:

- The cursor pad to move the text cursor.
-  or  [DEL] 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**.

Press  [CALC HOME]

5. Return to the current session on the **Text Editor**. Notice that the displayed session is exactly the same as you left it.

Press  [

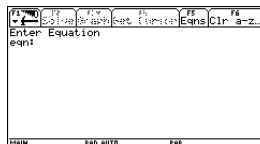
Numeric Solver

Consider the equation $a=(m2-m1)/(m2+m1)*g$, where the known values are $m2=10$ and $g=9.8$. If you assume that $a=1/3 g$, find the value of $m1$.

Steps and keystrokes

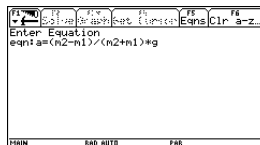
Display

1. Use **[APPS]** to display the **Numeric Solver**.



2. Enter the equation.

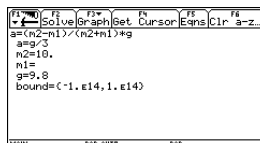
When you press **[ENTER]** or **⏏**, the screen lists the variables used in the equation.



Press **A** **[M2]** **[M1]** **[÷]** **[M2]** **[+]**
[M1] **[×]** **G** **[ENTER]**

3. Enter values for each variable, except the unknown variable $m1$.

Define $m2$ and g first. Then define a . (You must define g before you can define a in terms of g .) Accept the default for bound. If a variable has been defined previously, its value is shown as a default.



Press **⏏** **10** **⏏** **9.8** **⏏** **⏏** **G** **[÷]** **3**

4. Move the cursor to the unknown variable $m1$.

Optionally, you can enter an initial guess for $m1$. Even if you enter a value for all variables, the Numeric Solver solves for the variable marked by the cursor.

Press $\odot \odot$

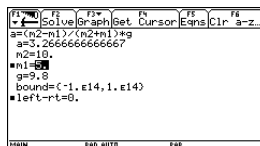


$g/3$ is evaluated when you move the cursor off the line.

5. Solve for the unknown variable.

To check the solution's accuracy, the left and right sides of the equation are evaluated separately. The difference is shown as left-rt. If the solution is precise, left-rt=0.

Press $\boxed{F2}$



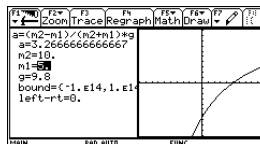
■ marks the calculated values.

6. Graph the solution using a **ZoomStd** viewing window.

The graph is displayed in a split screen. You can explore the graph by tracing, zooming, etc.

The variable marked by the cursor (unknown variable $m1$) is on the x axis, and left-rt is on the y axis.

Press $\boxed{F3}$ 3



7. Return to the **Numeric Solver** and exit the split screen.

You can press **ENTER** or **↵** to redisplay the list of variables.

Press **2nd** **[←]** **F3** **2**

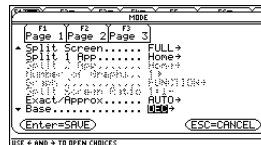
Number Bases

Calculate 10 binary (base 2) + F hexadecimal (base 16) + 10 decimal (base 10). Then, use the **▶** operator to convert an integer from one base to another. Finally, see how changing the Base mode affects the displayed results.

1. Display the **MODE** dialog box, Page 2. For **Base** mode, select **DEC** as the default number base.

Integer results are displayed according to the **Base** mode. Fractional and floating-point results are always displayed in decimal form.

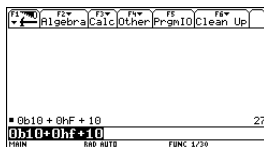
Press **MODE** **F2** (use **↵** to move to **Base** mode) **▶** **1** **ENTER**



2. Calculate $0b10+0hF+10$.

To enter a binary or hex number, you must use the 0b or 0h prefix (zero and the letter B or H). Otherwise, the entry is treated as a decimal number.

Note: The 0b or 0h prefix is a zero, not the letter O, followed by B or H.



Press 0 B 10 $+$ 0 H F $+$ 10 [ENTER]

3. Add 1 to the result and convert it to binary.

[2nd] [>] displays the \blacktriangleright conversion operator.

Press $+$ 1 [2nd] [>] BIN [ENTER]

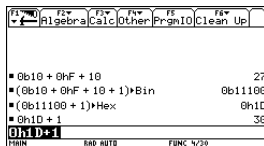
4. Add 1 to the result and convert it to hexadecimal.

Press $+$ 1 [2nd] [>] HEX [ENTER]

5. Add 1 to the result and leave it in the default decimal base.

Results use the 0b or 0h prefix to identify the base.

Press $+$ 1 [ENTER]



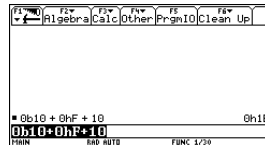
6. Change the **Base** mode to **HEX**.

When **Base** = **HEX** or **BIN**, the magnitude of a result is restricted to certain size limitations.

Press **MODE** **F2** (use \odot to move to **Base** mode) \downarrow 2 **ENTER**

7. Calculate $0b10+0hF+10$.

Press 0 B 10 **+** 0 HF **+** 10 **ENTER**

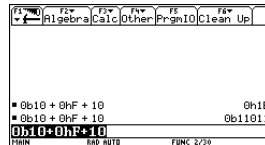


8. Change the **Base** mode to **BIN**.

Press **MODE** **F2** (use \odot to move to **Base** mode) \downarrow 3 **ENTER**

9. Re-enter $0b10+0hF+10$.

Press **ENTER**



Memory and Variable Management

Assign values to a variety of variable data types. Use the **VAR-LINK** screen to view a list of the defined variables. Then move a variable to the user data archive memory and explore the ways in which you can and cannot access an archived variable. (Archived

variables are locked automatically.) Finally, unarchive the variable and delete the unused variables so that they will not take up memory.

Steps and keystrokes

Display

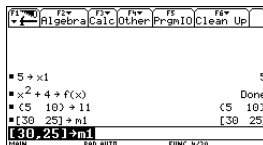
- From the Home screen, assign variables with the following variable types.

Expression: $5 \rightarrow X1$

Function: $x^2 + 4 \rightarrow f(x)$

List: $\{5, 10\} \rightarrow L1$

Matrix: $[30, 25] \rightarrow M1$



Press \blacklozenge [CALC HOME] [CLEAR] 5 [STO] X1
 [ENTER] X \wedge 2 + 4 [STO] F (X) [ENTER]
 [2nd] [{ 5 , 10 }] [STO] L1 [ENTER] [2nd]
 [[30 , 25]] [STO] M1 [ENTER]

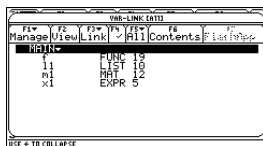
- Suppose you start to perform an operation using a function variable but can't remember its name.

5*

Press 5 \times

- Display the VAR-LINK screen.

This example assumes that the variables assigned above are the only ones defined.

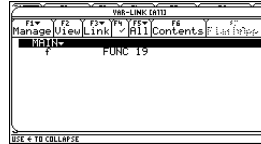
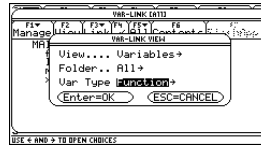


Press [2nd] [VAR-LINK]

4. Change the screen's view to show only function variables.

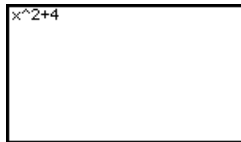
Although this may not seem particularly useful in an example with four variables, consider how useful it could be if there were many variables of all different types.

Press **[F2]** **⇩** **⇩** **⇩** **5** **[ENTER]**



5. Highlight the **f** function variable, and view its contents.

Notice that the function was assigned using **f(x)** but is listed as **f** on the screen.



Press **⇩** **[F6]**

6. Close the Contents window.

Press **[ESC]**

7. With the **f** variable still highlighted, close **VAR-LINK** to paste the contents of the variable to the entry line. Notice that "(" is pasted.

5*(f(

Press **[ENTER]**

Steps and keystrokes**Display**

8. Complete the operation.

Press 2

5*f(2)

Archiving a variable

Steps and keystrokes

1. Redisplay **VAR-LINK**, and highlight the variable you want to archive.

The previous change in view is no longer in effect. The screen lists all defined variables.

Press **[2nd] [VAR-LINK]** (use **⊖** to highlight **x1**)

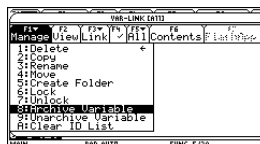
Display



2. Use the **[F1] Manage** toolbar menu to archive the variable.

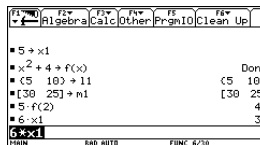
x indicates the variable is archived.

Press **[F1] 8**



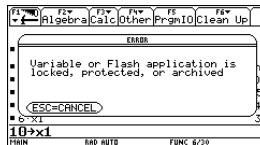
3. Return to the Home screen and use the archived variable in a calculation.

Press **[♦] [CALC HOME] 6 [x] X1 [ENTER]**



4. Attempt to store a different value to the archived variable.

Press 10 $\text{STO}\blacktriangleright$ $X1$ ENTER



5. Cancel the error message.

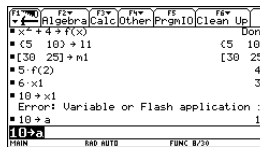
Press ESC

6. Use **VAR-LINK** to unarchive the variable.

Press 2nd $[\text{VAR-LINK}]$ (use \odot to highlight $x1$) F1 9

7. Return to the Home screen and store a different value to the unarchived variable.

Press \blacklozenge $[\text{CALC HOME}]$ ENTER



Deleting variables

Steps and keystrokes

1. Display **VAR-LINK**, and 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.

Note: Instead of using **[F5]** (if you don't want to delete all your variables), you can select individual variables. Highlight each variable to delete and press **[F4]**.

Press **[F5]** 1

Display



2. Use **[F1]** to delete.

Note: You can press **[←]** (instead of **[F1]** 1) to delete the marked variables.

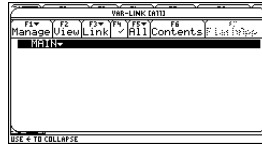
Press **[F1]** 1

3. Confirm the deletion.

Press **[ENTER]**



4. 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.

Press **ENTER**

5. Close **VAR-LINK** 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.

Press **ESC**

Operating the Calculator

Turning the Calculator On and Off

You can turn your graphing calculator 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 calculator turn itself off automatically.

Turning the Calculator On

Press **[ON]**.

- If you turned the unit off by pressing **[2nd] [OFF]**, the unit returns to either the Apps desktop or the Home screen.
- If you turned the unit off by pressing **[♦] [OFF]** or if the unit turned itself off through APD, the unit returns to whichever application you used last.

Turning the Calculator Off

You can use either of the following keys to turn off your graphing calculator.

Press:	Description
$\boxed{2\text{nd}} \text{ [OFF]}$ (press $\boxed{2\text{nd}}$ and then press [OFF])	Settings and memory contents are retained by the Constant Memory™ feature. However: <ul style="list-style-type: none">You cannot use $\boxed{2\text{nd}} \text{ [OFF]}$ if an error message is displayed.When you turn the calculator on again, it displays either the Home screen or the Apps desktop (regardless of the last application you used).
$\boxed{\blacklozenge} \text{ [OFF]}$ (press $\boxed{\blacklozenge}$ and then press [OFF])	Similar to $\boxed{2\text{nd}} \text{ [OFF]}$ except: <ul style="list-style-type: none">You can use $\boxed{\blacklozenge} \text{ [OFF]}$ if an error message is displayed.When you turn the calculator on again, it will be exactly as you left it.

Note: [OFF] is the second function of the $\boxed{\text{ON}}$ key.

APD (Automatic Power Down)

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

When you press $\boxed{\text{ON}}$, the calculator 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. If a program is running, but waiting for a key press, APD will occur after several minutes of inactivity.

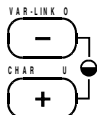
Setting the Display Contrast

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 graphing calculator 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:
Decrease (lighten) the contrast	 and 
Increase (darken) the contrast	 and 





Contrast keys



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 .

When to Replace Batteries

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 alkaline batteries.

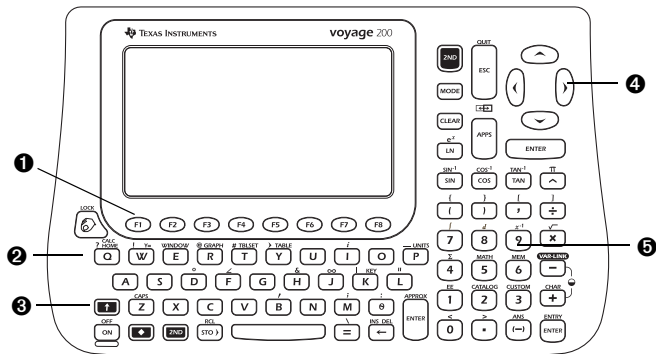
Note: The display may be very dark after you change batteries. Use   to lighten the display.

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

Indicator in status line	Description
	Batteries are low.
	Replace batteries as soon as possible.

The Voyage™ 200 Keyboard

With its 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. The keyboard is divided into several areas of related keys.



- ❶ **[F1] – [F8]** open toolbar menus.
- ❷ QWERTY keyboard functions the same as a computer keyboard.
- ❸ **[↕]**, **[2nd]**, **[↑]** and **[↻]** add functionality by increasing the available key commands.
- ❹ **[←]**, **[→]**, **[↶]**, and **[↷]** move the cursor.
- ❺ Numeric keypad performs math and scientific functions.

Moving the Cursor

To move the cursor in a particular direction, press the appropriate cursor key (**[←]**, **[→]**, **[↶]**, or **[↷]**).

Some applications also let you press:

- **[2nd]** **[←]** or **[2nd]** **[→]** to move to the beginning or end of a line.
- **[2nd]** **[↶]** or **[2nd]** **[↷]** to move up or down one screen at a time.
- **[↕]** **[↶]** or **[↕]** **[↷]** to move to the top or bottom of a page.

- and , and , and , or and to move diagonally. (Press the indicated cursor keys simultaneously.)

Modifier Keys

Modifier Keys

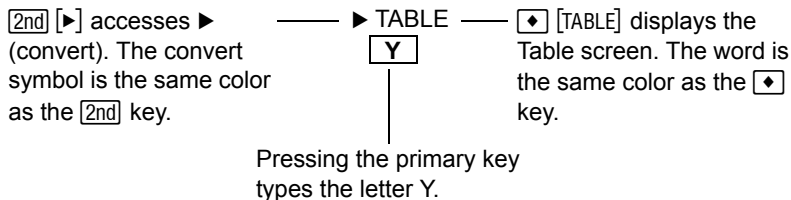
Modifier	Description
(second)	Accesses the second function of the next key you press. On the keyboard, these are printed in the same color as the key.
(diamond)	Activates keys that select certain applications, menu items, and other operations from the keyboard. On the keyboard, these are printed in the same color as the key.
(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.

Note: Information is available about using and .

Examples of [2nd] and [diamond] Modifiers

The key is one of several keys that can perform three operations, depending on whether you first press or .

The following Voyage™ 200 Graphing Calculator example shows using the $\boxed{2\text{nd}}$ or \blacklozenge modifier key with the Y alphabetic key.

















Some keys perform only one additional operation, which may require either $\boxed{2\text{nd}}$ or \blacklozenge , depending on the color in which the operation is printed on the keyboard and where it is positioned above the key.

When you press a modifier such as $\boxed{2\text{nd}}$ or \blacklozenge , a 2ND or \blacklozenge indicator appears in the status line at the bottom of the display. If you press a modifier by accident, press it again (or press $\boxed{\text{ESC}}$) to cancel its effect.

Other Important Keys You Need to Be Familiar With

Key	Description
\blacklozenge $\boxed{\text{Y=}}$	Displays the Y= Editor.
\blacklozenge $\boxed{\text{WINDOW}}$	Displays the Window Editor.
\blacklozenge $\boxed{\text{GRAPH}}$	Displays the Graph screen.
\blacklozenge $\boxed{\text{TBLSET}}$	Sets parameters for the Table screen.
\blacklozenge $\boxed{\text{TABLE}}$	Displays the Table screen.



Key	Description
 :  X (cut)  C (copy)  V (paste)	These keys let you edit entered information by performing a cut, copy, or paste operation.
2nd [⇄]	Toggles between the last two chosen Apps or between split screen portions.
2nd [CUSTOM]	Toggles the custom menu on and off.
2nd [▶]	Converts measurement units.
 2nd [-]	Designates a measurement unit.
	Deletes the character to the left of the cursor (backspaces).
2nd [INS]	Toggles between insert and overwrite mode for entering information.
 [DEL]	Deletes the character to the right of the cursor.
 2nd [1]	Enters the “ <i>with</i> ” operator, which is used in symbolic calculations.
2nd [∫], 2nd [d]	Performs integrations and derivatives.
2nd [∠]	Designates an angle in polar, cylindrical, and spherical coordinates.
2nd [MATH]	Displays the MATH menu.
2nd [MEM]	Displays the MEMORY screen.
2nd [VAR-LINK]	Displays the VAR-LINK screen for managing variables and Flash applications.

Key	Description
 [RCL]	Recalls the contents of a variable.
  [UNITS]	Displays the UNITS dialog box.
 [CHAR]	Displays the CHAR menu, which lets you select Greek letters, international accented characters, etc.
 [ENTRY],  [ANS]	Recalls the previous entry and the last answer, respectively.

Entering Alphabetic Characters

Alphabetic characters are used in expressions such as x^2+y^2 to enter variable names and in the Text Editor (*Text Editor* module).

Typing Alphabetic Characters on the Voyage™ 200

To:	Press:
Type a single lowercase alpha character.	the letter key
Type a single uppercase alpha character.	 and then the letter key (status line shows )
Type a space.	spacebar
Turn on lowercase alpha-lock.	(no action needed)

To:	Press:
Turn on uppercase ALPHA-lock.	$\boxed{2\text{nd}}$ $\boxed{[\text{CAPS}]}$
Turn off alpha-lock.	$\boxed{2\text{nd}}$ $\boxed{[\text{CAPS}]}$ (turns off uppercase lock)

For Special Characters

Use the $\boxed{2\text{nd}}$ $\boxed{[\text{CHAR}]}$ menu to select from a variety of special characters. For more information, refer to “Entering Special Characters” in the *Text Editor* module.

Entering Numbers

The keypad lets you enter positive and negative numbers for your calculations. You can also enter numbers in scientific notation.

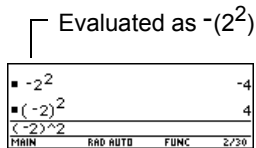
Entering a Negative Number

1. Press the negation key $\boxed{[-]}$. (Do not use the subtraction key $\boxed{[-]}$.)
2. Type the number.

To see how your graphing calculator evaluates a negation in relation to other functions, refer to the Equation Operating System (EOS™) hierarchy in the *Technical Reference*

module. For example, it is important to know that functions such as x^2 are evaluated before negation.

Use \square and \square to include parentheses if you have any doubt about how a negation will be evaluated.



If you use \square instead of \square (or vice versa), you may get an error message or you may get unexpected results. For example:

- $9 \times \square 7 = -63$
– but –
 $9 \times \square 7$ displays an error message.
- $6 \square 2 = 4$
– but –
 $6 \square 2 = -12$ since it is interpreted as $6(-2)$, implied multiplication.
- $\square 2 \square 4 = 2$
– but –
 $\square 2 \square 4$ subtracts 2 from the previous answer and then adds 4.

Important: Use \square for subtraction and use \square for negation.

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 and the magnitude of the number.

■ 1.2345	1.2345
123.45E-2	
MAIN	RAD AUTO FUNC 1/20

Represents 123.45×10^{-2}

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.

Definitions

Expression	<p>Consists of numbers, variables, operators, functions, and their arguments that evaluate to a single answer. For example: $\pi r^2 + 3$.</p> <ul style="list-style-type: none">• 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	<p>Performs an operation such as +, −, *, ^.</p> <ul style="list-style-type: none">• Operators require an argument before and after the operator. For example: 4+5 and 5^2.
Function	<p>Returns a value.</p> <ul style="list-style-type: none">• Functions require one or more arguments (enclosed in parentheses) after the function. For example: $\sqrt{(5)}$ and min(5,8).
Instruction	<p>Initiates an action.</p> <ul style="list-style-type: none">• 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. <p>Note: For instructions, do not put the arguments in parentheses.</p>

Notes:

- The *Technical Reference* module describes all of the built-in functions and instructions.

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

Implied Multiplication

The graphing calculator recognizes implied multiplication, provided it does not conflict with a reserved notation.

	If you enter:	The calculator interprets it as:
Valid	2π	$2*\pi$
	$4 \sin(46)$	$4*\sin(46)$
	$5(1+2)$ 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 the *Technical Reference* module. To change the order of evaluation or just to ensure that an expression is evaluated in the order you require, use parentheses.

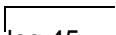
Calculations inside a pair of parentheses are completed first. For example, in $4(1+2)$, EOS first evaluates $(1+2)$ and then multiplies the answer by 4.

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 $3.76 \div (-7.9 + \sqrt{5}) + 2 \log 45$.  Type the function name in this example.

Press	Display
-------	---------

3.76 \div	3.76/(-7.9+ $\sqrt{($
((-) 7.9 (+)	[2nd] [$\sqrt{}$] inserts $\sqrt{()}$ because its argument must be in parentheses.
[2nd] [$\sqrt{}$]	

5))	3.76/(-7.9+ $\sqrt{(5)}$
	Use) once to close $\sqrt{(5)}$ and again to close $(-7.9 + \sqrt{5})$.

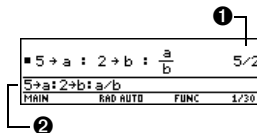
+ 2	3.76/(-7.9+ $\sqrt{(5)}$)+2log(45)
LOG	log requires () around its argument.
(45)	

ENTER	$\frac{3.76}{-7.9 + \sqrt{5}} + 2 \cdot \log(45)$ $\frac{3.76 / (-7.9 + \sqrt{5}) + 2 \log(45)}{2.64258}$				
	<table border="1"> <tr> <td>MAIN</td> <td>RND AUTO</td> <td>FUNC</td> <td>1/30</td> </tr> </table>	MAIN	RND AUTO	FUNC	1/30
MAIN	RND AUTO	FUNC	1/30		

Note: You can also select **log** by using [2nd] [CATALOG]

Entering Multiple Expressions on a Line

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

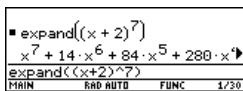


- 1 Displays last result only.
- 2 \blacktriangleright is displayed when you press $\boxed{\text{STO} \blacktriangleright}$ to store a value to a variable.

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, \blacktriangleright is displayed at the end of the line.



To view the entire entry or answer:

1. Press \odot to move the cursor from the entry line up into the history area. This highlights the last answer.

- As necessary, use \leftarrow and \rightarrow to highlight the entry or answer you want to view. For example, \leftarrow moves from answer to entry, up through the history area.
- Use \rightarrow and \leftarrow or $2\text{nd} \rightarrow$ and $2\text{nd} \leftarrow$ to scroll right and left.


```

expand((x + 2)^7)
| 0 · x^3 + 672 · x^2 + 448 · x + 128
expand((x+2)^7)
MIN      RAD AUTO  FUNC  1/1
      
```

Note: When you scroll to the right, \blacktriangleleft is displayed at the beginning of the line.
- To return to the entry line, press ESC .

Continuing a Calculation

When you press ENTER to evaluate an expression, the graphing calculator 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 calculator:
\oplus , \ominus , \otimes , \oslash , \wedge , or $\text{STO}\rightarrow$	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 $3.76 \div (-7.9 + \sqrt{5})$. Then add 2 log 45 to the result.

Press

Display

3.76 \div () (-) 7.9 (+)

2nd (✓) 5 () ()

ENTER

+ 2 LOG

() 45 ()

ENTER

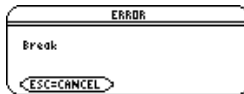
3.76	- .66385
-7.9 + 15	
- .66384977522033 + 2 · log(2.64258
ans(1)+2log(45)	
MAIN	RAD AUTO FUNC 2/30

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, BUSY 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 modes and their settings that affect the display formats. You can check or change your current mode settings.

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	OFF
$\pi^2, \frac{\pi}{2}, \sqrt{\frac{x-3}{2}}$	$\pi^2, \pi/2, \sqrt{((x-3)/2)}$

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.



```
1: AUTO
2: EXACT
3: APPROXIMATE
```

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

■ $2.5 \cdot 2$	5
■ $2.5 \cdot 3$	$15/2$
■ $6/3$	2
■ $6/4$	$3/2$
$6/4$	
MAIN	RAD EXACT FUNC 4/30

Shows whole-number results.

Shows simplified fractional results.

■ $2 \cdot \pi$	$2 \cdot \pi$
■ $\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$
■ $\sqrt{4/7}$	$2 \cdot \sqrt{1/7}$
$\sqrt{4/7}$	$\frac{2}{7}$
MAIN	RAD EXACT FUNC 3/30

Shows symbolic π .

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

■ $\sqrt{4/7}$	$2 \cdot \sqrt{1/7}$
■ $\sqrt{4/7}$.755929
$\sqrt{4/7}$	
MAIN	RAD EXACT FUNC 4/30

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

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

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

Note: Results are rounded to the precision of your graphing calculator and displayed according to current mode settings.

■ $2.5 \cdot 2$	5.
■ $2.5 \cdot 3$	7.5
■ $6 \div 3$	2.
■ $6 \div 4$	1.5
5/4	
MAIN	RAD APPROX FUNC 4/30

Fractional results are evaluated numerically.

■ $2 \cdot \pi$	6.28319
■ $\frac{\sqrt{2}}{2}$.707107
■ $\sqrt[3]{4 \div 7}$.755929
$\sqrt[3]{(4 \div 7)}$	
MAIN	RAD APPROX FUNC 3/30

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.

■ $2 \cdot \pi$	$2 \cdot \pi$
■ $2 \cdot .\pi$	6.28319
■ $\sqrt[3]{4 \div 7}$	$\frac{2 \cdot \sqrt[3]{7}}{7}$
■ $\frac{4.}{7}$.755929
$\sqrt[3]{(4. \div 7)}$	
MAIN	RAD AUTO FUNC 4/30

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

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

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
$\sqrt{(2)/2}$	$\frac{\sqrt{2}}{2}$.707107	$\frac{\sqrt{2}}{2}$
$\pi*2$	$2\cdot\pi$	6.28319	$2\cdot\pi$
$\pi*2.$	$2\cdot\pi$	6.28319	6.28319

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

Note: To evaluate an entry in APPROXIMATE form, regardless of the current setting, press \blacklozenge **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 calculator calculates and retains all decimal results with up to 14 significant digits (although a maximum of 12 are displayed).

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

Notes:

- Regardless of the **Display Digits** setting, the full value is used for internal floating-point calculations to ensure maximum accuracy.
- A result is automatically shown in scientific notation if its magnitude cannot be displayed in the selected number of digits.

Exponential Format Mode

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



Setting	Example	Description
NORMAL	12345.6	If a result cannot be displayed in the number of digits specified by the Display Digits mode, the calculator switches from NORMAL to SCIENTIFIC for that result only.
SCIENTIFIC	$\begin{array}{c} 1.23456E\ 4 \\ \hline \textcircled{1} \quad \textcircled{2} \end{array}$	1.23456×10^4
ENGINEERING	$\begin{array}{c} 12.3456E\ 3 \\ \hline \textcircled{3} \quad \textcircled{4} \end{array}$	12.3456×10^3

- ❶ Always 1 digit to the left of the decimal point.
- ❷ Exponent (power of 10).
- ❸ May have 1, 2, or 3 digits to the left of the decimal point.
- ❹ Exponent is a multiple of 3.

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

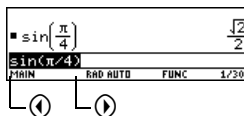
Editing an Expression in the Entry Line

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 calculator 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 beginning.

➡ moves the cursor to the end of the expression.







Moving the Cursor

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

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 ➡




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.

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.	 (once only) If there are no characters to the right of the cursor,  erases the entire entry line.

Clearing the Entry Line

To clear the entry line, press:

-  if the cursor is at the beginning or end of the entry line.
– or –
-   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 calculator has both an insert and an overwrite mode. By default, the calculator is in the insert mode. To toggle between the insert and overwrite modes, press **2nd** [INS].

If 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.

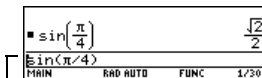
Note: Look at the cursor to see if you're in insert or overwrite mode.

Replacing or Deleting Multiple Characters

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

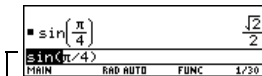
To highlight multiple characters:

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 \uparrow and press \leftarrow or \rightarrow to highlight characters left or right of the cursor.



Hold \uparrow and press \leftarrow \rightarrow \rightarrow \rightarrow .

To replace or delete the highlighted characters:

1. Type the new characters.
2. Press \leftarrow .



Note: When you highlight characters to replace, remember that some function keys automatically add an open parenthesis.

Menus

To leave the keyboard uncluttered, the calculator 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 modules.

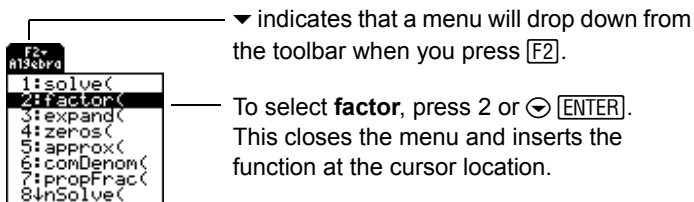
Displaying a Menu

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]	Apps desktop or APPLICATIONS menu — Lets you select from a list of applications.
[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 built-in functions and instructions. Also lets you select user-defined functions or Flash application functions (if any have been defined or loaded).
[2nd] [CUSTOM]	CUSTOM menu — Lets you access a menu that you can customize to list any available function, instruction, or character. The calculator includes a default custom menu, which you can modify or redefine. Refer to the <i>Calculator Home Screen</i> and/or the <i>Programming</i> module for more information on the custom menu.

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 \ominus and \oplus to highlight the item, and then press $\boxed{\text{ENTER}}$. (Note that pressing \ominus from the first item moves the highlight to the last item, and vice versa.)



factor(

Items Ending with ► (Submenus)

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



↓ 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.

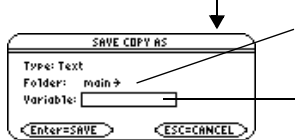
- To display the submenu for the highlighted item, press \odot . (This is the same as selecting that item.)
- To cancel the submenu without making a selection, press \ominus . (This is the same as pressing $\boxed{\text{ESC}}$.)
- To wrap to the last menu item directly from the first menu item, press \ominus . To wrap to the first menu item directly from the last menu item, press \odot .

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 select a folder name and type 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.

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.

Moving from One Toolbar Menu to Another

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

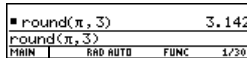
- Press the key (**F1**, **F2**, etc.) for the other toolbar menu.
– or –
- Use the cursor pad to move to the next (press **⏏**) or previous (press **⏐**) toolbar menu. Pressing **⏏** from the last menu moves to the first menu, and vice versa.

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

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 $\boxed{2nd}$ $\boxed{[MATH]}$ to display the **MATH** menu.
2. Press **1** to display the **Number** submenu. (Or press \boxed{ENTER} since the first item is automatically highlighted.)
3. Press **3** to select **round**. (Or press \odot \odot and \boxed{ENTER} .)
4. Press $\boxed{2nd}$ $\boxed{[\pi]}$ $\boxed{,}$ $\boxed{3}$ $\boxed{)}$ and then \boxed{ENTER} to evaluate the expression.



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

Selecting an Application

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

From the APPLICATIONS Menu

1. If the Apps desktop is off, press **[APPS]** to display a menu that lists the applications.

Note: To cancel the menu without making a selection, press **[ESC]**.

2. Select an application. Either:

- Use the cursor pad **⬅** or **➡** to highlight the application and then press **[ENTER]**.
– or –
- Press the number or letter for that application.

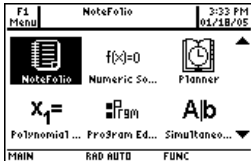


Application:	Lets you:
FlashApps	Display a list of Flash applications, if any.
Y= Editor	Define, edit, and select functions or equations for graphing.
Window Editor	Set window dimensions for viewing a graph.
Graph	Display graphs.
Table	Display a table of variable values that correspond to an entered function.
Data/Matrix Editor	Enter and edit lists, data, and matrices. You can perform statistical calculations and graph statistical plots.
Program Editor	Enter and edit programs and functions.
Text Editor	Enter and edit a text session.

Application:	Lets you:
Numeric Solver	Enter an expression or equation, define values for all but one variable, and then solve for the unknown variable.
Home	Enter expressions and instructions, and perform calculations.

From the Apps Desktop

Press the first letter of the application name, or use the cursor keys to highlight an application icon on the Apps desktop and press **[ENTER]**. (If you press the first letter of the application and more than one application begins with that letter, the first one alphabetically is highlighted). The application either opens directly or displays a dialog box. (Your Apps desktop may vary from the one shown below.)



The most common dialog box lists these options for the application:

Option	Description
Current	Returns the screen displayed when you last viewed the App. (If there is no current file/variable for the selected App, this option defaults to New if you press [ENTER] .)
Open	Lets you select an existing file.

Option	Description
New	Creates a new file with the name typed in the field.

Select an option and press **[ENTER]**. The application appears.

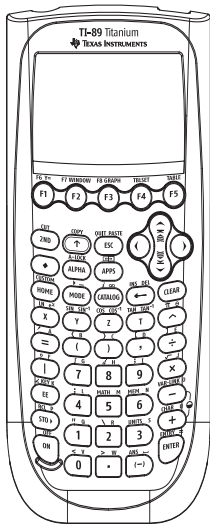
Note: The general term *variable* is used to refer to the application data files that you create.

Use any of these methods to return to the Apps desktop from within an application:

- Press **[APPS]**.
- In full-screen mode, press **[2nd] [QUIT]**.
- In split-screen mode, press **[2nd] [QUIT]** to open the full-screen view of the active application, then press **[2nd] [QUIT]** again.

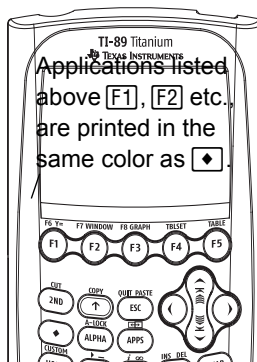
To return to the last open application from the Apps desktop, press **[2nd] [⇧]**.

From the Keyboard

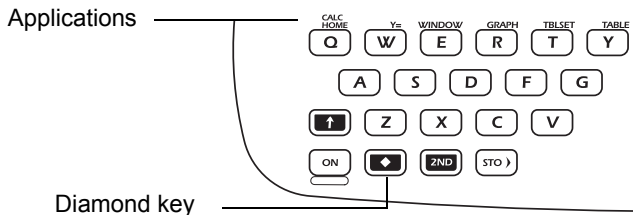


You can access commonly used applications from the keyboard. This guidebook uses the notation \blacklozenge [Y=], similar to the notation used in second functions.

Application:	Press:
Home	$\boxed{\text{HOME}}$ $\boxed{\text{[CALC HOME]}}$
Y= Editor	\blacklozenge [Y=]
Window Editor	\blacklozenge [WINDOW]
Graph	\blacklozenge [GRAPH]
Table Setup	\blacklozenge [TBLSET]
Table Screen	\blacklozenge [TABLE]



On the Voyage™ 200, some commonly used applications are listed above the QWERTY keys.



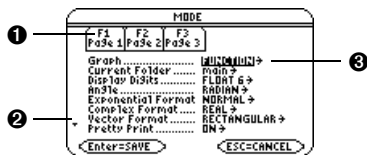
Note: $\boxed{2\text{nd}}$ key functions are not shown in the above graphic.

Setting Modes

Modes control how numbers and graphs are displayed and interpreted. Mode settings are retained by the Constant Memory™ feature when the graphing calculator 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.



- ❶ There are three pages of mode listings. Press **F1**, **F2**, or **F3** to quickly display a particular page.
- ❷ Indicates you can scroll down to see additional modes.
- ❸ → 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 **Page 2**, **Split 2 App** is not valid when **Split Screen = FULL**. When you scroll through the list, the cursor skips dimmed settings.

Changing Mode Settings

From the **MODE** dialog box:

1. Highlight the mode setting you want to change. Use \downarrow or \uparrow (with $\boxed{F1}$, $\boxed{F2}$, or $\boxed{F3}$) to scroll through the list.
2. Press \rightarrow or \leftarrow to display a menu that lists the valid settings. The current setting is highlighted.
3. Select the applicable setting. Either:
 - Use \downarrow or \uparrow to highlight the setting and press \boxed{ENTER} .
 - or –
 - Press the number or letter for that setting.

Note: To cancel a menu and return to the **MODE** dialog box without making a selection, press \boxed{ESC} .
4. Change other mode settings, if necessary.
5. When you finish all your changes, press \boxed{ENTER} to save the changes and exit the dialog box.

Important: If you press \boxed{ESC} instead of \boxed{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, 3D, or DE.
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 <i>Calculator Home Screen</i> .
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.
Angle	Units in which angle values are interpreted and displayed: RADIAN, DEGREE or GRADIAN.
Exponential Format	Notation used to display results: NORMAL, SCIENTIFIC, or ENGINEERING.
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.

Mode	Description
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 the <i>Split Screens</i> module.
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 <i>Calculator Home Screen</i> .
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.
Base	Lets you perform calculations by entering numbers in decimal (DEC), hexadecimal (HEX), or binary (BIN) form.
Unit System	Lets you select from three systems of measurement to specify the default units for displayed results: SI (metric or MKS); Eng/US (feet, pounds, etc.); or Custom .
Custom Units	Lets you select custom defaults. The mode is dimmed until you select Unit System, 3:CUSTOM .

Mode	Description
Language	Lets you localize the calculator into one of several languages, depending on which language Flash applications are installed.
Apps Desktop	Turns the Apps desktop ON or OFF.

Using the Clean Up Menu to Start a New Problem

On the Home screen, the **Clean Up** toolbar menu lets you start a new calculation from a cleared state without resetting the memory.

Clean Up Toolbar Menu

To display the **Clean Up** menu from the Home screen, press:

[F6]



Menu Item	Description
Clear a-z	Clears (deletes) all single-character variable names in the current folder, unless the variables are locked or archived. You will be prompted to press [ENTER] to confirm the action.

Menu Item	Description
	<p>Single-character variable names are often used in symbolic calculations such as:</p> <p>solve(a·x²+b·x+c=0,x)</p> <p>If any of the variables have already been assigned a value, your calculation may produce misleading results. To prevent this, you can select 1:Clear a–z before beginning the calculation.</p>
NewProb	<p>Places NewProb in the entry line. You must then press ENTER to execute the command.</p> <p>NewProb performs a variety of operations that let you begin a new problem from a cleared state without resetting the memory:</p> <ul style="list-style-type: none"> • Clears all single-character variable names in the current folder (same as 1:Clear a–z), unless the variables are locked or archived. • Turns off all functions and stat plots (FnOff and PlotsOff) in the current graphing mode. • Performs ClrDraw, ClrErr, ClrGraph, ClrHome, ClrIO, and ClrTable.
Restore custom default	<p>If a custom menu other than the default is in effect, this lets you restore the default. Refer to the <i>Calculator Home Screen</i> module for information on the custom menu.</p>

Notes:

- When defining a variable that you want to retain, use more than one character in the name. This prevents it from being deleted inadvertently by **1:Clear a–z**.
- For information about checking and resetting memory or other system defaults, refer to *Memory and Variable Management*.

Using the Catalog Dialog Box

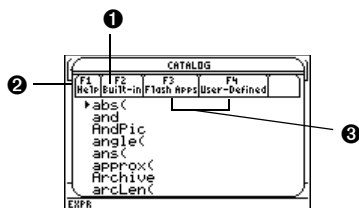
The CATALOG provides a way to access any built-in command (functions and instructions) from one convenient list. In addition, the CATALOG dialog box lets you select functions used in Flash applications or user-defined functions (if any have been loaded or defined).

Displaying the CATALOG

To display the CATALOG dialog box, press:

[2nd] **[CATALOG]**

The CATALOG defaults to **[F2]** **Built-in**, which displays an alphabetic list of all pre-installed commands (functions and instructions).



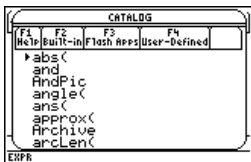
- 1 Defaults to **[F2]** **Built-in**.
- 2 **[F1]** Help displays a command's parameters in a dialog box.
- 3 **[F3]** and **[F4]** allow access to Flash application functions and User-Defined functions and programs.

Note: Options that are not currently valid are dimmed. For example, **[F3] Flash Apps** is dimmed if you have not installed a Flash application. **[F4] User-Defined** is dimmed if you have not created a function or a program.

Selecting a Built-in Command 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.

1. Press:
[2nd] [CATALOG]
2. Press **[F2] Built-in**.



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

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

3. Move the ► indicator to the command, and press **[ENTER]**.

To move the ► indicator:

Press or type:

One function or program at a time **[↓]** or **[↑]**

One page at a time **[2nd] [↓]** or **[2nd] [↑]**

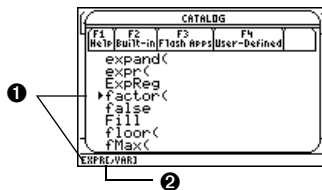
To move the ► indicator:**Press or type:**

To the first function that
begins with a specified letterThe letter key.

Note: From the top of the list, press \ominus to move to the bottom. From the bottom, press \odot to move to the top.

Information about Parameters

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



- ❶ Indicated command and its parameters
- ❷ Brackets [] indicate optional parameters

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

factor(*expression*) required

– or –

factor(*expression, variable*) optional

Note: For details about the parameters, refer to that command's description in the *Technical Reference* module.

Viewing CATALOG Help

You can display a command's parameters in a dialog box by pressing **[F1] Help**. The parameters are the same as those displayed on the status line.

Indicated
command and its
parameters.



Some commands, such as **CirDraw**, do not require parameters. If you select one of these commands, parameters will not display on the status line and you will see Unavailable if you press **[F1] Help**.

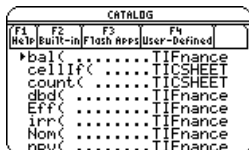
Press **[ESC]** to exit the CATALOG Help dialog box.

Selecting a Flash Application Function

A Flash application may contain one or more functions. When you select a function, its name is inserted in the entry line at the cursor location. Therefore, you should position the cursor as necessary before selecting the function.

1. Press:
[2nd] [CATALOG]

2. Press **[F3] Flash Apps**. (This option is dimmed if no Flash applications are installed.)



F1	F2	F3	F4
Help	Exit-In	Flash Apps	User-Defined
▶bal	TIFinance	
cellI	TICSHEET	
count	TICSHEET	
dbd	TIFinance	
Eff	TIFinance	
irr	TIFinance	
Nom	TIFinance	
npv	TIFinance	

- The list is alphabetized by function name. The left column lists functions. The right column lists the Flash application that contains the function.
- Information about a function is displayed in the status line.
- To exit without selecting a function, press **[ESC]**.

3. Move the ► indicator to the function, and press **[ENTER]**.

To move the ► indicator:	Press or type:
One function or program at a time	⏪ or ⏩
One page at a time	[2nd] ⏪ or [2nd] ⏩
To the first function that begins with a specified letter	The letter key.

Selecting a User-Defined Function or Program

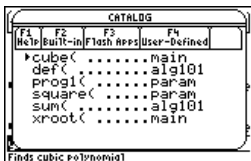
You can create your own functions or programs and then use **[F4] User-Defined** to access them. For instructions on how to create functions, see “Creating and Evaluating User-Defined Functions” in *Calculator Home Screen*, and “Overview of Entering a Function” in the *Programming* module. See *Programming* for instructions on how to create and run a program.

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

1. Press:

[2nd] [CATALOG]

2. Press **[F4]** **User-Defined**. (This option is dimmed if you have not defined a function or created a program.)



- The list is alphabetized by function / program name. The left column lists functions and programs. The right column lists the folder that contains the function or program.
- If the function or program's first line is a comment, the comment text is displayed in the status line.
- To exit without selecting a function or program, press **[ESC]**.

Note: Use the **VAR-LINK** screen to manage variables, folders, and Flash applications. See the *Memory and Variable Management* module.

3. Move the ▶ indicator to the function or program, and press **[ENTER]**.

To move the ▶ indicator:

Press or type:

One function or program at a time ◀ or ▶

One page at a time **[2nd]** ◀ or **[2nd]** ▶

To the first function or program that begins with a specified letter The letter key.

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 calculator encounters the name in an expression, it substitutes the variable's stored value.

Rules for Variable Names

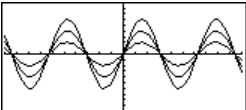
A variable name:

- 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 calculator. Preassigned names include:
 - Built-in functions (such as **abs**) and instructions (such as **LineVert**). Refer to the *Technical Reference* module.
 - System variables (such as **xmin** and **xmax**, which are used to store graph-related values). Refer to the *Technical Reference* module 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

DataTypes	Examples
Expressions	2.54, 1.25E6, 2π , $x_{\min}/10$, $2+3i$, $(x-2)^2$, $\sqrt{2}/2$
Lists	{2 4 6 8}, {1 1 2}
Matrices	$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 & 0 \\ 3 & 4 & 6 \end{bmatrix}$
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 $\boxed{\text{STO}}\blacktriangleright$. The store symbol (\rightarrow) is displayed.

3. Type the variable name.

Note: TI-89 Titanium users should use $\boxed{\alpha}$ as necessary when typing variable names.

$5 + 8^3 \rightarrow \text{num1}$	517
$5 + 8^3 \rightarrow \text{num1}$	
MAIN	RAD AUTO FUNC 1/30

4. Press $\boxed{\text{ENTER}}$.

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

Displaying a Variable

1. Type the variable name.
2. Press $\boxed{\text{ENTER}}$.

num1	517
num1	
MAIN	RAD AUTO FUNC 1/30

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

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

num1	517
num1 + a	a + 517
num1 + a	
MAIN	RAD AUTO FUNC 2/30

Note: Refer to *Symbolic Manipulation* for information about symbolic manipulation.

Using a Variable in an Expression

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

Note: To view a list of existing variable names, use **[2nd]** **[VAR-LINK]** as described in *Memory and Variable Management*.

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

3 · num1	1551
num1	517
num1	
MAIN	RAD AUTO FUNC 2/30

The variable's value did not change.

3 · num1 → num1	1551
num1	1551
num1	
MAIN	RAD AUTO FUNC 2/30

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.

F1=	F2=	F3=	F4=	F5=	F6=
Tools	13	Calc	Other	Pr	Clean Up
RECALL VARIABLE					
Recall: num1					
<Enter>=OK			<ESC>=CANCEL		
2:4					
MAIN	RAD AUTO	FUNC	0/30		

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

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 calculator, including several important mode settings.


Status Line Indicators



- ❶ Current Folder
- ❷ Modifier Key
- ❸ Angle Mode
- ❹ Exact/Approx Mode
- ❺ Graph Number
- ❻ Graph Mode
- ❼ Replace Batteries
- ❽ History Pairs, Busy/Pause, Locked Variable

Indicator	Meaning
Current Folder	Shows the name of the current folder. Refer to “Using Folders to Store Independent Sets of Variables” in <i>Calculator Home Screen</i> . MAIN is the default folder.
Modifier Key	Shows which modifier key is in effect, as described below.
2nd	2nd — will use the second function of the next key you press.

Indicator	Meaning
◆	◆ — will use the diamond feature of the next key you press.
⬆	⬆ — will type the uppercase letter for the next key you press.
↔	When used in combination with a cursor key, the Voyage™ 200 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
GRAD	Gradian
Exact/Approx Mode	Shows how answers are calculated and displayed. To change the Exact/Approx mode, use the MODE key.
AUTO	Auto
EXACT	Exact
APPROX	Approximate
Graph Number	If the screen is split to show two independent graphs, this indicates which graph is active — G1 or G2 . (Displays GR#1 or GR#2 on the Voyage™ 200.)
Graph Mode	Indicates the type of graphs that can be plotted. To change the Graph mode, use the MODE key.
FUNC	y(x) functions

Indicator	Meaning
PAR	$\mathbf{x(t)}$ and $\mathbf{y(t)}$ parametric equations
POL	$\mathbf{r(\theta)}$ polar equations
SEQ	$\mathbf{u(n)}$ sequences
3D	$\mathbf{z(x,y)}$ 3D equations
DE	$\mathbf{y'(t)}$ differential equations
Battery	Displayed only when the batteries are getting low. If BATT is shown with a black background, change the batteries as soon as possible.
History Pairs, Busy/Pause, Archived	The information shown in this part of the status line depends on the application you are using.
23/30	Displayed on the Home screen to show the number of entry/answer pairs in the history area. Refer to History Information on the Status Line in the <i>Calculator Home Screen</i> module.
BUSY	A calculation or graph is in progress.
PAUSE	You paused a graph or program.
	The variable opened in the current editor (Data/Matrix Editor, Program Editor, or Text Editor) is locked or archived and cannot be modified.

Notes:

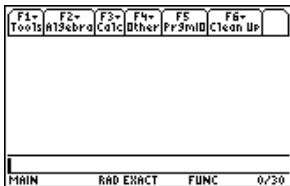
- To cancel $\boxed{2nd}$, $\boxed{\blacklozenge}$, $\boxed{\alpha}$, or $\boxed{\uparrow}$, press the same key again or press a different modifier key.

- If the next key you press does not have a diamond feature or an associated letter, the key performs its normal operation.

Calculator Home Screen

Calculator Home Screen

The calculator Home screen is the starting point for math operations, including executing instructions, evaluating expressions, and viewing results.



A blank calculator Home screen

This module describes the parts of the calculator Home screen, how to scroll through or modify the history area; how to use cut, copy, and paste, and more.

Note: The term “calculator Home screen” is used in this module. Other modules use the term “Home screen.” Both terms are interchangeable and refer to the same screen.

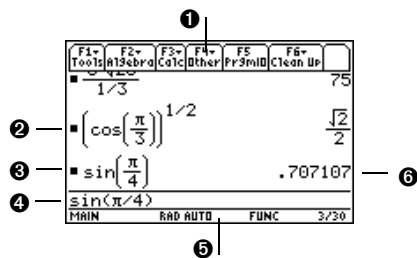
Displaying the Calculator Home Screen

When you first turn on your Voyage™ 200 Graphing Calculator, the Apps desktop is displayed. To display the calculator Home screen, highlight the Home icon and press **[ENTER]**. You can also display the calculator Home screen by pressing **[♦] [CALC HOME]**

(Voyage™ 200). If you turn off the Apps Desktop mode, the calculator Home screen appears automatically.

Parts of the Calculator Home Screen

The following example contains previously entered data and describes the main parts of the calculator Home screen. Entry/answer pairs in the history area are displayed in “pretty print.” Pretty print displays expressions in the same form in which they are written on the board or in textbooks.



1 Toolbar

Lets you display menus for selecting operations applicable to the calculator Home screen. To display a toolbar menu, press $\boxed{F1}$, $\boxed{F2}$, etc.

2 Pretty Print Display

Shows exponents, roots, fractions, etc., in traditional form.

3 Last Entry

Your last entry.

4 Entry Line

Where you enter expressions or instructions.

5 Status Line

Shows the current state of the calculator, including several important mode settings.

6 Last Answer

Result of your last entry. Note that results are not displayed on the entry line. Note: \blacklozenge $\boxed{\text{ENTER}}$ (Approx) was used in this example.

The following example shows an answer that is not on the same line as the expression. Note that the answer is longer than the screen width. An arrow (\blacktriangleright) indicates the answer is continued. The entry line contains ellipsis (...). Ellipsis indicates the entry is longer than the screen width.

The image shows a calculator screen with a fraction calculation. The top status bar displays function keys: F1, F2, F3, F4, F5, F6. Below it are icons for Tools, 1/3, e, Brackets, Calc, Other, Pr, 3rd, and Clean Up. The main display shows the expression $\text{comDenom}\left(\frac{y^2 + y}{(x + 1)^2} + y^2 + y\right)$. Below this, the numerator $x^2 \cdot y^2 + x^2 \cdot y + 2 \cdot x \cdot y^2 + 2 \cdot y$ is shown above the denominator $x^2 + 2 \cdot x + 1$. An arrow points from the end of the denominator to the right. Below the screen, four numbered annotations are present: 1 points to the expression, 2 points to the fraction bar, 3 points to the arrow, and 4 points to the ellipsis in the bottom status bar.

1 Last Entry

"Pretty print" is ON. Exponents, roots, fractions, etc., are displayed in the same form in which they are traditionally written.

2 History Area

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

3 Answer Continues

Highlight the answer and press \blacktriangleright to scroll right and view the rest of it. Note that the answer is not on the same line as the expression.

⏪ Expression Continues (...)

Press \blacktriangleright to scroll right and view the rest of the entry. Press $\boxed{2\text{nd}} \blacktriangleleft$ or $\boxed{2\text{nd}} \blacktriangleright$ to go to the beginning or end of the entry line.

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.

Scrolling through the History Area

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

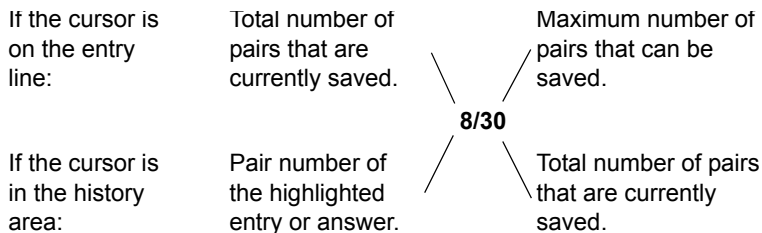
To:	Do this:
View entries or answers that have scrolled off the screen	<ul style="list-style-type: none">• From the entry line, press \blacktriangleleft to highlight the last answer.• Continue using \blacktriangleleft to move the cursor from answer to entry, up through the history area.
Go to the oldest or newest history pair	If the cursor is in the history area, press $\boxed{\blacklozenge} \blacktriangleleft$ or $\boxed{\blacklozenge} \blacktriangleright$, respectively.

To:	Do this:
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 beginning or end), respectively.
Return the cursor to the entry line	Press [ESC], or press ⏪ until the cursor is back on the entry line.

Note: An example of viewing a long answer is available.


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 CirHome on the entry line.
Delete a particular entry/answer pair	Move the cursor to either the entry or answer. Press ⬅ or [CLEAR] .

Saving the Calculator Home Screen Entries as a Text Editor Script

To save all the entries in the history area, you can save the calculator 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

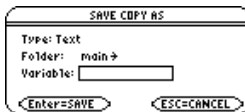
From the calculator Home screen:

1. Press **[F1]** and select **2:Save Copy As**.



- Specify a folder and text variable that you want to use to store the entries.

Note: Only the entries are saved, not the answers.



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.

Note: For information about folders, see the *Memory and Variable Management module*.

- 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 calculator Home screen. (On the calculator Home screen's **F1** toolbar menu, **1:Open** is not available.) Instead:

1. Use the Text Editor to open the variable containing the saved calculator 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 calculator Home screen.

This split screen shows the Text Editor (with the command line script) and the restored calculator Home screen.

Note: For complete information on using the Text Editor and executing a command script, refer to the *Text Editor* module.

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 clipboard, which is an area in memory that serves as a temporary storage location.

Auto-paste vs. Cut/Copy/Paste

Auto-paste 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.
2. Press **[F1]** and select **4:Cut** or **5:Copy**.

Clipboard = (empty or the previous contents)



After cut

After copy

solve(=0, x)
MAIN RAD AUTO FUNC 0/30

solve(x^4-3x^3-6x^2+8x=0, ...
MAIN RAD AUTO FUNC 0/30

Clipboard = $x^4-3x^3-6x^2+8x$

Clipboard = $x^4-3x^3-6x^2+8x$

Note: You can cut, copy or paste without having to use the **[F1]** toolbar menu. Press:



X, C, or V

Cutting is not the same as deleting. When you delete information, it is not placed in the clipboard and cannot be retrieved.

Note: When you cut or copy information, it replaces the clipboard's previous contents, if any.

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 key 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.



solve($x^4-3x^3-6x^2+8x=0, \dots$)
MAIN RAD AUTO FUNC 0/230


b) Press:



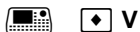
c) For this example, press  to evaluate the entry.

2. Paste the copied information into a new entry.

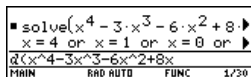
a) Begin a new entry and place the cursor where you want to paste the copied information.

b) Press  1 to select the d (differentiate) function.


c) Press:

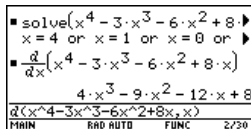


to paste the copied expression.



■ solve($x^4-3x^3-6x^2+8x$)
 $x=4$ or $x=1$ or $x=0$ or ▶
 $\frac{d}{dx}(x^4-3x^3-6x^2+8x)$
MAIN RAD AUTO FUNC 1/230

d) Complete the new entry, and press .



■ solve($x^4-3x^3-6x^2+8x$)
 $x=4$ or $x=1$ or $x=0$ or ▶
■ $\frac{d}{dx}(x^4-3x^3-6x^2+8x)$
 $4x^3-9x^2-12x+8$
 $\frac{d}(x^4-3x^3-6x^2+8x, x)$
MAIN RAD AUTO FUNC 2/230

Note: You can also reuse an expression by creating a user-defined function.

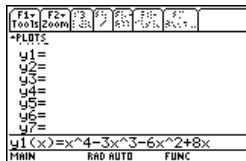
3. Paste the copied information into a different application.

a) Press \blacklozenge [Y=] to display the Y= Editor.

b) Press $\boxed{\text{ENTER}}$ to define $y_1(x)$.

c) Press:

 \blacklozenge **V**
to paste.



d) Press $\boxed{\text{ENTER}}$ to save the new definition.

Note: By copying and pasting, you can easily transfer information from one application to another.

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

When you press $\boxed{\text{ENTER}}$ to evaluate an expression, the Voyage™ 200 Graphing Calculator 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. As shown below, set the initial variable value and then enter the variable expression. Next, reenter to increment

Voyage™ 200**Display**

0 [STO▶]

NUM

[ENTER]

■ 0 → num	0
0 → num	
MAIN	RAD AUTO FUNC 1/30

NUM [+ 1] [STO▶]

NUM

[2nd] [:] NUM [^] 2

[ENTER]

■ 0 → num	0
■ num + 1 → num : num ²	1
num+1→num:num ²	
MAIN	RAD AUTO FUNC 2/30

[ENTER] [ENTER]

■ 0 → num	0
■ num + 1 → num : num ²	1
■ num + 1 → num : num ²	4
■ num + 1 → num : num ²	9
num+1→num:num ²	
MAIN	RAD AUTO FUNC 4/30

Note: Reexecuting an entry “as is” is useful for iterative calculations that involve variables.

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

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

The example below uses 8 as the first guess and then displays the answer in its approximate floating-point form. You can edit and reexecute using 7.95 and continue until the answer is as accurate as you want.

Voyage™ 200	Display
8 [STO▶] R [2nd] [:] [2nd] [π] R [^] 2 [ENTER]	$8 \rightarrow r : \pi \cdot r^2 \quad 64 \cdot \pi$ $8 \rightarrow r : \pi \cdot r^2$ MAIN RAD AUTO FUNC 1/30
[◀] [ENTER]	$8 \rightarrow r : \pi \cdot r^2 \quad 64 \cdot \pi$ $8 \rightarrow r : \pi \cdot r^2 \quad 201.062$ $8 \rightarrow r : \pi \cdot r^2$ MAIN RAD AUTO FUNC 2/30
[⏪] [▶] [DEL] 7.95 [ENTER]	$8 \rightarrow r : \pi \cdot r^2 \quad 64 \cdot \pi$ $8 \rightarrow r : \pi \cdot r^2 \quad 201.062$ $7.95 \rightarrow r : \pi \cdot r^2 \quad 198.557$ $7.95 \rightarrow r : \pi \cdot r^2$ MAIN RAD AUTO FUNC 3/30

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

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.

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.

To recall:	Press:	Effect:
Previous entries	$\boxed{2\text{nd}}$ [ENTRY] repeatedly	Each press recalls the entry prior to the one shown on the entry line.

Note: You can also use the entry function to recall any previous entry. Refer to **entry()** in the *Technical Reference* module.

For example:

If the entry line contains the last entry, $\boxed{2\text{nd}}$ [ENTRY] recalls this entry.

If the entry line is edited or cleared, $\boxed{2\text{nd}}$ [ENTRY] recalls this entry.

F1	F2	F3	F4	F5	F6
Tools	13	Calc	Other	Pr	Clean Up
■ 8 * r : pi * r^2 64 * pi					
■ 8 * r : pi * r^2 201.062					
■ 7.95 * r : pi * r^2 198.557					
7.95 * r : pi * r^2					
MAIN		RAD AUTO		FUNC 3/30	

Recalling the Last Answer

Each time you evaluate an expression, the Voyage™ 200 stores the answer to the variable **ans(1)**. To insert this variable in the entry line, press $\boxed{2\text{nd}}$ [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 \times 4.2 [ENTER]

2. Find the yield.

147 \div $\text{[2nd] [ANS] [ENTER]}$

■ 1.7·4.2	7.14
■ $\frac{147}{7.14}$	20.5882
147÷ans(1)	
MIN	RND AUTO FUNC 2/20

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

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.

Note: Refer to **ans()** in the *Technical Reference* module.

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 $\boxed{2\text{nd}}$ [ENTRY] and $\boxed{2\text{nd}}$ [ANS] as described in the previous section, but there are differences.

For entries:	Pasting lets you:	$\boxed{2\text{nd}}$ [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:	$\boxed{2\text{nd}}$ [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.

Note: You can also paste information by using the $\boxed{F1}$ toolbar menu.

Auto-Pasting an Entry or Answer

1. On the entry line, place the cursor where you want to insert the entry or answer.
2. Press \odot to move the cursor up into the history area. This highlights the last answer.

3. Use \leftarrow and \rightarrow to highlight the entry or answer to auto-paste.

- \leftarrow moves from answer to entry up through the history area.
- You can use \rightarrow to highlight items that have scrolled off the screen

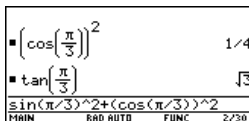


A calculator screen showing a history list. The first entry is $\cos\left(\frac{\pi}{3}\right)^2$ with a result of 1/4. The second entry is $\tan\left(\frac{\pi}{3}\right)$ with a result of $\sqrt{3}$. The third entry is $\sin\left(\frac{\pi}{3}\right)^2 + \cos\left(\frac{\pi}{3}\right)^2$. The status bar at the bottom shows "MAIN", "RAD AUTO", "FUNC", and "2/2".

Note: To cancel auto-paste and return to the entry line, press $\left[\text{ESC}\right]$. To view an entry or answer too long for one line (indicated by \blacktriangleright at the end of the line), use \uparrow and \downarrow or $\left[2\text{nd}\right] \uparrow$ and $\left[2\text{nd}\right] \downarrow$.

4. Press $\left[\text{ENTER}\right]$.

The highlighted item is inserted in the entry line.



A calculator screen showing the same history list as before. The first entry, $\cos\left(\frac{\pi}{3}\right)^2$, is now highlighted. The entry line at the bottom now contains $\sin\left(\frac{\pi}{3}\right)^2 + \cos\left(\frac{\pi}{3}\right)^2$. The status bar at the bottom shows "MAIN", "RAD AUTO", "FUNC", and "2/30".

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.

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 Voyage™ 200 Graphing Calculator'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).

$$\text{cube}(x) = x^3$$

① ② ③

$$\text{xroot}(x,y) = y^{1/x}$$

① ② ③

- ① Function name
- ② Argument list
- ③ Definition

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.



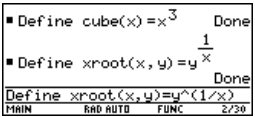
Note: Function names follow the same rules as variable names. Refer to “Storing and Recalling Variable Values” in *Operating the Calculator*.

In the argument list, be sure to use the same arguments that are used in the definition. For example, **cube(n) = x³** 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
	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 <i>Programming</i> or information on creating a user-defined function.

Creating a Multi-Statement Function

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.

Note: For information about similarities and differences between functions and programs, refer to *Programming*.

For example, suppose you want to create a function that sums a series of reciprocals based on an entered integer (**n**):

$$\frac{1}{n} + \frac{1}{n-1} + \dots + \frac{1}{1}$$

When creating the definition of a multi-statement function, it may be helpful to visualize it first in a block form.

```
❶ Func
❷ Local temp,i
   If fPart(nn)≠0 or nn≤0
❸   Return "bad argument"
   0→temp
❹ For i,nn,1,-1
   approx(temp+1/i)→temp
   EndFor
❺ Return temp
❶ EndFunc
```

- ❶ **Func** and **EndFunc** must begin and end the function.
- ❷ Variables not in the argument list must be declared as local.
- ❸ Returns a message if nn is not an integer or if nn≤0.
- ❹ Sums the reciprocals.
- ❺ Returns the sum.

When entering a multi-statement function on the calculator 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 a colon to separate each statement.

```
Define sumrecip(nn)=Func:Local temp,i: ... :EndFunc
```

Use argument names that will never be used when calling the function or program.

On the calculator Home screen:

Multi-statement functions show as

```
Define sumrecip(nn)=Func
Done
Define sumrecip(nn)=Func:
MAIN      RAD AUTO  FUNC  0/30
```

Enter a multi-statement function on one line. Be sure to include colons.















Note: It's easier to create a complicated multi-statement function in the Program Editor than on the calculator Home screen. Refer to *Programming*.

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.

```
xroot(3, 125)      5
3 + x : 125 + y : xroot(x, y)  5
3 · xroot(3, 125)  15
sumrecip(20)      sumrecip(20)
sumrecip(20)
MAIN      RAD AUTO  FUNC  7/30
```


Displaying and Editing a Function Definition

To:	Do this:
Display a list of all user-defined functions	Press  [VAR-LINK] to display the VAR-LINK screen. You may need to use the  View toolbar menu to specify the Function variable type. (Refer to <i>Memory and Variable Management</i> .) – or – Press:   [CATALOG] 
Display a list of Flash application functions	Press:   [CATALOG] 
Display the definition of a user-defined function	From the VAR-LINK screen, highlight the function and display the Contents menu.   – or – From the calculator Home screen, press  [RCL]. Type the function name but not the argument list (such as xroot), and press  twice. – or – From the Program Editor, open the function. (Refer to <i>Programming</i> .)
Edit the definition	From the calculator Home screen, use  [RCL] to display the definition. Edit the definition as necessary. Then use  or Define to save the new definition. – or – From the Program Editor, open the function, edit it, and save your changes. (Refer to <i>Programming</i> .)

Note: You can view a user-defined function in the CATALOG dialog box, but you cannot use the CATALOG to view or edit its definition.

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 Voyage™ 200 Graphing Calculator 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.



Press \leftarrow or $\boxed{2nd} \leftarrow$ to scroll left.

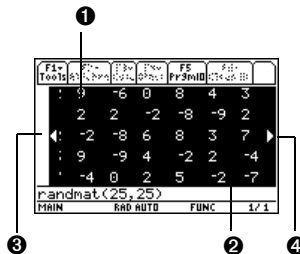
expand((x+2) ⁷)
x ⁷ + 14 · x ⁶ + 84 · x ⁵ + 280 · x ⁴
expand((x+2) ⁷)
MAIN RND AUTO FUNC 1/1

Press \rightarrow or $\boxed{2nd} \rightarrow$ to scroll right.

- 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.

- 1  Press \leftarrow or \leftarrow to scroll up
- 2  Press \leftarrow to scroll down
- 3 Press \leftarrow or \leftarrow to scroll left
- 4 Press \rightarrow or \rightarrow to scroll right



If There Is not Enough Memory

A $\ll \dots \gg$ symbol is displayed when the Voyage™ 200 does not have enough free memory to display the answer.

For example:

seq(n, n, 1, 2500) $\ll \dots \gg$			
seq(n, n, 1, 2500)			
MAIN	RAD AUTO	FUNC	1/1

Note: This example uses the seq function to generate a sequential list of integers from 1 to 2500.

When you see the $\ll \dots \gg$ 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 and/or Flash applications. Use \leftarrow [VAR-LINK] as described in *Memory and Variable Management*.

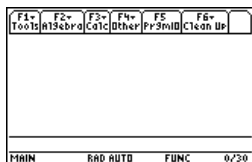
- If possible, break the problem into smaller parts that can be calculated and displayed with less memory.

Using the Custom Menu

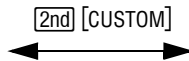
The Voyage™ 200 Graphing Calculator has a custom menu that you can turn on and off at any time. You can use the default custom menu or create your own as described in the *Programming* module.

Turning the Custom Menu On and Off

When you turn on the custom menu, it replaces the normal toolbar menu. When you turn it off, the normal menu returns. For example, from the calculator Home screen's normal toolbar menu, press **2nd** [CATALOG] to toggle the custom menu on and off.












Calculator Home screen normal toolbar menu



Custom menu

Note: You can also turn the custom menu on and off by entering **CustmOn** or **CustmOff** in the entry line and pressing **ENTER**.

Unless the menu has been modified, the default custom menu appears.

Menu	Function
 Var	Common variable names.
 f(x)	Function names such as $f(x)$, $g(x)$, and $f(x,y)$.
 Solve	Items related to solving equations.
 Unit	Common units such as $_m$, $_ft$, and $_l$.
 Symbol	Symbols such as #, ?, and ~.
International  	Commonly accented characters such as è, é, and ê.
Tool  	ClrHome , NewProb , and CustmOff .

Note: A custom menu can give you quick access to commonly used items. The *Programming* module shows you how to create custom menus for the items you use most often.

Restoring the Default Custom Menu

If a custom menu other than the default is displayed and you want to restore the default:

1. From the calculator Home screen, use **[2nd]** [CATALOG] to turn off the custom menu and display the calculator Home screen's normal toolbar menu.
2. Display the **Clean Up** toolbar menu, and select **3:Restore custom default**.



This pastes the commands used to create the default menu into the entry line.

Note: The previous custom menu is erased. If that menu was created with a program, it can be recreated later by running the program again.

3. Press **[ENTER]** to execute the commands and restore the default.

Finding the Software Version and ID Number

In some situations, you may need to find out information about your Voyage™ 200 Graphing Calculator, particularly the software version and the unit's ID number.

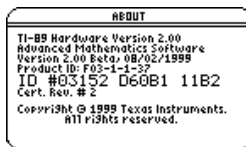
Displaying the “About” Screen

1. From either the calculator Home screen or the Apps desktop, press **[F1]** and then select **A>About**.



Your screen will be different from the one shown to the right.

2. Press **ENTER** or **ESC** to close the screen.



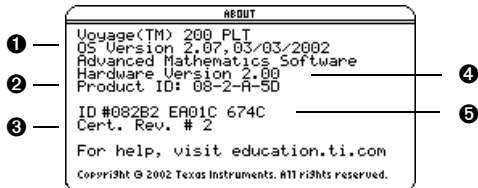
When Do You Need this Information?

The information on the About screen is intended for situations such as:

- If you obtain new or upgraded software or Flash applications for your Voyage™ 200, you may need to provide your current software version and/or the ID number of your unit.
- If you have difficulties with your Voyage™ 200 and need to contact technical support, knowing the software version may make it easier to diagnose the problem.

The About screen displays the following information about your calculator:

- Hardware version
- OS (Advanced Mathematics Software) version
- Product identifier (Product ID)
- Unit ID
- Apps certificate revision number (Cert. Rev.)



- ① OS version
- ② Product identifier
- ③ Apps certificate revision number
- ④ Hardware version
- ⑤ Unit ID (required to obtain certificates for installing purchased Apps)

Your screen will be different from the one shown above.

Symbolic Manipulation

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 Voyage™ 200 Graphing Calculator 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.

■	$2 \cdot x + x + y$	$3 \cdot x + y$
$2x+x+y$		
MAIN	RAD AUTO	FUNC 1/30

■	$5 \rightarrow x$	5
■	$2 \cdot x + x + y$	$y + 15$
$2x+x+y$		
MAIN	RAD AUTO	FUNC 2/30

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.

■	$\frac{d}{dx}(x^3)$	$3 \cdot x^2$
$d(x^3, x)$		
MAIN	RAD AUTO	FUNC 1/30

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

Note: 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.

```

■ d/dx(x^3) 75
■ x 5
x
MAIN RAD AUTO FUNC 2/30

```

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

Determining If a Variable Is Exists

Method:

Enter the variable name.

Example:

If defined, the variable's value is displayed.

```

■ x 5
■ y y
y
MAIN RAD AUTO FUNC 2/30

```

If undefined, the variable name is displayed.

Use the `isVar()` function.

If defined, "true" is displayed.

```

■ isVar(x) true
■ isVar(y) false
isVar(y)
MAIN RAD AUTO FUNC 2/30

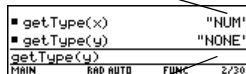
```

If undefined, "false" is displayed.

Method:**Example:**


Use the **getType** function.

If defined, the variable's type is displayed.



```
■ getType(x)          "NUM"
■ getType(y)          "NONE"
getType(y)
MAIN                END AUTO      FUNC      2/20
```

If undefined, "NONE" is displayed.

Note: Use  [VAR-LINK] to view a list of defined variables, as described in *Memory and Variable Management*.

Deleting a Defined Variable

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

To delete:

One or more specified variables

Do this:

Use the **DelVar** function.

```
■ DelVar x Done
■ DelVar x,y,test,radius Done
DelVar x,y,test,radius
MAIN RAD AUTO FUNC 2/30
```

You can also delete variables by using the VAR-LINK screen (**2nd** [VAR-LINK]) as described in *Memory and Variable Management*.

All variables of a specific type

Use the **DelType** function.

Note: The DelType function deletes all variables of the specified type in all folders.

```
■ DelType "num" Done
DelType "num"
MAIN RAD AUTO FUNC 1/30
```

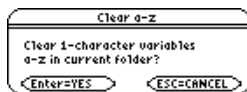
To delete:

All one-letter variables (a – z) in the current folder.

Note: For information about folders, refer to the *Calculator Home Screen* module.

Do this:

From the Home screen Clean Up menu, select **1:Clear a-z**. You will be prompted to press **ENTER** to confirm the deletion.



Temporarily Overriding a Variable

By using the “with” operator (|), you can:

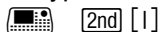
- Temporarily override a variable’s defined value.
- Temporarily define a value for an undefined variable.

27 → x	27
x ² x = 3	9
x	27
x	
MIN	RAD AUTO FUNC 3/30

De1Var x	Done
x ² x = 3	9
x	x
x	
MIN	DEGRUTO FUNC 3/30

Note: For more information about the | operator, refer to Typing the “With” Operator.

To type the “with” operator (|), press:



Using Exact, Approximate, and Auto Modes

The Exact/Approx mode settings, which are described briefly in *Operating the Handheld*, directly affect the precision and accuracy with which the Voyage™ 200 Graphing Calculator calculates a result. This section describes these mode settings as they relate to symbolic manipulation.

EXACT Setting

When Exact/Approx = EXACT, the handheld 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**, \int , **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 handheld.

- 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.	<p>As you use more complicated rational numbers and irrational constants, calculations can:</p> <ul style="list-style-type: none"> • 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 handheld 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: $d(\mathbf{y}(\mathbf{x}), \mathbf{x}, 2.0)$ transforms to $d(\mathbf{y}(\mathbf{x}), \mathbf{x}, 2)$.
- 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 \int (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. Approximate results are sometimes more compact and comprehensible than exact results.	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 .
If you do not plan to use symbolic computations, approximate results are similar to familiar, traditional numeric calculators.	Symbolic operations such as limits and integration are less likely to give satisfying results in the APPROXIMATE setting. 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 handheld 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 $.1666666666667$

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 $x/6 + .1666666666667 y$

and

$\{1/2 - 1/3, 0.5 - 1/3\}$ transforms to $\{1/6, .1666666666667\}$

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

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.

Disadvantages

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 Voyage™ 200 Graphing Calculator 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.

■ 5 → num	5
■ 7 · num	35
7 * num	
MAIN	RBD AUTO FUNC 3/30

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

■ a → num	a
■ 5 → a	5
■ 7 · num	35
7 * num	
MAIN	RBD AUTO FUNC 3/30

Default simplification does not modify variables that use path names to indicate a folder. For example, $x+classx$ does not simplify to $2x$.

Note: For information about folders, refer to the *Calculator Home Screen* module.

- 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.

■	$2 \cdot y \cdot 3$	$6 \cdot y$
■	$y \cdot x \cdot 3 + x^2 + 1$	$x^2 + 3 \cdot x \cdot y + 1$
<hr/>		
	$y * x * 3 + x^2 + 1$	
MAIN	RAD AUTO	FUNC 2/30

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.

■	$x^2 \cdot x \cdot y$	$x^3 \cdot y$
■	$3 \cdot x + x + 7$	$4 \cdot x + 7$
<hr/>		
	$3x + x + 7$	
MAIN	RAD AUTO	FUNC 2/30

- Identities involving zeros and ones are exploited.

■	$x + 0.$	x
■	$1 \cdot x$	x
■	$1 \cdot x$	x
■	$x \cdot 1$	x
■	$x \cdot 1.$	x
<hr/>		
	$x^1.$	
MAIN	RAD AUTO	FUNC 6/30

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).

■	1^x	1
■	$(1.)^x$	$1.$
■	x^0	1
■	$x \cdot 0.$	1
<hr/>		
	$x^0.$	
MAIN	RAD AUTO	FUNC 4/30

- Polynomial greatest common divisors are canceled.

$\frac{x^2 + 5x + 6}{x + 2} \quad x + 3$
$\frac{(x^2 + 5x + 6)}{(x + 2)(x + 3)}$
MAIN RAD AUTO FUNC 1/30

- Polynomials are expanded unless no key cancellation can occur.

$\frac{(x + 1)^2 - x^2}{(x + 2)^2 \cdot (x + 1)} \quad 2 \cdot x + 1$
$\frac{(x + 1)^2 - x^2}{(x + 2)^2 \cdot (x + 1)} \cdot \frac{(x + 1) \cdot (x + 2)^2}{(x + 1) \cdot (x + 2)^2}$
MAIN RAD AUTO FUNC 2/30

No key cancellation

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

$\frac{2 \cdot x}{x^2 - 1} - \frac{1}{x - 1} \quad \frac{1}{x + 1}$
$\frac{1}{x} + \frac{1}{y} \quad \frac{1}{x} + \frac{1}{y}$
$\frac{1}{x} + \frac{1}{y}$
MAIN RAD AUTO FUNC 2/30

No key cancellation

- Functional identities are exploited. For example:

$$\ln(2x) = \ln(2) + \ln(x)$$

and

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

$\ln(2 \cdot x) - \ln(x) \quad \ln(2)$
$y \cdot (\sin(x))^2 + y \cdot (\cos(x))^2 \quad y$
$y \cdot \sin(x)^2 + y \cdot \cos(x)^2$
MAIN RAD AUTO FUNC 2/30

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.

For example:

x cannot be simplified.

De1Var	x	Done
	$\frac{d}{dx}(x^3)$	$3 \cdot x^2$
$\frac{d}{dx}(x^3, x)$		
MAIN	RAD AUTO	FUNC 2/20

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

5 → x	5
	$\frac{d}{dx}(x^3)$
$\frac{d}{dx}(x^3, x)$	
MAIN	RAD AUTO FUNC 2/20

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.

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

De1Var	t	Done
	$t \rightarrow x$	t
	$\frac{d}{dx}(x^3)$	$3 \cdot t^2$
$\frac{d}{dx}(x^3, x)$		
MAIN	RAD AUTO	FUNC 3/20

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

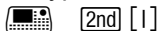
5 → t	5
	$t \rightarrow y$
	$\frac{d}{dx}(x^3)$
$\frac{d}{dx}(x^3, x)$	
MAIN	RAD AUTO FUNC 3/20

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 (|), press:



Substituting for a Variable

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

$(x + 2)^2 x = 1$	9
$\pi \cdot r^2 r = 5$	$25 \cdot \pi$
$\frac{d}{dx}(x^3) x = 5$	75
$\frac{d}{dx}(x^3, x) x = 5$	
MAIN	RAD AUTO FUNC 3/20

First derivative of x^3
at $x = 5$

$(x + 2)^2 x = a + 1$	$(a + 3)^2$
$(x + 2)^2 x = a + 1$	
MAIN	RAD AUTO FUNC 1/20

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

$(x^2 + y^2)^{1/2} x = 3 \text{ and } y = 4$	5
$(x^2 + y^2)^{1/2} x = 3 \text{ and } y = 4$	
MAIN	RAD AUTO FUNC 1/20

Substituting for a Simple Expression

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

The calculator screen shows the expression $(\sin(x))^3 + 2 \cdot \sin(x) + 1$ being edited. The variable s is substituted for $\sin(x)$, resulting in $s^3 + 2 \cdot s + 1$. The screen also shows the variable s being defined as $\sin(x)$.

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.

The calculator screen shows the expression $a \cdot \cos(x) + (\cos(x))^2$ being edited. The variable c is substituted for $\cos(x)$, resulting in $c^2 + 2 \cdot c$. The screen also shows the variable c being defined as $\cos(x)$.

Note: $\text{acos}(x)$ is different from $a \cdot \cos(x)$.

Substituting Complex Values

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

The calculator screen shows the expression $|x|$ being edited. The complex value $a + bi$ is substituted for x , resulting in $\sqrt{a^2 + b^2}$. The screen also shows the variable x being defined as $2 + 3i$.

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

$$x + yi \rightarrow z$$

Then you can use z as a complex variable. You can also use $z_$. For more information see the $z_$ (underscore) topic in the *Technical Reference* module.

Note:

- For an overview of complex numbers, refer to the *Technical Reference* module.
- To get the complex i , press $\boxed{2\text{nd}}$ $[i]$. Do not simply type the letter i on the keyboard.

Be Aware of the Limitations of Substitutions

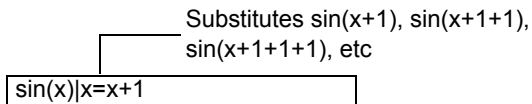
- Substitution occurs only where there is an exact match for the substitution.

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

$x^4 + 3 \cdot x^2 \mid x^2 = y$	$x^4 + 3 \cdot y$
$x^4 + 3 \cdot x^2 \mid x = y^{1/2}$	$y^2 + 3 \cdot y$
$x^4 + 3 \cdot x^2 \mid x = y^{(1/2)}$	
MIN	RAD AUTO FUNC 2/30

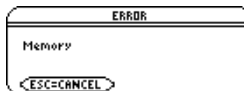
Define the substitution in simpler terms for a more complete substitution.

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

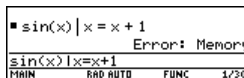


When you enter a substitution that causes an infinite recursion:

- An error message is displayed.

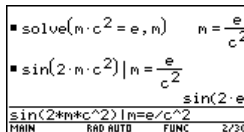


- When you press $\boxed{\text{ESC}}$, an error is shown in the history area.



- 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.

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



- Substituting for more general expressions (either $m \cdot c^2 = e$ or $c^2 \cdot m = e$) may not work as you anticipate.

No match for substitution

$\sin(2 \cdot m \cdot c^2) m \cdot c^2 = e$			
		$\sin(2 \cdot c^2 \cdot m)$	
$\sin(2 * m * c^2) m * c^2 = e$			
MAIN	RAD AUTO	FUNC	1/30

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

Specifying Domain Constraints

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

$$\ln(x \cdot y) = \ln(x) + \ln(y)$$

only if x and/or y is not negative

$$\sin^{-1}(\sin(\theta)) = \theta$$

only if $\theta \geq -\pi/2$ and $\leq \pi/2$ radians

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

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

$\ln(x \cdot y) - \ln(x)$			
		$\ln(x \cdot y) - \ln(x)$	
$\ln(x \cdot y) - \ln(x) x > 0$			
$\ln(x \cdot y) - \ln(x) x > 0$			
MAIN	RAD AUTO	FUNC	2/30

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

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

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

$\sin^4(\sin(\theta))$	$\sin^4(\sin(\theta))$
$\sin^4(\sin(\theta)) \mid \theta \geq -\frac{\pi}{2} \text{ and } \theta \leq \frac{\pi}{2}$	θ
$\sin(\theta) \mid \theta \geq -\pi/2 \text{ and } \theta \leq \pi/2$	
MAIN	RAD AUTO FUNC 2/30

With a constraint, the expression can be simplified.

Note: For \geq or \leq , press \blacklozenge [$>$] or \blacklozenge [$<$]. You can also use $\boxed{2\text{nd}}$ [MATH] **8** or $\boxed{2\text{nd}}$ [CHAR] **2** to select them from a menu.

Using Substitutions vs. Defining a Variable

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

$(x+2)^2 \mid x=1$	9
$1 \rightarrow x$	1
$(x+2)^2$	9
$(x+2)^2$	
MAIN	RAD AUTO FUNC 3/30

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.

$\text{DelVar } x$	Done
$(x+2)^2 \mid x=1$	9
$\frac{x^2+2x+1}{x^2-1}$	$\frac{x+1}{x-1}$
$(x^2+2x+1)/(x^2-1)$	
MAIN	RAD AUTO FUNC 3/30

Storing $1 \rightarrow x$ affects the subsequent calculations. calculation.

■ $1 \rightarrow x$	1
■ $(x + 2)^2$	9
■ $\frac{x^2 + 2 \cdot x + 1}{x^2 - 1}$	undef
$(x^2 + 2x + 1) / (x^2 - 1)$	
MIN	RAD AUTO FUNC 3/30

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

Overview of the Algebra Menu

You can use the $\boxed{F2}$ **Algebra** toolbar menu to select the most commonly used algebraic functions.

The Algebra Menu

From the Home screen, press $\boxed{F2}$ to display:



This menu is also available from the **MATH** menu. Press $\boxed{2nd}$ $\boxed{[MATH]}$ and then select **9:Algebra**.

Note: For a complete description of each function and its syntax, refer to the *Technical Reference* module.

Menu Item	Description
solve	Solves an equation for a specified variable. This returns real solutions only, regardless of the Complex Format mode setting. Displays answers with " <i>and</i> " and " <i>or</i> " connecting solutions. (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. Displays in a list.
approx	Evaluates an expression using floating-point arithmetic, where possible. This is equivalent to using MODE to set Exact/Approx = APPROXIMATE (or using MODE 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: <div data-bbox="253 153 422 194" style="border: 1px solid black; padding: 2px;"> <pre>1:tExpand(2:tCollect(</pre> </div>
	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: <div data-bbox="253 463 409 517" style="border: 1px solid black; padding: 2px;"> <pre>1:cSolve(2:cFactor(3:cZeros(</pre> </div> <p>These are the same as solve, factor, and zeros; but they also compute complex results.</p>
Extract	Displays the submenu: <div data-bbox="253 700 422 772" style="border: 1px solid black; padding: 2px;"> <pre>1:getNum(2:getDenom(3:left(4:right(</pre> </div> <p>getNum — Applies comDenom and then returns the resulting numerator.</p> <p>getDenom — Applies comDenom and then returns the resulting denominator.</p> <p>left — Returns the left-hand side of an equation or inequality.</p>

Menu Item	Description
	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 $\boxed{F2}$ **Algebra** toolbar menu. For complete information about any function, refer to the *Technical Reference* module. 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.

■	$x + 3 + x + 2$	$2 \cdot x + 5$
	$(x+3)+(x+2)$	
MAIN	RAD AUTO	FUNC 1/30

■	$\frac{x^2 + 5 \cdot x + 6}{x + 2}$	$x + 3$
	$(x^2+5x+6)/(x+2)$	
MAIN	RAD AUTO	FUNC 1/30

Factoring and Expanding Polynomials

Use the **factor** ($\boxed{F2}$ 2) and **expand** ($\boxed{F2}$ 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.

```
■ factor(x5 - 1)
  (x - 1) · (x4 + x3 + x2 + x + 1)
■ expand((x - 1) · (x4 + x3 + x2 + x + 1))
  x5 - 1
expand(ans(1))
MAIN      RAD AUTO      FUNC      2/30
```

Finding Prime Factors of a Number

The **factor** ($\boxed{F2}$ 2) function lets you do more than simply factor an algebraic polynomial.

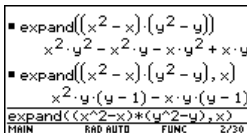
You can find prime factors of a rational number (either an integer or a ratio of integers).

```
■ factor(1729)      7 · 13 · 19
■ factor( $\frac{21475}{1548}$ )   $\frac{5^2 \cdot 859}{2^2 \cdot 3^2 \cdot 43}$ 
factor(21475/1548)
MAIN      RAD AUTO      FUNC      2/30
```

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.



```
■ expand((x^2-x).(y^2-y))
x^2.y^2-x^2.y-x.y^2+x.y
■ expand((x^2-x).(y^2-y),x)
x^2.y.(y-1)-x.y.(y-1)
expand((x^2-x)*(y^2-y),x)
MAIN RAD AUTO FUNC 2/20
```

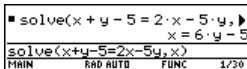
Then do a partial expansion with respect to x.

Solving an Equation

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

solve(*equation*, *var*)

Solve $x + y - 5 = 2x - 5y$ for x.



```
■ solve(x+y-5=2x-5y,▶)
x=6-y-5
solve(x+y-5=2x-5y,x)
MAIN RAD AUTO FUNC 1/20
```

Notice that solve displays only the final result.

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

$$x + y - 5 \quad \square \quad 2x - 5y$$

$$\square \quad 2x$$

$$\square \quad y$$

$$+ \quad 5$$

$$\times \quad (-) \quad 1$$

$x + y - 5 = 2 \cdot x - 5 \cdot y$ $x + y - 5 = 2 \cdot x - 5 \cdot y$
$(x + y - 5 = 2 \cdot x - 5 \cdot y) - 2 \cdot x$ $-x + y - 5 = -5 \cdot y$
$(-x + y - 5 = -5 \cdot y) - y$ $-x - 5 = -6 \cdot y$
$(-x - 5 = -6 \cdot y) + 5$ $-x = 5 - 6 \cdot y$
$(-x = 5 - 6 \cdot y) \cdot -1$ $x = 6 \cdot y - 5$
ans<1>*1
MIN RAD AUTO FUNC 5/30

Note: An operation such as $\square 2 \times$ subtracts $2x$ from both sides.

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.

Method	Example
Use the solve function for a one-step solution.	solve ($2x-3y=4$ and $-x+7y=-12$, $\{x,y\}$)
Use the solve function with substitution () for step-by-step manipulation.	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. See "Symbolic Manipulation" in the <i>Previews</i> chapter, which solved for $x = -8/11$ and $y = -20/11$.

Method**Example**

Use the **simult** function with a matrix.

Enter the coefficients as a matrix and the results as a constant column matrix.

$$\text{simult}\left(\begin{bmatrix} 2 & -3 \\ -1 & 7 \end{bmatrix}, \begin{bmatrix} 4 \\ -12 \end{bmatrix}\right)$$

$$\begin{bmatrix} -8/11 \\ -20/11 \end{bmatrix}$$

$$\text{ult}(2, -3; -1, 7), [4; -12]$$

MAIN RAD AUTO FUNC 1/30

Use the **rref** function with a matrix.

Enter the coefficients as an augmented matrix.

$$\text{rref}\left(\begin{bmatrix} 2 & -3 & 4 \\ -1 & 7 & -12 \end{bmatrix}\right)$$

$$\begin{bmatrix} 1 & 0 & -8/11 \\ 0 & 1 & -20/11 \end{bmatrix}$$

$$\text{rref}(2, -3, 4; -1, 7, -12)$$

MAIN RAD AUTO FUNC 1/30

Note: The **simult** and **rref** matrix functions are not on the $\boxed{F2}$ **Algebra** menu. Use $\boxed{2nd}$ $\boxed{[MATH]}$ **4** or the **Catalog**.

Finding the Zeros of an Expression

Use the **zeros** ($\boxed{F2}$ 4) function.

zeros(*expression*, *var*)

Use the expression x

sin(x) + cos(x).

Find the zeros with respect to x in the interval $0 \leq x$ and $x \leq 3$.

Note: For \geq or \leq , type $\boxed{\blacklozenge}$ [$>$] or $\boxed{\blacklozenge}$ [$<$]. You can also use $\boxed{2nd}$ [MATH] **8** or $\boxed{2nd}$ [CHAR] **2** to select them from a menu.

■ zeros(x·sin(x)+cos(x),x)▶			
{2.79839}			
n(x)+cos(x),x)105x and x			
MIN	RAD AUTO	FUNC	1/30

Use the “*with*” operator 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

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.

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

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

■ propFrac $\left(\frac{x^4 - 2 \cdot x^2 + x}{2 \cdot x^2 + x + 4} \right)$
 $\frac{31 \cdot x + 60}{8 \cdot (2 \cdot x^2 + x + 4)} + \frac{x^2}{2} - \frac{x}{4} \rightarrow$
■ comDenom $\left(\frac{31 \cdot x + 60}{8 \cdot (2 \cdot x^2 + x + 4)} \right) \rightarrow$
 $\frac{x^4 - 2 \cdot x^2 + x}{2 \cdot x^2 + x + 4}$
+x+4)>+(x^2/2-x/4)-15/8)
MIN RAD AUTO FUNC 2/30

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

In this example:

- $\frac{31x+60}{8}$ 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 the *Technical Reference* module.

Menu Item	Description
d differentiate	Differentiates an expression with respect to a specified variable.

\int 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).
deSolve	Symbolically solves many 1st and 2nd order differential equations, with or without initial conditions.
impDif	Computes implicit derivatives for equations in two variables in which one variable is defined implicitly in terms of another.

Note: The d symbol for differentiate is a special symbol. It is not the same as typing the letter **D** on the keyboard. Use $\boxed{\text{F3}}$ **1** or $\boxed{\text{2nd}}$ [d].


Common Calculus Operations

This section gives examples for some of the functions available from the $\boxed{\text{F3}}$ **Calc** toolbar menu. For complete information about any calculus function, refer to the *Technical Reference* module.

Integrating and Differentiating

Use the \int **integrate** ($\boxed{\text{F3}}$ **2**) and d **differentiate** ($\boxed{\text{F3}}$ **1**) functions.

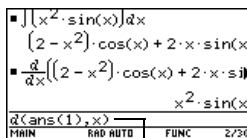
$$\int (\text{expression}, \text{var} [\text{,low}] [\text{,up}])$$

 lets you specify limits or a constant of integration

$$d (\text{expression}, \text{var} [\text{,order}])$$

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

Differentiate the answer with respect to x .



$\int (x^2 \cdot \sin(x)) dx$
 $(2 - x^2) \cdot \cos(x) + 2 \cdot x \cdot \sin(x)$
 $\frac{d}{dx} ((2 - x^2) \cdot \cos(x) + 2 \cdot x \cdot \sin(x))$
 $x^2 \cdot \sin(x)$

To get d , use $\boxed{\text{F3}}$ **1** or $\boxed{\text{2nd}}$ [d]. Do not simply type the letter **D** on the keyboard.

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

Finding a Limit

Use the **limit** ($\boxed{F3}$ **3**) function.

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

└ negative number = from left
positive number = from right
omitted number or 0 = both

Find the limit of $\sin(3x) / x$ as x approaches 0.

$\lim_{x \rightarrow 0} \left(\frac{\sin(3 \cdot x)}{x} \right)$	3
$\text{limit}(\sin(3x)/x, x, 0)$	
MAIN	BAD AUTO FUNC 1/30

Note: You can find a limit for an expression, list, or matrix.

Finding a Taylor Polynomial

Use the **taylor** ($\boxed{\text{F3}}$ 9) function.

taylor(*expression*, *var*, *order* [*point*])

└ if omitted, expansion point is 0

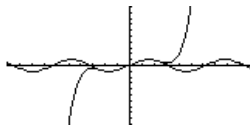
Find a 6th order Taylor polynomial for **sin(x)** with respect to x .

Store the answer as a user-defined function named **y1(x)**.

Then graph **sin(x)** and the Taylor polynomial.

Calculator screen showing the Taylor polynomial of $\sin(x)$ of order 6. The input is `taylor(sin(x), x, 6)`. The result is $\frac{x^5}{120} - \frac{x^3}{6} + x$. The screen also shows the user-defined function `y1(x)` and the answer `ans(1)→y1(x)`.

Graph $\sin(x)$: Graph $y_1(x)$



Important: Degree-mode scaling by $\pi/180$ may cause calculus application results to appear in a different form.

User-Defined Functions and Symbolic Manipulation

You can use a user-defined function as an argument for the Voyage™ 200 's built-in algebra and calculus functions.

For Information about Creating a User-Defined Function

Refer to:

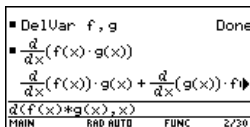
- “Creating and Evaluating User-Defined Functions” in the *Calculator Home Screen* module.
- “Graphing a Function Defined on the Home Screen” and “Graphing a Piecewise Defined Function” in the *Calculator Home Screen* module.
- “Overview of Entering a Function” in the *Programming* module.

Undefined Functions

You can use functions such as $f(x)$, $g(t)$, $r(\theta)$, 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 .



Note: To select **d** from the Calc toolbar menu, press $\boxed{F3}$ 1 (or press $\boxed{2nd}$ \boxed{d} on the keyboard).

Single-Statement Functions

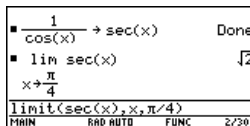
You can use user-defined functions consisting of a single expression. For example:

- Use **STO►** to create a user-defined secant function, where:

$$\sec x = \frac{1}{\cos x}$$

Then find the limit of **sec(x)** as x approaches $\pi/4$.

Note: To select **limit** from the Calc toolbar menu, press **F3 3**.



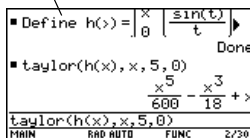
- Use **Define** to create a user-defined function **h(x)**, where:

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

Then find a 5th order Taylor polynomial for **h(x)** with respect to x .

Note: To select \int from the **Calc** toolbar menu, press **F3 2** (or press **2nd** **[f]** on the keyboard). To select Taylor, press **F3 9**.

Define $h(x) = \int(\sin(t)/t, 0, x)$.



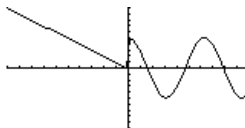
Multi-Statement vs. Single-Statement Functions

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:**

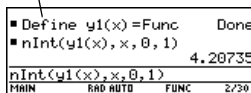
$x < 0$ $-x$
 $x \geq 0$ $5 \cos(x)$



- If you were to 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
```



Then numerically integrate $y_1(x)$ with respect to x .

Note: To select **nInt** from the Calc toolbar menu, press **[F3] B:nInt**.

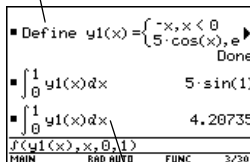
- Create an equivalent single-statement user-defined function.

Use the Voyage™ 200 Graphing Calculator's built-in **when** function.

Then integrate $y_1(x)$ with respect to x .

Note: To select \int from the Calc toolbar menu, press $\boxed{\text{F3}}$ $\mathbf{2}$ (or press $\boxed{2\text{nd}}$ $\boxed{[f]}$ on the keyboard).

Define
 $y_1(x) = \text{when}(x < 0,$
 $-x, 5\cos(x))$



Press $\boxed{\blacktriangle}$ $\boxed{\text{ENTER}}$
 for a floating-point
 result.

If You Get an Out-of-Memory Error

The Voyage™ 200 Graphing Calculator stores intermediate results in memory and then deletes them when the calculation is complete. Depending on the complexity of the calculation, the handheld may run out of memory before a result can be calculated.

Freeing Up Memory

- Delete unneeded variables and/or Flash applications, particularly large-sized ones.
 - Use $\boxed{2\text{nd}}$ $\boxed{[\text{VAR-LINK}]}$ as described in *Memory and Variable Management* to view and delete variables and/or Flash applications.
- 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*c²**, substitute e for **m*c²**.
 - 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 are $\frac{x}{d} + \frac{y}{d}$. 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 .

■ solve(x = x, x)	true
■ 5 > x : x < 3	false
5 > x : x < 3	
MAIN	RAD AUTO FUNC 2/30

$5 < 3$ is false.

@n1 ... @n255

This notation indicates an “arbitrary integer” that represents any integer.

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. Use Clean Up 2:**NewProb** to reset to @n1.

Note: For @, press:



2nd R

A solution is at every integer multiple of π .

■ solve(sin(x) = 0, x)	$x = n_1 \cdot \pi$
■ solve(sin(x) = 1, x)	$x = 2 \cdot n_2 \cdot \pi + \frac{\pi}{2}$
solve(sin(x) = 1, x)	
MAIN	RAD AUTO FUNC 2/30

Both @n and @n2 represent any arbitrary integer, but this notation identifies separate arbitrary integers.

@1 ... @255

This notation indicates an “arbitrary constant” that represents any integer.

When an arbitrary constant occurs multiple times in the same session, each occurrence is numbered consecutively. After it reaches 255, arbitrary integer consecutive numbering restarts at @0. Use Clean Up 2:**NewProb** to reset to @1.

Note: For @, press:



2nd **R**

∞ , e

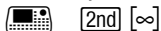
∞ represents infinity, and e represents the constant **2.71828...** (base of the natural logarithms).

■ $\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$ e
limit((1+1/n)^n,n,∞)
MAIN RAD AUTO FUNC 1/30

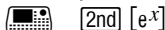
These constants are often used in entries as well as results.

Notes:

For ∞ , press:



For e , press:



undef

This indicates that the result is undefined.

Mathematically undefined

$\pm\infty$ (undetermined sign)

Non-unique limit

■ $\frac{0}{0}$ undef
■ $\frac{1}{0}$ undef
■ $\lim_{x \rightarrow -\infty} \sin(x)$ undef
limit(sin(x),x,-∞)
MAIN RAD AUTO FUNC 3/30

Constants and Measurement Units

Entering Constants or Units

You can use a menu to select from a list of available constants and units, or you can type them directly from the keyboard.

From a Menu

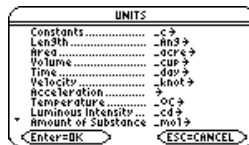
The following shows how to select a unit, but you can use the same general procedure to select a constant.

From the Home screen:

1. Type the value or expression.
2. Display the **UNITS** dialog box. Press:



6.3



3. Use \ominus and $\omin�$ to move the cursor to the applicable category.

Note: Use $\boxed{2nd}$ \ominus and $\boxed{2nd}$ $\omin�$ to scroll one page at a time through the categories.

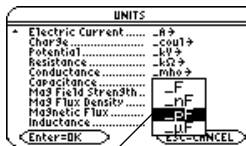


4. To select the highlighted (default) unit, press **[ENTER]**.

– or –

To select a different unit from the category, press **[↓]**. Then highlight the applicable unit, and press **[ENTER]**.

Note: If you created a user-defined unit for an existing category, it is listed in the menu.



You can also move the cursor by typing the first letter of a unit.

The selected unit is placed in the entry line. Constant and unit names always begin with an underscore (_).

6.3_μF

From the Keyboard

If you know the abbreviation that the Voyage™ 200 uses for a particular constant or unit, you can type it directly from the keyboard. For example:

256_m

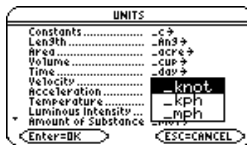
- The first character must be an underscore (_). For _, press: **[2nd] [-]**
- A space or a multiplication symbol (*) before the underscore is optional. For example, **256_m**, **256 _m**, and **256*_m** are equivalent.
- However, if you are adding units to a variable, you must put a space or * before the underscore. For example, **x_m** is treated as a variable, not as x with a unit.

Note: You can type units in either uppercase or lowercase characters.

Combining Multiple Units

You may need to combine two or more units from different categories.

For example, suppose you want to enter a velocity in meters per second. In the UNITS dialog box, however, the **Velocity** category does not contain this unit.



You can enter meters per second by combining **_m** and **_s** from the **Length** and **Time** categories, respectively.

3*9.8 _m/_s

Combine the units **_m** and **_s**. There is no pre-defined **m/_s** unit.

Note: Create a user-defined unit for frequently used combinations.

Using Parentheses with Units in a Calculation

In a calculation, you may need to use parentheses () to group a value and its units so that they are evaluated properly. This is particularly true for division problems. For example:

To calculate:	Enter:
$\frac{100 \text{ m}}{2 \text{ s}}$	<div style="border: 1px solid black; padding: 5px; display: inline-block;">$100_m/(2_s) \quad 50 \cdot \frac{\text{m}}{\text{s}}$</div> <p>You must use parentheses for (2_s). This is important for division.</p> <p>If you omit the parentheses, you will get unexpected units. For example:</p> <div style="border: 1px solid black; padding: 5px; display: inline-block;">$100_m/2_s \quad 50. \text{ m } \text{ s}$</div>

Note: If you have any doubt about how a value and its units will be evaluated, group them within parentheses ().

Here's why you get unexpected units if you do not use parentheses. In a calculation, a unit is treated similar to a variable. For example: **100_m** is treated as **100*_m** and **2_s** is treated as **2*_s**. Without parentheses, the entry is calculated as:

$$100_m / 2_s = \frac{100 \cdot \text{m}}{2} \text{ s} = 50. \text{ m } \text{ s}.$$

Converting from One Unit to Another

You can convert from one unit to another in the same category, including any user-defined units.

For All Units Except Temperature

If you use a unit in a calculation, it is converted and displayed automatically in the current default unit for that category, unless you use the ► conversion operator as described later. The following examples assume that your default units are set to the SI system of metric units.

Notes:

- Refer to the list of pre-defined units.
- From the UNITS dialog box, you can select available units from a menu.

To multiply 20 times 6 kilometers.

20*6_km

■ 20 * 6 _ km	120000. _ m
20*6_km	
MIN	RAD AUTO FUNC 1/20

Shown in the default unit for Length, (_m in SI system).

If you want to convert to a unit other than the default, use the ► conversion operator.

expression_unit1 ► _unit2

└─ For ►, press [2nd] [►].

To convert 4 light years to kilometers:

4_ltyr ► _km

To convert 186000 miles/second to kilometers/hour:

186000_mi/_s ► _km/_hr

4_ltyr ► _km	3.78421E13 _km		
186000 _mi/_s ► _km/_hr	1.07762E9 _km/_hr		
186000_mi/_s ► _km/_hr			
MMIN	RAD AUTO	FUNC	2/20

If an expression uses a combination of units, you can specify a conversion for some of the units only. Any units for which you do not specify a conversion will be displayed according to your defaults.

To convert 186000 miles/second from miles to kilometers:

$$186000_mi/_s \blacktriangleright _km$$

To convert 186000 miles/second from seconds to hours:

$$186000_mi/_s \blacktriangleright 1/_hr$$

Because a Time conversion is not specified, it is shown in its default unit ($_s$ in this example).

		299338.	$\frac{_m}{_s}$
■	$186000 \frac{_m}{_s}$	$\blacktriangleright \frac{1}{_hr}$	
		1.07762E12	$\frac{_m}{_hr}$
<hr/>			
186000_mi/_s		$\blacktriangleright 1/_hr$	
MAIN	RAD AUTO	FUNC	2/30

Because a Length conversion is not specified, it is shown in its default unit ($_m$ in this example).

To enter meters per second squared:

$$27_m/_s^2$$

To convert meters per second squared from seconds to hours:

$$27_m/_s^2 \blacktriangleright 1/_hr^2$$

		27.	$\frac{_m}{_s^2}$
■	$27 \frac{_m}{_s^2}$		$\frac{_m}{_s^2}$
<hr/>			
27_m/_s^2			
MAIN	RAD AUTO	FUNC	1/30

			$\frac{1}{_hr^2}$
■	$27 \frac{_m}{_s^2}$	$\blacktriangleright \frac{1}{_hr^2}$	
		3.4992E8	$\frac{_m}{_hr^2}$
<hr/>			
27_m/_s^2		$\blacktriangleright 1/_hr^2$	
MAIN	RAD AUTO	FUNC	2/30

For Temperature Values

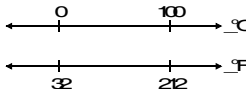
To convert a temperature value, you must use **tmpCnv()** instead of the ► operator.

tmpCnv(expression °tempUnit1, °tempUnit2)
└─ For °, press **2nd** [°]

For example, to convert 100 °C to °F:

tmpCnv(100 °c, °f)

■ tmpCnv(100 °C, °F)	212. °F
tmpCnv(100 °c, °f)	
MAIN	RAD AUTO FUNC 1/30



For Temperature Ranges

To convert a temperature range (the difference between two temperature values), use $\Delta\text{tmpCnv}()$.

$\Delta\text{tmpCnv}(\text{expression_}\text{tempUnit1}, \text{_}\text{tempUnit2})$

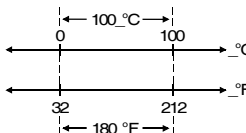
For example, to convert a 100_°C range to its equivalent range in _°F:

$\Delta\text{tmpCnv}(100_c, _f)$

Note: For Δ , press:

2nd **G** **↑** **D**

■ $\Delta\text{tmpCnv}(100_c, _f)$			
100. °C			
180. °F			
MAIN	RAD AUTO	FUNC	1/30



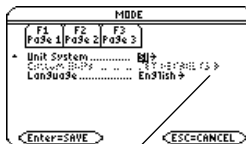
Setting the Default Units for Displayed Results

All results involving units are displayed in the default unit for that category. For example, if the default unit for **Length** is **_m**, any length result is displayed in meters (even if you entered **_km** or **_ft** in the calculation).

If You're Using the SI or ENG/US System

The SI and ENG/US systems of measurement (set from **Page 3** of the MODE screen) use built-in default units, which you cannot change.

The default units for these systems are available.

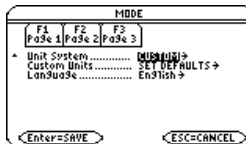


If Unit System=SI or ENG/US, the Custom Units item is dimmed. You cannot set a default for individual categories.

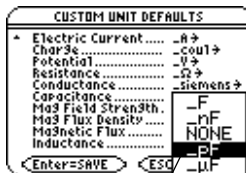
Setting Custom Defaults

To set custom defaults:

1. Press **MODE** **F3** **3** to set **Unit System = CUSTOM**.
2. Press **↵** to highlight **SET DEFAULTS**.
3. Press **↵** to display the **CUSTOM UNIT DEFAULTS** dialog box.



- For each category, you can highlight its default, press **⏎**, and select a unit from the list.
- Press **ENTER** twice to save your changes and exit the **MODE** screen.



You can also move the cursor by typing the first letter of a unit.

Notes:

- You can also use **setUnits()** or **getUnits()** to set or return information about default units. Refer to the *Technical Reference* module.
- When the **CUSTOM UNIT DEFAULTS** dialog box first appears, it shows the current default units.

What is the NONE Default?

Many categories let you select NONE as the default unit.

This means that results in that category are displayed in the default units of its components.



For example, **Area = Length²**, so **Length** is the component of **Area**.

- If the defaults are **Area = _acre** and **Length = _m** (meters), area results are shown with **_acre** units.
- If you set **Area = NONE**, area results are shown with **_m²** units.

Note: NONE is not available for base categories such as **Length** and **Mass** that have no components.

Creating Your Own User-Defined Units

In any category, you can expand the list of available units by defining a new unit in terms of one or more pre-defined units. You can also use “standalone” units.

Why Use Your Own Units?

Some example reasons to create a unit are:

- You want to enter length values in dekameters. Define *10_m* as a new unit named *_dm*.
- Instead of entering *_m/_s²* as an acceleration unit, you define that combination of units as a single unit named *_ms2*.
- You want to calculate how many times someone blinks. You can use *_blinks* as a valid unit without defining it. This “standalone” unit is treated similar to a variable that is not defined. For instance, *3_blinks* is treated the same as *3a*.

Note: If you create a user-defined unit for an existing category, you can select it from the UNITS dialog box menu. But you cannot use **MODE** to select the unit as a default for displayed results.

Rules for User-Defined Unit Names

The naming rules for units are similar to variables.

- Can have up to 8 characters.
- First character must be an underscore. For `_`, press:
`[2nd] [-]`
- Second character can be any valid variable name character except `_` or a digit. For example, `_9f` is not valid.
- Remaining characters (up to 6) can be any valid variable name character except an underscore.

Defining a Unit

Define a unit the same way you store to a variable.

definition \rightarrow `_newUnit`

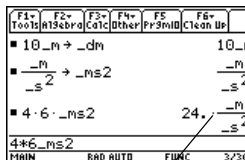
└─ For \rightarrow , press `[STO]`

For example, to define a dekameter unit:

`10_m \rightarrow _dm`

To define an acceleration unit:

`_m/_s^2 \rightarrow _ms2`



F1-	F2-	F3-	F4-	F5-	F6-
Tools	13	Calc	Other	Pr	Clean Up
10_m \rightarrow _dm					10_m
_m/_s^2 \rightarrow _ms2					_m/_s^2
4*6*_ms2				24*_m/_s^2	_m/_s^2
4*6*_ms2					_m/_s^2
MMIN	RAD AUTO	FUNC			3/30

Assuming unit defaults for Length and Time are set to `_m` and `_s`.

To calculate 195 blinks in 5 minutes as
_blinks/_min:

195_blinks/(5_min)

■ 195_blinks	
5_min	
.65_blinks	
_s	
195_blinks/(5_min)	
MMIN	RAD AUTO FUNC 1/20

Assuming unit
default for Time is
set to _s.

Notes:

- User-defined units are displayed in lowercase characters, regardless of the case you use to define them.
- User-defined units such as dm are stored as variables. You can delete them the same as you would any variable.

List of Pre-Defined Constants and Units

This section lists the pre-defined constants and units by category. You can select any of these from the UNITS dialog box. If you use **MODE** to set default units, note that categories with only one defined unit are not listed.

Defaults for SI and ENG/US

The SI and ENG/US systems of measurement use built-in default units. In this section, the built-in defaults are indicated by (SI) and (ENG/US). In some categories, both systems use the same default.

Some categories do not have default units.

Constants

	Description	Value
_c	speed of light	2.99792458E8_m/_s
_Cc	coulomb constant	8.9875517873682E9_N•_m ² /_coul ²
_g	acceleration of gravity	9.80665_m/_s ²
_Gc	gravitational constant	6.6742E-11_m ³ /_kg/_s ²
_h	Planck's constant	6.6260693E-34_J•_s
_k	Boltzmann's constant	1.3806505E-23_J/_°K
_Me	electron rest mass	9.1093826E-31_kg
_Mn	neutron rest mass	1.67492728E-27_kg
_Mp	proton rest mass	1.67262171E-27_kg
_Na	Avogadro's number	6.0221415E23 /_mol
_q	electron charge	1.60217653E-19_coul
_Rb	Bohr radius	5.291772108E-11_m
_Rc	molar gas constant	8.314472_J/_mol/_°K
_Rdb	Rydberg constant	10973731.568525 /_m
_Vm	molar volume	2.2413996E-2_m ³ /_mol
_ε0	permittivity of a vacuum	8.8541878176204E-12_F/_m
_σ	Stefan-Boltzmann constant	5.670400E-8_W/_m ² /_°K ⁴
_φ0	magnetic flux quantum	2.06783372E-15_Wb

	Description	Value
μ_0	permeability of a vacuum	$1.2566370614359 \times 10^{-6} \text{ N/A}^2$
μ_B	Bohr magneton	$9.27400949 \times 10^{-24} \text{ J} \cdot \text{m}^2/\text{Wb}$

Notes:

- The calculator simplifies unit expressions and displays results according to your default units. Therefore, constant values displayed on your screen may appear different from the values in this table.
- For Greek characters, refer to *Quick Reference Key Table*.
- These values represent the most up-to-date constants available at time of printing from the CODATA Internationally recommended values of the Fundamental Physical Constants available on the National Institute of Standards and Technology (NIST) web site. (<http://physics.nist.gov/cuu/Constants/index.html>).

Length

_Ang	angstrom	_mi	mile
_au	astronomical unit	_mil	1/1000 inch
_cm	centimeter	_mm	millimeter
_fath	fathom	_Nmi	nautical mile
_fm	fermi	_pc	parsec
_ft	foot (ENG/US)	_rod	rod
_in	inch	_yd	yard

_km	kilometer	_μ	micron
_lyr	light year	_Å	angstrom
_m	meter (SI)		

Area

_acre	acre	NONE (SI) (ENG/US)	
_ha	hectare		

Volume

_cup	cup	_ml	milliliter
_floz	fluid ounce	_pt	pint
_flozUK	British fluid ounce	_qt	quart
_gal	gallon	_tbsp	tablespoon
_galUK	British gallon	_tsp	teaspoon
_l	liter	NONE (SI) (ENG/US)	

Time

_day	day	_s	second (SI) (ENG/US)
_hr	hour	_week	week
_min	minute	_yr	year

_ms	millisecond	_μs	microsecond
_ns	nanosecond		

Velocity

_knot	knot	_mph	miles per hour
_kph	kilometers per hour	NONE (SI) (ENG/US)	

Acceleration

no pre-defined units		
----------------------	--	--

Temperature

_°C	°Celsius (For °, press [2nd] [°].)	_°K	°Kelvin
_°F	°Fahrenheit	_°R	°Rankine (no default)

Luminous Intensity

_cd	candela (no default)		
-----	----------------------	--	--

Amount of Substance

_mol	mole (no default)		
------	-------------------	--	--

Mass

_amu	atomic mass unit	_oz	ounce
_gm	gram	_slug	slug
_kg	kilogram (SI)	_ton	ton
_lb	pound (ENG/US)	_tonne	metric ton
_mg	milligram	_tonUK	long ton
_mton	metric ton		

Force

_dyne	dyne	_N	newton (SI)
_kgf	kilogram force	_tonf	ton force
_lbf	pound force (ENG/US)		

Energy

_Btu	British thermal unit (ENG/US)	_J	joule (SI)
_cal	calorie	_kcal	kilocalorie

_erg	erg	_kWh	kilowatt-hour
_eV	electron volt	_latm	liter-atmosphere
_ftlb	foot-pound		

Power

_hp	horsepower (ENG/US)	_W	watt (SI)
_kW	kilowatt		

Pressure

_atm	atmosphere	_mmHg	millimeters of mercury
_bar	bar	_Pa	pascal (SI)
_inH2O	inches of water	_psi	pounds per square inch (ENG/US)
_inHg	inches of mercury	_torr	millimeters of mercury
_mmH2O	millimeters of water		

Viscosity, Kinematic

_St	stokes		
-----	--------	--	--

Viscosity, Dynamic

_P	poise		
----	-------	--	--

Frequency

_GHz	gigahertz	_kHz	kilohertz
_Hz	hertz (SI) (ENG/US)	_MHz	megahertz

Electric Current

_A	ampere (SI) (ENG/US)	_mA	milliampere
_kA	kiloampere	_μA	microampere

Charge

_coul	coulomb (SI) (ENG/US)		
-------	-----------------------	--	--

Potential

_kV	kilovolt	_V	volt (SI) (ENG/US)
_mV	millivolt	_volt	volt

Resistance

_k Ω	kilo ohm	_ohm	ohm
_M Ω	megaohm	_ Ω	ohm (SI) (ENG/US)

Conductance

_mho	mho (ENG/US)	_siemens	siemens (SI)
_mmho	millimho	_ μ mho	micromho

Capacitance

_F	farad (SI) (ENG/US)	_pF	picofarad
_nF	nanofarad	_ μ F	microfarad

Mag Field Strength

_Oe	oersted	NONE	(SI) (ENG/US)
-----	---------	------	---------------

Mag Flux Density

_Gs	gauss	_T	tesla (SI) (ENG/US)
-----	-------	----	---------------------

Magnetic Flux

_Wb	weber (SI) (ENG/US)		
-----	---------------------	--	--

Inductance

_henry	henry (SI) (ENG/US)	_nH	nanohenry
_mH	millihenry	_μH	microhenry

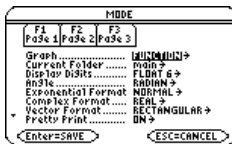
Basic Function Graphing

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

1. Set **Graph** mode (MODE) to **FUNCTION**.
Also set **Angle** mode, if necessary.



2. Define x and y components on Y= Editor (Y=).
3. Select (F4) which defined functions to graph.




Note: To turn off any stat data plots, press F5 5 or use F4 to deselect them.


4. Set the display style for a function.

F6

This is optional. For multiple equations, this helps visually distinguish one from another.



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

 **Zoom** also changes the viewing window.

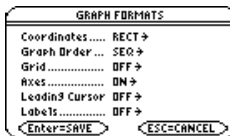
```
xmin=-10.  
xmax=10.  
xscl=1.  
ymin=10.  
ymax=10.  
yscl=1.  
xres=2.
```


6. Change the graph format if necessary.

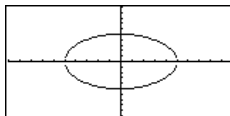
 **9**

– or –

 **F**





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



Exploring the Graph

From the Graph screen, you can:

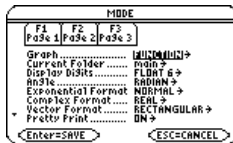
- Display the coordinates of any pixel by using the free-moving cursor, or of a plotted point by tracing a function.
- Use the  **Zoom** toolbar menu to zoom in or out on a portion of the graph.
- Use the  **Math** toolbar menu to find a zero, minimum, maximum, etc.

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 Voyage™ 200 Graphing Calculator 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 *Operating the Calculator*.
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 module specifically describes **y(x)** function graphs, the calculator lets you select from six Graph mode settings.

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
DIFFERENTIAL EQUATION	y'(t) differential equations

Angle Mode

When using trigonometric functions, set the Angle mode for the units (RADIAN, DEGREE or GRADIAN) 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.

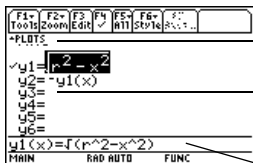
MAIN	RAD AUTO	FUNC
	Angle Mode	Graph Mode

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

1. Press \blacklozenge [Y=] to display the Y= Editor.



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

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.

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

2. Press \downarrow and \uparrow to move the cursor to any undefined function. (Use 2^{nd} \downarrow and 2^{nd} \uparrow to scroll one page at a time.)
 3. Press \boxed{ENTER} or $\boxed{F3}$ 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. Only floats and lists of floats will produce a plot.
- Note:** For an undefined function, you do not need to press \boxed{ENTER} or $\boxed{F3}$. When you begin typing, the cursor moves to the entry line.
5. When you complete the expression, press \boxed{ENTER} .

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

Note: If you accidentally move the cursor to the entry line, press \boxed{ESC} to move it back to the function list.

Editing a Function

From the Y= Editor:

1. Press \leftarrow and \rightarrow to highlight the function.
2. Press $\boxed{\text{ENTER}}$ or $\boxed{\text{F3}}$ to move the cursor to the entry line.
3. Do any of the following:
 - Use \uparrow and \downarrow to move the cursor within the expression and edit it. Refer to “Editing an Expression in the Entry Line” in *Operating the Calculator*.
– or –
 - Press $\boxed{\text{CLEAR}}$ once or twice to clear the old expression, and then type the new one.
4. Press $\boxed{\text{ENTER}}$.

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

Note: To cancel any editing changes, press $\boxed{\text{ESC}}$ instead of $\boxed{\text{ENTER}}$.

Clearing a Function

From the Y= Editor:

To erase:

Do this:

A function from the function list

Highlight the function and press $\boxed{\leftarrow}$ or $\boxed{\text{CLEAR}}$.







To erase:	Do this:
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.

You don't have to clear a function to prevent it from being graphed. You can select the functions you want to graph.

Shortcuts to Move the Cursor

From the Y= Editor:

Press:	To:
  or  	Go to function 1 or to the last defined function, respectively. If the cursor is on or past the last defined function,   goes to function 99.

From the Home Screen or a Program

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 *Additional Graphing Topics*.
- “Overview of Entering a Function” in *Programming*.
- Store an expression directly to a function variable. Refer to:
 - “Storing and Recalling Variable Values” in *Operating the Calculator*.
 - “Creating and Evaluating User-Defined Functions” in *Calculator Home Screen*.

Note: 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.

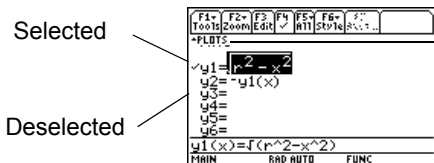
Selecting Functions to Graph

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 \blacklozenge [Y=] 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 $y1=$.

To select or deselect:**Do this:**

A specified function

- Move the cursor to highlight the function.
- Press **[F4]**.

This procedure selects a deselected function or deselects a selected function.

All functions

- Press **[F5]** to display the **All** toolbar menu.
- Select the applicable item.



You don't have to select a function when you enter or edit it; it is selected automatically. To turn off any stat plots, press **[F5]** **5** or use **[F4]** to deselect them.

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 the *Technical Reference* module.
- Use the **PlotsOn** and **PlotsOff** commands for stat plots. Refer to the *Technical Reference* module.

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. Select the **Style** menu and 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.

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.

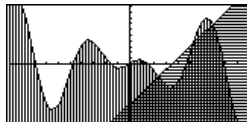
Style	Description
Animate	A round cursor moves along the leading edge of the graph but does not leave a path.
Path	A round cursor moves along the leading edge of the graph and does leave a path.
Above	Shades the area above the graph.
Below	Shades the area below the graph.

To set Line as the style for all functions, press $\boxed{F5}$ and select **4:Reset Styles**.

If You Use Above or Below Shading

The Voyage™ 200 Graphing Calculator 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

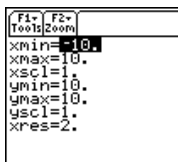
You can also set a function's style from the Home screen or a program. Refer to the **Style** command in the *Technical Reference* module.

Defining the Viewing Window

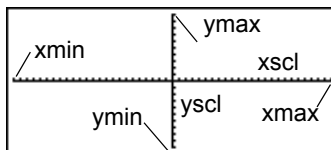
The viewing window represents the portion of the coordinate plane displayed on the Graph screen. By setting Window variables, you can define the viewing window's boundaries and other attributes. Function graphs, parametric graphs, etc., have their own independent set of Window variables.

Displaying Window Variables in the Window Editor

Press \blacklozenge [WINDOW] to display the Window Editor.



Window Variables
(shown in Window Editor)



Corresponding Viewing Window
(shown on Graph screen)

Variable	Description
xmin, xmax, ymin, ymax	Boundaries of the viewing window.
xscl, yscl	Distance between tick marks on the x and y axes.

Variable	Description
xres	Sets pixel resolution (1 through 10) for function graphs. The default is 2. <ul style="list-style-type: none">• 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.

To turn off tick marks, set **xscl=0** and/or **yscl=0**. Small values of **xres** improve the graph's resolution but may reduce the graphing speed.

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.

Values are stored as you type them; you do not need to press **ENTER**. **ENTER** simply moves the cursor to the next Window variable. If you type an expression, it is evaluated when you move the cursor to a different Window variable or leave the Window Editor.

From the Home Screen or a Program

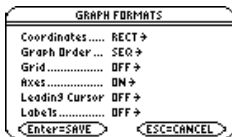
You can also store values directly to the Window variables from the Home screen or a program. Refer to “Storing and Recalling Variable Values” in *Operating the Calculator*.

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

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**.

You also can display the GRAPH FORMATS dialog box from the Y= Editor, Window Editor, or Graph screen. Press: **◆ F**

Format	Description
Coordinates	Shows cursor coordinates in rectangular (RECT) or polar (POLAR) form, or hides (OFF) the coordinates.

Format	Description
Graph Order	Graphs functions one at a time (SEQ) or all at the same time (SIMUL). Not available when Discontinuity Detection is set to ON.
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.
Discontinuity Detection	Eliminates (ON) or allows (OFF) faux asymptotes and connections in a jump discontinuity.

To turn off tick marks, define the viewing window so that $x\text{scl}$ and/or $y\text{scl} = 0$.

Changing Settings

From the GRAPH FORMATS dialog box:

1. Move the cursor to highlight the format setting.
2. Press \odot 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 $\boxed{\text{ENTER}}$.
 - 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.

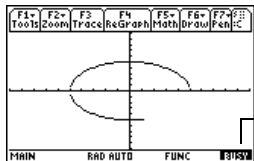
Note: To cancel a menu or exit the dialog box without saving any changes, use **[ESC]** instead of **[ENTER]**.

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]**. The Voyage™ 200 Graphing Calculator automatically graphs the selected functions.



BUSY indicator shows while graphing is in progress.

If you select an **[F2]** **Zoom** operation from the Y= Editor or Window Editor, the Voyage™ 200 automatically displays the Graph screen.

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.
- Use a Zoom operation.

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.

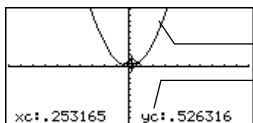
- Changed a stat plot definition.

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 a cursor pad arrow. The cursor moves from the center of the screen, and its coordinates are displayed.




$$y_1(x) = x^2$$

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**.

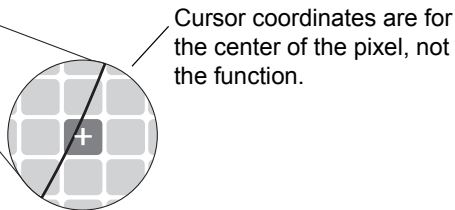
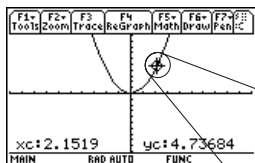
If your screen does not show coordinates, set the graph format so that **Coordinates = RECT** or **POLAR**. Press:

 **F**

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

Note: 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.

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.



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 $\boxed{F3}$.

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.

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.

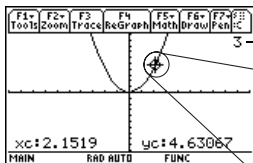
If any stat plots are graphed, the trace cursor appears on the lowest-numbered stat plot.

Moving along a Function

To move the trace cursor:	Do this:
To the previous or next plotted point	Press $\boxed{\leftarrow}$ or $\boxed{\rightarrow}$.
Approximately 5 plotted points (it may be more or less than 5, depending on the xres Window variable)	Press $\boxed{2nd} \boxed{\leftarrow}$ or $\boxed{2nd} \boxed{\rightarrow}$.
To a specified x value on the function	Type the x value and press \boxed{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.



Function number being traced.
For example: $y_3(x)$.

Trace coordinates are those of the function, not the pixel.

If your screen does not show coordinates, set the graph format so that **Coordinates = RECT** or **POLAR**. Press:



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.

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.

Note: Use QuickCenter to trace a function that goes above or below the window.

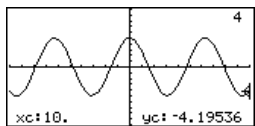
Moving from Function to Function

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

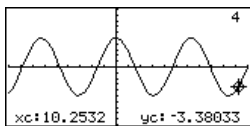
The “previous or next” function is based on the order of the selected functions in the Y= Editor, not the appearance of the functions as graphed on the screen.

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.



Before automatic pan



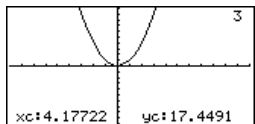
After automatic pan

After an automatic pan, the cursor continues tracing.

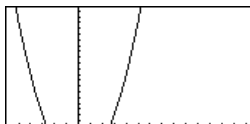
Note: Automatic panning does not work if stat plots are displayed or if a function uses a shaded display style.

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.



Before using QuickCenter



After using QuickCenter

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

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

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:

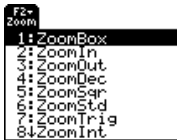
- If Smart Graph regraphed the screen, the cursor appears at the middle x value.
- If Smart Graph does not regraph the screen, the cursor appears at its previous location (before you displayed the other application).

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

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 the *Technical Reference* module.

Note: If you select a **Zoom** tool from the Y=Editor or Window Editor, the Voyage™ 200 Graphing Calculator automatically displays the Graph screen.

Zoom Tool	Description
ZoomBox	Lets you draw a box and zoom in on that box.
ZoomIn , ZoomOut	Lets 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

Zoom Tool	Description						
ZoomTrig	<p>Sets Window variables to preset values that are often appropriate for graphing trig functions. Centers the origin and sets:</p> <table> <tr> <td>$\Delta x = \pi/24$</td> <td>(.130899... radians or 7.5 degrees)</td> <td>$y_{\min} = -4$ $y_{\max} = 4$</td> </tr> <tr> <td>$x_{\text{scl}} = \pi/2$</td> <td>(1.570796... radians or 90 degrees)</td> <td>$y_{\text{scl}} = 0.5$</td> </tr> </table>	$\Delta x = \pi/24$	(.130899... radians or 7.5 degrees)	$y_{\min} = -4$ $y_{\max} = 4$	$x_{\text{scl}} = \pi/2$	(1.570796... radians or 90 degrees)	$y_{\text{scl}} = 0.5$
$\Delta x = \pi/24$	(.130899... radians or 7.5 degrees)	$y_{\min} = -4$ $y_{\max} = 4$					
$x_{\text{scl}} = \pi/2$	(1.570796... radians or 90 degrees)	$y_{\text{scl}} = 0.5$					
ZoomInt	Lets you select a new center point, and then sets Δx and Δy to 1 and sets x scl and y scl to 10.						
ZoomData	Adjusts Window variables so that all selected stat plots are in view.						
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 .						

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

Zooming In with a Zoom Box

1. From the $\boxed{F2}$ **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 \boxed{ENTER} .

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

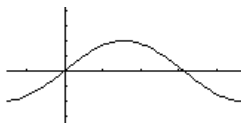
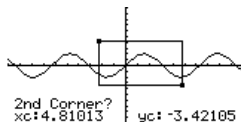
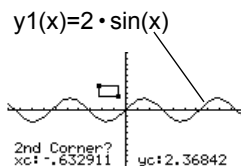
Note: To move the cursor in larger increments, use $\boxed{2nd}$ \blacktriangleright , $\boxed{2nd}$ \blacktriangleleft , etc.

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 \boxed{ENTER} .

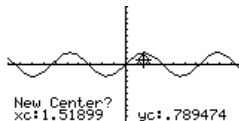
The Graph screen shows the zoomed area. You can cancel **ZoomBox** by pressing \boxed{ESC} before you press \boxed{ENTER} .



Zooming In and Out on a Point

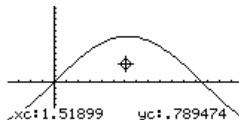
1. From the **F2** **Zoom** menu, select **2:ZoomIn** or **3:ZoomOut**.

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 Voyage™ 200 adjusts the Window variables by the **Zoom** factors defined in **SetFactors**.

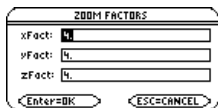


- For a **ZoomIn**, the x variables are divided by **xFact**, and the y variables are divided by **yFact**.
new $x_{min} = x_{min}/xFact$, etc.
- For a **ZoomOut**, the x variables are multiplied by **xFact**, and the y variables are multiplied by **yFact**.
new $x_{min} = x_{min} * xFact$, etc.

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.

- Note:** To exit without saving any changes, press **[ESC]**.
2. Use **[Left]** and **[Right]** 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 **[Up]** or **[Down]** 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

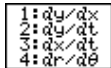
You can restore the Window variables to their default values at any time. From the **[F2] Zoom** menu, select **6:ZoomStd**.

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 $\boxed{F5}$ 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.).

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.
$\int f(x)dx$	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.

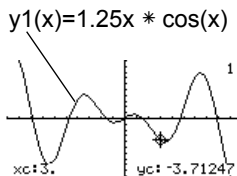
Math Tool	Description
Shade	<p>Depends on the number of functions graphed.</p> <ul style="list-style-type: none"> If only one function is graphed, this shades the function's area above or below the x axis. If two or more functions are graphed, this shades the area between any two functions within an interval.

Note: For Math results, cursor coordinates are stored in system variables x_c and y_c (r_c and θ_c if you use polar coordinates). Derivatives, integrals, distances, etc., are stored in the system variable *sysMath*.

Finding $y(x)$ at a Specified Point

- From the **Graph** screen, press $\boxed{F5}$ and select **1:Value**.
- Type the x value, which must be a real value between **xmin** and **xmax**. The value can be an expression.
- Press \boxed{ENTER} .

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



- Press \odot or \ominus to move the cursor between functions at the entered x value. The corresponding y value is displayed.

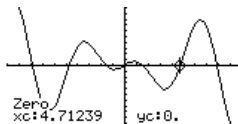
If you press \odot or \ominus , the free-moving cursor appears. You may not be able to move it back to the entered x value.

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

Finding a Zero, Minimum, or Maximum within an Interval

1. From the **Graph** screen, press F5 and select **2:Zero**, **3:Minimum**, or **4:Maximum**.
2. As necessary, use \ominus and $\omin�$ to select the applicable function.
Note: Typing x values is a quick way to set bounds.
3. Set the lower bound for x. Either use \downarrow and \uparrow to move the cursor to the lower bound or type its x value.
4. Press ENTER . A \blacktriangleright 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.

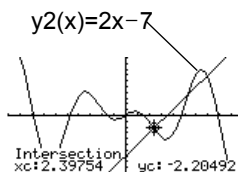


Finding the Intersection of Two Functions within an Interval

1. From the **Graph** screen, press F5 and select **5:Intersection**.
2. Select the first function, using \ominus or $\omin�$ 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 \downarrow and \uparrow to move the cursor to the lower bound or type its x value.

- Press $\boxed{\text{ENTER}}$. A \blacktriangleright at the top of the screen marks the lower bound.
- Set the upper bound, and press $\boxed{\text{ENTER}}$.

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



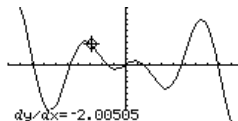
Finding the Derivative (Slope) at a Point

- From the **Graph** screen, press $\boxed{\text{F5}}$ and select **6:Derivatives**. Then select **1:dy/dx** from the submenu.
- As necessary, use \ominus and $\omin�$ to select the applicable function.

- Set the derivative point. Either move the cursor to the point or type its x value.

- Press $\boxed{\text{ENTER}}$.

The derivative at that point is displayed.



Finding the Numerical Integral over an Interval

- From the **Graph** screen, press $\boxed{\text{F5}}$ and select **7:f(x)dx**.
- As necessary, use \ominus and $\omin�$ to select the applicable function.

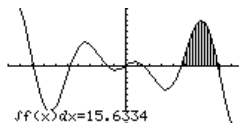
Note: Typing x values is a quick way to set the limits.

- Set the lower limit for x . Either use \leftarrow and \rightarrow to move the cursor to the lower limit or type its x value.
- Press $\boxed{\text{ENTER}}$. A \blacktriangleright at the top of the screen marks the lower limit.

Note: To erase the shaded area, press $\boxed{\text{F4}}$ (**ReGraph**).

- Set the upper limit, and press $\boxed{\text{ENTER}}$.

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

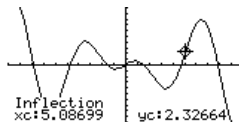


Finding an Inflection Point within an Interval

- From the **Graph** screen, press $\boxed{\text{F5}}$ and select **8:Inflection**.
- As necessary, use \uparrow and \downarrow to select the applicable function.
- Set the lower bound for x . Either use \leftarrow and \rightarrow to move the cursor to the lower bound or type its x value.
- Press $\boxed{\text{ENTER}}$. A \blacktriangleright at the top of the screen marks the lower bound.

- Set the upper bound, and press $\boxed{\text{ENTER}}$.

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

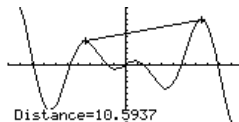


Finding the Distance between Two Points

- From the **Graph** screen, press $\boxed{\text{F5}}$ and select **9:Distance**.

- As necessary, use \ominus and \oplus to select the function for the first point.
- Set the first point. Either use \uparrow or \downarrow to move the cursor to the point or type its x value.
- Press $\boxed{\text{ENTER}}$. A + marks the point.
- If the second point is on a different function, use \ominus and \oplus to select the function.
- Set the second point. (If you use the cursor to set the point, a line is drawn as you move the cursor.)
- Press $\boxed{\text{ENTER}}$.

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

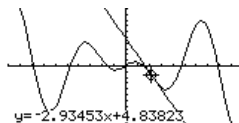


Drawing a Tangent Line

- From the **Graph** screen, press $\boxed{\text{F5}}$ and select **A:Tangent**.
 - As necessary, use \ominus and \oplus to select the applicable function.
- Note:** To erase a drawn tangent line, press $\boxed{\text{F4}}$ (**ReGraph**).

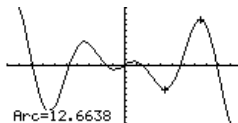
- Set the tangent point. Either move the cursor to the point or type its x value.
- Press $\boxed{\text{ENTER}}$.

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



Finding an Arc Length

1. From the **Graph** screen, press $\boxed{F5}$ and select **B:Arc**.
2. As necessary, use \ominus and $\omin�$ to select the applicable function.
3. Set the first point of the arc. Either use $\textcircled{1}$ or $\textcircled{2}$ to move the cursor or type the x value.
4. Press $\boxed{\text{ENTER}}$. A + marks the first point.
5. Set the second point, and press $\boxed{\text{ENTER}}$.
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 $\boxed{F5}$ 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 $\boxed{\text{ENTER}}$.
 - Below the x axis, press:
N

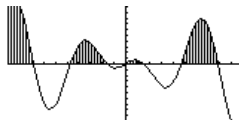
3. Set the lower bound for x . Either use \leftarrow and \rightarrow to move the cursor to the lower bound or type its x value.

Note: If you do not press \leftarrow or \rightarrow , or type an x value when setting the lower and upper bound, x_{\min} and x_{\max} will be used as the lower and upper bound, respectively.

4. Press $\boxed{\text{ENTER}}$. A \blacktriangleright at the top of the screen marks the lower bound.

5. Set the upper bound, and press $\boxed{\text{ENTER}}$.

The bounded area is shaded. To erase the shaded area, press $\boxed{\text{F4}}$ (**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 $\boxed{\text{F5}}$ and select **C:Shade**. The screen prompts for **Above?**
2. As necessary, use \ominus or \oplus to select a function. (Shading will be above this function.)
3. Press $\boxed{\text{ENTER}}$. The cursor moves to the next graphed function, and the screen prompts for **Below?**
4. As necessary, use \ominus or \oplus to select another function. (Shading will be below this function.)
5. Press $\boxed{\text{ENTER}}$.

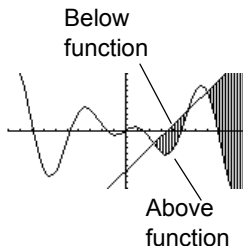
6. Set the lower bound for x . Either use \leftarrow and \rightarrow to move the cursor to the lower bound or type its x value.

Note: If you do not press \leftarrow or \rightarrow , or type an x value when setting the lower and upper bound, x_{\min} and x_{\max} will be used as the lower and upper bound, respectively.

7. Press $\boxed{\text{ENTER}}$. A \blacktriangleright at the top of the screen marks the lower bound.

8. Set the upper bound, and press $\boxed{\text{ENTER}}$.

The bounded area is shaded. To erase the shaded area, press $\boxed{\text{F4}}$ (**ReGraph**).



Polar Graphing

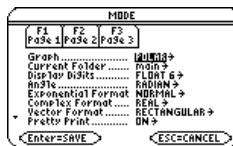
Overview of Steps in Graphing Polar Equations

To graph polar equations, use the same general steps used for $y(x)$ functions as described in *Basic Function Graphing*. Any differences that apply to polar equations are described on the following pages.

Graphing Polar Equations

1. Set **Graph** mode ($\overline{[MODE]}$) to **POLAR**. Also set **Angle** mode, if necessary.
2. Define x and y components on $Y=$ Editor ($\overline{[Y=]}$).
3. Select ($\overline{[F4]}$) which defined equations to graph. Select the x or y component, or both.

Note: To turn off any stat data plots press $\overline{[F5]}$ 5 or use $\overline{[F4]}$ to deselect them.



4. Set the display style for an equation. You can set either the x or y component.

F6

This is optional. For multiple equations, this helps visually distinguish one from another.



5. Define the viewing window (**◀** [WINDOW]).

F2 Zoom also changes the viewing window.

```

θmin=0.
θmax=12.5663706144
θstep=1.3089969389957
xmin=-10.
xmax=10.
xscl=1.
ymin=10.
ymax=10.
yscl=1.
  
```

6. Change the graph format if necessary.

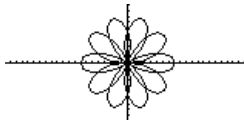
F1 9

– or –

◀ F



7. 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

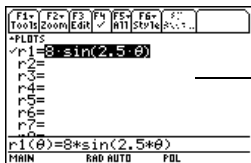
This module assumes that you already know how to graph $y(x)$ functions as described in *Basic Function Graphing*. This section describes the differences that apply to polar equations.

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 $r_1(\theta)$ through $r_9(\theta)$.

You can use the **Define** command from the Home screen (see the *Technical Reference* module) 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 $r(\theta)$ 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 $r(\theta)$ 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 **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.

Variable	Description
θ_{\min} , θ_{\max}	Smallest and largest θ values to evaluate.

Variable	Description
θstep	Increment for the θ value. Polar equations are evaluated at: $r(\theta_{\min})$ $r(\theta_{\min}+\theta\text{step})$ $r(\theta_{\min}+2(\theta\text{step}))$... not to exceed ... $r(\theta_{\max})$
x_{\min} , x_{\max} , y_{\min} , y_{\max}	Boundaries of the viewing window.
$xscl$, $yscl$	Distance between tick marks on the x and y axes.

Note: You can use a negative θstep . If so, θ_{\min} must be greater than θ_{\max} .

Standard values (set when you select **6:ZoomStd** from the $\boxed{F2}$ **Zoom** toolbar menu) are:

$\theta_{\min} = 0.$		$x_{\min} = -10.$	$y_{\min} = -10.$
$\theta_{\max} = 2\pi$	(6.2831853... radians or 360 degrees)	$x_{\max} = 10.$	$y_{\max} = 10.$
$\theta\text{step} = \pi/24$	(.1308996... radians or 7.5 degrees)	$xscl = 1.$	$yscl = 1.$

You may need to change the standard values for the θ variables (θ_{\min} , θ_{\max} , θstep) to ensure that enough points are plotted.

Setting the Graph Format

To display coordinates as r and θ values, use:

F1 9

– or –

F

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.

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. <ul style="list-style-type: none">• 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 = $\pi/24$).

Tool	For Polar Graphs:
------	-------------------

- F3 Trace** Lets you move the cursor along a graph one θ step at a time.
- When you begin a trace, the cursor is on the first selected equation at θ min.
 - 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-89 Titanium / Voyage™ 200 Graphing Calculator 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/d θ** at a point defined for a specified θ value.

During a trace, you can also evaluate $r(\theta)$ by typing the θ value and pressing **ENTER**.

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

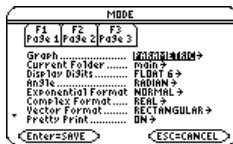
Parametric Graphing

Overview of Steps in Graphing Parametric Equations

To graph parametric equations, use the same general steps used for $y(x)$ functions as described in *Basic Function Graphing*. Any differences that apply to parametric equations are described on the following pages.

Graphing Parametric Equations

1. Set **Graph** mode ($\boxed{\text{MODE}}$) to **PARAMETRIC**. Also set **Angle** mode, if necessary.



2. Define x and y components on $Y=$ Editor ($\boxed{\blacklozenge}$ $\boxed{[Y=]}$).
3. Select ($\boxed{\text{F4}}$), which defined equations to graph. Select the x or y component, or both.



Note: To turn off any stat data plots, press $\boxed{\text{F5}}$ **5** or use $\boxed{\text{F4}}$ to deselect them.

4. Set the display style for an equation. You can set either the x or y component.

F6

This is optional. For multiple equations, this helps visually distinguish one from another.



5. Define the viewing window (**◀** [WINDOW]).

F2 **Zoom** also changes the viewing window.

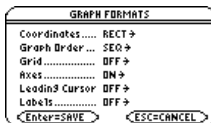
```
tmin=0.
tmax=3.
tstep=.02
xmin=-2.
xmax=25.
xsc1=5.
ymin=-2.
ymax=10.
yrc1=5.
```

6. Change the graph format if necessary.

F1 **9**

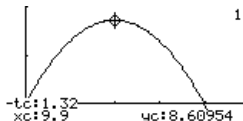
– or –

◀ **F**



7. 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

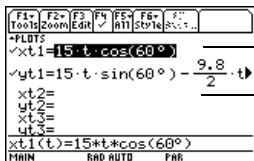
This module assumes that you already know how to graph $y(x)$ functions as described in Basic Function Graphing. This section describes the differences that apply to parametric equations.

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 *Tables*.)



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:

Enter:	Instead of:	Because:
$t*\cos(60)$	$t\cos(60)$	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 Voyage™ 200 Graphing Calculator simply returns the function name, not a number.

Note: When using **t**, be sure implied multiplication is valid for your situation. You can use the **Define** command from the Home screen (see the *Technical Reference* module) 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 *Tables*. 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 Voyage™ 200 automatically sets the y component to **Dot**.

Note: 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 **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.

Variable	Description
tstep	Increment for the t value. Parametric equations are evaluated at: $x(t_{\min})$ $y(t_{\min})$ $x(t_{\min}+t_{\text{step}})$ $y(t_{\min}+t_{\text{step}})$ $x(t_{\min}+2(t_{\text{step}}))$ $y(t_{\min}+2(t_{\text{step}}))$... not to exceed not to exceed ... $x(t_{\max})$ $y(t_{\max})$
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 $\boxed{F2}$ **Zoom** toolbar menu) are:

$t_{\min} = 0$		$x_{\min} = -10.$	$y_{\min} = -10.$
$t_{\max} = 2\pi$	(6.2831853... radians or 360 degrees)	$x_{\max} = 10.$	$y_{\max} = 10.$
$t_{\text{step}} = \pi/24$	(.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.

Note: During a trace, you can also evaluate $x(t)$ and $y(t)$ by typing the t value and pressing **ENTER**. 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: <ul style="list-style-type: none">• 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 = $\pi/24$).
F3 Trace	Lets you move the cursor along a graph one tstep at a time. <ul style="list-style-type: none">• 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 Voyage™ 200 will not automatically pan the viewing window. However, you can use QuickCenter.

Tool	For Parametric Graphs:
F5 Math	<p>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:</p> <ul style="list-style-type: none">• 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.

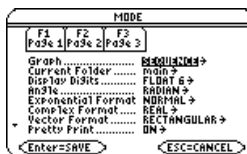
Sequence Graphing

Overview of Steps in Graphing Sequences

To graph sequences, use the same general steps used for $y(x)$ functions as described in *Basic Function Graphing*. Any differences are described on the following pages.

Graphing Sequences

1. Set **Graph** mode (MODE) to **SEQUENCE**.
Also set **Angle** mode, if necessary.



2. Define sequences and, if needed, initial values on **Y= Editor** (\blacktriangleright [Y=]).
3. Select (F4) which defined sequences to graph. Do not select initial values.




Note: To turn off any stat data plots, press F5 5 or use F4 to deselect them.


4. Set the display style for a sequence.

F6

For sequences, the default style is **Square**.



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

 **Zoom** also changes the viewing window.

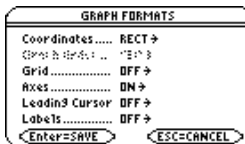
```
nmin=0,
nmax=50,
PlotStart=1,
PlotStep=1,
xmin=0,
xmax=50,
xsc1=10,
ymin=0,
ymax=6000,
yrc1=1000.
```

6. Change the graph format if necessary.

 **9**

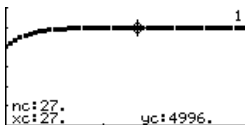
— OR —

 **F**





7. 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 sequence.
- Use the  **Zoom** toolbar menu to zoom in or out on a portion of the graph.
- Use the  **Math** toolbar menu to evaluate a sequence. Only **1:Value** is available for sequences.
- Plot sequences on **Time** (the default), **Web**, or **Custom** axes.

Note: You can also evaluate a sequence while tracing. Simply enter the n value directly from the keyboard.

Differences in Sequence and Function Graphing

This module assumes that you already know how to graph $y(x)$ functions as described in *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

```
F1- F2- F3- F4- F5- F6- F7-
Tool 20000 Edit ✓ All Stop Res...
-PLOTS
✓ u1=iPart(.8*u1(n-1)+1000)
u1=4000
u2=
u3=
u4=
u1(n)=iPart(.8*u1(n-1)+1000)
MAIN RAD AUTO SEQ
```

You can define sequences $u_1(n)$ through $u_{99}(n)$.

Use u_i only for recursive sequences, which require one 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. You must use a list to enter two or more initial values.

```
✓ u3=u3(n-1)+u3(n-2)
u3={1 0}
u4=
u4=
u3={1,0}
MAIN RAD AUTO SEQ
```

Enter $\{1,0\}$ even though $\{1 \ 0\}$ is shown in the sequence list.

If a sequence requires an initial value but you do not enter one, you will get an error when graphing.

On the Y= Editor, Axes lets you select the axes that are used to graph the sequences. Optionally, for sequences only, you can select different axes for the graph. TIME is the default.

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.

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.

Note: You can use the **Define** command from the Home screen (see *Technical Reference*) to define functions and equations for any graphing mode, regardless of the current mode.

Selecting Sequences

With TIME and WEB axes, the Voyage™ 200 graphs only the selected sequences. If you entered any sequences that require an initial value, you must enter the corresponding u_i 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.

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 <i>for graphing only</i> . 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.

Note: Both **nmin** and **nmax** must be positive integers, although **nmin** can be zero; **nmin**, **nmax**, **plotstrt** and **plotstep** must be integers ≥ 1 . If you do not enter integers, they will be rounded to integers.

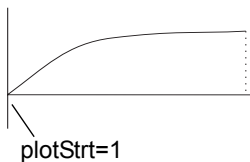
Standard values (set when you select **6:ZoomStd** from the $\boxed{F2}$ **Zoom** toolbar menu) are:

$nmin = 1.$	$xmin = -10.$	$ymin = -10.$
$nmax = 10.$	$xmax = 10.$	$ymax = 10.$
$plotstr = 1.$	$xsc1 = 1.$	$ysc1 = 1.$
$plotstep = 1.$		

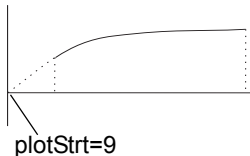
You may need to change the standard values for the n and plot variables to ensure that sufficient points are plotted.

To see how **plotstr** affects graph, look at the following examples of a recursive sequence.

This graph is plotted beginning with the 1st term.



This graph is plotted beginning with the 9th term.

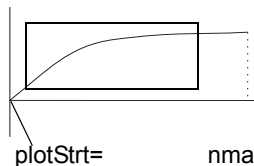


Note: Both of these graphs use the same Window variables, except for **plotstr**.

With TIME axes (from 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.

Tool	For Sequence Graphs:
Free-Moving Cursor	Works just as it does for function graphs.

Tool For Sequence 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 (**nmmin, 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.

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.

During a trace, you can evaluate a sequence by typing a value for n and pressing **ENTER**. You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.

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 module.

Displaying the AXES Dialog Box

From the Y= Editor, Axes:

- 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).

Item	Description
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.

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 Voyage™ 200:

- 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 $u_1(n) = \sqrt{5 - u_1(n-1)}$ and an initial value of $u_1=1$.

The Voyage™ 200 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 $\boxed{F3}$. The web is then drawn step-by-step as you move the trace cursor (you must have an initial value before using Trace).

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 $\boxed{F3}$ to trace the web and display its coordinates.

The web:

1. Starts on the x axis at the initial value u_i (when **plotstr** = 1).
2. Moves vertically (either up or down) to the sequence.

- Moves horizontally to the $y=x$ reference line.
- Repeats this vertical and horizontal movement until $n=nmax$.

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).

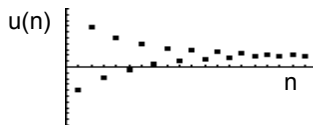
Example: Convergence

- On the Y= Editor (\blacklozenge [Y=]), define $u1(n) = -.8u1(n-1) + 3.6$. Set initial value $u1 = -4$.
- Set **Axes = TIME**.
- On the Window Editor (\blacklozenge [WINDOW]), set the Window variables.

$nmin=1$	$xmin=0$	$ymin=-10$
$nmax=25$	$xmax=25$	$ymax=10$
$plotstrt=1$	$xscl=1$	$yscl=1$
$plotstep=1$		

- Graph the sequence (\blacklozenge [GRAPH]).

By default, a sequence uses the **Square** display style.



- On the Y= Editor, set **Axes = WEB** and **Build Web = AUTO**.

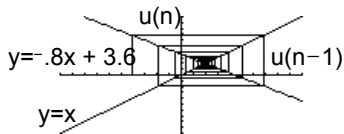
6. On the Window Editor, change the Window variables.

$n_{\min}=1$	$x_{\min}=-10$	$y_{\min}=-10$
$n_{\max}=25$	$x_{\max}=10$	$y_{\max}=10$
$\text{plotstr}=1$	$x_{\text{scl}}=1$	$y_{\text{scl}}=1$
$\text{plotstep}=1$		

7. Regraph the sequence.

Web plots are always shown as lines, regardless of the selected display style.

Note: During a trace, you can move the cursor to a specified n value by typing the value and pressing **ENTER**.



8. Press **F3**. As you press **↓**, the trace cursor follows the web. The screen displays the cursor coordinates n_c , x_c , and y_c (where x_c and y_c represent $u(n-1)$ and $u(n)$, respectively).

As you trace to larger values of n_c , you can see x_c and y_c approach the convergence point.

Note: When the n_c value changes, the cursor is on the sequence. The next time you press **↓**, n_c stays the same but the cursor is now on the $y=x$ reference line.

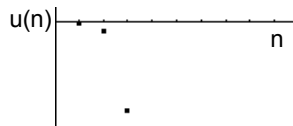
Example: Divergence

1. On the Y= Editor (\blacklozenge [Y=]), define $u1(n) = 3.2u1(n-1) - .8(u1(n-1))^2$. Set initial value $u1 = 4.45$.
2. Set **Axes = TIME**.
3. On the Window Editor (\blacklozenge [WINDOW]), set the Window variables.

nmin=0	xmin=0	ymin=-75
nmax=10	xmax=10	ymin=10
plotstrt=1	xscl=1	yscl=1
plotstep=1		

4. Graph the sequence (\blacklozenge [GRAPH]).

Because the sequence quickly diverges to large negative values, only a few points are plotted.

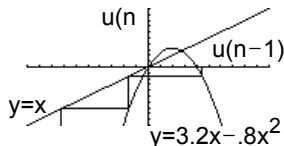


5. On the Y= Editor, set **Axes = WEB** and **Build Web = AUTO**.
6. On the Window Editor (\blacklozenge [WINDOW]), set the Window variables.

nmin=0	xmin=-10	ymin=-10
nmax=10	xmax=10	ymin=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

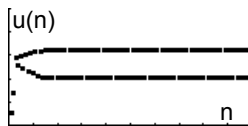
This example shows how the initial value can affect a sequence.

1. On the Y= Editor (\blacklozenge [Y=]), use the same sequence defined in the divergence example: $u_1(n) = 3.2u_1(n-1) - .8(u_1(n-1))^2$. Set initial value $u_{i1} = 0.5$.
2. Set **Axes = TIME**.
3. On the Window Editor (\blacklozenge [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		

4. Graph the sequence (\blacklozenge [GRAPH]).

Note: Compare this graph with the divergence example. This is the same sequence with a different initial value.



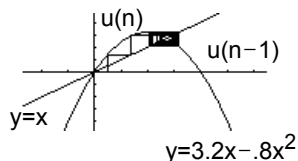
5. On the Y= Editor, set **Axes = WEB** and **Build Web = AUTO**.

6. On the Window Editor (\square [WINDOW]), set the Window variables.

$n_{\min}=1$	$x_{\min}=2.68$	$y_{\min}=4.7$
$n_{\max}=100$	$x_{\max}=6.47$	$y_{\max}=47$
$\text{plotstrt}=1$	$x_{\text{scl}}=1$	$y_{\text{scl}}=1$
$\text{plotstep}=1$		

7. Regraph the sequence.

Note: The web moves to an orbit oscillating between two stable points.

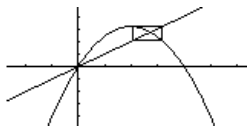


8. Press \square [F3]. Then use \downarrow to trace the web.

As you trace to larger values of n_c , notice that x_c and y_c oscillate between 2.05218 and 3.19782.

9. On the Window Editor, set **plotstrt=50**. Then regraph the sequence.

Note: By starting the web plot at a later term, the stable oscillation orbit is shown more clearly.



Using Custom Plots

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 foxes that maintain population equilibrium in a certain region.

R = Number of rabbits

M = Growth rate of rabbits if there are no foxes (use .05)

K = Rate at which foxes can kill rabbits (use .001)

W = Number of foxes

G = Growth rate of foxes if there are rabbits (use .0002)

D = Death rate of foxes 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 (\square [Y=]), define the sequences and initial values for R_n and W_n .

$$u1(n) = u1(n-1) * (1 + .05 - .001 * u2(n-1))$$

$$ui1 = 200$$

$$u2(n) = u2(n-1) * (1 + .0002 * u1(n-1) - .03)$$

$$ui2 = 50$$

Note: Assume there are initially 200 rabbits and 50 foxes.

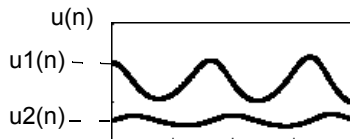
2. Set **Axes = TIME**.

3. On the Window Editor (\blacklozenge [WINDOW]), set the Window variables.

$n_{\min}=0$	$x_{\min}=0$	$y_{\min}=0$
$n_{\max}=400$	$x_{\max}=400$	$y_{\max}=300$
$\text{plotstr}=1$	$x_{\text{scl}}=100$	$y_{\text{scl}}=100$
$\text{plotstep}=1$		

4. Graph the sequence
(\blacklozenge [GRAPH]).

Note: Use $\boxed{F3}$ to individually trace the number of rabbits $u1(n)$ and foxes $u2(n)$ over time (n) .

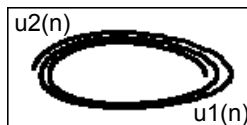


5. On the Y= Editor, set **Axes = CUSTOM**, **X Axis = u1**, and **Y Axis = u2**.
6. On the Window Editor (\blacklozenge [WINDOW]), set the Window variables.

$n_{\min}=0$	$x_{\min}=84$	$y_{\min}=25$
$n_{\max}=400$	$x_{\max}=237$	$y_{\max}=75$
$\text{plotstr}=1$	$x_{\text{scl}}=50$	$y_{\text{scl}}=10$
$\text{plotstep}=1$		

7. Regraph the sequence.

Note: Use $\boxed{F3}$ to trace both the number of rabbits (x_c) and foxes (y_c) over the cycle of 400 generations.



3. Set Window variables (\blacklozenge [WINDOW]) so that **nmin** has the same value as **tblStart**.

```
nmin=1.
nmax=10.
plotStart=1.
plotStep=1.
xmin=-10.
xmax=10.
xsc1=1.
ymin=-10.
ymax=10.
ysc1=1.
```

4. Display the table (\blacklozenge [TABLE]).

F1	F2	F3	F4	F5	F6	F7
Tools	Setup	Eq	Stat	Calc	Draw	Table
	u1					
1.	1.					
2.	1.					
3.	2.					
4.	3.					
5.	5.					
n=1.						
FMIN RAD AUTO SEQ						

Fibonacci sequence is
in column 2.

5. Scroll down the table (\ominus or $\boxed{2nd}$ \ominus) to see more of the sequence.

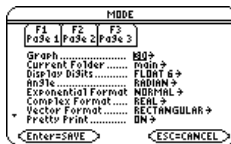
3D Graphing

Overview of Steps in Graphing 3D Equations

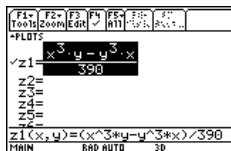
To graph 3D equations, use the same general steps used for $y(x)$ functions as described in *Basic Function Graphing*. Any differences that apply to 3D equations are described on the following pages.

Graphing 3D Equations

1. Set Graph mode ($\boxed{\text{MODE}}$) to **3D**. Also set **Angle** mode, if necessary.



2. Define 3D equations on Y= Editor (\blacklozenge $\boxed{[Y=]}$).
3. Select ($\boxed{F4}$) which equation to graph. You can select only one 3D equation.



To turn off any stat data plots, press $\boxed{F5}$ 5 or use $\boxed{F4}$ to deselect them.

4. Define the viewing cube (\blacklozenge $\boxed{[\text{WINDOW}]}$).

For 3D graphs, the viewing window is called the viewing cube. $\boxed{F2}$ **Zoom** also changes the viewing cube.

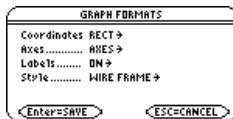
```
eyeθ=20.  
eyeφ=70.  
eyew=0.  
xmin=-10.  
xmax=10.  
xgrid=14.  
ymin=-10.  
ymax=10.  
ygrid=14.  
zmin=-10.  
zmax=10.  
ncontour=5.
```

5. Change the graph format if necessary.

[F1] 9

– or –

[F] 9



Note: To help you see the orientation of 3D graphs, turn on **Axes and Labels**.

6. Graph the selected equations

[F] [GRAPH].



Note: Before displaying the graph, the screen shows the “percent evaluated.”

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.

You can also evaluate **$z(x,y)$** while tracing. Type the x value and press **[ENTER]**; then type the y value and press **[ENTER]**.

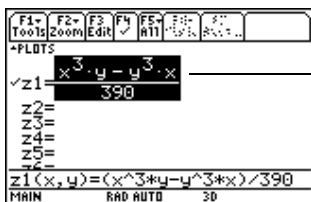
Differences in 3D and Function Graphing

This module assumes that you already know how to graph $y(x)$ functions as described in 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 $z99(x,y)$.

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.

Note: You can use the **Define** command from the Home screen (see the Technical Reference module) to define functions and equations for any graphing mode, regardless of the current mode.

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 Style toolbar menu is dimmed.

For 3D equations, however, you can use:

 9

– or –

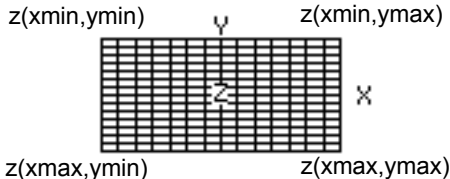
 F

to set the Style format to WIRE FRAME or HIDDEN SURFACE.

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.

Variable	Description
$eye\theta$, $eye\phi$, $eye\psi$	Angles (always in degrees) used to view the graph.

Variable	Description
xmin, xmax, ymin, ymax, zmin, zmax	Boundaries of the viewing cube.
xgrid, ygrid	<p>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.</p> <p>The incremental value along x and y is calculated as:</p> $x \text{ increment} = \frac{x_{\max} - x_{\min}}{x_{\text{grid}}}$ $y \text{ increment} = \frac{y_{\max} - y_{\min}}{y_{\text{grid}}}$ <p>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.</p> 
ncontour	The number of contours evenly distributed along the displayed range of z values.

Note: If you enter a fractional number for **xgrid** or **ygrid**, it is rounded to the nearest whole number ≥ 1 . The 3D mode does not have **scl** Window variables, so you cannot set tick marks on the axes.

Standard values (set when you select **6:ZoomStd** from the $\boxed{F2}$ **Zoom** toolbar menu) are:

eye θ = 20.	xmin = -10.	ymin = -10.	zmin = -10.
eye ϕ = 70.	xmax = 10.	ymax = 10.	zmax = 10.
eye ψ = 0.	xgrid = 14.	ygrid = 14.	ncontour = 5.

You may need to increase the standard values for the grid variables (**xgrid**, **ygrid**) to ensure that enough points are plotted.

Note: Increasing the grid variables decreases the graphing speed.

Setting the Graph Format

The Axes and Style formats are specific to the 3D graphing mode.

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 use:

$\boxed{F1}$ **9**

– or –



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, 3:ZoomOut, 5:ZoomSqr, 6:ZoomStd, A:ZoomFit, B:Memory, C:SetFactors
- Only **x (xmin, xmax)**, **y (ymin, ymax)**, and **z (zmin, zmax)** Window variables are affected.
- The **grid (xgrid, ygrid)** and **eye (eye θ , eye ϕ , eye ψ)** 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**.

Tool	For 3D Graphs:
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 .





Note: 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

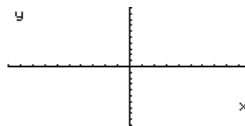
On a 3D surface, the cursor always follows along a grid wire.

Cursor Key	Moves the cursor to the next grid point in the:
	Positive x direction
	Negative x direction
	Positive y direction
	Negative y direction

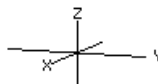
Note: You can move the cursor only within the x and y boundaries set by Window variables **xmin**, **xmax**, **ymin**, and **ymax**.

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.



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.



$\text{eye}\theta=20$ $\text{eye}\phi=70$ $\text{eye}\psi=0$

Note: To show the axes and their labels from the Y= Editor, Window Editor, or Graph screen, use:

F

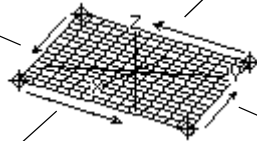
Simple Example of Moving the Cursor

The following graph shows a sloped plane that has the equation $z_1(x,y) = -(x + y) / 2$. Suppose you want to trace around the displayed boundary.

When you press $\boxed{F3}$, the trace cursor appears at the midpoint of the xy grid. Use the cursor pad to move the cursor to any edge.

⬇ moves in a positive x direction, up to xmax.

⬆ moves in a negative y direction, back to ymin.



⬆ moves in a positive y direction, up to ymax.

⬇ moves in a negative x direction, back to xmin.

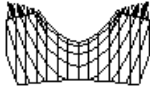
By displaying and labeling the axes, you can more easily see the pattern in the cursor movement. To move grid points closer together, you can increase Window variables **xgrid** and **ygrid**.

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

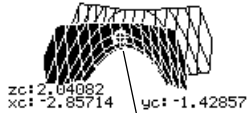
On more complex shapes, the cursor may appear as if it is not on a grid point. This is an optical illusion caused when the cursor is on a hidden surface.

For example, consider a saddle shape $z_1(x,y) = (x^2 - y^2) / 3$. The following graph shows the view looking down the y axis.



```
eyeθ=90.  
eyeφ=70.  
eyeψ=0.  
xmin=-10.  
xmax=10.  
xgrid=14.  
ymin=-10.  
ymax=10.  
ygrid=14.  
zmin=-10.  
zmax=10.  
ncontour=5.
```

Now look at the same shape at 10° from the x axis ($\text{eye}\theta = 10$).



You can move the cursor so that it does not appear to be on a grid point.



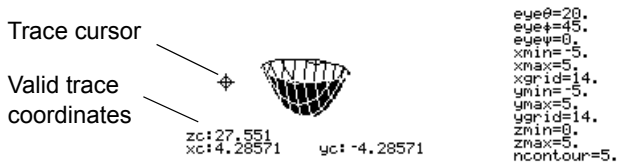
If you cut away the front side, you can see the cursor is actually on a grid point on the hidden back side.

Note: To cut away the front of the saddle in this example, set $\text{xmax}=0$ to show only negative x values.

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 $\mathbf{z(x,y)}$ for the corresponding x and y values.

For example, suppose you trace the paraboloid $z(x,y) = x^2 + .5y^2$ graphed with the indicated Window variables. You can easily move the cursor to a position such as:



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.

Note: QuickCenter lets you center the viewing cube on the cursor's location. Simply press **ENTER**.

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).

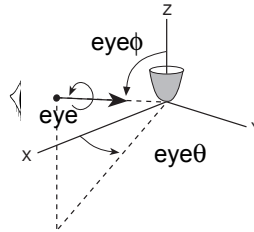
Rotating and/or Elevating the Viewing Angle

In 3D graphing mode, the **eyeθ** and **eyeφ** Window variables let you set viewing angles that determine your line of sight. The **eyeψ** Window variable lets you rotate the graph around that line of sight.

How the Viewing Angle Is Measured

The viewing angle has three components:


- **eye θ** — angle in degrees from the positive x axis.
- **eye ϕ** — angle in degrees from the positive z axis.
- **eye ψ** — angle in degrees by which the graph is rotated counter-clockwise around the line of sight set by **eye θ** and **eye ϕ** .



```
eyeθ=20.  
eyeφ=70.  
eyeψ=0.  
xmin=-10.  
xmax=10.  
xgrid=14.  
ymin=-10.  
ymax=10.  
ygrid=14.  
zmin=-10.  
zmax=10.  
ncontour=5.
```

Do not enter a $^{\circ}$ symbol. For example, type 20, 70, and 0, not 20 $^{\circ}$, 70 $^{\circ}$ and 0 $^{\circ}$.

Note: When **eye ψ =0**, the z axis is vertical on the screen. When **eye ψ =90**, the z axis is rotated 90 $^{\circ}$ counterclockwise and is horizontal.

In the Window Editor ( [WINDOW]), always enter **eye θ** , **eye ϕ** , and **eye ψ** in degrees, regardless of the current angle mode.

Effect of Changing eye θ theta

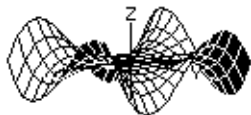
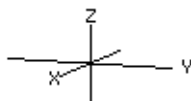
The view on the Graph screen is always oriented along the viewing angle. From this point of view, you can change **eye θ** to rotate the viewing angle around the z axis.

$$z1(x,y) = (x^3y - y^3x) / 390$$

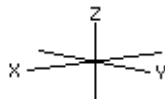
In this example **eye ϕ** =
70



eye θ = 20



eye θ = 50



eye θ = 80



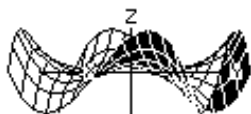
Note: This example increments **eye θ** by 30.

Effect of Changing $\text{eye}\phi$

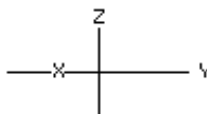
By changing $\text{eye}\phi$, you can elevate your viewing angle above the xy plane. If $90 < \text{eye}\phi < 270$, the viewing angle is below the xy plane.

$$z1(x,y) = (x^3y - y^3x) / 390$$

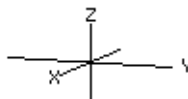
In this example $\text{eye}\theta = 20$



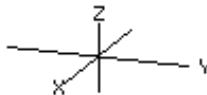
$\text{eye}\phi = 90$



$\text{eye}\phi = 70$



$\text{eye}\phi = 50$



Note: This example starts on the xy plane ($\text{eye}\phi = 90$) and decrements $\text{eye}\phi$ by 20 to elevate the viewing angle.

Effect of Changing $\text{eye}\psi$

The view on the Graph screen is always oriented along the viewing angles set by $\text{eye}\theta$ and $\text{eye}\phi$. You can change $\text{eye}\psi$ to rotate the graph around that line of sight.

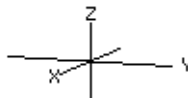
Note: During rotation, the axes expand or contract to fit the screen's width and height. This causes some distortion as shown in the example.

$$z1(x,y)=(x^3y-y^3x) / 390$$

In this example,
eye θ =20 and eye ϕ =70



eye ψ = 0



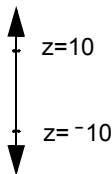
eye ψ = 45



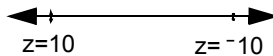
eye ψ = 90



When **eye ψ =0**, the z axis runs the height of the screen.




When **eye ψ =90**, the z axis runs the width of the screen.



As the z axis rotates 90° , its range (-10 to 10 in this example) expands to almost twice its original length. Likewise, the x and y axes expand or contract.

From the Home Screen or a Program

The **eye** values are stored in the system variables **eye θ** , **eye ϕ** , and **eye ψ** . You can access or store to these variables as necessary.

 To type ϕ or ψ , press **2nd** **G** **F** or **2nd** **G** **Y** respectively. You can also press **2nd** **[CHAR]** and use the Greek menu.

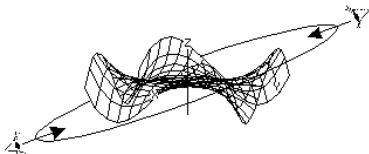
Animating a 3D Graph Interactively

After plotting any 3D graph, you can change the viewing angle interactively by using the cursor.

The Viewing Orbit

When using \leftarrow and \rightarrow to animate a graph, think of it as moving the viewing angle along its “viewing orbit” around the graph.

Moving along this orbit can cause the z axis to wobble slightly during the animation.



Note: The viewing orbit affects the eye Window variables in differing amounts.

Animating the Graph

To:	Do this:
Animate the graph incrementally.	Press and release the cursor quickly.
Move along the viewing orbit.	⬅ or ➡
Change the viewing orbit's elevation. (primarily increases or decreases eye ϕ)	⤴ or ⤵
Animate the graph continuously.	Press and hold the cursor for about 1 second, and then release it. To stop, press ESC , ENTER , ON , or the space bar.
Change between 4 animation speeds (increase or decrease the incremental changes in the eye Window variables).	Press + or - .
Change the viewing angle of a non- animated graph to look along the x, y, or z axis.	Press X, Y or Z, respectively.
Return to the initial eye angle values.	Press 0 (zero).

Notes: If the graph is shown in expanded view, it returns to normal view automatically when you press a cursor key.

- After animating the graph, you can stop and then re-start the animation in the same direction by pressing:
ENTER or space bar
- During an animation, you can switch to the next graph format style by pressing:
F
- You can view a graphic that shows the eye angles.

Animating a Series of Graph Pictures

You can also animate a graph by saving a series of graph pictures and then flipping (or cycling) through those pictures. Refer to “Animating a Series of Graph Pictures” Additional Graphing Topics. This method gives you more control over the Window variable values, particularly **eye ψ** , which rotates the graph.

Changing the Axes and Style Formats

With its default settings, Voyage™ 200 Graphing Calculator 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

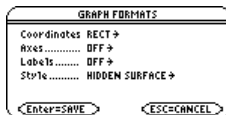
From the Y= Editor, Window Editor, or Graph screen, press:

F1 **9**

– or –



- 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

To display the valid **Axes** settings, highlight the current setting and press **⏏**.



$$z1(x,y) = x^2 + .5y^2$$

- **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.

Note: Setting Labels = ON is helpful when you display either type of 3D axes.

Examples of Style Settings

Note: 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 \downarrow .

```
1: WIRE FRAME
2: HIDDEN SURFACE
3: CONTOUR LEVELS
4: WIRE AND CONTOUR
5: IMPLICIT PLOT
```

- WIRE FRAME — Shows the 3D shape as a transparent wire frame.
- HIDDEN SURFACES — Uses shading to differentiate the two sides of the 3D shape.



Later sections in this module describe CONTOUR LEVELS, WIRE AND COUNTOUR, and implicit plots.

Be Aware of Possible Optical Illusions

The eye angles used to view a graph ($\mathbf{eye\theta}$, $\mathbf{eye\phi}$, and $\mathbf{eye\psi}$ 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.

**Looking down
from above the xy plane**



$\text{eye}\theta = 20, \text{eye}\phi = 55, \text{eye}\psi = 0$

**Looking up
from below the xy plane**



$\text{eye}\theta = 20, \text{eye}\phi = 120, \text{eye}\psi = 0$



Note: The first two examples show the graphs as displayed on the screen. The second two examples use artificial shading (which is not displayed on the screen) to show the front of the box.

To minimize the effect of optical illusions, use the GRAPH FORMATS dialog box to set Style = HIDDEN SURFACE.

Contour Plots

In a contour plot, a line is drawn to connect adjacent points on the 3D graph that have the same z value. This module discusses the CONTOUR LEVELS and WIRE AND CONTOUR graph format styles.

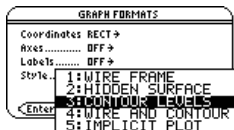
Selecting the Graph Format Style

In 3D graphing mode, define an equation and graph it as you would any 3D equation, with the following exception. Display the GRAPH FORMATS dialog box by pressing **F1** 9 from the Y= Editor, Window editor, or Graph screen. Then set:

Style = CONTOUR LEVELS

– Or –

Style = WIRE AND CONTOUR



- For CONTOUR LEVELS, only the contours are shown.
 - The viewing angle is set initially so that you are viewing the contours by looking down the z axis. You can change the viewing angle as necessary.
 - The graph is shown in expanded view. To switch between expanded and normal view, press \square .
 - The Labels format is set to OFF automatically.
- For WIRE AND CONTOUR, the contours are drawn on a wire frame view. The viewing angle, view (expanded or normal), and Labels format retain their previous settings.

Notes:

- From the Graph screen, you can switch from one graph format style to the next (skipping IMPLICIT PLOT) by pressing:
F
- Pressing:
F
to select CONTOUR LEVELS does not affect the viewing angle, view, or Labels

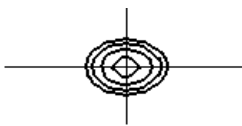
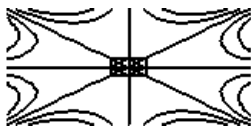
format as it does if you use:



Style	$z1(x,y)=(x^3y-y^3x) / 390$	$z1(x,y)=x^2+.5y^2-5$
-------	-----------------------------	-----------------------

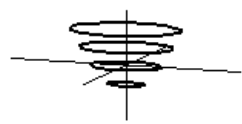
Looking down z axis

CONTOUR
LEVELS

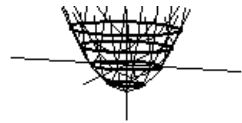


Using eyeθ=20, eyeφ=70, eyeψ=0

CONTOUR
LEVELS




WIRE AND
CONTOUR



Note: These examples use the same x, y, and z Window variable values as a **ZoomStd** viewing cube. If you use **ZoomStd**, press Z to look down the z axis. Do not confuse the contours with the grid lines. The contours are darker.

How Are Z Values Determined?

You can set the ncontour Window variable ( [WINDOW]) to specify the number of contours that will be evenly distributed along the displayed range of z values, where:

$$\text{increment} = \frac{z_{\text{max}} - z_{\text{min}}}{n_{\text{contour}} + 1}$$

The z values for the contours are:

$z_{\text{min}} + \text{increment}$
 $z_{\text{min}} + 2(\text{increment})$
 $z_{\text{min}} + 3(\text{increment})$
 \vdots
 $z_{\text{min}} + n_{\text{contour}}(\text{increment})$

```
eyeθ=20.  
eyeφ=70.  
eyeψ=0.  
xmin=-10.  
xmax=10.  
xgrid=14.  
ymin=-10.  
ymax=10.  
ygrid=14.  
zmin=-10.  
zmax=10.  
ncontour=5.
```

The default is 5. You can set this to 0 through 20.

If **ncontour=5** and you use the standard viewing window (**zmin=-10** and **zmax=10**), the increment is 3.333. Five contours are drawn for $z=-6.666$, -3.333 , 0 , 3.333 , and 6.666 .

Note, however, that a contour is not drawn for a z value if the 3D graph is not defined at that z value.

Drawing a Contour for the Z Value of a Selected Point Interactively

If a contour graph is currently displayed, you can specify a point on the graph and draw a contour for the corresponding z value.

1. To display the **Draw** menu, press:

F6



2. Select **7:Draw Contour**.

3. Either:

- Type the point's x value and press **ENTER**, and then type the y value and press **ENTER**.
– or –
- Move the cursor to the applicable point. (The cursor moves along the grid lines.) Then press **ENTER**.

For example, suppose the current graph is $z_1(x,y)=x^2+.5y^2-5$. If you specify $x=2$ and $y=3$, a contour is drawn for $z=3.5$.


Note: Any existing contours remain on the graph. To remove the default contours, display the Window editor (**◀** **[WINDOW]**) and set **ncontour=0**.

Drawing Contours for Specified Z Values

From the Graph screen, display the Draw menu and then select **8:DrwCtour**. The Home screen is displayed automatically with **DrwCtour** in the entry line. You can then specify one or more z values individually or generate a sequence of z values.

Some examples are:

DrwCtour 5	Draws a contour for $z=5$.
DrwCtour {1,2,3}	Draws contours for $z=1, 2, \text{ and } 3$.
DrwCtour seq(n,n,-10,10,2)	Draws contours for a sequence of z values from -10 through 10 in steps of 2 ($-10, -8, -6, \text{ etc.}$).

Note: To remove the default contours, use  [WINDOW] and set **ncontour=0**.

The specified contours are drawn on the current 3D graph. (A contour is not drawn if the specified z value is outside the viewing cube or if the 3D graph is not defined at that z value.)

Notes about Contour Plots

For a contour plot:

- You can use the cursor keys to animate the contour plot.
- You cannot trace (**F3**) the contours themselves. However, you can trace the wire frame as seen when **Style=WIRE AND CONTOUR**.
- It may take awhile to evaluate the equation initially.

- Because of possible long evaluation times, you first may want to experiment with your 3D equation by using Style=WIRE FRAME. The evaluation time is much shorter. Then, after you're sure you have the correct Window variable values, display the Graph Formats dialog box and set Style=CONTOUR LEVELS or WIRE AND CONTOUR.



Example: Contours of a Complex Modulus Surface

The complex modulus surface given by $z(a,b) = \text{abs}(f(a+bi))$ shows all the complex zeros of any polynomial $y=f(x)$.

Example

In this example, let $f(x)=x^3+1$. By substituting the general complex form $x+yi$ for x , you can express the complex surface equation as $z(x,y)=\text{abs}((x+y i)^3+1)$.

- Use **MODE** to set **Graph=3D**.
- Press **[Y=]**, and define the equation:

$$z1(x,y)=\text{abs}((x+y*i)^3+1)$$



- Press **[WINDOW]**, and set the Window variables as shown.

```

eyeθ=-90.
eyeφ=0.
eyeψ=0.
xmin=-1.5
xmax=1.5
xgrid=14.
ymin=-1.5
ymax=1.5
ygrid=14.
zmin=-1.
zmax=2.
ncontour=10.

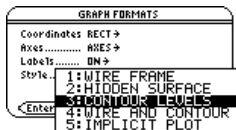
```

4. Display the Graph Formats dialog box:

◆ F

Turn on the axes, set

Style = CONTOUR LEVELS, and return to the Window editor.



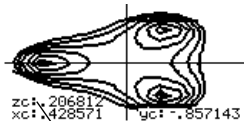
5. Press \square [GRAPH] to graph the equation.

It will take awhile to evaluate the graph; so be patient. When the graph is displayed, the complex modulus surface touches the xy plane at exactly the complex zeros of the polynomial:

$$-1, \frac{1}{2} + \frac{\sqrt{3}}{2}i, \text{ and } \frac{1}{2} - \frac{\sqrt{3}}{2}i$$

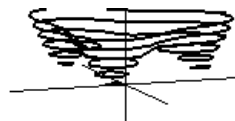
6. Press \square [F3], and move the trace cursor to the zero in the fourth quadrant.

The coordinates let you estimate $.428 - .857i$ as the zero.



The zero is precise when $z=0$.

7. Press \square [ESC]. Then use the cursor keys to animate the graph and view it from different eye angles.



This example shows $\text{eye}\theta=70$, $\text{eye}\phi=70$, and $\text{eye}\psi=0$.

Notes:

- For more accurate estimates, increase the **xgrid** and **ygrid** Window variables. However, this increases the graph evaluation time.
- When you animate the graph, the screen changes to normal view. Use \boxtimes to toggle between normal and expanded views.

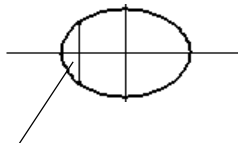
Implicit Plots

An implicit plot is used primarily as a way to graph 2D implicit forms that cannot be graphed in function graphing mode. Technically, an implicit plot is a 3D contour plot with a single contour drawn for $z=0$ only.

Explicit and Implicit Forms

In 2D function graphing mode, equations have an explicit form $y=f(x)$, where y is unique for each value of x .

Many equations, however, have an implicit form $f(x,y)=g(x,y)$, where you cannot explicitly solve for y in terms of x or for x in terms of y .



y is not unique for each x , so you cannot graph this in function graphing mode.

By using implicit plots in 3D graphing mode, you can graph these implicit forms without solving for y or x .

Rearrange the implicit form as an equation set to zero.

$$f(x,y)-g(x,y)=0$$

In the Y= Editor, enter the non-zero side of the equation. This is valid because an implicit plot automatically sets the equation equal to zero.

$$z1(x,y)=f(x,y)-g(x,y)$$

For example, given the ellipse equation shown to the right, enter the implicit form in the Y= Editor.

$$\begin{aligned} &\text{If } x^2+.5y^2=30, \\ &\text{then } z1(x,y)=x^2+.5y^2-30. \end{aligned}$$

Notes: You can also graph many implicit forms if you either:

- Express them as parametric equations.
- Break them into separate, explicit functions.

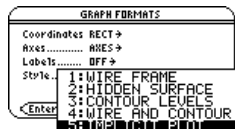
Selecting the Graph Format Style

In 3D graphing mode, define an appropriate equation and graph it as you would any 3D equation, with the following exception. Display the GRAPH FORMATS dialog box from

the Y= Editor, Window editor, or Graph screen:

 F

and then set **Style = IMPLICIT PLOT**.




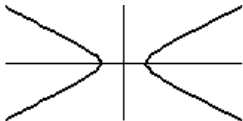

Note: From the Graph screen, you can switch to the other graph format styles by pressing:

F

However, to return to IMPLICIT PLOT press:

 F

- The viewing angle is set initially so that you are viewing the plot by looking down the z axis. You can change the viewing angle as necessary.
- The plot is shown in expanded view. To switch between expanded and normal view, press .
- The Labels format is set to OFF automatically.

Style	$x^2 - y^2 = 4$ $z1(x,y) = x^2 - y^2 - 4$	$\sin(x) + \cos(y) = e(x*y)$ $z1(x,y) = \sin(x) + \cos(y) - e(x*y)$
IMPLICIT PLOT		

Note: These examples use the same x, y, and z Window variable values as a **ZoomStd** viewing cube. If you use **ZoomStd**, press Z to look down the z axis.

Notes About Implicit Plots

For an implicit plot:

- The **ncontour** Window variable has no affect. Only the **z=0** contour is drawn, regardless of the value of ncontour. The displayed plot shows where the implicit form intersects the xy plane.
- You can use the cursor keys to animate the plot.
- You cannot trace (**F3**) the implicit plot itself. However, you can trace the unseen wire frame graph of the 3D equation.
- It may take awhile to evaluate the equation initially.
- Because of possible long evaluation times, you first may want to experiment with your 3D equation by using **Style=WIRE FRAME**. The evaluation time is much shorter. Then, after you're sure you have the correct Window variable values, set **Style=IMPLICIT PLOT**.



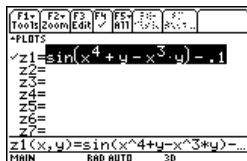
Example: Implicit Plot of a More Complicated Equation

You can use the **IMPLICIT PLOT** graph format style to plot and animate a complicated equation that cannot be graphed otherwise. Although it may take a long time to evaluate such a graph, the visual results can justify the time required.

Example

Graph the equation $\sin(x^4+y-x^3y) = .1$.

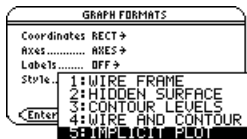
1. Use **MODE** to set **Graph=3D**.
2. Press \diamond [Y=], and define the equation:
 $z1(x,y)=\sin(x^4+y-x^3y)-.1$



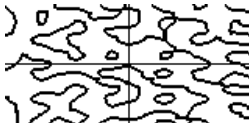
3. Press \diamond [WINDOW], and set the Window variables as shown.

```
eyeθ=-90.  
eyeφ=0.  
eyeψ=0.  
xmin=-10.  
xmax=10.  
xgrid=14.  
ymin=-10.  
ymax=10.  
ygrid=14.  
zmin=-10.  
zmax=10.  
ncontour=5.
```

4. Press:
 \diamond **F**
Turn on the axes, set **Style = IMPLICIT PLOT**, and return to the Window editor.



5. Press \diamond [GRAPH] to graph the equation.
It will take awhile to evaluate the graph; so be patient.



The graph shows where $\sin(x^4+y-x^3y) = .1$

6. Use the cursor keys to animate the graph and view it from different eye angles.



Note: For more detail, increase the **xgrid** and **ygrid** Window variables. However, this increases the graph evaluation time.

In expanded view, this example shows $\text{eye}\theta = -127.85$, $\text{eye}\phi = 52.86$, and $\text{eye}\psi = -18.26$.

Note: When you animate the graph, the screen changes to normal view. Press \square to switch between normal and expanded views.

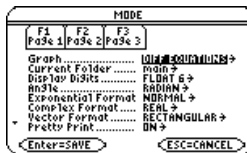
Differential Equation Graphing

Overview of Steps in Graphing Differential Equations

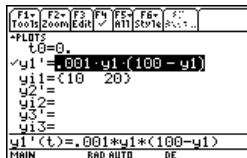
To graph differential equations, use the same general steps used for $y(x)$ functions as described in *Basic Function Graphing*. Any differences are described on the following pages.

Graphing Differential Equations

1. Set **Graph** mode ($\boxed{\text{MODE}}$) to **DIFF EQUATIONS**. Also set **Angle** mode, if necessary.



2. Define equations and, optionally, initial conditions on Y= Editor ($\boxed{\blacklozenge}$ [Y=]).
3. Select ($\boxed{\text{F4}}$) which defined functions to graph.



Note: To turn off any stat data plots, press $\boxed{\text{F5}}$ 5 or use $\boxed{\text{F4}}$ to deselect them.

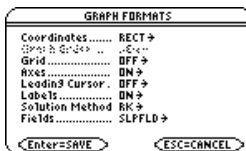
4. Set the display style for a function.



5. Set the graph format. **Solution Method** and **Fields** are unique to differential equations.

F1 9

— or —



Note: The **Fields** format is critical, depending on the order of the equation.

6. Set the axes as applicable, depending on the Fields format.




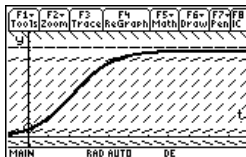
Note: Valid **Axes** settings depend on the Fields format.

7. Define the viewing window (**F2** [WINDOW]).

Note: Depending on the **Solution Method** and **Fields** formats, different Window variables are displayed. **F2** **Zoom** also changes the viewing window.

```
t0=0.
tmax=10.
tstep=.1
tplot=0.
xmin=-10.
xmax=110.
xsc1=10.
ymin=-10.
ymax=120.
ysc1=10.
ncurves=0.
dftol=.001
fldres=20.
```

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



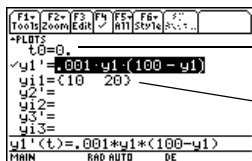
Differences in Diff Equations and Function Graphing

This module assumes that you already know how to graph $y(x)$ functions as described in *Basic Function Graphing*. This section describes the differences.

Setting the Graph Mode

Use **MODE** to set **Graph = DIFF EQUATIONS** before you define differential equations or set Window variables. The Y= Editor and the Window Editor let you enter information for the current Graph mode setting only.

Defining Differential Equations on the Y= Editor



Use t_0 to specify when initial conditions occur. You can also set t_0 in the Window Editor.

Use y_i to specify one or more initial conditions for the corresponding differential equation. You can define differential equations $y_1'(t)$ through $y_{99}'(t)$.

Note: You can use the **Define** command from the Home screen to define functions and equations.

When entering equations in the Y= Editor, do not use **y(t)** formats to refer to results. For example:

Do not use implied multiplication between a variable and parenthetical expression. If you do, it is treated as a function call.

Enter: $y1' = .001y1*(100-y1)$

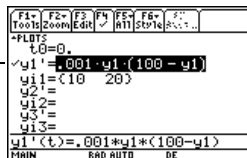
Not: $y1' = .001y1(t)*(100-y1(t))$

Only 1st-order equations can be entered in the Y= Editor. To graph 2nd- or higher-order equations, you must enter them as a system of 1st-order equations.

Detailed information is available on setting initial conditions.

Selecting Differential Equations

You can use **F4** to select a differential equation, but not its initial condition.



Important: Selecting $y1'$ will graph the $y1$ solution curve, not the derivative $y1'$, depending on the axis setting.

Selecting the Display Style

With the Style menu, only the **Line**, **Dot**, **Square**, **Thick**, **Animate**, and **Path** styles are available. **Dot** and **Square** mark only those discrete values (in **tstep** increments) at which a differential equation is plotted.



F6

Setting Graph Formats

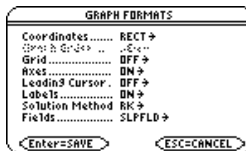
From the Y= Editor, Window Editor, or Graph screen, press:

F1 9

— OR —



F



The formats affected by differential equations are:

Graph format	Description
Graph Order	Not available.
Solution Method	Specifies the method used to solve the differential equations. <ul style="list-style-type: none">• RK — Runge-Kutta method. For information about the algorithm used for this method, refer to the <i>Technical Reference</i> module.• EULER — Euler method.• The method lets you choose either greater accuracy or speed. Typically, RK is more accurate than EULER but takes longer to find the solution.

Graph format	Description
Fields	<p>Specifies whether to draw a field for the differential equation.</p> <ul style="list-style-type: none"> • SLPFLD — Draws a slope field for only one 1st-order equation, with t on the x axis and the solution on the y axis. • DIRFLD — Draws a direction field for only one 2nd-order equation (or system of two 1st-order equations), with axes determined by the custom axes settings. • FLDOFF — Does not display a field. This is valid for equations of any order, but you must use it for 3rd- or higher-order. You must enter the same number of initial conditions for all equations in the $Y=$ Editor.

Important: The Fields graph format is critical in successfully graphing differential equations.

Note: If you press **ENTER** while a slope or direction field is being drawn, the graph pauses after the field is drawn but before the solutions are plotted. Press **ENTER** again to continue. To cancel graphing, press **ON**.

Setting Axes


In the $Y=$ Editor, **Axes** may or may not be available, depending on the current graph format.

If it is available, you can select the axes that are used to graph the differential equations.



Axes	Description
TIME	Plots t on the x axis and y (the solutions to the selected differential equations) on the y axis.
CUSTOM	Lets you select the x and y axes.

Window Variables

Differential equation graphs use the following Window variables. Depending on the **Solution Method** and **Fields** graph formats, not all of these variables are listed in the Window Editor ( [WINDOW]) at the same time.

Variable	Description
t0	Time at which the initial conditions entered in the Y= Editor occur. You can set t0 in the Window Editor and Y= Editor. (If you set t0 in the Y= Editor, tplot is set to the same value automatically.)
tmax, tstep	Used to determine the t values where the equations are plotted: $y'(t_0)$ $y'(t_0+tstep)$ $y'(t_0+2*tstep)$... not to exceed ... $y'(tmax)$ If Fields = SLPFLD , tmax is ignored. Equations are plotted from t0 to both edges of the screen in tstep increments.

Variable	Description
tplot	First t value plotted. If this is not a tstep increment, plotting begins at the next tstep increment. In some situations, the first points evaluated and plotted starting at t0 may not be interesting visually. By setting tplot greater than t0 , you can start the plot at the interesting area, which speeds up the graphing time and avoids unnecessary clutter on the Graph screen.

Note: If $t_{max} < t_0$, **tstep** must be negative. If **Fields=SLPFLD**, **tplot** is ignored and is assumed to be the same as **t0**.

Variable	Description
xmin, xmax, ymin, ymax	Boundaries of the viewing window.
xscl, yscl	Distance between tick marks on the x and y axes.
ncurves	Number of solution curves (0 through 10) that will be drawn automatically if you do not specify an initial condition. By default, ncurves = 0 .

Variable	Description
	<p>When ncurves is used, t0 is set temporarily at the middle of the screen and initial conditions are distributed evenly along the y axis, where:</p> $increment = \frac{ymax - ymin}{ncurves + 1}$ <p>The y values for the initial conditions are: ymin + increment ymin + 2*(increment) ⋮ ymin + ncurves*(increment)</p>
diftol	(Solution Method = RK only) Tolerance used by the RK method to help select a step size for solving the equation; must be $\geq 1E-14$.
fldres	(Fields = SLPFLD or DIRFLD only) Number of columns (1 through 80) used to draw a slope or direction field across the full width of the screen.
Estep	(Solution Method = EULER only) Euler iterations between tstep values; must be an integer >0 . For more accuracy, you can increase Estep without plotting additional points.
dtime	(Fields = DIRFLD only) Point in time at which a direction field is drawn.

Standard values (set when you select **6:ZoomStd** from the $\boxed{F2}$ **Zoom** toolbar menu) are:

$t0 = 0.$	$xmin = -1.$	$ymin = -10.$	$ncurves = 0.$
$tmax = 10.$	$xmax = 10.$	$ymax = 10.$	$difftol = .001$
$tstep = .1$	$xscl = 1.$	$yscl = 1.$	$Estep = 1.$
$tplot = 0.$			$fldres = 14.$
			$dtime = 0.$

You may need to change the standard values for the t variables to ensure that sufficient points are plotted.






The **fldpic** System Variable

When a slope or direction field is drawn, a picture of the field is stored automatically to a system variable named **fldpic**. If you perform an operation that regraphs the plotted equations but does not affect the field, the Voyage™ 200 Graphing Calculator reuses the picture in **fldpic** instead of having to redraw the field. This can speed up the regraphing time significantly.

fldpic is deleted automatically when you exit the differential equation graphing mode or when you display a graph with **Fields = FLDOFF**.

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.

Tool	For Differential Equation Graphs:
Free-Moving Cursor	Works just as it does for function graphs.
F2 Zoom	Works just as it does for function graphs. <ul style="list-style-type: none">• Only x (xmin, xmax, xscl) and y (ymin, ymax, yscl) Window variables are affected.• The t Window variables (t0, tmax, tstep, tplot) are not affected unless you select 6:ZoomStd (which sets all Window variables to their standard values).
F3 Trace	Lets you move the cursor along the curve one tstep at a time. To move approximately ten plotted points at a time, press 2nd  or 2nd  . <p>If you enter initial conditions in the Y= Editor or let the ncurves Window variable plot curves automatically, you can trace the curves. If you use:</p> <p> F8</p> <p>IC from the Graph screen to select initial conditions interactively, you cannot trace the curves.</p> <p>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.</p> <p>Use  or  to view results on all plotted curves.</p>

Tool	For Differential Equation Graphs:
------	-----------------------------------

F5 Math

Only **1:Value** is available.

- With TIME axes, the **y(t)** solution value (represented by yc) is displayed for a specified t value.
 - With CUSTOM axes, the values that correspond to x and y depend on the axes you choose.
-

Note: During a trace, you can move the cursor to a particular point by typing a value for t and pressing **ENTER**. You can use QuickCenter at any time during a trace, even if the cursor is still on the screen.

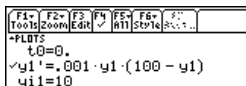
Setting the Initial Conditions

You can enter initial conditions in the Y= Editor, let the Voyage™ 200 Graphing Calculator calculate initial conditions automatically, or select them interactively from the Graph screen.

Entering Initial Conditions in the Y= Editor

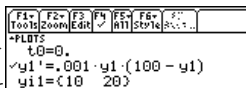
You can specify one or more initial conditions in the Y= Editor. To specify more than one, enter them as a list enclosed in braces $\{ \}$ and separated by commas.

To enter initial conditions for the y_1' equation, use the y_{i1} line, etc.



To specify when the initial conditions occur, use t_0 . This is also the first t evaluated for the graph.

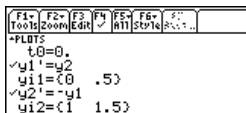
To graph a family of solutions, enter a list of initial conditions.



Enter $\{10,20\}$ even though $\{10\ 20\}$ is displayed.

For a 2nd- or higher-order differential equation, you must define a system of 1st-order equations in the Y= Editor.

If you enter initial conditions, you must enter the same number of initial conditions for each equation in the system. Otherwise, a Dimension error occurs.



If You Do Not Enter an Initial Condition in the Y= Editor

If you do not enter initial conditions, the **ncurves** Window variable (\square [WINDOW]) specifies the number of solution curves graphed automatically. By default, **ncurves** = 0. You can

enter a value from 0 through 10. However, the **Fields** graph format and the **Axes** setting determine whether **ncurves** is used.

If Fields =	Then:
SLPFLD	Uses ncurves , if not set to 0, to graph curves.
DIRFLD	Ignores ncurves . Does not graph any curves.
FLDOFF	Uses ncurves if Axes = TIME (or if Axes = Custom and the x axis is t). Otherwise, a Diff Eq setup error occurs.

When **ncurves** is used, **t0** is set temporarily at the middle of the Graph screen. However, the value of **t0** as set in the Y= Editor or Window Editor is not changed.

Notes:

- Without entering initial conditions, use SLPFLD (with **ncurves=0**) or DIRFLD to display a slope or direction field only.
- SLPFLD is for a single 1st-order equation only. DIRFLD is for a 2nd-order equation (or system of two 1st-order equations) only.

Selecting an Initial Condition Interactively from the Graph Screen

When a differential equation is graphed (regardless of whether a solution curve is displayed), you can select a point on the Graph screen and use it as an initial condition.

If Fields =

Do this:

SLPFLD

Press:

– or –



DIRFLD

Specify an initial condition. Either:

- Move the cursor to the applicable point and press **ENTER**.
– or –
- For each of the two coordinates, type a value and press **ENTER**.
 - For SLPFLD (1st-order only), enter values for **t0** and **y(t0)**.
 - For DIRFLD (2nd-order or system of two 1st-order equations only), enter values for both **y(t0)** initial conditions, where **t0** is the value set in the Y= Editor or Window Editor.

A circle marks the initial condition and the solution curve is drawn.

If Fields =**Do this:**

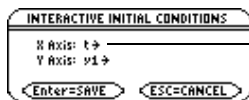
FLDOFF

- Press:



F8

You are prompted to select the axes for which you want to enter initial conditions.



t is a valid selection. It will let you specify a value for t0.

Your selections will be used as the axes for the graph.

- You can accept the defaults or change them. Then press **ENTER**.
 - Specify an initial condition as described for SLPFLD or DIRFLD.
-

Note: With SLPFLD or DIRFLD, you can select initial conditions interactively regardless of whether you enter initial conditions in the Y= Editor. With FLDOFF, you can select initial conditions interactively. However, if three or more equations are entered, you must enter a single value (not a list) as the initial condition for each equation in the Y= Editor. Otherwise, a Dimension error occurs when graphing.

Note about Tracing a Solution Curve

When you enter initial conditions in the Y= Editor or let **ncurves** graph solution curves automatically, you can use **F3** to trace the curves. However, you cannot trace a curve drawn by selecting an initial condition interactively. These curves are drawn, not plotted.

Defining a System for Higher-Order Equations

In the Y= Editor, you must enter all differential equations as 1st-order equations. If you have an nth-order equation, you must transform it into a system of n 1st-order equations.

Transforming an Equation into a 1st-Order System

A system of equations can be defined in various ways, but the following is a general method.

1. Rewrite the original differential equation as necessary.

$$y'' + y' + y = e^x$$

- a) Solve for the highest-ordered derivative.

$$y'' = e^x - y' - y$$

- b) Express it in terms of y and t.

$$y'' = e^t - y' - y$$

- c) On the right side of the equation only, substitute to eliminate any references to derivative values.

Note: To produce a 1st-order equation, the right side must contain non-derivative variables only.

In place of:	Substitute:
y	y_1
y'	y_2
y''	y_3
y'''	y_4
$y^{(4)}$	y_5
\vdots	\vdots

$$y'' = e^t - y_2 - y_1$$

Do not substitute on the left side at this time.

- d) On the left side of the equation, substitute for the derivative value as shown below.

In place of:	Substitute:
y'	y_1'
y''	y_2'
y'''	y_3'
$y^{(4)}$	y_4'
\vdots	\vdots

$$y_2' = e^t - y_2 - y_1$$

2. On the applicable lines in the Y= Editor, define the system of equations as:

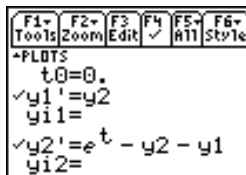
$$y1' = y2$$

$$y2' = y3$$

$$y3' = y4$$

– up to –

y_n' = your nth-order equation



Note: Based on the above substitutions, the y' lines in the Y= Editor represent:

$$y1' = y'$$

$$y2' = y''$$

etc.

Therefore, this example's 2nd-order equation is entered on the $y2'$ line.

In a system such as this, the solution to the $y1'$ equation is the solution to the nth-order equation. You may want to deselect any other equations in the system.

Example of a 2nd-Order Equation

The 2nd-order differential equation $y'' + y = 0$ represents a simple harmonic oscillator. Transform this into a system of equations for the Y= Editor. Then, graph the solution for initial conditions $y(0) = 0$ and $y'(0) = 1$.

Example

1. Press **MODE** and set **Graph=DIFF EQUATIONS**.

2. Define a system of equations for the 2nd-order equation.

Rewrite the equation and make the necessary substitutions.

$$\begin{aligned}y'' + y &= 0 \\ y'' &= -y \\ y'' &= -y1 \\ y2' &= -y1\end{aligned}$$

3. In the Y= Editor ($\square \blacktriangleright [Y=]$), enter the system of equations.

4. Enter the initial conditions:
yi1=0 and **yi2=1**

Note: **t0** is the time at which the initial conditions occur. It is also the first **t** evaluated for the graph. By default, **t0=0**.

yi1 is the initial condition for y(0).



yi2 is the initial condition for y'(0).

5. Press:

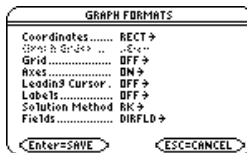
F1 **9**

— or —



F

and set **Axes = ON**, **Labels = OFF**,
Solution Method = RK, and
Fields = DIRFLD.



Important: For 2nd-order equations, you must set **Fields=DIRFLD** or **FLDOFF**.

6. In the Y= Editor, press:



and make sure **Axes = CUSTOM** with **y1** and **y2** as the axes.



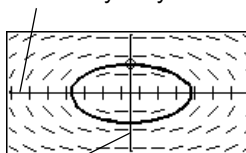
Important: Fields=DIRFLD cannot plot a time axis. An **Invalid Axes** error occurs if **Axes=TIME** or if t is set as a **CUSTOM** axis.

7. In the Window Editor (\blacklozenge [WINDOW]), set the Window variables.

t0=0	xmin=-2	ncurves=0
tmax=10	xmax=2	diftol=.001
tstep=.1	xscl=1	fidres=14
tplot=0	ymin=-2	dtime=0
	ymax=2	
	yscl=1	

8. Display the Graph screen (\blacklozenge [GRAPH]).

x axis = y1 = y



y axis = y2 = y'

If you select **ZoomSqr** (F2 5), you can see that the phase-plane orbit is actually a circle. However, **ZoomSqr** will change your Window variables.

To examine this harmonic oscillator in more detail, use a split screen to graph the manner in which y and y' change with respect to time (t).

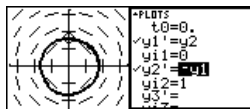
9. Press **MODE** and change the mode settings on **Page 2** as shown. Then close the **MODE** dialog box, which redraws the graph.

Note: To display different graphs in both parts of a split screen, you must use the **2-graph** mode.



10. Press **2nd** **[+]** to switch to the right side of the split screen.
11. Use **F4** to select y_1' and y_2' .

The right side uses the same equations as the left side. However, no equations are selected initially in the right side.



12. Press:

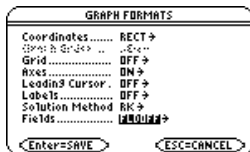
F1 **9**

— or —



F

Set **Fields = FLDOFF**.



Important: Because **Fields=DIRFLD** cannot plot a time axis, you must change the **Fields** setting. **FLDOFF** turns off all fields.

13. In the **Y= Editor**, press:



F7

and make sure **Axes = TIME**.



14. In the Window Editor, change y_{\min} and y_{\max} as shown to the right.

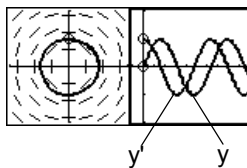
$$y_{\min} = -2.$$

$$y_{\max} = 2.$$

Note: When you enter **2-graph** mode, Window variables for the right side are set to their defaults.

15. Press \square [GRAPH] to display the Graph screen for graph #2.

The left side shows the phase-plane orbit. The right side shows the solution curve and its derivative.



16. To return to a full screen of the original graph, press \square [2nd] \square [F1] to switch to the left side. Then press \square [MODE] and change the **Split Screen** setting.

Split Screen = FULL

Example of a 3rd-Order Equation

For the 3rd-order differential equation $y''' + 2y'' + 2y' + y = \sin(x)$, write a system of equations to enter in the Y= Editor. Then graph the solution as a function of time. Use initial conditions $y(0) = 0$, $y'(0) = 1$, and $y''(0) = 1$.

Example

1. Press **MODE** and set **Graph=DIFF EQUATIONS**.

2. Define a system of equations for the 3rd-order equation.

Rewrite the equation and make the necessary substitutions.

$$y''' + 2y'' + 2y' + y = \sin(x)$$

$$y''' = \sin(x) - 2y'' - 2y' - y$$

$$y''' = \sin(t) - 2y'' - 2y' - y$$

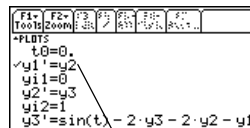
$$y''' = \sin(t) - 2y_3 - 2y_2 - y_1$$

$$y_3' = \sin(t) - 2y_3 - 2y_2 - y_1$$

3. In the Y= Editor (**Y=**), enter the system of equations.

4. Enter the initial conditions:
y1=0, y2=1, and y3=1

Note: **t0** is the time at which the initial conditions occur. By default, **t0=0**.



Important: The solution to the **y1'** equation is the solution to the 3rd-order equation.

5. Be sure that only **y1'** is selected. Use **F4** to deselect any other equations.

6. Press:

[F1] 9

— or —



[F]

Set **Axes = ON**, **Labels = ON**,
Solution Method = RK, and
Fields = FLDOFF.

Important: For 3rd- or higher-order equations, you must set **Fields=FLDOFF**. Otherwise, an **Undefined variable** error occurs when graphing.

7. In the Y= Editor, press:



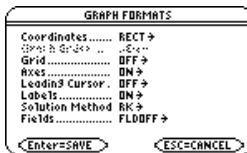
[F7]

Set **Axes = TIME**.


Note: With **Axes=TIME**, the solution to the selected equation is plotted against time (t).

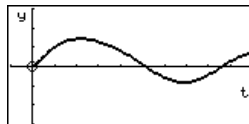
8. In the Window Editor (**[F2] [WINDOW]**), set the Window variables.

t0=0	xmin=-1	ncurves=0
tmax=10	xmax=10	diftol=.001
tstep=.1	xscl=1.	
tplot=0	ymin=-3	
	ymax=3	
	yscl=1	



9. Display the Graph screen ( [GRAPH]).

Note: To find the solution at a particular time, use  to trace the graph.



Setting Axes for Time or Custom Plots

Setting the axes can give you great flexibility in graphing differential equations. Custom axes are particularly effective for showing different kinds of relationships.

Displaying the AXES Dialog Box

From the Y= Editor, press:

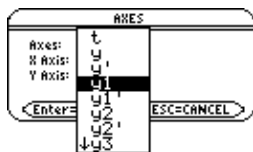


If **Fields = SLPFLD**, **Axes** is unavailable.



Item	Description
Axes	TIME — Plots t on the x axis and y (solutions to all selected differential equations) on the y axis. CUSTOM — Lets you select the x and y axes.

Item	Description
X Axis, Y Axis	Active only when Axes = CUSTOM, these let you select what you want to plot on the x and y axes.



t — time

y — solutions (y_1 , y_2 , etc.) of all selected differential equations

y' — values of all selected differential equations (y_1' , y_2' , etc.)

y_1 , y_2 , etc. — the solution to the corresponding differential equation, regardless of whether that equation is selected

y_1' , y_2' , etc. — the value of the right-hand side of the corresponding differential equation, regardless of whether that equation is selected

Note: t is not valid for either Axis when **Fields=DIRFLD**. If you select t, an Invalid axes error occurs when graphing.

Example of Time and Custom Axes

Using the predator-prey model from biology, determine the numbers of rabbits and foxes that maintain population equilibrium in a certain region. Graph the solution using both time and custom axes.

Predator-Prey Model

Use the two coupled 1st-order differential equations:

$$y_1' = -y_1 + 0.1y_1 \cdot y_2 \quad \text{and} \quad y_2' = 3y_2 - y_1 \cdot y_2$$


where:

y_1 = Population of foxes

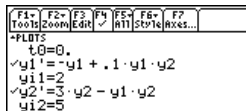
y_{i1} = Initial population of foxes (2)

y_2 = Population of rabbits

y_{i2} = Initial population of rabbits (5)

1. Use **MODE** to set **Graph = DIFF EQUATIONS**.
2. In the Y= Editor ( [Y=]), define the differential equations and enter the initial conditions.

Note: To speed up graphing times, clear any other equations in the Y= Editor. With **FLDOFF**, all equations are evaluated even if they are not selected.



```
F1- F2- F3 F4 F5 F6+ F7  
[Loc] [Zoom] [Edit] [Rt] [Sto] [Recs...]  
*FLBTS  
t0=0.  
✓y1'=-y1+.1·y1·y2  
y11=2  
✓y2'=3·y2-y1·y2  
y12=5
```

3. Press:

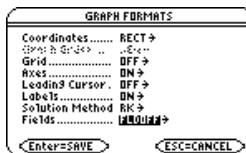
[F1] 9

— or —



◆ F

Set Axes = ON, Labels = ON,
Solution Method = RK, and
Fields = FLDOFF.



4. In the Y= Editor, press:



[F7]

Set Axes = TIME.



5. In the Window Editor (◆ [WINDOW]), set the Window variables.

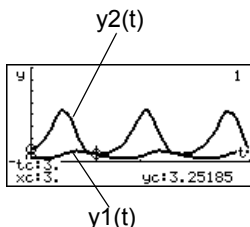
t0=0	xmin=-1	ncurves=0
tmax=10	xmax=10	diftol=.001
tstep= $\pi/24$	xscl=5	
tplot=0	ymin=-10	
	ymax=40	
	yscl=5	

6. Graph the differential equations

(◆ [GRAPH]).

7. Press $\boxed{F3}$ to trace. Then press $3 \boxed{\text{ENTER}}$ to see the number of foxes (y_1) and rabbits (y_2) at $t=3$.

Note: Use \leftarrow and \rightarrow to move the trace cursor between the curves for y_1 and y_2 .



8. Return to the Y= Editor. Press:

$\boxed{F1} 9$

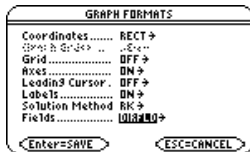
— or —



$\blacklozenge F$

Set Fields = DIRFLD.

Note: In this example, DIRFLD is used for two related differential equations that do not represent a 2nd-order equation.

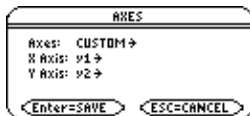


9. Press:

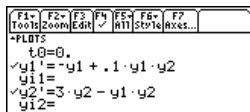


$\boxed{F7}$

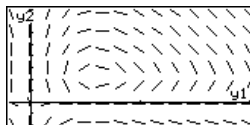
Confirm that the axes are set as shown.



10. In the Y= Editor, clear the initial conditions for y_1 and y_2 .



11. Return to the Graph screen, which displays only the direction field.



12. To graph a family of solutions, return to the Y= Editor and enter the initial conditions shown below.

```

F1- F2- F3 F4 F5- F6- F7
Tools Zoom Edit All Stop Axes...
---FLBT---
t0=0.
✓y1'=-y1+.1·y1·y2
y1={2 6 7}
✓y2'=3·y2-y1·y2
y2={5 12 18}

```

$yi1=\{2,6,7\}$ and $yi2=\{5,12,18\}$

Note: Use a list to specify more than one initial condition.

13. Return to the Graph screen, which displays a curve for each pair of initial conditions.
14. Press **F3** to trace. Then press **3** **ENTER** to see the number of foxes (x_c) and rabbits (y_c) at $t=3$.



Because $t_0=0$ and $t_{max}=10$, you can trace in the range $0 \leq t \leq 10$.

Note: Use \odot and \ominus to move the trace cursor from one initial condition curve to another.

Example Comparison of RK and Euler

Consider a logistic growth model $dP/dt = .001 * P * (100 - P)$, with the initial condition $P(0) = 10$. Use the **BldData** instruction to compare the graphing points calculated by the RK and Euler solution methods. Then plot those points along with a graph of the equation's exact solution.

Example

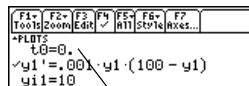
1. Press **MODE** and set **Graph=DIFF EQUATIONS**.
2. Express the 1st-order equation in terms of $y1'$ and $y1$.

$$y1' = .001y1*(100 - y1)$$

Do not use implied multiplication between the variable and parentheses. If you do, it is treated as a function call.

3. Enter the equation in the Y= Editor

(**Y=**).



4. Enter the initial condition:

$$y1=10$$

t_0 is the time at which the initial condition occurs. By default, $t_0=0$.

5. Press:

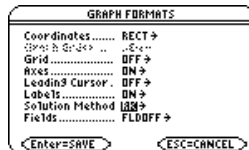
F1 9

— or —



F

Set **Solution Method = RK** and **Fields = FLDOFF**.



Note: To speed up graphing times, clear any other equations in the Y= Editor. With **FLDOFF**, all equations are evaluated even if they are not selected.

6. In the Window Editor ( [WINDOW]), set the Window variables.

t0=0. xmin=-1. ncurves=0.
tmax=100. xmax=100. diftol=.001
① tstep=1. xscl=1.
tplot=0. ymin=-10.
 ymax=10
 yscl=1.

① **Important:** Change **tstep** from .1 (its default) to 1. Otherwise, **BldData** calculates too many rows for the data variable and a Dimension error occurs.

7. In the Home screen ( HOME)



 [CALC HOME]

use **BldData** to create a data variable containing the **RK** graphing points.

BldData rklog

8. Return to the Y= Editor, press:

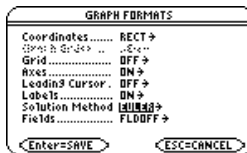
 9

— or —



 F

Set **Solution Method = EULER.**



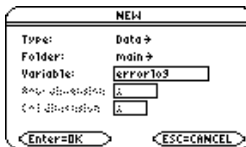
Note: You do not need to graph the equation before using **BldData**. For more information about **BldData**, refer to the *Technical Reference* module.

9. Return to the Home screen, and use **BldData** to create a data variable containing the **Euler** graphing points.

BldData eulerlog

10. Use the **Data/Matrix Editor** (**[APPS]**) to create a new data variable named **errorlog**.

Note: **errorlog** lets you combine the data in **rklog** and **eulerlog** so that you can view the two sets of data side by side.



11. In this new data variable, define the **c1**, **c2**, and **c3** column headers to refer to data in **rklog** and **eulerlog**. Also, enter column titles as shown.

To define a column header, move the cursor to that column, press **[F4]**, type the reference expression (such as **rklog[1]** for **c1**), and press **[ENTER]**.

Note: **rklog[1]** and **rklog[2]** refer to column 1 and 2 in **rklog**, respectively. Likewise with **eulerlog[2]**.

	F1 Tools	F2 Plot Setup	F3 C1) Header	F4 C2) Header	F5 C3) Header	F6 F7 Plot
DATA	time	RK	Euler			
	c1	c2	c3			
1	0.	10.	10.			
2	1.	10.937	10.9			
3	2.	11.949	11.871			
4	3.	13.042	12.917			

c3=eulerlog[2]
MAIN RAD AUTO DE

- ❶ c1=rklog[1] or c1=eulerlog[1]
- ❷ c2=rklog[2]
- ❸ c3= eulerlog[2]

12. In the **Data/Matrix Editor**, press **[F2]**. Then press **[F1]** and define **Plot 1** for the **RK** data, as shown to the right.



13. Define **Plot 2** for the **Euler** data. Use the values shown to the right.

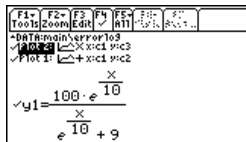
Plot Type=xyline
Mark=Cross
x=c1
y=c3

14. Return to the Y= Editor, press **[MODE]**, and set **Graph = FUNCTION**.

15. The exact solution to the differential equation is given below. Enter it as y1.

$$y1 = (100 * e^{(x/10)}) / (e^{(x/10)} + 9)$$

Note: You can use `deSolve()` to find this exact, general solution. ,



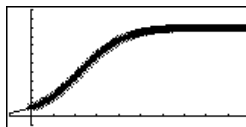
You can use \uparrow to scroll up to see Plot 1 and Plot 2.

16. In the Window Editor, set the Window variables.

$$\begin{array}{lll} \text{xmin}=-10 & \text{ymin}=-10. & \text{xres}=2. \\ \text{xmax}=100 & \text{ymax}=120. & \\ \text{xscl}=10 & \text{yscl}=10. & \end{array}$$

17. Display the Graph screen (\blacklozenge [GRAPH]).

Note: The fuzzy line on the graph indicates differences between the **RK** and **Euler** values.

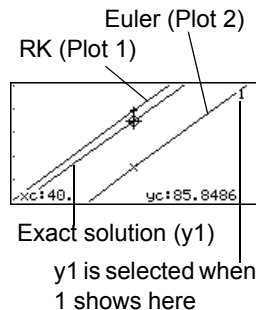


18. In the Window Editor, set the Window variables to zoom in so that you can examine the differences in more detail.

$$\begin{array}{lll} \text{xmin}=39.7 & \text{ymin}=85.5 & \text{xres}=2 \\ \text{xmax}=40.3 & \text{ymax}=86 & \\ \text{xscl}=.1 & \text{yscl}=.1 & \end{array}$$

19. Return to the Graph screen.

20. Press $\boxed{F3}$ to trace, and then press \odot or \ominus until **y1** is selected. (1 shows in upper right corner.) Then enter 40.



By moving the trace cursor to trace each solution to $xc = 40$, you can find that:

- The exact solution (**y1**) is 85.8486, rounded to six digits.
- The **RK** solution (**Plot 1**) is 85.8952.
- The **Euler** solution (**Plot 2**) is 85.6527.

You can also use the **Data/Matrix Editor** to open the errorlog data variable and scroll to **time = 40**.

Example of the `deSolve()` Function

The `deSolve()` function lets you solve many 1st- and 2nd-order ordinary differential equations exactly.

Example

For a general solution, use the following syntax. For a particular solution, refer to the Technical Reference module.

deSolve(*1stOr2ndOrderODE*, *independentVar*, *dependentVar*)

Using the logistic 1st-order differential equation, find the general solution for y with respect to t .

deSolve($y' = 1/1000 y*(100-y)$, t , y)

For ' $'$, type 2nd [$'$].

Do not use implied multiplication between the variable and parentheses. If you do, it will be treated as a function call.

Notes:

- For maximum accuracy, use $1/1000$ instead of $.001$. A floating-point number can introduce round-off errors.
- This example does not involve graphing, so you can use any Graph mode.

Before using **deSolve()**, clear any existing t and y variables. Otherwise, an error occurs.

1. In the Home screen **HOME**



[CALC HOME]

use **deSolve()** to find the general solution.

$$y' = \frac{1}{1000} \cdot y \cdot (100 - \frac{t}{10})$$

$$y = \frac{100 \cdot e^{\frac{t}{10}}}{e^{\frac{t}{10}} + 100 \cdot @1}$$

@1 represents a constant. You may get a different constant (@2, etc.).

2. Use the solution to define a function.
 - a) Press **⇐** to highlight the solution in the history area. Then press **ENTER** to autopaste it into the entry line.
 - b) Insert the **Define** instruction at the beginning of the line. Then press **ENTER**.

$$\text{Define } y = \frac{100 \cdot e^{\frac{t}{10}}}{e^{\frac{t}{10}} + 100 \cdot @1}$$

$$\text{Define } y = 100 * e^{\frac{t}{10}} / (e^{\frac{t}{10}} + 100 * @1)$$

Note: Press **2nd** **↓** to move to the beginning of the entry line.

3. For an initial condition $y=10$ with $t=0$, use **solve()** to find the @1 constant.

Note: If you got a different constant (@2, etc.), solve for that constant.

$$\text{solve}(y=10, @1) | t=0$$

$$@1 = \frac{9}{100}$$

For @, type



⇐ **STO▶**



2nd **R**

4. Evaluate the general solution (y) with the constant $@1=9/100$ to obtain the particular solution shown.

	$\frac{t}{10}$
$y @1 = \frac{9}{100}$	$\frac{100 \cdot e^{\frac{t}{10}}}{e^{\frac{t}{10}} + 9}$
$y @1 = 9 / 100$	
MAIN	RAD AUTO FUNC 4/20

You can also use `deSolve()` to solve this problem directly. Enter:

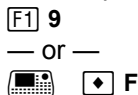
$$\text{deSolve}(y' = 1/1000 y*(100-y) \text{ and } y(0)=10,t,y)$$

Troubleshooting with the Fields Graph Format

If you have difficulties graphing a differential equation, this section can help you correct the problem. Many problems may be related to your Fields graph format setting.

Setting the Fields Graph Format

From the Y= Editor, Window Editor, or Graph screen, press:



What Order Equation Are You Graphing?

If the equation is:

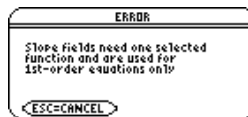
Valid Fields settings are:

1st-order

SLPFLD or FLDOFF

If the equation is:	Valid Fields settings are:
2nd-order (system of two 1st-order equations)	DIRFLD or FLDOFF
3rd- or higher-order (system of three or more 1st-order equations)	FLDOFF

Because **Fields = SLPFLD** is the default setting, a common error message is shown to the right.



When you see this or any other error message:

- For your order of equation, use the previous table to find the valid Fields settings. Change to the applicable setting.
- For a particular Fields setting, check the following for information that applies to that setting.

Fields=SLPFLD

In the
Y= Editor

Use $\boxed{F4}$ to select one and only one 1st-order equation. You can enter mulNotele equations, but only one at a time can be selected.

The selected equation must not refer to any other equation in the Y= Editor. For example:

If $y1'=y2$, an Undefined variable error occurs when you graph.



In the Graph
screen

If the slope field is drawn but no solution curve is plotted,
specify an initial condition.

Fields=DIRFLD

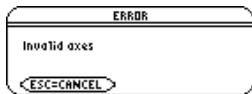
In the
Y= Editor

Enter a valid system of two 1st-order equations. For information about defining a valid system for a 2nd-order equation, refer to Example of a 2nd-Order Equation.

Set **Axes = CUSTOM**:

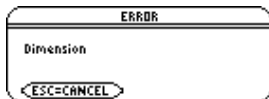


If **Axes = TIME**, an Invalid axes error occurs when you graph.



If you enter initial conditions in the Y= Editor, the equations referenced by the custom axes must have the same number of initial conditions.

Otherwise, a Dimension error occurs when you graph.



With custom
axes

Set axes that are valid for your system of equations. Do not select t for either axis. Otherwise, an Invalid axes error occurs when you graph.

The two axes must refer to different equations in your system of equations. For example, y1 vs. y2 is valid, but y1 vs. y1' gives an Invalid axes error.

In the Graph screen

If the direction field is drawn but no curve is plotted, enter initial conditions in the Y= Editor or select one interactively from the Graph screen. If you did enter initial conditions, select **ZoomFit**:



F2 **A**

The **ncurves** Window variable is ignored with DIRFLD. Default curves are not drawn automatically.

Notes

With DIRFLD, the equations referenced by the custom axes determine which equations are graphed, regardless of which equations are selected in the Y= Editor.

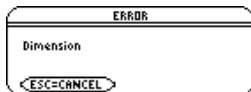
If your system of equations refers to t , the direction field (not the plotted curves) is drawn with respect to one particular time, which is set by the $dtime$ Window variable.

Fields=FLDOFF

In the Y= Editor

If you enter a 2nd- or higher-order equation, enter it as a valid system of equations.

All equations (selected or not) must have the same number of initial conditions. Otherwise, a Dimension error occurs when you graph.

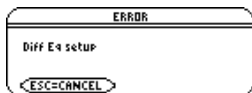


To set **Axes = TIME** or **CUSTOM**, press:



F7

With custom axes If X Axis is not t, you must enter at least one initial condition for each equation in the Y= Editor (whether the equation is selected or not).
Otherwise, a Diff Eq setup error occurs when you graph.



In the Graph screen If no curve is graphed, set an initial condition. If you did enter initial conditions in the Y= Editor, select **ZoomFit**:



F2 **A**

A 1st-order equation may look different with FLDOFF than with SLPFLD. This is because FLDOFF uses the **tplot** and **tmax** Window variables (page 9), which are ignored with SLPFLD.

Notes For 1st-order equations, use FLDOFF and Axes = Custom to plot axes that are not possible with SLPFLD. For example, you can plot t vs. y_1' (where SLPFLD plots t vs. y_1). If you enter mulNote1 1st-order equations, you can plot one equation or its solution vs. another by specifying them as the axes.

If You Use the Table Screen to View Differential Equations

You can use the Table screen to view the points for a differential equation graph. However, the table may show different equations than those graphed. The table shows only the selected equations, regardless of whether those equations will be plotted with your current **Fields** and **Axes** settings

Tables

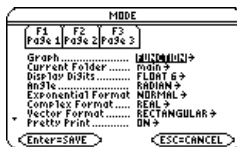
Overview of Steps in Generating a Table

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

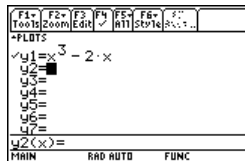
1. Set **Graph** mode and, if necessary, **Angle** mode (**MODE**).

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



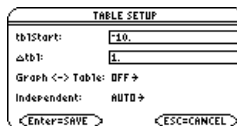
2. Define functions on Y= Editor (**Y=**).
3. Select (**F4**) which defined functions to display in the table.


Note: For information on defining and selecting functions with the Y= Editor, refer to *Basic Function Graphing*.

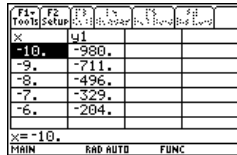


4. Set up the initial table parameters (**TBLSET**).

Note: You can specify an automatic table that is based on initial values or that matches a graph, or a manual (ask) table.



5. Display the table ( [TABLE]).



F1	F2	F3	F4	F5	F6
X	w1				
-10.	-980.				
-9.	-711.				
-8.	-496.				
-7.	-329.				
-6.	-204.				
X=-10.					
MAIN	RAD AUTO				FUNC

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**.

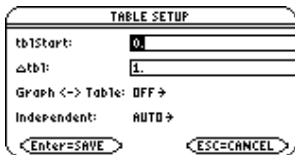


TABLE SETUP

tblStart: 0

Δtbl: 1

Graph <-> Table: OFF

Independent: AUTO

<Enter>=SAVE <ESC>=CANCEL

Setup Parameter Description

tblStart If **Independent = AUTO** and **Graph <-> Table = OFF**, this specifies the starting value for the independent variable.

Δtbl If **Independent = AUTO** and **Graph <-> Table = OFF**, this specifies the incremental value for the independent variable. Δtbl 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 **Δtbl**.
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 and the split screen size.

Setup Parameter	Description
Independent	AUTO — The Voyage™ 200 automatically generates a series of values for the independent variable based on tblStart , Δtbl , and Graph < - > Table . ASK — Lets you build a table manually by entering specific values for the independent variable.

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

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

Note: “–” means that any value entered for this parameter is ignored for the indicated type of table.

In SEQUENCE graphing mode, use integers for **tblStart** and Δtbl .

Changing the Setup Parameters

From the **TABLE SETUP** dialog box:

1. Use \ominus and \oplus 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 \ominus or \oplus to remove the highlighting. Then edit the existing value.
Graph < - > Table or Independent	Press \ominus or \oplus to display a menu of valid settings. Then either: <ul style="list-style-type: none">• Move the cursor to highlight the setting and press ENTER. — or —• Press the number for that setting.

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

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

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 **Δ tbl**. Refer to “Storing and Recalling Variable Values” in *Operating the Calculator*.

- Set **Graph < - > Table** and Independent by using the **setTable** function. Refer to the *Technical Reference* module.

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 Δtbl .

Before You Begin

Define and select the applicable functions on the Y= Editor (\blacklozenge [Y=]). This example uses $y_1(x) = x^3 - x/3$.

Then enter the initial table parameters (\blacklozenge [TBLSET]).

TABLE SETUP	
tblStart:	1.
Δtbl :	.1
Graph <-> Table:	OFF \rightarrow
Independent:	AUTO \rightarrow
Enter=SAVE ESC=CANCEL	

Displaying the Table Screen

To display the Table screen, press \blacklozenge [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.

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).

F1 Tools	F2 Setup	F3 Header	F4 Func	F5 Draw	F6 Table
x	y1				
1.	.66667				
1.1	.96433				
1.2	1.328				
1.3	1.7637				
1.4	2.2773				
y1(x)=.666666666666667					
MAIN		RAD AUTO		FUNC	

Entry line shows full value of highlighted cell.

Note: You can scroll back from the starting value by pressing \leftarrow or $\boxed{2nd} \leftarrow$.

To move the cursor:

Press:

One cell at a time

\leftarrow , \rightarrow , \uparrow , or \downarrow

One page at a time

$\boxed{2nd}$ and then \leftarrow , \rightarrow , \uparrow , or \downarrow

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

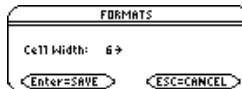
- When you scroll down or up, the variable and function names are always visible across the top of the screen.
- When you scroll right or left, the values of the independent variable are always visible along the left side of the screen.


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.

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

To change the cell width from the **Table** screen:



1. Press **[F1] 9**
— or —
 **[F]**
2. Press **[Down Arrow]** or **[Up Arrow]** 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

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.

Notes:

- If a function is undefined at a particular value, undef is displayed in the cell.
- Use `MODE` to set the display modes.

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.

Full Precision	If cell width is:			
	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

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

If Results are 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.

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

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.

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

4. Press **[ENTER]** to save the edited function and update the table. The edited function is also saved in the Y= Editor.

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.

Building a Manual (Ask) Table

If **Independent = ASK** on the TABLE SETUP dialog box, the Voyage™ 200 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]**.

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).

F1 Tools	F2 Setup	F3 Cell	F4 Header	F5 Del Row	F6 Ins Row
x	y1				

$x=$ MAIN RAD AUTO FUNC

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 (row 1, row 3).
 - 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.

Note: 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.

Enter values in any numerical order.

Enter a new value here.

Shows full value of highlighted cell.

F1 Tools	F2 Setup	F3 Header	F4 Del Row	F5 Ins Row	F6
x	41				
1.	.66667				
8.	509.33				
3.2	31.701				
22.	10641.				
12.6	1996.2				
u1(x)=10640.666666667					
MAIN		RAD AUTO		FUNC	

Note: In this example, you can move the cursor to highlight any cell in the independent variable column, but you can enter values in column 1 only.

Entering a List in the Independent Variable Column

1. Move the cursor to highlight any cell in the independent variable column.
2. Press $\boxed{F4}$ 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.

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

4. Press $\boxed{\text{ENTER}}$ 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.

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.

Additional Graphing Topics

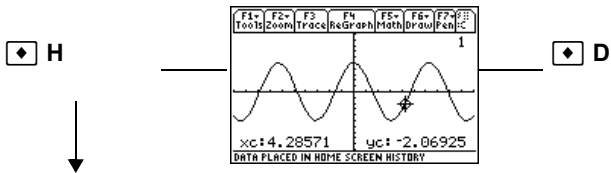
Collecting Data Points from a Graph

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 $y_1(x)=5*\cos(x)$.)
2. Display the coordinates or math results you want to collect.
3. Save the information to the Home screen or the *sysData* variable. \blacklozenge **H** (Home screen) or \blacklozenge **D** (*sysData* variable)
4. Repeat the process as necessary.

Note: To display coordinates or math results, trace a function with $\boxed{F3}$ or perform an $\boxed{F5}$ **Math** operation (such as **Minimum** or **Maximum**). You can also use the free-moving cursor.



Displayed coordinates are added to the Home screen's history area (but not the entry line) as a single-row matrix or vector.

Displayed coordinates are stored in a data variable named *sysData*, which you can open in the Data/Matrix Editor.

F1	F2	F3	F4	F5	F6	F7	F8
Tools	1/3	Calc	Pr	9	MIO	Clean	Up
■ [1.93277310924 -1.770618]▶ [1.93277 -1.77062]							
■ [3.10924369748 -4.997384]▶ [3.10924 -4.99738]							
■ [4.28571428571 -2.069225]▶ [4.28571 -2.06923]							
MAIN RAD AUTO FUNC 3/30							

F1	F2	F3	F4	F5	F6	F7	F8
Tools	Plot	Setup	Cat	Header	Calc	Util	Stat
DATA							
	x1	x2					
1	1.9328	-1.771					
2	3.1092	-4.997					
3	4.2857	-2.069					
4							
r3c1=4.28571428571							
MAIN RAD AUTO FUNC							

Note: 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*, this command 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
Differential Equation	t

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.

Method	Description
Copy and paste	<ol style="list-style-type: none">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]. <p>Note: Instead of using [F1] 5 or [F1] 6 to copy and paste, use: [C] (copy) or [V] (paste)</p>
[STO▶]	<p>Store the expression to a Y= function name.</p> <p>$2x^3+3x^2-4x+12 \rightarrow y1(x)$</p> <p>_____ Use the complete function name: $y1(x)$, not just $y1$.</p> <p>Note: To copy an expression from the Home screen's history area to the entry line, use the auto-paste feature or copy and paste.</p>
Define command	<p>Define the expression as a user-defined Y= function.</p> <p>Define $y1(x)=2x^3+3x^2-4x+12$</p> <p>Note: Define is available from the Home screen's [F4] toolbar menu.</p>

Method	Description
$\boxed{2\text{nd}}$ $\boxed{\text{RCL}}$	<p>If the expression is already stored to a variable:</p> <ol style="list-style-type: none"> 1. Display the Y= Editor, highlight the desired function, and press $\boxed{\text{ENTER}}$. 2. Press $\boxed{2\text{nd}}$ $\boxed{\text{RCL}}$. Type the variable name that contains the expression, and press $\boxed{\text{ENTER}}$ twice. <p>Important: To recall a function variable such as f1(x), type only f1, not the full function name.</p> <ol style="list-style-type: none"> 3. Press $\boxed{\text{ENTER}}$ to save the recalled expression in the Y= Editor's function list. <p>Note: $\boxed{2\text{nd}}$ $\boxed{\text{RCL}}$ 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).</p>

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.

If the expression is in terms of:

Use the **Graph** command as shown in this example:

The native independent variable

$\boxed{\text{Graph } 1.25x * \cos(x)}$

For function graphing, x is the native variable.

If the expression is in terms of:

Use the **Graph** command as shown in this example:

A non-native independent variable

Graph 1.25a*cos(a),a

Specify the independent variable; otherwise, you may get an error.

Note: **Graph** uses the current Window variable settings and is available from the Home screen's **F4** toolbar menu.

Graph does not work with sequence graphs or differential equations. For parametric, polar, and 3D graphs, use the following variations.

In PARAMETRIC graphing mode:

Graph $xExpr, yExpr, t$

In POLAR graphing mode:

Graph $expr, \theta$

In 3D graphing mode:

Graph $expr, x, y$

Note: 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 copy the expression to the Y= Editor. Instead, it temporarily suspends any functions selected on the Y= Editor. You can trace, zoom, or show and edit **Graph** expressions on the Table screen, just the same as Y= Editor functions.

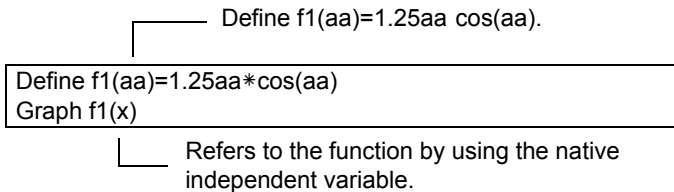
Clearing the Graph Screen

Each time you execute **Graph**, the new expression is added to the existing ones. To clear the graphs:

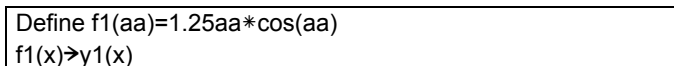
- Execute the **ClrGraph** command (available from the Home screen's **F4** **Other** toolbar menu).
 - or –
- Display the Y= Editor. The next time you display the Graph screen, it will use the functions selected on the Y= Editor.

Extra Benefits of User-Defined Functions

You can define a user-defined function in terms of any independent variable. For example:



and:



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.

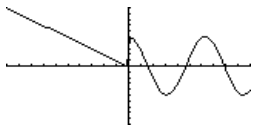
Using the When Function

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:
$x < 0$	$-x$
$x \geq 0$	$5 \cos(x)$



In the Y= Editor:

The function is “pretty printed” in this form.

Enter the function in this form.

```
*FLOTS
✓y1= { -x, x < 0
      { 5*cos(x), else
y2=
y3=
y4=
y5=
y6=
y1(x)=when(x<0, -x, 5*cos(x...
```

For three or more pieces, you can use nested **when** functions.

Note: To enter **when**, type it or use the **CATALOG**.

When:	Use expression:
--------------	------------------------

$$x < -\pi$$

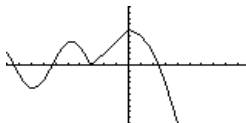
$$4 \sin(x)$$

$$x \geq -\pi \text{ and } x < 0$$

$$2x + 6$$

$$x \geq 0$$

$$6 - x^2$$



In the Y= Editor:

```
-PLOTS
y1={4·sin(x),x< -π, x< 0
    {2·x+6,else
    6-x^2,else
y2=
y3=
y4=
y1(x)=when(x<0,when(x<-π,...
```

where:

$$y1(x)=\text{when}(x<0,\text{when}(x<-\pi,4*\sin(x),2x+6),6-x^2)$$

This nested function is in effect when $x < 0$.

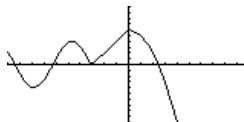
Nested functions quickly become complex and difficult to visualize.

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 *Programming*.

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.

```
❶ Func
    If  $x < -\pi$  Then
        Return  $4 * \sin(x)$ 
    ElseIf  $x \geq -\pi$  and  $x < 0$  Then
        Return  $2x + 6$ 
    Else
        Return  $6 - x^2$ 
    EndIf
❶ EndFunc
```

❶ **Func** and **EndFunc** must begin and end the function.

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
```

In the Y= Editor:

Only **Func** is shown for a multi-statement function.

```
Y1=Func
```

Enter a multi-statement function on one line. Be sure to include colons.

```
Y1(x)=Func:If x< -π Then:R...
```

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.

Information is available on copying a function from the Home screen to the Y= Editor.

From the Program Editor, 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

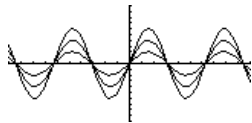
By entering a list in an expression, you can plot a separate function for each value in the list. (You cannot graph a family of curves in SEQUENCE or 3D graphing mode.)

Examples Using the Y= Editor

Enter the expression $\{2,4,6\} \sin(x)$ and graph the functions.

Note: Enclose list elements in braces ($\overline{[2nd]} [\{ \}$ and $\overline{[2nd]} []$) and separate them with commas.

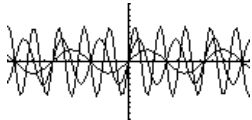
```
▬FLBTS
✓y1={2 4 6}·sin(x)
y2=
y3=
y4=
y5=
y6=
y7=
y8=
y9=
y10=
y1(x)={2,4,6}·sin(x)
```



Graphs three functions:
 $2 \sin(x)$, $4 \sin(x)$, $6 \sin(x)$

Enter the expression $\{2,4,6\} \sin(\{1,2,3\} x)$ and graph the functions.

```
▬FLBTS
✓y1={2 4 6}·sin({1 2 }
y2=
y3=
y4=
y5=
y6=
y7=
y8=
y9=
y10=
y1(x)={2,4,6}·sin({1,2,3}...
```



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.

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.

<small>↑PLOTS</small>
✓y1={2 4 6}·sin(x)
✓y2={1 2 3}·x + 4
✓y3=cos(x)

For these example functions, the TI-89 Titanium / Voyage™ 200 Graphing Calculator 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.

Note: To set graph formats from the Y= Editor, Window Editor, or Graph screen, press:

 F

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 calculator'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 *Split Screens*.

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.

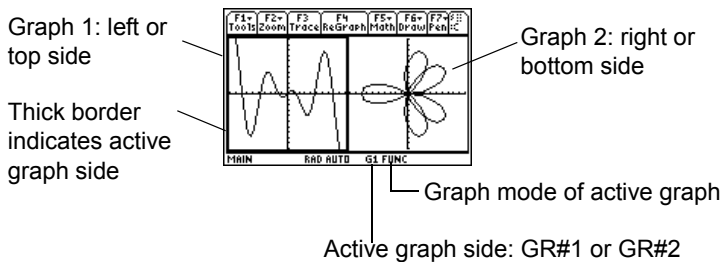
- Page 1:**
- **Graph = Graph mode for top or left side of the split**

- 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 (Voyage™ 200 only)

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 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).

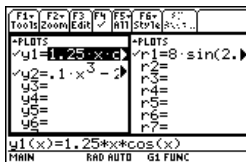
Independent graph-related applications (Y= Editor, Graph screen, etc.) can be displayed on both sides of the screen at the same time.

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

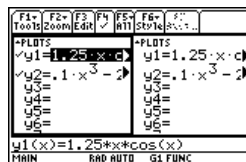
Even in two-graph mode, there is actually only one Y= Editor, which maintains a single function list for each Graph mode setting. However, if both sides use the same graphing mode, each side can select different functions from that single list.

- 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 $\boxed{F4}$ to select different functions and stat plots (indicated by \checkmark) for each side.
- If you set a display style for a function, that style is used by both sides. $\boxed{F6}$



- Suppose Graph 1 and Graph 2 are set for function graphing. Although both sides show the same function list, you can select (\checkmark) different functions for graphing

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.

Using a Split Screen

For more complete information about split screens, refer to *Split Screens*.

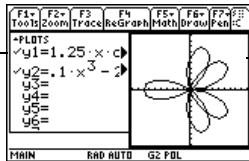
- To switch from one graph side to the other, press $\boxed{2\text{nd}} \boxed{[\pm]}$ (second function of $\boxed{\text{APPS}}$).
- To display different applications:
 - Switch to the applicable graph side and display the application as you normally would.
 - or –
 - Use $\boxed{\text{MODE}}$ to change **Split 1 App** and/or **Split 2 App**.
- To exit two-graph mode:
 - Use $\boxed{\text{MODE}}$ to set **Number of Graphs = 1**, or exit the split screen by setting **Split Screen = FULL**.
 - or –
 - Press $\boxed{2\text{nd}} \boxed{[\text{QUIT}]}$ twice. This always exits a split screen and returns to a full-sized Home screen.

Note: You can display non-graph-related applications (such as the Home screen) on only one side at a time.

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 $r(\theta)$ equations that are not shown.

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.

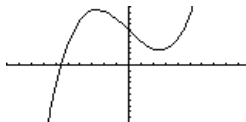
DrawFunc *expression*

DrawParm *expression1, expression2* [,*tmin*] [,*tmax*] [,*tstep*]

DrawPol *expression* [,*θmin*] [,*θmax*] [,*θstep*]

For example:

1. Define $y_1(x) = .1x^3 - 2x + 6$ on the Y= Editor, and graph the function.



2. On the Graph screen, press: $\boxed{F6}$ and select **2:DrawFunc**.

To display the Home screen and put **DrawFunc** in the entry line, press: $\boxed{F6}$ 2



3. On the Home screen, specify the function to draw.

$\boxed{\text{DrawFunc } y_1(x) - 6}$

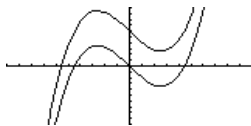
4. Press **ENTER** to draw the function on the Graph screen.

You cannot trace, zoom, or perform a math operation on a drawn function.

Note: To clear the drawn function, press

F4

– or –



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 $y_1(x) = .1x^3 - 2x + 6$ as shown above.

1. On the Graph screen, press: and select **3:DrawInv**

To display the Home screen and put **DrawInv** in the entry line, press:

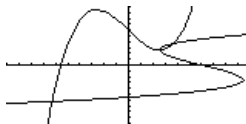
F6 **3**

2. On the Home screen, specify the inverse function.

DrawInv $y_1(x)$

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.

From the Graph screen:

- **[F6]**
and select **1:ClrDraw**.
– or –
- Press **[F4]** to regraph.



Note: You can also enter **ClrDraw** on the Home screen’s entry line.

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. **F7**
and select **1:Pencil**.
2. Move the cursor to the applicable location.

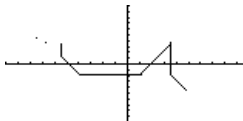


To draw a:	Do this:
Point (pixel-sized)	Press ENTER .
Freehand line	Press and hold ↻ , and move the cursor to draw the line. To quit drawing the line, release ↑ or ↻ .

Note: When drawing a freehand line, you can move the cursor diagonally.

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**.



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).

Erasing Individual Parts of a Drawing Object

From the Graph screen:

1. **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**.

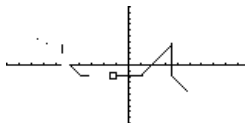
Along a freehand line

Press and hold **↶**, and move the cursor to erase the line.To quit, release **↑** or **↶**.

Note: These techniques also erase parts of graphed functions.

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

From the Graph screen:

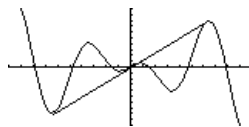
1. **F7**
and select **3:Line**.

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.)

Note: Use **[2nd]** to move the cursor in larger increments; **[2nd]** **[↓]**, etc.

After drawing the line, you are still in **Line** mode.

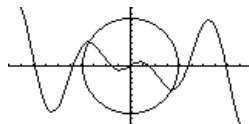
- To continue drawing another line, move the cursor to a new 1st point.
- To quit, press **[ESC]**.



Drawing a Circle

From the Graph screen:

1. **[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]**.



Note: Use **[2nd]** to move the cursor in larger increments; **[2nd]** **[↓]**, etc.

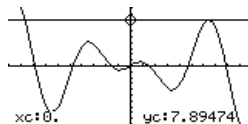
Drawing a Horizontal or Vertical Line

From the Graph screen:

1. **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.
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**.



Note: Use **2nd** to move the cursor in larger increments; **2nd** **⏏**, etc.

Drawing a Tangent Line

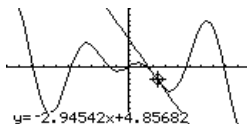
To draw a tangent line, use the **F5** **Math** toolbar menu. From the Graph screen:

1. Press **F5** and select **A:Tangent**.

2. As necessary, use \ominus and $\omin�$ to select the applicable function.

3. Move the cursor to the tangent point, and press $\boxed{\text{ENTER}}$.

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



Note: To set the tangent point, you can also type its x value and press $\boxed{\text{ENTER}}$.

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.

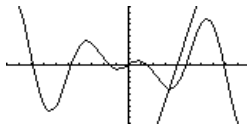
1. $\boxed{\text{F6}}$

and select **6:DrawSlp**. This switches to the Home screen and puts **DrawSlp** in the entry line.

2. Complete the command, and press $\boxed{\text{ENTER}}$.

$\boxed{\text{DrawSlp } 4,0,6.37}$

The calculator automatically switches to the **Graph** screen and draws the line.

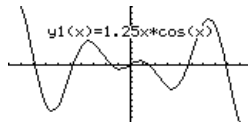


Typing Text Labels

From the Graph screen:

1. **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**.

Note: The text cursor indicates the upper-left corner of the next character you type.

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 *Programming*.

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; and then:

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.  

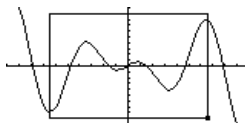
and select **8:Save Picture**.





A box is shown around the outer edge of the screen.






Note: You cannot save a portion of a 3D graph.

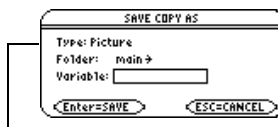
2. Set the 1st corner of the box by moving its top and left sides. Then press .



Note: Use  and  to move the top or bottom, and use  and  to move the sides.

3. Set the 2nd corner by moving the bottom and right sides. Then press .

4. Specify the folder and a unique variable name.
5. Press . After typing in an input box such as **Variable**, you must press  twice.



Note: When saving a portion of a graph, Type is automatically fixed as Picture.

Opening a Graph Picture

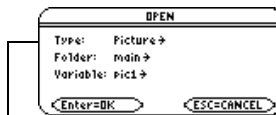
When you open a graph picture, it is superimposed over the current Graph screen. To display only the picture, use the Y= Editor to deselect any other functions before opening the graph picture.

From the Graph screen:

1. Press **[F1]** and select **1:Open**.
2. Select the type (**Picture**), folder, and variable that contain the graph picture you want to open.

Note: If a variable name is not shown on the dialog box, there are no graph pictures in the folder.

3. Press **[ENTER]**.



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, it may appear shifted from the underlying graph.

You can specify which screen pixel to use as the upper-left corner.

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 *Memory and Variable Management*.

From a Program or the Home Screen

To save (**store**) and open (**recall**) a graph picture, use the **StoPic**, **RclPic**, **AndPic**, **XorPic**, and **RpicPic** commands as described in the *Technical Reference* module.

To display a series of graph pictures as an animation, use the **CyclePic** command. For an example, refer to CyclePic Command.

Animating a Series of Graph Pictures

As described earlier in this module, 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*]

❶

❷

❸

❹

❺

- ❶ base name of pictures in quotes, such as "pic"
- ❷ # of pictures to cycle
- ❸ seconds between
- ❹ # of times to repeat cycle

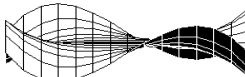
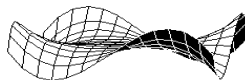
Example

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 the

Technical Reference module. For information about using the Program Editor, refer to *Programming*.

Program Listing**Every Other Graph from Program**

```
:cyc ()
:Prgm
:local I
:ⓄSet mode and Window variables
:setMode ("graph", "3d")
:70→eyeθ
:-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 **Ⓢ**. Press:

2nd **X**

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

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:
F1 **9**
– or –
F

A graph database does not include drawn objects or stat plots.

Note: In two-graph mode, the elements for both graphs are saved in a single database.

Saving the Current Graph Database

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

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 ($\overline{2nd}$ [VAR-LINK]) described in *Memory and Variable Management*.

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 the *Technical Reference* module.

Split Screens

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.

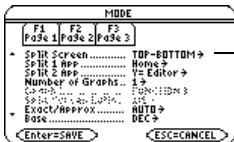
Setting the Split Screen Mode

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 *Operating the Calculator*.

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.

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

You can open different applications after the split screen is displayed.

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

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 <i>Additional Graphing Topics</i> .	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 <i>Additional Graphing Topics</i> .

Split Screens and Pixel Coordinates

The calculator has commands that use pixel coordinates to draw lines, circles, etc., on the Graph screen. The following charts show how the **Split Screen** and **Split Screen Ratio** mode settings affect the number of pixels available on the Graph screen.

Note:

- For a list of drawing commands, refer to “Drawing on the Graph Screen” in *Programming*.
- Due to the border that indicates the active application, split screens have a smaller displayable area than a full screen.

TI-89 Titanium:

Split	Ratio	Split 1 App		Split 2 App	
		x	y	x	y
FULL	N/A	0 – 158	0 – 76	N/A	N/A

Split	Ratio	Split 1 App		Split 2 App	
		x	y	x	y
TOP-BOTTOM	1:1	0 – 154	0 – 34	0 – 154	0 – 34
LEFT-RIGHT	1:1	0 – 76	0 – 72	0 – 76	0 – 72

Voyage™ 200:

Split	Ratio	Split 1 App		Split 2 App	
		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 Calculator

Turning the calculator off does not exit the split screen mode.

If the calculator is turned off:

When you turn the calculator 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]**.

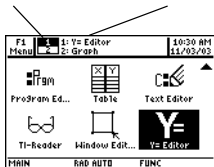
The split screen is just as you left it.

Split-Screen Status Indicators on the Apps Desktop

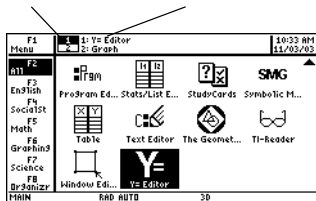
To return to the Apps desktop, press **[APPS]**. The split-screen status appears at the top of the Apps desktop with the names of the open Apps and the portions of the screen in which each App is displayed.

Note: The Apps desktop always appears in the full-screen view.

Split-screen indicator Names of open Apps



Split-screen indicator Names of open Apps



Split screen indicator	Description
	<p>Top-bottom split screen</p> <ul style="list-style-type: none"> • 1 indicates the application that will appear in the top portion of the screen. • 2 indicates the application that will appear in the bottom portion of the screen. <p>The highlighted numeral indicates the active portion of the split screen.</p>

1 2

Left-right split screen

- **1** indicates the application that will appear in the left portion of the screen.
- **2** indicates the application that will appear in the right portion of the screen.

The highlighted numeral indicates the active portion of the split screen.

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.

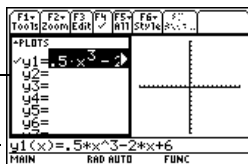
The Active Application

- 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.

Toolbar is for Y= Editor.

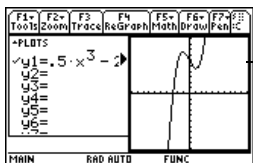
Thick border indicates the Y= Editor is active.

Entry line is full width when Y= Editor is active.



Switching between Applications

Press $\boxed{2\text{nd}} \boxed{[\text{=}]}$ (second function of $\boxed{\text{APPS}}$) to switch from one application to the other.



- Toolbar is for Graph screen.
- Thick border indicates the Graph screen is active.
- Graph screen does not have an entry line.

Opening a Different Application

- Method 1:
1. Use $\boxed{2\text{nd}} \boxed{[\text{=}]}$ to switch to the application you want to replace.
 2. Use $\boxed{\text{APPS}}$ or $\boxed{\blacklozenge}$ (such as $\boxed{\blacklozenge} \boxed{[\text{WINDOW}]}$) to select the new application.

If you select an application that is already displayed, the calculator switches to that application.

- Method 2:
3. Press $\boxed{\text{MODE}}$ and then $\boxed{\text{F2}}$.
 4. Change **Split 1 App** and/or **Split 2 App**.

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

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

Using 2nd QUIT to Display the Home Screen

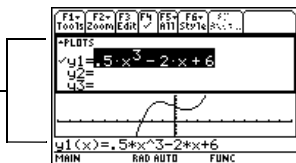
Note: Pressing $\boxed{2nd}$ [QUIT] twice always exits the split screen mode.

If the Home screen:	Pressing $\boxed{2nd}$ [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.

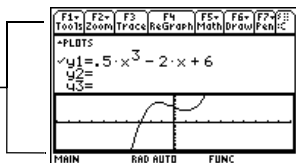
When Using a Top-Bottom Split

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.



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



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

Data/Matrix Editor

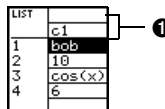
Overview of List, Data, and Matrix Variables

To use the Data/Matrix Editor effectively, you must understand list, data, and matrix variables.

List 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.
 - ❶ Column title and header cells are not saved as part of the list.



	c1
1	bob
2	10
3	cos(x)
4	6

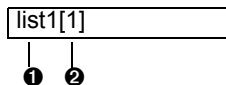
If you enter more than one column of elements in a list variable, it is converted automatically into a data variable.

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.

▀	{bob	10	cos(x)	6	1	▶
	{bob	10	cos(x)	6	1	▶
	c:10,cos(x),6,1,hi)+list1					
MIN	RND	QUIT	FUNC	1/20		

To refer to a specified element in a list, use the format shown to the right.



- ① Name of list variable
- ② Element number (or index number)

Note: After creating a list in the Data/Matrix Editor, you can use the list in any application (such as the Home screen).

Data Variable

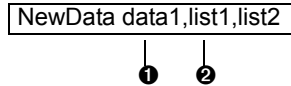
A data variable is essentially a collection of lists that may or may not be related. In the Data/Matrix Editor, a data variable:

- 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.
- Must have continuous columns; blank or empty cells are not allowed within a column.

DATA	c1	c2	c3
1	fred	stone	95
2	sally	ross	75
3	jane	smith	97
4	nick	castle	83

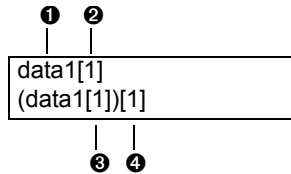
Note: For stat calculations, columns must have the same length.

From the Home screen or a program, you can use the **NewData** command to create a data variable that consists of existing lists.



- ❶ Name of data variable to create
- ❷ Names of existing lists

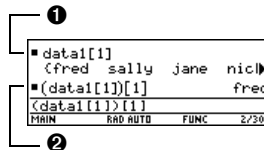
Although you cannot directly display a data variable on the Home screen, you can display a specified column or element.



- ❶ Name of data variable
- ❷ Column number
- ❸ Column number
- ❹ Element number in the column

For example:

- ❶ Displays column 1 of the variable data1.
- ❷ 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:

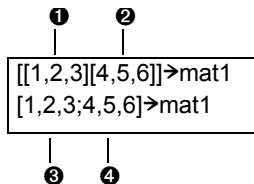
- Looks similar to a data variable, but all columns must have the same length.
- Is initially created with 0 in each cell. You can then enter the applicable value in place of 0.

MAT			
2x3	c1	c2	c3
1	1	2	3
2	4	5	6

Shows the size of the matrix.

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 row 1
- 2 row 2
- 3 row 1
- 4 row 2



Although you enter the matrix as shown above, it is pretty printed in the history area in traditional matrix form.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	\rightarrow mat1	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	\rightarrow mat1	
MIN	RAD AUTO	FUNC 1/30

After creating a matrix in the Data/Matrix Editor, you can use the matrix in any application (such as the Home screen).

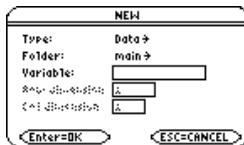
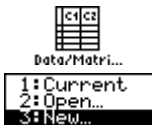
Note: Use brackets to refer to a specific element in a matrix. For example, enter **mat1[2,1]** to access the 1st element in the 2nd row.


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.

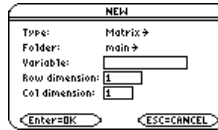
Creating a New Data, Matrix, or List Variable

1. Press **[APPS]** and then select the Data/Matrix icon. Press **[ENTER]**.
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 <i>the Calculator Home Screen module</i> .

Item	Lets you:
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, your calculator displays the Home screen.

- 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, launch Data/Matrix Editor again 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**.
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.

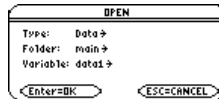
1. From the Data/Matrix Editor, press **[F1]** and select **1:Open**.

– or –

From any application, launch Data/Matrix Editor again 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.

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 ($\text{[2nd]} \text{[VAR-LINK]}$). For information about VAR-LINK, refer to *Memory and Variable Management*.

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

A blank Data/Matrix Editor screen is shown below. When the screen is displayed initially, the cursor highlights the cell at row 1, column 1.

- 1 Variable type
- 2 Column headers
- 3 Row numbers
- 4 Row and column number of highlighted cell
- 5 Column title cells, used to type a title for each column

F1 Tools	F2 Plot Setup	F3 Cell Header	F4 Header	F5 Calc	F6 Util	F7 Stat
DATA		c1	c2	c3		
1						
2						
3						
4						
r1c1=						
MAIN		RAD AUTO		FUNC		

When values are entered, the entry line shows the full value of the highlighted cell.

Note: Use the title cell at the very top of each column to identify the information in that column.

Entering or Editing a Value in a Cell

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.

Note: 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.

Variable Type	After pressing 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 ⬆️
Go to row 1 in the current column or to the last row that contains data for any column on the screen, respectively. If the cursor is in or past that last row, ⬆️ ⬇️ goes to row 999.	⬆️ ⬅️ or ⬆️ ⬇️
Go to column 1 or to the last column that contains data, respectively. If the cursor is in or past that last column, ⬆️ ⬇️ goes to column 99.	⬆️ ⬆️ or ⬆️ ⬇️

Note: To enter a value from the entry line, you can also use ⬇️ 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, your device handles the gaps automatically.

- In a list variable, a cell in the gap is undefined until you enter a value for the cell.

LIST	
1	c1
2	
3	

→

LIST	
3	c1
4	3
5	undef
6	5

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.

DATA	c1	c2	c3
1	1		
2			
3			
4			

→

DATA	c1	c2	c3
1	1		undef
2			undef
3			45
4			

- 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.

MAT			
2x3	c2	c3	c4
1	2	3	
2	5	6	
3			
4			

→

MAT			
3x4	c2	c3	c4
1	2	3	0
2	5	6	0
3	0	0	12
4			

Note: Although you specify the size of a matrix when you create it, you can easily add additional rows and/or columns.

Changing the Cell Width

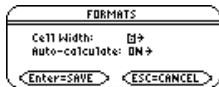
The cell width affects how many characters are displayed in any cell. To change the cell width in the Data/Matrix Editor:

1. To display the **FORMATS** dialog box, press:

F1 9

– or –

F



Cell width is the maximum number of characters that can be displayed in a cell. All cells have the same cell width.

- Note:** Remember, to see a number in full precision, you can always highlight the cell and look at the entry line.
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

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:

and select **5:Clear Column**.
(This item is not available for a matrix.)
-

All columns

Press and select **8:Clear Editor**. When prompted for confirmation, press (or to cancel).

Note: For a list or data variable, a clear column is empty. For a matrix, a clear column contains zeros.

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**.

Notes:

- **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**.
- 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 \leftarrow or \rightarrow to remove the highlighting. Then edit the old expression.

Note: To cancel any changes, press $\boxed{\text{ESC}}$ before pressing $\boxed{\text{ENTER}}$.

You can use an expression that: For example:

Generates a series of numbers. $c1=\text{seq}(x^2,x,1,5)$
 $c1=\{1,2,3,4,5\}$

Refers to another column. $c2=2*c1$
 $c4=c1*c2-\sin(c3)$

Note: The seq function is described in the Technical Reference module. If you refer to an empty column, you will get an error message unless **Auto-calculate = OFF**.

3. Press $\boxed{\text{ENTER}}$, \uparrow , or \downarrow to save the definition and update the columns.

DATA	c1	c2	c3
1	1	2	3
2	2	4	
3	3	6	
4	4	8	

Formula bar: $c1=1$

① $c1=\text{seq}(x,x,1,7)$; ② $c2=2*c1$; ③ You cannot directly change a locked cell (■) since it is defined by the column header.

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.

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: <code>c1=list1</code>
Home screen or a program	Use the NewData command as described in the <i>Technical Reference</i> module. For example:

NewData *datavar*, *list1* [, *list2*] [, *list3*] ...

① ②

- ① Data variable. If this data variable already exists, it will be redefined based on the specified lists.
- ② Existing list variables to copy to columns in the data variable.

Note: If you have a CBL 2™ or CBR™, use these techniques for your collected lists. Use `[2nd] [VAR-LINK]` to see existing list variables.

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 the *Technical Reference* module.

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.

To turn Auto-calculate off and on from the Data/Matrix Editor:

1. Press:

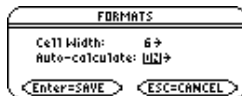
F1 9

– or –

F

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**.

Note: You may want to set **Auto-calculate = OFF** to make changes without recalculating each time, enter a definition such as $c1=c2+c3$ before you enter columns 2 and 3, or override any errors in a definition until you can debug the error.

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 the *Technical Reference* module. This section describes how to use the functions in the Data/Matrix Editor. The *Technical Reference* module 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*])

①

②

① Column used as the base for the shift.

② Number of elements to shift (positive shifts up; negative shifts down).

Default is -1.

For example, for a two-element shift up and down:

	①	②	
c1	c2	c3	
1	3	undef	
2	4	undef	
3	undef	1	③
4	undef	2	
	⑤	④	

① c2=shift(c1,2)

② c3=shift(c1,-2)

③ Shifted columns have the same length as the base

④ Last two elements of c1 shift down and out the bottom; undefined elements shift into the top.

⑤ 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 the 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:

c1	c2
1	1
2	3
3	6
4	10

c2=cumSum(c1)

1+2

1+2+3+4

Note: To enter **cumSum**, type it, select it from the 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:

1. Move the cursor to any cell in the “key” column.
2. In this example, move the cursor to the second column (c2) to sort by last name.

c1	c2	c3
fred	stone	95
sally	ross	75
jane	smith	97
nick	castle	93

Note: For a **list** variable, this is the same as sorting a single column.

3. Press:
F6
and select **4:Sort Col**, adjust all.

c1	c2	c3
nick	castle	93
sally	ross	75
jane	smith	97
fred	stone	95

Note: This menu item is not available if any column is locked.

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, **L** 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

You can copy a:	To a:
List	List or data
Data	Data
Data column	List
Matrix	Matrix

Note: A list is automatically converted to a data variable if you enter more than one column of information.

Procedure

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.



Note: If you type the name of an existing variable, its contents will be replaced.

❶ 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.



❶ List variable to copy to.

❷ Data column that will be copied to the list. By default, this shows the column that contains the cursor.

Statistics and Data Plots

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.

1. Set Graph mode (**MODE**) to **FUNCTION**.
2. Enter stat data in the Data/Matrix Editor.
Note: Refer to the Data/Matrix Editor module for details on entering data in the Data/Matrix Editor.
3. Perform stat calculations to find stat variables or fit data to a model (**F5**).

F1	F2	F3	F4	F5	F6	F7
Tools	Plot	Status	Calc	Header	Calc	Plot
DATA						
	C1	C2	C3			
1	150	4				
2	250	9				
3	500	31				
4	500	20				
Σ C1 = 150						
MAIN RAD AUTO FUNC						

4. Define and select stat plots (**F2** and then **F1**).
5. Define the viewing window (**◀** [WINDOW]).

```
main\build Calculate
Calculation Type..... MedMed >
X..... C1
Y..... C2
Store REGEQ to..... Y1(X) >
Free and Categories?.. ND >
Stat.....
Calc.....
Main Calc..... C1
<Enter>=SAVE      <ESC>=CANCEL
```

```
main\build
F1 F2 F3 F4
Define Copy Clear
Plot 1: [◀] [x] [c1] [y] [c2]
Plot 2:
Plot 3:
Plot 4:
Plot 5:
Plot 6:
Plot 7:
Plot 8:
Plot 9:
```

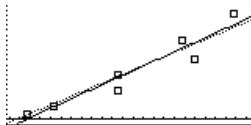
6. Change the graph format if necessary.

F1 9

— OR —



F



7. Graph the selected equations

[F2] [GRAPH].

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 **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

Pathname of the data variable

main\build Calculate	
Calculation Type	CubicRe3 ↕
X	C1
Y	C2
Store RESID to	none ↕
Freq and Categories?	YES ↕
Freq	
Category	
* Include Categories	<input checked="" type="checkbox"/>
<Enter>=SAVE <ESC>=CANCEL	

2. Specify applicable settings for the active items.

Item	Description
Calculation Type	Select the type of calculation.
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.
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: To use an existing list variable for x, y, Freq, or Category, type the list name instead of a column number. An example using Freq, Category, and Include Categories is available.

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:

For Calculation Type = OneVar

For Calculation Type = LinReg

STAT VARS	
Σ	=33.428571
Σx	=234.
Σx^2	=14576.
Sx	=25.012378
nStat	=7.
minX	=4.
41	=9.
medStat	=31.
Enter=OK	

STAT VARS	
y=a*x+b	
a	=.081561
b	=-12.012431
corr	=.957317
R ²	=.916457
Enter=OK	

When ▼ is shown instead of =, you can scroll for additional results.

Note: Any undefined data points (shown as **undef**) are ignored in a stat calculation.

4. To close the **STAT VARS** screen, press **ENTER**.

Redisplaying the STAT VARS Screen

The Data/Matrix Editor's 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).

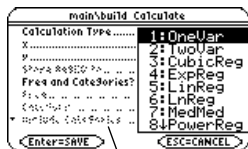
Statistical Calculation Types

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.

Calc Type	Description
OneVar	One-variable statistics — Calculates the statistical variables.
TwoVar	Two-variable statistics — Calculates the statistical variables.
CubicReg	<p>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.</p> <ul style="list-style-type: none"> • For four points, the equation is a polynomial fit. • For five or more points, it is a polynomial regression.
ExpReg	<p>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 $\ln(y)$.</p>
LinReg	<p>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.</p>
LnReg	<p>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.</p>
Logistic	<p>Logistic regression — Fits the data to the model $y=a/(1+b*e^{(c*x)})+d$ and updates all the system statistics variables.</p>
MedMed	<p>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.</p>

Calc Type	Description
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. <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • For five points, the equation is a polynomial fit. • For six or more points, it is a polynomial regression.
SinReg	Sinusoidal regression — Calculates the sinusoidal regression and updates all the system statistics variables. The output is always in radians, regardless of the angle mode setting.

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.

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 the *Technical Reference* module for information about each command.

Important: These commands perform a statistical 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 *Memory and Variable Management*. All statistical variables are cleared when you edit the data or change the calculation type. Other conditions that clear the variables are listed.

Calculated Variables

Statistical variables are stored as system variables. However, **regCoef** and **regeq** are treated as a list and a function variable, respectively.

	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	

	One Var	Two Var	Regressions
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

†† **corr** is defined for a linear regression only; R^2 is defined for all polynomial regressions.

Note:

- 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]**.
- 1st quartile is the median of points between **minX** and **medStat**, and 3rd quartile is the median of points between **medStat** and **maxX**.

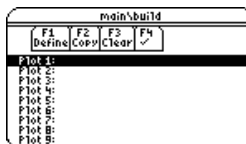
Defining a Statistical Plot

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.



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.

Pathname of the data variable

4. Specify applicable settings for the active items.

Item	Description
Plot Type	Select the type of plot.
Mark	Select the symbol used to plot the data points: Box (<input type="checkbox"/>) , Cross (x), Plus (+), Square (<input type="checkbox"/>) , 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.
Freq and Categories?	Select NO or YES . Note that Freq, Category, and Include Categories are active only when Freq and Categories? = YES. (Freq is active only for Plot Type = Box Plot or Histogram.)

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.

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.
- To use an existing list variable for x, y, Freq, or Category, type the list name instead of the column number.
- An example using Freq, Category, and Include Categories is available.

5. Press **ENTER** (after typing in an input box, press **ENTER** twice).

The **Plot Setup** screen is redisplayed.

The plot you just defined is automatically selected for graphing.

Notice the shorthand definition for the plot.



Plot Type = Scatter
Mark = Box
Plot 1: [] □ x:c1 y:c2
x = c1 y = c2

Note: Any undefined data points (shown as **undef**) are ignored in a stat 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.

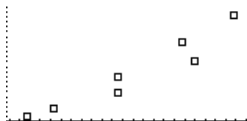
Statistical Plot Types

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

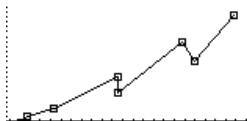
Data points from x and y are plotted as coordinate pairs. Therefore, the columns or lists that you specify for x and y must be the same length.

- 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.



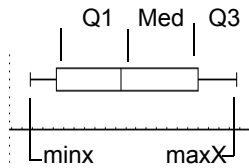
You may want to sort all the columns in the Data/Matrix Editor before plotting.

 **F6 3** or **F6 4**

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.
- Use NewPlot to show statistical data as a modified box plot.
- Select Mod Box Plot as the Plot Type when you define a plot in the Data/Matrix Editor.

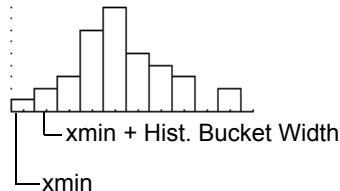
A modified box plot excludes points outside the interval $[Q1 - X, Q3 + X]$, where X is defined as $1.5(Q3 - Q1)$. These points, called outliers, are plotted individually beyond the box plot's whiskers, using the mark that you select.

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.

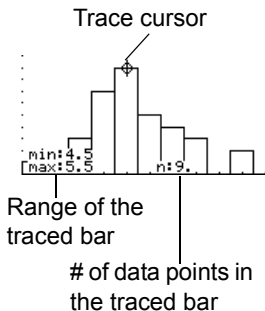
- When defining the plot, you can specify the **Hist. Bucket Width** (default is 1) to set the width of each bar.
- A data point at the edge of a bar is counted in the bar to the right.

$$\text{Number of bars} = \frac{\text{xmax} - \text{xmin}}{\text{Hist. Bucket Width}}$$



- **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 \blacktriangle [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.

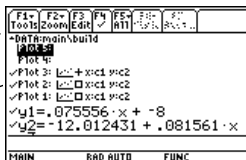
For example, PLOTS 23 means that Plots 2 & 3 are selected.



To see the list of stat plots, use \leftarrow to scroll above the $y(x)$ functions.

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: 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.

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.

Note: You can not use  **[F6]** to set a plot's display style. However, the plot definition lets you select the mark used for the plot.

To Graph Plots and Y= Functions

As necessary, you can select and graph stat plots and $y(x)$ functions at the same time.


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 *Basic Function Graphing*).

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  **Zoom** toolbar menu. (Although you can use any zoom, **ZoomData** is optimized for st plots.)

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**.



Note:  **Zoom** is available on the Y= Editor, Window Editor, and Graph screen.

Changing the Graph Format

Press:

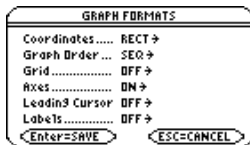
F1 9

— or —



F

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.

Plot Type	Description
Scatter or xylene	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.

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.

When you press \ominus or $\omin�$ 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 statistical 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.

Test scores	Frequency values
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 ①
92 ①
89
91
95 ②
95 ②
95 ②

① Frequency of 2

② Frequency of 3

Note: A frequency value of 0 effectively removes the data point from analysis.

To use frequency values, specify the frequency column when you perform a statistical calculation or define a stat plot. For example:

Set this to YES.

Type the column number (or list name) that contains the frequency values.

main\data1 Calculate

Calculation Type	OneVar
X	C1
Sigma Squared	Data 1
Freq and Categories?	YES
Freq	C2
Cate30P	C2

Enter=SAVE ESC=CANCEL

Note: You can also use frequency values from a list variable instead of a column.

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.

Category Value	Used to indicate:
1	10th grade girl
2	10th grade boy
3	11th grade girl
4	11th grade boy

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.

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

To use category values, specify the category column and the category values to include in the analysis when you perform a statistical calculation or define a stat plot.

Set this to YES.

Type the column number (or list name) that contains the category values.

main\data1 Calculate

Calculation Type	OneVar →
X	CL
Y	CL
Store Result?	Auto →
Free and Categories?	YES →
Free	
Calc300Y	CE
Include Categories	{1,2}
◀Enter=SAVE ▶	
◀Esc=CANCEL ▶	

Within braces { }, type the category values to use, separated by commas. (Do not type a column number or list name.)

Note: You can also use category values from a list variable instead of a column.

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}

Note: To analyze the whole class, leave the Category input box blank. Any category values are ignored.

If You Have a CBL 2™ or CBR™

The Calculator-Based Laboratory™ System (CBL 2) and Calculator-Based Ranger™ System (CBR) are optional accessories, available separately, that let you collect data from a variety of real-world experiments. Voyage™ 200, CBL 2 and CBR programs are available from the TI web site at education.ti.com.

How CBL 2™ Data Is Stored

When you collect data with the CBL 2, that data is initially stored in the CBL 2 unit itself. You must then retrieve the data (transfer it to the Voyage™ 200) by using the **Get** command, which is described in the *Technical Reference* module.

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 statistical calculations.

When you transfer the collected information to the Voyage™ 200, you can specify the list variable names that you want to use.

For example, you can use the CBL 2 to collect temperature data over a period of time. When you transfer the data, suppose you store:

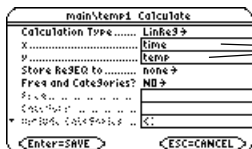
- Temperature data in a list variable called *temp*.
- Time data in a list variable called *time*.

After you store the CBL 2 information on the Voyage™ 200, there are two ways to use the CBL 2 list variables.

Note: For specifics about using the CBL 2 and retrieving data to the Voyage™ 200, refer to the guidebook that comes with the CBL 2 unit.

Referring to the CBL 2™ Lists

When you perform a statistical calculation or define a plot, you can refer explicitly to the CBL 2 list variables. For example:



The screenshot shows a calculator screen titled "main/temp1 Calculate". The screen displays several menu options with list variable names entered instead of column numbers:

- Calculation Type LinRc3 →
- X time
- Y temp
- Slope ResEQ to time →
- Freq and CateSeries? NO →
- Statistics
- Categories
- Horizontal CateSeries?

At the bottom, there are two buttons: "Enter=SAVE" and "ESC=CANCEL".

Type the CBL 2 list variable name instead of a column number

Creating a Data Variable with the CBL 2™ Lists

You can create a new data variable that consists of the necessary CBL 2 list variables.

- From the Home screen or a program, use the **NewData** command.

NewData *dataVar*, *list1* [*,list2*] [*,list3*] ...

— CBL 2 list variable names. In the new data variable, *list1* will be copied to column 1, *list2* 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.

- From the Data/Matrix Editor, create a new, empty data variable with the applicable name. For each CBL 2 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*.

F1	F2	F3	F4	F5	F6	F7
Tools	Plot Setup	Cat	Header	Calc	Mat	Stat
DATA	TIME	TEMP				
	c1	c2	c3			
1	1	120				
2	2	95				
3	3	85				
4	4	79				
c1, Title="TIME"						
MAIN <input type="checkbox"/> END AUTO <input type="checkbox"/> FUNC						

Note: To define or clear a column header, use **[F4]**. For more information, refer to the *Data/Matrix Editor* module.

At this point, the columns are linked to the CBL 2 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 lists, clear the column header for each column. The information remains in the column, but the column is no longer linked to the CBL 2 list.

You can also use the Calculator-Based Ranger™ (CBR) to explore the mathematical and scientific relationships between distance, velocity, acceleration, and time using data collected from activities you perform.

Programming

Running an Existing Program

After a program is created (as described in the remaining sections of this module), 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

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.

Note: Use **[2nd]** **[VAR-LINK]** to list existing **PRGM** variables.

Highlight a variable and press **[ENTER]** to paste its name to the entry line.

prog1(x,y)

└ If arguments are required

3. Press **[ENTER]**.

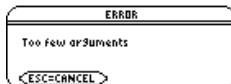
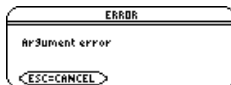
Note: Arguments specify initial values for a program.

When you run a program, the Voyage™ 200 Graphing Calculator automatically checks for errors. For example, the following message is displayed if you:

- Do not enter () after the program name.

This error message appears if you:

- 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.

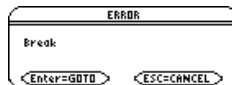
Note: The Voyage™ 200 also checks for run-time errors that are found within the program itself.

“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?

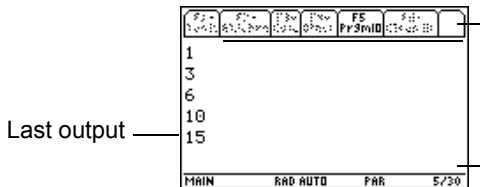
Depending on the commands in the program, the Voyage™ 200 automatically displays information on the applicable screen.

- 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 Voyage™ 200 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.



On the Program I/O screen:
 [F5] toolbar is available; all others are dimmed.
 There is no entry line.

Note: To clear any previous output, enter the **ClrIO** command in your program. You can also execute **ClrIO** from the Home 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.

Note: If Home screen calculations don't work after you run a program, you may be on the Program I/O screen.

Leaving the Program I/O Screen

From the Program I/O screen:

- Press [F5] to toggle between the Home screen and the Program I/O screen.
 – or –
- Press [ESC], [2nd] [QUIT], or [HOME] [] [CALC HOME] to display the Home screen.
 – or –
- Display any other application screen (with [APPS], [] [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

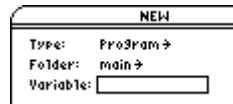
1. Press **[APPS]** and then select **Program Editor**.
2. Select **3:New**.
3. Specify the applicable information for the new program or function.




Program Ed...



1:Current
2:Open...
3:New...



NEW
Type: Program ↗
Folder: main ↗
Variable:

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 the <i>Calculator Home Screen</i> module.

Item	Lets you:
------	-----------

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.”

This is the template for a program. Functions have a similar template.



```
F1 Tools F2 Control F3 I/O F4 Var F5 Find... F6 Mode
: Prgm()
: Prgm
: EndPrgm
MAIN RAD AUTO PAR
```

You can now use the Program Editor as described in the remaining sections of this module.

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.

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, launch Program Editor again 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**.
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**.
– or –
From another application, launch Program Editor again 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**.
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 the *Memory and Variable Management* module.

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.





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.



```
F1- F2- F3- F4- F5- F6-
File Control View Find... Mode
:prgm()
:Prgm
:
:EndPrgm
MAIN RAD AUTO PAR
```

Note: Use the cursor pad to scroll through the program for entering or editing commands. Use   or   to go to the top or bottom of a program, respectively.

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 the *Text Editor* module.

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.

Note: Entering a command does not execute that command. It is not executed until you run the program.

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.

```
:prog1()  
:Prgm  
① :ⓈDisplays sum of 1 thru n  
:Request "Enter an integer",n  
② :expr(n)→n:ⓈConvert to numeric expression  
:-----
```

- ① Description of the program .
- ② Description of **expr**.

Note: Use comments to enter information that is useful to someone reading the program code.

To enter the comment symbol, press:

- **2nd** **X**
– or –
- Press **F2** and select **9:Ⓢ**

Controlling the Flow of a Program

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)

```

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 :cos($\pi/4$)→maximum
does not store that result. If you :Disp maximum
need to refer to a result later,
store it to a variable.

Note: A list of output commands is available.

Getting Values into a Program

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 :Disp 12*6
the program itself. :cos($\pi/4$)→maximum
- Include input commands that :Input "Enter a value",i
prompt the users to enter the :Request "Enter an
necessary values when they integer",n
run the program.
- Require the users to pass prog1(3,5)
one or more values to the
program when they run it.

Note: A list of input commands is available.

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.

- 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(x,y,r) ❶  
.Prgm  
:FnOff  
:ZoomStd  
:ZoomSqr  
:Circle x,y,r  
:LineHorz y+r  
:EndPrgm
```

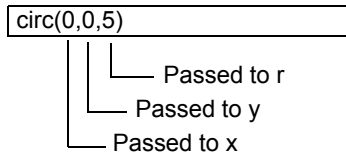
❶ Only **circ()** is initially displayed on the blank template; be sure to edit this line.

Note: In this example, you cannot use circle as the program name because it conflicts with a command name.

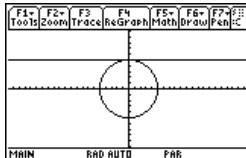
Before drawing the circle, the program turns off any selected Y= Editor functions, displays a standard viewing window, and “squares” the window.

- 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.



Note: This example assumes that the user enters values that can be displayed by the viewing window set up by **ZoomStd** and **ZoomSqr**.

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?

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 Voyage™ 200 Graphing Calculator'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.

Note: You can create a function from the Home screen, but the Program Editor is more convenient for complex, multi-line functions.

Differences Between Functions and Programs

This guidebook sometimes uses 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

For...EndFor

Lbl

Return

Define

Goto

Local

While...EndWhile

Exit

If...EndIf (all forms)

Loop...EndLoop

→ (**STO▶** key)

- Can use all built-in TI-89 Titanium / Voyage™ 200 functions except:

setFold
setTable

setGraph
switch

setMode

- 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.

Note: Information about local variables is available.

Entering a Function

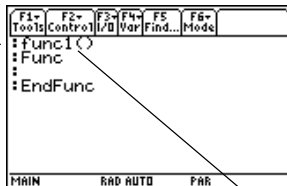
When you create a new function in the Program Editor, the Voyage™ 200 displays a blank “template.”

Note: Use the cursor pad to scroll through the function for entering or editing commands.

Function name, which you specify when you create a new function.

Enter your commands between **Func** and **EndFunc**.

All function lines begin with a colon.



Be sure to edit this line to include any necessary arguments. Remember to use argument names in the definition that will never be used when calling the function.

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

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(x)
:Func
:x^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(x)
:Func
:If x<0
:  Return 0
:x^3
:EndFunc

```

Note: This example calculates the cube if $x \geq 0$; otherwise, it returns a 0.

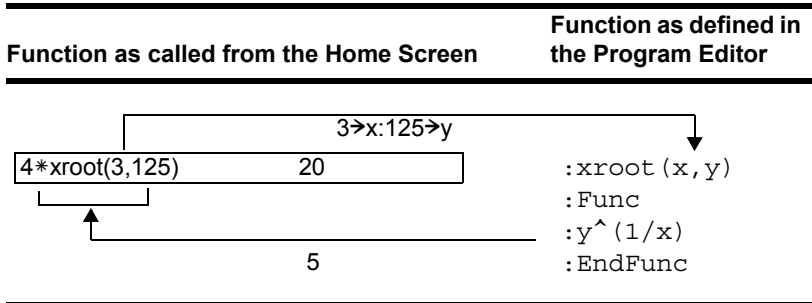
The argument x is automatically treated as a local variable. However, if the example needed another variable, the function would need to declare it as local by using the `Local` command.

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

The following function returns the x th root of a value y ($\sqrt[x]{y}$). Two values must be passed to the function: x and y .

Note: Because x and y in the function are local, they are not affected by any existing x or y variable.



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(x,y)  
:Prgm  
:  Disp x,y  
: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.

```
:subtest1()  
:Prgm  
❶ :local subtest2  
❷ :Define subtest2(x,y)=Prgm  
: : Disp x,y  
❷ :EndPrgm  
:●Beginning of main program  
:For i,1,4,1  
❸ : subtest2(i,I*1000)  
:EndFor  
:EndPrgm
```

- ❶ Declares the subroutine as a local variable.
- ❷ Defines the subroutine.
- ❸ Calls the subroutine.

Note: Use the Program Editor's **[F4] Var** toolbar menu to enter the Define and Prgm...EndPrgm commands.

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

Scope	Description
System (Global) Variables	<p>Variables with reserved names that are created automatically to store data about the state of the calculator. For example, Window variables (xmin, xmax, ymin, ymax, etc.) are globally available from any folder.</p> <ul style="list-style-type: none">• 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.

Scope	Description
Folder Variables	<p>Variables that are stored in a particular folder.</p> <ul style="list-style-type: none">• 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 path name. For example: 5→class\start (class = Variable name; start = Folder Name) <p>After the program stops, any folder variables created by the program still exist and still take up memory.</p>
Local Variables	<p>Temporary variables that exist only while a program is running. When the program stops, local variables are deleted automatically.</p> <ul style="list-style-type: none">• 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.

Note: If a program has local variables, a graphed function cannot access them. For example:

```
Local a
5→a
Graph a*cos(x)
```

may display an error or an unexpected result (if **a** is an existing variable in the current folder).

Circular Definition Errors

When evaluating a user-defined function or running a program, you can specify an argument that includes the same variable that was used to define the function or create the program. However, to avoid Circular definition errors, you must assign a value for **x** or **i** variables that are used in evaluating the function or running the program.

For example:

❶ `x+1→x`

– or –

```
For i,i,10,1
```

❶ `Disp i`
`EndFor`

❶ Causes a `Circular definition` error message if **x** or **i** does not have a value. The error does not occur if **x** or **i** has already been assigned a value.

Variable-Related Commands and Functions

Command	Description
<code>[STO▶]</code> key	Stores a value to a variable. As on the Home screen, pressing <code>[STO▶]</code> enters a <code>→</code> symbol.
Archive	Moves specified variables from RAM to user data archive memory.
BldData	Lets you create a data variable based on the graph information entered in the Y= Editor, Window Editor, etc.
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.
<code>DelType</code>	Deletes unarchived variables of the specified type in all folders.
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.
<code>isArchiv()</code>	Indicates if the variable is archived or not.
<code>isLocked()</code>	Indicates if the variable is locked or not.
<code>isVar()</code>	Indicates if the variable is in the symbol table or not.
Local	Declares one or more variables as local variables.

Command	Description
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.
Unarchiv	Moves specified variables from user data archive memory to RAM.
Unlock	Unlocks a locked variable.

Note: The **Define**, **DelVar**, and **Local** commands are available from the Program Editor's **F4** **Var** toolbar menu.

Using Local Variables in Functions or Programs

A local variable is a temporary variable that exists only while a user-defined function is being evaluated or a user-defined program is running.

Example of a Local Variable

The following program segment shows a **For...EndFor loop** (which is discussed later in this module). The variable *i* is the loop counter. In most cases, the variable *i* is used only while the program is running.

```
❶ :Local I
   :For i,0,5,1
   : Disp I
   :EndFor
   :Disp i
```

❶ Declares variable *i* as local.

Note: 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.

If you declare variable *i* as local, it is deleted automatically when the program stops so that it does not use up memory.

What Causes an Undefined Variable Error Message?

An **Undefined** variable error message displays when you evaluate a user-defined function or run a user-defined program that references a local variable that is not initialized (assigned a value).

This example is a multi-statement function, rather than a program. Line breaks are shown here, but you would type the text in the entry line as one continuous line, such as: **Define fact(n)=Func:Local...** where the ellipsis indicates the entry line text continues off-screen.

For example:

```
Define fact(n)=Func:
```

```
❶ Local m:
While n>1:
    n*m>m: n-1>n:
EndWhile:
Return m:
EndFunc
```

❶ Local variable m is not assigned an initial value.

In the example above, the local variable m exists independently of any variable m that exists outside of the function.

You Must Initialize Local Variables

All local variables must be assigned an initial value before they are referenced.

```
Define fact(n)=Func:
```

```
❶ Local m: 1>m:
While n>1:
    n*m>m: n-1>n:
EndWhile:
Return m:
EndFunc
```

❶ 1 is stored as the initial value for m .

The calculator cannot use a local variable to perform symbolic calculations.

To Perform Symbolic Calculations

If you want a function or program to perform symbolic calculations, you must use a global variable instead of a local. However, you must be certain that the global variable does not already exist outside of the program. The following methods can help.

- Refer to a global variable name, typically with two or more characters, that is not likely to exist outside of the function or program.
- Include **DeIVar** within the function or program to delete the global variable, if it exists, before referring to it. (**DeIVar** does not delete locked or archived variables.)

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

Note: See the *Technical Reference* module for syntax for all commands and functions.

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.
format	Returns an expression as a character string based on the format template (fixed, scientific, engineering, etc.)
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.

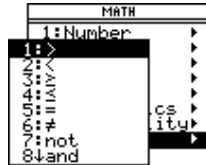
Command	Description
right	Returns a specified number of characters from the right side (end) of a string.
rotate	Rotates the characters in a string. The default is -1 (rotate right one character).
shift	Shifts the characters in a string and replaces them with spaces. The default is -1 (shift right one character and replace with one space). Examples: <code>shift("abcde",2)⇒"cde "</code> and <code>shift("abcde")⇒" abcd"</code>
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 module).

Entering a Test Operator

- Type the operator directly from the keyboard.
– or –
- Press $\boxed{2\text{nd}}$ [MATH] and select **8:Test**. Then select the operator from the menu.
– or –
- Display the built-in functions. Press: $\boxed{2\text{nd}}$ [CATALOG]
The test operators are listed near the bottom of the $\boxed{F2}$ Built-in 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).

Operator	True if:	Example
>	Greater than	$a > 8$
<	Less than	$a < 0$
\geq	Greater than or equal to	$a + b \geq 100$
\leq	Less than or equal to	$a + 6 \leq b + 1$
=	Equal	$\text{list1} = \text{list2}$
\neq	Not equal to	$\text{mat1} \neq \text{mat2}$

Note: From the keyboard, you can type:

`>=` for \geq

`<=` for \leq

`/=` for \neq

(To get the / character, press $\boxed{\div}$.)

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 $a \leq 10$
or	At least one test is true	$a \leq 0$ 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:

<code>not x > 2</code>	is true if	$x \leq 2$
	is false if	$x > 2$

Note: If you use **not** from the Home screen, it is shown as \sim in the history area. For example, **not x > 2** is shown as $\sim(x > 2)$.

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 ❶  
:EndIf
```

❶ 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:

```
:If x>5  
❶ :   Disp "x is greater than 5"  
❷ :Disp x
```

- ❶ Executed only if $x > 5$; otherwise, skipped.
- ❷ Always displays the value of x .

In this example, you must store a value to x before executing the **If** command.

Note: Use indentation to make your programs easier to read and understand.

If...Then...EndIf Structures

To execute one group of commands if a conditional test is true, use the structure:

```
:If x>5 Then  
❶ :   Disp "x is greater than 5"  
❶ :   2*x>x  
❷ :EndIf  
   :Disp x
```

- ❶ Executed only if $x > 5$.
- ❷ Displays value of:
 - $2x$ if $x > 5$
 - x if $x \leq 5$

Note: **Endif** marks the end of the **Then** block that is executed if the condition is true.

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:

```
:If x>5 Then
❶ : Disp "x is greater than 5"
❶ : 2*x>x
:Else
❷ : Disp "x is less than or
❷ equal to 5"
: 5*x>x
:EndIf
❸ :Disp x
```

- ❶ Executed only if $x > 5$.
- ❷ Executed only if $x \leq 5$.
- ❸ Displays value of:
 - $2x$ if $x > 5$
 - $5x$ if $x \leq 5$

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 the *Technical Reference* module 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*

_____ 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*

_____ 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
❶ : Goto GT5
❷ :Disp x
:-----
:-----
:Lbl GT5
:Disp "The number was > 5"
```

❶ If $x > 5$, branches directly to label GT5.

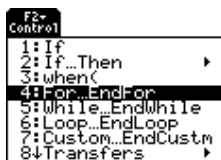
❷ For this example, the program must include commands (such as **Stop**) that prevent **Lbl GT5** from being executed if $x \leq 5$.

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.



When you select a loop, the loop command and its corresponding **End** command are inserted at the cursor location.

:For | ❶

:EndFor

❶ 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.

Note: A loop command marks the start of the loop. The corresponding **End** command marks the end of 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:

Note: The ending value can be less than the beginning value, but the increment must be negative.

For(*variable*, *begin*, *end* [, *increment*])

❶

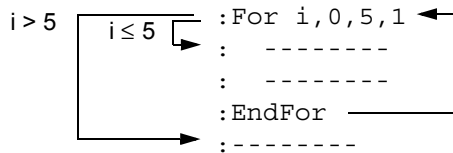
❷

❸

❹

- ❶ variable used as a counter
- ❷ counter value used the first time **For** is executed
- ❸ exits the loop when variable exceeds this value
- ❹ added to the counter each subsequent time **For** is executed

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:

```
:For i, 0, 5, 1
❶ : Disp I
:EndFor
❷ :Disp i
```

❶ Displays 0, 1, 2, 3, 4, and 5.

❷ Displays 6. When *variable* increments to 6, the loop is not executed.

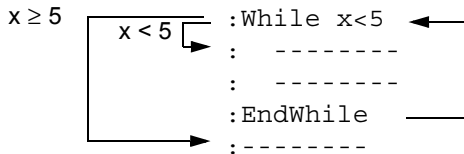
Note: You can declare the counter variable as local if it does not need to be saved after the program stops.

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:


```
❶ :0→x
   :While x<5
❷ :   Disp x
❸ :   x+1→x
   :EndWhile
❹ :Disp x
```

- ❶ Initially sets x.
- ❷ Displays 0, 1, 2, 3, and 4.
- ❸ Increments x.
- ❹ 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:

```
:0→x
:Loop
:  Disp x
:  x+1→x
❶ :  If x>5
:    Exit
:EndLoop
❷ :Disp x
```

❶ An **If** command checks the condition.

❷ Exits the loop and jumps to here when x increments to 6.

Note: The **Exit** command exits from the current loop.

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 Voyage™ 200

Programs can contain commands that change the configuration of the calculator. Because mode changes are particularly useful, the Program Editor's **Mode** toolbar menu makes it easy to enter the correct syntax for the **setMode** command.

Configuration Commands

Command	Description
getConfig	Returns a list of calculator characteristics.
getFold	Returns the name of the current folder.
getMode	Returns the current setting for a specified mode.
getUnits	Returns a list of default units.
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 , Δtbl , etc.)
setUnits	Sets default units for displayed results.
switch	Sets the active window in a split screen, or returns the number of the active window.

Note: The parameter/mode strings used in the **setMode()**, **getMode()**, **setGraph()**, and **setTable()** functions do not translate into other languages when used in a program. See the *Technical Reference* module.

Entering the SetMode 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.



Note: The **Mode** menu does not let you set the **Current Folder** mode. To set this mode, use the **setFold** command.

3. Select a mode to display a menu of its valid settings.
4. Select a setting.

The correct syntax is `:setMode ("Graph" , "FUNCTION")` inserted into your program.

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

Command	Description
getKey	Returns the key code of the next key pressed. See the <i>Technical Reference</i> module for a listing of key codes.
Input	Prompts the user to enter an expression. The expression is treated according to how it is entered. For example: <ul style="list-style-type: none">• 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 x_c and y_c (r_c and θ_c 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".

Command	Description
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.

Note: String input cannot be used in a calculation. To convert a string to a numeric expression, use the **expr** command.

Output Commands

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.
DispHome	Displays the current contents of the Home 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.

Command	Description
Pause	Suspends program execution until the user presses <code>ENTER</code> . 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.

Notes:

- In a program, simply performing a calculation does not display the result. You must use an output command.
- After **Disp** and **Output**, the program immediately continues. You may want to add a **Pause** command.

Graphical User Interface Commands

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.
CustmOn... CustmOff	Activates or removes a custom toolbar.

Command	Description
Custom...	Defines a program block that displays a custom toolbar when the user presses 2nd [CATALOG]. That toolbar remains in effect until the user presses 2nd [CATALOG] again or changes applications.
EndCustm	
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.

Notes:


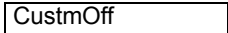
- When you run a program that sets up a custom toolbar, that toolbar is still available even after the program has stopped.
- **Request** and **Text** are stand-alone commands that can also be used outside of a dialog box or toolbar program block.

Creating a Custom Menu

The custom menu feature lets you create your own toolbar menu. A custom menu can contain any available function, instruction, or set of characters. The calculator has a default custom menu that you can modify or redefine.

Turning the Custom Menu On and Off

When you create a custom menu, you can let the user turn it on and off manually, or you can let a program turn it on and off automatically.

To:	Do this:
Turn on the custom menu	<p>From the Home screen or any other application:</p> <ul style="list-style-type: none">Press [2nd] [CATALOG]. <p>From the Home screen or a program:</p> <ul style="list-style-type: none">Execute the CustmOn command.
Turn off the custom menu	<p>From any application:</p> <ul style="list-style-type: none">Press [2nd] [CATALOG] again. – or –Go to a different application. <p>Using the default custom menu on the Home screen:</p> <ol style="list-style-type: none">Select the Tools menu:  [F7] Select 3:CustmOff. This pastes CustmOff in the entry line.Press [ENTER].  <p>You can also use CustmOff in a program.</p>

Note: When the custom menu is turned on, it replaces the normal toolbar menu. Unless a different custom menu has been created, the default custom menu is displayed.

Defining a Custom Menu

To create a custom menu, use the following general structure.

Custom

: Title *title of F1 menu*:

: Item *item 1*

: Item *item 2*

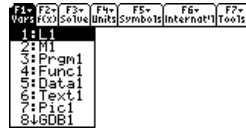
: ...

: Title *title of F2 menu*

: ...

: Title *title of F3 menu*

: ...



Note: When the user selects a menu item, the text defined by that **Item** command is pasted to the current cursor location.

For example:

```
:Custom
:Title "Vars"
:Item "L1":Item "M1":Item "Prgm1":Item "Func1":Item "Data1"
:Item "Text1":Item "Pic1":Item "GDB1":Item "Str1"

❶ :Title "f(x)"
❶ :Item "f(x)":Item "g(x)":Item "f(x,y)":Item "g(x,y)"
❶ :Item "f(x+h)":Item "Define f(x) ="

:Title "Solve"
:Item "Solve(":Item " and ":Item "{x,y}"
:Item "Solve( and ,{x,y})"
```



```

② :Title "Units"
② :Item "_m/_s^2":Item "_ft/_s^2":Item "_m":Item "_ft":Item "_l"
② :Item "_gal":Item "_\o\C":Item "_\o\F":Item "_kph":Item "_mph"

:Title "Symbols"
:Item "#":Item "\beta\:Item "?":Item "~":Item "&"

:Title "Internat'l"
:Item "\e`\:Item "\e'\":Item "\e^\":Item "\a`\"
:Item "\u`\:Item "\u^\":Item "\o^\":Item "\c,\":Item "\u..\\"

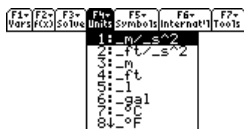
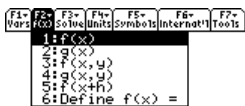
:Title "Tools"
:Item "ClrHome":Item "NewProb":Item "CustmOff"

:EndCustm

:CustmOn

```

Note: The following may be slightly different than the default custom menu on your calculator.



Note: See how "_\o\C" and "_\o\F" display as °C and °F in the menu. Similarly, see the international accented characters.

To modify the default custom menu, use **3:Restore custom default** (as described below) to get the commands for the default menu. Copy those commands, use the Program

Editor to create a new program, and paste them into the blank program. Then modify the commands as necessary.

Note: This inserts all the commands on a single line. You do not need to split them into separate lines.

You can create and use only one custom menu at a time. If you need more, write a separate program for each custom menu. Then run the program for the menu you need.

Restoring the Default Custom Menu

To restore the default:

1. From the Home screen's normal menu (not the custom menu), select **Clean Up:**

F6

2. Select **3:Restore custom default.**



This pastes the commands used to create the default menu into the entry line.

3. Press **ENTER** to execute the commands and restore the default.

When you restore the default, any previous custom menu is erased. If the previous menu was created with a program, you can run the program again if you want to reuse the menu later.

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 \leftrightarrow Table or Independent table parameters. (To set the other two table parameters, you can store the applicable values to the tblStart and Δtbl system variables.)
Table	Builds and displays a table based on one or more expressions or functions.

Graphing Commands

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.

Command	Description
Graph	Graphs one or more specified expressions, using the current graphing mode.
Input	Displays the Graph screen and lets the user update the variables x_c and y_c (r_c and θ_c 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 –	Perform all of the Zoom operations that are available from the $\boxed{F2}$ toolbar menu on the Y= Editor, Window Editor, and Graph screen.
ZoomTrig	

Note: More information is available about using **setMode**.

Graph Picture and Database Commands

Command	Description
AndPic	Displays the Graph screen and superimposes a stored graph picture by using AND logic.

Command	Description
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.

Note: For information about graph pictures and databases, also refer to *Additional Graphing Topics*.

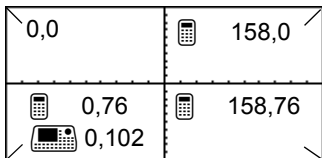
Drawing on the Graph Screen

To create a drawing object on the Graph screen, use the commands listed in this section.

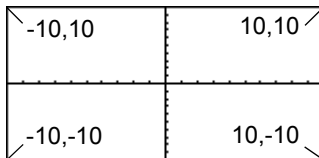
Pixel vs. Point Coordinates

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).



Pixel coordinates
(independent of viewing window)



Point coordinates
(for standard viewing window)

Note: For information about pixel coordinates in split screens, refer to the *Data/Matrix Editor* module.

Many drawing commands have two forms: one for pixel coordinates and one for point coordinates.

Note: Pixel commands start with **Pxl**, such as **PxlChg**.

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.

Command	Description
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.
DrwCtour	Draws contours in 3D graphing mode.
Shade	Draws two expressions and shades the areas where $expression1 < expression2$.

Accessing Another Voyage™ 200, a CBL 2, or a CBR

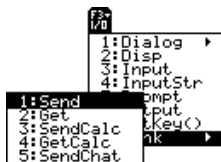
If you link two graphing calculators (described in the *Connectivity* module), programs on both units can transmit variables between them. If you link a Voyage™ 200 to a

Calculator-Based Laboratory™ (CBL 2™) or a Calculator-Based Ranger™ (CBR™), a program on the Voyage™ 200 can access the CBL 2 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 Voyage™ 200

When two calculators are linked, one acts as a receiving unit and the other as a sending unit.

Command	Description
GetCalc	Executed on the receiving unit. Sets up the unit to receive a variable via the I/O port. <ul style="list-style-type: none">• 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. <ul style="list-style-type: none">• Before the sending unit executes SendCalc, the receiving unit must execute GetCalc.

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 *Connectivity*.

Accessing a CBL 2 or CBR

For additional information, refer to the manual that comes with the CBL 2 or CBR unit.

Command	Description
Get	Gets a variable from an attached CBL 2 or CBR and stores it in the graphing calculator.
Send	Sends a list variable from the graphing calculator to the CBL 2 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 graphing calculator 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 <code>Errornum</code> to zero.
PassErr	Passes an error to the next level of the Try...EndTry block.

Example: Using Alternative Approaches

The example in the *Previews* module 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 uses **InputStr** for input, a **While...EndWhile** loop to calculate the result, and **Text** to display the result.

```
:prog1()  
:Prgm  
❶ :InputStr "Enter an integer",n  
❷ :expr(n)→n  
  :0→temp:1→I  
❸ :While i≤n  
  : temp+i→temp  
  : i+1→I  
❹ :EndWhile  
❺ :Text "The answer is "&string(temp)  
  :EndPrgm
```

- ❶ Prompts for input on Program I/O screen.
- ❷ Converts string entered with **InputStr** to an expression.
- ❸ Loop calculation.
- ❹ Displays output in a dialog box.

Note: For ≤, type 0 (zero). For &, press:

2nd H

Example 2

This example uses **Prompt** for input, **Lbl**, and **Goto** to create a loop, and **Disp** to display the result.

```
:prog2()  
:Prgm  
❶ :Prompt n  
  :0→temp:1→I  
❷ :Lbl top  
  : temp+i→temp  
  : i+1→I  
  : If i≤n  
❷ :   Goto top  
❸ :Disp temp  
  :EndPrgm
```

- ❶ Prompts for input on Program I/O screen.
- ❷ Loop calculation.
- ❸ Displays output on Program I/O screen.

Note: Because **Prompt** returns n as a number, you do not need to use **expr** to convert n .

Example 3

This example uses **Dialog...EndDlog** to create dialog boxes for input and output. It uses **Loop...EndLoop** to calculate the result.

```
:prog3()  
:Prgm  
❶ :Dialog  
: : Title "Enter an integer"  
: : Request "Integer",n  
❷ :EndDlog  
❷ :expr(n)→n  
:0→temp:0→I  
❸ :Loop  
: : temp+i→temp  
: : i+1→I  
: : If i>n  
: : Exit  
❸ :EndLoop  
❹ :Dialog  
: : Title "The answer is"  
: : Text string(temp)  
❹ :EndDlog  
:EndPrgm
```

- ❶ Defines a dialog box for input.
- ❷ Converts string entered with **Request** to an expression.
- ❸ Loop calculation.
- ❹ Defines a dialog box for output.

Example 4

This example uses built-in functions to calculate the result without using a loop.

```
:prog4 ()  
:Prgm  
❶ :Input "Enter an integer",n  
❷ :sum(seq(i,i,1,n))→temp  
❸ :Disp temp  
:EndPrgm
```

- ❶ Prompts for input on Program I/O.
- ❷ Calculates sum.
- ❸ Displays output on Program I/O screen.

Note: Because **Input** returns *n* as a number, you do not need to use **expr** to convert *n*.

Function	Used in this example to:
----------	--------------------------

seq	Generate the sequence of integers from 1 to n .
------------	---

seq(*expression*, *var*, *low*, *high* [,*step*])

❶ ❷ ❸ ❸ ❹

- ❶ expression used to generate the sequence
- ❷ variable that will be incremented
- ❸ initial and final values of *var*
- ❹ increment for *var* ; if omitted, uses 1

sum	Sum the integers in the list generated by seq .
------------	--

Assembly-Language Programs

You can run programs written for the Voyage™ 200 Graphing Calculator in assembly language. Typically, assembly-language programs run much faster and provide greater control than the keystroke programs that you write with the built-in Program Editor.

Where to Get Assembly-Language Programs

Assembly-language programs, as well as keystroke programs, are available on the Texas Instruments web site at education.ti.com.

The programs available from this site provide additional functions or features that are not built into the Voyage™ 200. Check the Texas Instruments web site for up-to-date information.

After downloading a program from the web to your computer, use a USB cable or TI-GRAPH LINK™ computer-to-calculator cable and TI Connect software to send the program to your Voyage™ 200.

For Flash App installation instructions, see education.ti.com/guides.

Note about TI-GRAPH LINK

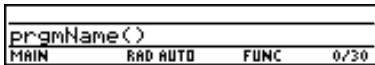
If you have a TI-GRAPH LINK™ computer-to-calculator cable and software for the TI-89 or TI-92 Plus, be aware that the TI-GRAPH LINK software is not compatible with the Voyage™ 200. The cable, however, works with all units. Use TI Connect software on your computer.

You can purchase computer-to-calculator and unit-to-unit cables from the TI Online Store at education.ti.com/buy.

Running an Assembly-Language Program

After a Voyage™ 200 assembly-language program is stored on your unit, you can run the program from the Home screen just as you would any other program.

- If the program requires one or more arguments, type them within the (). Refer to the program's documentation to find out about required arguments.



- If the program is not in the current folder, be sure to specify the pathname.

You can call an assembly-language program from another program as a subroutine, delete it, or use it the same as any other program.

Shortcuts to Run a Program

On the Home screen, you can use keyboard shortcuts to run up to **six** user-defined or assembly-language programs. However, the programs must have the following names.

On Home screen, press:	To run a program, if any, named:
◆ 1	kbdprgm1()
⋮	⋮
◆ 6	kbdprgm6()

The programs must be stored in the `MAIN` folder. Also, you cannot use a shortcut to run a program that requires an argument.

If you have a program with a different name and you would like to run it with a keyboard shortcut, copy or rename the existing program to `kbdprgm1()`, etc.

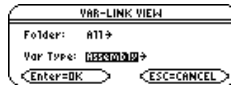
You Cannot Edit an Assembly-Language Program

You cannot use your Voyage™ 200 to edit an assembly-language program. The built-in Program Editor will not open assembly-language programs.

Displaying a List of Assembly-Language Programs

To list the assembly-language programs stored in memory:

1. Display the **VAR-LINK** screen (`[2nd] [VAR-LINK]`).
2. Press `[F2] View`.
3. Select the applicable folder (or All folders) and set **Var Type = Assembly**.
4. Press `[ENTER]` to display the list of assembly-language programs.



VAR-LINK [ALL]

F1- Manas	F2 View...	F3-F4 [Link]	F5- [AT]	F6 Contents...
CP1PES	ASM	553		
window	ASM	269		

Note: Assembly-language programs have an `ASM` data type.

For Information about Writing an Assembly-Language Program

The information required to teach a novice programmer how to write an assembly-language program is beyond the scope of this book. However, if you have a working knowledge of assembly language, please check the Texas Instruments web site (education.ti.com) for specific information about how to access Voyage™ 200 features.

The graphing calculator also includes an **Exec** command that executes a string consisting of a series of Motorola 68000 op-codes. These codes act as another form of an assembly-language program. Check the Texas Instruments web site for available information.

Note: You must use a computer to write assembly-language programs. You cannot create assembly-language programs from the calculator keyboard.

Warning: **Exec** gives you access to the full power of the microprocessor. Please be aware that you can easily make a mistake that locks up the calculator and causes you to lose your data. We suggest you make a backup of the calculator contents before attempting to use the **Exec** command.

Text Editor

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 the Text Editor icon. Press **[ENTER]**.



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.



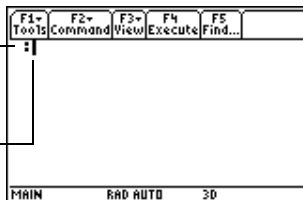
Item	Description
Type	Automatically set as Text and cannot be changed.
Folder	Shows the folder in which the text variable will be stored. For information about folders, refer to the <i>Calculator Home Screen module</i> . To use a different folder, press ⏏ to display a menu of existing folders. Then select a folder.

Item	Description
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.

- Press **ENTER** (after typing in an input box such as **Variable**, you must press **ENTER** twice) to display an empty Text Editor screen.

A colon marks the beginning of a paragraph.

The blinking cursor shows where typed text will appear.



You can now use the Text Editor as described in the remaining sections of this module.

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.

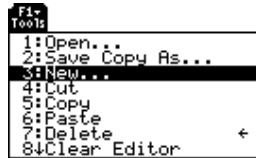
Resuming the Current Session

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, launch Text Editor again 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**.
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**.
— or —

From any application, launch Text Editor again and select **2:Open**.

2. Select the applicable folder and text variable.
3. Press **[ENTER]**.



Note: By default, **Variable** shows the first existing text variable in alphabetic order.

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**.
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 *Memory and Variable Management*.

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.

All text paragraphs begin with a space and a colon.

The beginning space is used in command scripts and lab reports.



Blinking text cursor







You do not need to press **ENTER** at the end of each line. At the end of a line, the next character you type 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.









Using a TI-GGRAPH LINK™ unit to computer cable and TI Connect™ software with the Voyage™ 200 Graphing Calculator, you can use the computer keyboard to type a text file and then send that file to the Voyage™ 200. This is useful if you need to create a lengthy text file.

For information about obtaining cables or updated TI Connect™ software, check the TI web site at education.ti.com, or contact Texas Instruments at TI-Cares™.

Notes:

- Use the cursor pad to scroll through a session or position the text cursor.
- Press **[2nd]**  or **[2nd]**  to scroll up or down one screen at a time, and   or   to go to the top or bottom of the text session.

Typing Alphabetic Characters

To:	Press:
Type a single lowercase alpha character.	 the letter key
Type a single uppercase alpha character.	  and then the letter key (status line shows )
Type a space.	 spacebar
Turn on lowercase alpha-lock.	 (no action needed)
Turn on uppercase ALPHA-lock.	 [2nd] [CAPS]
Turn off alpha-lock.	 [2nd] [CAPS] (turns off uppercase lock)

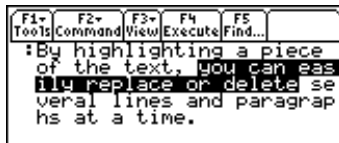
Deleting Characters

To delete:	Press:
The character to the left of the cursor	← or [F1] 7
The character to the right of the cursor	◆ [DEL] (same as ◆ ←)
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.


Highlighting Text

To:	Do this:
Highlight text	<p>Move the cursor to the beginning or end of the text. Hold [↑] and press:</p> <ul style="list-style-type: none">◀ 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.




Note: To remove highlighting without replacing or deleting, move the cursor.

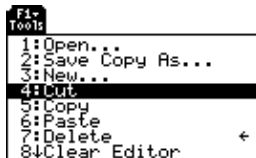
Replacing or Deleting Highlighted Text

To:	Do this:
Replace highlighted text	Type the new text.
Delete highlighted text	Press  .

Cutting, Copying, and Pasting Text

Cutting and copying both place highlighted text into the clipboard of the Voyage™ 200. 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 .
3. Select the applicable menu item.
 - To move the text, select **4:Cut**.
— or —
 - To copy the text, select **5:Copy**.



Note: You can press:



 X,  C,  V

to cut, copy, and paste without having to use the  toolbar menu.

4. Move the text cursor to the location where you want to insert the text.
5. Press **[F1]** and then select **6:Paste**.

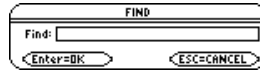
You can use this general procedure to cut, copy, and paste text:

- 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.

Note: The **FIND** dialog box retains the last search text you entered. You can type over it or edit it.

4. Press **[ENTER]** twice.

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 Voyage™ 200 is in insert mode. To toggle between insert and overwrite mode, press **[2nd] [INS]**.

If the Voyage™ 200 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.

Note: Look at the shape of the cursor to see if you're in insert or overwrite mode.

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 characters from the keyboard. To see which characters are available from the keyboard, you can display a map that shows the characters and their corresponding keys.

Selecting Characters from 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.

Note: For accented characters, select International. Commonly used international characters are also available from the default custom menu (**[2nd] [CUSTOM]**).



↓ indicates that you can scroll.

Displaying the Keyboard Map

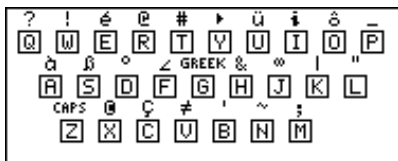
The keyboard map shows several shortcuts that let you enter certain special characters from the keyboard. It also shows some shortcuts for other calculator features.

The keyboard map does not display all available shortcuts. Refer to the inside front and the inside back covers of this guidebook for a complete list of shortcut keys.

To access the shortcuts, first press the **2nd** key. Some special characters are marked on the keyboard, but most are not.

On the Voyage™ 200:

- Press **◆** [**KEY**] to display the keyboard map.
- Press **ESC** to exit the map.



Voyage™ 200 Graphing Calculator Keyboard map

Voyage™ 200 keyboard map feature shortcuts:

- GREEK (**2nd** **G**) — Accesses the Greek character set (described later in this section).

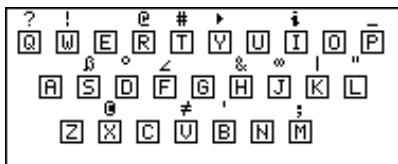
- CAPS ([2nd] [CAPS])— Turns Caps Lock on and off.
- Accent marks — (é, ü, ô, à, ç, and ~) are added to the next letter you press (described later in this section).

Typing Special Symbols from the Keyboard

Note: To help you find the applicable keys, these maps show only the special symbols.

On the Voyage™ 200:

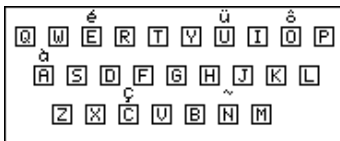
- ▶ Press [2nd] and then the key for the symbol. For example: [2nd] H displays &.



These special symbols are not affected by whether Caps Lock is on or off.

Typing Accent Marks from the Voyage™ 200 Keyboard

Pressing an accent mark key does not display an accented letter. The accent mark will be added to the next letter you press.



1. Press **[2nd]** and then the key for the accent mark.

Note: To help you find the applicable keys, this map shows only the accent mark keys.

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.

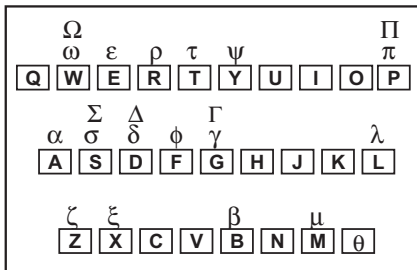
Accent Mark	Valid Letters (lowercase or uppercase)	Examples
´	A, E, I, O, U, Y	é, É
¨	A, E, I, O, U, y (but not Y)	ü, Ü
ˆ	A, E, I, O, U	ô, Ô
`	A, E, I, O, U	à, À
ç	C	ç, Ç
˜	A, O, N	ñ, Ñ

Typing Greek Letters from the Keyboard

Press the key combination that accesses the Greek character set on your calculator. Then select the applicable alpha character on the keyboard to enter a Greek letter.

On the Voyage™ 200:

- ▶ Press **[2nd] G** to access the Greek character set.



Note: If you press a key combination that does not access a Greek letter, you get the normal letter for that key. Your calculator does not display a map of Greek letters; the map shown here is for reference only.

Several keys let you access lowercase and uppercase Greek letters. For example:

On the Voyage™ 200:

- Press **[2nd] G** to access the Greek character set.
- Press **[2nd] G + letter** to access lowercase Greek letters. Example:
[2nd] G W displays ω

- Press **2nd** **G** **↑** + letter to access uppercase Greek letters. Example:
2nd **G** **↑** W displays Ω

For a List of All Special Characters

For a list of all special characters, refer to the *Technical Reference* module.

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

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).

Note: This does not insert a new line for the command, it simply marks an existing line as a command line.

4. Type a command just as you would on the Home screen.

The line can contain only the command, with no additional text.

Note: You can mark a line as a command either before or after typing the command on that line.



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]**.

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.

Note: To examine the result on the Home screen, use a split screen or press



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 the *Calculator Home Screen* module.

Example

1. Type your script. Press **[F2]** and select **1:Command** to mark the command lines.

```
F1- F2- F3- F4- F5-
Tools Command View Execute Find...
1:undo for graph
2:x^3-2x^2x-1+f(x)
C:zeros(f(x),x)
C:d(f(x),x)>df(x)
C:zeros(df(x),x)
C:d(df(x),x)>ddf(x)
C:-4→xmin:4→xmax
C:-10→ymin:10→ymax
C:Graph f(x)
```

2. Press **[F3]** and select **1:Script view**.

3. Move the cursor to the first command line. Then press **[F4]** to execute the command.

Note: Some commands take longer to execute. Wait until the **Busy** indicator disappears before pressing **[F4]** again.

```
F1- F2- F3- F4- F5-
Tools Command View Execute Find...
C:d(df(x),x)>ddf(x)
C:-4→xmin:4→xmax
C:-10→ymin:10→ymax
C:Graph f(x)
```

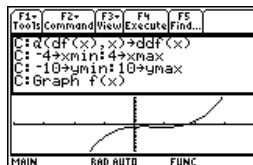
■ -4 → xmin : 4 → xmax 4
■ -10 → ymin : 10 → ymax 10

4. Continue using **[F4]** to execute each command, but stop just before executing the Graph command.

5. Execute the Graph command.

Note: In this example, the Graph command displays the Graph screen in place of the Home screen.

6. Press **[F3]** and select **2:Clear split** to return to a full screen Text Editor.



Numeric Solver

Displaying the Solver and Entering an Equation

After you display the Numeric Solver, start by entering the equation that you want to solve.

Displaying the Numeric Solver

To display the Numeric Solver, press **[APPS]** and then select the Numeric Solver icon. Press **[ENTER]**.

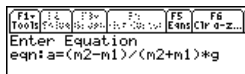


f(x)=0
Numeric So...

The Numeric Solver screen shows the last entered equation, if any.

Entering an Equation

On the **eqn:** line, type in your equation.



F1 F2 F3 F4 FS F6
1000 1000 1000 1000 1000 1000
Enter Equation
eqn: a=(m2-m1)/(m2+m1)*g

You can:

Type an equation directly.

For example:

$a=(m_2-m_1)/(m_2+m_1)*g$
 $a+b=c+\sin(d)$

You can:

Refer to a function or equation defined elsewhere.

Notes:

- Do not use system function names (such as **y1(x)** or **r1(θ)**) as simple variables (**y1** or **r1**).
- Be careful with implied multiplication. For example, **a(m2+m1)** is treated as a function reference, not as **a*(m2+m1)**.

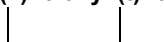
For example:

Suppose you defined **y1(x)** on either the:

- Y= Editor:
y1(x)=1.25x*cos(x)
– or –
- Home screen:
Define y1(x)=1.25x*cos(x)

In the Numeric Solver, you then would enter:

y1(x)=0 or **y1(t)=0**, etc.

 The argument does not have to match the one used to define the function or equation.

Type an expression without an = sign.

Note: When you define the variables, you can either define **exp** or solve for it.

e+f–ln(g)

After you press **ENTER**, the expression is set equal to a system variable called **exp** and entered as:

exp=e+f–ln(g)

Recall a previously entered equation or open a saved equation.

Note: After you press **ENTER** the current equation is stored automatically to the system variable **eqn**.

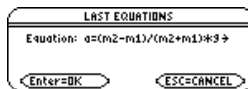
Refer to the applicable heading later in this section.

Recalling Previously Entered Equations

Your most recently entered equations (up to 11 with the default setting) are retained in memory. To recall one of these equations:

1. From the Numeric Solver screen, press **F5**.

A dialog box displays the most recently entered equation.

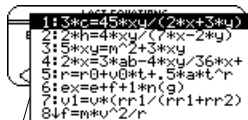


2. Select an equation.

- To select the displayed equation, press **ENTER**.
- To select a different equation, press **⏵** to display a list. Then select the one you want.

Note: You can specify how many equations are retained. From the Numeric Solver, press **F1** and select **9:Format** (or use **⏵ F**). Then select a number from 1 through 11.

3. Press **ENTER**.



Only unique equations are listed. If you re-enter the same equation 5 times, it appears only once.

Saving Equations for Future Use

Because the number of equations that you can recall with **F5 Eqns** is limited, a particular equation may not be retained indefinitely.

To store the current equation for future use, save it to a variable.

1. From the Numeric Solver screen, press **F1** and select **2:Save Copy As**.
2. Specify a folder and a variable name for the equation.
3. Press **ENTER** twice.



Note: An equation variable has an EXPR data type, as shown on the MEMORY and VAR-LINK screens.

Opening a Saved Equation

To open a previously saved equation variable:

1. From the Numeric Solver screen, press **F1** and select **1:Open**.



- Select the applicable folder and equation variable.
- Press **ENTER**.



Variable eqn contains the current equation; it always appears alphabetically in the list.

Defining the Known Variables

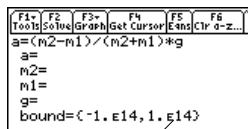
After you type an equation in the Numeric Solver, enter the applicable values for all variables except the unknown variable.

Defining the List of Variables

After typing your equation on the **eqn:** line, press **ENTER** or \odot .

The screen lists the variables in the order they appear in the equation. If a variable is already defined, its value is shown. You can edit these variable values.

Note: If an existing variable is locked or archived, you cannot edit its value.

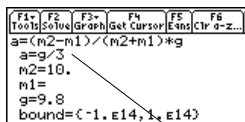


The solution must be within the specified bounds, which you can edit.

Enter a number or expression for all variables except the one you want to solve for.

Notes and Common Errors

- If you define a variable:
 - In terms of another variable in the equation, that variable must be defined first.
 - In terms of another variable that is not in the equation, that variable must already have a value; it cannot be undefined.
 - As an expression, it is evaluated when you move the cursor off the line. The expression must evaluate to a real number

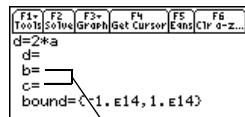


```
F1- F2- F3- F4- F5- F6-
Tools Solve Graph Get Cursor EAns Clr a-z...
a=(m2-m1)/(m2+m1)*g
a=g/3
m2=10.
m1=
g=9.8
bound=C-1. E14, 1. E14
```

Since a is defined in terms of g, you must define g before a. When you move the cursor to another line, g/3 is evaluated.

- If the equation contains a variable already defined in terms of other variables, those other variables are listed.

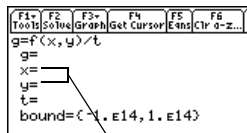
Note: When you assign a value to a variable in the Numeric Solver, that variable is defined globally. It still exists after you leave the solver.



```
F1- F2- F3- F4- F5- F6-
Tools Solve Graph Get Cursor EAns Clr a-z...
d=2*a
d=
b=
c=
bound=C-1. E14, 1. E14
```

If variable a was defined previously as $b+c>a$, then b and c are listed instead of a.

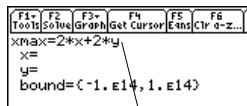
- If you refer to a previously defined function, any variables used as arguments in the function call are listed, not the variables used to define the function.



If $f(a,b)$ was defined previously as $\sqrt{a^2+b^2}$ and your equation contains $f(x,y)$, then x and y are listed, not a and b .

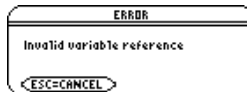
- If the equation contains a system variable (x_{\min} , x_{\max} , etc.), that variable is not listed. The solver uses the system variable's existing value.

Note: You cannot solve for a system variable other than \exp . Also, if the equation contains a system variable, you cannot use $\boxed{F3}$ to graph.

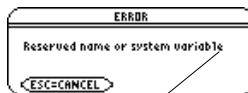


In the standard viewing window, $x_{\max}=10$.

- Although you can use a system variable in the equation, an error occurs if you use $\boxed{F3}$ to graph the solution.



- If you see the error shown to the right, delete the entered variable value. Then edit the equation to use a different variable.



For example, $y_1(x)$ is undefined and you use y_1 .

Note: This error occurs if you use a reserved name incorrectly or refer to an undefined system function as a simple variable without parentheses.

Editing the Equation

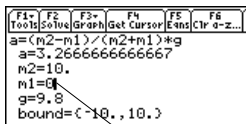
In the Numeric Solver, press \odot until the cursor is on the equation. The screen automatically changes to show only the **eqn:** line. Make your changes, and then press

ENTER or \odot to return to the list of variables.

Specifying an Initial Guess and/or Bounds (Optional)

To find a solution more quickly or to find a particular solution (if multiple solutions exist), you can optionally:

- Enter an initial guess for the unknown variable. The guess must be within the specified bounds.
- Enter lower and upper bounds close to the solution.



Initial guess must be within the bounds.

For the bounds, you can also enter variables or expressions that evaluate to appropriate values (**bound={lower,upper}**) or a valid list variable that contains a two-element list (**bound=list**). The bounds must be two floating point elements with the first one less than or equal to the second one.

Note: You can also select an initial guess graphically.

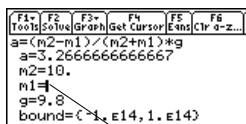
Solving for the Unknown Variable

After you type an equation in the Numeric Solver and enter values for the known variables, you are ready to solve for the unknown variable.

Finding the Solution

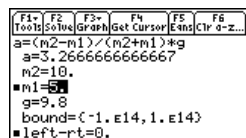
With all known variables defined:

1. Move the cursor to the unknown variable.



Put the cursor on the variable you want to solve for.

2. Press **[F2] Solve**.
3. A **■** marks the solution and **left-right**. The **■** disappears when you edit a value, move the cursor to the equation, or leave the solver.



Note: To stop (break) a calculation, press **[ON]**. The unknown variable shows the value being tested when the break occurred.

Using the solution and your entered values, the left and right sides of the equation are evaluated separately. **left-rt** shows the difference, which indicates the solution's accuracy. The smaller the value, the more accurate the solution. If the solution is precise, **left-rt=0**.

If you:**Do this:**

Want to solve for other values

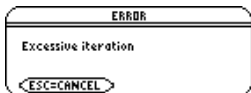
Edit the equation or variable values.

Want to find a different solution for an equation with multiple solutions

Enter an initial guess and/or a new set of bounds close to the other solution.

See the message:

Press **[ESC]**. The unknown variable shows the value being tested when the error occurred.



- The **left-rt** value may be small enough for you to accept the result.
- If not, enter a different set of bounds.

Note: An iterative process is used to solve an equation. If the iterative process cannot converge on a solution, this error occurs.

Graphing the Solution

You can graph an equation's solutions any time after defining the known variables, either before or after you solve for the unknown variable. By graphing the solutions, you can

see how many solutions exist and use the cursor to select an accurate initial guess and bounds.

Displaying the Graph

In the Numeric Solver, leave the cursor on the unknown variable. Press $\boxed{F3}$ and select:

- 1:Graph View
- or –
- 3:ZoomStd
- or –
- 4:ZoomFit

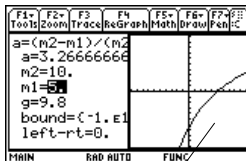


Graph View uses the current Window variable values.

For information about **ZoomStd** and **ZoomFit**, refer to *Basic Function Graphing*.

The graph is shown in a split screen, where:

- The unknown variable is plotted on the x axis.
- **left-rt** is plotted on the y axis.



Solutions for the equation exist at **left-rt=0**, where the graph crosses the x axis.

The current graph format settings are used.

Note: For more information, refer to the *Split Screens* module.

You can explore the graph by using the free-moving cursor, tracing, zooming, etc., as described in *Basic Function Graphing*.

How the Graph Affects Various Settings

When you use the Numeric Solver to display a graph:

- The following modes are changed automatically to these settings:

Mode	Setting
Graph	FUNCTION Any functions selected in the Y= Editor will not be graphed.
Split Screen	LEFT-RIGHT
Number of Graphs	1

Note: If you were previously using different mode settings, you will need to reselect those settings manually.

- All stat plots are deselected.
- After you leave the Numeric Solver, the Graph screen may continue to display the equation's solution, ignoring any selected Y= functions. If so, display the Y= Editor and then return to the Graph screen. Also, the graph is reset when you change the Graph mode or use **ClrGraph** from the Home screen ($\boxed{F4}$ 5) or a program.

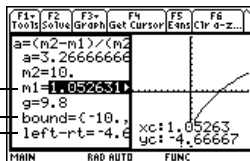
Selecting a New Initial Guess from the Graph

To use the graph cursor to select an initial guess:

1. Move the cursor (either free-moving or trace) to the point that you want to use as the new guess.
2. Use $\boxed{2nd} \boxed{[+=]}$ to make the Numeric Solver screen active.
3. Make sure the cursor is on the unknown variable, and press $\boxed{F4}$.

Note: Cursor coordinate xc is the unknown variable value, and yc is the **left-rt** value.

4. Press $\boxed{F2}$ to re-solve the equation.



$\boxed{F4}$ sets the graph cursor's xc value as an initial guess and the yc value as left-rt. The graph's $xmin$ and $xmax$ values are set as the bounds.

Returning to a Full Screen

From the split screen:

- To display the Numeric Solver full screen, use $\boxed{2nd} \boxed{[+=]}$ to make the solver screen active, press $\boxed{F3}$, and then select **2:Clear Graph View**.
– or –
- To display the Home screen, press $\boxed{2nd} \boxed{[QUIT]}$ twice.

Clearing Variables Before Leaving the Numeric Solver

When you solve an equation, its variables still exist after you leave the Numeric Solver. If the equation contains single-character variables, their values may inadvertently affect later symbolic calculations. Before leaving the Numeric Solver, you may want to:

1. Press:



to clear all single-character variables in the current folder.

2. Press  to confirm the action.

The screen returns to the solver's **eqn:** line.

Note: Any time you want to clear single-character variables listed in the solver, use:



Number Bases

Entering and Converting Number Bases

Regardless of the Base mode, you must always use the appropriate prefix when entering a binary or hexadecimal number.

Entering a Binary or Hexadecimal Number

To enter a binary number, use the form:

0b *binaryNumber* (for example: **0b11100110**)

└─ Binary number with up to 32 digits
└─ Zero, not the letter O, and the letter b

To enter a hexadecimal number, use the form:

0h *hexadecimalNumber* (for example: **0h89F2C**)

└─ Hexadecimal number with up to 8 digits
└─ Zero, not the letter O, and the letter h

Note: You can type the **b** or **h** in the prefix, as well as hex characters **A – F**, in uppercase or lowercase.

If you enter a number without the **0b** or **0h** prefix, such as 11, it is always treated as a decimal number. If you omit the **0h** prefix on a hexadecimal number containing **A – F**, all or part of the entry is treated as a variable.

Converting between Number Bases

Use the **►** conversion operator

integerExpression **► Bin**

integerExpression **► Dec**

integerExpression **► Hex**

For **►**, press **[2nd] [►]**. Also, you can select base conversions from the MATH/Base menu.

For example, to convert 256 from decimal to binary:

256 **► Bin**

Note: If your entry is not an integer, a Domain error is displayed.

To convert 101110 from binary to hexadecimal:

0b101110 **► Hex**

For a binary or hex entry, you must use the 0b or 0h prefix.

■ 256►Bin	0b100000000
■ 0b101110►Hex	0h2E
0b101110►hex	0h2E
MODE	RAD AUTO FUNC 2/20

Results use the 0b or 0h prefix to identify.

Alternate Method for Conversions

Instead of using **►**, you can:

1. Use **[MODE]** to set the **Base** mode to the base that you want to convert to.

If Base mode = BIN:

■ 256	0b1000
256	
MODE	RAD AUTO FUNC

- From the Home screen, type the number that you want to convert (using the correct prefix) and press **[ENTER]**.

If Base mode = HEX:

0b101110	0h2E		
0b101110			
MAIN	RAD AUTO	FUNC	1/30

Performing Math Operations with Hex or Bin Numbers

For any operation that uses an integer number, you can enter a hexadecimal or binary number. Results are displayed according to the Base mode. However, results are restricted to certain size limits when Base = HEX or BIN.

Setting the Base Mode for Displayed Results

- Press **[MODE]** **[F2]** to display **Page 2** of the **MODE** screen.
- Scroll to the **Base** mode, press **[▶]**, and select the applicable setting.
- Press **[ENTER]** to close the **MODE** screen.



The **Base** mode controls the displayed format of integer results only.

Note: The Base mode affects output only. You must always use the **0h** or **0b** prefix to enter a hex or binary number.

Fractional and floating-point results are always shown in decimal form.

Dividing When Base = HEX or BIN

When Base=HEX or BIN, a division result is displayed in hexadecimal or binary form only if the result is an integer.

To ensure that division always produces an integer, use **intDiv()** instead of $\frac{\square}{\square}$.

If Base mode = HEX:

■	0b101101 - 0b101	0h28
■	254 + 1	0hFF
■	0h5A2C · 6	0h21D08
■	0hA8F + 0b1001101101	
■	0hC45A + 0h6FD2	0hCFC
	0hc45a+0h6fd2	0h1342C
MAIN	RAD AUTO	FUNC 5/20

0h prefix in result identifies the base.

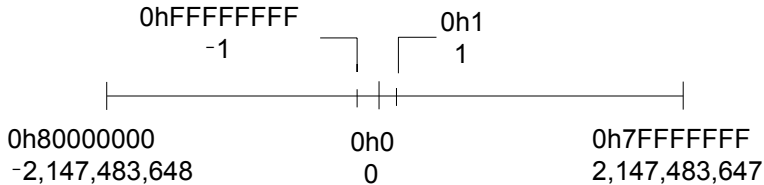
If Base mode = HEX:

■	0hFF	255
■	0h2	2
■	0hFF	127.5
■	0h2	
■	intDiv(0hFF, 0h2)	0h7F
	intDiv(0hFF, 0h2)	
MAIN	RAD AUTO	FUNC 3/20

Press \blacklozenge **ENTER** to display the result in APPROXIMATE form.

Size Limitations When Base = HEX or BIN

When Base=HEX or BIN, an integer result is stored internally as a signed, 32-bit binary number, which uses the range (shown in hexadecimal and decimal):



If a result's magnitude is too large to be stored in a signed, 32-bit binary form, a symmetric modulo operation brings the result into the range. Any number greater than 0h7FFFFFFF is affected. For example, 0h80000000 through 0hFFFFFFF become negative numbers.

Comparing or Manipulating Bits

The following operators and functions let you compare or manipulate bits in a binary number. You can enter an integer in any number base. Your entries are converted to binary automatically for the bitwise operation, and results are displayed according to the Base mode. Boolean Operations

Operator with syntax	Description
not <i>integer</i>	Returns the one's complement, where each bit is flipped.
(-) <i>integer</i>	Returns the two's complement, which is the one's complement + 1.

Operator with syntax	Description
<i>integer1</i> and <i>integer2</i>	In a bit-by-bit and comparison, the result is 1 if both bits are 1; otherwise, the result is 0. The returned value represents the bit results.
<i>integer1</i> or <i>integer2</i>	In a bit-by-bit or comparison, the result is 1 if either bit is 1; the result is 0 only if both bits are 0. The returned value represents the bit results.
<i>integer1</i> xor <i>integer2</i>	In a bit-by-bit xor comparison, the result is 1 if either bit (but not both) is 1; the result is 0 if both bits are 0 or both bits are 1. The returned value represents the bit results.

Note: You can select these operators from the MATH/Base menu. For an example using each operator, refer to the *Technical Reference* module.

Suppose you enter:

0h7AC36 **and** 0h3D5F

Internally, the hexadecimal integers are converted to a signed, 32-bit binary number.

Then corresponding bits are compared.

If Base mode = HEX:

■ 0h7AC36 and 0h3D5F			
			0h2C16
0h7ac36 and 0h3d5f			
MAIN	RAD	AUTO	FUNC 1/20

If Base mode = BIN:

■ 0h7AC36 and 0h3D5F			
			0b10110000010110
0h7ac36 and 0h3d5f			
MAIN	RAD	AUTO	FUNC 1/20

```

0h7AC36 = 0b00000000000001111010110000110110
  and
0h3D5F   0b00000000000000000000000011110101011111
          0b0000000000000000000010110000010110 = 0h2C16

```

└─ Leading zeros are not shown in the result.

Note: If you enter an integer that is too large to be stored in a signed, 32-bit binary form, a symmetric modulo operation brings the value into the range.

The result is displayed according to the Base mode.

Rotating and Shifting Bits

Function with syntax	Description
rotate (<i>integer</i>) – or – rotate (<i>integer</i> , <i>#ofRotations</i>)	If <i>#ofRotations</i> is: <ul style="list-style-type: none"> • omitted — bits rotate once to the right (default is -1). • negative — bits rotate the specified number of times to the right. • positive — bits rotate the specified number of times to the left. In a right rotation, the rightmost bit rotates to the leftmost bit; vice versa for a left rotation.

Function with syntax	Description
shift (<i>integer</i>) – or – shift (<i>integer</i> , # <i>ofShifts</i>)	<p>If #<i>ofShifts</i> is:</p> <ul style="list-style-type: none"> omitted — bits shift once to the right (default is -1). negative — bits shift the specified number of times to the right. positive — bits shift the specified number of times to the left. <p>In a right shift, the rightmost bit is dropped and 0 or 1 is inserted to match the leftmost bit. In a left shift, the leftmost bit is dropped and 0 is inserted as the rightmost bit.</p>

Suppose you enter:

shift(0h7AC36)

If Base mode = HEX:

■	shift(0h7AC36)	0h3D61B
	shift(0h7ac36)	
MIN	RAD AUTO	FUNC 1/30

Internally, the hexadecimal integer is converted to a signed, 32-bit binary number.

If Base mode = BIN:

Then the shift is applied to the binary number.

■	shift(0h7AC36)	0b11110101100011011
	shift(0h7ac36)	
MIN	RAD AUTO	FUNC 1/30

Each bit shifts to the right.

7AC36 = 0b000000000000001111010110000110110

Inserts 0 if leftmost bit is 0,
or 1 if leftmost bit is 1

Dropped

b000000000000000111101011000011011 = 0h3D61B

Leading zeros are not shown in the result.

The result is displayed according to the Base mode.

Note: If you enter an integer that is too large to be stored in a signed, 32-bit binary form, a symmetric modulo operation brings the value into the range.

Memory and Variable Management

Checking and Resetting Memory

The **MEMORY** screen shows the amount of memory (in bytes) used by all variables in each data type, regardless of whether the variables are stored in RAM or the user data archive. You can also use this screen to reset the memory.

Displaying the MEMORY Screen

Press [2nd] [MEM]. (The numbers on your **MEMORY** screen may vary from those shown.)

Data Type	Value	Data Type	Value
Expr	61	Text	74
List	80	DOB	0
Matrix	238	Data	0
Function	0	Other	154
Prgm/Asn	269	History	1572
Picture	2241	System	126062
String	0	FlashApp	894319
		Archive	219
		RAM free	131312
		Flash ROM free	1923452

Prgm/Asn: Includes programs written for the Voyage™ 200 as well as any assembly-language programs you have loaded.

History: Size of history pairs saved in the Home screen's history area.

FlashApp: Size of Flash applications.

RAM free: Free space in RAM.

Flash ROM free: Free space in Flash ROM.

Note: To display the size of individual variables and determine if they are in the user data archive, use the **VAR-LINK** screen.

To close the screen, press [ENTER]. To reset the memory, use the following procedure.

Resetting the Memory

From the **MEMORY** screen:

1. Press **[F1]**.
2. Select the applicable item.



Item	Description
RAM	1:All RAM: Resetting RAM erases all data and programs from RAM. 2:Default: Resets all system variables and modes to their original factory settings. This does not affect any user-defined variables, functions, or folders.
Flash ROM	1:Archive: Resetting Archive erases all data and programs from Flash ROM. 2:Flash Apps: Resetting Flash Apps erases all Flash applications from Flash ROM. 3:Both: Resetting both erases all data, programs, and Flash applications from Flash ROM.
All Memory	Resetting will delete all data, programs, and Flash applications from RAM and Flash ROM.

Important: To delete individual (instead of all) variables, use **VAR-LINK**.

3. When prompted for confirmation, press **[ENTER]**.

The Voyage™ 200 displays a message when the reset is complete.

Note: To cancel the reset, press **[ESC]** instead of **[ENTER]**.

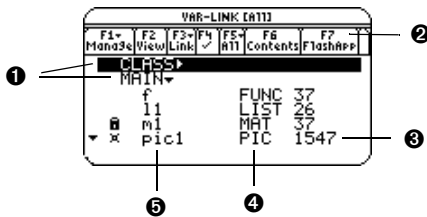
4. Press **[ENTER]** to acknowledge the message.

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.

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.



- 1 Folder names (alphabetically listed)
- 2 Shows installed Flash applications
- 3 Size in bytes
- 4 Data type
- 5 Variable names (alphabetically listed)

This...	Indicates this...
▶	Collapsed folder view (to right of folder name).

This...	Indicates this...
▼	Expanded folder view (to right of folder name).
▼	You can scroll for more variables and/or folders (in bottom left corner of screen).
✓	If selected with F4 .
🔒	Locked
🗳️	Archived

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.

Note: Type a letter repeatedly to cycle through the names that start with that letter.

Variable Types as Listed on VAR-LINK

Type	Description
ASM	Assembly-language program
DATA	Data
EXPR	Expression (includes numeric values)
FUNC	Function
GDB	Graph database
LIST	List
MAT	Matrix
PIC	Picture of a graph
PRGM	Program
STR	String
TEXT	Text Editor session

Types not listed above are miscellaneous data types used by software applications.

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.

Press:**To:****ESC**

Return to the current application without pasting the highlighted name.

Displaying Information about Variables on the Home Screen

From the Home screen, you can display information about variables without opening the VAR-LINK screen.

- To determine if a variable with a given name exists in the system table, Enter the **IsVar()** function on the Home screen.

IsVar (*var_name*)

└─ **IsVar** is a function, which requires you to enclose the variable name in parentheses.

- To determine if a variable is archived, use the **IsArchiv()** function.

IsArchiv (*var_name*)

- To determine if a variable is locked, use the **IsLocked()** function.

IsLocked (*var_name*)

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

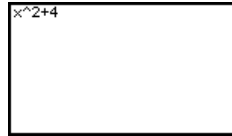
You can show all variable types except **ASM**, **DATA**, **GDB**, and variables created by **Flash Apps**. For example, you must open a **DATA** variable in the Data/Matrix Editor.

1. On **VAR-LINK**, move the cursor to highlight the variable.

2. Press:

[F6]

If you highlight a folder, the screen shows the number of variables in that folder.



3. To return to **VAR-LINK**, press any key.

Note: You cannot edit the contents from this screen.

Selecting Items from the List

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, then press [F4] .
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.

To select:**Do this:**

All folders and all variables

Press \leftarrow to expand the folder, then press $\boxed{F5}$ **All** and select **1:Select All**.



Choosing **3:Select Current** selects the last set of items transmitted to your unit during the current **VAR-LINK** session.

Choosing **4:Expand All** or **5:Collapse All** expands or collapses your folders or Flash applications.

Note: Press either \leftarrow or \rightarrow to toggle between expanded or collapsed view when you have a folder highlighted.

Folders and Variables

Folders give you a convenient way to manage variables by organizing them into related groups.

The Voyage™ 200 has one built-in folder named **MAIN**. Unless you create other folders and designate a user-created folder as the current folder, all variables are stored in the **MAIN** folder by default. A system variable or a variable with a reserved name 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

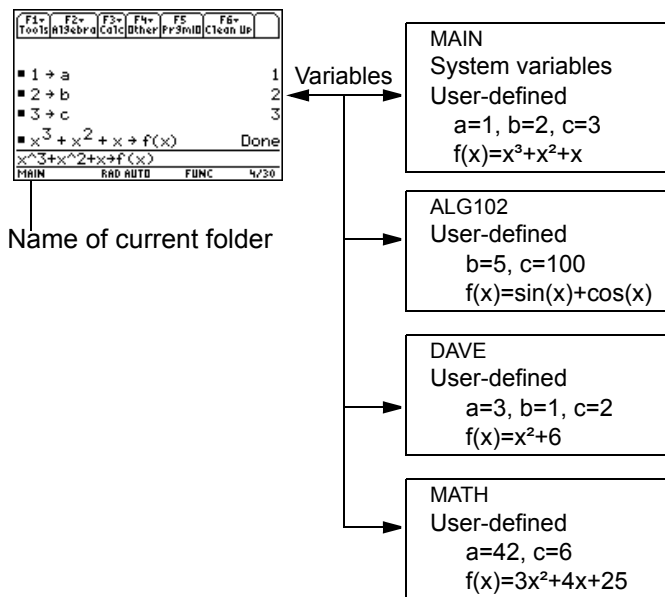
(**TblStart**, **ΔTbl**, etc.)

Y= Editor functions

(**y1(x)**, etc.)

By creating additional folders, you can store independent sets of user-defined variables (including user-defined functions). For example, you can create separate folders for different Voyage™ 200 applications (Math, Text Editor, etc.) or classes. You can store a user-defined variable in any existing folder.

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.



You cannot create a folder within another folder.

The system variables in the **MAIN** folder are always directly accessible, regardless of the current folder.

Note: User-defined variables are stored in the “current folder” unless you specify otherwise.

Creating a Folder from the VAR-LINK Screen

1. Press **[2nd]** [VAR-LINK].
2. Press **[F1]** **Manage** and select **5:Create Folder**.



3. Type a unique folder name up to eight characters, and press **[ENTER]** twice.

After you create a new folder from **VAR-LINK**, that folder is not automatically set as the current folder.

Creating a Folder from the Home Screen

Enter the **NewFold** command on the Home screen.

NewFold *folderName*

└─ Folder name to create. This new folder is set automatically as the current folder.

Setting the Current Folder from the Home Screen

Enter the **setFold** function on the Home screen.

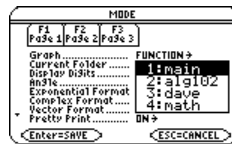
setFold (*folderName*)

└ **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

1. Press **MODE**.
2. Highlight the **Current Folder** setting.
3. Press **⏏** to display a menu of existing folders.



Note: To cancel the menu or exit the dialog box without saving any changes, press **ESC**.

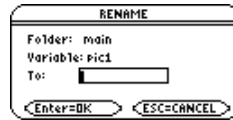
4. Select the applicable folder. Either:
 - Highlight the folder name and press **ENTER**.
 - or –
 - Press the corresponding number or letter for that folder.
5. Press **ENTER** to save your changes and close the dialog box.

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.



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:

folderName \ *variableName*

– or –

folderName \ *functionName*

For example:

If Current Folder = MAIN	Folders and Variables															
<table border="1"><tbody><tr><td>■ 1 → a</td><td>1</td></tr><tr><td>■ $x^3 + x^2 + x + f(x)$</td><td>Done</td></tr><tr><td>■ 42 → math\ a</td><td>42</td></tr><tr><td>■ $3 \cdot x^2 + 4 \cdot x + 25 + \text{math}\ f(x)$</td><td>Done</td></tr><tr><td colspan="2">$3x^2+4x+25+\text{math}\ f(x)$</td></tr><tr><td>MAIN</td><td>RAD AUTO FUNC 4/30</td></tr></tbody></table>	■ 1 → a	1	■ $x^3 + x^2 + x + f(x)$	Done	■ 42 → math\ a	42	■ $3 \cdot x^2 + 4 \cdot x + 25 + \text{math}\ f(x)$	Done	$3x^2+4x+25+\text{math}\ f(x)$		MAIN	RAD AUTO FUNC 4/30	<table border="1"><tbody><tr><td>MAIN</td></tr><tr><td>a=1</td></tr><tr><td>$f(x)=x^3+x^2+x$</td></tr></tbody></table>	MAIN	a=1	$f(x)=x^3+x^2+x$
■ 1 → a	1															
■ $x^3 + x^2 + x + f(x)$	Done															
■ 42 → math\ a	42															
■ $3 \cdot x^2 + 4 \cdot x + 25 + \text{math}\ f(x)$	Done															
$3x^2+4x+25+\text{math}\ f(x)$																
MAIN	RAD AUTO FUNC 4/30															
MAIN																
a=1																
$f(x)=x^3+x^2+x$																
<table border="1"><tbody><tr><td>■ 4 · a</td><td>4</td></tr><tr><td>■ 4 · math\ a</td><td>168</td></tr><tr><td>■ f(5)</td><td>155</td></tr><tr><td>■ math\ f(5)</td><td>120</td></tr><tr><td colspan="2">$\text{math}\ f(5)$</td></tr><tr><td>MAIN</td><td>RAD AUTO FUNC 4/30</td></tr></tbody></table>	■ 4 · a	4	■ 4 · math\ a	168	■ f(5)	155	■ math\ f(5)	120	$\text{math}\ f(5)$		MAIN	RAD AUTO FUNC 4/30	<table border="1"><tbody><tr><td>MATH</td></tr><tr><td>a=42</td></tr><tr><td>$f(x)=3x^2+4x+25$</td></tr></tbody></table>	MATH	a=42	$f(x)=3x^2+4x+25$
■ 4 · a	4															
■ 4 · math\ a	168															
■ f(5)	155															
■ math\ f(5)	120															
$\text{math}\ f(5)$																
MAIN	RAD AUTO FUNC 4/30															
MATH																
a=42																
$f(x)=3x^2+4x+25$																

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 open application's entry line. If you paste a variable name that is not in the current folder, the pathname (*folderName\variableName*) is pasted.

Listing Only a Specified Folder and/or Variable Type, or Flash application

If you have a lot of variables, folders, or Flash applications, 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. (To cancel a menu, press **[ESC]**.)

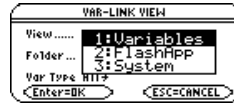
View — Allows you to choose variables, Flash applications, or system variables to view.

Note: To list system variables (window variables, etc.), select **3:System**.

Folder — Always lists **1:All** and **2:main**, but lists other folders only if you have created them.

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, variable type, or Flash application.

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.



Note: 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.

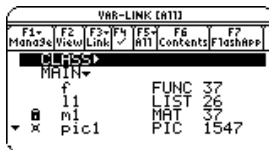
Locking or Unlocking Variables Folders, or Flash Applications

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. When a Flash application is locked, you cannot delete it.

1. On **VAR-LINK**, select the variables, folders, or Flash application.

- Press **[F1] Manage** and select **6:Lock** or **7:UnLock**.

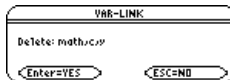
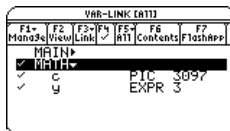
-  indicates a locked variable or folder in RAM.
-  indicates an archived variable, which is locked automatically.



Deleting a Folder from the VAR-LINK Screen

When you delete a folder from the **VAR-LINK** screen, all of the variables in that folder are also deleted. You cannot delete the **MAIN** folder.

- Press **[2nd] [VAR-LINK]**.
- Press **[F4]** to select the folder(s) to delete. (The folder's variables become selected automatically.)
- Press **[F1] 1:Delete** or **[←]**.
- Press **[ENTER]** to confirm the deletion of the folder and all its variables.



Deleting a Variable or a Folder from the Home Screen

Before deleting a folder from the Home screen, you must first delete all the variables stored in that folder.

- To delete a variable, enter the **DelVar** command on the calculator Home screen.

DelVar *var1* [, *var2*] [, *var3*] ...

- To delete all variables of a specific type, enter the **DelType** command on the calculator Home screen.

DelType *var_type* where *var_type* is the variable type.

Note: The **DelType** command deletes all variables of the specified type in all folders.

- To delete an empty folder, enter the **DelFold** command on the calculator Home screen.

DelFold *folder1* [, *folder2*] [, *folder3*] ...

Note: You cannot delete the **MAIN** 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, Y= Editor, Table Editor, or Data/Matrix Editor — The cursor must be on the entry line.
- Text Editor, Window Editor, Numeric Solver, or Program Editor — The cursor can be anywhere on the screen.

You can also paste a variable name to the current cursor location in many Flash applications.

Procedure

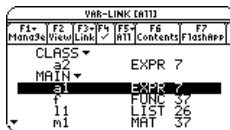
Starting from an application listed above:

1. Position the cursor where you want to insert the variable name.

sin(|

2. Press $\boxed{2nd}$ [VAR-LINK].

3. Highlight the applicable variable.



Note: You can also highlight and paste folder names.

4. Press \boxed{ENTER} to paste the variable name.

sin(a1|

Note: This pastes the variable's name, not its contents. Use $\boxed{2nd}$ [RCL], instead of $\boxed{2nd}$ [VAR-LINK], to recall a variable's contents.

5. Finish typing the expression.

sin(a1)

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.

Archiving and Unarchiving a Variable

To archive or unarchive one or more variables interactively, use the **VAR-LINK** screen. You can also perform these operations from the Home screen or a program.

Why Would You Want to Archive a Variable?

The user data archive lets you:

- Store data, programs, or any other variables to a safe location where they cannot be edited or deleted inadvertently.
- Create additional free RAM by archiving variables. For example:
 - You can archive variables that you need to access but do not need to edit or change, or variables that you are not using currently but need to retain for future use.
Note: You cannot archive variables with reserved names or system variables.
 - If you acquire additional programs for your Voyage™ 200, particularly if they are large, you may need to create additional free RAM before you can install those programs.

Additional free RAM can improve performance times for certain types of calculations.

From the VAR-LINK Screen

To archive or unarchive:

1. Press **[2nd]** **[VAR-LINK]** to display the **VAR-LINK** screen.
2. Select one or more variables, which can be in different folders. (You can select an entire folder by selecting the folder name.)

Note: To select a single variable, highlight it. To select multiple variables, highlight each variable and press **[F4]** ✓.

3. Press **[F1]** and select either:

8:Archive Variable

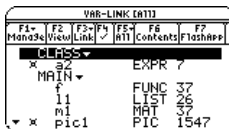
– Or –

9:Unarchive Variable



If you select **8:Archive Variable**, the variables are moved to the user data archive.

⌘ = archived variables



You can access an archived variable just as you would any locked variable. For all purposes, an archived variable is still in its original folder; it is simply stored in the user data archive instead of RAM.

Note: An archived variable is locked automatically. You can access the variable, but you cannot edit or delete it.

From the Home Screen or a Program

Use the **Archive** and **Unarchiv** commands:

Archive *variable1, variable2, ...*

Unarchiv *variable1, variable2, ...*

If a Garbage Collection Message Is Displayed

If you use the user data archive extensively, you may see a Garbage Collection message. This occurs if you try to archive a variable when there is not enough free archive memory. However, the Voyage™ 200 will attempt to rearrange the archived variables to make additional room.

Responding to the Garbage Collection Message

When you see the message to the right:

- To continue archiving, press **ENTER**.
- or –
- To cancel, press **ESC**.



Note: If batteries are low, replace them before performing garbage collection, or archive memory may be lost.

After garbage collection, depending on how much additional space is freed, the variable may or may not be archived. If not, you can unarchive some variables and try again.

Why not Perform Garbage Collection Automatically, without a Message?

The message:

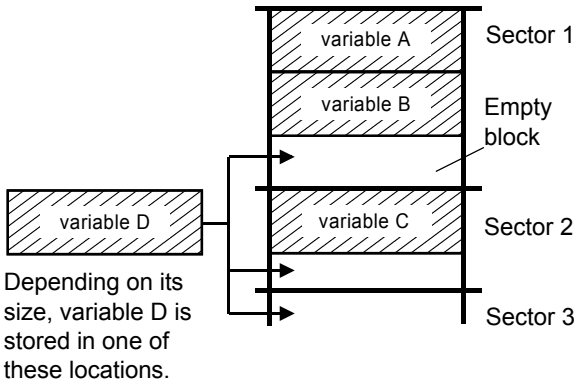
- Lets you know why an archive will take longer than usual. It also alerts you that the archive may fail if there is not enough memory.
- Can alert you when a program is caught in a loop that repetitively fills the user data archive. Cancel the archive and investigate the reason.

Why Is Garbage Collection Necessary?

The user data archive is divided into sectors. When you first begin archiving, variables are stored consecutively in sector 1. This continues to the end of the sector. If there is not enough space left in the sector, the next variable is stored at the beginning of the next sector. Typically, this leaves an empty block at the end of the previous sector.

Each variable that you archive is stored in the first empty block large enough to hold it.

Note: An archived variable is stored in a continuous block within a single sector; it cannot cross a sector boundary.

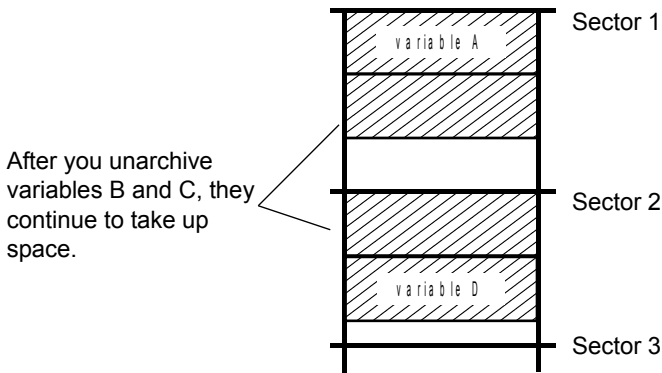


This process continues to the end of the last sector. Depending on the size of individual variables, the empty blocks may account for a significant amount of space.

Note: Garbage collection occurs when the variable you are archiving is larger than any empty block.

How Unarchiving a Variable Affects the Process

When you unarchive a variable, it is copied to RAM but it is not actually deleted from user data archive memory.



Unarchived variables are “marked for deletion,” meaning they will be deleted during the next garbage collection.

If the MEMORY Screen Shows Enough Free Space

Even if the **MEMORY** screen shows enough free space to archive a variable, you may still get a Garbage Collection message.

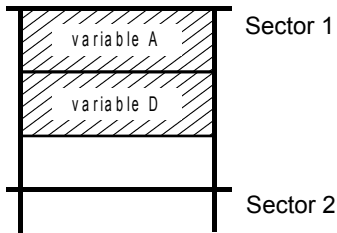
The Voyage™ 200 has 2.7 MB of user-available Flash ROM memory. All 2.7 MB can be used for Flash Apps, but only approximately 1 MB of it can be used for user data archive.

The Garbage Collection Process

The garbage collection process:

- Deletes unarchived variables from the user data archive.

- Rearranges the remaining variables into consecutive blocks.



Memory Error When Accessing an Archived Variable

An archived variable is treated the same as a locked variable. You can access the variable, but you cannot edit or delete it. In some cases, however, you may get a **Memory Error** when you try to access an archived variable.

What Causes the Memory Error?

The **Memory Error** message is displayed if there is not enough free RAM to access the archived variable. This may cause you to ask, “If the variable is in the user data archive, why does it matter how much RAM is available?” The answer is that the following operations can be performed only if a variable is in RAM.

- Opening a text variable in the Text Editor.
- Opening a data variable, list, or matrix in the Data/Matrix Editor.
- Opening a program or function in the Program Editor.

- Running a program or referring to a function.

Note: A temporary copy lets you open or execute an archived variable. However, you cannot save any changes to the variable.

So that you do not have to unarchive variables unnecessarily, the Voyage™ 200 performs a “behind-the-scenes” copy. For example, if you run a program that is in the user data archive, the Voyage™ 200:

1. Copies the program to RAM.
2. Runs the program.
3. Deletes the copy from RAM when the program is finished.

The error message is displayed if there is not enough free RAM for the temporary copy.

Note: Except for programs and functions, referring to an archived variable does not copy it. If variable **ab** is archived, it is not copied if you perform **6*ab**.

Correcting the Error

To free up enough RAM to access the variable:

1. Use the **VAR-LINK** screen (2nd [VAR-LINK]) to determine the size of the archived variable that you want to access.
2. Use the **MEMORY** screen (2nd [MEM]) to check the RAM free size.
3. Free up the needed amount of memory by:
 - Deleting unnecessary variables from RAM.

- Archiving large variables or programs (moving them from RAM to the user data archive).

Note: Typically, the RAM free size must be larger than the archived variable.

Connectivity

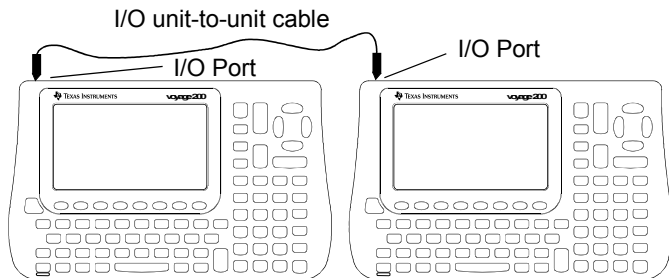
Connecting Two Units

The Voyage™ 200 Graphing Calculator comes with a cable that lets you connect two units. Once connected, you can transmit information between two units. An I/O unit-to-unit cable is included with the Voyage™ 200; use the calculator's I/O port with this cable.

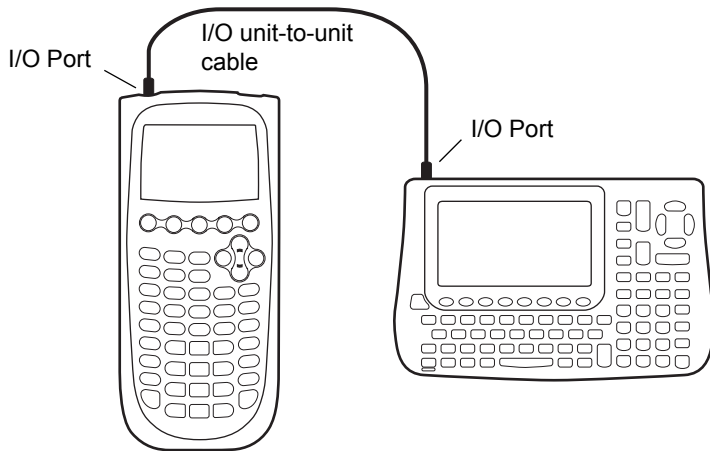
Connecting before Sending or Receiving

Using firm pressure, insert one end of the cable into the link port of each unit. Either unit can send or receive, depending on how you set them up from the **VAR-LINK** screen.

You can link a TI-89 Titanium or Voyage™ 200 to another TI-89 Titanium, Voyage™ 200, TI-89, or TI-92 Plus.



Two Voyage™ 200 calculators linked together



A TI-89 Titanium and a Voyage™ 200 linked together

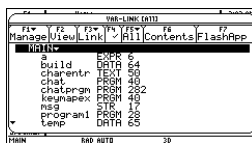
Transmitting Variables, Flash Applications, and Folders

Transmitting variables is a convenient way to share any variable listed on the **VAR-LINK** screen — functions, programs, etc. You can also transmit Flash applications (Apps) and folders.

Setting Up the Units

Flash applications will transfer only between certain units. You can transfer an App from a Voyage™ 200 to another Voyage™ 200, or from a Voyage™ 200 to a TI-92 Plus.

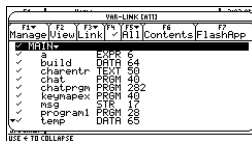
1. Connect two graphing calculators using the appropriate cable.
2. On the *sending* unit, press **[2nd]** **[VAR-LINK]** to display the **VAR-LINK** screen.
3. On the *sending* unit, select the variables, folders, or Flash applications you want to send.



- To select a single variable, Flash application, or folder, move the cursor to highlight it and press **[F4]** to place a checkmark (✓) beside it.



- If on the default **VAR-LINK** screen, this selects the folder and its contents. Collapsed folders become expanded when selected.



- If selecting a Flash App (from the F7 tab), this selects the App folder and its contents. A checkmark appears beside the folder, but not beside the contents. Collapsed Flash App folders do not automatically become expanded.



- To select multiple variables, Flash applications, or folders, highlight each one and press **[F4]** to place a checkmark (✓) beside it. Use **[F4]** again to deselect any that you do not want to transmit.
- To select all variables, Flash applications, or folders use **[F5]** **All 1:Select All**.



4. On the *receiving* unit, press **[2nd]** **[VAR-LINK]** to display the **VAR-LINK** screen. (The sending unit remains on the **VAR-LINK** screen.)
5. On both the *receiving* and the sending unit, press **[F3]** **Link** to display the menu options.



6. On the *receiving* unit, select **2:Receive**.

The message **VAR-LINK: WAITING TO RECEIVE** and the **BUSY** indicator are displayed in the status line of the receiving unit.

7. On the *sending* unit, select **1:Send**

This starts the transmission.

During transmission, a progress bar is displayed in the status line of the receiving unit. When transmission is complete, the **VAR-LINK** screen is updated on the receiving unit.

Note: Before transferring a purchased App, the receiving unit must have the appropriate certificate, if required. A certificate is a file that is generated by TI. Free and concept Apps do not require a certificate.

Rules for Transmitting Variables, Flash Applications, or Folders

Unlocked and unarchived variables that have the same name on both the sending and receiving units will be overwritten from the sending unit.

Locked variables that have the same name on both the sending and receiving units must be unlocked on the receiving unit before they can be overwritten from the sending unit. If

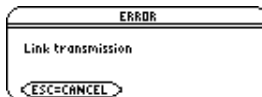
archived variables have the same names on both the sending and receiving units, a message asks you to confirm that you will allow the variables to be overwritten.

If you select:	What happens:
Unlocked variable	The variable is transmitted to the current folder and it remains unlocked on the receiving unit.
Locked variable	The variable is transmitted to the current folder and it remains locked on the receiving unit.
Archived variable	The variable is transmitted to the current folder and it remains archived on the receiving unit.
Unlocked Flash application	If the receiving unit has the correct certification, the Flash application is transmitted. It remains unlocked on the receiving unit.
Locked Flash application	If the receiving unit has the correct certification, the Flash application is transmitted. It remains locked on the receiving unit.
Unlocked Folder	The folder and its selected contents are transmitted. The folder remains unlocked on the receiving unit.
Locked Folder	The folder and its selected contents are transmitted. The folder becomes unlocked on the receiving unit.

Canceling a Transmission

From either the sending or receiving unit:

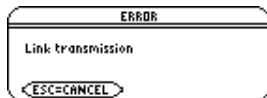
1. Press **[ON]**.
An error message is displayed.
2. Press **[ESC]** or **[ENTER]**.



Common Error and Notification Messages

Shown on:	Message and Description:
-----------	--------------------------

Sending unit



This is displayed after several seconds if:

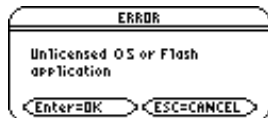
- A cable is not attached to the sending unit's link port.
– or –
- A receiving unit is not attached to the other end of the cable.
– or –
- The receiving unit is not set up to receive.

Press **[ESC]** or **[ENTER]** to cancel the transmission.

Note: The sending unit may not always display this message. Instead, it may remain **BUSY** until you cancel the transmission.

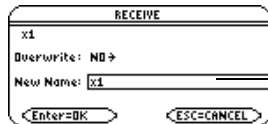
Shown on:**Message and Description:**

Sending unit



The receiving unit does not have the correct certification for the operating system (OS) or Flash application being sent.

Receiving unit



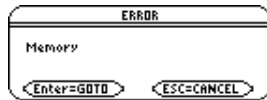
New Name is active only if you change Overwrite to NO.

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 = YES**.)
- To store the variable to a different name, set **Overwrite = 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 = SKIP** and press **[ENTER]**.
- To cancel the transmission, press **[ESC]**.

Shown on:**Message and Description:**

Receiving unit



The receiving unit does not have enough memory for what is being sent. Press **[ESC]** or **[ENTER]** to cancel the transmission.

Deleting Variables, Flash Applications, or Folders

1. Press **[2nd]** **[VAR-LINK]** to display the **VAR-LINK** screen.
2. Select the variables, folders, or Flash applications to delete.
 - To select a single variable, Flash application, or folder, move the cursor to highlight it and press **[F4]** to place a checkmark (✓) beside it.
 - If on the default **VAR-LINK** screen, this selects the folder and its contents. Collapsed folders become expanded when selected.
 - If selecting a Flash App (from the F7 tab), this selects the App folder and its contents. A checkmark appears beside the folder, but not beside the contents. Collapsed Flash App folders do not automatically become expanded.

Note: You cannot delete the **Main** folder.

- To select multiple variables, Flash applications, or folders highlight each one and press **[F4]** to place a checkmark (✓) beside it. Use **[F4]** again to deselect any that you do not want to transmit.
- To select all variables, Flash applications, or folders use **[F5]** **All 1:Select All**.

3. Press **[F1]** and choose **1:Delete**.
– or –
Press **[↵]**. A confirmation message appears.
4. Press **[ENTER]** to confirm the deletion.

Where to Get Flash Applications (Apps)

For up-to-date information about available Flash applications, check the Texas Instruments Web site at education.ti.com.

Many Apps no longer require a certificate. If you try to transfer an App from one unit to another and receive an **Unlicensed OS or Flash application** message, try downloading the App again from the Texas Instruments Web site at education.ti.com.

You can download a Flash application and/or certificate from the Texas Instruments Web site to a computer, and use a USB Silver Edition cable to install the application or certificate on your Voyage™ 200 Graphing Calculator.

For Flash App installation instructions, see education.ti.com/guides.

Transmitting Variables under Program Control

You can use a program containing **GetCalc** and **SendCalc** to transmit a variable from one device to another.

SendCalc sends a variable to the link port, where a linked device can receive the variable. The linked device must be on the Home screen or must execute **GetCalc** from a program.

The “Chat” Program

The following program uses **GetCalc** and **SendCalc**. The program sets up two loops that let the linked devices take turns sending and receiving/displaying a variable named **msg**. **InputStr** lets each user enter a message in the **msg** variable.

```
:Chat ()
:Prgm
:ClrIO
:Disp "On first unit to send","
  enter 1;","On first to receive,"
:InputStr " 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
```

①

②

③

④

⑤

⑥

Notes:

- ❶ Sets up this unit to receive and display the variable msg.
- ❷ Then lets this user enter a message in msg and send it.
- ❸ Loop executed by the unit that receives the first message.
- ❹ Lets this user enter a message in msg and send it.
- ❺ Then sets up this unit to receive and display msg.
- ❻ Loop executed by the unit that sends the first message.

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.

Running the Program

This procedure assumes that:

- The two devices are linked with the connecting cable.
- The Chat program is loaded on both devices.
 - Use each device's Program Editor to enter the program.
– or –
 - Enter the program on one device and then use **VAR-LINK** to transmit the program variable to the other device.

To run the program on both devices:

1. On the Home screen of each device, enter **chat()**.

- When each device displays its initial prompt, respond as shown below.

On the:	Type:
Device that will send the first message.	1 and press ENTER .
Device that will receive the first message.	0 and press ENTER .

- Take turns typing a message and pressing **ENTER** to send the variable **msg** to the other device.

Stopping the Program

Because the **Chat** program sets up an infinite loop on both devices, press **ON** (on both devices) to break the program. If you press **ESC** to acknowledge the error message, the program stops on the Program I/O screen. Press **F5** or **ESC** to return to the Home screen.

Upgrading the Operating System (OS)

You can upgrade the OS on your Voyage™ 200 using your computer. You can also transfer the OS from one unit to another identical model (for example, from a TI-89 Titanium to a TI-89 Titanium or from a Voyage™ 200 to a Voyage™ 200).

Installing OS software resets all device memory to the original factory settings. This means that all user-defined variables (in both RAM and the user data archive), functions, programs, lists, and folders (except the Main folder) will be deleted. It is possible that

Flash applications could also be deleted. You should use TI Connect software to back up your data to your computer before installing a new OS on your calculator.

See the important information concerning batteries before performing an OS upgrade.

Important Operating System Download Information

New batteries should be installed before beginning an OS download.

If you are operating your Voyage™ 200 in a language other than English, you should ensure that you have the most current localizer application installed when you upgrade the OS software. If you do not have the most current localizer installed, prompts, error messages and status information related to new functionality in the OS may not display correctly.

When in OS download mode, the Automatic Power Down™ (APD™) feature does not function. If you leave your device in download mode for an extended time before you actually start the downloading process, your batteries may become depleted. You will then need to replace the depleted batteries with new batteries before downloading.

If you accidentally interrupt the transfer before it is complete, you will need to reinstall the OS. Again, remember to install new batteries before downloading.

Backing Up Your Unit Before an Operating System Installation

When you install an OS upgrade, the installation process:

- Deletes all user-defined variables (in both RAM and the user data archive), functions, programs, and folders.

- Could delete all Flash applications.
- Resets all system variables and modes to their original factory settings. This is equivalent to using the **MEMORY** screen to reset all memory.

To retain any existing variables or Flash applications, do the following before installing the upgrade:

- **Important:** Install new batteries.
- Transmit the variables or Flash applications to another device.
– or –
- Use a USB cable or TI Connectivity Cable USB and TI Connect™ software (education.ti.com/downloadticonnect) to send the variables and/or Flash applications to a computer.

Where to Get Operating System Upgrades

For up-to-date information about available OS upgrades, check the Texas Instruments Web site at education.ti.com/downloadticonnect.

You can download an OS upgrade, Localizer or Flash application from the Texas Instruments Web site to a computer, and use a USB Silver Edition cable to install the OS or application on your Voyage™ 200.

For complete information, refer to the instructions on the web.

Transferring the Operating System

OS software will transfer only from a TI-89 Titanium to a TI-89 Titanium, TI-89 to a TI-89, from a Voyage™ 200 to a Voyage™ 200, or from a TI-92 Plus to a TI-92 Plus.

To transfer the Operating System (OS) from unit to unit:

1. Link two like units together, for example, a TI-89 Titanium to a TI-89 Titanium; or a Voyage™ 200 to a Voyage™ 200.
2. On the receiving and the sending unit, press **[2nd] [VAR-LINK]** to display the **VAR-LINK** screen.
3. On the receiving and the sending unit, press **[F3] Link** to display the menu options.
4. On the receiving unit, select **5:Receive OS**.

A warning message displays. Press **[ESC]** to halt the process, or press **[ENTER]** to proceed. Pressing **[ENTER]**, displays **VAR-LINK: WAITING TO RECEIVE** and **BUSY** in the status line of the receiving unit.

5. On the sending unit, select **4:Send OS**.

A warning message displays. Press **[ESC]** to halt the process, or press **[ENTER]** to start the transmission.

Important:

- For each receiving unit, remember to back up information as necessary and install new batteries.
- Be sure both the sending and receiving units are in the **VAR-LINK** screen.

During the transfer, the receiving unit shows how the transfer is progressing. When the transfer is complete:

- The sending unit returns to the **VAR-LINK** screen.
- The receiving unit returns to either the Apps desktop or the Home screen. You may need to use **[◀] [□]** (lighten) or **[▶] [□]** (darken) to adjust the contrast.

Do Not Attempt to Cancel an Operating System Transfer

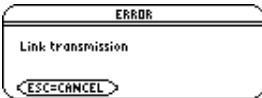
After the transfer starts, the receiving unit's existing OS is effectively deleted. If you interrupt the transfer before it is complete, the receiving unit will not operate properly. You will then need to reinstall the OS upgrade.

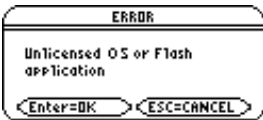
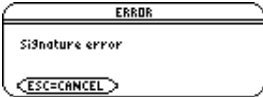

If You are Upgrading the Operating System on Multiple Units

To perform an OS upgrade on multiple units, download and install the OS into one unit and then transfer the OS upgrade from one unit to another. This method is faster than installing it on each unit via a computer. OS upgrades are released free of charge and you do not need to obtain a certificate before you download or install them.

Error Messages

Most error messages are displayed on the sending unit. Depending on when the error occurs during the transfer process, you may see an error message on the receiving unit.

Error Message	Description
	The sending and receiving units are not connected properly, or the receiving unit is not set up to receive.

Error Message	Description
	<p>The certificate on the receiving unit is not valid for the operating system (OS) or App on the sending unit. You must obtain and install a valid certificate.</p> <p>If the App no longer requires a certificate, you can download it again from the Texas Instruments Web site at education.ti.com and then install the App again on your calculator.</p>
	<p>An error occurred during the transfer. The current OS in the receiving unit is corrupted. You must reinstall the product software from a computer.</p>
	<p>Replace the batteries on the unit displaying this message.</p>

Collecting and Transmitting ID Lists

The **VAR-LINK** screen **[F3] 6:Send ID List** menu option allows collection of electronic ID numbers from individual TI-89 Titanium, TI-89, Voyage™ 200, or TI-92 Plus devices.

ID Lists and Group Certificates




The ID list feature provides a convenient way to collect device IDs for group purchase of commercial applications. After the IDs are collected, transmit them to Texas Instruments so a group certificate can be issued.

A group certificate allows distribution of purchased software to multiple TI-89 Titanium, TI-89, Voyage™ 200, or TI-92 Plus units. The software can be loaded, deleted from, and reloaded to the devices as often as needed for as long as the software remains listed in the group certificate. You may add new ID numbers and/or new commercial applications to a group certificate.

Collecting ID Lists

You can use one device to collect all of the IDs, or use several collection units and then consolidate their ID lists onto one device.

To send an ID number from one device to another, first connect two units by using a USB unit-to-unit cable or I/O unit-to-unit cable.

Step:	On the:	Do this:
1.	Collecting unit (Receiving unit)	Display the Home screen. Press:  [CALC HOME]
2.	Sending unit	a. Press  [VAR-LINK] to display the VAR-LINK screen. b. Press  Link and select 6:Send ID List .



Step:	On the:	Do this:
		The sending unit adds a copy of its unique ID number to the collection unit's ID list. The sending unit always retains its own ID number, which cannot be deleted from the device.
3.	Additional units	Repeat steps 1 and 2 until all the IDs are collected onto one device. Depending on available memory in the collection device, it is possible to collect over 4,000 IDs.

Notes:

- You cannot view the ID list on the sending or collecting units.
- Each time an ID list is successfully sent from one device to another, the ID list is automatically deleted from the sending unit.
- If an ID is collected from a device twice, the duplicate ID is automatically deleted from the list.

Clearing the ID List

The ID list remains on the collection device after it is uploaded to the computer. You can then use the collection device to upload the list to other computers.

To clear the ID list from the collection unit:

1. Press **[2nd]** **[VAR-LINK]** to display the **VAR-LINK** screen.
2. Press **[F1]** **Manage** and select **A:Clear ID List**.



Compatibility among the TI-89 Titanium, Voyage™ 200, TI-89, and TI-92 Plus

In general, TI-89 Titanium, TI-89, Voyage™ 200, and TI-92 Plus data and programs are compatible with each other, with a few exceptions.

Most functions of the TI-89 Titanium are compatible with the TI-89, Voyage™ 200, and TI-92 Plus. The TI-89 Titanium and the TI-89 are similar, except that the TI-89 Titanium has more memory (more room for Apps and user archive) and the TI-89 Titanium has a USB port. The Voyage™ 200 is the same as the TI-92 Plus except it has more memory, and thus more room for applications (Apps).

All data is compatible among the TI-89 Titanium, TI-89, Voyage™ 200, and TI-92 Plus, but some programs written for one may not run or may not run the same on the other because of differences in the device's screen sizes and keyboards and the USB port on the TI-89 Titanium.

Other incompatibilities can occur because of different versions of the operating system. To download the latest version of the operating system, visit the Texas Instruments Web site at education.ti.com/downloadticonnect.

Link Transmission Table

To → From ↓	TI-89 Titanium	TI-89	Voyage™ 2 00	TI-92 Plus
TI-89 Titanium	OS Apps Variables	Apps Variables	Variables	Variables
TI-89	Apps Variables	OS Apps Variables	Variables	Variables
Voyage™ 2 00	Variables	Variables	OS Apps Variables	Apps Variables
TI-92 Plus	Variables	Variables	Apps Variables	OS Apps Variables

Activities

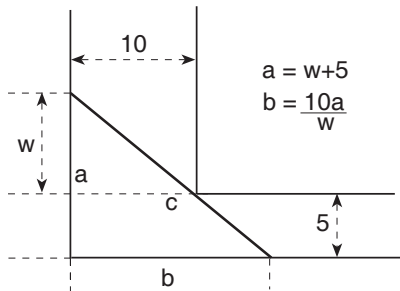
Analyzing the Pole-Corner Problem

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

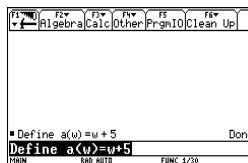
The maximum length of a pole c is the shortest line segment touching the interior corner and opposite sides of the two hallways as shown in the diagram below.

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.

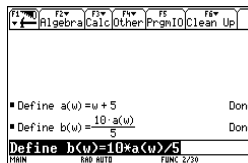


1. Define the expression for side **a** in terms of **w** and store it in **a(w)**.

Note: When you want to define a function, use multiple character names as you build the definition.

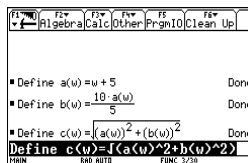


2. Define the expression for side **b** in terms of **w** and store it in **b(w)**.



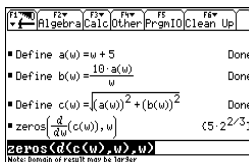
3. Define the expression for side **c** in terms of **w** and store it in **c(w)**.

Enter: **Define c(w) = $\sqrt{(a(w))^2 + b(w)^2}$**



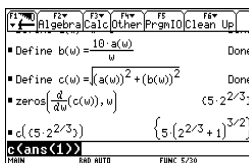
4. Use the **zeros()** function to compute the zeros of the first derivative of **c(w)** to find the minimum value of **c(w)**.

Note: The maximum length of the pole is the minimum value of **c(w)**.



5. Compute the exact maximum length of the pole.

Enter: **c** (**2nd**) [**ANS**])

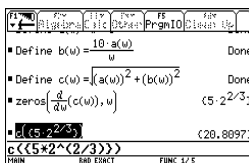


6. Compute the approximate maximum length of the pole.

Result: Approximately 20.8097 feet.

Note: Use the auto-paste feature to copy the result from step 4 to the entry line inside the parentheses of **c()** and press

♦ **ENTER**.



Deriving the Quadratic Formula

This activity shows you how to derive the quadratic formula:

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

Detailed information about using the functions in this example can be found in *Symbolic Manipulation*.

Performing Computations to Derive the Quadratic Formula

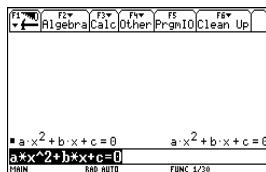
Perform the following steps to derive the quadratic formula by completing the square of the generalized quadratic equation.

1. Clear all one-character variables in the current folder.

F6

Choose **1:Clear a-z** and press **ENTER** to confirm.

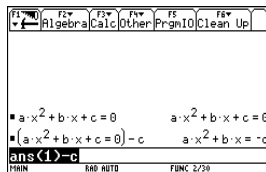
2. On the Home screen, enter the generalized quadratic equation:
 $ax^2+bx+c=0$.



3. Subtract c from both sides of the equation.

2nd **ANS** **-** **C**

Note: This example uses the result of the last answer to perform computations on the Voyage™ 200. This feature reduces keystroking and chances for error.



4. Divide both sides of the equation by the leading coefficient a .

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean	Up
$a \cdot x^2 + b \cdot x + c = 0 \qquad a \cdot x^2 + b \cdot x + c = 0$ $\left(a \cdot x^2 + b \cdot x + c = 0 \right) \cdot \frac{1}{a} \qquad a \cdot x^2 + b \cdot x + c = 0$ $\frac{a \cdot x^2 + b \cdot x + c}{a} = \frac{0}{a} \qquad \frac{x \cdot (a \cdot x + b)}{a} = \frac{-c}{a}$					
ans(1)/a					
MAIN		END		FUNC 3/30	

5. Use the **expand ()** function to expand the result of the last answer.

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean	Up
$a \cdot x^2 + b \cdot x + c = 0 \qquad a \cdot x^2 + b \cdot x + c = 0$ $\left(a \cdot x^2 + b \cdot x + c = 0 \right) \cdot \frac{1}{a} \qquad a \cdot x^2 + b \cdot x + c = 0$ $\frac{a \cdot x^2 + b \cdot x + c}{a} = \frac{-c}{a} \qquad \frac{x \cdot (a \cdot x + b)}{a} = \frac{-c}{a}$					
$\text{expand} \left(\frac{x \cdot (a \cdot x + b)}{a} = \frac{-c}{a} \right) \qquad x^2 + \frac{b \cdot x}{a} = \frac{-c}{a}$					
expand(ans(1))					
Note: Domain of Result Expr: bc, bR34r					

6. Complete the square by adding $\left(\frac{b/a}{2}\right)^2$ to both sides of the equation.

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean	Up
$\text{expand} \left(\frac{x \cdot (a \cdot x + b)}{a} = \frac{-c}{a} \right) \qquad x^2 + \frac{b \cdot x}{a} = \frac{-c}{a}$					
$\left(x^2 + \frac{b \cdot x}{a} - \frac{-c}{a} \right) + \left(\frac{\frac{b}{a}}{2} \right)^2$ $x^2 + \frac{b \cdot x}{a} + \frac{b^2}{4 \cdot a^2} = \frac{b^2}{4 \cdot a^2} - \frac{c}{a}$					
ans(1)+(b/a)/2^2					
MAIN		END		FUNC 5/30	

7. Factor the result using the **factor ()** function.

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean	Up
$x^2 + \frac{b \cdot x}{a} + \frac{b^2}{4 \cdot a^2} = \frac{b^2}{4 \cdot a^2} - \frac{c}{a}$					
$\text{factor} \left(x^2 + \frac{b \cdot x}{a} + \frac{b^2}{4 \cdot a^2} = \frac{b^2}{4 \cdot a^2} - \frac{c}{a} \right)$ $\frac{(2 \cdot a \cdot x + b)^2}{4 \cdot a^2} = \frac{-(4 \cdot a \cdot c - b^2)}{4 \cdot a^2}$					
factor(ans(1))					
MAIN		END		FUNC 6/30	

8. Multiply both sides of the equation by $4a^2$.

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean	Up
$\frac{(2 \cdot a \cdot x + b)^2}{4 \cdot a^2} = \frac{-(4 \cdot a \cdot c - b^2)}{4 \cdot a^2}$					
$4 \cdot a^2 \left(\frac{(2 \cdot a \cdot x + b)^2}{4 \cdot a^2} = \frac{-(4 \cdot a \cdot c - b^2)}{4 \cdot a^2} \right)$ $(2 \cdot a \cdot x + b)^2 = -(4 \cdot a \cdot c - b^2)$					
4a^2*ans(1)					
Note: Domain of Result Expr: bc, bR34r					

9. Take the square root of both sides of the equation with the constraint that $a > 0$ and $b > 0$ and $x > 0$.

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean Up	
$4 \cdot a^2 \left(\frac{(2 \cdot a \cdot x + b)^2}{4 \cdot a^2} = \frac{(4 \cdot a \cdot c - b^2)}{4 \cdot a^2} \right)$ $(2 \cdot a \cdot x + b)^2 = (4 \cdot a \cdot c - b^2)$ $\sqrt{(2 \cdot a \cdot x + b)^2} = \sqrt{(4 \cdot a \cdot c - b^2)} \mid a > 0 \text{ and } b > 0$ $2 \cdot a \cdot x + b = \sqrt{b^2 - 4 \cdot a \cdot c}$					
Ans(1) a > 0 and b > 0 and x > 0					
MIN	END	QUIT	FUNC	B/20	

10. Solve for x by subtracting b from both sides and then dividing by $2a$.

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean Up	
$(2 \cdot a \cdot x + b)^2 = (4 \cdot a \cdot c - b^2)$ $\sqrt{(2 \cdot a \cdot x + b)^2} = \sqrt{(4 \cdot a \cdot c - b^2)} \mid a > 0 \text{ and } b > 0$ $2 \cdot a \cdot x + b = \sqrt{b^2 - 4 \cdot a \cdot c}$ $(2 \cdot a \cdot x + b = \sqrt{b^2 - 4 \cdot a \cdot c}) - b$ $2 \cdot a \cdot x = \sqrt{b^2 - 4 \cdot a \cdot c} - b$					
Ans(1) - b					
MIN	END	QUIT	FUNC	B/20	

Note: This is only one of the two general quadratic solutions due to the constraint in step 9.

F1	F2	F3	F4	F5	F6
Algebra	Calc	Other	PrnIO	Clean Up	
$(2 \cdot a \cdot x + b = \sqrt{b^2 - 4 \cdot a \cdot c}) - b$ $2 \cdot a \cdot x = \sqrt{b^2 - 4 \cdot a \cdot c} - b$ $2 \cdot a \cdot x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}$ $x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}$					
Ans(1) / (2a)					
<small>Note: Domain of result may be 1st-3rd</small>					

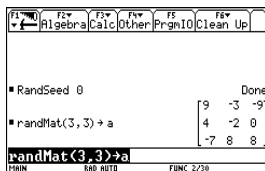
Exploring a Matrix

This activity shows you how to perform several matrix operations.

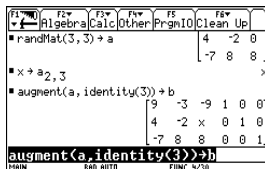
Exploring a 3x3 Matrix

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.

1. 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**.



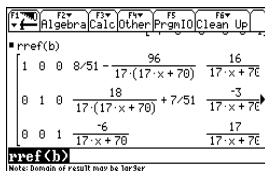
2. Replace the **[2,3]** element of the matrix with the variable **x**, and then use the **augment()** function, to augment the 3x3 identity to **a** and store the result in **b**.



3. 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.

Note: Use the cursor in the history area to scroll the result.



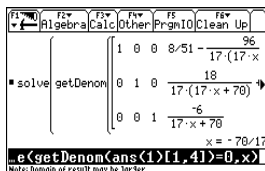
4. Solve for the value of x that will cause the inverse of the matrix to be invalid.

Enter:

solve(getDenom([2nd] [ANS] [1,4])=0,x)

Result: $x = -70/17$

Note: Use the cursor in the history area to scroll the result.



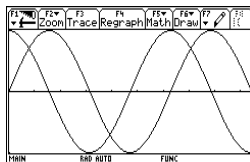
Exploring $\cos(x) = \sin(x)$

This activity uses two methods to find where $\cos(x) = \sin(x)$ for the values of x between 0 and 3π .

Method 1: Graph Plot

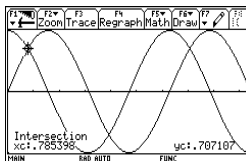
Perform the following steps to observe where the graphs of the functions $y_1(x)=\cos(x)$ and $y_2(x)=\sin(x)$ intersect.

1. In the **Y= Editor**, set $y_1(x)=\cos(x)$ and $y_2(x)=\sin(x)$.
2. In the **Window Editor**, set $x_{\min}=0$ and $x_{\max}=3\pi$.
3. Press **[F2]** and select **A:ZoomFit**.



4. Find the intersection point of the two functions.

Note: 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.



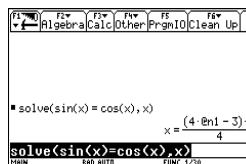
5. Note the x and y coordinates. (Repeat steps 4 and 5 to find the other intersections.)

Method 2: Symbolic Manipulation

Perform the following steps to solve the equation $\sin(x)=\cos(x)$ with respect to x .

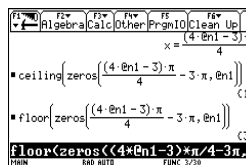
1. On the Home screen, enter **solve(sin(x)=cos(x),x)**.

The solution for x is where **@n1** is any integer.



2. Using the **ceiling()** and **floor()** functions, find the ceiling and floor values for the intersection points as shown.

Note: Move the cursor into the history area to highlight the last answer. Press **ENTER** to copy the result of the general solution.



3. Enter the general solution for x and apply the constraint for $@n1$ as shown.

Compare the result with Method 1.

Note: To get the *with* operator, press:

$\boxed{2nd}\boxed{[K]}$.

The calculator screen shows the following steps and results:

$$x = \frac{(4 \cdot @n1 - 3) \cdot \pi}{4} \quad | \quad @n1 = (1 \ 2 \ 3)$$

$$x = \left\{ \frac{\pi}{4} \quad \frac{5 \cdot \pi}{4} \quad \frac{9 \cdot \pi}{4} \right\}$$

$$x = \frac{(4 \cdot @n1 - 3) \cdot \pi}{4} \quad | \quad @n1 = (1 \ 2 \ 3)$$

$$x = \{ .785398 \quad 3.92699 \quad 7.06858 \}$$

The final line shows the result with the constraint: $x = (4 * @n1 - 3) * \pi / 4 | @n1 = (1, 2, 3)$

Finding Minimum Surface Area of a Parallelepiped

This activity 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 *Symbolic Manipulation* and *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.

- On the Home screen, define the function **sa(x,y,v)** for the surface area of a parallelepiped.

Enter: **define sa(x,y,v)=2*x*y + 2v/x+2v/y**

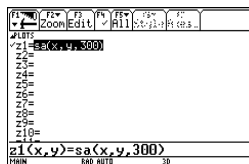
The calculator screen shows the function definition process:

Define sa(x,y,v)=2*x*y + $\frac{2 \cdot v}{x}$ + $\frac{2 \cdot v}{y}$

Done

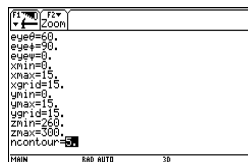
Define sa(x,y,v)=2*x*y+2v/x+2v/y

2. Select the 3D Graph mode. Then enter the function for $z_1(x,y)$ as shown in this example with volume $v=300$.

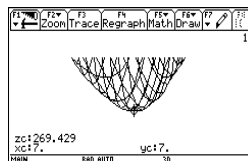


3. Set the Window variables to:

eye= [60,90,0]
x= [0,15,15]
y= [0,15,15]
z= [260,300]
ncontour= [5]



4. Graph the function and use **Trace** to go to the point close to the minimum value of the surface area function.

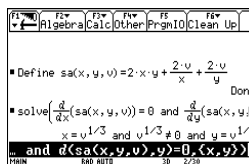


Finding the Minimum Surface Area Analytically

Perform the following steps to solve the problem analytically on the Home screen.

1. Solve for x and y in terms of v .

Enter: **solve(d(sa(x,y,v),x)=0 and d(sa(x,y,v),y)=0,{x,y})**



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 $f(x)=x^3-2x^2+x-7$

C : $1 \rightarrow a:3.22 \rightarrow b$

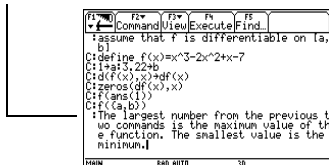
C : $d(f(x),x) \rightarrow df(x)$

C : $\text{zeros}(df(x),x)$

C : $f(\text{ans}(1))$

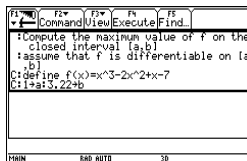
C : $f(\{a,b\})$

: The largest number from the previous two commands is the maximum value of the function. The smallest number is the minimum value.



```
CommandViewExecuteFind
:assume that f is differentiable on [a,
b]
C:define f(x)=x^3-2x^2+x-7
C:1→a:3.22→b
C:d(f(x),x)→df(x)
C:zeros(df(x),x)
C:ans(1)
C:f({a,b})
: The largest number from the previous t
wo commands is the maximum value of th
e function. The smallest value is the
minimum.
```

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**.



```
CommandViewExecuteFind
:Compute the maximum value of f on the
closed interval [a,b]
:assume that f is differentiable on [a
b]
C:define f(x)=x^3-2x^2+x-7
C:1→a:3.22→b
```

4. Press $\boxed{F4}$ repeatedly to execute each line in the script one at a time.

Note: Press $\boxed{F4}$ and select **2:Clear split** to go back to a full-sized **Text Editor** screen.

```

F4 0) F4+ F4- F4 F4 F4
CommandViewExecuteFind
Compute the maximum value of f on the
closed interval [a,b]
Assume that f is differentiable on [a
b]
C:define f(x)=x^3-2x^2+x-7
C:R:=a3,22b
Done
Define f(x)=x^3-2x^2+x-7 Done

```

```

F4 0) F4+ F4- F4 F4 F4
CommandViewExecuteFind
C:1:=a3,22b
C:d(f(x),x)=df(x)
C:zeros(df(x),x)
C:f(aans())
C:f(a,b)
The largest number from the previous
zeros(df(x),x)
f()
f((a b)) (-7 8.86945)

```

5. To see the results of the script on a full-sized screen, go to the Home screen.

Note: Press $\boxed{2nd}$ \boxed{QUIT} twice to display the Home screen.

```

F4 0) F4+ F4- F4 F4 F4
AlgebraCalcOtherPrngIOClean Up
Define f(x)=x^3-2x^2+x-7 Done
1+a : 3.22+b 3.22
d(f(x),x)+df(x) Done
zeros(df(x),x)
f()
f((a b)) (-7 8.86945)
((a,b))

```

Decomposing a Rational Function

This activity 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 *Basic Function Graphing* and *Symbolic Manipulation*.

Decomposing a Rational Function

To examine the decomposition of the rational function $f(x) = (x^3 - 10x^2 - x + 50)/(x - 2)$ on a graph:

- On the Home screen, enter the rational function as shown below and store it in a function $f(x)$.

Enter: $(x^3 - 10x^2 - x + 50)/(x - 2) \rightarrow f(x)$

Note: Actual entries are displayed in reverse type in the example screens.

TI-84 Plus calculator screen showing the rational function $(x^3 - 10x^2 - x + 50)/(x - 2)$ being stored to $f(x)$. The screen displays the function in reverse type: $(x^3 - 10x^2 - x + 50)/(x - 2) \rightarrow f(x)$.

- Use the proper fraction function (**propFrac**) to split the function into a quotient and remainder.

TI-84 Plus calculator screen showing the **propFrac** function being used to decompose the rational function. The screen displays the function in reverse type: $(x^3 - 10x^2 - x + 50)/(x - 2)$ and the result: $\frac{16}{x - 2} + x^2 - 8x - 17$.

- Copy the last answer to the entry line.
–or–

Enter: $16/(x - 2) + x^2 - 8x - 17$

Note: Move the cursor into the history area to highlight the last answer. Press **ENTER** to copy it to the entry line.

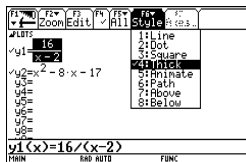
TI-84 Plus calculator screen showing the last answer from the previous screen being copied to the entry line. The screen displays the function in reverse type: $(x^3 - 10x^2 - x + 50)/(x - 2)$ and the result: $\frac{16}{x - 2} + x^2 - 8x - 17$.

- Edit the last answer in the entry line. Store the remainder to $y1(x)$ and the quotient to $y2(x)$ as shown.

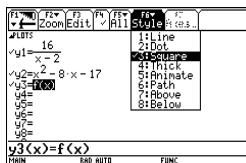
Enter: $16/(x - 2) \rightarrow y1(x)$; $x^2 - 8x - 17 \rightarrow y2(x)$

TI-84 Plus calculator screen showing the last answer from the previous screen being edited to store the remainder to $y1(x)$ and the quotient to $y2(x)$. The screen displays the function in reverse type: $(x^3 - 10x^2 - x + 50)/(x - 2)$ and the result: $\frac{16}{x - 2} + x^2 - 8x - 17$.

5. In the **Y= Editor**, select the thick graphing style for $y_2(x)$.



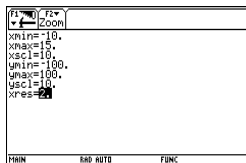
6. Add the original function $f(x)$ to $y_3(x)$ and select the square graphing style.



7. In the **Window Editor**, set the window variables to:

$$x = [-10, 15, 10]$$

$$y = [-100, 100, 10]$$

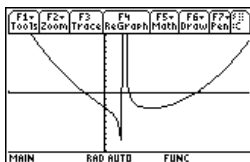
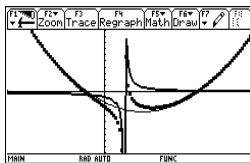


8. Draw the graph.

Note: Be sure the Graph mode is set to Function.

Observe that the global behavior of the $f(x)$ function is basically represented by the quadratic quotient $y_2(x)$. The rational expression is basically a quadratic function as x gets very large in both the positive and negative directions.

The lower graph is $y_3(x)=f(x)$ graphed separately using the line style.



Studying Statistics: Filtering Data by Categories

This activity provides a statistical study of the weights of high school students using categories to filter the data.

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

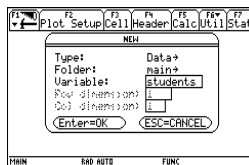
Category (C2)	Academic Year and Sex
1	Freshman boys
2	Freshman girls
3	Sophomore boys
4	Sophomore girls
5	Junior boys
6	Junior girls
7	Senior boys
8	Senior girls

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.

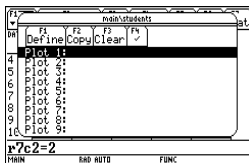
1. Start the **Data/Matrix Editor**, and create a new Data variable named **students**.



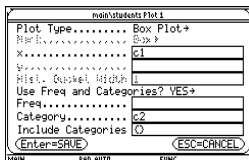
2. Enter the data and categories from Table 2 into columns **c1** and **c2**, respectively.

	c1	c2	c3	c4	c5
4	120	1			
5	140	1			
6	85	2			
7	80	2			
8	90	2			
9	80	2			
10	95	2			

3. Open the **F2 Plot Setup** toolbar menu.
Note: Set up several box plots to compare different subsets of the entire data set.



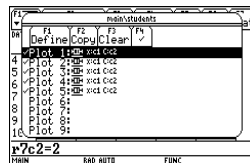
4. Define the plot and filter parameters for **Plot 1** as shown in this screen.



5. Copy Plot 1 to Plot 2.



6. Repeat step 5 and copy Plot 1 to Plot 3, Plot 4, and Plot 5.



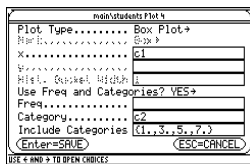
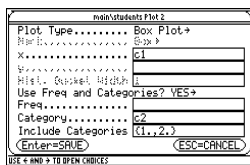
7. Press $\boxed{F1}$, and modify the **Include Categories** item for Plot 2 through Plot 5 to the following:

Plot 2: {1,2}
(freshman boys, girls)

Plot 3: {7,8}
(senior boys, girls)

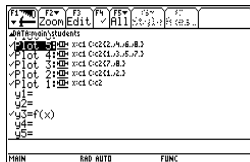
Plot 4: {1,3,5,7}
(all boys)

Plot 5: {2,4,6,8}
(all girls)

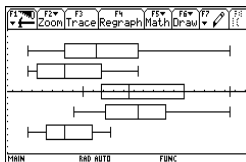


8. In the **Y= Editor**, deselect any functions that may be selected from a previous activity.

Note: Only Plot 1 through Plot 5 should be selected.

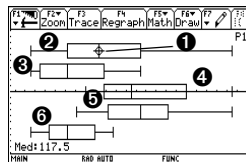


9. Display the plots by pressing **[F2]** and selecting **9:Zoomdata**.



10. Use the **Trace** tool to compare the median student weights for different subsets.

- ① median, all students
- ② all students
- ③ all freshmen
- ④ all seniors
- ⑤ all boys
- ⑥ all girls



CBL 2™ Program for the Voyage™ 200

This activity provides a program that can be used when the Voyage™ 200 is connected to a Calculator-Based Laboratory™ (CBL 2™) unit. This program works with the “Newton’s Law of Cooling” experiment. You can use your computer keyboard to type lengthy text and then use TI Connect™ software to send it to the calculator. More CBL 2™ programs are available from the TI Web site at educaton.ti.com.

Program Instruction	Description
:cooltemp()	Program name
:Prgm	
:Local i	Declare local variable; exists only at run time.

Program Instruction	Description
:setMode("Graph","FUNCTION")	Set up the Voyage™ 200 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 Voyage™ 200 Program IO (input/output) screen.
:-10→xmin:99→xmax:10→xscl	Set up the Window variables.
:-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™ unit.
:Send{1,2,1}	Set up Chan. 2 of the CBL 2™ to AutoID to record temp.
:Disp "Press ENTER to start"	Prompt the user to press <input type="text" value="ENTER"/> .
:Disp "graphingTemperature."	
: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.

Program Instruction	Description
:Send{3,1,-1,0}	Send the Trigger command to the CBL 2™; 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™ and store it in a list.
:PtOn i,data[i]	Plot the temperature data on a graph.
:EndFor	
:seq(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.

You can also use the Calculator-Based Ranger™ system (CBR™) to explore the mathematical and scientific relationships between distance, velocity, acceleration, and time using data collected from activities you perform.

Studying the Flight of a Hit Baseball

This activity 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.



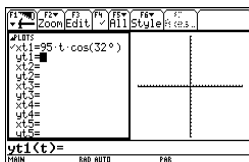
2. Set the modes for **Page 2** as shown in this screen.



3. In the **Y= Editor** on the left side, enter the equation for the distance of the ball at time t for $xt1(t)$.

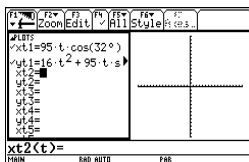
$$xt1(t)=95*t*\cos(32^\circ)$$

Note: Press $\boxed{2nd} \boxed{[^\circ]}$ to obtain the degree symbol.



4. In the **Y= Editor**, enter the equation for the height of the ball at time t for $yt1(t)$.

$$yt1(t)=-16*t^2+95*t*\sin(32^\circ)$$

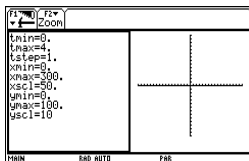


5. Set the Window variables to:

t values= [0,4,.1]

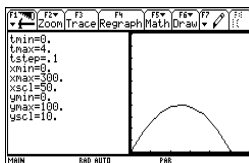
x values= [0,300,50]

y values= [0,100,10]



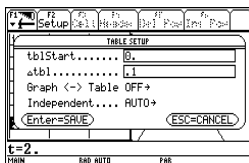
6. Switch to the right side and display the graph.

Note: Press $\boxed{2nd} \boxed{+}$.



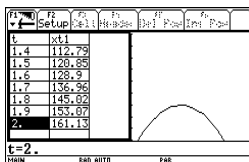
7. Display the **TABLE SETUP** dialog box, and change **tblStart** to **0** and **Δt** to **0.1**.

Note: Press \blacklozenge [TBLSET].



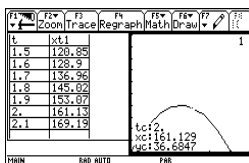
8. Display the table in the left side and press \odot to highlight **t=2**.

Note: Press \blacklozenge [TABLE].



9. Switch to the right side. Press $\boxed{F3}$, and trace the graph to show the values of **x** and **y** when **tc=2**.

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**.



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.

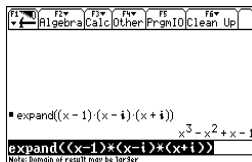
Visualizing Complex Zeros of a Cubic Polynomial

This activity describes graphing the complex zeros of a cubic polynomial.

Visualizing Complex Roots

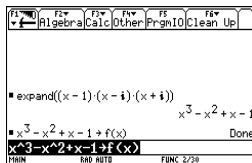
Perform the following steps to expand the cubic polynomial $(x-1)(x-i)(x+i)$, find the absolute value of the function, graph the modulus surface, and use the **Trace** tool to explore the modulus surface.

1. On the Home screen, use the **expand()** function to expand the cubic expression $(x-1)(x-i)(x+i)$ and see the first polynomial.



2. Copy and paste the last answer to the entry line and store it in the function **f(x)**.

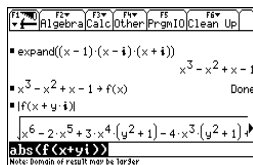
Note: Move the cursor into the history area to highlight the last answer and press **[ENTER]**, to copy it to the entry line.



3. Use the **abs()** function to find the absolute value of **f(x+y)**.

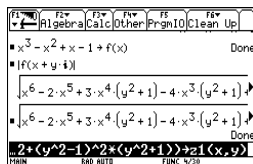
(This calculation may take about 2 minutes.)

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.



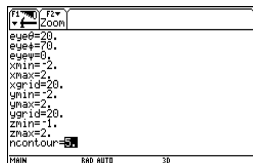
4. Copy and paste the last answer to the entry line and store it in the function **z1(x,y)**.

Note: The graph of **z1(x,y)** will be the modulus surface.



5. Set the unit to 3D graph mode, turn on the axes for graph format, and set the Window variables to:

eye= [20,70,0]
x= [-2,2,20]
y= [-2,2,20]
z= [-1,2]
ncontour= [5]



6. In the **Y=Editor**, press:

F

and set the Graph Format variables to:

Axes= ON

Labels= ON

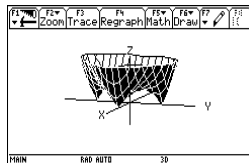
Style= HIDDEN SURFACE

Note: Calculating and drawing the graph takes about three minutes.

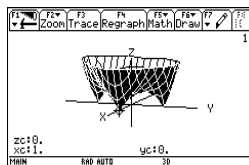


7. Graph the modulus surface.

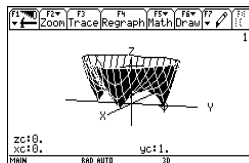
The 3D graph is used to visually display a picture of the roots where the surface touches the **xy** plane.



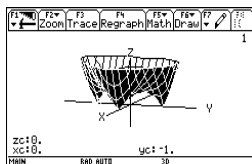
8. Use the Trace tool to explore the function values at **x=1** and **y=0**.



9. Use the Trace tool to explore the function values at **x=0** and **y=1**.



10. 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.

Solving a Standard Annuity Problem

This activity 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 .

TI-84 Plus calculator screen showing the equation $\text{solve}(s=p \cdot (1+i)^n, p) \mid p=(1+i)^n \cdot s$ and $\text{solve}(s=p \cdot (1+i)^n, p)$ entered.

2. Enter the equation to solve for n .

TI-84 Plus calculator screen showing the equation $\text{solve}(s=p \cdot (1+i)^n, p) \mid p=(1+i)^n \cdot s$ and $\text{solve}(s=p \cdot (1+i)^n, n)$ entered, along with the formula $n = \frac{\ln(\frac{s}{p})}{\ln(1+i)}$ and $\frac{s}{p} > 0$.

3. Enter the equation to solve for i using the *with* operator.

$\text{solve}(s=p \cdot (1+i)^n, i) \mid s=2000 \text{ and } p=1000$
and $n=6$

Result: The interest rate is 12.246%.

TI-84 Plus calculator screen showing the equation $\text{solve}(s=p \cdot (1+i)^n, i) \mid s=2000 \text{ and } p=1000$ and $n=6$ entered, along with the formula $n = \frac{\ln(\frac{s}{p})}{\ln(1+i)}$ and $\frac{s}{p} > 0$.

Note:

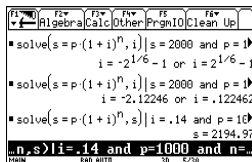
- To enter the “*with*” (|) operator: $\boxed{2\text{nd}} \boxed{[K]}$
- Press $\boxed{\blacklozenge} \boxed{\text{ENTER}}$ to obtain a floating-point result.

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 .

$\text{solve}(s=p*(1+i)^n,s) \mid i=.14 \text{ and } p=1000 \text{ and } n=6$



Result: The future value at 14% interest is 2,194.97.

Computing the Time-Value-of-Money

This activity 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 the electronic chapter *Programming*, which is available from the TI Web site at education.ti.com and on the CD in this package.

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 = beginning 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
```

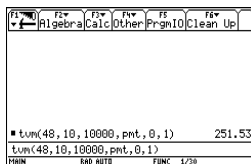
Note: You can use your computer keyboard to type lengthy text and then use TI Connect™ software to send it to the Voyage™ 200.

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.



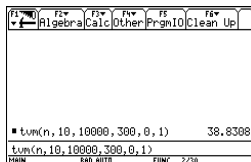
$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$
Algebra	Calc	Other	Frq	IO	Clean	Up	
* $tvm(48, 10, 10000, pmt, 0, 1)$							251.53
$tvm(48, 10, 10000, pmt, 0, 1)$							
2ND	MODE	2/20	FUNC 1/20				

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.



$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$	$\frac{P/Y}{Y}$
Algebra	Calc	Other	Frq	IO	Clean	Up	
* $tvm(n, 10, 10000, 300, 0, 1)$							38.8308
$tvm(n, 10, 10000, 300, 0, 1)$							
2ND	MODE	2/20	FUNC 1/20				

Finding Rational, Real, and Complex Factors

This activity shows how to find rational, real, or complex factors of expressions. Detailed information about the steps used in this example can be found in *Symbolic Manipulation*.

Finding Factors

Enter the expressions shown below on the Home screen.

1. **factor(x³-5x)** [ENTER] displays a rational result.

The calculator screen shows the input **factor(x³-5x)** and the result $x(x^2 - 5)$. The top of the screen displays function keys: F1, F2, F3, F4, F5, F6. Below these are menu options: Algebra, Calc, Other, Frq, IO, Clean Up. At the bottom, it shows DIM, 800 8010, and FUNC 1/20.

2. **factor(x³+5x)** [ENTER] displays a rational result.

The calculator screen shows the input **factor(x³+5x)** and the result $x(x^2 + 5)$. The top of the screen displays function keys: F1, F2, F3, F4, F5, F6. Below these are menu options: Algebra, Calc, Other, Frq, IO, Clean Up. At the bottom, it shows DIM, 800 8010, and FUNC 1/20.

3. **factor(x³-5x,x)** [ENTER] displays a real result.

The calculator screen shows the input **factor(x³-5x,x)** and the result $x(x + \sqrt{5})(x - \sqrt{5})$. The top of the screen displays function keys: F1, F2, F3, F4, F5, F6. Below these are menu options: Algebra, Calc, Other, Frq, IO, Clean Up. At the bottom, it shows DIM, 800 8010, and FUNC 1/20.

4. **cfactor(x³+5x,x)** [ENTER] displays a complex result.

The calculator screen shows the input **cfactor(x³+5x,x)** and the result $x(x + \sqrt{5}i)(x - \sqrt{5}i)$. The top of the screen displays function keys: F1, F2, F3, F4, F5, F6. Below these are menu options: Algebra, Calc, Other, Frq, IO, Clean Up. At the bottom, it shows DIM, 800 8010, and FUNC 1/20.

Simulation of Sampling without Replacement

This activity simulates drawing different colored balls from an urn without replacing them. Detailed information about the steps used in this example can be found in the electronic chapter *Programming*.

Sampling-without- Replacement Function

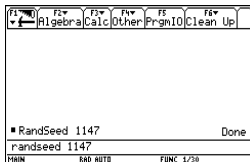
In the **Program Editor**, define **drawball()** as a function that can be called with two parameters. The first parameter is a list where each element is the number of balls of a certain color. The second parameter is the number of balls to select. This function returns a list where each element is the number of balls of each color that were selected.

```
:drawball(urnlist,drawnum)           :For j,1,colordim,1
:Func                                 :cumSum(templist)→urncum
:Local templist,drawlist,colordim,  :If pick ≤ urncum[j] Then
    numballs,i,pick,urncum,j        :drawlist[j]+1→drawlist[j]
:If drawnum>sum(urnlist)             :templist[j]-1→templist[j]
:Return "too few balls"              :Exit
:dim(urnlist)→colordim               :EndIf
:urnlist→templist                    :EndFor
:newlist(colordim)→drawlist          :EndFor
:For i,1,drawnum,1                   :Return drawlist
:sum(templist)→numballs              :EndFunc
:rand(numballs)→pick
(continued in next column)
```

Sampling without Replacement

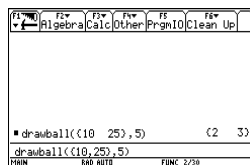
Suppose an urn contains n_1 balls of a color, n_2 balls of a second color, n_3 balls of a third color, etc. Simulate drawing balls without replacing them.

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)**.

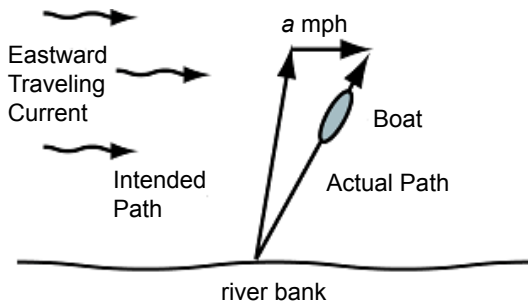
Result: 2 red balls and 3 white balls.



Using Vectors to Determine Velocity

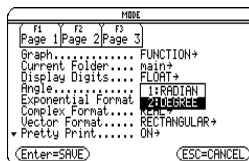
A small fishing boat leaves from the south bank of the Allegheny River and heads at an 80° angle with an engine speed of 20 knots. However, the eastward force of the current carries the boat along so it actually travels at a 60° angle with the shore.

How fast is the current, and how fast does the boat actually travel?



1. Set the modes for **Page 1** as shown in this screen. (Show angles in degrees and display all digits with a floating decimal point.)

Press: **[MODE]** \odot \odot \odot . On the Angle option, select **2:DEGREE**. On the Display Digits option, select **E:FLOAT**.



2. Set the modes for **Page 2** as shown in this screen. (Display answers in decimal form.)

Press: **[MODE]** **[F2]** \odot \odot . On the Exact/Approx option, select **3:APPROXIMATE**.



3. Enter vectors describing the initial path of the boat, water current, and resultant path of the boat.

Store these vectors as \mathbf{i} , \mathbf{c} , and \mathbf{r} . Use the value \mathbf{a} for the unknown speed of the current. Use the value \mathbf{b} for the speed of the boat.

Enter:

$[20,80^\circ] \rightarrow \mathbf{i}$

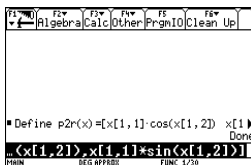
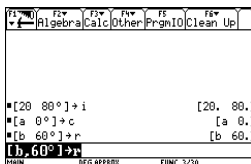
$[a,0^\circ] \rightarrow \mathbf{c}$

$[b,60^\circ] \rightarrow \mathbf{r}$

Vectors are commonly written in either polar or rectangular form, so it is useful to convert polar vectors into rectangular form.

4. Define function $\mathbf{p2r}$.

Enter: **Define** $\mathbf{p2r}(x)=[x[1,1]*\cos(x[1,2]), x[1,1]*\sin(x[1,2])]$



When converted to rectangular form, the sum of vectors i and c equals the resultant vector r .

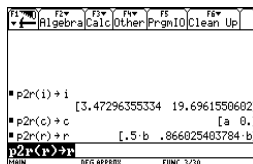
5. Using function **p2r**, convert vectors i , c , and r to rectangular form.

Enter:

p2r(i)→ i

p2r(c)→ c

p2r(r)→ r



Because the vectors are equal, the x-coordinate of $i+c$ must equal the x-coordinate of the resultant vector r . Likewise, the y-coordinate of $i+c$ must equal the y-coordinate of resultant vector r .

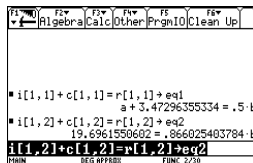
6. Set up two equations involving vectors $i+c$ and r .

- Equation 1 sets the x-coordinates equal to each other.
- Equation 2 sets the y-coordinates equal.

Store these equations into **eq1** and **eq2**, respectively. Enter:

i[1,1]+c[1,1]=r[1,1]→**eq1**

i[1,2]+c[1,2]=r[1,2]→**eq2**

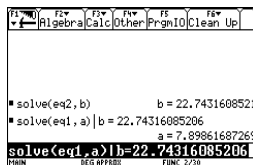


7. Solve **eq2** for **b** to calculate the actual speed of the boat.

solve(eq2,b)

8. Substitute the known value of **b** into **eq1**, and solve **eq1** for **a** to determine **a**, the speed of the eastward traveling current.

solve(eq1,a) | b



The boat travels at a speed of 22.7 knots, and the water current is approximately 7.9 knots.

Appendix A: Functions and Instructions

Categorical Listing of Operations..... 781

Alphabetical Listing of Operations..... 785

This section describes the syntax and action of each Voyage™ 200 function and instruction that is included in the operating system (OS). See modules relating to calculator software applications (Apps) for functions and instructions specific to those Apps.

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.

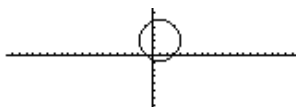
Note: Regraphing erases all drawn items.

Arguments are shown in *italics*.
Arguments in [] brackets are optional. Do not type the brackets.

Syntax line shows the order and the type of arguments that you supply. Be sure to separate multiple arguments with a comma (,).

In a ZoomSqr viewing window:

ZoomSqr:Circle 1.2.3 [ENTER]



Explanation of the function or instruction.

Categorical Listing of Operations

This section lists the Voyage™ 200 functions and instructions in functional groups along with the page numbers where they are described.

Algebra	("with")	912	cFactor()	790	comDenom()	793	
	cSolve()	798	cZeros	802	expand()	816	
	factor()	818	getDenom()	824	getNum()	825	
	nSolve()	848	propFrac()	855	randPoly()	861	
	solve()	878	tCollect()	888	tExpand()	888	
	zeros()	895					
Calculus	f() (integrate)	907	$\Pi()$ (product)	908	$\Sigma()$ (sum)	908	
	arcLen()	787	avgRC()	788	d()	804	
	deSolve()	807	fMax()	820	fMin()	820	
	ImpDif()	829	limit()	833	nDeriv()	844	
	nInt()	846	' (prime)	910	seq()	868	
	taylor()	887					
Graphics	AndPic	786	BldData	789	Circle	791	
	ClrDraw	791	ClrGraph	792	CyclePic	802	
	DrawFunc	811	DrawInv	811	DrawParm	812	
	DrawPol	812	DrawSlp	812	DrwCtour	813	
	FnOff	820	FnOn	821	Graph	827	
	Line	833	LineHorz	834	LineTan	835	
	LineVert	835	NewPic	845	PtChg	855	
	PtOff	855	PtOn	856	ptTest()	856	
	PtText	856	PxlChg	856	PxlCrcl	856	
	PxlHorz	856	PxlLine	857	PxlOff	857	
	PxlOn	857	pxlTest()	857	PxlText	857	
	PxlVert	858	RclGDB	861	RclPic	861	
	RplcPic	866	Shade	873	StoGDB	882	
	StoPic	883	Style	883	Trace	890	
	XorPic	894	ZoomBox	896	ZoomData	897	
	ZoomDec	897	ZoomFit	898	ZoomIn	898	
	ZoomInt	898	ZoomOut	899	ZoomPrev	899	
	ZoomRcl	899	ZoomSqr	899	ZoomStd	900	
	ZoomSto	900	ZoomTrig	900			
	Lists	+ (add)	900	- (subtract)	901	* (multiply)	901
		/ (divide)	902	- (negate)	904	^ (power)	903
augment()		788	crossP()	797	cumSum()	801	
dim()		809	dotP()	811	explist()	816	
left()		832	Δ list()	836	listmat()	836	
matlist()		841	max()	841	mid()	842	
min()		843	newList()	845	polyEval()	853	
product()		854	right()	864	rotate()	864	
shift()		874	SortA	880	SortD	880	
sum()		884					

Math

+ (add)	900	- (subtract)	901	* (multiply)	901
/ (divide)	902	- (negate)	904	% (percent)	905
! (factorial)	907	√() (sqr. root)	908	^ (power)	903
° (gradian)	909	° (degree)	909	∠ (angle)	910
° , ' , "	910	_ (underscore)	910	▶ (convert)	911
10^()	911	Ob , Oh	913	▶Bin	788
▶Cylind	802	▶DD	805	▶Dec	805
▶DMS	811	▶Grad	785	▶Hex	828
▶Polar	853	▶Rect	862	▶Sphere	881
abs()	785	and	785	angle	786
approx()	787	ceiling()	789	conj()	793
cos	794	cos⁻¹()	795	cosh()	796
cosh⁻¹()	796	cot()	796	cot⁻¹()	797
coth()	797	coth⁻¹()	797	csc()	797
csc⁻¹()	798	csch()	798	cosh⁻¹()	798
E	813	e^()	813	exact()	815
▶ln	836	▶logbase	838	floor()	819
fPart()	822	gcd()	822	imag()	829
impDif()	829	int()	830	intDiv()	830
iPart()	831	isPrime()	831	lcm()	832
ln()	836	log()	838	max()	841
min()	843	mod()	843	nCr()	844
nPr()	847	P▶Rx()	850	P▶Ry()	850
r (radian)	909	R▶Pθ()	860	R▶Pr()	860
real()	861	remain()	863	rotate()	864
round()	865	sec()	866	root()	864
sec⁻¹()	867	sech()	867	sech⁻¹()	867
shift()	874	sign()	875	sin()	876
sin⁻¹()	876	sinh()	877	sinh⁻¹()	877
tan()	886	tan⁻¹()	886	tanh()	887
tanh⁻¹()	887	tmpCnv()	889	ΔtmpCnv()	889
x⁻¹	911				

Matrices

+ (add)	900	- (subtract)	901	* (multiply)	901
/ (divide)	902	- (negate)	904	.+ (dot add)	903
.- (dot subt.)	903	. (dot mult.)	904	./ (dot divide)	904
.^ (dot power)	904	^ (power)	903	augment()	788
colDim()	792	colNorm()	792	crossP()	797
cumSum()	801	data▶mat	804	det()	808
diag()	809	dim()	809	dotP()	811
eigVc()	814	eigVl()	814	Fill	819
identity()	828	list▶mat()	836	LU	840
mat▶data	840	mat▶list()	841	max()	841
mean()	841	median()	841	min()	843
mRow()	843	mRowAdd()	843	newMat()	845
norm()	847	product()	854	QR	858
randMat()	861	ref()	862	rowAdd()	865
rowDim()	865	rowNorm()	865	rowSwap()	866
rref()	866	simult()	875	stdDev()	881
stdDevPop()	882	subMat()	884	sum()	884
T	885	unitV()	891	variance()	892
x⁻¹	911				

Programming	=	905	≠	905	<	906
	≤	906	>	906	≥	906
	# (indirection)	909	> (store)	912	Ⓢ (comment)	913
	and	785	ans()	787	Archive	787
	checkTmr()	790	ClockOff	791	ClockOn	791
	ClrErr	791	ClrGraph	792	ClrHome	792
	ClrIO	792	ClrTable	792	CopyVar	794
	CustmOff	801	CustmOn	801	Custom	801
	Cycle	802	dayOfWk()	805	Define	806
	DelFold	806	DelType	807	DelVar	807
	Dialog	809	Disp	810	DispG	810
	DispHome	810	DispTbl	810	DropDown	812
	Else	814	Elseif	814	EndCustm	814
	EndDlog	814	EndFor	814	EndFunc	814
	EndIf	814	EndLoop	815	EndPrgm	815
	EndTBar	815	EndTry	815	EndWhile	815
	entry()	815	Exec	815	Exit	816
	For	821	format()	821	Func	822
	Get	823	GetCalc	823	getConfig()	823
	getDate()	823	getDtFmt()	824	getDtStr()	824
	getFold()	824	getKey()	824	getMode()	825
	getTime()	825	getTmFmt()	825	getTmStr()	825
	getTmZn()	826	getType()	826	getUnits()	826
	Goto	827	If	828	Input	829
	InputStr	830	isClkOn()	831	Item	832
	Lbl	832	isArchiv()	831	isLocked ()	831
	isVar()	831	left()	832	Local	837
	Lock	837	Loop	839	MoveVar	843
	NewFold	845	NewProb	846	not	847
	or	849	Output	849	part()	850
	PassErr	852	Pause	852	PopUp	854
	Prgm	854	Prompt	855	Rename	863
	Request	863	Return	863	right()	864
	Send	867	SendCalc	867	SendChat	868
	setDate()	868	setDtFmt()	868	setFold()	869
	setGraph()	869	setMode()	870	setTable()	871
	setTime()	871	setTmFmt()	871	setTmZn()	872
	startTmr()	881	setUnits()	872	Stop	883
	Style	883	switch()	884	Table	885
	Text	888	Then	888	timeCnv()	888
	Title	889	Toolbar	890	Try	890
	Unarchiv	891	Unlock	891	when()	892
	While	893	xor	893		

Statistics

! (factorial)	907	BldData	789	CubicReg	800
cumSum()	801	ExpReg	818	LinReg	835
LnReg	837	Logistic	839	mean()	841
median()	841	MedMed	842	nCr()	844
NewData	844	NewPlot	846	nPr()	847
OneVar	848	PlotsOff	853	PlotsOn	853
PowerReg	854	QuadReg	859	QuartReg	859
rand()	860	randNorm()	861	RandSeed	861
ShowStat	875	SinReg	878	SortA	880
SortD	880	stdDev()	881	stdDevPop()	882
TwoVar	891	variance()	892		

Strings

& (append)	907	# (indirection)	909	char()	790
dim()	809	expr()	817	format()	821
inString()	830	left()	832	mid()	842
ord()	849	right()	864	rotate()	864
shift()	874	string()	883		

Alphabetical Listing of Operations

Operations whose names are not alphabetic (such as +, !, and >) are listed at the end of this appendix, starting on page 900. 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.

$\text{abs}(\{\pi/2, -\pi/3\})$ [ENTER]

$\{\frac{\pi}{2}, \frac{\pi}{3}\}$

$\text{abs}(2-3i)$ [ENTER]

$\sqrt{13}$

$\text{abs}(z)$ [ENTER]

$|z|$

$\text{abs}(x+yi)$ [ENTER]

$\sqrt{x^2+y^2}$

and MATH/Test and MATH/Base menus

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.

$x \geq 3$ and $x \geq 4$ [ENTER]

$x \geq 4$

$\{x \geq 3, x \leq 0\}$ and $\{x \geq 4, x \leq -2\}$ [ENTER]

$\{x \geq 4, x \leq -2\}$

integer1 and integer2 ⇒ *integer*

Compares two real integers bit-by-bit using an **and** operation. Internally, both integers are converted to signed, 32-bit binary numbers. When corresponding bits are compared, the result is 1 if both bits are 1; otherwise, the result is 0. The returned value represents the bit results, and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

In Hex base mode:

0h7AC36 and 0h3D5F [ENTER]

0h2C16

↳ **Important:** Zero, not the letter O.

In Bin base mode:

0b100101 and 0b100 [ENTER]

0b100

In Dec base mode:

37 and 0b100 [ENTER]

4

Note: A binary entry can have up to 32 digits (not counting the 0b prefix). A hexadecimal entry can have up to 8 digits.

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:

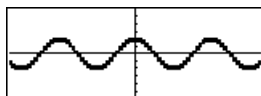
$$y1(x) = \cos(x) \ominus$$

[F6] Style = 3:Square

[F2] Zoom = 7:ZoomTrig

[F1] = 2:Save Copy As...

Type = Picture. Variable = PIC1

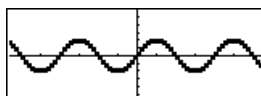


$$y2(x) = \sin(x)$$

[F6] Style = 3:Square

y1 = no checkmark (F4 to deselect)

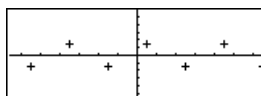
[F2] Zoom = 7:ZoomTrig



[CALC HOME]

AndPic PIC1 [ENTER]

Done



angle() MATH/Complex menu

angle(*expression*) \Rightarrow *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:

$$\text{angle}(0+2i) \text{ [ENTER]}$$

90

In Gradian angle mode:

$$\text{angle}(0+3i) \text{ [ENTER]}$$

100

In Radian angle mode:

$$\text{angle}(1+i) \text{ [ENTER]}$$

$\frac{\pi}{4}$

$$\text{angle}(z) \text{ [ENTER]}$$

$$\text{angle}(x+iy) \text{ [ENTER]}$$

$$\begin{aligned} \blacksquare \text{ angle}(z) &= \frac{-\pi \cdot (\text{sign}(z) - 1)}{2} \\ \blacksquare \text{ angle}(x + i \cdot y) &= \frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right) \end{aligned}$$

angle(*list*) \Rightarrow *list*

angle(*matrix*) \Rightarrow *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:

$$\text{angle}(\{1+2i, 3+0i, 0-4i\}) \text{ [ENTER]}$$

$$\blacksquare \text{ angle}(\{1+2i, 3+0i, 0-4i\}) \text{ [ENTER]}$$

$$\left\{ \frac{\pi}{2} - \tan^{-1}(1/2), 0, \frac{-\pi}{2} \right\}$$

ans() 2nd [ANS] key

ans() \Rightarrow *value*

ans(integer) \Rightarrow *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) \Rightarrow *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(list) \Rightarrow *list*

approx(matrix) \Rightarrow *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([$\sqrt{2}$), $\sqrt{3}$]) ENTER [1.414... 1.732...]

Archive CATALOG

Archive *var1*, *var2* [, *var3*] ...

Moves the specified variables from RAM to the user data archive memory.

You can access an archived variable the same as you would a variable in RAM. However, you cannot delete, rename, or store to an archived variable because it is locked automatically.

To unarchive variables, use **Unarchiv**.

10 \rightarrow arctest ENTER 10

Archive arctest ENTER Done

5*arctest ENTER 50

15 \rightarrow arctest ENTER



ESC

Unarchiv arctest ENTER Done

15 \rightarrow arctest ENTER 15

arclen() MATH/Calculus menu

arclen(expression1, var, start, end) \Rightarrow *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) \Rightarrow *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*

augment(*matrix1*; *matrix2*) ⇒ *matrix*

Returns a new matrix that is *matrix2* appended to *matrix1*. When the "," character is used, the matrices must have equal row dimensions, and *matrix2* is appended to *matrix1* as new columns. When the ";" character is used, the matrices must have equal column dimensions, and *matrix2* is appended to *matrix1* as new rows. Does not alter *matrix1* or *matrix2*.

[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** $\begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{bmatrix}$
 [5.6]►M2 **ENTER** $\begin{bmatrix} 5 & 6 \end{bmatrix}$
 augment(M1;M2) **ENTER** $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

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**).

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** $2 \cdot (x - .4995)$
 avgRC(x^2-x+2,x,.1) **ENTER** $2 \cdot (x - .45)$
 avgRC(x^2-x+2,x.3) **ENTER** $2 \cdot (x+1)$

►Bin MATH/Base menu

integer1 ►Bin ⇒ *integer*

Converts *integer1* to a binary number. Binary or hexadecimal numbers always have a 0b or 0h prefix, respectively.

└─ Zero, not the letter O, followed by b or h.

0b *binaryNumber*

0h *hexadecimalNumber*

└─ A binary number can have up to 32 digits. A hexadecimal number can have up to 8.

Without a prefix, *integer1* is treated as decimal (base 10). The result is displayed in binary, regardless of the Base mode.

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

256►Bin **ENTER** 0b100000000
 0h1F►Bin **ENTER** 0b11111

BldData CATALOG

BldData [dataVar]

Creates data variable *dataVar* based on the information used to plot the current graph. **BldData** is valid in all graphing modes.

If *dataVar* is omitted, the data is stored in the system variable *sysData*.

Note: The first time you start the Data/Matrix Editor after using **BldData**, *dataVar* or *sysData* (depending on the argument you used with **BldData**) is set as the current data variable.

The incremental values used for any independent variables (*x* in the example to the right) are calculated according to the Window variable values.

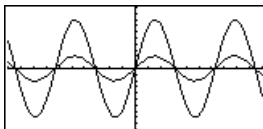
For information about the increments used to evaluate a graph, refer to the module that describes that graphing mode.

3D graphing mode has two independent variables. In the sample data to the right, notice that *x* remains constant as *y* increments through its range of values.

Then, *x* increments to its next value and *y* again increments through its range. This pattern continues until *x* has incremented through its range.

In function graphing mode and Radian angle mode:

8* $\sin(x)$ → $y_1(x)$ [ENTER] Done
 2* $\sin(x)$ → $y_2(x)$ [ENTER] Done
 ZoomStd [ENTER]



♦ [CALC HOME]

BldData [ENTER] Done
 [APPS] 6 [ENTER]

DATA	x	y1	y2
	c1	c2	c3
1	-10.	4.3522	1.0888
2	-9.832	3.168	.792
3	-9.664	1.8945	.47363
4	-9.496	.56769	.14192

Note: The following sample data is from a 3D graph.

DATA	x	y	z1
	c1	c2	c3
1	-10.	-10.	0.
2	-10.	-8.5715	5.8309
3	-10.	-7.1438	9.706
4	-10.	-5.7149	8.677

ceiling() MATH/Number menu

ceiling(*expression?*) ⇒ *integer*

Returns the nearest integer that is ≥ the argument.

The argument can be a real or a complex number.

Note: See also **floor**().

ceiling(*list?*) ⇒ *list*

ceiling(*matrix?*) ⇒ *matrix*

Returns a list or matrix of the ceiling of each element.

ceiling(0.456) [ENTER] 1.

ceiling({-3.1,1.2,5}) [ENTER] {-3. 1 3.}

ceiling([0. -3.2*i*; 1.3,4]) [ENTER] $\begin{bmatrix} 0 & -3. \\ 2. & 4 \end{bmatrix}$

cFactor() MATH/Algebra/Complex menu

cFactor(*expression*1, *var*) ⇒ *expression*

cFactor(a^3*x^2+a*x^2+a^3+a) [ENTER]

cFactor(*list*1, *var*) ⇒ *list*

cFactor(*matrix*1, *var*) ⇒ *matrix*

cFactor(x^2+4/9) [ENTER]

cFactor(*expression*1) returns *expression*1 factored with respect to all of its variables over a common denominator.

*expression*1 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(x^2+3) [ENTER]

$x^2 + 3$

cFactor(x^2+a) [ENTER]

$x^2 + a$

cFactor(*expression*1, *var*) returns *expression*1 factored with respect to variable *var*.

cFactor(a^3*x^2+a*x^2+a^3+a, x) [ENTER]

$a \cdot (a^2 + 1) \cdot (x + \sqrt{a}) \cdot (x + \sqrt{a})$

*expression*1 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(x^2+3, x) [ENTER]

$(x + \sqrt{3} \cdot i) \cdot (x + -\sqrt{3} \cdot i)$

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.

cFactor(x^2+a, x) [ENTER]

$(x + \sqrt{a} \cdot i) \cdot (x + \sqrt{a} \cdot i)$

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]

$x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3$

cFactor(ans(1), x) [ENTER]

$(x + 1.11 - 1.07 \cdot i)$

Note: See also **factor()**.

char() MATH/String menu

char(*integer*) ⇒ *character*

char(38) [ENTER]

"&"

Returns a character string containing the character numbered *integer* from the TI-89 Titanium/Voyage™ 200 character set. See Appendix B for a complete listing of character codes. The valid range for *integer* is 0–255.

char(65) [ENTER]

"A"

checkTmr() CATALOG

checkTmr(*starttime*) ⇒ *integer*

startTmr() [ENTER]

148083315

Returns an integer representing the number of seconds that have elapsed since a timer was started. *starttime* is an integer returned from the **startTmr()** function.

checkTmr(148083315)

34

You can also use a list or matrix of *starttime* integers. Valid *starttime* integers must fall between 0 and the current time of the clock. You can run multiple timers simultaneously.

startTmr()>Timer1

⋮

startTmr()>Timer2

⋮

checkTmr(Timer1)>Timer1Value

⋮

checkTmr(Timer2)>Timer2Value

Note: See also **startTmr()** and **timeCnv()**.

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).

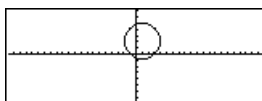
If $drawMode = 0$, turns off the circle.

If $drawMode = -1$, inverts pixels along the circle.

Note: Regraphing erases all drawn items. See also **PxlCrcl**.

In a ZoomSqr viewing window:

ZoomSqr:Circle 1,2,3 [ENTER]



ClockOff CATALOG

ClockOff

Turns the clock OFF.

ClockOn CATALOG

ClockOn

Turns the clock ON.

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** and **Try**.

Program listing:

```
:clearerr()
:Prgm
:PlotsOff:FnOff:ZoomStd
:For i,0,238
:Δx*i+xmin→xcord
: Try
: PtOn xcord,ln(xcord)
: Else
: If errornum=800 or
  errornum=260 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** or **Table**.)

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.

For functions such as **solve()** that return arbitrary constants or integers (@1, @2, etc.), **ClrHome** resets the suffix to 1.

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*) \Rightarrow *expression*

colDim([0,1,2;3,4,5]) **[ENTER]**

3

Returns the number of columns contained in *matrix*.

Note: See also **rowDim()**.

colNorm() MATH/Matrix/Norms menu

colNorm(*matrix*) \Rightarrow *expression*

[1, -2,3;4,5, -6] \rightarrow mat **[ENTER]**

Returns the maximum of the sums of the absolute values of the elements in the columns in *matrix*.

$$\begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix}$$
 9

Note: Undefined matrix elements are not allowed. See also **rowNorm()**.

comDenom() MATH/Algebra menu

comDenom(*expression1*, *var*) \Rightarrow *expression*

comDenom(*list1*, *var*) \Rightarrow *list*

comDenom(*matrix1*, *var*) \Rightarrow *matrix*

comDenom(*expression1*) returns a reduced ratio of a fully expanded numerator over a fully expanded denominator.

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.

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.

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.

Hint: Enter this **comden()** function definition and routinely try it as an alternative to **comDenom()** and **factor()**.

comDenom((y^2+y)/(x+1)^2+y^2+y)

[ENTER]

$$\text{comDenom}\left(\frac{y^2+y}{(x+1)^2} + y^2 + y\right) \\ \frac{x^2 \cdot y^2 + x^2 \cdot y + 2 \cdot x \cdot y^2 + 2 \cdot x \cdot y}{x^2 + 2 \cdot x + 1}$$

comDenom((y^2+y)/(x+1)

^2+y^2+y,x)

[ENTER]

$$\text{comDenom}\left(\frac{y^2+y}{(x+1)^2} + y^2 + y, x\right) \\ \frac{x^2 \cdot y \cdot (y+1) + 2 \cdot x \cdot y \cdot (y+1)}{x^2 + 2 \cdot x + 1}$$

comDenom((y^2+y)/(x+1)

^2+y^2+y,y)

[ENTER]

$$\text{comDenom}\left(\frac{y^2+y}{(x+1)^2} + y^2 + y, y\right) \\ \frac{y^2 \cdot (x^2 + 2 \cdot x + 2) + y \cdot (x^2 + 2 \cdot x + 1)}{x^2 + 2 \cdot x + 1}$$

comDenom(exprn,abc) \rightarrow comden

(exrn) [ENTER]

Don

comden((y^2+y)/(x+1)^2+y^2+y)

[ENTER]

$$\text{comden}\left(\frac{y^2+y}{(x+1)^2} + y^2 + y\right) \\ \frac{(x^2 + 2 \cdot x + 2) \cdot y \cdot (y+1)}{(x+1)^2}$$

comden(1234x^2*(y^3-y)+2468x*(y^2-1)) [ENTER]

$$1234 \cdot x \cdot (x \cdot y + 2) \cdot (y^2 - 1)$$

conj() MATH/Complex menu

conj(*expression1*) \Rightarrow *expression*

conj(*list1*) \Rightarrow *list*

conj(*matrix1*) \Rightarrow *matrix*

Returns the complex conjugate of the argument.

Note: All undefined variables are treated as real variables.

conj(1+2i) [ENTER]

$$1 - 2 \cdot i$$

conj([2,1-3i;-i,-7]) [ENTER]

$$\begin{bmatrix} 2 & 1+3 \cdot i \\ i & -7 \end{bmatrix}$$

conj(z)

$$\bar{z}$$

conj(x+iy)

$$x - i \cdot y$$

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 (\Rightarrow) 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.

$x+y \Rightarrow a$	$\boxed{\text{ENTER}}$	$x + y$
$10 \Rightarrow x$	$\boxed{\text{ENTER}}$	10
CopyVar a.b	$\boxed{\text{ENTER}}$	Done
$a \Rightarrow c$	$\boxed{\text{ENTER}}$	$y + 10$
DelVar x	$\boxed{\text{ENTER}}$	Done
b	$\boxed{\text{ENTER}}$	$x + y$
c	$\boxed{\text{ENTER}}$	$y + 10$

cos()

$\boxed{\text{COS}}$ key

$\text{cos}(\text{expression}) \Rightarrow \text{expression}$

$\text{cos}(\text{list}) \Rightarrow \text{list}$

$\text{cos}(\text{expression})$ returns the cosine of the argument as an expression.

$\text{cos}(\text{list})$ returns a list of the cosines of all elements in *list*.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

$\text{cos}((\pi/4)^\circ)$ $\boxed{\text{ENTER}}$ $\frac{\sqrt{2}}{2}$

$\text{cos}(45)$ $\boxed{\text{ENTER}}$ $\frac{\sqrt{2}}{2}$

$\text{cos}\{0.60, 90\}$ $\boxed{\text{ENTER}}$ $\{1 \ 1/2 \ 0\}$

In Gradian angle mode:

$\text{cos}\{0.50, 100\}$ $\boxed{\text{ENTER}}$ $\{1 \ \frac{\sqrt{2}}{2} \ 0\}$

In Radian angle mode:

$\text{cos}(\pi/4)$ $\boxed{\text{ENTER}}$ $\frac{\sqrt{2}}{2}$

$\text{cos}(45^\circ)$ $\boxed{\text{ENTER}}$ $\frac{\sqrt{2}}{2}$

cos(*squareMatrix1*) \Rightarrow *squareMatrix*

Returns the matrix cosine of *squareMatrix1*. This is *not* the same as calculating the cosine of each element.

When a scalar function $f(A)$ operates on *squareMatrix1* (A), the result is calculated by the algorithm:

1. Compute the eigenvalues (λ_i) and eigenvectors (V_i) of A .

squareMatrix1 must be diagonalizable. Also, it cannot have symbolic variables that have not been assigned a value.

2. Form the matrices:

$$B = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \text{ and } X = [V_1, V_2, \dots, V_n]$$

3. Then $A = X B X^{-1}$ and $f(A) = X f(B) X^{-1}$. For example, $\cos(A) = X \cos(B) X^{-1}$ where:

$$\cos(B) = \begin{bmatrix} \cos(\lambda_1) & 0 & \dots & 0 \\ 0 & \cos(\lambda_2) & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \cos(\lambda_n) \end{bmatrix}$$

All computations are performed using floating-point arithmetic.

In Radian angle mode:

$\cos([1.5, 3; 4.2, 1; 6. - 2, 1])$ [ENTER]

$$\begin{bmatrix} .212... & .205... & .121... \\ .160... & .259... & .037... \\ .248... & -.090... & .218... \end{bmatrix}$$

cos⁻¹ () [2nd] [COS⁻¹] key

cos⁻¹(*expression1*) \Rightarrow *expression*

cos⁻¹(*list1*) \Rightarrow *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 a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

$\cos^{-1}(1)$ [ENTER] 0

In Gradian angle mode:

$\cos^{-1}(0)$ [ENTER] 100

In Radian angle mode:

$\cos^{-1}(\{0, .2, .5\})$ [ENTER] $\left\{\frac{\pi}{2}, 1.369..., 1.047...\right\}$

cos⁻¹(*squareMatrix1*) \Rightarrow *squareMatrix*

Returns the matrix inverse cosine of *squareMatrix1*. This is *not* the same as calculating the inverse cosine of each element. For information about the calculation method, refer to **cos**().

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode and Rectangular complex format mode:

$\cos^{-1}([1.5, 3; 4.2, 1; 6. - 2, 1])$ [ENTER]

$$\begin{bmatrix} 1.734...+0.64...i^{\circ} - 1.490...+2.105...i^{\circ} \\ -.725...+1.515...i^{\circ} .623...+ .778...i^{\circ} \\ -2.083...+2.632...i^{\circ} 1.790...- 1.271...i^{\circ} \end{bmatrix}$$

cosh() MATH/Hyperbolic menu**cosh**(*expression1*) \Rightarrow *expression*cosh(1.2) **[ENTER]** 1.810...**cosh**(*list1*) \Rightarrow *list*cosh({0,1.2}) **[ENTER]** {1 1.810...}**cosh** (*expression1*) returns the hyperbolic cosine of the argument as an expression.**cosh** (*list1*) returns a list of the hyperbolic cosines of each element of *list1*.**cosh**(*squareMatrix1*) \Rightarrow *squareMatrix*

In Radian angle mode:

Returns the matrix hyperbolic cosine of *squareMatrix1*. This is *not* the same as calculating the hyperbolic cosine of each element. For information about the calculation method, refer to **cos()**.

cosh([1.5,3;4.2,1;6,-2.1])

[ENTER]

421.255	253.909	216.905
327.635	255.301	202.958
226.297	216.623	167.628

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.**cosh⁻¹()** MATH/Hyperbolic menu**cosh⁻¹**(*expression1*) \Rightarrow *expression*cosh⁻¹(1) **[ENTER]** 0**cosh⁻¹**(*list1*) \Rightarrow *list*cosh⁻¹({1,2,1,3}) **[ENTER]**
{0 1.372... cosh⁻¹(3)}**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**(*squareMatrix1*) \Rightarrow *squareMatrix*

In Radian angle mode and Rectangular complex format mode:

Returns the matrix inverse hyperbolic cosine of *squareMatrix1*. This is *not* the same as calculating the inverse hyperbolic cosine of each element. For information about the calculation method, refer to **cos()**.cosh⁻¹([1.5,3;4.2,1;6,-2.1]) **[ENTER]**

2.525...+1.734... <i>i</i> ^{°°} - .009...- 1.490... <i>i</i> ^{°°} ...
.486...- .725... <i>i</i> ^{°°°°} 1.662...+ .623... <i>i</i> ^{°°°°} ...
- .322...- 2.083... <i>i</i> ^{°°} 1.267...+1.790... <i>i</i> ^{°°} ...

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.**cot()** MATH/Trig menu**cot**(*expression1*) \Rightarrow *expression*

In Degree angle mode:

cot(*list1*) \Rightarrow *list*cot(45) **[ENTER]** 1Returns the cotangent of *expression1* or returns a list of the cotangents of all elements in *list1*.

In Gradian angle mode:

cot(50) **[ENTER]** 1**Note:** The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Radian angle mode:

cot({1,2,1,3}) **[ENTER]**
{ $\frac{1}{\tan(1)}$ - .584... $\frac{1}{\tan(3)}$ }

cot⁻¹()		MATH/Trig menu	
$\text{cot}^{-1}(\text{expression1}) \Rightarrow \text{expression}$		In Degree angle mode:	
$\text{cot}^{-1}(\text{list1}) \Rightarrow \text{list}$		$\text{cot}^{-1}(1)$ [ENTER]	45
Returns the angle whose cotangent is <i>expression1</i> or returns a list containing the inverse cotangents of each element of <i>list1</i> .		In Gradian angle mode:	
		$\text{cot}^{-1}(1)$ [ENTER]	50
Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.		In Radian angle mode:	
		$\text{cot}^{-1}(1)$ [ENTER]	$\frac{\pi}{4}$

coth()		MATH/Hyperbolic menu	
$\text{coth}(\text{expression1}) \Rightarrow \text{expression}$		$\text{coth}(1.2)$ [ENTER]	1.199...
$\text{coth}(\text{list1}) \Rightarrow \text{list}$		$\text{coth}(\{1.3,2\})$ [ENTER]	$\left\{ \frac{1}{\tanh(1)} \right.$ 1.003... $\left. \right\}$
Returns the hyperbolic cotangent of <i>expression1</i> or returns a list of the hyperbolic cotangents of all elements of <i>list1</i> .			

coth⁻¹()		MATH/Hyperbolic menu	
$\text{coth}^{-1}(\text{expression1}) \Rightarrow \text{expression}$		$\text{coth}^{-1}(3.5)$ [ENTER]	.293...
$\text{coth}^{-1}(\text{list1}) \Rightarrow \text{list}$		$\text{coth}^{-1}(\{-2,2,1,6\})$ [ENTER]	$\left\{ \frac{-\ln(3)}{2} \right.$.518... $\left. \frac{\ln(7/5)}{2} \right\}$
Returns the inverse hyperbolic cotangent of <i>expression1</i> or returns a list containing the inverse hyperbolic cotangents of each element of <i>list1</i> .			

crossP()		MATH/Matrix/Vector ops menu	
$\text{crossP}(\text{list1}, \text{list2}) \Rightarrow \text{list}$		$\text{crossP}(\{a1,b1\}, \{a2,b2\})$ [ENTER]	$\{0 \ 0 \ a1 \cdot b2 - a2 \cdot b1\}$
Returns the cross product of <i>list1</i> and <i>list2</i> as a list.			
<i>list1</i> and <i>list2</i> must have equal dimension, and the dimension must be either 2 or 3.		$\text{crossP}(\{0,1,2,2,-5\}, \{1,-.5,0\})$ [ENTER]	$\{-2.5 \ -5. \ -2.25\}$
$\text{crossP}(\text{vector1}, \text{vector2}) \Rightarrow \text{vector}$		$\text{crossP}([1,2,3],[4,5,6])$ [ENTER]	$[-3 \ 6 \ -3]$
Returns a row or column vector (depending on the arguments) that is the cross product of <i>vector1</i> and <i>vector2</i> .		$\text{crossP}([1,2],[3,4])$ [ENTER]	$[0 \ 0 \ -2]$
Both <i>vector1</i> and <i>vector2</i> 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.			

csc()		MATH/Trig menu	
$\text{csc}(\text{expression1}) \Rightarrow \text{expression}$		In Degree angle mode:	
$\text{csc}(\text{list1}) \Rightarrow \text{list}$		$\text{csc}(45)$ [ENTER]	$\sqrt{2}$
Returns the cosecant of <i>expression1</i> or returns a list containing the cosecants of all elements in <i>list1</i> .		In Gradian angle mode:	
		$\text{csc}(50)$ [ENTER]	$\sqrt{2}$
		In Radian angle mode:	
		$\text{csc}(\{1,\pi/2,\pi/3\})$ [ENTER]	$\left\{ \frac{1}{\sin(1)} \right.$ 1 $\left. \frac{2 \cdot \sqrt{3}}{3} \right\}$

csc⁻¹() **MATH/Trig menu**

csc⁻¹(*expression1*) ⇒ *expression* In Degree angle mode:
csc⁻¹(*list1*) ⇒ *list* csc⁻¹(1) **[ENTER]** 90

Returns the angle whose cosecant is *expression1* or returns a list containing the inverse cosecants of each element of *list1*.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Gradian angle mode:
 csc⁻¹(1) **[ENTER]** 100

In Radian angle mode:
 csc⁻¹({1.4,6}) **[ENTER]**

$$\left\{ \frac{\pi}{2} \sin^{-1}(1/4) \sin^{-1}(1/6) \right\}$$

csch() **MATH/Hyperbolic menu**

csch(*expression1*) ⇒ *expression* csch(3) **[ENTER]** $\frac{1}{\sinh(3)}$
csch(*list1*) ⇒ *list* csch({1.2,1.4}) **[ENTER]**

Returns the hyperbolic cosecant of *expression1* or returns a list of the hyperbolic cosecants of all elements of *list1*.

$$\left\{ \frac{1}{\sinh(1)} .248... \frac{1}{\sinh(4)} \right\}$$

csc⁻¹() **MATH/Hyperbolic menu**

csc⁻¹(*expression1*) ⇒ *expression* csc⁻¹(1) **[ENTER]** sinh⁻¹(1)
csc⁻¹(*list1*) ⇒ *list* csc⁻¹({1.2,1.3}) **[ENTER]**

Returns the inverse hyperbolic cosecant of *expression1* or returns a list containing the inverse hyperbolic cosecants of each element of *list1*.

$$\{\sinh^{-1}(1) .459... \sinh^{-1}(1/3)\}$$

cSolve() **MATH/Algebra/Complex menu**

cSolve(*equation, var*) ⇒ *Boolean expression* cSolve(x^3=-1,x) **[ENTER]**
 solve(x^3=-1,x) **[ENTER]**

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.

Although the TI-89 Titanium/Voyage™ 200 processes all undefined variables that do not end with an underscore () as if they were real, **cSolve()** can solve polynomial equations for complex solutions.

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.

Note: See also **cZeros()**, **solve()**, and **zeros()**.

Note: If *equation* is non-polynomial with functions such as **abs()**, **angle()**, **conj()**, **real()**, or **imag()**, you should place an underscore **_** (**2nd** **[_]**) at the end of *var*. By default, a variable is treated as a real value.

If you use *var_*, the variable is treated as complex.

You should also use *var_* for any other variables in *equation* that might have unreal values. Otherwise, you may receive unexpected results.

Display Digits mode in Fix 2:

`exact(cSolve(x^5+4x^4+5x^3-6x-3=0,x))` **[ENTER]**

`cSolve(ans(1),x)` **[ENTER]**

<ul style="list-style-type: none"> ■ <code>exact(cSolve(x^5+4x^4+5x^3-6x-3=0,x))</code> ■ <code>cSolve(x^5+4x^4+5x^3-6x-3=0,x)</code>

z is treated as real:

`cSolve(conj(z)=1+i,z)` **[ENTER]**

z=1+i

z_ is treated as complex:

`cSolve(conj(z_)=1+i,z_)` **[ENTER]**

z_=1-i

cSolve(*equation1 and equation2 [and ...]*,
{*varOrGuess1, varOrGuess2 [, ...]*})
⇒ *Boolean expression*

Returns candidate complex solutions to the simultaneous algebraic equations, where each *varOrGuess* specifies a variable that you want to solve for.

Optionally, you can specify an initial guess for a variable. Each *varOrGuess* must have the form:

variable
– or –
variable = *real or non-real number*

For example, *x* is valid and so is *x*=3+i.

If all of the equations are polynomials and if you do NOT specify any initial guesses, **cSolve()** uses the lexical Gröbner/Buchberger elimination method to attempt to determine **all** complex solutions.

Complex solutions can include both real and non-real solutions, as in the example to the right.

Note: The following examples use an underscore **_** so that the variables will be treated as complex.

`cSolve(u_*v_-u=v_ and v_^2=-u_,{u_,v_})`
[ENTER]

$$u_ = 1/2 + \frac{\sqrt{3}}{2} \cdot i \text{ and } v_ = 1/2 - \frac{\sqrt{3}}{2} \cdot i$$

$$\text{or } u_ = 1/2 - \frac{\sqrt{3}}{2} \cdot i \text{ and } v_ = 1/2 + \frac{\sqrt{3}}{2} \cdot i$$

$$\text{or } u_ = 0 \text{ and } v_ = 0$$

Simultaneous *polynomial* equations can have extra variables that have no values, but represent given numeric values that could be substituted later.

`cSolve(u_*v_-u=c_*v_ and v_^2=-u_,{u_,v_})` **[ENTER]**

$$u_ = \frac{-(\sqrt{1-4 \cdot c} + 1)}{4} \text{ and } v_ = \frac{\sqrt{1-4 \cdot c} + 1}{2}$$

or

$$u_ = \frac{-(\sqrt{1-4 \cdot c} - 1)}{4} \text{ and } v_ = \frac{-(\sqrt{1-4 \cdot c} - 1)}{2}$$

$$\text{or } u_ = 0 \text{ and } v_ = 0$$

You can also include solution variables that do not appear in the equations. These solutions show how families of solutions might contain arbitrary constants of the form $@k$, where k is an integer suffix from 1 through 255. The suffix resets to 1 when you use **ClrHome** or **[F1]**:Clear Home.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list solution variables. If your initial choice exhausts memory or your patience, try rearranging the variables in the equations and/or *varOrGuess* list.

If you do not include any guesses and if any equation is non-polynomial in any variable but all equations are linear in all solution variables, **cSolve()** uses Gaussian elimination to attempt to determine all solutions.

If a system is neither polynomial in all of its variables nor linear in its solution variables, **cSolve()** determines at most one solution using an approximate iterative method. To do so, the number of solution variables must equal the number of equations, and all other variables in the equations must simplify to numbers.

A non-real guess is often necessary to determine a non-real solution. For convergence, a guess might have to be rather close to a solution.

`cSolve(u_*v_-u_v_and
v_^2=-u_{u_v_w})` **[ENTER]**

$$u = 1/2 + \frac{\sqrt{3}}{2} \cdot i \text{ and } v = 1/2 - \frac{\sqrt{3}}{2} \cdot i$$

and $w = 0$

or

$$u = 1/2 - \frac{\sqrt{3}}{2} \cdot i \text{ and } v = 1/2 + \frac{\sqrt{3}}{2} \cdot i$$

and $w = 0$

or $u = 0$ and $v = 0$ and $w = 0$

`cSolve(u+v=e^{w_} and u-v=i, {u_v_})`
[ENTER]

$$u = \frac{e^w}{2} + 1/2 \cdot i \text{ and } v = \frac{e^w - i}{2}$$

`cSolve(e^{z_}=w_ and w=z_^2, {w_z_})`
[ENTER]

$$w = .494... \text{ and } z = -.703...$$

`cSolve(e^{z_}=w_ and w=z_^2, {w_z_}=1+i)`
[ENTER]

$$w = .149... + 4.891... \cdot i \text{ and } z = 1.588... + 1.540... \cdot i$$

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 *x*list.

list2 represents *y*list.

list3 represents frequency.

list4 represents category codes.

list5 represents category include list.

Note: *list1* through *list4* must be a variable name or $c1$ – $c9$ (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be $c1$ – $c9$.

In function graphing mode.

$\{0, 1, 2, 3\} \rightarrow L1$ **[ENTER]**

$\{0, 1, 2, 3\}$

$\{0, 2, 3, 4\} \rightarrow L2$ **[ENTER]**

$\{0, 2, 3, 4\}$

CubicReg L1, L2 **[ENTER]**

Done

ShowStat **[ENTER]**



[ENTER]

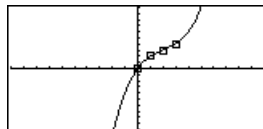
regeq(x) $\rightarrow y1(x)$ **[ENTER]**

Done

NewPlot 1, 1, L1, L2 **[ENTER]**

Done

[GRAPH]



cumSum() MATH/List menu

cumSum(*list*) ⇒ *list*

cumSum({1,2,3,4}) **ENTER**

Returns a list of the cumulative sums of the elements in *list*, starting at element 1.

{1 3 6 10}

cumSum(*matrix*) ⇒ *matrix*

Returns a *matrix* of the cumulative sums of the elements in *matrix*. Each element is the cumulative sum of the column from top to bottom.

[1,2:3,4:5,6] > m1 **ENTER**

cumSum(m1) **ENTER**

1	2
3	4
5	6
1	2
4	6
9	12

CustmOff CATALOG

CustmOff

See **Custom** program listing example.

Removes a custom toolbar.

CustmOn and **CustmOff** enable a program to control a custom toolbar. Manually, you can press **2nd** [CUSTOM] to toggle a custom toolbar on and off. Also, a custom toolbar is removed automatically when you change applications.

CustmOn CATALOG

CustmOn

See **Custom** program listing example.

Activates a custom toolbar that has already been set up in a **Custom...EndCustm** block.

CustmOn and **CustmOff** enable a program to control a custom toolbar. Manually, you can press **2nd** [CUSTOM] to toggle a custom toolbar on and off.

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 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
:Item
:Item
:Item
:Title
:Item
:Item
:Title
:EndCustm
:EndPrgm
```

```
"Lists"
>List1"
"Scores"
  "L3"
"Fractions"
  "f(x)"
  "h(x)"
  "Graph"
```

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.

Save three pics named pic1, pic2, and pic3.

Enter: CyclePic "pic",3,5,4,-1

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 $[r\angle\theta, z]$.

vector must have exactly three elements. It can be either a row or a column.

[2.2.3] ►Cylind [ENTER]

$[2 \cdot \sqrt{2} \angle \frac{\pi}{4} \quad 3]$

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 **explist(cSolve(expression=0, var), var)**. Otherwise, **cZeros()** is similar to **zeros()**.

Note: See also **cSolve()**, **solve()**, and **zeros()**.

Note: If *expression* is non-polynomial with functions such as **abs()**, **angle()**, **conj()**, **real()**, or **imag()**, you should place an underscore _ ([2nd] []) at the end of *var*. By default, a variable is treated as a real value. If you use *var_*, the variable is treated as complex.

You should also use *var_* for any other variables in *expression* that might have unreal values. Otherwise, you may receive unexpected results.

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}
```

z is treated as real:

```
cZeros(conj(z)-1-i,z) [ENTER]
{1+i}
```

z_ is treated as complex:

```
cZeros(conj(z_)-1-i,z_) [ENTER]
{1- i}
```


cZeros({expression1, expression2 [, ...]},
{varOrGuess1, varOrGuess2 [, ...]}) ⇒ matrix

Returns candidate positions where the expressions are zero simultaneously. Each *varOrGuess* specifies an unknown whose value you seek.

Optionally, you can specify an initial guess for a variable. Each *varOrGuess* must have the form:

variable
– or –
variable = *real or non-real number*

For example, x is valid and so is $x=3+i$.

If all of the expressions are polynomials and you do NOT specify any initial guesses, **cZeros()** uses the lexical Gröbner/Buchberger elimination method to attempt to determine **all** complex zeros.

Complex zeros can include both real and non-real zeros, as in the example to the right.

Each row of the resulting matrix represents an alternate zero, with the components ordered the same as the *varOrGuess* list. To extract a row, index the matrix by {row}.

Note: The following examples use an underscore *_* so that the variables will be treated as complex.

`cZeros({u*_v - u - v_._v_^2+u_},
{u_._v_})` **[ENTER]**

$$\begin{bmatrix} 1/2 - \frac{\sqrt{3}}{2} \cdot i & 1/2 + \frac{\sqrt{3}}{2} \cdot i \\ 1/2 + \frac{\sqrt{3}}{2} \cdot i & 1/2 - \frac{\sqrt{3}}{2} \cdot i \\ 0 & 0 \end{bmatrix}$$

Extract row 2:

`ans(1)[2]` **[ENTER]**

$$\left[1/2 + \frac{\sqrt{3}}{2} \cdot i \quad 1/2 - \frac{\sqrt{3}}{2} \cdot i \right]$$

Simultaneous *polynomials* can have extra variables that have no values, but represent given numeric values that could be substituted later.

`cZeros({u*_v - u - (c*_v_).
v_^2+u_}, {u_._v_})` **[ENTER]**

$$\begin{bmatrix} \frac{-(\sqrt{1-4 \cdot c_+1})^2}{4} & \frac{\sqrt{1-4 \cdot c_+1}}{2} \\ \frac{-(\sqrt{1-4 \cdot c_-1})^2}{4} & \frac{-(\sqrt{1-4 \cdot c_-1})}{2} \\ 0 & 0 \end{bmatrix}$$

You can also include unknown variables that do not appear in the expressions. These zeros show how families of zeros might contain arbitrary constants of the form $@k$, where k is an integer suffix from 1 through 255. The suffix resets to 1 when you use **ClrHome** or **[F1] 8:Clear Home**.

`cZeros({u*_v - u - v_._v_^2+u_},
{u_._v_._w_})` **[ENTER]**

$$\begin{bmatrix} 1/2 - \frac{\sqrt{3}}{2} \cdot i & 1/2 + \frac{\sqrt{3}}{2} \cdot i & @1 \\ 1/2 + \frac{\sqrt{3}}{2} \cdot i & 1/2 - \frac{\sqrt{3}}{2} \cdot i & @1 \\ 0 & 0 & @1 \end{bmatrix}$$

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list unknowns. If your initial choice exhausts memory or your patience, try rearranging the variables in the expressions and/or *varOrGuess* list.

If you do not include any guesses and if any expression is non-polynomial in any variable but all expressions are linear in all unknowns, **cZeros()** uses Gaussian elimination to attempt to determine all zeros.

`cZeros({u+_v_ - e^(w_).u - v_ - j_}, {u_._v_})`
[ENTER]

$$\begin{bmatrix} \frac{e^w}{2} + 1/2 \cdot i & \frac{e^w - j}{2} \end{bmatrix}$$

If a system is neither polynomial in all of its variables nor linear in its unknowns, **cZeros()** determines at most one zero using an approximate iterative method. To do so, the number of unknowns must equal the number of expressions, and all other variables in the expressions must simplify to numbers.

$$\text{cZeros}(\{e^{(z_)} - w_w - z_^2\}, \{w_z\})$$

[.494... -.703...]

A non-real guess is often necessary to determine a non-real zero. For convergence, a guess might have to be rather close to a zero.

$$\text{cZeros}(\{e^{(z_)} - w_w - z_^2\}, \{w_z = 1 + j\})$$

[.149...+4.89...j 1.588...+1.540...j]

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.

$$d(3x^3 - x + 7, x) \text{ [ENTER]} \quad 9x^2 - 1$$

$$d(3x^3 - x + 7, x, 2) \text{ [ENTER]} \quad 18 \cdot x$$

$$d(f(x) * g(x), x) \text{ [ENTER]}$$

$$\frac{d}{dx}(f(x)) \cdot g(x) + \frac{d}{dx}(g(x)) \cdot f(x)$$

$$d(\sin(f(x)), x) \text{ [ENTER]}$$

$$\cos(f(x)) \frac{d}{dx}(f(x))$$

$$d(x^3, x) |_{x=5} \text{ [ENTER]} \quad 75$$

$$d(x^2 * y^3, x, y) \text{ [ENTER]} \quad 6 \cdot y^2 \cdot x$$

$$d(x^2, x, -1) \text{ [ENTER]} \quad \frac{x^3}{3}$$

$$d\{x^2, x^3, x^4\}, x) \text{ [ENTER]} \quad \{2 \cdot x \quad 3 \cdot x^2 \quad 4 \cdot x^3\}$$

data mat CATALOG/MATH/List menu

data mat *data, mat[, row1] [, col1] [, row2] [, col2]*

data mat d1.m1.1. . . 1 ENTER

Converts data to a matrix.

Done

Each argument *[row1] [, col1] [, row2] [, col2]* can be individually omitted. If *row1* is omitted the default is 1. If *col1* is omitted the default is 1. If *row2* is omitted, the default is "max row." If *col2* is omitted, the default is "max column."

The DATA structure allows empty cells. Rows do not have to be equal size. When data is saved as a matrix, empty cells will be populated with "undef."

dayOfWk() CATALOG

dayOfWk(*year, month, day*) \Rightarrow *integer*

dayOfWk(1948,9,6)

2

Returns an integer from 1 to 7, with each integer representing a day of the week. Use **dayOfWk()** to determine on which day of the week a particular date would occur.

Note: May not give accurate results for years prior to 1583 (pre-Gregorian calendar).

Enter the year as a four-digit integer. The month and day can be either one- or two-digit integers.

Integer values:

- 1 = Sunday
- 2 = Monday
- 3 = Tuesday
- 4 = Wednesday
- 5 = Thursday
- 6 = Friday
- 7 = Saturday

►DD MATH/Angle menu

number ►DD \Rightarrow *value*

list1 ►DD \Rightarrow *list*

matrix1 ►DD \Rightarrow *matrix*

Returns the decimal equivalent of the argument expressed in degrees. The argument is a number, list, or matrix that is interpreted by the Mode setting in gradians, radians or degrees.

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 Gradian angle mode:

1 ►DD [ENTER] (9/10)°

In Radian angle mode:

1.5 ►DD [ENTER] 85.9°

►Dec MATH/Base menu

integer1 ►Dec \Rightarrow *integer*

0b10011 ►Dec [ENTER]

19

Converts *integer1* to a decimal (base 10) number. A binary or hexadecimal entry must always have a 0b or 0h prefix, respectively.

0h1F ►Dec [ENTER]

31

└ Zero, not the letter O, followed by b or h.

0b *binaryNumber*

0h *hexadecimalNumber*

└ A binary number can have up to 32 digits. A hexadecimal number can have up to 8.

Without a prefix, *integer1* is treated as decimal. The result is displayed in decimal, regardless of the Base mode.

Define CATALOG

<p>Define <i>funcName</i>(<i>arg1Name</i>, <i>arg2Name</i>, ...) = <i>expression</i></p> <p>Creates <i>funcName</i> as a user-defined function. You then can use <i>funcName</i>(()), just as you use built-in functions. The function evaluates <i>expression</i> using the supplied arguments and returns the result.</p> <p><i>funcName</i> cannot be the name of a system variable or built-in function.</p> <p>The argument names are placeholders; you should not use those same names as arguments when you use the function.</p> <p>Note: This form of Define is equivalent to executing the expression: <i>expression</i> → <i>funcName</i>(<i>arg1Name</i>, <i>arg2Name</i>). This command also can be used to define simple variables; for example, Define a=3.</p>	<pre>Define g(x,y)=2x-3y [ENTER] Done g(1,2) [ENTER] -4 1→a:2→b:g(a,b) [ENTER] -4 Define h(x)=when(x<2,2x-3, -2x+3) [ENTER] Done h(-3) [ENTER] -9 h(4) [ENTER] -5 Define eigenv1(a)= cZeros(det(identity(dim(a) [1])-x*a),x) [ENTER] Done eigenv1([-1,2;4,3]) [ENTER] { 2·√3 - 1 / 11, -(2·√3 + 1) / 11 }</pre>
--	--

<p>Define <i>funcName</i>(<i>arg1Name</i>, <i>arg2Name</i>, ...) = Func <i>block</i></p> <p>EndFunc</p> <p>Is identical to the previous form of Define, except that in this form, the user-defined function <i>funcName</i>(()) can execute a block of multiple statements.</p> <p><i>block</i> can be either a single statement or a series of statements separated with the ":" character. <i>block</i> also can include expressions and instructions (such as If, Then, Else, and For). This allows the function <i>funcName</i>(()) to use the Return instruction to return a specific result.</p> <p>Note: It is usually easier to author and edit this form of Function in the program editor rather than on the entry line.</p>	<pre>Define g(x,y)=Func:If x>y Then :Return x:Else:Return y:EndIf :EndFunc [ENTER] Done g(3, -7) [ENTER] 3</pre>
--	--

<p>Define <i>progName</i>(<i>arg1Name</i>, <i>arg2Name</i>, ...) = Prgm <i>block</i></p> <p>EndPrgm</p> <p>Creates <i>progName</i> as a program or subprogram, but cannot return a result using Return. Can execute a block of multiple statements.</p> <p><i>block</i> can be either a single statement or a series of statements separated with the ":" character. <i>block</i> also can include expressions and instructions (such as If, Then, Else, and For) without restrictions.</p> <p>Note: It is usually easier to author and edit a program block in the Program Editor rather than on the entry line.</p>	<pre>Define listnpt()=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 listnpt() [ENTER] Enter name of list</pre>
--	---

DelFold CATALOG

<p>DelFold <i>folderName1</i>, <i>folderName2</i> [, <i>folderName3</i>] ...</p> <p>Deletes user-defined folders with the names <i>folderName1</i>, <i>folderName2</i>, etc. An error message is displayed if the folders contain any variables.</p> <p>Note: You cannot delete the main folder.</p>	<pre>NewFold games [ENTER] Done (creates the folder games) DelFold games [ENTER] Done (deletes the folder games)</pre>
--	---

DelType CATALOG

DelType <i>var_type</i>	Deltype "LIST" ENTER	Done
Deletes all unlocked variables of the type specified by <i>var_type</i> .		
Note: Possible values for <i>var_type</i> are:		
ASM, DATA, EXPR, FUNC, GDB, LIST, MAT, PIC, PRGM, STR, TEXT, AppVar_type_name, All.		

DelVar CATALOG

DelVar <i>var1</i> [, <i>var2</i>] [, <i>var3</i>] ...	$2 \rightarrow a$ ENTER	2
Deletes the specified variables from memory.	$(a+2)^2$ ENTER	16
	DelVar a ENTER	Done
	$(a+2)^2$ ENTER	$(a+2)^2$

deSolve() MATH/Calculus menu

deSolve(*1stOr2ndOrderOde*, *independentVar*, *dependentVar*) \Rightarrow a general solution

Returns an equation that explicitly or implicitly specifies a general solution to the 1st- or 2nd-order ordinary differential equation (ODE). In the ODE:

- Use a prime symbol (' , press **2nd** [']) to denote the 1st derivative of the dependent variable with respect to the independent variable.
- Use two prime symbols to denote the corresponding second derivative.

The ' symbol is used for derivatives within **deSolve**() only. In other cases, use **d**).

The general solution of a 1st-order equation contains an arbitrary constant of the form @k, where k is an integer suffix from 1 through 255. The suffix resets to 1 when you use **ClrHome** or **F1** 8: Clear Home. The solution of a 2nd-order equation contains two such constants.

Apply **solve**() to an implicit solution if you want to try to convert it to one or more equivalent explicit solutions.

When comparing your results with textbook or manual solutions, be aware that different methods introduce arbitrary constants at different points in the calculation, which may produce different general solutions.

Note: To type a prime symbol ('), press **2nd** ['] .

deSolve(y''+2y'+y=x^2,x,y) ENTER	
$y = (@1 \cdot x + @2) \cdot e^{-x} + x^2 - 4 \cdot x + 6$	
right(ans(1)) \rightarrow temp ENTER	
$(@1 \cdot x + @2) \cdot e^{-x} + x^2 - 4 \cdot x + 6$	
d(temp,x,2)+2*d(temp,x)+temp-x^2 ENTER	0
DelVar temp ENTER	Done

deSolve(y'=(cos(y))^2*x,x,y) ENTER	
$\tan(y) = \frac{x^2}{2} + @3$	

solve(ans(1),y) ENTER	
$y = \tan^{-1} \left(\frac{x^2 + 2 \cdot @3}{2} \right) + @n1 \cdot \pi$	

Note: To type an @ symbol, press:

2nd R

ans(1) @3=c-1 and @n1=0 ENTER	
$y = \tan^{-1} \left(\frac{x^2 + 2 \cdot (c-1)}{2} \right)$	

deSolve(*1stOrderOde and initialCondition*, *independentVar*, *dependentVar*)
 \Rightarrow a particular solution

Returns a particular solution that satisfies *1stOrderOde* and *initialCondition*. This is usually

sin(y)=(y*e^x+cos(y))y' \rightarrow ode ENTER	
sin(y)=(e^x*y+cos(y))*y'	

deSolve(ode and y(0)=0,x,y) \rightarrow soln **ENTER**

easier than determining a general solution, substituting initial values, solving for the arbitrary constant, and then substituting that value into the general solution.	$\frac{-(2 \cdot \sin(y) + y^2)}{2} = -(e^{-1}) \cdot e^{y \cdot \sin(y)}$	
<i>initialCondition</i> is an equation of the form:	<code>soIn x=0 and y=0</code> [ENTER]	true
<code>dependentVar (initialIndependentValue) = initialDependentValue</code>	<code>d(right(eq)-left(eq),x)/ (d(left(eq)-right(eq),y)) → impdif(eq,x,y)</code> [ENTER]	Done
The <i>initialIndependentValue</i> and <i>initialDependentValue</i> can be variables such as <i>x0</i> and <i>y0</i> that have no stored values. Implicit differentiation can help verify implicit solutions.	<code>ode y'=impdif(soIn,x,y)</code> [ENTER]	true
	<code>DelVar ode,soIn</code> [ENTER]	Done

deSolve(2ndOrderOde and initialCondition1 and initialCondition2, independentVar, dependentVar) ⇒ a particular solution	<code>deSolve(y''=y^(-1/2) and y(0)=0 and y'(0)=0,t,y)</code> [ENTER]	$\frac{2 \cdot y^{3/4}}{3} = t$
Returns a particular solution that satisfies 2ndOrderOde and has a specified value of the dependent variable and its first derivative at one point.	<code>solve(ans(1),y)</code> [ENTER]	$y = \frac{2^{2/3} \cdot (3 \cdot t)^{4/3}}{4} \text{ and } t \geq 0$
For <i>initialCondition1</i> , use the form:		
<code>dependentVar (initialIndependentValue) = initialDependentValue</code>		
For <i>initialCondition2</i> , use the form:		
<code>dependentVar^1 (initialIndependentValue) = initial1stDerivativeValue</code>		

deSolve(2ndOrderOde and boundaryCondition1 and boundaryCondition2, independentVar, dependentVar) ⇒ a particular solution	<code>deSolve(w'' - 2w'/x + (9+2/x^2)w = x * e^x and w(pi/6)=0 and w(pi/3)=0,x,w)</code> [ENTER]	$w = \frac{e^{\frac{\pi}{3}} \cdot x \cdot \cos(3 \cdot x)}{10} - \frac{e^{\frac{\pi}{6}} \cdot x \cdot \sin(3 \cdot x)}{10} + \frac{x \cdot e^x}{10}$
Returns a particular solution that satisfies 2ndOrderOde and has specified values at two different points.		

det() MATH/Matrix menu

<code>det(squareMatrix, tol) ⇒ expression</code>	<code>det([a,b;c,d])</code> [ENTER]	$a \cdot d - b \cdot c$
Returns the determinant of <i>squareMatrix</i> .	<code>det([1,2;3,4])</code> [ENTER]	-2
Optionally, any matrix element is treated as zero if its absolute value is less than <i>tol</i> . This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, <i>tol</i> is ignored.	<code>det(identity(3) - x * [1, -2, 3; -2, 4, 1; -6, -2, 7])</code> [ENTER]	$-(98 \cdot x^3 - 55 \cdot x^2 + 12 \cdot x - 1)$
<ul style="list-style-type: none"> If you use \square [ENTER] or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic. If <i>tol</i> is omitted or not used, the default tolerance is calculated as: $5E-14 \cdot \max(\mathbf{dim}(\mathbf{squareMatrix})) \cdot \mathbf{rowNorm}(\mathbf{squareMatrix})$ 	<code>[1E20,1:0.1]→mat1</code> <code>det(mat1)</code> [ENTER] <code>det(mat1..1)</code> [ENTER]	$\begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix}$ 0 1.E20

diag() MATH/Matrix menu

diag(list) ⇒ *matrix*

diag(rowMatrix) ⇒ *matrix*

diag(columnMatrix) ⇒ *matrix*

diag({2,4,6}) **[ENTER]**

$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$

Returns a matrix with the values in the argument list or matrix in its main diagonal.

diag(squareMatrix) ⇒ *rowMatrix*

Returns a row matrix containing the elements from the main diagonal of *squareMatrix*.

[4.6.8;1.2.3;5.7.9] **[ENTER]**

$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$

squareMatrix must be square.

diag(ans(1)) **[ENTER]**

[4 2 9]

Dialog CATALOG

Dialog

block

EndDialog

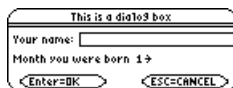
Generates a dialog box when the program is executed.

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  
:EndDialog  
:EndPrgm
```



dim() MATH/Matrix/Dimensions menu

dim(list) ⇒ *integer*

dim({0.1.2}) **[ENTER]**

3

Returns the dimension of *list*.

dim(matrix) ⇒ *list*

dim([1,-1.2;-2.3.5]) **[ENTER]**

{2 3}

Returns the dimensions of *matrix* as a two-element list {rows, columns}.

dim(string) ⇒ *integer*

dim("Hello") **[ENTER]**

5

Returns the number of characters contained in character string *string*.

dim("Hello"&" there") **[ENTER]**

11

Disp CATALOG

Disp [*exprOrString1*] [, *exprOrString2*] ...

Displays the current contents of the Program I/O screen. If one or more *exprOrString* is specified, each expression or character string is displayed on a separate line of the Program I/O screen.

An expression can include conversion operations such as **►DD** and **►Rect**. You can also use the **►** operator to perform unit and number base conversions.

If Pretty Print = ON, expressions are displayed in pretty print.

From the Program I/O screen, you can press **F5** to display the Home screen, or a program can use **DispHome**.

```
Disp "Hello" ENTER           Hello
Disp cos(2.3) ENTER         - .666..
{1,2,3,4}►L1 ENTER
Disp L1 ENTER                {1 2 3 4}
Disp 180_min►hr ENTER      3. ◦_hr
```

Note: To type an underscore (_), press **2nd** [**_**].
To type **►**, press **2nd** [**►**].

DispG CATALOG

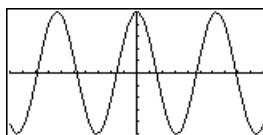
DispG

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
⋮
```



DispHome CATALOG

DispHome

Displays the current contents of the Home screen.

Program segment:

```
⋮
:Disp "The result is: ".xx
:Pause "Press Enter to quit"
:DispHome
:EndPrgm
```

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
```

F1	F2	F3	F4	F5	F6	F7
Tools	Setup	Flt	Fix	Eng	Mode	Draw
x	y1					
-2.	-2.081					
-1.	2.7015					
0.	5.					
1.	2.7015					
2.	-2.081					
x=-2.						
MAIN RAD AUTO FUNC						

►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 910 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 $\overline{\text{ENTER}}$ 45° 22' 15.6"
 {45.371.60} ►DMS $\overline{\text{ENTER}}$ {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}) $\overline{\text{ENTER}}$
 $a \cdot d + b \cdot e + c \cdot f$

dotP({1.2},{5.6}) $\overline{\text{ENTER}}$ 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]) $\overline{\text{ENTER}}$
 $a \cdot d + b \cdot e + c \cdot f$

dotP([1.2,3],[4,5,6]) $\overline{\text{ENTER}}$ 32

DrawFunc CATALOG

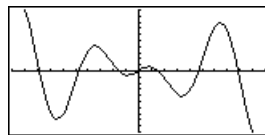
DrawFunc *expression*

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) $\overline{\text{ENTER}}$



DrawInv CATALOG

DrawInv *expression*

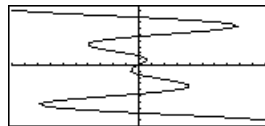
Draws the inverse of *expression* by plotting x values on the y axis and y values on the x axis.

x is the independent variable.

Note: Regraphing erases all drawn items.

In function graphing mode and ZoomStd window:

DrawInv 1.25x*cos(x) $\overline{\text{ENTER}}$



DrawParm CATALOG

DrawParm *expression1, expression2*
[, *tmin*] [, *tmax*] [, *tstep*]

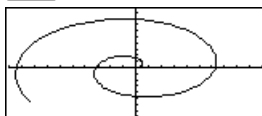
Draws the parametric equations *expression1* and *expression2*, using *t* as the independent variable.

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*]

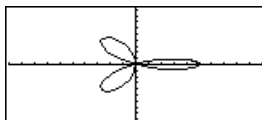
Draws the polar graph of *expression*, using θ as the independent variable.

Defaults for θ *min*, θ *max*, and θ *step* are the current settings for the Window variables θ *min*, θ *max*, and θ *step*. 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 $5*\cos(3*\theta),0,3.5,.1$ **ENTER**



DrawSlp CATALOG

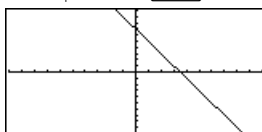
DrawSlp *x1, y1, slope*

Displays the graph and draws a line using the formula $y - y_1 = \text{slope} \cdot (x - x_1)$.

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.

DrwCtour CATALOG

DrwCtour *expression*

DrwCtour *list*

Draws contours on the current 3D graph at the z values specified by *expression* or *list*. The 3D graphing mode must already be set. **DrwCtour** automatically sets the graph format style to CONTOUR LEVELS.

By default, the graph automatically contains the number of equally spaced contours specified by the *ncontour* Window variable. **DrwCtour** draws contours in addition to the defaults.

To turn off the default contours, set *ncontour* to zero, either by using the Window screen or by setting 0 to the *ncontour* system variable.

In 3D graphing mode:

$(1/5)x^2+(1/5)y^2-10 \rightarrow z1(x,y)$ [ENTER]

$-10 \rightarrow x_{min}:10 \rightarrow x_{max}$ [ENTER]

$-10 \rightarrow y_{min}:10 \rightarrow y_{max}$ [ENTER]

$-10 \rightarrow z_{min}:10 \rightarrow z_{max}$ [ENTER]

$0 \rightarrow ncontour$ [ENTER]

DrwCtour $\{-9,-4.5,-3.0,4.5,9\}$ [ENTER]

Done
10
10
10
0



Use the cursor to change the viewing angle. Press 0 (zero) to return to the original view.

To toggle between different graph format styles, press:
[F]

Press X, Y, or Z to look down the corresponding axis.

E [2nd] [EE] **key**

mantissa **E** *exponent*

Enters a number in scientific notation. The number is interpreted as *mantissa* $\times 10^{\textit{exponent}}$.

Hint: If you want to enter a power of 10 without causing a decimal value result, use $10^{\textit{integer}}$.

$2.3 \text{E} 4$ [ENTER]

23000.

$2.3 \text{E} 9+4.1 \text{E} 15$ [ENTER]

$4.1 \text{E} 15$

$3 * 10^4$ [ENTER]

30000

e^() [2nd] [e^x] **key**

$e^{\textit{expression}1} \Rightarrow \textit{expression}$

Returns e raised to the *expression1* power.

Note: On the TI-89 Titanium, pressing [alpha] [e^x] to display $e^{\wedge}(\)$ is different from pressing [alpha] [E]. On the Voyage 200, pressing [2nd] [e^x] to display e^{\wedge} is different from accessing the character e from the QWERTY keyboard.

You can enter a complex number in $re^{i\theta}$ polar form. However, use this form in Radian angle mode only; it causes a Domain error in Degree or Gradian angle mode.

$e^{\wedge}(1)$ [ENTER]

e

$e^{\wedge}(1.)$ [ENTER]

2.718...

$e^{\wedge}(3)^2$ [ENTER]

e^9

$e^{\wedge}(\textit{list}1) \Rightarrow \textit{list}$

Returns e raised to the power of each element in *list1*.

$e^{\wedge}(\{1,1.,0.,.5\})$ [ENTER]

{ e 2.718... 1 1.648...}

$e^{\wedge}(\textit{squareMatrix}1) \Rightarrow \textit{squareMatrix}$

Returns the matrix exponential of *squareMatrix1*. This is *not* the same as calculating e raised to the power of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

$e^{\wedge}([1.5,3;4.2,1;6.-2,1])$ [ENTER]

782.209	559.617	456.509
680.546	488.795	396.521
524.929	371.222	307.879

eigVc() MATH/Matrix menu

eigVc(*squareMatrix*) ⇒ *matrix*

Returns a matrix containing the eigenvectors for a real or complex *squareMatrix*, where each column in the result corresponds to an eigenvalue. Note that an eigenvector is not unique; it may be scaled by any constant factor. The eigenvectors are normalized, meaning that if $V = [x_1, x_2, \dots, x_n]$, then:

$$\sqrt{x_1^2 + x_2^2 + \dots + x_n^2} = 1$$

squareMatrix is first balanced with similarity transformations until the row and column norms are as close to the same value as possible. The *squareMatrix* is then reduced to upper Hessenberg form and the eigenvectors are computed via a Schur factorization.

In Rectangular complex format mode:

```
[-1,2,5;3,-6,9;2,-5,7]>m1 ENTER
```

$$\begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix}$$

```
eigVc(m1) ENTER
```

$$\begin{bmatrix} -.800... & .767... & .767... \\ .484... & .573...+.052...i & .573...-.052...i \\ .352... & .262...+.096...i & .262...-.096...i \end{bmatrix}$$

eigVl() MATH/Matrix menu

eigVl(*squareMatrix*) ⇒ *list*

Returns a list of the eigenvalues of a real or complex *squareMatrix*.

squareMatrix is first balanced with similarity transformations until the row and column norms are as close to the same value as possible. The *squareMatrix* is then reduced to upper Hessenberg form and the eigenvalues are computed from the upper Hessenberg matrix.

In Rectangular complex format mode:

```
[-1,2,5;3,-6,9;2,-5,7]>m1 ENTER
```

$$\begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix}$$

```
eigVl(m1) ENTER
```

{ -4.409... 2.204...+.763...i 2.204...-.763...i }

Else See **If**, page 828.

Elseif **CATALOG** See also **If**, page 828.

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 801.

EndDlog See **Dialog**, page 809.

EndFor See **For**, page 821.

EndFunc See **Func**, page 822.

EndIf See **If**, page 828.

EndLoop See **Loop**, page 839.

EndPrgm See **Prgm**, page 854.

EndTBar See **ToolBar**, page 890.

EndTry See **Try**, page 890.

EndWhile See **While**, page 893.

entry() CATALOG

entry() \Rightarrow *expression*

entry(integer) \Rightarrow *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:

$$1+1/x \quad \text{ENTER} \quad \frac{1}{x} + 1$$

$$1+1/\text{entry}(1) \quad \text{ENTER} \quad 2 - \frac{1}{x+1}$$

$$\text{ENTER} \quad \frac{1}{2 \cdot (2 \cdot x+1)} + 3/2$$

$$\text{ENTER} \quad 5/3 - \frac{1}{3 \cdot (3 \cdot x+2)}$$

$$\text{entry}(4) \quad \text{ENTER} \quad \frac{1}{x} + 1$$

exact() MATH/Number menu

exact(expression1 [, tol]) \Rightarrow *expression*

exact(list1 [, tol]) \Rightarrow *list*

exact(matrix1 [, tol]) \Rightarrow *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).

$$\text{exact}(.25) \quad \text{ENTER} \quad 1/4$$

$$\text{exact}(.333333) \quad \text{ENTER} \quad \frac{333333}{1000000}$$

$$\text{exact}(.33333...001) \quad 1/3$$

$$\text{exact}(3.5x+y) \quad \text{ENTER} \quad \frac{7x}{2} + y$$

$$\text{exact}({.2,.33,4.125}) \quad \text{ENTER} \quad \left\{1/5 \frac{33}{100} 33/8\right\}$$

Exec CATALOG

Exec string [, *expression1*] [, *expression2*] ...

Executes a *string* consisting of a series of Motorola 68000 op-codes. These codes act as a form of an assembly-language program. If needed, the optional *expressions* let you pass one or more arguments to the program.

For more information, check the TI Web site:
http://www.ti.com/calculator

Warning: **Exec** gives you access to the full power of the microprocessor. Please be aware that you can easily make a mistake that locks up the calculator and causes you to lose your data. We suggest you make a backup of the calculator contents before attempting to use the **Exec** command.

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))` [ENTER]
{ -1 2 }

expand() MATH/Algebra menu

expand(*expression1* [, *var1*]) ⇒ *expression*

expand(*list1* [, *var1*]) ⇒ *list*

expand(*matrix1* [, *var1*]) ⇒ *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^2+x*y))` [ENTER]

$$\blacksquare \text{ expand} \left\{ \frac{x^2 - x + y^2 - y}{x^2 \cdot y^2 - x^2 \cdot y - x \cdot y^2} \cdot \frac{1}{x-1} - \frac{1}{x} + \frac{1}{y-1} - \frac{1}{y} \right\}$$

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^2+x*y),y)` [ENTER]

$$\blacksquare \text{ expand} \left\{ \frac{x^2 - x + y^2 - y}{x^2 \cdot y^2 - x^2 \cdot y - x \cdot y^2} \cdot \frac{1}{y-1} - \frac{1}{y} + \frac{1}{x \cdot (x-1)} \right\}$$

`expand(ans(1),x)` [ENTER]

$$\blacksquare \text{ expand} \left\{ \frac{1}{y-1} - \frac{1}{y} + \frac{1}{x \cdot (x-1)} \cdot \frac{1}{x-1} - \frac{1}{x} + \frac{1}{y \cdot (y-1)} \right\}$$

Even when there is only one variable, using *var* might make the denominator factorization used for partial fraction expansion more complete.

Hint: For rational expressions, **propFrac()** is a faster but less extreme alternative to **expand()**.

Note: See also **comDenom()** 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.

expand(expression1,[var]) also distributes absolute values, **sign()**, and exponentials, regardless of *var*.

Note: See also **tExpand()** for trigonometric angle-sum and multiple-angle expansion.

$$\text{expand}((x^3+x^2-2)/(x^2-2)) \text{ [ENTER]} \quad \frac{2 \cdot x}{x^2-2} + x+1$$

$$\text{expand}(\text{ans}(1),x) \text{ [ENTER]} \quad \frac{1}{x-\sqrt{2}} + \frac{1}{x+\sqrt{2}} + x+1$$

$$\ln(2x*y)+\sqrt{(2x*y)} \text{ [ENTER]} \quad \ln(2 \cdot x \cdot y) + \sqrt{(2 \cdot x \cdot y)}$$

$$\text{expand}(\text{ans}(1)) \text{ [ENTER]} \quad \ln(x \cdot y) + \sqrt{2} \cdot \sqrt{(x \cdot y)} + \ln(2)$$

$$\text{expand}(\text{ans}(1))|y>=0 \text{ [ENTER]} \quad \ln(x) + \sqrt{2} \cdot \sqrt{x} \cdot \sqrt{y} + \ln(y) + \ln(2)$$

$$\text{sign}(x*y)+\text{abs}(x*y)+e^{(2x+y)} \text{ [ENTER]} \quad e^{2 \cdot x+y} + \text{sign}(x \cdot y) + |x \cdot y|$$

$$\text{expand}(\text{ans}(1)) \text{ [ENTER]} \quad \text{sign}(x) \cdot \text{sign}(y) + |x| \cdot |y| + (e^x)^2 \cdot e^y$$

expr() MATH/String menu

expr(string) ⇒ *expression*

Returns the character string contained in *string* as an expression and immediately executes it.

$$\text{expr}("1+2+x^2+x") \text{ [ENTER]} \quad x^2 + x + 3$$

$$\text{expr}("\text{expand}((1+x)^2)") \text{ [ENTER]} \quad x^2 + 2 \cdot x + 1$$

"Define cube(x)=x^3" → funcstr [ENTER]
"Define cube(x)=x^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 ...}

Done

ExpReg L1,L2 **[ENTER]**

ShowStat **[ENTER]**



[ENTER]

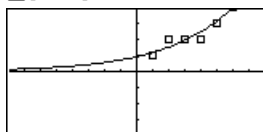
Regeq(x) → y1(x) **[ENTER]**

Done

NewPlot 1,1,L1,L2 **[ENTER]**

Done

[GRAPH]



factor() MATH/Algebra menu

factor(*expression1*, *var1*) ⇒ *expression*

factor(*list1*, *var1*) ⇒ *list*

factor(*matrix1*, *var1*) ⇒ *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(*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($a^3 * x^2 - a * x^2 - a^3 + a$) **[ENTER]**

$a \cdot (a - 1) \cdot (a + 1) \cdot (x - 1) \cdot (x + 1)$

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($a^3 * x^2 - a * x^2 - a^3 + a, x$) **[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.

Note: See also **comDenom()** for a fast way to achieve partial factoring when **factor()** is not fast enough or if it exhausts memory.

Note: See also **cFactor()** for factoring all the way to complex coefficients in pursuit of linear factors.

```
factor(x^5+4x^4+5x^3-6x-3) [ENTER]
x^5 + 4 · x^4 + 5 · x^3 - 6 · x - 3
factor(ans(1),x) [ENTER]
(x - .964...) · (x + .611...) ·
(x + 2.125...) · (x^2 + 2.227... ·
x + 2.392...)
```

factor(rationalNumber) returns the rational number factored into primes. For composite numbers, the computing time grows exponentially with the number of digits in the second-largest factor. For example, factoring a 30-digit integer could take more than a day, and factoring a 100-digit number could take more than a century.

Note: To stop (break) a computation, press **[ON]**.

If you merely want to determine if a number is prime, use **isPrime()** instead. It is much faster, particularly if *rationalNumber* is not prime and if the second-largest factor has more than five digits.

```
factor(152417172689) [ENTER]
123457 · 1234577
isPrime(152417172689) [ENTER]
false
```

Fill MATH/Matrix menu

Fill *expression, matrixVar* ⇒ *matrix*

Replaces each element in variable *matrixVar* with *expression*.

matrixVar must already exist.

```
[1,2;3,4] → amatrix [ENTER]
[ 1 2 ]
[ 3 4 ]
Fill 1.01.amatrix [ENTER]
Done
amatrix [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(list) ⇒ *list*

floor(matrix) ⇒ *matrix*

Returns a list or matrix of the floor of each element.

Note: See also **ceiling()** and **int()**.

```
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()** and **max()**.

fMax(1 - (x - a)^2 - (x - b)^2, x) **[ENTER]**

$$x = \frac{a+b}{2}$$

fMax(.5x^3 - x - 2, x) **[ENTER]**

$$x = \infty$$

fMax(.5x^3 - x - 2, x) | x ≤ 1 **[ENTER]**

$$x = -.816\dots$$

fMax(a * x^2, x) **[ENTER]**

$$x = \infty \text{ or } x = -\infty \text{ or } x = 0 \text{ or } a = 0$$

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()** and **min()**.

fMin(1 - (x - a)^2 - (x - b)^2, x) **[ENTER]**

$$x = \infty \text{ or } x = -\infty$$

fMin(.5x^3 - x - 2, x) | x ≥ 1 **[ENTER]**

$$x = 1$$

fMin(a * x^2, x) **[ENTER]**

$$x = \infty \text{ or } x = -\infty \text{ or } x = 0 \text{ or } a = 0$$

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*][*d*]", where [*n*] 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.

format(1.234567, "f3") **ENTER**

"1.235"

format(1.234567, "s2") **ENTER**

"1.23E 0"

format(1.234567, "e3") **ENTER**

"1.235E 0"

format(1.234567, "g3") **ENTER**

"1.235"

format(1234.567, "g3") **ENTER**

"1,234.567"

format(1.234567, "g3,r:") **ENTER**

"1:235"

G[R][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.

[Rc]: Any of the above specifiers may be suffixed with the R radix flag, where *c* is a single character that specifies what to substitute for the radix point.

fPart() MATH/Number menu

fPart(*expression*) \Rightarrow *expression*

fPart(-1.234) **ENTER** - .234

fPart(*list*) \Rightarrow *list*

fPart(*matrix*) \Rightarrow *matrix*

fPart({1, -2.3, 7.003}) **ENTER**
{0 - .3 .003}

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.

Func CATALOG

Func

block

EndFunc

Required as the first statement in a multi-statement function definition.

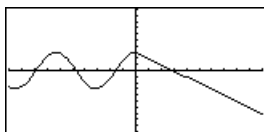
block can be either a single statement or a series of statements separated with the ";" character.

Note: **when()** also can be used to define and graph piecewise-defined functions.

In function graphing mode, define a piecewise function:

```
Define g(x)=Func:If x<0 Then
:Return 3*cos(x):Else:Return
3-x:EndIf:EndFunc ENTER Done
```

Graph g(x) **ENTER**



gcd() MATH/Number menu

gcd(*number1*, *number2*) \Rightarrow *expression*

gcd(18,33) **ENTER** 3

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.

In Auto or Approximate mode, the **gcd** of fractional floating-point numbers is 1.0.

gcd(*list1*, *list2*) \Rightarrow *list*

gcd({12,14,16},{9,7,5}) **ENTER**
{3 7 1}

Returns the greatest common divisors of the corresponding elements in *list1* and *list2*.

gcd(*matrix1*, *matrix2*) \Rightarrow *matrix*

gcd([2,4;6,8],[4,8;12,16]) **ENTER**
 $\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$

Returns the greatest common divisors of the corresponding elements in *matrix1* and *matrix2*.

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 1,1,99  
: 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"  
:
```

 **GetCalc** *var,[port]*

Retrieves a value from the link port and stores it in variable *var* on the receiving TI-89 Titanium.

If the port is not specified, or *port* = 0 is specified, the TI-89 Titanium waits for data from either port.

If *port* = 1, the TI-89 Titanium waits for data from the USB port.

If *port* = 2, the TI-89 Titanium waits for data from the I/O port.

getConfig() CATALOG

getConfig() ⇒ *ListPairs*

Returns a list of calculator attributes. The attribute name is listed first, followed by its value.

getConfig() [\[ENTER\]](#)

```
{ "Product Name" "Advanced  
Mathematics Software"  
"Version" "2.00. 09/25/1999"  
"Product ID" "01-1-4-80"  
"ID #" "01012 34567 ABCD"  
"Cert. Rev. #" 0  
"Screen Width" 240  
"Screen Height" 120  
"Window Width" 240  
"Window Height" 91  
"RAM Size" 262144  
"Free RAM" 192988  
"Archive Size" 720896  
"Free Archive" 720874 }
```

Note: Your screen may display different attribute values. The Cert. Rev. # attribute appears only if you have purchased and installed additional software into the calculator.

getDate() CATALOG

getDate() ⇒ *list*

Returns a list giving the date according to the current value of the clock. The list is in {*year,month,day*} format.

getDate() [\[ENTER\]](#)

```
{2002 2 22}
```

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** $x \cdot y$

getDtFmt() CATALOG

getDtFmt() ⇒ *integer*

Returns an integer representing the date format that is currently set on the device.

Integer values:

1 = MM/DD/YY

2 = DD/MM/YY

3 = MM.DD.YY

4 = DD.MM.YY

5 = YY.MM.DD

6 = MM-DD-YY

7 = DD-MM-YY

8 = YY-MM-DD

getDtStr() CATALOG

getDtStr([integer]) ⇒ *string*

Returns a string of the current date in the current date format. For example, a returned string of 28/09/02 represents the 28th day of September, 2002 (when the date format is set to DD/MM/YY).

If you enter the optional integer that corresponds to a date format, the string returns the current date in the specified format.

Optional integer values:

1 = MM/DD/YY

2 = DD/MM/YY

3 = MM.DD.YY

4 = DD.MM.YY

5 = YY.MM.DD

6 = MM-DD-YY

7 = DD-MM-YY

8 = YY-MM-DD

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 **[F]**, second function **[2nd]**, option **[⏏]**, alpha **[alpha]**, and drag **[⏏]**) are not recognized by themselves; however, they modify the keycodes of the key that follows them. For example: **[⏏]** **[X]** ≠ **[X]** ≠ **[2nd]** **[X]**.

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) \Rightarrow string

getMode("ALL") \Rightarrow 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()**.

Note: To set or return information about the Unit System mode, use **setUnits()** or **getUnits()** instead of **setMode()** or **getMode()**.

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"
"Base" "DEC" }

Note: Your screen may display different mode settings.

getNum() MATH/Algebra/Extract menu

getNum(expression1) \Rightarrow 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

getTime() CATALOG

getTime() \Rightarrow list

Returns a list giving the time according to the current value of the clock. The list is in {hour,minute,second} format. The time is returned in the 24 hour format.

getTmFmt() CATALOG

getTmFmt() \Rightarrow integer

Returns an integer representing the clock time format that is currently set on the device.

Integer values:

12 = 12 hour clock

24 = 24 hour clock

getTmStr() CATALOG

getTmStr(integer) \Rightarrow string

Returns a string of the current clock time in the current time format.

If you enter the optional integer that corresponds to a clock time format, the string returns the current time in the specified format.

Optional integer values:

12 = 12 hour clock

24 = 24 hour clock

getTmZn() CATALOG

getTmZn() ⇒ *integer*

Returns an integer representing the time zone that is currently set on the device.

The returned integer represents the number of minutes the time zone is offset from Greenwich Mean Time (GMT), as established in Greenwich, England. For example, if the time zone is offset from GMT by two hours, the device would return 120 (minutes).

Integers for time zones west of GMT are negative.

Integers for time zones east of GMT are positive.

If Greenwich Mean Time is 14:07:07, it is:

8:07:07 a.m. in Denver, Colorado (Mountain Daylight Time) (-360 minutes from GMT)

16:07:07 p.m. in Brussels, Belgium (Central European Standard Time) (+120 minutes from GMT)

getType() CATALOG

getType(*var*) ⇒ *string*

Returns a string indicating the 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+3 → temp [ENTER]          2 + 3i
getType(temp) [ENTER]      "EXPR"
```

```
DelVar temp [ENTER]         Done
getType(temp) [ENTER]      "NONE"
```

Data Type	Variable Contents
"ASM"	Assembly-language program
"DATA"	Data type
"EXPR"	Expression (includes complex/arbitrary/undefined, ∞ , $-\infty$, TRUE, FALSE, pi, <i>e</i>)
"FUNC"	Function
"GDB"	Graph data base
"LIST"	List
"MAT"	Matrix
"NONE"	Variable does not exist
"NUM"	Real number
"OTHER"	Miscellaneous data type for future use by software applications
"PIC"	Picture
"PRGM"	Program
"STR"	String
"TEXT"	Text type
"VAR"	Name of another variable

getUnits() CATALOG

getUnits() ⇒ *list*

Returns a list of strings that contain the current default units for all categories except constants, temperature, amount of substance, luminous intensity, and acceleration. *list* has the form:

```
{ "system" "cat1" "unit1" "cat2" "unit2" ... }
```

The first string gives the system (SI, ENG/US, or CUSTOM). Subsequent pairs of strings give a category (such as Length) and its default unit (such as *_m* for meters).

To set the default units, use **setUnits()**.

getUnits() [ENTER]

```
{"SI" "Area" "NONE"
 "Capacitance" "_F"
 "Charge" "_coul"
 ... }
```

Note: Your screen may display different default units.

Goto CATALOG

Goto *labelName*

Transfers program control to the label *labelName*.

labelName must be defined in the same program using a **Lbl** instruction.

Program segment:

```
:  
:0→temp  
:1→i  
:Lbl TOP  
: temp+i→temp  
: If i<10 Then  
: i+1→i  
: Goto TOP  
: EndIf  
:Disp temp  
:
```

►Grad CATALOG/MATH/Angle menu

►Grad *expression*

Converts an expression to gradian angle measure.

In Degree angle mode:

1.5 ►Grad **[ENTER]** 1.66667⁶

In Radian angle mode:

1.5 ►Grad **[ENTER]** 95.493⁶

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** 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.

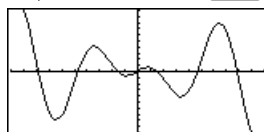
Some valid variations of this instruction are:

Function graphing	Graph <i>expr</i> , <i>x</i>
Parametric graphing	Graph <i>xExpr</i> , <i>yExpr</i> , <i>t</i>
Polar graphing	Graph <i>expr</i> , θ
Sequence graphing	Not allowed.
3D graphing	Graph <i>expr</i> , <i>x</i> , <i>y</i>
Diff Equations graphing	Not allowed.

Note: Use **ClrGraph** to clear these functions, or go to the Y= Editor to re-enable the system Y= functions.

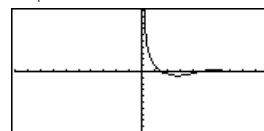
In function graphing mode and ZoomStd window:

Graph 1.25a*cos(a).a **[ENTER]**



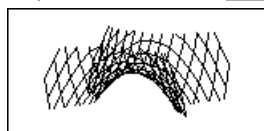
In parametric graphing mode and ZoomStd window:

Graph time.2cos(time)/time.time **[ENTER]**



In 3D graphing mode:

Graph (v^2 - w^2)/4.v.w **[ENTER]**



►Hex MATH/Base menu*integer1* ►Hex ⇒ *integer*

256 ►Hex [ENTER]

0h100

Converts *integer1* to a hexadecimal number. Binary or hexadecimal numbers always have a 0b or 0h prefix, respectively.

0b111100001111 ►Hex [ENTER]

0hF0F

└ Zero, not the letter O, followed by b or h.

0b *binaryNumber*0h *hexadecimalNumber*

└ A binary number can have up to 32 digits. A hexadecimal number can have up to 8.

Without a prefix, *integer1* is treated as decimal (base 10). The result is displayed in hexadecimal, regardless of the Base mode.

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

identity() MATH/Matrix menu**identity**(*expression*) ⇒ *matrix*

identity(4) [ENTER]

Returns the identity matrix with a dimension of *expression*.

expression must evaluate to a positive integer.

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

If CATALOG**If** *Boolean expression*
*statement***If** *Boolean expression* **Then**
block
EndIf

Program segment:

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.

```

:
:
: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(1+2i) [ENTER] 2

imag(*expression1*) returns the imaginary part of the argument. imag(z) [ENTER] 0

Note: All undefined variables are treated as real variables. See also **real()**. imag(x+iy) [ENTER] y

imag(*list1*) ⇒ *list* imag({-3,4-i,i}) [ENTER] {0 -1 1}

Returns a list of the imaginary parts of the elements.

imag(*matrix1*) ⇒ *matrix* imag([a.b:c.d]) [ENTER] $\begin{bmatrix} 0 & 0 \\ c & d \end{bmatrix}$

Returns a matrix of the imaginary parts of the elements.

ImpDif() MATH/Calculus Menu, CATALOG

ImpDif(*equation, independentVar, dependentVar, order 1*) ⇒ *expression* impDif(x^2+y^2=100,x,y)[ENTER] -x/y

where the order defaults to 1.

Computes the implicit derivative for equations in which one variable is defined implicitly in terms of another.

Indirection See #(), page 909.

Input CATALOG

Input Program segment: ⋮

Pauses the program, displays the current Graph screen, and lets you update variables *xc* and *yc* (also *rc* and *θc* for polar coordinate mode) by positioning the graph cursor. :● Get 10 points from the Graph Screen

When you press [ENTER], the program resumes. :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(*list*) ⇒ *list*

int(*matrix*) ⇒ *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 907.

iPart() MATH/Number menu

iPart(*number*) ⇒ *integer*

iPart(-1.234) **ENTER** -1.

iPart(*list*) ⇒ *list*

iPart(*matrix*) ⇒ *matrix*

iPart({3/2, -2.3, 7.003}) **ENTER**
{1 -2. 7.}

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.

isArchiv() CATALOG

isArchiv(*var_name*) ⇒ *true,false*

isArchiv(PROG1) **ENTER** True

Determines if *var_name* is archived or not. Returns true if *var_name* is archived. Returns false if *var_name* is not archived.

isClkOn() CATALOG

isClkOn() ⇒ *true,false*

Determines if the clock is ON or OFF. Returns true if the clock is ON. Returns false if the clock is OFF.

isLocked() CATALOG

isLocked(*var_name*) ⇒ *true,false*

isLocked(PROG1) **ENTER** False

Determines if *var_name* is locked or not. Returns true if *var_name* is locked or archived. Returns false if *var_name* is not locked or archived.

isPrime() MATH/Test menu

isPrime(*number*) ⇒ *Boolean constant expression*

IsPrime(5) **ENTER** true

IsPrime(6) **ENTER** false

Returns true or false to indicate if *number* is a whole number ≥ 2 that is evenly divisible only by itself and 1.

If *number* exceeds about 306 digits and has no factors ≤ 1021 , **isPrime**(*number*) displays an error message.

If you merely want to determine if *number* is prime, use **isPrime**() instead of **factor**(). It is much faster, particularly if *number* is not prime and has a second-largest factor that exceeds about five digits.

Function to find the next prime after a specified number:

```
Define nextPrim(n)=Func:Loop:  
n+1>n:if isPrime(n):return n:  
EndLoop:EndFunc ENTER Done
```

nextPrim(7) **ENTER** 11

isVar() CATALOG

isVar(*var_name*) ⇒ *true,false*

isArchiv(PROG1) **ENTER** True

Determines if *var_name* is in use. Returns true if *var_name* exists. Returns false if *var_name* does not exist.

Item CATALOG

Item *itemNameString*

See **Custom** example.

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.

Lbl CATALOG

Lbl *labelName*

Program segment:

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.

```

:
:Lbl lbl1
:InputStr "Enter password". str1
:If str1#password
: Goto lbl1
:Disp "Welcome to ..."
:

```

lcm() MATH/Number menu

lcm(*number1, number2*) ⇒ *expression*

lcm(6,9) **ENTER** 18

lcm(*list1, list2*) ⇒ *list*

lcm(*matrix1, matrix2*) ⇒ *matrix*

lcm({1/3, -14,16},{2/15,7.5}) **ENTER**
{2/3 14 80}

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.

left() MATH/String menu

left(*sourceString, num*) ⇒ *string*

left("Hello",2) **ENTER** "He"

Returns the leftmost *num* characters contained in character string *sourceString*.

If you omit *num*, returns all of *sourceString*.

left(*list1, num*) ⇒ *list*

left({1,3, -2,4},3) **ENTER** {1 3 -2}

Returns the leftmost *num* elements contained in *list1*.

If you omit *num*, returns all of *list1*.

left(*comparison*) ⇒ *expression*

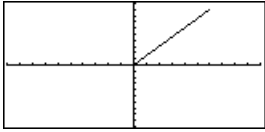
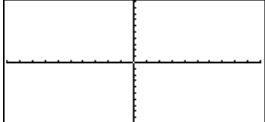
left(x<3) **ENTER** x

Returns the left-hand side of an equation or inequality.

limit() MATH/Calculus menu

limit (<i>expression1</i> , <i>var</i> , <i>pointA</i> , <i>direction</i>) ⇒ <i>expression</i>	<code>limit(2x+3,x,5)</code> <code>ENTER</code>	13
limit (<i>list1</i> , <i>var</i> , <i>pointA</i> , <i>direction</i>) ⇒ <i>list</i>	<code>limit(1/x,x,0.1)</code> <code>ENTER</code>	∞
limit (<i>matrix1</i> , <i>var</i> , <i>pointA</i> , <i>direction</i>) ⇒ <i>matrix</i>	<code>limit(sin(x)/x,x,0)</code> <code>ENTER</code>	1
Returns the limit requested.	<code>limit((sin(x+h)-sin(x))/h,h,0)</code> <code>ENTER</code>	$\cos(x)$
<i>direction</i> : negative=from left, positive=from right, otherwise=both. (If omitted, <i>direction</i> defaults to both.)	<code>limit((1+1/n)^n,n,∞)</code> <code>ENTER</code>	<i>e</i>
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() uses methods such as L'Hopital's rule, so there are unique limits that it cannot determine. If <i>expression1</i> contains undefined variables other than <i>var</i> , you might have to constrain them to obtain a more concise result.	<code>limit(a^x,x,∞)</code> <code>ENTER</code>	undef
	<code>limit(a^x,x,∞) a>1</code> <code>ENTER</code>	∞
	<code>limit(a^x,x,∞) a>0 and a<1</code> <code>ENTER</code>	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 <i>xStart</i> , <i>yStart</i> , <i>xEnd</i> , <i>yEnd</i> , <i>drawMode</i>	In the ZoomStd window, draw a line and then erase it.
Displays the Graph screen and draws, erases, or inverts a line segment between the window coordinates (<i>xStart</i> , <i>yStart</i>) and (<i>xEnd</i> , <i>yEnd</i>), including both endpoints.	<code>Line 0.0,6.9</code> <code>ENTER</code>
If <i>drawMode</i> = 1, draws the line (default).	
If <i>drawMode</i> = 0, turns off the line.	<code>[♦] [CALC HOME]</code>
If <i>drawMode</i> = -1, turns a line that is on or off or off to on (inverts pixels along the line).	<code>Line 0.0,6.9,0</code> <code>ENTER</code>
Note: Regrapping erases all drawn items. See also PxlLine .	

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).

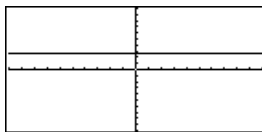
If $drawMode = 0$, turns off the line.

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**.

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.

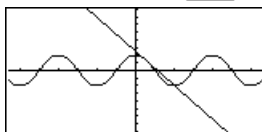
Note: In the example shown, *expression1* is graphed separately. **LineTan** does not graph *expression1*.

In function graphing mode and a ZoomTrig window:

Graph $\cos(x)$

\blacktriangledown [CALC HOME]

LineTan $\cos(x), \pi/4$ [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).

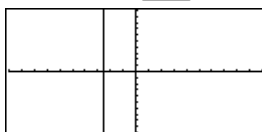
If *drawMode* = 0, turns off the line.

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**.

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 x list.

list2 represents y list.

list3 represents frequency.

list4 represents category codes.

list5 represents category include list.

Note: *list1* through *list4* must be a variable name or $c1$ – $c9$ (columns in the last data variable shown in the Data/Matrix Editor). *list5* does not have to be a variable name and cannot be $c1$ – $c9$.

In function graphing mode:

$\{0,1,2,3,4,5,6\} \rightarrow L1$ [ENTER]

$\{0,1,2, \dots\}$

$\{0,2,3,4,3,4,6\} \rightarrow L2$ [ENTER]

$\{0,2,3, \dots\}$

LinReg L1,L2 [ENTER]

Done

ShowStat [ENTER]



[ENTER]

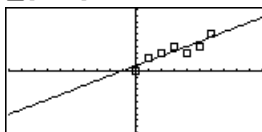
Regeq(x) \rightarrow $y1(x)$ [ENTER]

Done

NewPlot 1,1,L1,L2 [ENTER]

Done

\blacktriangledown [GRAPH]



Δlist() MATH/List menu

list(*list1*) ⇒ *list*

Δ list({20.30,45.70}) **ENTER**

{10,15.25}

Returns a list containing the differences between consecutive elements in *list1*. Each element of *list1* is subtracted from the next element of *list1*. The resulting list is always one element shorter than the original *list1*.

listmat() MATH/List menu

listmat(*list* [, *elementsPerRow*]) ⇒ *matrix*

listmat({1,2,3}) **ENTER**

[1 2 3]

Returns a matrix filled row-by-row with the elements from *list*.

listmat({1,2,3,4,5},2) **ENTER**

1	2
3	4
5	0

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.

ln MATH/String menu

ln *expression* ⇒ *expression*

Log(x) **ln** **ENTER**

Causes the input expression to be converted to an expression containing only natural logs (ln).

$$\frac{\ln(x)}{\ln(10)}$$

ln() **LN** key

ln(*expression1*) ⇒ *expression*

ln(2.0) **ENTER**

.693...

ln(*list1*) ⇒ *list*

Returns the natural logarithm of the argument.

If complex format mode is REAL:

ln({-3,1,2,5}) **ENTER**

Error: Non-real result

For a list, returns the natural logarithms of the elements.

If complex format mode is RECTANGULAR:

ln({-3,1,2,5}) **ENTER**

{ln(3) + π • i .182... ln(5)}

ln(*squareMatrix1*) ⇒ *squareMatrix*

Returns the matrix natural logarithm of *squareMatrix1*. This is *not* the same as calculating the natural logarithm of each element. For information about the calculation method, refer to **cos()** on.

In Radian angle mode and Rectangular complex format mode:

ln([1.5,3;4,2,1;6,-2,1]) **ENTER**

1.831...+1.734...• <i>i</i> ^{°°°°} .009...-1.490...• <i>i</i> ^{°°°°} ...
.448...- .725...• <i>i</i> ^{°°°°} 1.064...+ .623...• <i>i</i> ^{°°°°} ...
-.266...-2.083...• <i>i</i> ^{°°} 1.124...+1.790...• <i>i</i> ^{°°} ...

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

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

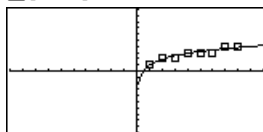
Regeq(x) → y1(x) **ENTER**

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.

{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 (<i>expression1</i> , <i>expression2</i>) \Rightarrow <i>expression</i>	$\log(2.0)$ [ENTER] .301...
log (<i>list1</i> , <i>expression2</i>) \Rightarrow <i>list</i>	If complex format mode is REAL: $\log(\{-3.1.2.5\})$ [ENTER] Error: Non-real result
Returns the base- <i>expression2</i> logarithm of the argument.	If complex format mode is RECTANGULAR: $\log(\{-3.1.2.5\})$ [ENTER] $\left\{ \frac{\ln(3)}{\ln(10)} + \frac{\pi}{\ln(10)} \cdot i \ .079... \frac{\ln(5)}{\ln(10)} \right\}$
For a list, returns the base- <i>expression2</i> logarithm of the elements.	
If <i>expression 2</i> is omitted, 10 is used.	

log (<i>squareMatrix1</i>) \Rightarrow <i>squareMatrix</i>	In Radian angle mode and Rectangular complex format mode: $\log([1.5.3;4.2.1;6. -2.1])$ [ENTER] $\left[\begin{array}{ccc} .795...+.753... \cdot i & .003...-.647... \cdot i & \dots \\ .194...-.315... \cdot i & .462...+.270 \cdot i & \dots \\ -.115...-.904... \cdot i & .488...+.777... \cdot i & \dots \end{array} \right]$
Returns the matrix base- <i>expression2</i> logarithm of <i>squareMatrix1</i> . This is <i>not</i> the same as calculating the base- <i>expression2</i> logarithm of each element. For information about the calculation method, refer to cos() .	
<i>squareMatrix1</i> must be diagonalizable. The result always contains floating-point numbers.	

log (<i>x</i> , <i>b</i>) \Rightarrow <i>expression</i>	$\text{Log}(10,3) - \text{log}(5,3)$ [ENTER] $\text{Log}_5(2)$
log (<i>squareMatrix1</i>) \Rightarrow <i>squareMatrix</i>	$\text{Log}(2.0,4)$ [ENTER] .5
In a list, returns the base- <i>expression2</i> logarithm of the elements.	

►logbase MATH/String menu

<i>expression</i> ► logbase (<i>expression1</i>) \Rightarrow <i>expression</i>	$\text{Log}(10,3) - \text{log}(5,5)$ ► logbase (5) [ENTER] $\frac{\log_5(30)}{\log_5(3)}$
Causes the input expression to be simplified to an expression using base <i>expression1</i> .	

Logistic MATH/Statistics/Regressions menu

Logistic *list1*, *list2* [, *iterations*], [*list3*] [, *list4*, *list5*]

Calculates the logistic regression and updates all the system statistics variables.

All the lists must have equal dimensions except for *list5*.

list1 represents *x*list.

list2 represents *y*list.

list3 represents frequency.

list4 represents category codes.

list5 represents category include list.

iterations specifies the maximum number of times a solution will be attempted. If omitted, 64 is used. Typically, larger values result in better accuracy but longer execution times, and vice versa.

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:

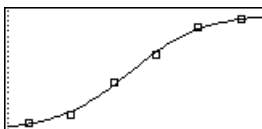
{1,2,3,4,5,6} → L1 [ENTER] {1 2 3 ...}
 {1,1,3,2,5,3,5,4,5,4,8} → L2 [ENTER]

Logistic L1,L2 [ENTER] {1 1.3 2.5 ...} Done
 ShowStat [ENTER]



[ENTER] regeq(x) → y1(x) [ENTER] Done
 NewPlot 1,1,1,L1,L2 [ENTER] Done

[GRAPH] [F2] 9



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
:Lb1 End
:Disp "The number of rolls is", i
:
:
    
```

LU MATH/Matrix menu

LU *matrix*, *lMatName*, *uMatName*, *pMatName*, *tol*

[6,12,18:5,14,31:3,8,18]►m1 **ENTER**

Calculates the Doolittle LU (lower-upper) decomposition of a real or complex *matrix*. The lower triangular matrix is stored in *lMatName*, the upper triangular matrix in *uMatName*, and the permutation matrix (which describes the row swaps done during the calculation) in *pMatName*.

$$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$$

LU m1, lower, upper, perm **ENTER** Done

lMatName * *uMatName* = *pMatName* * *matrix*

lower **ENTER**

$$\begin{bmatrix} 1 & 0 & 0 \\ 5/6 & 1 & 0 \\ 1/2 & 1/2 & 1 \end{bmatrix}$$

Optionally, any matrix element is treated as zero if its absolute value is less than *tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *tol* is ignored.

upper **ENTER**

$$\begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$$

- If you use **◻** **ENTER** or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.

perm **ENTER**

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- If *tol* is omitted or not used, the default tolerance is calculated as:

$$5e-14 * \max(\text{dim}(\text{matrix})) * \text{rowNorm}(\text{matrix})$$

[m,n:o,p]►m1 **ENTER**

$$\begin{bmatrix} m & n \\ o & p \end{bmatrix}$$

The **LU** factorization algorithm uses partial pivoting with row interchanges.

LU m1, lower, upper, perm **ENTER** Done

lower **ENTER**

$$\begin{bmatrix} 1 & 0 \\ m & 1 \\ o & 1 \end{bmatrix}$$

upper **ENTER**

$$\begin{bmatrix} o & p & \\ 0 & n - \frac{m \cdot p}{o} & \end{bmatrix}$$

perm **ENTER**

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

mat►data MATH/List menu

mat►data *mat,data*,*[row1]*,*[col1]*,*[row2]*,*[col2]*

mat►data,m1,d1,1,1,1 **ENTER**

Converts a matrix to data.

Done

Each argument *[row1]*,*[col1]*,*[row2]*,*[col2]* can be individually omitted. If *row1* is omitted the default is 1. If *col1* is omitted the default is 1. If *row2* is omitted, the default is "max row." If *col2* is omitted, the default is "max column."

mat▶list() MATH/List menu

mat▶list (<i>matrix</i>) ⇒ <i>list</i>	mat▶list([1.2.3]) ENTER	{1 2 3}
Returns a list filled with the elements in <i>matrix</i> . The elements are copied from <i>matrix</i> row by row.	[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 (<i>expression1</i> , <i>expression2</i>) ⇒ <i>expression</i>	max(2.3.1.4) ENTER	2.3
max (<i>list1</i> , <i>list2</i>) ⇒ <i>list</i>		
max (<i>matrix1</i> , <i>matrix2</i>) ⇒ <i>matrix</i>	max({1.2},{-4.3}) ENTER	{1 3}

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 (<i>list</i>) ⇒ <i>expression</i>	max({0.1,-7.1.3..5}) ENTER	1.3
--	-----------------------------------	-----

Returns the maximum element in *list*.

max (<i>matrix1</i>) ⇒ <i>matrix</i>	max([1.-3.7;-4.0..3]) ENTER	[1 0 7]
---	------------------------------------	---------

Returns a row vector containing the maximum element of each column in *matrix1*.

Note: See also **fMax()** and **min()**.

mean() MATH/Statistics menu

mean (<i>list</i> , <i>freqlist</i>) ⇒ <i>expression</i>	mean({.2.0.1,-.3..4}) ENTER	.26
---	------------------------------------	-----

Returns the mean of the elements in *list*.

Each <i>freqlist</i> element counts the number of consecutive occurrences of the corresponding element in <i>list</i> .	mean({1.2.3},{3.2.1}) ENTER	5/3
---	------------------------------------	-----

mean (<i>matrix1</i> , <i>freqmatrix</i>) ⇒ <i>matrix</i>	In vector format rectangular mode:	
--	------------------------------------	--

Returns a row vector of the means of all the columns in <i>matrix1</i> .	mean([.2.0;-1.3;.4,-.5]) ENTER	[-.133... .833...]
--	---------------------------------------	--------------------

Each <i>freqmatrix</i> element counts the number of consecutive occurrences of the corresponding element in <i>matrix1</i> .	mean([1/5.0;-1.3;2/5,-1/2]) ENTER	[-2/15 5/6]
--	--	-------------

	mean([1.2;3.4;5.6],[5.3;4.1;6.2]) ENTER	[47/15, 11/3]
--	--	---------------

median() MATH/Statistics menu

median (<i>list</i>) ⇒ <i>expression</i>	median({.2.0.1,-.3..4}) ENTER	.2
---	--------------------------------------	----

Returns the median of the elements in *list*.

median (<i>matrix1</i>) ⇒ <i>matrix</i>	median([.2.0;1,-.3;.4,-.5]) ENTER	[.4 -.3]
--	--	----------

Returns a row vector containing the medians of the columns in *matrix1*.

Note: All entries in the list or matrix must simplify to numbers.

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.

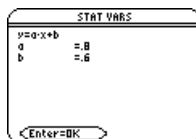
list4 represents category codes.

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.

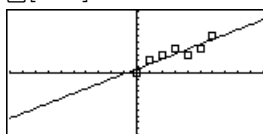
In function graphing mode:

```
{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*, *startA*, *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]
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*, *startA*, [*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 (<i>expression1</i> , <i>expression2</i>) ⇒ <i>expression</i>	<code>min(2.3,1.4)</code> [ENTER]	1.4
min (<i>list1</i> , <i>list2</i>) ⇒ <i>list</i>		
min (<i>matrix1</i> , <i>matrix2</i>) ⇒ <i>matrix</i>	<code>min({1.2},{-4.3})</code> [ENTER]	{ -4 2 }
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 (<i>list</i>) ⇒ <i>expression</i>	<code>min({0.1,-7.1.3..5})</code> [ENTER]	-7
Returns the minimum element of <i>list</i> .		
min (<i>matrix1</i>) ⇒ <i>matrix</i>	<code>min([1,-3.7;-4.0..3])</code> [ENTER]	$\begin{bmatrix} -4 & -3 & .3 \end{bmatrix}$
Returns a row vector containing the minimum element of each column in <i>matrix1</i> .		
Note: See also fMin() and max() .		

mod() MATH/Number menu		
mod (<i>expression1</i> , <i>expression2</i>) ⇒ <i>expression</i>	<code>mod(7.0)</code> [ENTER]	7
mod (<i>list1</i> , <i>list2</i>) ⇒ <i>list</i>		
mod (<i>matrix1</i> , <i>matrix2</i>) ⇒ <i>matrix</i>	<code>mod(7.3)</code> [ENTER]	1
Returns the first argument modulo the second argument as defined by the identities:		
$\text{mod}(x,0) = x$	<code>mod(-7.3)</code> [ENTER]	2
$\text{mod}(x,y) = x - y \text{ floor}(x/y)$	<code>mod(7,-3)</code> [ENTER]	-2
	<code>mod(-7,-3)</code> [ENTER]	-1
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.		
	<code>mod({12,-14.16},{9.7,-5})</code> [ENTER]	$\begin{bmatrix} 3 & 0 & -4 \end{bmatrix}$
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() .		

MoveVar CATALOG		
MoveVar <i>var</i> , <i>oldFolder</i> , <i>newFolder</i>	<code>{1.2.3.4}→L1</code> [ENTER]	{1 2 3 4}
Moves variable <i>var</i> from <i>oldFolder</i> to <i>newFolder</i> . If <i>newFolder</i> does not exist, MoveVar creates it.	<code>MoveVar L1,Matn,Games</code> [ENTER]	Done

mRow() MATH/Matrix/Row ops menu		
mRow (<i>expression</i> , <i>matrix1</i> , <i>index</i>) ⇒ <i>matrix</i>	<code>mRow(-1/3,[1.2:3.4],2)</code> [ENTER]	$\begin{bmatrix} 1 & 2 \\ -1 & -4/3 \end{bmatrix}$
Returns a copy of <i>matrix1</i> with each element in row <i>index</i> of <i>matrix1</i> multiplied by <i>expression</i> .		

mRowAdd() MATH/Matrix/Row ops menu		
mRowAdd (<i>expression</i> , <i>matrix1</i> , <i>index1</i> , <i>index2</i>) ⇒ <i>matrix</i>	<code>mRowAdd(-3,[1.2:3.4],1,2)</code> [ENTER]	$\begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}$
Returns a copy of <i>matrix1</i> with each element in row <i>index2</i> of <i>matrix1</i> replaced with:		
$\text{expression} \times \text{row } \textit{index1} + \text{row } \textit{index2}$	<code>mRowAdd(n,[a,b:c,d],1,2)</code> [ENTER]	$\begin{bmatrix} a & b \\ a \cdot n + c & b \cdot n + d \end{bmatrix}$

nCr() MATH/Probability menu

nCr(*expression1*, *expression2*) ⇒ *expression*

For integer *expression1* and *expression2* with $expression1 \geq expression2 \geq 0$, **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.

$$nCr(z, 3) \quad \frac{z \cdot (z-2) \cdot (z-1)}{6}$$

$$ans(1)|z=5 \quad 10$$

$$nCr(z, c) \quad \frac{z!}{c!(z-c)!}$$

nCr(*expression*, 0) ⇒ 1

$$ans(1)/nPr(z, c) \quad \frac{1}{c!}$$

nCr(*expression*, *negInteger*) ⇒ 0

nCr(*expression*, *posInteger*) ⇒ $expression \cdot (expression-1) \dots (expression-posInteger+1) / posInteger!$

nCr(*expression*, *nonInteger*) ⇒ $expression! / ((expression-nonInteger)! \cdot nonInteger!)$

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.

$$nCr(\{5,4,3\}, \{2,4,2\}) \text{ [ENTER]} \\ \{10 \ 1 \ 3\}$$

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.

$$nCr([6,5;4,3], [2,2;2,2]) \text{ [ENTER]} \\ \begin{bmatrix} 15 & 10 \\ 6 & 3 \end{bmatrix}$$

nDeriv() MATH/Calculus menu

nDeriv(*expression1*, *var**h*, *h*) ⇒ *expression*

nDeriv(*expression1*, *var*, *list*) ⇒ *list*

nDeriv(*list*, *var**h*, *h*) ⇒ *list*

nDeriv(*matrix*, *var**h*, *h*) ⇒ *matrix*

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.

When using *list* or *matrix*, the operation gets mapped across the values in the list or across the matrix elements.

Note: See also **avgRC**() and **d**()

$$nDeriv(\cos(x), x, h) \text{ [ENTER]}$$

$$\frac{-\cos(x-h) - \cos(x+h)}{2 \cdot h}$$

$$\text{limit}(nDeriv(\cos(x), x, h), h, 0) \text{ [ENTER]} \\ -\sin(x)$$

$$nDeriv(x^3, x, 0.01) \text{ [ENTER]} \\ 3 \cdot (x^2 + .000033)$$

$$nDeriv(\cos(x), x) | x=\pi/2 \text{ [ENTER]} \\ -1.$$

$$nDeriv(x^2, x, \{.01, .1\}) \text{ [ENTER]} \\ \{2 \cdot x \ 2 \cdot x\}$$

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.

$$\text{NewData mydata}.\{1,2,3\}.\{4,5,6\} \text{ [ENTER]} \\ \text{Done}$$

(Go to the Data/Matrix Editor and open the *var* mydata to display the data variable below.)

DATA	c1	c2	c3
1	1	4	
2	2	5	
3	3	6	
4			

NewData *dataVar*, *matrix*

Creates data variable *dataVar* based on *matrix*.

NewData sysData, matrix

Loads the contents of *matrix* into the system data variable *sysData*.

NewFold CATALOG**NewFold** *folderName*NewFold games

Done

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.

newList() CATALOG**newList**(*numElements*) ⇒ *list*newList(4)

{0 0 0 0}

Returns a list with a dimension of *numElements*. Each element is zero.

newMat() CATALOG also Math/Matrix menu**newMat**(*numRows*, *numColumns*) ⇒ *matrix*newMat(2,3) $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

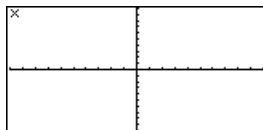
Returns a matrix of zeros with the dimension *numRows* by *numColumns*.

NewPic CATALOG**NewPic** *matrix*, *picVar* [, *maxRow*] [, *maxCol*]NewPic [1,1;2,2;3,3;4,4;5,5;
5,1;4,2;2,4;1,5].xpPic

Done

Creates a pic variable *picVar* based on *matrix*. *matrix* must be an $n \times 2$ 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*.



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
- 5 = modified box plot

mark specifies the display type of the mark.

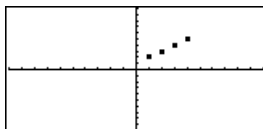
- 1 = □ (box)
- 2 = × (cross)
- 3 = + (plus)
- 4 = ■ (square)
- 5 = • (dot)

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 [2nd] [GRAPH] to display:



NewProb CATALOG

NewProb

NewProb [ENTER] Done

Performs a variety of operations that let you begin a new problem from a cleared state without resetting the memory.

- Clears all single-character variable names (Clear a–z) in the current folder, unless the variables are locked or archived.
- Turns off all functions and stat plots (**FnOff** and **PlotsOff**) in the current graphing mode.
- Performs **ClrDraw**, **ClrErr**, **ClrGraph**, **ClrHome**, **ClrIO**, and **ClrTable**.

nInt() MATH/Calculus menu

nInt(*expression1*, *var*, *lower*, *upper*) ⇒ *expression*

nInt($e^{-x^2} \cdot x \cdot -1.1$) [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 $\int(\textit{expression1}, \textit{var}, \textit{lower}, \textit{upper})$. This approximation is a weighted average of some sample values of the integrand in the interval $\textit{lower} < \textit{var} < \textit{upper}$.

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.

A warning is displayed ("Questionable accuracy") when it seems that the goal has not been achieved.

nInt(cos(x) · x · -π · π + 1 E - 12) [ENTER] -1.041... E - 12
 $\int(\cos(x) \cdot x \cdot -\pi \cdot \pi + 10^{(-12)})$ [ENTER]
 $-\sin\left(\frac{1}{1000000000000}\right)$
 ans(1) [2nd] [ENTER] -1. E - 12

Nest **nInt()** to do multiple numeric integration. Integration limits can depend on integration variables outside them.

$nInt(nInt(e^{-(x*y)} / \sqrt{x^2 - y^2}, y, -x, x), x, 0, 1)$ **ENTER** 3.304...

Note: See also **j()**.

norm() MATH/Matrix/Norms menu

norm(*matrix*) \Rightarrow *expression*

Returns the Frobenius norm.

$norm([a, b, c, d])$ **ENTER**

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

$norm([1, 2, 3, 4])$ **ENTER**

$$\sqrt{30}$$

not MATH/Test menu

not *Boolean expression1* \Rightarrow *Boolean expression*

Returns true, false, or a simplified *Boolean expression1*.

$not\ 2 \geq 3$ **ENTER**

true

$not\ x < 2$ **ENTER**

$x \geq 2$

$not\ not\ innocent$ **ENTER**

innocent

not *integer1* \Rightarrow *integer*

Returns the one's complement of a real integer. Internally, *integer1* is converted to a signed, 32-bit binary number. The value of each bit is flipped (0 becomes 1, and vice versa) for the one's complement. Results are displayed according to the Base mode.

You can enter the integer in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, the integer is treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

In Hex base mode:

$not\ 0h7AC36$ **ENTER**

0hFFF85C9

↳ **Important:** Zero, not the letter O.

In Bin base mode:

$0b100101$ ▶ dec **ENTER**

37

$not\ 0b100101$ **ENTER**

$0b11111111111111111111111111111111011010$

$ans(1)$ ▶ dec **ENTER**

- 38

Note: A binary entry can have up to 32 digits (not counting the 0b prefix). A hexadecimal entry can have up to 8 digits.

Note: To type the ▶ conversion operator, press **2nd** **[▶]**. You can also select base conversions from the MATH/Base menu.

nPr() MATH/Probability menu

nPr(*expression1*, *expression2*) \Rightarrow *expression*

For integer *expression1* and *expression2* with $expression1 \geq expression2 \geq 0$, **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) \Rightarrow 1

nPr(*expression*, *negInteger*) \Rightarrow
 $1 / ((expression + 1) \cdot (expression + 2) \cdot \dots \cdot (expression - negInteger))$

nPr(*expression*, *posInteger*) \Rightarrow
 $expression \cdot (expression - 1) \cdot \dots \cdot (expression - posInteger + 1)$

nPr(*expression*, *nonInteger*) \Rightarrow *expression!*
(expression - nonInteger)!

nPr(*list1*, *list2*) \Rightarrow *list*

Returns a list of permutations based on the corresponding element pairs in the two lists. The arguments must be the same size list.

$nPr(z, 3)$ **ENTER**

$z \cdot (z - 2) \cdot (z - 1)$

$ans(1) | z = 5$ **ENTER**

60

$nPr(z, -3)$ **ENTER**

$$\frac{1}{(z+1) \cdot (z+2) \cdot (z+3)}$$

$nPr(z, c)$ **ENTER**

$$\frac{z!}{(z - c)!}$$

$ans(1) * nPr(z - c, -c)$ **ENTER**

1

$nPr(\{5, 4, 3\}, \{2, 4, 2\})$ **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*, *varOrGuess*) ⇒ *number or error_string*

Iteratively searches for one approximate real numeric solution to *equation* for its one variable. Specify *varOrGuess* as:

variable

– or –

variable = *real number*

For example, *x* is valid and so is *x=3*.

nSolve() is often much faster than **solve()** or **zeros()**, particularly if the "|" operator is used to constrain the search to a small interval containing 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."

If you use **nSolve()** in a program, you can use **getType()** to check for a numeric result before using it in an algebraic expression.

Note: See also **cSolve()**, **cZeros()**, **solve()**, and **zeros()**.

nSolve(x^2+5x-25=9,x) **ENTER**

3.844...

nSolve(x^2=4,x=-1) **ENTER**

-2.

nSolve(x^2=4,x=1) **ENTER**

2.

Note: If there are multiple solutions, you can use a guess to help find a particular solution.

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**

STAT VARS	
\bar{x}	=3.142857
Σx	=25
Σx^2	=90
<i>Sx</i>	=1.864454
<i>nStat</i>	=7
<i>minR</i>	=0
<i>q1</i>	=2
<i>medStat</i>	=3
ENTER=OK	

or **MATH/Test menu**

Boolean expression1 **or** *Boolean expression2* ⇒ *Boolean expression* $x \geq 3$ or $x \geq 4$ [ENTER] $x \geq 3$

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**.

Program segment:
 :
 :
 If $x < 0$ or $x \geq 5$
 Goto END
 :
 :
 If choice=1 or choice=2
 Disp "Wrong choice"
 :

integer1 **or** *integer2* ⇒ *integer*

Compares two real integers bit-by-bit using an **or** operation. Internally, both integers are converted to signed, 32-bit binary numbers. When corresponding bits are compared, the result is 1 if either bit is 1; the result is 0 only if both bits are 0. The returned value represents the bit results, and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

Note: See **xor**.

In Hex base mode:
 0h7AC36 or 0h3D5F [ENTER] 0h7BD7F
 └─ **Important:** Zero, not the letter O.

In Bin base mode:
 0b100101 or 0b100 [ENTER] 0b100101

Note: A binary entry can have up to 32 digits (not counting the 0b prefix). A hexadecimal entry can have up to 8 digits.

ord() **MATH/String menu**

ord(*string*) ⇒ *integer* ord("hello") [ENTER] 104
ord(*list1*) ⇒ *list* char(104) [ENTER] "h"
 Returns the numeric code of the first character in character string *string*, or a list of the first characters of each list element. ord(char(24)) [ENTER] 24
 See Appendix B for a complete listing of character codes. 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*).

An expression can include conversion operations such as **►DD** and **►Rect**. You can also use the **►** operator to perform unit and number base conversions.

If Pretty Print = ON, *exprOrString* is "pretty printed."

From the Program I/O screen, you can press [F5] to display the Home screen, or a program can use **DispHome**.

Program segment:
 :
 :
 :RandSeed 1147
 :ClrIO
 :For i,1,90,10
 : Output i, rand(100),"Hello"
 :EndFor
 :

Result after execution:

```

      Hello
Hello
Hello  Hello
      Hello
Hello  Hello
      Hello
      Hello
  
```

P>Rx() MATH/Angle menu**P>Rx**(*rExpression*, *θExpression*) ⇒ *expression***P>Rx**(*rList*, *θList*) ⇒ *list***P>Rx**(*rMatrix*, *θMatrix*) ⇒ *matrix*Returns the equivalent x-coordinate of the (*r*, *θ*) pair.**Note:** The *θ* argument is interpreted as either a degree, gradian or radian angle, according to the current angle mode. If the argument is an expression, you can use °, ° or ° to override the angle mode setting temporarily.

In Radian angle mode:

P>Rx(*r*, *θ*) [ENTER] $\cos(\theta) \cdot r$ **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*, *θExpression*) ⇒ *expression***P>Ry**(*rList*, *θList*) ⇒ *list***P>Ry**(*rMatrix*, *θMatrix*) ⇒ *matrix*Returns the equivalent y-coordinate of the (*r*, *θ*) pair.**Note:** The *θ* argument is interpreted as either a degree, gradian or radian angle, according to the current angle mode. If the argument is an expression, you can use °, ° or ° to override the angle mode setting temporarily.

In Radian angle mode:

P>Ry(*r*, *θ*) [ENTER] $\sin(\theta) \cdot r$ **P>Ry**(4, 60°) [ENTER] $2 \cdot \sqrt{3}$ **P>Ry**({-3, 10, 1.3}, {π/3, -π/4, 0}) [ENTER]
{ $\frac{-3 \cdot \sqrt{3}}{2}$ -5·√2 0. }**part()** CATALOG**part**(*expression1*, *nonNegativeInteger*)This advanced programming function lets you identify and extract all of the sub-expressions in the simplified result of *expression1*.For example, if *expression1* simplifies to $\cos(\pi * x + 3)$:

- The **cos()** function has one argument: $(\pi * x + 3)$.
- The sum of $(\pi * x + 3)$ has two operands: $\pi * x$ and 3.
- The number 3 has no arguments or operands.
- The product $\pi * x$ has two operands: π and x .
- The variable x and the symbolic constant π have no arguments or operands.

If x has a numeric value and you press [] [ENTER], the numeric value of $\pi * x$ is calculated, the result is added to 3, and then the cosine is calculated.**cos()** is the **top-level** operator because it is applied **last**.**part**(*expression1*) ⇒ *number*Simplifies *expression1* and returns the number of top-level arguments or operands. This returns 0 if *expression1* is a number, variable, or symbolic constant such as π , e , i , or ∞ .**part**($\cos(\pi * x + 3)$) [ENTER] 1**Note:** $\cos(\pi * x + 3)$ has one argument.**part**(*expression1*, 0) ⇒ *string*Simplifies *expression1* and returns a string that contains the top-level function name or operator. This returns **string**(*expression1*) if *expression1* is a number, variable, or symbolic constant such as π , e , i , or ∞ .**part**($\cos(\pi * x + 3)$, 0) [ENTER] "cos"

part(expression1, n) ⇒ *expression*

Simplifies *expression1* and returns the *n*th argument or operand, where *n* is > 0 and ≤ the number of top-level arguments or operands returned by **part(expression1)**. Otherwise, an error is returned.

By combining the variations of **part()**, you can extract all of the sub-expressions in the simplified result of *expression1*. As shown in the example to the right, you can store an argument or operand and then use **part()** to extract further sub-expressions.

Note: When using **part()**, do not rely on any particular order in sums and products.

Expressions such as $(x+y+z)$ and $(x-y-z)$ are represented internally as $(x+y)+z$ and $(x-y)-z$. This affects the values returned for the first and second argument. There are technical reasons why **part(x+y+z,1)** returns $y+x$ instead of $x+y$.

Similarly, $x*y*z$ is represented internally as $(x*y)*z$. Again, there are technical reasons why the first argument is returned as $y*x$ instead of $x*y$.

When you extract sub-expressions from a matrix, remember that matrices are stored as lists of lists, as illustrated in the example to the right.

part(cos($\pi*x+3$),1) [ENTER]

$3+\pi*x$

Note: Simplification changed the order of the argument.

part(cos($\pi*x+3$)) [ENTER]

1

part(cos($\pi*x+3$),0) [ENTER]

"cos"

part(cos($\pi*x+3$),1)⇒ temp [ENTER]

$3+\pi*x$

temp [ENTER]

$\pi*x+3$

part(temp,0) [ENTER]

"+"

part(temp) [ENTER]

2

part(temp,2) [ENTER]

3

part(temp,1)⇒ temp [ENTER]

$\pi*x$

part(temp,0) [ENTER]

"*"

part(temp) [ENTER]

2

part(temp,1) [ENTER]

π

part(temp,2) [ENTER]

x

part(x+y+z) [ENTER]

2

part(x+y+z,2) [ENTER]

z

part(x+y+z,1) [ENTER]

$y+x$

part(x*y*z) [ENTER]

2

part(x*y*z,2) [ENTER]

z

part(x*y*z,1) [ENTER]

$y*x$

part([a,b,c;x,y,z],0) [ENTER]

"{"

part([a,b,c;x,y,z]) [ENTER]

2

part([a,b,c;x,y,z],2)⇒ temp [ENTER]

{x y z}

part(temp,0) [ENTER]

"{"

part(temp) [ENTER]

3

part(temp,3) [ENTER]

z

delVar temp [ENTER]

Done

The example Program Editor function to the right uses **getType()** and **part()** to partially implement symbolic differentiation. Studying and completing this function can help teach you how to differentiate manually. You could even include functions that the cannot differentiate, such as Bessel functions.

```

:d(y,x)
:Func
:Local f
:If getType(y)="VAR"
: Return when(y=x,1,0,0)
:If part(y)=0
: Return 0 ● y=π,∞,i,numbers
:part(y,0)➤f
:If f="-" ● if negate
: Return -d(part(y,1),x)
:If f="-" ● if minus
: Return d(part(y,1),x)
-d(part(y,2),x)
:If f="+"
: Return d(part(y,1),x)
+d(part(y,2),x)
:If f="*"
: Return part(y,1)*d(part(y,2),x)
+part(y,2)*d(part(y,1),x)
:If f="{"
: Return seq(d(part(y,k),x),
k,1,part(y))
:Return undef
:EndFunc

```

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**.)

See **ClrErr** program listing example.

Pause CATALOG

Pause [expression]

Suspends program execution. If you include *expression*, displays *expression* on the Program I/O screen.

expression can include conversion operations such as **DD** and **Rect**. You can also use the **►** operator to perform unit and number base conversions.

If the result of *expression* is too big to fit on a single screen, you can use the cursor pad to scroll the display.

Program execution resumes when you press **ENTER**.

Program segment:

```

:
:ClrIO
:DelVar temp
:l→temp[1]
:l→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
:

```

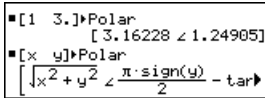
PlotsOff CATALOG

PlotsOff [1] [, 2] [, 3] ... [, 9]	PlotsOff 1.2.5 <input type="text" value="ENTER"/>	Done
Turns off the specified plots for graphing. When in 2-graph mode, only affects the active graph.	PlotsOff <input type="text" value="ENTER"/>	Done
If no parameters, then turns off all plots.		

PlotsOn CATALOG

PlotsOn [1] [, 2] [, 3] ... [, 9]	PlotsOn 2.4.5 <input type="text" value="ENTER"/>	Done
Turns on the specified plots for graphing. When in 2-graph mode, only affects the active graph.	PlotsOn <input type="text" value="ENTER"/>	Done
If you do not include any arguments, turns on all plots.		

►Polar MATH/Matrix/Vector ops menu

<i>vector</i> ►Polar	[1,3.] ►Polar <input type="text" value="ENTER"/>
Displays <i>vector</i> in polar form $[r \angle \theta]$. The vector must be of dimension 2 and can be a row or a column.	[x,y] ►Polar <input type="text" value="ENTER"/>
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.	

complexValue ►Polar	In Radian angle mode:
Displays <i>complexValue</i> in polar form.	$3+4i$ ►Polar <input type="text" value="ENTER"/> $e^{i \cdot (\frac{\pi}{3} - \tan^{-1}(3/4))} \cdot 5$
<ul style="list-style-type: none">Degree angle mode returns $(r \angle \theta)$.Radian angle mode returns $re^{i\theta}$.	$(4 \angle \pi/3)$ ►Polar <input type="text" value="ENTER"/> $e^{i \cdot \frac{\pi}{3}} \cdot 4$
<i>complexValue</i> can have any complex form. However, an $re^{i\theta}$ entry causes an error in Degree angle mode.	In Gradian angle mode:
Note: You must use the parentheses for an $(r \angle \theta)$ polar entry.	4 ►Polar <input type="text" value="ENTER"/> $(4 \angle 100)$
	In Degree angle mode:
	$3+4i$ ►Polar <input type="text" value="ENTER"/> $(5 \angle 90 - \tan^{-1}(3/4))$

polyEval() MATH/List menu

polyEval (list1, expression1) \Rightarrow expression	polyEval({a,b,c}.x) <input type="text" value="ENTER"/>
polyEval (list1, list2) \Rightarrow expression	$a \cdot x^2 + b \cdot x + c$
Interprets the first argument as the coefficient of a descending-degree polynomial, and returns the polynomial evaluated for the value of the second argument.	polyEval({1,2,3,4},2) <input type="text" value="ENTER"/> 26
	polyEval({1,2,3,4},{2,-7}) <input type="text" value="ENTER"/> {26 -262}

PopUp CATALOG

PopUp *itemList, var*

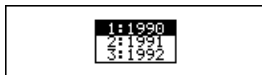
Displays a pop-up menu containing the character strings from *itemList*, waits for you to select an item, and stores the number of your selection in *var*.

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 *x*list.

list2 represents *y*list.

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 ...}

Done

PowerReg L1,L2 **[ENTER]**

ShowStat **[ENTER]**



[ENTER]

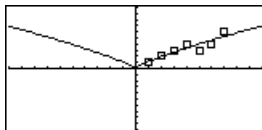
Regeq(x)→y1(x) **[ENTER]**

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:

:prgname()

:Prgm

:

:EndPrgm

Product (PI) See Π(), page 908.

product() MATH/List menu

product(*list*, [*start*], [*end*]) ⇒ *expression*

product({1,2,3,4}) **[ENTER]** 24

Returns the product of the elements contained in *list*. *Start* and *end* are optional. They specify a range of elements.

product({2,x,y}) **[ENTER]** 2 • x • y

product({4,5,8,9},2,3) **[ENTER]** 40

product(*matrix1*[, *start1*, *end1*]) \Rightarrow *matrix*

Returns a row vector containing the products of the elements in the columns of *matrix1*. *Start* and *end* are optional. They specify a range of rows.

product([1.2,3;4.5,6;7.8,9]) **[ENTER]** [28 ;

product([1.2,3;4.5,6;7.8,9],
1,2) **[ENTER]** [4,10,18]

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*[, *var1*]) \Rightarrow *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()**.

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]**

$$\blacksquare \text{propFrac}\left\{\frac{x^2+x+1}{x+1} + \frac{y^2+y+1}{y+1}\right\}$$
$$\frac{1}{x+1} + x + \frac{y^2+y+1}{y+1}$$

propFrac(ans(1))

$$\blacksquare \text{propFrac}\left\{\frac{1}{x+1} + x + \frac{y^2+y}{y+1}\right\}$$
$$\frac{1}{x+1} + x + \frac{1}{y+1} + y$$

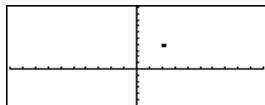
PtChg CATALOG

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]**

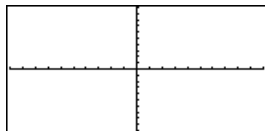


PtOff CATALOG

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]**

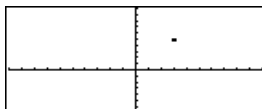


PtOn CATALOG

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 $(x, y) \Rightarrow$ Boolean constant expression
ptTest $(xList, yList) \Rightarrow$ 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

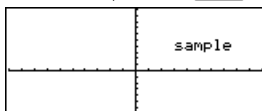
PtText CATALOG

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**



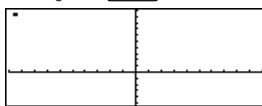
PxlChg CATALOG

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**



PxlCrc1 CATALOG

PxlCrc1 $row, col, r [, drawMode]$

Displays the Graph screen and draws a circle centered at pixel coordinates (row, col) with a radius of r pixels.

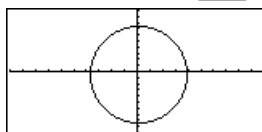
If $drawMode = 1$, draws the circle (default).

If $drawMode = 0$, turns off the circle.

If $drawMode = -1$, inverts pixels along the circle.

Note: Regraphing erases all drawn items. See also **Circle**.

PxlCrc1 50.125.40.1 **ENTER**



PxlHorz CATALOG

PxlHorz $row [, drawMode]$

Displays the Graph screen and draws a horizontal line at pixel position row .

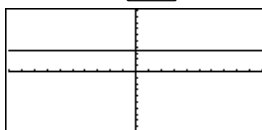
If $drawMode = 1$, draws the line (default).

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**.

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.

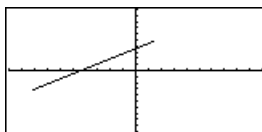
If *drawMode* = 1, draws the line (default).

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**.

PxlLine 80.20.30.150.1 **ENTER**



PxlOff CATALOG

PxlOff *row, col*

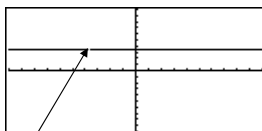
PxlOff *rowList, colList*

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**



25.50

PxlOn CATALOG

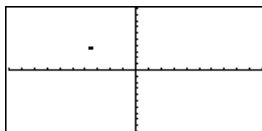
PxlOn *row, col*

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*) \Rightarrow *Boolean expression*

pxlTest (*rowList, colList*) \Rightarrow *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**

♦ [CALC HOME]

PxlTest(25.50) **ENTER**

true

PxlOff 25.50 **ENTER**

♦ [CALC HOME]

PxlTest(25.50) **ENTER**

false

PxlText CATALOG

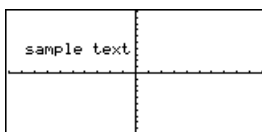
PxlText *string, row, col*

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*.

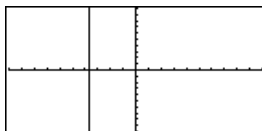
If *drawMode* = 1, draws the line (default).

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**.

PxlVert 50.1 **ENTER**



QR MATH/Matrix menu

QR *matrix*, *qMatName*, *rMatName* [, *tol*]

Calculates the Householder QR factorization of a real or complex *matrix*. The resulting Q and R matrices are stored to the specified *MatNames*. The Q matrix is unitary. The R matrix is upper triangular.

Optionally, any matrix element is treated as zero if its absolute value is less than *tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *tol* is ignored.

- If you use **□** **ENTER** or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.

- If *tol* is omitted or not used, the default tolerance is calculated as:

$$5E-14 * \max(\dim(\mathit{matrix})) * \mathit{rowNorm}(\mathit{matrix})$$

The QR factorization is computed numerically using Householder transformations. The symbolic solution is computed using Gram-Schmidt. The columns in *qMatName* are the orthonormal basis vectors that span the space defined by *matrix*.

The floating-point number (9) in m1 causes results to be calculated in floating-point form.

[1.2,3;4.5,6;7.8,9.]>m1 **ENTER**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix}$$

QR m1.qm.rm **ENTER**

Done

qm **ENTER**

$$\begin{bmatrix} .123... & .904... & .408... \\ .492... & .301... & -.816... \\ .861... & -.301... & .408... \end{bmatrix}$$

rm **ENTER**

$$\begin{bmatrix} 8.124... & 9.601... & 11.078... \\ 0. & .904... & 1.809... \\ 0. & 0. & 0. \end{bmatrix}$$

[m,n;o,p]>m1 **ENTER**

$$\begin{bmatrix} m & n \\ 0 & p \end{bmatrix}$$

QR m1.qm.rm **ENTER**

Done

qm **ENTER**

$$\begin{bmatrix} \frac{m}{\sqrt{m^2+0^2}} & \frac{-\text{sign}(m \cdot p - n \cdot o) \cdot o}{\sqrt{m^2+0^2}} \\ 0 & \frac{m \cdot \text{sign}(m \cdot p - n \cdot o)}{\sqrt{m^2+0^2}} \end{bmatrix}$$

rm **ENTER**

$$\begin{bmatrix} \sqrt{m^2+0^2} & \frac{m \cdot n + o \cdot p}{\sqrt{m^2+0^2}} \\ 0 & \frac{|m \cdot p - n \cdot o|}{\sqrt{m^2+0^2}} \end{bmatrix}$$

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]

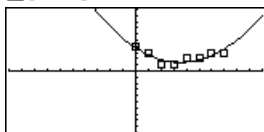
Regeq(x) → y1(x) **[ENTER]**

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 ...}

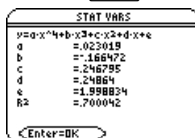
{4,3,1,2,4,2,1,4,6} → L2 **[ENTER]**

{4 3 1 ...}

QuartReg L1,L2 **[ENTER]**

Done

ShowStat **[ENTER]**



[ENTER]

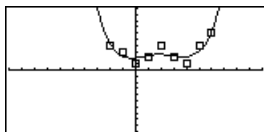
Regeq(x) → y1(x) **[ENTER]**

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]

$$\begin{array}{l} \blacksquare \text{R►P}\theta(x, y) \\ 90 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right) \end{array}$$

In Gradian angle mode:

R►Pθ(*x*, *y*) [ENTER]

$$\begin{array}{l} \blacksquare \text{R►P}\theta(x, y) \\ 100 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right) \end{array}$$

In Radian angle mode:

R►Pθ(3, 2) [ENTER]

R►Pθ([3, -4, 2], [0, $\pi/4$, 1.5]) [ENTER]

$$\begin{array}{l} \blacksquare \text{R►P}\theta(3, 2) \quad \tan^{-1}(2/3) \\ \blacksquare \text{R►P}\theta([3 \ -4 \ 2], [0 \ \frac{\pi}{4} \ 1.5]) \\ \left[0 \ \tan^{-1}\left(\frac{16}{\pi}\right) + \frac{\pi}{2} \ .643501 \right] \end{array}$$

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, $\pi/4$, 1.5]) [ENTER]

$$\begin{array}{l} \blacksquare \text{R►Pr}(3, 2) \quad \sqrt{13} \\ \blacksquare \text{R►Pr}(x, y) \quad \sqrt{x^2 + y^2} \\ \blacksquare \text{R►Pr}([3 \ -4 \ 2], [0 \ \frac{\pi}{4} \ 1.5]) \\ \left[3 \ \frac{\sqrt{\pi^2 + 256}}{4} \ 2.5 \right] \end{array}$$

►Rad CATALOG/MATH/Angle menu►Rad *expression*

Converts an expression to radian angle measure.

In Degree angle mode:

1.5 ►Rad [ENTER] .02618°

In Gradian angle mode:

1.5 ►Rad [ENTER] .023562°

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] .158...

rand(6) [ENTER] 5

rand(-100) [ENTER] -49

randMat() MATH/Probability menu

randMat(*numRows*, *numColumns*) \Rightarrow *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*) \Rightarrow *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**

.492...

randNorm(3,4.5) **ENTER**

-3.543...

randPoly() MATH/Probability menu

randPoly(*var*, *order*) \Rightarrow *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**

$-2 \cdot x^5 + 3 \cdot x^4 - 6 \cdot x^3 + 4 \cdot x - 6$

RandSeed MATH/Probability menu

RandSeed *number*

If *number* = 0, sets the seeds to the factory defaults for the random-number generator. If *number* \neq 0, it is used to generate two seeds, which are stored in system variables seed1 and seed2.

RandSeed 1147 **ENTER**

Done

rand() **ENTER**

.158...

RclGDB CATALOG

RclGDB *GDBvar*

Restores all the settings stored in the Graph database variable GDBvar.

For a listing of the settings, see **StoGDB**.

Note: It is necessary to have something saved in GDBvar before you can restore it.

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(*expression*) \Rightarrow *expression*

Returns the real part of the argument.

Note: All undefined variables are treated as real variables. See also **imag()**.

real(2+3i) **ENTER**

2

real(z) **ENTER**

z

real(x+iy) **ENTER**

x

real(list) \Rightarrow *list* `real({a+ib,3,i})` `[ENTER]` {a 3 0}

Returns the real parts of all elements.

real(matrix) \Rightarrow *matrix* `real([a+ib,3;c,i])` `[ENTER]` $\begin{bmatrix} a & 3 \\ c & 0 \end{bmatrix}$

Returns the real parts of all elements.

►Rect MATH/Matrix/Vector ops menu

vector►**Rect** `[3,∠π/4,∠π/6]►Rect` `[ENTER]`

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.

$$\left[\frac{3 \cdot \sqrt{2}}{4} \quad \frac{3 \cdot \sqrt{2}}{4} \quad \frac{3 \cdot \sqrt{3}}{2} \right]$$

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.

$$\begin{aligned} & [a \cdot \angle b \cdot \angle c] \text{ [ENTER]} & [a \cdot \cos(b) \cdot \sin(c) \\ & a \cdot \sin(b) \cdot \sin(c) & a \cdot \cos(c) \end{aligned}$$

Note: See also ►**Polar**.

complexValue►**Rect** In Radian angle mode:

Displays *complexValue* in rectangular form $a+bi$. The *complexValue* can have any complex form. However, an $re^{b\theta}$ entry causes an error in Degree angle mode.

$$4e^{(\pi/3)} \text{ ►Rect [ENTER]} \quad 4 \cdot e^{i\pi/3}$$

$$(4\angle\pi/3) \text{ ►Rect [ENTER]} \quad 2+2 \cdot \sqrt{3} \cdot i$$

Note: You must use parentheses for an $(r\angle\theta)$ polar entry.

In Gradian angle mode:

$$(1\angle 100) \text{ ►Rect [ENTER]} \quad i$$

In Degree angle mode:

$$(4\angle 60) \text{ ►Rect [ENTER]} \quad 2+2 \cdot \sqrt{3} \cdot i$$

Note: To type ►**Rect** from the keyboard, press `[2nd] [►]` for the ► operator. To type \angle , press `[2nd] [∠]`.

ref() MATH/Matrix menu

ref(matrix), *tol* \Rightarrow *matrix* `ref([-2,-2.0,-6;1,-1.9,-9;-5,`

Returns the row echelon form of *matrix*.

`2.4,-4])` `[ENTER]`

Optionally, any matrix element is treated as zero if its absolute value is less than *tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *tol* is ignored.

$$\begin{bmatrix} 1 & -2/5 & -4/5 & 4/5 \\ 0 & 1 & 4/7 & 11/7 \\ 0 & 0 & 1 & -62/71 \end{bmatrix}$$

- If you use `[◻] [ENTER]` or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.

$$[a \cdot b \cdot c : e \cdot f \cdot g] \text{ ►m1 [ENTER]} \quad \begin{bmatrix} a & b & c \\ e & f & g \end{bmatrix}$$

- If *tol* is omitted or not used, the default tolerance is calculated as:

$$\text{ref(m1) [ENTER]} \quad \begin{bmatrix} 1 & \frac{f}{e} & \frac{g}{e} \\ 0 & 1 & \frac{a \cdot g - c \cdot e}{a \cdot f - b \cdot e} \end{bmatrix}$$

$$5E-14 * \max(\text{dim}(\text{matrix})) * \text{rowNorm}(\text{matrix})$$

Note: See also **rref()**.

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:

$$\begin{aligned} \text{remain}(x,0) &= x \\ \text{remain}(x,y) &= x - y * \text{iPart}(x/y) \end{aligned}$$

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**() .

```
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]
                                     [ 1 -1
                                      2  1 ]
```

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* [, *alphaOn/Off*]

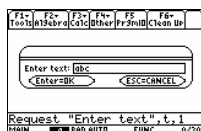
If Request is inside a Dialog...EndDialog construct, it creates an input box for the user to type in data. If it is a stand-alone instruction, it creates a dialog box for this input. In either case, if *var* contains a string, it is displayed and highlighted in the input box as a default choice. *promptString* must be ≤ 20 characters.

This instruction can be stand-alone or part of a dialog construct.

The optional *alphaOn/Off* argument can be any expression. If it evaluates to zero, alpha-lock is set to OFF. If it evaluates to anything other than zero, alpha-lock is set to ON. If the optional argument is not used, alpha-lock defaults to ON.

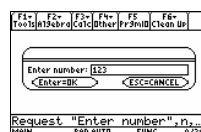
If more than one Request command appears within a Dialog...EndDialog construct, the first alpha setting is used and subsequent ones are ignored.

Request "Enter text",t,1 [ENTER]



The argument turned on alpha-lock in the above example.

Request "Enter number",n,0



The argument turned off alpha-lock in the above example.

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.

Note: Enter the text as one long line on the Home screen (without line breaks).

```
Define factorial(n)=Func
:local answer,count:1 → answer
:For count,1,n
:answer*count → answer:EndFor
:Return answer:EndFunc [ENTER] Done
factorial(3) [ENTER] 6
```

right() MATH/List menu

right (<i>list1</i> , <i>num</i>) ⇒ <i>list</i>	<code>right({1,3,-2.4},3)</code> ENTER	{3 -2 4}
Returns the rightmost <i>num</i> elements contained in <i>list1</i> .		
If you omit <i>num</i> , returns all of <i>list1</i> .		
right (<i>sourceString</i> , <i>num</i>) ⇒ <i>string</i>	<code>right("Hello",2)</code> ENTER	"lo"
Returns the rightmost <i>num</i> characters contained in character string <i>sourceString</i> .		
If you omit <i>num</i> , returns all of <i>sourceString</i> .		
right (<i>comparison</i>) ⇒ <i>expression</i>	<code>right(x<3)</code> ENTER	3
Returns the right side of an equation or inequality.		

root() CATALOG/MATH/Number menu

root (<i>expression</i>) ⇒ <i>root</i>	<code>root(8,3)</code> ENTER	2
Computes an <i>n</i> th root of <i>x</i> where <i>x</i> can be a real or complex floating point constant, an integer or complex rational constant, or a general symbolic expression.	<code>root(3,3)</code> ENTER	3 ^{1/3}
	<code>root(3,0,3)</code> ENTER	1.442249570

rotate() MATH/Base menu

rotate (<i>integer1</i> , #ofRotations) ⇒ <i>integer</i>	In Bin base mode:	
Rotates the bits in a binary integer. You can enter <i>integer1</i> in any number base; it is converted automatically to a signed, 32-bit binary form. If the magnitude of <i>integer1</i> is too large for this form, a symmetric modulo operation brings it within the range.	<code>rotate(0b1111010110000110101)</code> ENTER	0b10000000000000111101011000011010
	<code>rotate(256,1)</code> ENTER	0b1000000000
	In Hex base mode:	
If #of Rotations is positive, the rotation is to the left. If #of Rotations is negative, the rotation is to the right. The default is -1 (rotate right one bit).	<code>rotate(0h78E)</code> ENTER	0h3C7
For example, in a right rotation:	<code>rotate(0h78E,-2)</code> ENTER	0h800001E3
	<code>rotate(0h78E,2)</code> ENTER	0h1E38
<p>↳ Each bit rotates right.</p> <pre> 0b00000000000001111010110000110101 ↑ Rightmost bit rotates to leftmost. </pre>	Important: To enter a binary or hexadecimal number, always use the 0b or 0h prefix (zero, not the letter O).	
produces:		
0b1000000000000111101011000011010		
The result is displayed according to the Base mode.		
rotate (<i>list1</i> , #ofRotations) ⇒ <i>list</i>	In Dec base mode:	
Returns a copy of <i>list1</i> rotated right or left by #of Rotations elements. Does not alter <i>list1</i> .	<code>rotate({1,2,3,4})</code> ENTER	{4 1 2 3}
If #of Rotations is positive, the rotation is to the left. If #of Rotations is negative, the rotation is to the right. The default is -1 (rotate right one element).	<code>rotate({1,2,3,4},-2)</code> ENTER	{3 4 1 2}
	<code>rotate({1,2,3,4},1)</code> ENTER	{2 3 4 1}

rotate (<i>string</i> 1, #ofRotations) ⇒ <i>string</i>	rotate("abcd") <input type="text" value="ENTER"/>	"dabc"
Returns a copy of <i>string</i> 1 rotated right or left by #of Rotations characters. Does not alter <i>string</i> 1.	rotate("abcd", -2) <input type="text" value="ENTER"/>	"cdab"
If #of Rotations is positive, the rotation is to the left. If #of Rotations is negative, the rotation is to the right. The default is -1 (rotate right one character).	rotate("abcd", 1) <input type="text" value="ENTER"/>	"bcda"

round() MATH/Number menu

round (<i>expression</i> 1[, <i>digits</i>]) ⇒ <i>expression</i>	round(1.234567, 3) <input type="text" value="ENTER"/>	1.235
Returns the argument rounded to the specified number of digits after the decimal point.		
<i>digits</i> must be an integer in the range 0–12. If <i>digits</i> is not included, returns the argument rounded to 12 significant digits.		
Note: Display digits mode may affect how this is displayed.		

round (<i>list</i> 1[, <i>digits</i>]) ⇒ <i>list</i>	round({ π , $\sqrt{2}$ }, 4) <input type="text" value="ENTER"/>	{3.1416 1.4142 .6931}
Returns a list of the elements rounded to the specified number of digits.		

round (<i>matrix</i> 1[, <i>digits</i>]) ⇒ <i>matrix</i>	round([ln(5), ln(3); π , $e^{(1)}$], 1) <input type="text" value="ENTER"/>	$\begin{bmatrix} 1.6 & 1.1 \\ 3.1 & 2.7 \end{bmatrix}$
Returns a matrix of the elements rounded to the specified number of digits.		

rowAdd() MATH/Matrix/Row ops menu

rowAdd (<i>matrix</i> 1, <i>rIndex</i> 1, <i>rIndex</i> 2) ⇒ <i>matrix</i>	rowAdd([3, 4; -3, -2], 1, 2) <input type="text" value="ENTER"/>	$\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$
Returns a copy of <i>matrix</i> 1 with row <i>rIndex</i> 2 replaced by the sum of rows <i>rIndex</i> 1 and <i>rIndex</i> 2.	rowAdd([a, b; c, d], 1, 2) <input type="text" value="ENTER"/>	$\begin{bmatrix} a & b \\ a+c & b+d \end{bmatrix}$

rowDim() MATH/Matrix/Dimensions menu

rowDim (<i>matrix</i>) ⇒ <i>expression</i>		$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
Returns the number of rows in <i>matrix</i> .	[1, 2; 3, 4; 5, 6] ► M1 <input type="text" value="ENTER"/>	
Note: See also colDim ().	rowdim(M1) <input type="text" value="ENTER"/>	3

rowNorm() MATH/Matrix/Norms menu

rowNorm (<i>matrix</i>) ⇒ <i>expression</i>	rowNorm([-5, 6; -7; 3, 4; 9; 9, -9, -7]) <input type="text" value="ENTER"/>	25
Returns the maximum of the sums of the absolute values of the elements in the rows in <i>matrix</i> .		
Note: All matrix elements must simplify to numbers. See also colNorm ().		

rowSwap() MATH/Matrix/Row ops menu

rowSwap(*matrix1*, *rIndex1*, *rIndex2*) \Rightarrow *matrix*

[1.2:3.4:5.6] \rightarrow Mat **ENTER**

Returns *matrix1* with rows *rIndex1* and *rIndex2* exchanged.

rowSwap(Mat.1,3) **ENTER**

1	2
3	4
5	6

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 **RcPic**.

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*, *tol*) \Rightarrow *matrix*

rref([-2. -2.0, -6;1. -1.9, -9;
-5.2,4. -4]) **ENTER**

Returns the reduced row echelon form of *matrix1*.

1	0	0	66/71
0	1	0	147/71
0	0	1	-62/71

Optionally, any matrix element is treated as zero if its absolute value is less than *tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *tol* is ignored.

rref([a,b,x:c,d,y]) **ENTER**

1	0	$\frac{d \cdot x - b \cdot y}{a \cdot d - b \cdot c}$
0	1	$\frac{-(c \cdot x - a \cdot y)}{a \cdot d - b \cdot c}$

- If you use \square **ENTER** or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.
- If *tol* is omitted or not used, the default tolerance is calculated as:

$$5e^{-14} * \max(\dim(\text{matrix1})) * \text{rowNorm}(\text{matrix1})$$

Note: See also **rref**().

sec() MATH/Trig menu

sec(*expression1*) \Rightarrow *expression*

sec(*list1*) \Rightarrow *list*

In Degree angle mode:

sec(45) **ENTER**

$\sqrt{(2)}$

sec({1.2.3.4}) **ENTER**

Returns the secant of *expression1* or returns a list containing the secants of all elements in *list1*.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

$\left\{ \frac{1}{\cos(1)} \ 1.000\dots \frac{1}{\cos(4)} \right\}$

sec⁻¹() **MATH/Trig menu**

sec⁻¹(*expression1*) ⇒ *expression*
sec⁻¹(*list1*) ⇒ *list*

Returns the angle whose secant is *expression1* or returns a list containing the inverse secants of each element of *list1*.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:
 sec⁻¹(1) [ENTER] 0

In Gradian angle mode:
 sec⁻¹(√2) [ENTER] 50

In Radian angle mode:
 sec⁻¹({1.2,5}) [ENTER] { 0 $\frac{\pi}{3}$ cos⁻¹(1/5) }

sech() **MATH/Hyperbolic menu**

sech(*expression1*) ⇒ *expression*
sech(*list1*) ⇒ *list*

Returns the hyperbolic secant of *expression1* or returns a list containing the hyperbolic secants of the *list1* elements.

sech(3) [ENTER] $\frac{1}{\cosh(3)}$

sech({1.2,3,4}) [ENTER] { $\frac{1}{\cosh(1)}$.198... $\frac{1}{\cosh(4)}$ }

sech⁻¹() **MATH/Hyperbolic menu**

sech⁻¹(*expression1*) ⇒ *expression*
sech⁻¹(*list1*) ⇒ *list*

Returns the inverse hyperbolic secant of *expression1* or returns a list containing the inverse hyperbolic secants of each element of *list1*.

In Radian angle and Rectangular complex mode:
 sech⁻¹(1) [ENTER] 0

sech⁻¹({1,-2.2,1}) [ENTER] {0 $(\frac{2 \cdot \pi}{3}) \cdot i$ 1.074... $\cdot i$ }

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, where another unit linked to that port can receive the variable value. The receiving unit must be on the Home screen or must execute **GetCalc** from a program.

If you send from a TI-89, TI-92 Plus, or Voyage™ 200 to a TI-92, an error occurs if the TI-92 executes **GetCalc** from a program. In this case, the sending unit must use **SendChat** instead.

Program segment:
 :
 :a+b>x
 :SendCalc x
 :

SendCalc *var[,port]*

Sends contents of *var* from a TI-89 Titanium to another TI-89 Titanium.

If the port is not specified, or *port = 0* is specified, the TI-89 Titanium sends data using the USB port if connected, if not, it will send using the I/O port.

If *port = 1*, the TI-89 Titanium sends data using the USB port only.

If *port = 2*, the TI-89 Titanium sends data using the I/O port only.

SendChat CATALOG

SendChat *var*

A general alternative to **SendCalc**, this is useful if the receiving unit is a TI-92 (or for a generic "chat" program that allows either a TI-92, Voyage™ 200, or TI-92 Plus to be used). Refer to **SendCalc** for more information.

SendChat sends a variable only if that variable is compatible with the TI-92, which is typically true in "chat" programs. However, **SendChat** will not send an archived variable, a TI-89 graph data base, etc.

Program segment:

```
:  
: a+b → x  
: SendChat 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...

setDate() CATALOG

setDate(*year, month, day*) ⇒ *listold*

Sets the clock to the date given in the argument and returns a list. (**Note:** The *year* must fall in the range 1997 - 2132.) The returned list is in {*yearold, monthold, dayold*} format. The returned date is the previous clock value.

Enter the year as a four-digit integer. The month and day can be either one- or two-digit integers.

setDate(2001, 10, 31) **[ENTER]**
{ 2001 11 1 }

setDtFmt() CATALOG

setDtFmt(*integer*) ⇒ *integerold*

Sets the date format for the desktop according to the argument and returns the previous date format value.

Integer values:

1 = MM/DD/YY 5 = YY.MM.DD
2 = DD/MM/YY 6 = MM-DD-YY
3 = MM.DD.YY 7 = DD-MM-YY
4 = DD.MM.YY 8 = YY-MM-DD

setFold() CATALOG

setFold (<i>newfolderName</i>) ⇒ <i>oldfolderString</i>	newFold chris <input type="text" value="ENTER"/>	Done
Returns the name of the current folder as a string and sets <i>newfolderName</i> as the current folder.	setFold(main) <input type="text" value="ENTER"/>	"chris"
The folder <i>newfolderName</i> must exist.	setFold(chris)⇒oldfoldr <input type="text" value="ENTER"/>	"main"
	1→ a <input type="text" value="ENTER"/>	1
	setFold(#oldfoldr) <input type="text" value="ENTER"/>	"chris"
	a <input type="text" value="ENTER"/>	a
	chris\ a <input type="text" value="ENTER"/>	1

setGraph() CATALOG

setGraph (<i>modeNameString</i> , <i>settingString</i>) ⇒ <i>string</i>	setGraph("Graph Order", "Seq") <input type="text" value="ENTER"/>	"SEQ"
Sets the Graph mode <i>modeNameString</i> to <i>settingString</i> , and returns the previous setting of the mode. Storing the previous setting lets you restore it later.	setGraph("Coordinates", "Off") <input type="text" value="ENTER"/>	"RECT"

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.

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" (not 3D graph mode) "Off", "Axes", "Box" (3D graph mode)
"Leading Cursor"	"Off", "On" ²
"Labels"	"Off", "On"
"Style"	"Wire Frame", "Hidden Surface", "Contour Levels", "Wire and Contour", "Implicit Plot" ³
"Seq Axes"	"Time", "Web", "U1-vs-U2" ⁴
"DE Axes"	"Time", "t-vs-y", "y-vs-y'", "y1-vs-y2", "y1-vs-y2'", "y1'-vs-y2'" ⁵
	Tip: To type a prime symbol ('), press <input type="text" value="2nd"/> ['].
"Solution Method"	"RK", "Euler" ⁵
"Fields"	"SlpFld", "DirFld", "FldOff" ⁵
"Discontinuity Detection"	"Off", "On" ⁶

¹Not available in Sequence, 3D, or Diff Equations graph mode. Also not available in Function graph mode with "Discontinuity Detection" set to "On."

²Not available in 3D graph mode.

³Applies only to 3D graph mode.

⁴Applies only to Sequence graph mode.

⁵Applies only to Diff Equations graph mode.

⁶Applies only to Function graphing mode, when "Graph Order" is set to "Seq."

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: To set or return information about the Unit System mode, use **setUnits()** or **getUnits()** instead of **setMode()** or **getMode()**.

setMode("Angle". "Degree") [ENTER]	"RADIAN"
sin(45) [ENTER]	$\frac{\sqrt{2}}{2}$
setMode("Angle". "Radian") [ENTER]	"DEGREE"
sin($\pi/4$) [ENTER]	$\frac{\sqrt{2}}{2}$
setMode("Angle". "Gradian") [ENTER]	"RADIAN"
sin(50) [ENTER]	$\frac{\sqrt{2}}{2}$
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 unit.

Mode Name	Settings
"Graph"	"Function", "Parametric", "Polar", "Sequence", "3D", "Diff Equations"
"Display Digits"	"Fix 0", "Fix 1", ..., "Fix 12", "Float", "Float 1", ..., "Float 12"
"Angle"	"Radian", "Degree", "Gradian"
"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", "Text Editor", "Numeric Solver", "Flash App"
"Split 2 App"	"Home", "Y= Editor", "Window Editor", "Graph", "Table", "Data/Matrix Editor", "Program Editor", "Text Editor", "Numeric Solver", "Flash App"

"Number of Graphs"	"1", "2"
"Graph2"	"Function", "Parametric", "Polar", "Sequence", "3D", "Diff Equations"
"Split Screen Ratio"	"1:1", "1:2", "2:1" (Voyage™ 200 only)
"Exact/Approx"	"Auto", "Exact", "Approximate"
"Base"	"Dec", "Hex", "Bin"
"Language"	"English", "Alternate Language"
"Apps Desktop"	"Off", "On"

setTable() CATALOG

setTable(*modelNameString*, *settingString*) ⇒ *string*

Sets the table parameter *modelNameString* to *settingString*, and returns the previous setting of the parameter. Storing the previous setting lets you restore it later.

modelNameString 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"

setTime() CATALOG

setTime(*hour*, *minute*, *second*) ⇒ *listold*

Sets the clock to the time given in the argument and returns a list. The list is in {*hourold*, *minuteold*, *secondold*} format. The returned time is the previous clock value.

Enter the hour in the 24 hour format, in which 13 = 1 p.m.

```
setTime(11.32.50)
{10 44 49}
```

setTmFmt() CATALOG

setTmFmt(*integer*) ⇒ *integerold*

Sets the time format for the desktop according to the argument and returns the previous time format value.

Integer values:

12 = 12 hour clock

24 = 24 hour clock

setTmZn() CATALOG

setTmZn(*integer*) ⇒ *integerold*

Sets the time zone according to the argument and returns the previous time zone value.

The time zone is defined by an integer that gives the minutes offset from Greenwich Mean Time (GMT), as established in Greenwich, England. For example, if the time zone is offset from GMT by two hours, the device would return 120 (minutes).

Integers for time zones west of GMT are negative.

Integers for time zones east of GMT are positive.

If Greenwich Mean Time is 14:07:07, it is:

7:07:07 a.m. in Denver, Colorado (Mountain Standard Time) (-420 minutes from GMT)

15:07:07 p.m. in Brussels, Belgium (Central European Standard Time) (+60 minutes from GMT)

setUnits() CATALOG

setUnits(*list1*) ⇒ *list*

Sets the default units to the values specified in *list1*, and returns a list of the previous defaults.

- To specify the built-in SI (metric) or ENG/US system, *list1* uses the form:

```
{"SI"} or {"ENG/US"}
```

- To specify a custom set of default units, *list1* uses the form:

```
{"CUSTOM", "cat1", "unit1" [, "cat2", "unit2", ...]}
```

where each *cat* and *unit* pair specifies a category and its default unit. (You can specify built-in units only, not user-defined units.) Any category not specified will use its previous custom unit.

- To return to the previous custom default units, *list1* uses the form:

```
{"CUSTOM"}
```

If you want different defaults depending on the situation, create separate lists and save them to unique list names. To use a set of defaults, specify that list name in **setUnits()**.

You can use **setUnits()** to restore settings previously saved with **setUnits()** → *var* or with **getUnits()** → *var*.

All unit names must begin with an underscore _.

```
[2nd] [-]
```

You can also select units from a menu by pressing:

```
[◻] [UNITS]
```

```
setUnits({"SI"}) [ENTER]
{"SI" "Area" "NONE"
"Capacitance" "_F" ...}
```

```
setUnits({"CUSTOM"."Length".
"_cm"."Mass"."_gm"}) [ENTER]
{"SI" "Length" "_m"
"Mass" "_kg" ...}
```

Note: Your screen may display different units.

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°

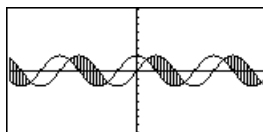
patRes specifies the resolution of the shading patterns:

- 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. **Shade** is not valid in 3D graphing mode.

In the ZoomTrig viewing window:

Shade $\cos(x) \cdot \sin(x)$

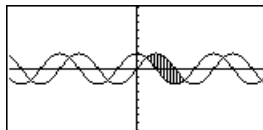


[CALC HOME]

C1rDraw

Shade $\cos(x) \cdot \sin(x) \cdot 0.5$

Done

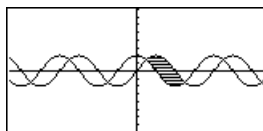


[CALC HOME]

C1rDraw

Shade $\cos(x) \cdot \sin(x) \cdot 0.5 \cdot 2$

Done

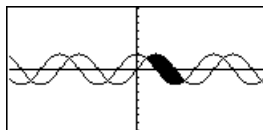


[CALC HOME]

C1rDraw

Shade $\cos(x) \cdot \sin(x) \cdot 0.5 \cdot 2 \cdot 1$

Done



ShowStat CATALOG

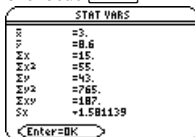
ShowStat

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(*expression*) ⇒ *expression*

sign(*list*) ⇒ *list*

sign(*matrix*) ⇒ *matrix*

For real and complex *expression*1, returns *expression*1/abs(*expression*1) when *expression*1 ≠ 0.

Returns 1 if *expression*1 is positive.

Returns -1 if *expression*1 is negative.

sign(0) returns ±1 if the complex format mode is REAL; otherwise, it returns itself.

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(1+abs(x)) **[ENTER]** 1

If complex format mode is REAL:
 sign([-3,0,3]) **[ENTER]** [-1 ±1 1]

simult() MATH/Matrix menu

simult(*coeffMatrix*, *constVector*, *tol*) ⇒ *matrix*

Returns a column vector that contains the solutions to a system of linear equations.

coeffMatrix must be a square matrix that contains the coefficients of the equations.

constVector must have the same number of rows (same dimension) as *coeffMatrix* and contain the constants.

Optionally, any matrix element is treated as zero if its absolute value is less than *tol*. This tolerance is used only if the matrix has floating-point entries and does not contain any symbolic variables that have not been assigned a value. Otherwise, *tol* is ignored.

- If you use **[MODE]** or set the mode to Exact/Approx=APPROXIMATE, computations are done using floating-point arithmetic.

- If *tol* is omitted or not used, the default tolerance is calculated as:

$$5E-14 * \max(\text{dim}(\text{coeffMatrix})) * \text{rowNorm}(\text{coeffMatrix})$$

Solve for x and y: $x + 2y = 1$
 $3x + 4y = -1$

simult([1,2;3,4],[1;-1]) **[ENTER]** $\begin{bmatrix} -3 \\ 2 \end{bmatrix}$

The solution is x = -3 and y = 2.

Solve: $ax + by = 1$
 $cx + dy = 2$

[a,b;c,d] → matx1 **[ENTER]** $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

simult(matx1,[1;2]) **[ENTER]** $\begin{bmatrix} -(2 \cdot b - d) \\ a \cdot d - b \cdot c \\ \frac{2 \cdot a - c}{a \cdot d - b \cdot c} \end{bmatrix}$

simult(*coeffMatrix*, *constMatrix*, *tol*) \Rightarrow *matrix*

Solves multiple systems of linear equations, where each system has the same equation coefficients but different constants.

Each column in *constMatrix* must contain the constants for a system of equations. Each column in the resulting matrix contains the solution for the corresponding system.

Solve: $x + 2y = 1$
 $x + 2y = 2$
 $3x + 4y = -1$
 $3x + 4y = -3$

simult([1,2;3,4],[1,2;-1,-3])
[ENTER]

$\begin{bmatrix} -3 & -7 \\ 2 & 9/2 \end{bmatrix}$

For the first system, $x = -3$ and $y = 2$. For the second system, $x = -7$ and $y = 9/2$.

sin() SIN key

sin(*expression*) \Rightarrow *expression*

sin(*list*) \Rightarrow *list*

sin(*expression*) returns the sine of the argument as an expression.

sin(*list*) returns a list of the sines of all elements in *list*.

Note: The argument is interpreted as a degree, gradian or radian angle, according to the current angle mode. You can use $^{\circ}$, g or r to override the angle mode setting temporarily.

In Degree angle mode:

sin($(\pi/4)^r$) [ENTER] $\frac{\sqrt{2}}{2}$

sin(45) [ENTER] $\frac{\sqrt{2}}{2}$

sin({0,60,90}) [ENTER] {0 $\frac{\sqrt{3}}{2}$ 1}

In Gradian angle mode:

sin(50) [ENTER] $\frac{\sqrt{2}}{2}$

In Radian angle mode:

sin($\pi/4$) [ENTER] $\frac{\sqrt{2}}{2}$

sin(45 $^{\circ}$) [ENTER] $\frac{\sqrt{2}}{2}$

sin(*squareMatrix*) \Rightarrow *squareMatrix*

Returns the matrix sine of *squareMatrix*. This is *not* the same as calculating the sine of each element. For information about the calculation method, refer to **cos()**.

squareMatrix must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:

sin([1.5,3;4.2,1;6,-2,1]) [ENTER]

$\begin{bmatrix} .942... & -.045... & -.031... \\ -.045... & .949... & -.020... \\ -.048... & -.005... & .961... \end{bmatrix}$

sin⁻¹() 2nd SIN⁻¹ key

sin⁻¹(*expression*) \Rightarrow *expression*

sin⁻¹(*list*) \Rightarrow *list*

sin⁻¹(*expression*) returns the angle whose sine is *expression* as an expression.

sin⁻¹(*list*) returns a list of the inverse sines of each element of *list*.

Note: The result is returned as a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

sin⁻¹(1) [ENTER] 90

In Gradian angle mode:

sin⁻¹(1) [ENTER] 100

In Radian angle mode:

sin⁻¹({0..2..5}) [ENTER] {0 .201... .523...}

$\sin^{-1}(\text{squareMatrix1}) \Rightarrow \text{squareMatrix}$

Returns the matrix inverse sine of *squareMatrix1*. This is *not* the same as calculating the inverse sine of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode and Rectangular complex format mode:

$\sin^{-1}([1.5,3;4,2.1;6,-2.1])$ **[ENTER]**

$\begin{bmatrix} -1.164... - 0.064... \cdot i^{\circ} & 1.490... - 2.105... \cdot i^{\circ} & \dots \\ .725... - 1.515... \cdot i^{\circ} & .947... - .773... \cdot i^{\circ} & \dots \\ 2.083... - 2.632... \cdot i^{\circ} & -1.790... + 1.271... \cdot i^{\circ} & \dots \end{bmatrix}$
--

sinh() MATH/Hyperbolic menu

$\sinh(\text{expression1}) \Rightarrow \text{expression}$

$\sinh(\text{list1}) \Rightarrow \text{list}$

sinh (*expression1*) returns the hyperbolic sine of the argument as an expression.

sinh (*list1*) 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...}

$\sinh(\text{squareMatrix1}) \Rightarrow \text{squareMatrix}$

Returns the matrix hyperbolic sine of *squareMatrix1*. This is *not* the same as calculating the hyperbolic sine of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:

$\sinh([1.5,3;4,2.1;6,-2.1])$ **[ENTER]**

360.954	305.708	239.604
352.912	233.495	193.564
298.632	154.599	140.251

sinh⁻¹() MATH/Hyperbolic menu

$\sinh^{-1}(\text{expression1}) \Rightarrow \text{expression}$

$\sinh^{-1}(\text{list1}) \Rightarrow \text{list}$

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... $\sinh^{-1}(3)$ }

$\sinh^{-1}(\text{squareMatrix1}) \Rightarrow \text{squareMatrix}$

Returns the matrix inverse hyperbolic sine of *squareMatrix1*. This is *not* the same as calculating the inverse hyperbolic sine of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:

$\sinh^{-1}([1.5,3;4,2.1;6,-2.1])$ **[ENTER]**

.041...	2.155...	1.158...
1.463...	.926...	.112...
2.750...	-1.528...	.572...

SinReg MATH/Statistics/Regressions menu

SinReg *list1*, *list2* [, [*iterations*], [*period*] [, *list3*, *list4*]

Calculates the sinusoidal regression and updates all the system statistics variables.

All the lists must have equal dimensions except for *list4*.

list1 represents *xlist*.

list2 represents *ylist*.

list3 represents category codes.

list4 represents category include list.

iterations specifies the maximum number of times (1 through 16) a solution will be attempted. If omitted, 8 is used. Typically, larger values result in better accuracy but longer execution times, and vice versa.

period specifies an estimated period. If omitted, the difference between values in *list1* should be equal and in sequential order. If you specify *period*, the differences between *x* values can be unequal.

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.

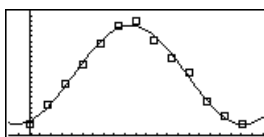
The output of **SinReg** is always in radians, regardless of the angle mode setting.

In function graphing mode:

```
seq(x,x,1,361,30)→L1 [ENTER]      {1 31 61 ...}
{5.5,8,11,13.5,16.5,19,19.5,17,
14.5,12.5,8.5,6.5,5.5}→L2 [ENTER]
                                         {5.5 8 11 ...}
SinReg L1,L2 [ENTER]
ShowStat [ENTER]                       Done
```



```
[ENTER]
regeq(x)→y1(x) [ENTER]              Done
NewPlot 1,1,1,1,L2 [ENTER]          Done
[GRAPH]
[F2] 9
```



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.

Solution candidates might not be real finite solutions for some combinations of values for undefined variables.

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.

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.

For inequalities of types \geq , \leq , $<$, or $>$, explicit solutions are unlikely unless the inequality is linear and contains only *var*.

For the EXACT setting of the Exact/Approx mode, portions that cannot be solved are returned as an implicit equation or inequality.

solve($a \cdot x^2 + b \cdot x + c = 0$), *x* [ENTER]

$$x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}$$

$$\text{or } x = \frac{-(\sqrt{b^2 - 4 \cdot a \cdot c} + b)}{2 \cdot a}$$

ans(1) | a=1 and b=1 and c=1
 [ENTER]

Error: Non-real result

solve((*x* - *a*) e^x = -*x* * (*x* - *a*), *x*) [ENTER]
 $x = a$ or $x = -.567...$

(*x*+1)(*x*-1)/(*x*-1)+*x*-3 [ENTER]

$2 \cdot x - 2$

solve(*entry*(1)=0, *x*) [ENTER]

$x = 1$

entry(2) | *ans*(1) [ENTER]

undef

limit(*entry*(3), *x*, 1) [ENTER]

0

solve($5x - 2 \geq 2x$), *x* [ENTER]

$x \geq 2/3$

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.

false is returned when no real solutions are found. true is returned if **solve()** can determine that any finite real value of *vars* satisfies the equation or inequality.

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.

Solutions might contain a unique new undefined variable of the form @n_j/with *j* being an integer in the interval 1–255. Such variables designate an arbitrary integer.

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.

Note: See also **cSolve()**, **cZeros()**, **nSolve()**, and **zeros()**.

solve(*equation1* and *equation2* [and ...], {*varOrGuess1*, *varOrGuess2* [, ...]}) ⇒ *Boolean expression*

Returns candidate real solutions to the simultaneous algebraic equations, where each *varOrGuess* specifies a variable that you want to solve for.

Optionally, you can specify an initial guess for a variable. Each *varOrGuess* must have the form:

variable
– or –
variable = *real or non-real number*

For example, *x* is valid and so is *x*=3.

If all of the equations are polynomials and if you do NOT specify any initial guesses, **solve()** uses the lexical Gröbner/Buchberger elimination method to attempt to determine **all** real solutions.

For example, suppose you have a circle of radius *r* at the origin and another circle of radius *r* centered where the first circle crosses the positive *x*-axis. Use **solve()** to find the intersections.

As illustrated by *r* in the example to the right, simultaneous *polynomial* equations can have extra variables that have no values, but represent given numeric values that could be substituted later.

You can also (or instead) include solution variables that do not appear in the equations. For example, you can include *z* as a solution variable

In Radian angle mode:

`solve(tan(x)=1/x,x)|x>0 and x<1` **ENTER** *x*

`solve(x=x+1,x)` **ENTER** false

`solve(x=x,x)` **ENTER** true

`2x-1≤1 and solve(x^2≠9,x)` **ENTER**
x ≤ 1 and *x* ≠ -3

In Radian angle mode:

`solve(sin(x)=0,x)` **ENTER** *x* = @n1 • π

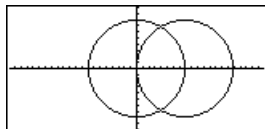
`solve(x^(1/3)=-1,x)` **ENTER** *x* = -1

`solve(√(x)=-2,x)` **ENTER** false

`solve(-√(x)=-2,x)` **ENTER** *x* = 4

`solve(y=x^2-2 and x+2y=-1,{x,y})` **ENTER**

x=1 and *y*=-1
or *x*=-3/2 and *y*=1/4



`solve(x^2+y^2=r^2 and (x-r)^2+y^2=r^2,{x,y})` **ENTER**

x = $\frac{r}{2}$ and *y* = $\frac{\sqrt{3} \cdot r}{2}$

or *x* = $\frac{r}{2}$ and *y* = $\frac{-\sqrt{3} \cdot r}{2}$

`solve(x^2+y^2=r^2 and (x-r)^2+y^2=r^2,{x,y,z})` **ENTER**

to extend the previous example to two parallel intersecting cylinders of radius r .

The cylinder solutions illustrate how families of solutions might contain arbitrary constants of the form $@k$, where k is an integer suffix from 1 through 255. The suffix resets to 1 when you use **ClrHome** or $\boxed{\text{F1}}$ 8:Clear Home.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list solution variables. If your initial choice exhausts memory or your patience, try rearranging the variables in the equations and/or *varO*Guess list.

If you do not include any guesses and if any equation is non-polynomial in any variable but all equations are linear in the solution variables, **solve()** uses Gaussian elimination to attempt to determine all real solutions.

If a system is neither polynomial in all of its variables nor linear in its solution variables, **solve()** determines at most one solution using an approximate iterative method. To do so, the number of solution variables must equal the number of equations, and all other variables in the equations must simplify to numbers.

Each solution variable starts at its guessed value if there is one; otherwise, it starts at 0.0.

Use guesses to seek additional solutions one by one. For convergence, a guess may have to be rather close to a solution.

$$x = \frac{r}{2} \text{ and } y = \frac{\sqrt{3} \cdot r}{2} \text{ and } z = @1$$

$$\text{or } x = \frac{r}{2} \text{ and } y = \frac{-\sqrt{3} \cdot r}{2} \text{ and } z = @1$$

solve($e^x(z) * y = 1$ and $x - y = \sin(z)$), {x,y} $\boxed{\text{ENTER}}$

$$x = \frac{e \cdot \sin(z) + 1}{e^z + 1} \text{ and } y = \frac{-(\sin(z) - 1)}{e^z + 1}$$

solve($e^x(z) * y = 1$ and $-y = \sin(z)$), {y,z} $\boxed{\text{ENTER}}$

$$y = .041... \text{ and } z = 3.183...$$

solve($e^x(z) * y = 1$ and $-y = \sin(z)$), {y,z=2 π } $\boxed{\text{ENTER}}$

$$y = .001... \text{ and } z = 6.281...$$

SortA MATH/List menu

SortA *listName1* [, *listName2*] [, *listName3*] ...

{2,1,4,3} \rightarrow list1 $\boxed{\text{ENTER}}$

{2,1,4,3}

SortA *vectorName1* [, *vectorName2*] [, *vectorName3*] ...

SortA list1 $\boxed{\text{ENTER}}$

Done

Sorts the elements of the first argument in ascending order.

list1 $\boxed{\text{ENTER}}$

{1 2 3 4}

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.

{4,3,2,1} \rightarrow list2 $\boxed{\text{ENTER}}$

{4 3 2 1}

SortA list2,list1 $\boxed{\text{ENTER}}$

Done

list2 $\boxed{\text{ENTER}}$

{1 2 3 4}

list1 $\boxed{\text{ENTER}}$

{4 3 2 1}

All arguments must be names of lists or vectors. All arguments must have equal dimensions.

SortD MATH/List menu

SortD *listName1* [, *listName2*] [, *listName3*] ...

{2,1,4,3} \rightarrow list1 $\boxed{\text{ENTER}}$

{2 1 4 3}

SortD *vectorName1* [, *vectorName2*] [, *vectorName3*] ...

{1,2,3,4} \rightarrow list2 $\boxed{\text{ENTER}}$

{1 2 3 4}

Identical to **SortA**, except **SortD** sorts the elements in descending order.

SortD list1,list2 $\boxed{\text{ENTER}}$

Done

list1 $\boxed{\text{ENTER}}$

{4 3 2 1}

list2 $\boxed{\text{ENTER}}$

{3 4 1 2}

►Sphere MATH/Matrix/Vector ops menu

vector►Sphere

Displays the row or column vector in spherical form $[r \angle \theta \angle \phi]$.

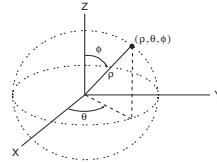
vector must be of dimension 3 and can be either a row or a column vector.

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
 $\boxed{\text{ENTER}}$ $[3.741... \angle 1.107... \angle .640...]$

$[2, \angle \pi/4, 3]$ ►Sphere
 $\boxed{\text{ENTER}}$ $[3.605... \angle .785... \angle .588...]$

$\boxed{\text{ENTER}}$ $[\sqrt{13} \angle \frac{\pi}{4} \angle \cos^{-1}(\frac{3 \cdot \sqrt{13}}{13})]$



startTmr() CATALOG

startTmr() \Rightarrow integer

Returns the current value of the clock in its integer representation, giving the *starttime* for a timer. You can enter the *starttime* as an argument in **checkTmr()** to determine how many seconds have elapsed.

You can run multiple timers simultaneously.

Note: See also **checkTmr()** and **timeCnv()**.

startTmr() $\boxed{\text{ENTER}}$ 148083315

checkTmr(148083315) 34

startTmr()►Timer1

⋮

startTmr()►Timer2

⋮

checkTmr(Timer1)►Timer1Value

⋮

checkTmr(Timer2)►Timer2Value

stdDev() MATH/Statistics menu

stdDev(list[, freqlist]) \Rightarrow expression

Returns the standard deviation of the elements in *list*.

Each *freqlist* element counts the number of consecutive occurrences of the corresponding element in *list*.

Note: *list* must have at least two elements.

stdDev({a,b,c}) $\boxed{\text{ENTER}}$

stdDev({1,2,5, -6,3, -2}) $\boxed{\text{ENTER}}$

$$\sqrt{\frac{3 \cdot (a^2 - a \cdot (b+c) + b^2 - b \cdot c)}{3}}$$

▪ stdDev({1 2 5 -6 3}) $\boxed{\text{ENTER}}$ $\frac{\sqrt{62}}{2}$

stdDev({1,3,2,5,-6,4},{3,2,5}) $\boxed{\text{ENTER}}$ 4.

stdDev(matrix1[, freqmatrix]) \Rightarrow matrix

Returns a row vector of the standard deviations of the columns in *matrix1*.

Each *freqmatrix* element counts the number of consecutive occurrences of the corresponding element in *matrix1*.

Note: *matrix1* must have at least two rows.

stdDev([1,2,5;-3,0,1;.5,.7,3]) $\boxed{\text{ENTER}}$
 $[2.179... \ 1.014... \ 2]$

stdDev([-1,2,5,3;2,5,7,3;6,-4].[4,2;3;.3;1,7]) $\boxed{\text{ENTER}}$
 $[2.7005, 5.44695]$

stdDevPop() MATH/Statistics menu

stdDevPop(*list*, *freqlist*) \Rightarrow *expression*

Returns the population standard deviation of the elements in *list*.

Each *freqlist* element counts the number of consecutive occurrences of the corresponding element in *list*.

Note: *list* must have at least two elements.

stdDevPop(*matrix1*[, *freqmatrix*]) \Rightarrow *matrix*

Returns a row vector of the population standard deviations of the columns in *matrix1*.

Each *freqmatrix* element counts the number of consecutive occurrences of the corresponding element in *matrix1*.

Note: *matrix1* must have at least two rows.

In Radian angle and auto modes:

stdDevPop({a,b,c}) **ENTER**

$$\sqrt{\frac{3(a^2 - a(b+c) + b^2 - b \cdot c)}{3}}$$

stdDevPop({1,2.5,-6,3,-2}) **ENTER**

$$\sqrt{\frac{1^2 + 2.5^2 + (-6)^2 + 3^2 + (-2)^2}{5}} = 4.65$$

stdDevPop({1,3,2.5,-6,4},{3,2,5}) **ENTER**

$$\sqrt{\frac{1^2 \cdot 3 + 3^2 \cdot 2 + 2.5^2 \cdot 5 + (-6)^2 \cdot 1 + 4^2 \cdot 1}{3+2+5}} = 4.11107$$

stdDevPop([[1,2,5],[-3,0,1]],[.5,.7,3]) **ENTER**

$$\left[\sqrt{\frac{1^2 + 2^2 + 5^2}{3}}, \sqrt{\frac{(-3)^2 + 0^2 + 1^2}{3}}, \sqrt{\frac{.5^2 + .7^2 + 3^2}{3}} \right] = [1.77951, .828654, \frac{2\sqrt{6}}{3}]$$

stdDevPop([-1,2,5,3;2,5,7,3;6,-4],[4,2;3,3;1,7]) **ENTER**

$$\left[\sqrt{\frac{(-1)^2 + 2^2 + 5^2 + 3^2}{4}}, \sqrt{\frac{2^2 + 5^2 + 7^2 + 3^2}{3}}, \sqrt{\frac{6^2 + (-4)^2}{1}} \right] = [2.52608, 5.21506]$$

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 or Diff Equations mode
- * Table flags
- * tblStart, Δ tbl, 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	Program segment:
Used as a program instruction to stop program execution.	: For i,1,10,1 If i=5 Stop EndFor :

StoPic CATALOG

StoPic <i>picVar</i> [, <i>pxlRow</i> , <i>pxlCol</i>] [, <i>width</i> , <i>height</i>]
Displays the graph screen and copies a rectangular area of the display to the variable <i>picVar</i> .
<i>pxlRow</i> and <i>pxlCol</i> , if included, specify the upper-left corner of the area to copy (defaults are 0, 0).
<i>width</i> and <i>height</i> , 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 → (store), page 912.

string() MATH/String menu

string (<i>expression</i>) ⇒ <i>string</i>	string(1.2345) ENTER	"1.2345"
Simplifies <i>expression</i> and returns the result as a character string.	string(1+2) ENTER	"3"
	string(cos(x)+√(3)) ENTER	"cos(x) + √(3)"

Style CATALOG

Style <i>equanum</i> , <i>stylePropertyString</i>	Style 1."thick" ENTER	Done
Sets the system graphing function <i>equanum</i> in the current graph mode to use the graphing property <i>stylePropertyString</i> .	Style 10."path" ENTER	Done
<i>equanum</i> must be an integer from 1–99 and the function must already exist.	Note: In function graphing mode, these examples set the style of $y_1(x)$ to "Thick" and $y_{10}(x)$ to "Path".	
<i>stylePropertyString</i> must be one of: "Line", "Dot", "Square", "Thick", "Animate", "Path", "Above", or "Below".		
Note that in parametric graphing, only the <i>xr</i> 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	
Diff Equations:	line, dot, square, thick, animate, path	
Note: Capitalization and blank spaces are optional when entering <i>stylePropertyString</i> names.		

subMat() CATALOG

subMat(*matrix1*[, *startRow*], [*startCol*], [*endRow*]
[, *endCol*]) ⇒ *matrix*

[1.2.3:4.5.6:7.8.9]►m1 **ENTER**

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Returns the specified submatrix of *matrix1*.

Defaults: *startRow*=1, *startCol*=1, *endRow*=last row, *endCol*=last column.

subMat(m1.2.1,3.2) **ENTER**

$$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$$

subMat(m1.2.2) **ENTER**

$$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$$

Sum (Sigma) See $\Sigma()$, page 908.

sum() MATH/List menu

sum(*list*[, *start*], *end*)] ⇒ *expression*

sum({1.2.3.4.5}) **ENTER**

15

Returns the sum of the elements in *list*.

sum({a.2a.3a}) **ENTER**

6 • a

Start and *end* are optional. They specify a range of elements.

sum(seq(n.n,1,10)) **ENTER**

55

sum({1.3.5.7.9}:3) **ENTER**

21

sum(*matrix1*[, *start*], *end*)] ⇒ *matrix*

sum([1.2.3:4.5.6]) **ENTER**

[5 7 9]

Returns a row vector containing the sums of the elements in the columns in *matrix1*.

sum([1.2.3:4.5.6:7.8.9]) **ENTER**

[12 15 18]

Start and *end* are optional. They specify a range of rows.

sum([1.2.3:4.5.6:7.8.9].2.3)

ENTER

[11.13.15]

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.

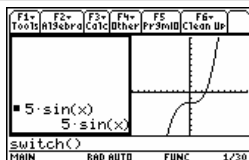
If *integer1* = 0, returns the active window number.

If *integer1* = 1, activates window 1 and returns the previously active window number.

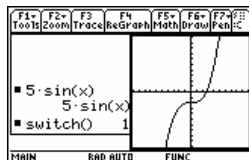
If *integer1* = 2, activates window 2 and returns the previously active window number.

If *integer1* is omitted, switches windows and returns the previously active window number.

integer1 is ignored if the TI-89 Titanium/Voyage™ 200 is not displaying a split screen.



switch() **ENTER**



T (transpose) **MATH/Matrix menu**

$matrix1^T \Rightarrow matrix$

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** 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, sequence, or diff equations graphing. As an alternative, you may want to use **BldData**.

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		

Table cos(time).time **ENTER**

x	1	2	3
0.	0.	1.	
1.	.67538	.5403	
2.	-1.04	-.4161	
3.	-3.712	-.99	
4.	-3.268	-.6536	

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 a degree, gradian or radian angle, according to the current angle mode. You can use $^{\circ}$, $^{\text{g}}$ or $^{\text{r}}$ to override the angle mode setting temporarily.

In Degree angle mode:

$\tan((\pi/4)^{\text{r}})$ ENTER 1

$\tan(45)$ ENTER 1

$\tan(\{0, 60, 90\})$ ENTER
{0 $\sqrt{3}$ undef}

In Gradian angle mode:

$\tan((\pi/4)^{\text{r}})$ ENTER $\frac{200 \bullet \tan(\frac{\pi}{4})}{\pi}$

$\tan(50)$ ENTER 1

$\tan(\{0, 50, 100\})$ ENTER
{0 1 undef}

In Radian angle mode:

$\tan(\pi/4)$ ENTER 1

$\tan(45^{\circ})$ ENTER 1

$\tan(\{\pi, \pi/3, -\pi, \pi/4\})$ ENTER
{0 $\sqrt{3}$ 0 1}

tan(squareMatrix1) ⇒ *squareMatrix*

Returns the matrix tangent of *squareMatrix1*. This is *not* the same as calculating the tangent of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:

$\tan([1.5, 3; 4.2, 1; 6, -2, 1])$ ENTER

$\begin{bmatrix} -28.291... & 26.088... & 11.114... \\ 12.117... & -7.835... & -5.481... \\ 36.818... & -32.806... & -10.459... \end{bmatrix}$

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 a degree, gradian or radian angle, according to the current angle mode setting.

In Degree angle mode:

$\tan^{-1}(1)$ ENTER 45

In Gradian angle mode:

$\tan^{-1}(1)$ ENTER 50

In Radian angle mode:

$\tan^{-1}(\{0, .2, .5\})$ ENTER
{0 .197... .463...}

tan⁻¹(squareMatrix1) ⇒ *squareMatrix*

Returns the matrix inverse tangent of *squareMatrix1*. This is *not* the same as calculating the inverse tangent of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:

$\tan^{-1}([1.5, 3; 4.2, 1; 6, -2, 1])$ ENTER

$\begin{bmatrix} -.083... & 1.266... & .622... \\ .748... & .630... & -.070... \\ 1.686... & -1.182... & .455... \end{bmatrix}$

tanh() MATH/Hyperbolic menu

tanh(*expression*) \Rightarrow *expression*

tanh(*list*) \Rightarrow *list*

tanh(*expression*) returns the hyperbolic tangent of the argument as an expression.

tanh(*list*) returns a list of the hyperbolic tangents of each element of *list*.

$\tanh(1.2)$ [ENTER] .833...

$\tanh(\{0,1\})$ [ENTER] {0 tanh(1)}

tanh(*squareMatrix*) \Rightarrow *squareMatrix*

Returns the matrix hyperbolic tangent of *squareMatrix1*. This is *not* the same as calculating the hyperbolic tangent of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode:

$\tanh([1.5,3;4.2,1;6,-2,1])$ [ENTER]

-.097...	.933...	.425...
.488...	.538...	-.129...
1.282...	-1.034...	.428...

tanh⁻¹() MATH/Hyperbolic menu

tanh⁻¹(*expression*) \Rightarrow *expression*

tanh⁻¹(*list*) \Rightarrow *list*

tanh⁻¹(*expression*) returns the inverse hyperbolic tangent of the argument as an expression.

tanh⁻¹(*list*) returns a list of the inverse hyperbolic tangents of each element of *list*.

In rectangular complex format mode:

$\tanh^{-1}(0)$ [ENTER] 0

$\tanh^{-1}(\{1,2,1,3\})$ [ENTER]
{ ∞ .518... -1.570... $\cdot i \frac{\ln(2)}{2} - \frac{\pi}{2}, i$ }

tanh⁻¹(*squareMatrix*) \Rightarrow *squareMatrix*

Returns the matrix inverse hyperbolic tangent of *squareMatrix1*. This is *not* the same as calculating the inverse hyperbolic tangent of each element. For information about the calculation method, refer to **cos()**.

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

In Radian angle mode and Rectangular complex format mode:

$\tanh^{-1}([1.5,3;4.2,1;6,-2,1])$ [ENTER]

-.099...	+1.64... $\cdot i^{\circ}$.267...	-1.491... $\cdot i^{\circ}$...
-.087...	-.725... $\cdot i^{\circ}$.479...	-.947... $\cdot i^{\circ}$...
.511...	-2.083... $\cdot i^{\circ}$	-.878...	+1.790... $\cdot i^{\circ}$...

taylor() MATH/Calculus menu

taylor(*expression1*, *var*, *order*, *point*) \Rightarrow *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.

$\text{taylor}(e^{\sqrt{x}}(\sqrt{x}), x, 2)$ [ENTER]

$\text{taylor}(e^t(t), t, 4) | t = \sqrt{x}$ [ENTER]

$\text{taylor}(e^{\sqrt{x}}, x, 2)$
$\text{taylor}(e^{\sqrt{x}}, x, 2, 0)$
$\text{taylor}(e^t, t, 4) t = \sqrt{x}$
$\frac{x^2}{24} + \frac{x^{3/2}}{6} + \frac{x}{2} + \sqrt{x} + 1$

$\text{taylor}(1/(x*(x-1)), x, 3)$ [ENTER]

$\text{taylor}\left(\frac{1}{x*(x-1)}, x, 3\right)$
$\text{taylor}\left(\frac{1}{x*(x-1)}, x, 3, 0\right)$

$\text{expand}(\text{taylor}(x/(x*(x-1)), x, 4)/x, x)$ [ENTER]

$\text{expand}\left(\frac{\text{taylor}\left(\frac{x}{x*(x-1)}, x, 4\right)}{x}, x\right)$
$-x^3 - x^2 - x - \frac{1}{x} - 1$

tCollect() MATHAlgebraTrig menu

tCollect(*expression*) \Rightarrow *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() MATHAlgebraTrig menu

tExpand(*expression*) \Rightarrow *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 $\pi/180$ interferes with the ability of **tExpand()** to recognize expandable forms. For best results, **tExpand()** should be used in Radian mode.

tExpand(sin(3 ϕ)) **[ENTER]**
$$4 \cdot \sin(\phi) \cdot (\cos(\phi))^2 - \sin(\phi)$$

tExpand(cos($\alpha - \beta$)) **[ENTER]**
$$\cos(\alpha) \cdot \cos(\beta) + \sin(\alpha) \cdot \sin(\beta)$$

Text CATALOG

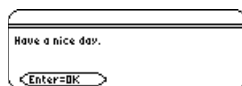
Text *promptString*

Displays the character string *promptString* dialog box.

If used as part of a **Dialog...EndDialog** 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 828.

timeCnv() CATALOG

timeCnv(*seconds*) \Rightarrow *list*

Converts seconds to units of time that can be more easily understood for evaluation. The list is in {*days,hours,minutes,seconds*} format.

Note: See also **checkTmr()** and **startTmr()**.

timeCnv(152442117) {1764 9 1 57}

Title CATALOG

Title *titleString*, [*Lb*]

Creates the title of a pull-down menu or dialog box when used inside a **ToolBar** or **Custom** construct, or a **Dialog...EndDialog** block.

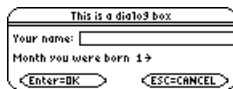
Note: *Lb* is only valid in the **ToolBar** construct. When present, it allows the menu choice to branch to a specified label inside the program.

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
:EndDialog
:

```



tmpCnv() CATALOG

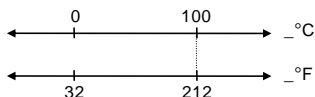
tmpCnv(*expression1*_*tempUnit1*, _*tempUnit2*)
 => *expression*_*tempUnit2*

Converts a temperature value specified by *expression1* from one unit to another. Valid temperature units are:

_°C Celsius
 _°F Fahrenheit
 _°K Kelvin
 _°R Rankine

└ For °, press [2nd] [°].
 For _, press [2nd] [_].

For example, 100_°C converts to 212_°F:



To convert a temperature range, use **ΔtmpCnv()** instead.

```

tmpCnv(100_°C, _°F) [ENTER]      212. °_F
tmpCnv(32_°F, _°C) [ENTER]      0. °_C
tmpCnv(0_°C, _°K) [ENTER]      273.15 °_K
tmpCnv(0_°F, _°R) [ENTER]      459.67 °_R

```

Note: To select temperature units from a menu, press:

◀ [UNITS]

ΔtmpCnv() CATALOG

ΔtmpCnv(*expression1*_*tempUnit1*, _*tempUnit2*)
 => *expression*_*tempUnit2*

Converts a temperature range (the difference between two temperature values) specified by *expression1* from one unit to another. Valid temperature units are:

_°C Celsius
 _°F Fahrenheit
 _°K Kelvin
 _°R Rankine

└ For °, press [2nd] [°].
 For _, press [2nd] [_].

1_°C and 1_°K have the same magnitude, as do 1_°F and 1_°R. However, 1_°C is 9/5 as large as 1_°F.

To get Δ, you can press ◀ [Δ] [↑] [D] (or [2nd] [CHAR] 15).

```

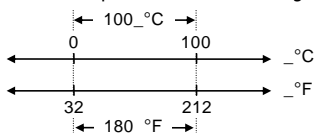
ΔtmpCnv(100_°C, _°F) [ENTER]      180. °_F
ΔtmpCnv(180_°F, _°C) [ENTER]      100. °_C
ΔtmpCnv(100_°C, _°K) [ENTER]      100. °_K
ΔtmpCnv(100_°F, _°R) [ENTER]      100. °_R
ΔtmpCnv(1_°C, _°F) [ENTER]      1.8 °_F

```

Note: To select temperature units from a menu, press:

◀ [UNITS]

For example, a 100_°C range (from 0_°C to 100_°C) is equivalent to a 180_°F range:



To convert a particular temperature value instead of a range, use **tmpCnv()**.

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 *errnum* 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** and **PassErr**.

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**

{0.2.3.4.3.4.6} → L2 **ENTER**

TwoVar L1,L2 **ENTER**

ShowStat **ENTER**



{0 1 2 ...}

{0 2 3 ...}

Done

Unarchiv CATALOG

Unarchiv *var1* [, *var2*] [, *var3*] ...

Moves the specified variables from the user data archive memory to RAM.

You can access an archived variable the same as you would a variable in RAM. However, you cannot delete, rename, or store to an archived variable because it is locked automatically.

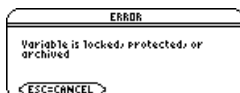
To archive variables, use **Archive**.

10 → arctest **ENTER**

Archive arctest **ENTER**

5* arctest **ENTER**

15 → arctest **ENTER**



ESC

Unarchiv arctest **ENTER**

15 → arctest **ENTER**

10

Done

50

Done

15

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**

$$\begin{bmatrix} \frac{\sqrt{14}}{14} \\ 14 \\ \frac{\sqrt{14}}{7} \\ 3 \cdot \sqrt{14} \\ 14 \end{bmatrix}$$

Unlock CATALOG

Unlock *var1*, *var2*] [, *var3*] ...

Unlocks the specified variables.

Note: The variables can be locked using the **Lock** command.

variance() MATH/Statistics menu

variance(*list* $\{$, *freqlist* $\}$) \Rightarrow *expression*

Returns the variance of *list*.

Each *freqlist* element counts the number of consecutive occurrences of the corresponding element in *list*.

Note: *list* must contain at least two elements.

$$\text{variance}(\{a,b,c\}) \text{ [ENTER]} = \frac{a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2}{3}$$

$$\text{variance}(\{1,2,5, \cdot 6,3, \cdot 2\}) \text{ [ENTER]} = 31/2$$

$$\text{variance}(\{1,3,5\}, \{4,6,2\}) \text{ [ENTER]} = 68/33$$

variance(*matrix1* $\{$, *freqmatrix* $\}$) \Rightarrow *matrix*

Returns a row vector containing the variance of each column in *matrix1*.

Each *freqmatrix* element counts the number of consecutive occurrences of the corresponding element in *matrix1*.

Note: *matrix1* must contain at least two rows.

$$\text{variance}([1,2,5; \cdot 3,0,1; \cdot 5, \cdot 7,3]) \text{ [ENTER]} = [4.75 \ 1.03 \ 4]$$

$$\text{variance}([-1,1,2,2; 3,4,5,1; \cdot 2,3,4,3], [6,3; 2,4; 5,1]) \text{ [ENTER]} = [3.91731, 2.08411]$$

when() CATALOG

when(*condition*, *trueResult* $\{$, *falseResult* $\}$, [*unknownResult*]) \Rightarrow *expression*

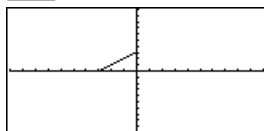
Returns *trueResult*, *falseResult*, or *unknownResult*, depending on whether *condition* is true, false, or unknown. Returns the input if there are too few arguments to specify the appropriate result.

Omit both *falseResult* and *unknownResult* to make an expression defined only in the region where *condition* is true.

Use an undef *falseResult* to define an expression that graphs only on an interval.

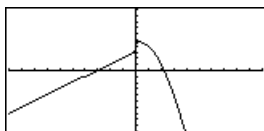
$$\text{when}(x < 0, x+3) | x=5 \text{ [ENTER]} \quad \text{when}(x < 0, 3+x)$$

ClrGraph [ENTER]
Graph when($x \geq -\pi$ 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]

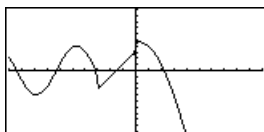


Nest **when()** to define expressions that have more than two pieces.

[CALC HOME]

ClrGraph [ENTER]
Graph when($x < 0, \text{when}(x < -\pi, 4 * \sin(x), 2x+3), 5-x^2$) [ENTER]

Done



when() is helpful for defining recursive functions.

```

when(n>0,n*factorial(n-1),1)
→ factorial(n) [ENTER]
factorial(3) [ENTER]
3! [ENTER]

```

Done
6
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",te
:
:
```

"With" See ↓ page 912.

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**.

```

true xor true [ENTER]           false
(5>3) xor (3>5) [ENTER]       true

```

integer1 **xor** *integer2* ⇒ *integer*

Compares two real integers bit-by-bit using an **xor** operation. Internally, both integers are converted to signed, 32-bit binary numbers. When corresponding bits are compared, the result is 1 if either bit (but not both) is 1; the result is 0 if both bits are 0 or both bits are 1. The returned value represents the bit results, and is displayed according to the Base mode.

You can enter the integers in any number base. For a binary or hexadecimal entry, you must use the 0b or 0h prefix, respectively. Without a prefix, integers are treated as decimal (base 10).

If you enter a decimal integer that is too large for a signed, 32-bit binary form, a symmetric modulo operation is used to bring the value into the appropriate range.

Note: See **or**.

In Hex base mode:

```

0h7AC36 xor 0h3D5F [ENTER]     0h79169
└─ Important: Zero, not the letter O.

```

In Bin base mode:

```

0b100101 xor 0b100 [ENTER]     0b100001

```

Note: A binary entry can have up to 32 digits (not counting the 0b prefix). A hexadecimal entry can have up to 8 digits.

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*) \Rightarrow *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), var)**.

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*.

Note: See also **cSolve()**, **cZeros()**, and **solve()**.

zeros($a \cdot x^2 + b \cdot x + c$, *x*) [ENTER]

$$\left\{ \frac{-\sqrt{b^2 - 4 \cdot a \cdot c} + b}{2 \cdot a}, \frac{\sqrt{b^2 - 4 \cdot a \cdot c} + b}{2 \cdot a} \right\}$$

$a \cdot x^2 + b \cdot x + c | x = \text{ans}(1)[2]$ [ENTER]

0

exact(zeros($a \cdot (e^x + x)$
($\text{sign}(x) - 1$), *x*)) [ENTER]

{}

exact(solve($a \cdot (e^x + x)$
($\text{sign}(x) - 1) = 0$, *x*)) [ENTER]

$e^x + x = 0$ or $x > 0$ or $a = 0$

zeros({*expression1, expression2, {varOrGuess1, varOrGuess2, ... }*}) \Rightarrow *matrix*

Returns candidate real zeros of the simultaneous algebraic *expressions*, where each *varOrGuess* specifies an unknown whose value you seek.

Optionally, you can specify an initial guess for a variable. Each *varOrGuess* must have the form:

variable

– or –

variable = *real or non-real number*

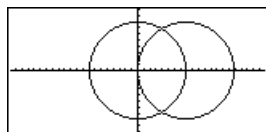
For example, *x* is valid and so is *x*=3.

If all of the expressions are polynomials and if you do NOT specify any initial guesses, **zeros()** uses the lexical Gröbner/Buchberger elimination method to attempt to determine **all** real zeros.

For example, suppose you have a circle of radius *r* at the origin and another circle of radius *r* centered where the first circle crosses the positive *x*-axis. Use **zeros()** to find the intersections.

As illustrated by *r* in the example to the right, simultaneous *polynomial* expressions can have extra variables that have no values, but represent given numeric values that could be substituted later.

Each row of the resulting matrix represents an alternate zero, with the components ordered the same as the *varOrGuess* list. To extract a row, index the matrix by [*row*].



zeros({ $x^2 + y^2 - r^2$,
($x - r$) $^2 + y^2 - r^2$ }, {*x, y*}) [ENTER]

$$\begin{bmatrix} r & \sqrt{3} \cdot r \\ 2 & 2 \\ r & -\sqrt{3} \cdot r \\ 2 & 2 \end{bmatrix}$$

Extract row 2:

ans(1)[2] [ENTER]

$$\begin{bmatrix} r & -\sqrt{3} \cdot r \\ 2 & 2 \end{bmatrix}$$

You can also (or instead) include unknowns that do not appear in the expressions. For example, you can include z as an unknown to extend the previous example to two parallel intersecting cylinders of radius r . The cylinder zeros illustrate how families of zeros might contain arbitrary constants in the form $@k$, where k is an integer suffix from 1 through 255. The suffix resets to 1 when you use **ClrHome** or **⏏** 8:Clear Home.

For polynomial systems, computation time or memory exhaustion may depend strongly on the order in which you list unknowns. If your initial choice exhausts memory or your patience, try rearranging the variables in the expressions and/or *varOrGuess* list.

If you do not include any guesses and if any expression is non-polynomial in any variable but all expressions are linear in the unknowns, **zeros()** uses Gaussian elimination to attempt to determine all real zeros.

If a system is neither polynomial in all of its variables nor linear in its unknowns, **zeros()** determines at most one zero using an approximate iterative method. To do so, the number of unknowns must equal the number of expressions, and all other variables in the expressions must simplify to numbers.

Each unknown starts at its guessed value if there is one; otherwise, it starts at 0.0.

Use guesses to seek additional zeros one by one. For convergence, a guess may have to be rather close to a zero.

zeros($\{x^2+y^2-r^2,$
 $(x-r)^2+y^2-r^2\},\{x,y,z\}$) **ENTER**

$$\begin{bmatrix} r & \sqrt{3}\cdot r \\ 2 & 2 \end{bmatrix} @1$$

$$\begin{bmatrix} r & -\sqrt{3}\cdot r \\ 2 & 2 \end{bmatrix} @1$$

zeros($\{x+e^z\cdot y-1,x-y-\sin(z)\},\{x,y,z\}$)
ENTER

$$\left[\frac{e\cdot\sin(z)+1}{e+1} \quad \frac{-(\sin(z)-1)}{e+1} \right]$$

zeros($\{e^z\cdot y-1,-y-\sin(z)\},\{y,z\}$) **ENTER**

$$[-.041\dots \quad 3.183\dots]$$

zeros($\{e^z\cdot y-1,-y-\sin(z)\},\{y,z=2\pi\}$) **ENTER**

$$[.001\dots \quad 6.281\dots]$$

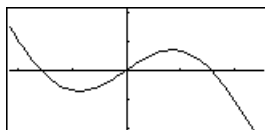
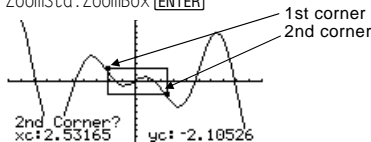
ZoomBox CATALOG

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}

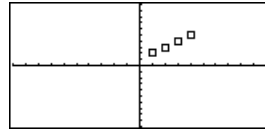
{2,3,4,5} → L2 **[ENTER]**

{2 3 4 5}

newPlot 1,1,L1,L2 **[ENTER]**

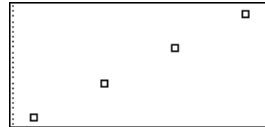
Done

ZoomStd **[ENTER]**



[♦] **[CALC HOME]**

ZoomData **[ENTER]**



ZoomDec CATALOG

ZoomDec

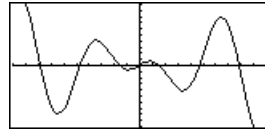
Adjusts the viewing window so that Δx and $\Delta y = 0.1$ and displays the Graph screen with the origin centered on the screen.

In function graphing mode:

$1.25x * \cos(x) \rightarrow y1(x)$ **[ENTER]**

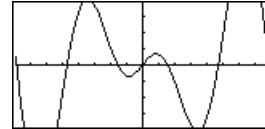
Done

ZoomStd **[ENTER]**



[♦] **[CALC HOME]**

ZoomDec **[ENTER]**



ZoomFit CATALOG

ZoomFit

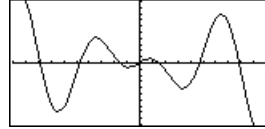
Displays the Graph screen, and calculates the necessary window dimensions for the dependent variables to view all the picture for the current independent variable settings.

In function graphing mode:

$1.25x * \cos(x) \rightarrow y1(x)$ [ENTER]

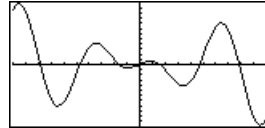
Done

ZoomStd [ENTER]



[♦] [CALC 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.

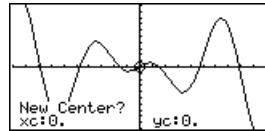
The magnitude of the zoom is dependent on the Zoom factors xFact and yFact. In 3D Graph mode, the magnitude is dependent on xFact, yFact, and zFact.

In function graphing mode:

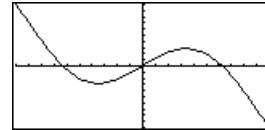
$1.25x * \cos(x) \rightarrow y1(x)$ [ENTER]

Done

ZoomStd:ZoomIn [ENTER]



[ENTER]



ZoomInt CATALOG

ZoomInt

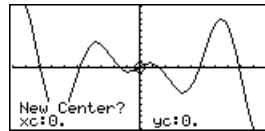
Displays the Graph screen, lets you set a center point for the zoom, and adjusts the window directions.

In function graphing mode:

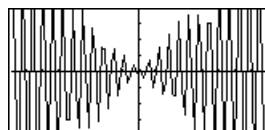
$1.25x * \cos(x) \rightarrow y1(x)$ [ENTER]

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.

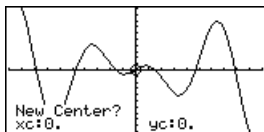
The magnitude of the zoom is dependent on the Zoom factors xFact and yFact. In 3D Graph mode, the magnitude is dependent on xFact, yFact, and zFact.

In function graphing mode:

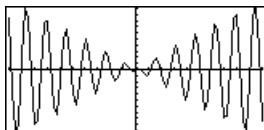
$1.25x * \cos(x) \rightarrow y1(x)$ **ENTER**

ZoomStd:ZoomOut **ENTER**

Done



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.

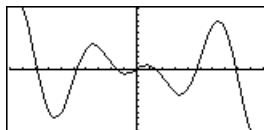
In 3D Graph mode, **ZoomSqr** lengthens the shortest two axes to be the same as the longest axis.

In function graphing mode:

$1.25x * \cos(x) \rightarrow y1(x)$ **ENTER**

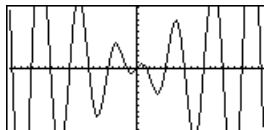
ZoomStd **ENTER**

Done



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:

eyeθ°=20, eyeφ°=70, eyeψ°=0

x: [-10, 10, 14], y: [-10, 10, 14],

z: [-10, 10], ncontour=5

Differential equations graphing:

t: [0, 10, .1, 0], x: [-1, 10, 1], y: [-10, 10, 1],

ncurves=0, Estep=1, diftol=.001, fldres=14,

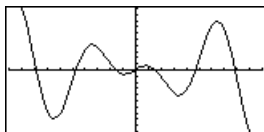
dtime=0

In function graphing mode:

1.25x*cos(x) → y1(x) **ENTER**

ZoomStd **ENTER**

Done



ZoomSto CATALOG

ZoomSto

Stores the current Window settings in the Zoom memory. You can use **ZoomRcl** to restore the settings.

ZoomTrig CATALOG

ZoomTrig

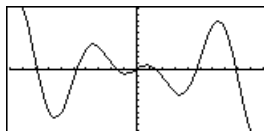
Displays the Graph screen, sets Δx to $\pi/24$, and x_{scl} to $\pi/2$, centers the origin, sets the y settings to [-4, 4, .5], and updates the viewing window.

In function graphing mode:

1.25x*cos(x) → y1(x) **ENTER**

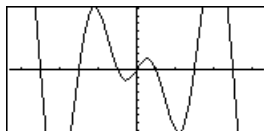
ZoomStd **ENTER**

Done



◀ [CALC HOME]

ZoomTrig **ENTER**



+ (add)

+ key

$expression1 + expression2 \Rightarrow expression$

Returns the sum of $expression1$ and $expression2$.

56 **ENTER**

ans(1)+4 **ENTER**

ans(1)+4 **ENTER**

ans(1)+4 **ENTER**

ans(1)+4 **ENTER**

56

60

64

68

72

$list1 + list2 \Rightarrow list$	$\{22, \pi, \pi/2\} \rightarrow L1$ <input type="button" value="ENTER"/>	$\{22 \pi \pi/2\}$
$matrix1 + matrix2 \Rightarrow matrix$	$\{10.5, \pi/2\} \rightarrow L2$ <input type="button" value="ENTER"/>	$\{10 \ 5 \ \pi/2\}$
Returns a list (or matrix) containing the sums of corresponding elements in <i>list1</i> and <i>list2</i> (or <i>matrix1</i> and <i>matrix2</i>).	$L1+L2$ <input type="button" value="ENTER"/>	$\{32 \ \pi+5 \ \pi\}$
Dimensions of the arguments must be equal.	$ans(1)+\{\pi, -5, -\pi\}$ <input type="button" value="ENTER"/>	$\{\pi+32 \ \pi \ 0\}$
	$[a,b;c,d]+[1,0;0,1]$ <input type="button" value="ENTER"/>	$\begin{bmatrix} a+1 & b \\ c & d+1 \end{bmatrix}$

$expression + list1 \Rightarrow list$	$15+\{10,15,20\}$ <input type="button" value="ENTER"/>	$\{25 \ 30 \ 35\}$
$list1 + expression \Rightarrow list$	$\{10,15,20\}+15$ <input type="button" value="ENTER"/>	$\{25 \ 30 \ 35\}$
Returns a list containing the sums of <i>expression</i> and each element in <i>list1</i> .		

$expression + matrix1 \Rightarrow matrix$	$20+[1,2;3,4]$ <input type="button" value="ENTER"/>	
$matrix1 + expression \Rightarrow matrix$		
Returns a matrix with <i>expression</i> added to each element on the diagonal of <i>matrix1</i> . <i>matrix1</i> must be square.		$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$

Note: Use **.+** (dot plus) to add an expression to each element.

- (subtract) key

$expression1 - expression2 \Rightarrow expression$	$6-2$ <input type="button" value="ENTER"/>	4
Returns <i>expression1</i> minus <i>expression2</i> .	$\pi - \pi/6$ <input type="button" value="ENTER"/>	$\frac{5 \cdot \pi}{6}$

$list1 - list2 \Rightarrow list$	$\{22, \pi, \pi/2\} - \{10.5, \pi/2\}$ <input type="button" value="ENTER"/>	
$matrix1 - matrix2 \Rightarrow matrix$		$\{12 \ \pi-5 \ 0\}$
Subtracts each element in <i>list2</i> (or <i>matrix2</i>) from the corresponding element in <i>list1</i> (or <i>matrix1</i>), and returns the results.	$[3,4] - [1,2]$ <input type="button" value="ENTER"/>	$[2 \ 2]$
Dimensions of the arguments must be equal.		

$expression - list1 \Rightarrow list$	$15 - \{10,15,20\}$ <input type="button" value="ENTER"/>	$\{5 \ 0 \ -5\}$
$list1 - expression \Rightarrow list$	$\{10,15,20\} - 15$ <input type="button" value="ENTER"/>	$\{-5 \ 0 \ 5\}$
Subtracts each <i>list1</i> element from <i>expression</i> or subtracts <i>expression</i> from each <i>list1</i> element, and returns a list of the results.		

$expression - matrix1 \Rightarrow matrix$	$20 - [1,2;3,4]$ <input type="button" value="ENTER"/>	
$matrix1 - expression \Rightarrow matrix$		$\begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$
<i>expression - matrix1</i> returns a matrix of <i>expression</i> times the identity matrix minus <i>matrix1</i> . <i>matrix1</i> 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.

*** (multiply) key**

$expression1 * expression2 \Rightarrow expression$	$2*3.45$ <input type="button" value="ENTER"/>	6.9
Returns the product of <i>expression1</i> and <i>expression2</i> .	$x*y*x$ <input type="button" value="ENTER"/>	$x^2 \cdot y$

$list1 * list2 \Rightarrow list$ {1.0,2,3}*{4,5,6} **ENTER** {4. 10 18}

Returns a list containing the products of the corresponding elements in $list1$ and $list2$. {2/a,3/2}*{a^2,b/3} **ENTER** {2·a $\frac{b}{2}$ }

Dimensions of the lists must be equal.

$matrix1 * matrix2 \Rightarrow matrix$ [1,2,3;4,5,6]*[a,d;b,e;c,f] **ENTER**

Returns the matrix product of $matrix1$ and $matrix2$.

The number of rows in $matrix1$ must equal the number of columns in $matrix2$.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix} = \begin{bmatrix} a+2\cdot b+3\cdot c & d+2\cdot e+3\cdot f \\ 4\cdot a+5\cdot b+6\cdot c & 4\cdot d+5\cdot e+6\cdot f \end{bmatrix}$$

$expression * list1 \Rightarrow list$ $\pi * \{4,5,6\}$ **ENTER** {4· π 5· π 6· π }

$list1 * expression \Rightarrow list$

Returns a list containing the products of $expression$ and each element in $list1$.

$expression * matrix1 \Rightarrow matrix$ [1,2;3,4]*.01 **ENTER** [.01 .02
.03 .04]

$matrix1 * expression \Rightarrow matrix$ $\lambda * \text{identity}(3)$ **ENTER** $\begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$

Returns a matrix containing the products of $expression$ and each element in $matrix1$.

Note: Use \cdot (dot multiply) to multiply an expression by each element.

/ (divide) **key**

$expression1 / expression2 \Rightarrow expression$ 2/3.45 **ENTER** .57971

Returns the quotient of $expression1$ divided by $expression2$. x^3/x **ENTER** x^2

$list1 / list2 \Rightarrow list$ {1.0,2,3}/{4,5,6} **ENTER** { .25 2/5 1/2 }

Returns a list containing the quotients of $list1$ divided by $list2$.

Dimensions of the lists must be equal.

$expression / list1 \Rightarrow list$ a/{3,a, \sqrt{a} } **ENTER** $\left\{ \frac{a}{3} \ 1 \ \sqrt{a} \right\}$

$list1 / expression \Rightarrow list$

Returns a list containing the quotients of $expression$ divided by $list1$ or $list1$ divided by $expression$. {a,b,c}/(a*b*c) **ENTER** $\left\{ \frac{1}{b\cdot c} \ \frac{1}{a\cdot c} \ \frac{1}{a\cdot b} \right\}$

$matrix1 / expression \Rightarrow matrix$ [a,b,c]/(a*b*c) **ENTER** $\left[\frac{1}{b\cdot c} \ \frac{1}{a\cdot c} \ \frac{1}{a\cdot b} \right]$

Returns a matrix containing the quotients of $matrix1$ / $expression$.

Note: Use \cdot (dot divide) to divide an expression by each element.

^ (power)  **key**

$expression1 \wedge expression2 \Rightarrow expression$
 $list1 \wedge list2 \Rightarrow list$

$4 \wedge 2$ [ENTER] 16

$\{a, 2, c\} \wedge \{1, b, 3\}$ [ENTER] $\{a \cdot 2^b \cdot c^3\}$

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.

$expression \wedge list1 \Rightarrow list$

$p \wedge \{a, 2, -3\}$ [ENTER] $\{p^a \ p^2 \ \frac{1}{p^3}\}$

Returns *expression* raised to the power of the elements in *list1*.

$list1 \wedge expression \Rightarrow list$

$\{1, 2, 3, 4\} \wedge -2$ [ENTER] $\{1 \ 1/4 \ 1/9 \ 1/16\}$

Returns the elements in *list1* raised to the power of *expression*.

$squareMatrix1 \wedge integer \Rightarrow matrix$

$[1, 2; 3, 4] \wedge 2$ [ENTER]
 $[1, 2; 3, 4] \wedge -1$ [ENTER]
 $[1, 2; 3, 4] \wedge -2$ [ENTER]

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.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2$	$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1}$	$\begin{bmatrix} -2 & 1 \\ 3/2 & -1/2 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2}$	$\begin{bmatrix} 11/2 & -5/2 \\ -15/4 & 7/4 \end{bmatrix}$

.+ (dot add)  **keys**


$matrix1 .+ matrix2 \Rightarrow matrix$
 $expression .+ matrix1 \Rightarrow matrix$

$[a, 2; b, 3] .+ [c, 4; 5, d]$ [ENTER]
 $x .+ [c, 4; 5, d]$ [ENTER]

$matrix1 .+ matrix2$ 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*.

$\begin{bmatrix} b & 3 \\ 5 & d \end{bmatrix} .+ \begin{bmatrix} 5 & d \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} a+c & 6 \\ b+5 & d+3 \end{bmatrix}$
$x .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} x+c & x+4 \\ x+5 & x+d \end{bmatrix}$

.- (dot subtr.)  **keys**



$matrix1 .- matrix2 \Rightarrow matrix$
 $expression .- matrix1 \Rightarrow matrix$

$[a, 2; b, 3] .- [c, 4; 5, d]$ [ENTER]
 $x .- [c, 4; 5, d]$ [ENTER]

$matrix1 .- matrix2$ 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*.


$\begin{bmatrix} b & 3 \\ 5 & d \end{bmatrix} .- \begin{bmatrix} d & 5 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} a-c & -2 \\ b-d & -2 \end{bmatrix}$
$x .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix}$	$\begin{bmatrix} x-c & x-4 \\ x-d & x-5 \end{bmatrix}$


.* (dot mult.)   **keys**

$matrix1 .* matrix2 \Rightarrow matrix$
 $expression .* matrix1 \Rightarrow matrix$



$matrix1 .* matrix2$ 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 $matrix1$.

$[a,2;b,3] .* [c,4;5,d]$ 

$x .* [a,b;c,d]$ 


$\begin{bmatrix} b & 3 \\ 5 & d \end{bmatrix} .* \begin{bmatrix} a & c \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} a \cdot c & 8 \\ 5 \cdot b & 3 \cdot d \end{bmatrix}$
$x .* \begin{bmatrix} a & b \\ c & d \end{bmatrix}$	$\begin{bmatrix} a \cdot x & b \cdot x \\ c \cdot x & d \cdot x \end{bmatrix}$


. / (dot divide)   **keys**

$matrix1 ./ matrix2 \Rightarrow matrix$
 $expression ./ matrix1 \Rightarrow matrix$

$matrix1 ./ matrix2$ 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 $matrix1$.

$[a,2;b,3] ./ [c,4;5,d]$ 

$x ./ [c,4;5,d]$ 


$\begin{bmatrix} b & 3 \\ 5 & d \end{bmatrix} ./ \begin{bmatrix} a & c \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} \frac{b}{a} & \frac{3}{c} \\ \frac{5}{5} & \frac{d}{d} \end{bmatrix}$
$x ./ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} \frac{x}{c} & \frac{x}{4} \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$


.^ (dot power)   **keys**

$matrix1.^ matrix2 \Rightarrow matrix$
 $expression.^ matrix1 \Rightarrow matrix$

$matrix1.^ matrix2$ 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 $expression$.

$[a,2;b,3].^ [c,4;5,d]$ 

$x.^ [c,4;5,d]$ 

$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} a^c & 16 \\ b^5 & 3^d \end{bmatrix}$
$x.^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} x^c & x^4 \\ x^5 & x^d \end{bmatrix}$

- (negate)  **key and MATH/Base menu**

$-expression1 \Rightarrow expression$
 $-list1 \Rightarrow list$
 $-matrix1 \Rightarrow matrix$


Returns the negation of the argument.

For a list or matrix, returns all the elements negated.


If $expression1$ is a binary or hexadecimal integer, the negation gives the two's complement.

-2.43 

-2.43

$-\{ -1.0,4,1.2E 19\}$ 

$\{1 -4 -1.2E 19\}$


$-a * -b$ 

$a \cdot b$

In Bin base mode:

$0b100101$  

37

$-0b100101$ 

Important: Zero, not the letter O.

$0b11111111111111111111111111111111011011$

$ans(1)$  

-37

Note: To type \blacktriangleright , press $[2nd]$ \blacktriangleright .

% (percent) CHAR/Punctuation menu $expression1 \% \Rightarrow expression$ $list1 \% \Rightarrow list$ $matrix1 \% \Rightarrow matrix$ 13%

.13

{1, 10, 100}%

{.01 .1 1.}

Returns $\frac{argument}{100}$.

For a list or matrix, returns a list or matrix with each element divided by 100.

= (equal) key $expression1 = expression2 \Rightarrow Boolean\ expression$ $list1 = list2 \Rightarrow Boolean\ list$ $matrix1 = matrix2 \Rightarrow Boolean\ matrix$ 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.

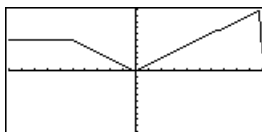
For lists and matrices, returns comparisons element by element.

Example function listing using math test symbols: =, \neq , <, \leq , >, \geq

```

:g(x)
:Func
:If x $\leq$ -5 Then
:  Return 5
: ElseIf x>-5 and x<0 Then
:  Return -x
: ElseIf x $\geq$ 0 and x $\neq$ 10 Then
:  Return x
: ElseIf x=10 Then
:  Return 3
: EndIf
:EndFunc

```

Graph $g(x)$  **\neq key** $expression1 \neq expression2 \Rightarrow Boolean\ expression$ $list1 \neq list2 \Rightarrow Boolean\ list$ $matrix1 \neq matrix2 \Rightarrow Boolean\ matrix$ 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 "=" (equal) example.

< **2nd [<] key**

$expression1 < expression2 \Rightarrow Boolean\ expression$
 $list1 < list2 \Rightarrow Boolean\ list$
 $matrix1 < matrix2 \Rightarrow Boolean\ matrix$

See "=" (equal) example.

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.

≤ **♦ 0 key**

$expression1 \leq expression2 \Rightarrow Boolean\ expression$
 $list1 \leq list2 \Rightarrow Boolean\ list$
 $matrix1 \leq matrix2 \Rightarrow Boolean\ matrix$

See "=" (equal) example.

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.

> **2nd [>] key**

$expression1 > expression2 \Rightarrow Boolean\ expression$
 $list1 > list2 \Rightarrow Boolean\ list$
 $matrix1 > matrix2 \Rightarrow Boolean\ matrix$

See "=" (equal) example.

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.

≥ **♦ . key**

$expression1 \geq expression2 \Rightarrow Boolean\ expression$
 $list1 \geq list2 \Rightarrow Boolean\ list$
 $matrix1 \geq matrix2 \Rightarrow Boolean\ matrix$

See "=" (equal) example.

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.

! (factorial) **2nd** **W** key

$expression!$ \Rightarrow $expression$
 $list!$ \Rightarrow $list$
 $matrix!$ \Rightarrow $matrix$

5! **[ENTER]** 120
 $\{5,4,3\}$! **[ENTER]** $\{120\ 24\ 6\}$
 $[1,2;3,4]$! **[ENTER]** $\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

Returns the factorial of the argument.

For a list or matrix, returns a list or matrix of factorials of the elements.

The TI-89 computes a numeric value for only non-negative whole-number values.

& (append) **2nd** **H** key

$string1$ & $string2$ \Rightarrow $string$

"Hello " & "Nick" **[ENTER]** "Hello Nick"

Returns a text string that is $string2$ appended to $string1$.

∫ (integrate) **2nd** **[f]** key

$\int(expression1, var[, lower][, upper]) \Rightarrow expression$
 $\int(list1, var[, order]) \Rightarrow list$
 $\int(matrix1, var[, order]) \Rightarrow matrix$

Returns the integral of $expression1$ with respect to the variable var from $lower$ to $upper$.

$\int(x^2, x, a, b)$ **[ENTER]** $\frac{b^3}{3} - \frac{a^3}{3}$

Returns an anti-derivative if $lower$ and $upper$ are omitted. A symbolic constant of integration such as C is omitted.

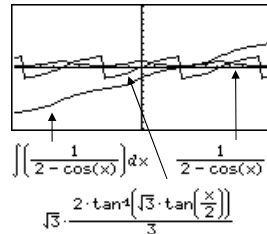
$\int(x^2, x)$ **[ENTER]** $\frac{x^3}{3}$

However, $lower$ is added as a constant of integration if only $upper$ is omitted.

$\int(a * x^2, x, c)$ **[ENTER]** $\frac{a * x^3}{3} + c$

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.

$\int(1/(2 - \cos(x)), x) \rightarrow tmp(x)$ **[ENTER]**
 ClrGraph:Graph tmp(x):Graph
 1/(2 - cos(x)):Graph $\sqrt{3}$
 $(2 \tan^{-1}(\sqrt{3} \cdot \tan(x/2)))/3$ **[ENTER]**



$\int()$ returns itself for pieces of $expression1$ that it cannot determine as an explicit finite combination of its built-in functions and operators.

$\int(b * e^{(-x^2)} + a/(x^2 + a^2), x)$ **[ENTER]**

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.

$$\int \left[b \cdot e^{-x^2} + \frac{a}{x^2 + a^2} \right] dx = b \cdot \int (e^{-x^2}) dx + \tan^{-1} \left(\frac{x}{a} \right)$$

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.

$\int()$ can be nested to do multiple integrals. Integration limits can depend on integration variables outside them.

Note: See also $nInt()$.

$$\int(e^{-(x^2)}.x.^{-1}.1) \text{ [ENTER]} \quad 1.493\dots$$

$$\int(\int(1n(x+y).y.0,x).x.0,a) \text{ [ENTER]}$$

$$\int_0^a \int_0^x 1n(x+y)dydx = \frac{a^2 \cdot 1n(a)}{2} + a^2 \cdot (1n(2) - 3/4)$$

$\sqrt{()}$ (square root) $\text{[2nd][}\sqrt{\text{] key}$

$\sqrt{(expression)}$ \Rightarrow *expression*
 $\sqrt{(list)}$ \Rightarrow *list*

Returns the square root of the argument.

For a list, returns the square roots of all the elements in *list*.

$$\sqrt{(4)} \text{ [ENTER]} \quad 2$$

$$\sqrt{(\{9.a.4\})} \text{ [ENTER]} \quad \{3 \sqrt{a} 2\}$$

$\Pi()$ (product) **MATH/Calculus menu**

$\Pi(expression1, var, low, high)$ \Rightarrow *expression*

Evaluates *expression1* for each value of *var* from *low* to *high*, and returns the product of the results.

$$\Pi(1/n,n,1,5) \text{ [ENTER]} \quad \frac{1}{120}$$

$$\Pi(k^2,k,1,n) \text{ [ENTER]} \quad (n!)^2$$

$$\Pi(\{1/n,n,2\},n,1,5) \text{ [ENTER]} \quad \left\{ \frac{1}{120} 120 32 \right\}$$

$\Pi(expression1, var, low, low-1)$ \Rightarrow 1

$$\Pi(k,k,4,3) \text{ [ENTER]} \quad 1$$

$\Pi(expression1, var, low, high)$ \Rightarrow $1/\Pi(expression1, var, high+1, low-1)$ if *high* < *low-1*

$$\Pi(1/k,k,4,1) \text{ [ENTER]} \quad 6$$

$$\Pi(1/k,k,4,1) * \Pi(1/k,k,2,4) \text{ [ENTER]} \quad 1/4$$

$\Sigma()$ (sum) **MATH/Calculus menu**

$\Sigma(expression1, var, low, high)$ \Rightarrow *expression*

Evaluates *expression1* for each value of *var* from *low* to *high*, and returns the sum of the results.

$$\Sigma(1/n,n,1,5) \text{ [ENTER]} \quad \frac{137}{60}$$

$$\Sigma(k^2,k,1,n) \text{ [ENTER]} \quad \frac{n \cdot (n+1) \cdot (2 \cdot n+1)}{6}$$

$$\Sigma(1/n^2,n,1,\infty) \text{ [ENTER]} \quad \frac{\pi^2}{6}$$

$\Sigma(expression1, var, low, low-1)$ \Rightarrow 0

$$\Sigma(k,k,4,3) \text{ [ENTER]} \quad 0$$

$\Sigma(expression1, var, low, high)$ \Rightarrow $-\Sigma(expression1, var, high+1, low-1)$ if *high* < *low-1*

$$\Sigma(k,k,4,1) \text{ [ENTER]} \quad -5$$

$$\Sigma(k,k,4,1) + \Sigma(k,k,2,4) \text{ [ENTER]} \quad 4$$

(indirection) CATALOG# *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
:
:

```

G (gradian) MATH/Angle menu

$expression1^{\circ} \Rightarrow expression$
 $list1^{\circ} \Rightarrow list$
 $matrix1^{\circ} \Rightarrow matrix$

This function gives you a way to use a gradian angle while in the Degree or Radian modes.

In Radian angle mode, multiplies *expression* by $\pi/200$. In Degree angle mode, multiplies *expression* 1 by $g/100$. In Gradian mode, returns *expression1* unchanged.

In Degree, Gradian or Radian mode:

$\cos(50^{\circ})$ **[ENTER]** $\frac{\sqrt{2}}{2}$
 $\cos(\{0,100^{\circ},200^{\circ}\})$ **[ENTER]** {1.0,-1}

r (radian) MATH/Angle menu

$expression1^r \Rightarrow expression$
 $list1^r \Rightarrow list$
 $matrix1^r \Rightarrow matrix$

In Degree angle mode, multiplies *expression1* by $180/\pi$. In Radian angle mode, returns *expression1* unchanged. In Gradian mode, multiplies *expression1* by $200/\pi$.

This function gives you a way to use a radian angle while in Degree and Gradian mode.

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, Gradian or Radian angle mode:

$\cos((\pi/4)^r)$ **[ENTER]** $\frac{\sqrt{2}}{2}$
 $\cos(\{0^r,(\pi/12)^r,-\pi^r\})$ **[ENTER]**
 $\{1 \frac{(\sqrt{3}+1) \cdot \sqrt{2}}{4} - 1\}$

o (degree) **[2nd] [°] key**

$expression^{\circ} \Rightarrow value$
 $list1^{\circ} \Rightarrow list$
 $matrix1^{\circ} \Rightarrow matrix$

In Radian angle mode, multiplies *expression* by $\pi/180$. In Degree angle mode, returns *expression* unchanged. In Gradian angle mode, multiplies *expression1* by $10/9$.

This function gives you a way to use a degree angle while in Gradian and Radian mode.

In Degree, Gradian or Radian angle mode:

$\cos(45^{\circ})$ **[ENTER]** $\frac{\sqrt{2}}{2}$
 $\cos(\{0,\pi/4,90^{\circ},30,12^{\circ}\})$ **[♦] [ENTER]**
 $\{1 .707... 0 .864...\}$

∠ (angle) **[2nd] [∠] key**

[radius, ∠θ_angle] ⇒ vector (polar input)
 [radius, ∠θ_angle, Z_coordinate] ⇒ vector
 (cylindrical input)
 [radius, ∠θ_angle, ∠φ_angle] ⇒ vector
 (spherical input)

[5. ∠60°, ∠45°] **[ENTER]**

In Radian mode and vector format set to:

Returns coordinates as a vector depending on the Vector Format mode setting: rectangular, cylindrical, or spherical.

$\left[\begin{array}{l} 5 \angle 60^\circ \angle 45^\circ \\ \frac{5\sqrt{2}}{4} \frac{5\sqrt{6}}{4} \frac{5\sqrt{2}}{2} \end{array} \right]$	rectangular
$\left[\begin{array}{l} 5 \angle 60^\circ \angle 45^\circ \\ \frac{5\sqrt{2}}{2} \angle \frac{\pi}{3} \frac{5\sqrt{2}}{2} \end{array} \right]$	cylindrical
$\left[\begin{array}{l} 5 \angle 60^\circ \angle 45^\circ \\ 5 \angle \frac{\pi}{3} \angle \frac{\pi}{4} \end{array} \right]$	spherical

(magnitude ∠ angle) ⇒ complexValue (polar input)

In Radian angle mode and Rectangular complex format mode:

Enters a complex value in (r∠θ) polar form. The angle is interpreted according to the current Angle mode setting.

5+3i (10∠π/4) **[ENTER]**

$$5 - 5\sqrt{2} + (3 - 5\sqrt{2}) \cdot i - 2.071... - 4.071... \cdot i$$

[◀] [ENTER]

° , ' , " **[2nd] [°] key (°), [2nd] ['] key ('), [2nd] ["] key (")**

dd° mm' ss.ss" ⇒ expression

In Degree angle mode:

dd A positive or negative number
 mm A non-negative number
 ss.ss A non-negative number

25°13'17.5" **[ENTER]**

25.221...

25°30' **[ENTER]**

51/2

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.

' (prime) **[2nd] ['] key**

variable'
 variable''

deSolve(y''=y^(-1/2) and y(0)=0 and y'(0)=0.t.y) **[ENTER]**

Enters a prime symbol in a differential equation. A single prime symbol denotes a 1st-order differential equation, two prime symbols denote a 2nd-order, etc.

$$\frac{2 \cdot y^{3/4}}{3} = t$$

_ (underscore) **[2nd] [-] key**

expression_unit

3_m_ft **[ENTER]**

9.842..._ft

Designates the units for an expression. All unit names must begin with an underscore.

Note: To type **▶**, press **[2nd] [▶]**.

You can use pre-defined units or create your own units. For a list of pre-defined units, refer to the module about constants and measurement units. You can press:

[▶] [UNITS] to select units from a menu, or you can type the unit names directly.

variable_

When *variable* has no value, it is treated as though it represents a complex number. By default, without the *_*, the variable is treated as real.

If *variable* has a value, the *_* is ignored and *variable* retains its original data type.

Note: You can store a complex number to a variable without using *_*. However, for best results in calculations such as **cSolve()** and **cZeros()**, the *_* is recommended.

Assuming *z* is undefined:

real(<i>z</i>)	[ENTER]	<i>z</i>
real(<i>z_</i>)	[ENTER]	real(<i>z_</i>)
imag(<i>z</i>)	[ENTER]	0
imag(<i>z_</i>)	[ENTER]	imag(<i>z_</i>)

► (convert) **[2nd] [►] key**

expression_unit1 ► *_unit2* ⇒ *expression_unit2* 3_m ► *_ft* **[ENTER]** 9.842... • *_ft*

Converts an expression from one unit to another. The units must be in the same category.

The *_* underscore character designates the units. For a list of valid pre-defined units, refer to the module about constants and measurement units. You can press **[►] [UNITS]** to select units from a menu, or you can type the unit names directly.

To get the *_* underscore when typing units directly, press **[2nd] [-]**

Note: The ► conversion operator does not handle temperature units. Use **tmpCnv()** and **ΔtmpCnv()** instead.

10^() CATALOG

10^ (*expression1*) ⇒ *expression* 10^(1.5) **[ENTER]** 31.622...

10^ (*list1*) ⇒ *list* 10^{0. - 2. 2. a} **[ENTER]** {1 $\frac{1}{100}$ 100 10^a}

Returns 10 raised to the power of the argument.

For a list, returns 10 raised to the power of the elements in *list1*.

10^ (*squareMatrix1*) ⇒ *squareMatrix* 10^{[1.5,3;4.2,1;6,-2,1]} **[ENTER]**

Returns 10 raised to the power of *squareMatrix1*. This is *not* the same as calculating 10 raised to the power of each element. For information about the calculation method, refer to **cos()**.

1.143...e7	8.171...e6	6.675...e6
9.956...e6	7.115...e6	5.813...e6
7.652...e6	5.469...e6	4.468...e6

squareMatrix1 must be diagonalizable. The result always contains floating-point numbers.

x⁻¹ CATALOG (^(^-1))

expression1 **x⁻¹** ⇒ *expression* 3.1⁻¹ **[ENTER]** .322581

list1 **x⁻¹** ⇒ *list* {a, 4, - .1, x-2}⁻¹ **[ENTER]** { $\frac{1}{a}$ $\frac{1}{4}$ - 10. $\frac{1}{x-2}$ }

Returns the reciprocal of the argument.

For a list, returns the reciprocals of the elements in *list1*.

$\text{squareMatrix1} \mathbf{x}^{-1} \Rightarrow \text{squareMatrix}$

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**

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} = \begin{bmatrix} -2 & 1 \\ 3/2 & -1/2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ a & 4 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{-2}{a-2} & \frac{1}{a-2} \\ \frac{a}{2 \cdot (a-2)} & \frac{-1}{2 \cdot (a-2)} \end{bmatrix}$$

| ("with") **[2nd] [1] key**

$\text{expression} | \text{Boolean expression1 [and Boolean expression2]...[and Boolean expressionM]}$

$x+1 | x=3$ **ENTER** 4

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".

$x+y | x=\sin(y)$ **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. $\text{expression} | \text{variable} = \text{value}$ will substitute value for every occurrence of variable in expression .

$x^3 - 2x + 7 \rightarrow f(x)$ **ENTER** Done

$f(x) | x=\sqrt{3}$ **ENTER** $\sqrt{3} + 7$

$(\sin(x))^2 + 2\sin(x) - 6 | \sin(x)=d$ **ENTER** $d^2 + 2d - 6$

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.

$\text{solve}(x^2 - 1 = 0, x) | x > 0$ and $x < 2$ **ENTER** $x = 1$

$\sqrt{x} * \sqrt{1/x} | x > 0$ **ENTER** 1

$\sqrt{x} * \sqrt{1/x}$ **ENTER** $\sqrt{\frac{1}{x}} \cdot \sqrt{x}$

Exclusions use the "not equals" (\neq or \neq) 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.

$\text{solve}(x^2 - 1 = 0, x) | x \neq 1$ **ENTER** $x = -1$

→ (store) **[STO] key**

$\text{expression} \rightarrow \text{var}$
 $\text{list} \rightarrow \text{var}$
 $\text{matrix} \rightarrow \text{var}$
 $\text{expression} \rightarrow \text{fun_name}(\text{parameter1}, \dots)$
 $\text{list} \rightarrow \text{fun_name}(\text{parameter1}, \dots)$
 $\text{matrix} \rightarrow \text{fun_name}(\text{parameter1}, \dots)$

$\pi/4 \rightarrow \text{myvar}$ **ENTER** $\frac{\pi}{4}$

$2\cos(x) \rightarrow Y1(x)$ **ENTER** Done

$\{1, 2, 3, 4\} \rightarrow \text{Lst5}$ **ENTER** $\{1 \ 2 \ 3 \ 4\}$

If variable var does not exist, creates var and initializes it to expression , list , or matrix .

$\{1, 2, 3; 4.5, 6\} \rightarrow \text{MatG}$ **ENTER** $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

If var already exists and if it is not locked or protected, replaces its contents with expression , list , or matrix .

$"\text{Hello}" \rightarrow \text{str1}$ **ENTER** "Hello"

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.

0b, 0h

0 **[alpha]** **[B]** **keys**

0 **[alpha]** **[H]** **keys**

0b *binaryNumber*

0h *hexadecimalNumber*

In Dec base mode:

0b10+0hF+10 **[ENTER]**

27

In Bin base mode:

0b10+0hF+10 **[ENTER]**

0b11011

In Hex base mode:

0b10+0hF+10 **[ENTER]**

0h1B

Denotes a binary or hexadecimal number, respectively. To enter a binary or hex number, you must enter the 0b or 0h prefix regardless of the Base mode. Without a prefix, a number is treated as decimal (base 10).

Results are displayed according to the Base mode.

Appendix B: Technical Reference

This section contains a comprehensive list of TI-89 Titanium / Voyage™ 200 error messages and character codes. It also includes information about how certain TI-89 Titanium / Voyage™ 200 operations are calculated.

TI-89 Titanium / Voyage™ 200 Error Messages

This section 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.

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 <code>If a<b</code> 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 <i>expression1,expression2</i> and PtOn <i>list1,list2</i> are both valid, but PtOn <i>expression,list</i> is a mismatch.
60	Argument must be a Boolean expression or integer
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, <code>DelVar 12</code> is invalid because a number cannot be a variable name.
150	Argument must be an empty folder name
160	Argument must be an expression For example, <code>zeros(2x+3=0,x)</code> is invalid because the first argument is an equation.
161	ASAP or Exec string too long
163	Attribute (8-digit number) of object (8-digit number) not found
165	Batteries too low for sending or receiving Install new batteries before sending or receiving.
170	Bound For the interactive graph math functions like <code>2:Zero</code> , 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.
- 185 Checksum error**
- 190 Circular definition**
This message is displayed to avoid running out of memory during infinite replacement of variable values during simplification. For example, $a+1 \rightarrow a$, where a is an undefined variable, will cause this error.
- 200 Constraint expression invalid**
For example, $\text{solve}(3x^2-4=0, x) \mid x<0 \text{ or } x>5$ would produce this error message because the constraint is separated by “or” and not “and.”
- 205 Data is too big to save to a variable. Please use F6 Util to reduce the size.**
The size of the data in the editor exceeds the maximum size that can be saved in a variable. The F6 Util menu provides operations that can be used to reduce the size of the data.
- 210 Data type**
An argument is of the wrong data type.
- 220 Dependent limit**
A limit of integration is dependent on the integration variable. For example, $\int(x^2, x, 1, x)$ is not allowed.
- 225 Diff Eq setup**
- 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, $\text{ans}(100)$ is not valid because the argument for **ans()** must be in the range 1–99.
- 270 Duplicate variable name**
- 280 Else and Elself invalid outside of If..EndIf block**
- 290 EndTry is missing the matching Else statement**
- 295 Excessive iteration**
- 300 Expected 2 or 3-element list or matrix**
- 307 Flash application extension (function or program) not found**
- 308 Flash application not found**

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, $\text{nSolve}(3x^2 - 4 = 0, x)$ is a valid equation; however, $\text{nSolve}(3x^2 - 4, x)$ is not an equation, and $\text{nSolve}(3x^2 - y = 0, x)$ 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, $\text{solve}(3x^2 - 4, x)$ 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.

335 Graph functions $y_1(x)$... $y_{99}(x)$ not available in Diff Equations mode

345 Inconsistent units

350 Index out of range

360 Indirection string is not a valid variable name

380 Invalid ans()

390 Invalid assignment

400 Invalid assignment value

405 Invalid axes

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.

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 Solver 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 the *Calculator Home Screen* module.

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.
- 575 Invalid polar complex**
- 580 Invalid program reference**
Programs cannot be referenced within functions or expressions such as $1+p(x)$ where p is a program.
- 585 Invalid relocation data in ASM program**
The necessary relocation data in the ASM (Assembly) program is missing or corrupted.
- 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**
- 605 Invalid use of units**
- 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.
- 665 Matrix not diagonalizable**

- 670 Memory**
- 673** The calculation required more memory than was available at that time. If you get this error when you run a large program, you may need to break the program into separate, smaller programs or functions (where one program or function calls another).
- 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**
- 765 No functions selected**
- 780 No solution found**
- 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 $y_1(x)=x^2$, 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.
- 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.
- 855 Rand type functions not allowed in 3D graphing**
- 860 Recursion is limited to 255 calls deep**
- 870 Reserved name or system variable**
- 875 ROM-resident routine not available**
- 880 Sequence setup**
- 885 Signature error**
- 890 Singular matrix**

- 895 Slope fields need one selected function and are used for 1st-order equations only**
- 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.
- 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**
- 955 Too many undefined variables**
- 960 Undefined variable**
- 965 Unlicensed OS or Flash application**
- 970 Variable in use so references or changes are not allowed**
- 980 Variable is locked, protected, or archived**
- 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 may specify more zeros**
- Warning: May produce false equation**
- Warning: Expected finite real integrand**
- Warning: May not be fully simplified**
- Warning: More solutions may exist**
- Warning: May introduce false solutions**
- Warning: Operation may lose solutions**
- Warning: Requires & returns 32 bit value**
- Warning: Overflow replaced by ∞ or $^{-}\infty$**
- Warning: Questionable accuracy**
- Warning: Questionable solution**
- Warning: Solve may specify more zeros**
- Warning: Trig argument too big to reduce**

Note: Domain of result may be larger

Note: Domain of result may be smaller

TI-89 Titanium / Voyage™ 200 Modes

This section describes the modes of the TI-89 Titanium /Voyage™ 200 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
2:PARAMETRIC	x(t) and y(t) parametric equations
3:POLAR	r(θ) polar equations
4:SEQUENCE	u(n) sequences
5:3D	z(x,y) 3D equations
6:DIFF EQUATIONS	y'(t) differential equations

Note: If you use a split screen with Number of Graphs = 2, Graph 1 is for the top or left part of the screen and Graph 2 is for the bottom or right part.

Current Folder

Specifies the current folder. You can set up multiple folders with unique configurations of variables, graph databases, programs, etc.

Note: For detailed information about using folders, see *Calculator Home Screen*.

1:main	Default folder included with the TI-89 Titanium / Voyage™ 200.
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-89 Titanium / Voyage™ 200 retains decimal numbers with 14 significant digits. For display purposes, such numbers are rounded to a maximum of 12 significant digits.

1:FIX 0	Results are always displayed with the selected number of decimal places.
2:FIX 1	
...	
D:FIX 12	
E:FLOAT	The number of decimal places varies, depending on the result.

F:FLOAT 1	If the integer part has more than the selected number of digits, the result is rounded and displayed in scientific notation.
G:FLOAT 2	
...	
Q:FLOAT 12	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
3:GRADIAN

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: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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-89 Titanium / Voyage™ 200 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: $a+bi$
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 $r = 3$, $\theta = 45$, and $z = 2$.

3:SPHERICAL Coordinates are in terms of r, θ , and ϕ . For example, [3, ∠45, ∠90] represents $r = 3$, $\theta = 45$, and $\phi = 90$.

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 , $\pi/2$, or $\sqrt{((x-3)/x)}$

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 the *Operating the Calculator* module.

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.

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 Screen Ratio is available on the Voyage™ 200 only.)

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 \downarrow from the Page 2 mode screen or when you press $\boxed{\text{APPS}}$. 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 (Voyage 200 only)

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-89 Titanium / Voyage™ 200 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 the *Operating the Calculator* module.

Base

Lets you perform calculations by entering numbers in decimal, binary, or hexadecimal form.

1:DEC	Decimal numbers use 0 - 9 in the base 10 format
2:HEX	Hexadecimal numbers use 0 - 9 and A - F in the base 16 format.
3:BIN	Binary numbers use 0 and 1 in the base 2 format.

Unit System

Lets you enter a unit for values in an expression, such as 6_m * 4_m or 23_m/_s * 10_s, convert values from one unit to another within the same category, and create your own user-defined units.

1:SI	Select SI for the metric system of measurements
------	---

2:ENG/US	Select ENG/US for the non-metric system of measurements
----------	---

3:CUSTOM	Allows you to select custom defaults.
----------	---------------------------------------

Custom Units

Lets you select custom defaults. This mode is dimmed until you select Unit System, 3:CUSTOM.

Language

Lets you localize the TI-89 Titanium / Voyage™ 200 into one of several languages, depending on which language Flash applications are installed.

1:English	Default language included with the TI-89 Titanium / Voyage™ 200 operating system (OS).
-----------	--

2: — (language Flash applications)	Alternate languages are available only if the respective language Flash applications have been installed.
---------------------------------------	---

Apps Desktop

Lets you turn the display of the Apps desktop on or off.

ON	Displays the navigable Apps desktop. The Apps desktop appears when you: <ul style="list-style-type: none">• Press [APPS].• Turn the unit on after it has been turned off by pressing [2nd] [OFF].• Press [2nd] [QUIT] from an App that is displayed in full screen mode.
----	---

OFF	<ul style="list-style-type: none">• Does not display the navigable Apps desktop.• The unit defaults to the calculator Home screen.• The calculator Home screen displays when you press [2nd] [QUIT].• The APPLICATIONS menu displays when you press [APPS].
-----	--

TI-89 Titanium / Voyage™ 200 Character Codes

The **char()** function lets you refer to any character by its numeric character code. For example, to display ♦ on the Program I/O screen, use `Disp char(127)`. You can use **ord()** to find the numeric code of a character. For example, `ord("A")` returns 65.

1. SOH	38. &	76. L	113. q	148. ω	186. ρ	223. β
2. STX	39. '	77. M	114. r	149. Ε	187. »	224. à
3. ETX	40. (78. N	115. s	150. e	188. d	225. á
4. EOT	41.)	79. O	116. t	151. i	189. j	226. â
5. ENQ	42. *	80. P	117. u	152. r	190. ∞	227. ã
6. ACK	43. +	81. Q	118. v	153. T	191. ç	228. ä
7. BELL	44. ,	82. R	119. w	154. x̄	192. À	229. å
8. BS	45. -	83. S	120. x	155. ȳ	193. Á	230. æ
9. TAB	46. .	84. T	121. y	156. ≤	194. Â	231. ç
10. LF	47. /	85. U	122. z	157. ≠	195. Ã	232. è
11. ␣	48. 0	86. V	123. {	158. ≥	196. Ä	233. é
12. FF	49. 1	87. W	124.	159. ∠	197. Å	234. ê
13. CR	50. 2	88. X	125. }	160. ...	198. Æ	235. ë
14. 🔒	51. 3	89. Y	126. ~	161. ¡	199. Ç	236. ì
15. ✓	52. 4	90. Z	127. ♦	162. ¢	200. È	237. í
16. ▪	53. 5	91. [128. α	163. £	201. É	238. î
17. ◀	54. 6	92. \	129. β	164. ¤	202. Ê	239. ï
18. ▶	55. 7	93.]	130. Γ	165. ¥	203. Ë	240. ð
19. ▲	56. 8	94. ^	131. γ	166. ¦	204. Ì	241. ñ
20. ▼	57. 9	95. _	132. Δ	167. §	205. Í	242. ò
21. ←	58. :	96. `	133. δ	168. √	206. Î	243. ó
22. →	59. ;	97. a	134. ε	169. ●	207. Ĭ	244. ô
23. ↑	60. <	98. b	135. ζ	170. Ⓐ	208. Đ	245. õ
24. ↓	61. =	99. c	136. θ	171. «	209. Ñ	246. ö
25. ◀	62. >	100. d	137. λ	172. ¬	210. Ò	247. ÷
26. ▶	63. ?	101. e	138. ξ	173. -	211. Ó	248. ø
27. †	64. @	102. f	139. Π	174. ®	212. Ô	249. ù
28. ∪	65. A	103. g	140. π	175. -	213. Õ	250. ú
29. ∩	66. B	104. h	141. ρ	176. °	214. Ö	251. û
30. ⊂	67. C	105. i	142. Σ	177. ±	215. ×	252. ü
31. ∈	68. D	106. j	143. σ	178. ²	216. Ø	253. ý
32. SPACE	69. E	107. k	144. τ	179. ³	217. Ù	254. þ
33. !	70. F	108. l	145. φ	180. ⁻¹	218. Ú	255. ÿ
34. "	71. G	109. m	146. φ	181. μ	219. Û	
35. #	72. H	110. n	147. Ω	182. ¶	220. Ü	
36. \$	73. I	111. o		183. •	221. Ý	
37. %	74. J	112. p		184. +	222. Þ	
	75. K			185. ¹		

TI-89 Titanium Key Codes

The **getKey()** function returns a value 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] [F6]** will return a value of 273.

Table 1: Key Codes for Primary Keys

Key	Modifier									
	None		↑		[2nd]		◆		[alpha]	
	Assoc.	Value	Assoc.	Value	Assoc.	Value	Assoc.	Value	Assoc.	Value
[F1]	F1	268	F1	268	F1	268	Y=	8460	F1	268
[F2]	F2	269	F2	269	F2	269	Window	8461	F2	269
[F3]	F3	270	F3	270	F3	270	Graph	8462	F3	270
[F4]	F4	271	F4	271	F4	271	Tblset	8463	F4	271
[F5]	F5	272	F5	272	F5	272	Table	8464	F5	272
◆			Copy	24576	Cut	12288				
[alpha]					a-lock					
[ESC]	ESC	264	ESC	264	QUIT	4360	PASTE	8456	ESC	264
[APPS]	APPS	265	APPS	265	Switch	4361		8457	APPS	265
[HOME]	HOME	277	HOME	277	CUST	4373	HOME	277	Home	277
[MODE]	MODE	266	MODE	266	▶	18	_	95	MODE	266
[CATALOG]	CATLG	278	CATLG	278	<i>i</i>	151	∞	190	CATLG	278
[←]	BS	257	BS	257	INS	4353	DEL	8447	BS	257
[CLEAR]	CLEAR	263	CLEAR	263	CLEAR	263		8455		
[X]	x	120	X	88	LN	4184	e ^x	8280	x	120
[Y]	y	121	Y	89	SIN	4185	SIN ⁻¹	8281	y	121
[Z]	z	122	Z	90	COS	4186	COS ⁻¹	8282	z	122
[T]	t	116	T	84	TAN	4180	TAN ⁻¹	8276	t	116
[^]	^	94	^	94	π	140	θ	136	^	94
[]		124	F	70	°	176	Format d/b	8316	f	102
[(]	(40	B	66	{	123			b	98
[)])	41	C	67	}	125	●	169	c	99
[,]	,	44	D	68	[91		8236	d	100
[÷]	/	47	E	69]	93	!	33	e	101
[*]	*	42	J	74	√	4138	&	38	j	106
[-]	-	45	O	79	VAR-LNK	4141	Contr. -		o	111
[+]	+	43	U	85	CHAR	4139	Contr. +		u	117

Table 1: Key Codes for Primary Keys

Key	Modifier									
	None		↑		2nd		♦		alpha	
	Assoc.	Value	Assoc.	Value	Assoc.	Value	Assoc.	Value	Assoc	Value
ENTER	CR	13	CR	13	ENTRY	4109	Approx	8205	CR	13
STO▶	STO▶	258	P	80	RCL	4354	@	64	p	112
=	=	61	A	65	'	39	≠	157	a	97
EE	EE	149	K	75	∠	159	SYMB	8341	k	107
(-)	-	173	SPACE	32	ANS	4372		8365	SPACE	32
.	.	46	W	87	>	62	≥	158	w	119
0	0	48	V	86	<	60	≤	156	v	118
1	1	49	Q	81	"	34		8241	q	113
2	2	50	R	50	\	92		8242	r	114
3	3	51	S3	83	CUST	4147		8243	s	115
4	4	52	L	76	:	58		8244	l	108
5	5	53	M	77	MATH	4149		8245	m	109
6	6	54	N	78	MEM	4150		8246	n	110
7	7	55	G	71	∫	4151		8247	g	103
8	8	56	H	72	d	4152		8248	h	104
9	9	57	I	73	;	59		8249	i	105

Table 2: Arrow Keys (including diagonal movement)

Key	Normal	f	2nd	↖	alpha
←	338	16722	4434	8530	33106
→	340	16724	4436	8532	33108
↖	344	16728	4440	8536	33112
↗	337	16721	4433	8529	33105
← and ↗	339	16723	4435	8531	33107
↖ and →	342	16726	4438	8534	33110
↖ and ↗	345	16729	4441	8537	33113
↖ and ↘	348	16732	4444	8540	33116

Table 3: Greek Letters (prefixed by **↖** **[]**)

Keys	Second modifier			
	alpha		f	
	Assoc.	Value	Assoc.	Value
[A]	α	128		
[B]	β	129		
[D]	δ	133	Δ	132
[E]	ϵ	134		
[F]	ϕ	145		
[G]	γ	131	Γ	130
[L]	λ	137		
[M]	μ	181		
[STO] [P]	π	140	Π	139
[R]	ρ	141		
[S]	σ	143	Σ	142
[T]	τ	144		
[W]	ω	148	Ω	147
[X]	ξ	138		
[Y]	ψ	146		
[Z]	ζ	135		

Voyage™ 200 Key Codes

The **getKey()** function returns a value 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 Codes for Primary Keys

Key	Modifier							
	None		1		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

Key	Modifier							
	None		↑		2nd		♦	
	Assoc.	Value	Assoc.	Value	Assoc.	Value	Assoc.	Value
[θ]	0	136	0	136	:	58		8328
[(-)]	-	173	-	173	ANS	4372		8365
[.]	.	46	.	46	>	62		8238
[0]	0	48	0	48	<	60		8240
[1]	1	49	1	49	E	149		8241
[2]	2	50	2	50	CATALOG	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	∫	4151		8247
[8]	8	56	8	56	<i>d</i>	4152		8248
[9]	9	57	9	57	x ⁻¹	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	j	106	J	74	∞	190		8266
K	k	107	K	75		124	KEY	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	UNITS	8272
Q	q	113	Q	81	?	63	CALCHOME	8273
R	r	114	R	82	@	64	GRAPH	8274

Key	Modifier							
	None		↑		2nd		◆	
	Assoc.	Value	Assoc.	Value	Assoc.	Value	Assoc.	Value
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			8282

Table 2: Arrow Keys (including diagonal movement)

Key	Normal	↑	2nd	↘	☞
↶	338	16722	4434	8530	33106
↷	340	16724	4436	8532	33108
↵	344	16728	4440	8536	33112
↶	337	16721	4433	8529	33105
↶ and ↷	339	16723	4435	8531	33107
↶ and ↷	342	16726	4438	8534	33110
↶ and ↷	345	16729	4441	8537	33113
↶ and ↷	348	16732	4444	8540	33116

Note: The Grab (☞) modifier only affects the arrow keys.

Table 3: Grave Accent Letters (prefixed by 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 Letters (prefixed by 2nd C)

Key	Assoc.	Normal	↑
C	ç	231	199

Table 5: Acute Accent Letters (prefixed by **[2nd]** E)

Key	Assoc.	Normal	[f]
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 Letters (prefixed by **[2nd]** G)

Key	Assoc.	Normal	[f]
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 Letters (prefixed by 2nd N)

Key	Assoc.	Normal	↑
N	ñ	241	209
O	õ	245	

Table 8: Caret Letters (prefixed by 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 Letters (prefixed by 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	

Entering Complex Numbers

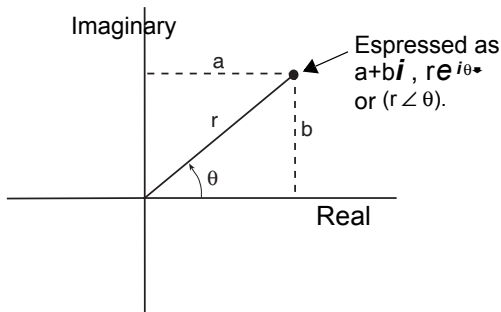
You can enter complex numbers in the polar form ($r \angle \theta$), where r is the magnitude and θ is the angle, or polar form $r e^{i\theta}$. You can also enter complex numbers in rectangular form $a+bi$.

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.

The point can be expressed in rectangular form or in either of two polar

The i symbol represents the imaginary number $\sqrt{-1}$.



As shown below, the form that you can enter depends on the current Angle mode.

You can use the form:	When the Angle mode setting is:
$a+bi$	Radian, Degree or Gradian
$r e^{i\theta}$	Radian only (In Degree or Gradian angle mode, this form causes a Domain error.)
$(r \angle \theta)$	Radian, Degree or Gradian

Use the following methods to enter a complex number.

To enter the:	Do this:
Rectangular form $a+bi$	Substitute the applicable values or variable names for a and b . a $\boxed{+}$ b $\boxed{2nd}$ $\boxed{[i]}$ Note: To get the i symbol, press $\boxed{2nd}$ $\boxed{[i]}$, do not simply type an alphabetic i . For example:



To enter the:	Do this:
---------------	----------

Polar form
 $re^{i\theta}$
 or
 $(r\angle\theta)$

Parentheses are required for the $(r\angle\theta)$ form.

Substitute the applicable values or variable names for r and θ , where θ is interpreted according to the Angle mode setting.

TI-89 Titanium:

α [R] \diamond [e^x] [2nd] [i] \diamond [θ]]
 or –
 ([α] [R] [2nd] [\angle] \diamond [θ]]

Important: Do not use the $re^{i\theta}$ polar form in Degree angle mode. It will cause a Domain error.

Note: To get the e symbol, press:

TI89 Titanium: \diamond [e^x].

Voyage™ 200: [2nd] [e^x]

Do not simply type an alphabetic e.

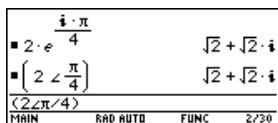
Tip: To get the \angle symbol, press [2nd] [\angle].

Tip: To enter θ in degrees for $(r\angle\theta)$, you can type a $^\circ$ symbol (such as 45°). To get the $^\circ$ symbol, press [2nd] [$^\circ$]. You should not use degrees or Gradian for $re^{i\theta}$.

Voyage™ 200:

R [2nd] [e^x] [2nd] [i] [θ]]
 or –
 (R [2nd] [i] [θ]]

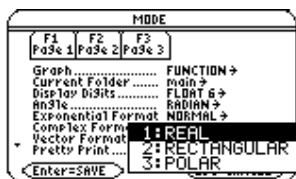
For example:



Results are shown in rectangular form, but you can select polar form.

Complex Format Mode for Displaying Results

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-89 Titanium / Voyage™ 200:**

REAL Will not display complex results unless you:

- Enter a complex number.

– or –

- Use a complex function such as **cFactor()**, **cSolve()**, or **cZeros()**.

If complex results are displayed, they will be shown in either $a+bi$ or $re^{i\theta}$ form.

Note: You can enter complex numbers in any form (or a mixture of all forms) depending on the Angle mode.

RECTANGULAR Displays complex results as $a+bi$.

POLAR Displays complex results as:

- $re^{i\theta}$ if the Angle mode = Radian

– or –

- $(r \angle \theta)$ if the Angle mode = Degree or Gradian

Using Complex Variables in Symbolic Calculations

Regardless of the Complex Format mode setting, variables that have no stored value and those that do not end with an underscore (_) are treated as real numbers. To perform complex symbolic analysis, you can use either of the following methods to set up a complex variable.

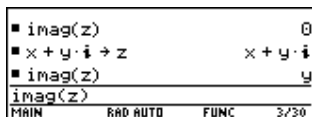
Method 1: Use an underscore _ (**TI-89 Titanium:** \square [_], **Voyage™ 200** \square [_]) as the last character in the variable name to designate a complex variable. For example:

$z_$ is treated as a complex variable if it does not have a stored value.



Method 2: Store an unreal value into any variable. For example:

$x+yi \rightarrow z$
Then z is treated as a complex variable.



Note: For best results in calculations such as **cSolve()** and **cZeros()**, use Method 1.

Complex Numbers and Degree Mode

Radian angle mode is recommended for complex number calculations. Internally, the TI-89 Titanium / Voyage™ 200 converts all entered trig values to radians, but it does not convert values for exponential, logarithmic, or hyperbolic functions.

In Degree angle mode, complex identities such as $e^{i(\theta)} = \cos(\theta) + i \sin(\theta)$ are not generally true because the values for \cos and \sin are converted to radians, while those for $e^{i(\)}$ are not. For example, $e^{i(45)} = \cos(45) + i \sin(45)$ is treated internally as $e^{i(\pi/4)} = \cos(\pi/4) + i \sin(\pi/4)$. Complex identities are always true in Radian angle mode.

Note: If you use Degree angle mode, you must make polar entries in the form $(r \angle \theta)$. In Degree or Gradian angle mode, an $re^{i\theta}$ entry causes an error.

Accuracy Information

To maximize accuracy, the TI-89 Titanium / Voyage™ 200 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 (x_{min} , x_{max} , y_{min} , y_{max} , 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

The Window variable x_{min} is the center of the leftmost pixel used, and x_{max} is the center of the rightmost pixel used. Δx is the distance between the centers of two horizontally adjacent pixels.

- Δx is calculated as $(x_{max} - x_{min}) / (\# \text{ of } x \text{ pixels} - 1)$.
- If Δx is entered from the Home screen or a program, x_{max} is calculated as $x_{min} + \Delta x * (\# \text{ of } x \text{ pixels} - 1)$.

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 Split Screens.

The Window variable y_{min} is the center of the bottom pixel used, and y_{max} is the center of the top pixel used. Δy is the distance between the centers of two vertically adjacent pixels.

- Δy is calculated as $(y_{max} - y_{min}) / (\# \text{ of } y \text{ pixels} - 1)$.
- If Δy is entered from the Home screen or a program, y_{max} is calculated as $y_{min} + \Delta y * (\# \text{ of } y \text{ pixels} - 1)$.

Cursor coordinates are displayed as eight characters (which may include a negative sign, decimal point, and exponent). The coordinate values (x_c , y_c , z_c , 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-89 Titanium / Voyage™ 200. 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

$y1(x)-y99(x)^*$	$y1'(t)-y99'(t)^*$	$y1-yi99^*$	$r1(\theta)-r99(\theta)^*$
$xt1(t)-xt99(t)^*$	$yt1(t)-yt99(t)^*$	$z1(x,y)-z99(x,y)^*$	$u1(n)-u99(n)^*$
$ui1-ui99^*$	xc	yc	zc
tc	rc	θc	nc
xfact	yfact	zfact	xmin
xmax	xscl	xgrid	ymin
ymax	yscl	ygrid	xres
Δx	Δy	zmin	zmax
zscl	eye θ	eye ϕ	eye ψ
ncontour	θ min	θ max	θ step
tmin	tmax	tstep	t0
tplot	ncurves	diftol	dtime
Estep	fldpic	fldres	nmin
nmax	plotStrt	plotStep	sysMath

Graph Zoom

zxmin	zxmax	zxsc1	zxgrid
zymin	zymax	zyscl	zygrid
zxres	$z\theta$ min	$z\theta$ max	$z\theta$ step
ztmin	ztmax	ztstep	zt0de
ztmaxde	ztstepde	ztplotde	zzmin
zzmax	zzscl	zeye θ	zeye ϕ
zeye ψ	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 Δ tbl tblInput

Data/Matrix

c1–c99 sysData*

Miscellaneous

main ok errornum

Solver

eqn* exp*

EOS (Equation Operating System) Hierarchy

This section describes the Equation Operating System (EOS™) that is used by the TI-89 Titanium / Voyage™ 200. 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 ($^{\circ}$, ', "), factorial (!), percentage (%), radian ($^{\text{r}}$), subscript ([]), transpose ($^{\text{T}}$)
5	Exponentiation, power operator (^)
6	Negation (-)
7	String concatenation (&)
8	Multiplication (*), division (/)
9	Addition (+), subtraction (-)
10	Equality relations: equal (=), not equal (\neq or \neq), less than (<), less than or equal (\leq or \leq), greater than (>), greater than or equal (\geq or \geq)
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-89 Titanium / Voyage™ 200 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 $10 \rightarrow r$ and $r \rightarrow s1$, then $\#s1=10$.

Post Operators

Post operators are operators that come directly after an argument, such as $5!$, 25% , or $60^\circ 15' 45''$. 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 $\boxed{-}$ 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.

Regression Formulas

This section describes how the statistical regressions are calculated.

Least-Squares Algorithm

Most of the regressions use non-linear recursive least-squares techniques to optimize the following cost function, which is the sum of the squares of the residual errors:

$$J = \sum_{i=1}^N [\text{residualExpression}]^2$$

where: *residualExpression* is in terms of x_i and y_i

x_i is the independent variable list

y_i is the dependent variable list

N is the dimension of the lists

This technique attempts to recursively estimate the constants in the model expression to make J as small as possible.

For example, $y = a \sin(bx + c) + d$ is the model equation for **SinReg**. So its residual expression is:

$$a \sin(bx_i + c) + d - y_i$$

For **SinReg**, therefore, the least-squares algorithm finds the constants a , b , c , and d that minimize the function:

$$J = \sum_{i=1}^N [a \sin(bx_i + c) + d - y_i]^2$$

Regressions

Regression	Description
CubicReg	Uses the least-squares algorithm to fit the third-order polynomial: $y = ax^3 + bx^2 + cx + d$ For four data points, the equation is a polynomial fit; for five or more, it is a polynomial regression. At least four data points are required.
ExpReg	Uses the least-squares algorithm and transformed values x and $\ln(y)$ to fit the model equation: $y = ab^x$
LinReg	Uses the least-squares algorithm to fit the model equation: $y = ax + b$ where a is the slope and b is the y-intercept.

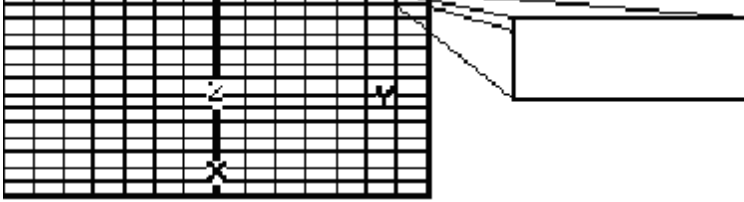
LnReg	<p>Uses the least-squares algorithm and transformed values $\ln(x)$ and y to fit the model equation:</p> $y = a + b \ln(x)$
Logistic	<p>Uses the least-squares algorithm to fit the model equation:</p> $y = a / (1 + b * e^{(c * x)}) + d$
MedMed	<p>Uses the median-median line (resistant line) technique to calculate summary points $x_1, y_1, x_2, y_2, x_3,$ and $y_3,$ and fits the model equation:</p> $y = ax + b$ <p>where a is the slope and b is the y-intercept.</p>
PowerReg	<p>Uses the least-squares algorithm and transformed values $\ln(x)$ and $\ln(y)$ to fit the model equation:</p> $y = ax^b$
QuadReg	<p>Uses the least-squares algorithm to fit the second-order polynomial:</p> $y = ax^2 + bx + c$ <p>For three data points, the equation is a polynomial fit; for four or more, it is a polynomial regression. At least three data points are required.</p>
QuartReg	<p>Uses the least-squares algorithm to fit the fourth-order polynomial:</p> $y = ax^4 + bx^3 + cx^2 + dx + e$ <p>For five data points, the equation is a polynomial fit; for six or more, it is a polynomial regression. At least five data points are required.</p>
SinReg	<p>Uses the least-squares algorithm to fit the model equation:</p> $y = a \sin(bx + c) + d$

Contour Levels and Implicit Plot Algorithm

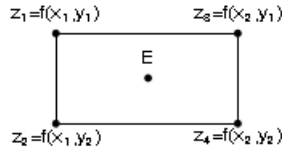
Contours are calculated and plotted by the following method. An implicit plot is the same as a contour, except that an implicit plot is for the $z=0$ contour only.

Algorithm

Based on your x and y Window variables, the distance between x_{min} and x_{max} and between y_{min} and y_{max} is divided into a number of grid lines specified by x_{grid} and y_{grid} . These grid lines intersect to form a series of rectangles.



For each rectangle, the equation is evaluated at each of the four corners (also called vertices or grid points) and an average value (E) is calculated:

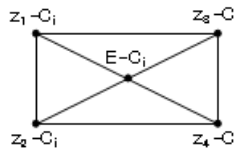


$$E = \frac{z_1 + z_2 + z_3 + z_4}{4}$$

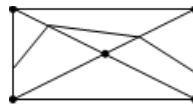
The E value is treated as the value of the equation at the center of the rectangle.

For each specified contour value (C_i)

- At each of the five points shown to the right, the difference between the point's z value and the contour value is calculated.



- A sign change between any two adjacent points implies that a contour crosses the line that joins those two points. Linear interpolation is used to approximate where the zero crosses the line.
- Within the rectangle, any zero crossings are connected with straight lines.
- This process is repeated for each contour value.



Each rectangle in the grid is treated similarly.

Runge-Kutta Method

For Runge-Kutta integrations of ordinary differential equations, the TI-89 Titanium / Voyage™ 200 uses the Bogacki-Shampine 3(2) formula as found in the journal *Applied Math Letters*, 2 (1989), pp. 1–9.

Bogacki-Shampine 3(2) Formula

The Bogacki-Shampine 3(2) formula provides a result of 3rd-order accuracy and an error estimate based on an embedded 2nd-order formula. For a problem of the form:

$$y' = f(x, y)$$

and a given step size h , the Bogacki-Shampine formula can be written:

$$F_1 = f(x_n, y_n)$$

$$F_2 = f\left(x_n + h \frac{1}{2}, y_n + h \frac{1}{2} F_1\right)$$

$$F_3 = f\left(x_n + h \frac{3}{4}, y_n + h \frac{3}{4} F_2\right)$$

$$y_{n+1} = y_n + h\left(\frac{2}{9} F_1 + \frac{1}{3} F_2 + \frac{4}{9} F_3\right)$$

$$x_{n+1} = x_n + h$$

$$F_4 = f(x_{n+1}, y_{n+1})$$

$$errest = h\left(\frac{5}{72} F_1 - \frac{1}{12} F_2 - \frac{1}{9} F_3 + \frac{1}{8} F_4\right)$$

The error estimate *errest* is used to control the step size automatically. For a thorough discussion of how this can be done, refer to *Numerical Solution of Ordinary Differential Equations* by L. F. Shampine (New York: Chapman & Hall, 1994).

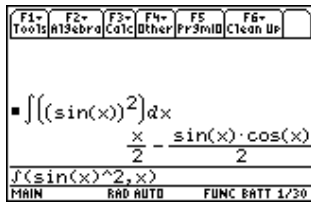
The TI-89 Titanium / Voyage™ 200 software does not adjust the step size to land on particular output points. Rather, it takes the biggest steps that it can (based on the error tolerance *diftol*) and obtains results for $x_n \leq x \leq x_{n+1}$ using the cubic interpolating polynomial passing through the point (x_n, y_n) with slope F_1 and through (x_{n+1}, y_{n+1}) with slope F_4 . The interpolant is efficient and provides results throughout the step that are just as accurate as the results at the ends of the step.

Battery Information

The TI-89 Titanium / Voyage™ 200 uses two types of batteries: four alkaline batteries, and a lithium battery as a backup for retaining memory while you change the alkaline batteries.

When to Replace the Batteries

As the alkaline 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 alkaline 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 text (BATT), you must replace the alkaline batteries immediately.



BATT indicator

To avoid loss of data, do not remove the lithium battery unless four fresh alkaline batteries are installed. Replace the lithium backup battery about every three or four years.

Note: To avoid loss of information stored in memory, the TI-89 Titanium / Voyage 200 must be off. Do not remove the alkaline 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.

Battery Precautions

Take these precautions when replacing batteries:

- Do not leave batteries within the reach of children.
- Do not mix new and used batteries. Do not mix brands (or types within brands) of batteries.
- Do not mix rechargeable and non-rechargeable batteries.
- Install batteries according to polarity (+ and -) diagrams.
- Do not place non-rechargeable batteries in a battery recharger.
- Properly dispose of used batteries immediately.
- Do not incinerate or dismantle batteries.

Replacing the Alkaline Batteries in the TI-89 Titanium

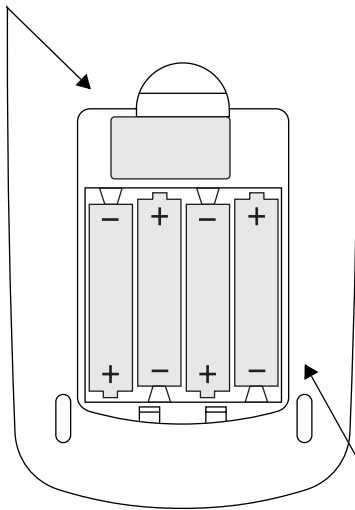
1. If the TI-89 Titanium is on, turn it off (press **2nd** [OFF]) to avoid loss of information stored in memory.
2. Slide the protective cover over the keyboard and place the device face down.
3. Push down on the battery cover latch, and then pull up to remove the cover.
4. Remove all four discharged AAA batteries.
5. Install four new AAA alkaline batteries, arranged according to the polarity (+ and -) diagram inside the battery compartment.
6. Replace the battery cover by inserting the two prongs into the two slots at the bottom of the battery compartment, and then push the cover until the latch snaps closed.

Replacing the Lithium Battery in the TI-89 Titanium

To replace the lithium backup battery, remove the battery cover and unscrew the tiny screw holding the BACK UP BATTERY cover in place.

Remove the old battery and install a new WR44SW or 303 battery, positive (+) side up. Replace the cover and the screw.

Lithium battery cover



AAA alkaline batteries

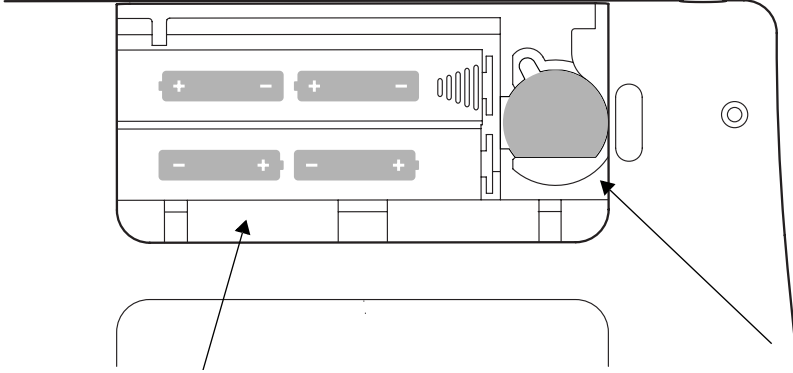
Replacing the Alkaline Batteries in the Voyage 200

1. If the Voyage™ 200 is on, turn it off (press **2nd** [OFF]) to avoid loss of information stored in memory.
2. Slide the protective cover over the keyboard and place the device face down.
3. Press the notched battery cover and slide it off, away from the device.
4. Remove all four discharged AAA batteries.
5. Install four new AAA alkaline batteries, arranged according to the polarity (+ and -) diagram inside the battery compartment.
6. Slide the battery cover onto the device, prong side first. Gently push the cover until the prongs snap into place.

Replacing the Lithium Battery in the Voyage 200

To replace the lithium backup battery, remove the battery cover. Insert a blunt object into the circular indentation next to the battery. Gently place a finger on the lithium battery and pry the battery out.

Slide in a new CR1616 or CR1620 battery, positive (+) side up. Press firmly to snap the new lithium battery into place.


















AAA alkaline battery compartment

Lithium battery

In Case of Difficulty

If you have difficulty operating the TI-89 Titanium / Voyage™ 200, the following suggestions may help you correct the problem.

If:	Suggested action:
You cannot see anything on the display.	Press \blacktriangle \oplus to darken or \blacktriangle \ominus to lighten the display contrast.
The BATT indicator is displayed.	Replace the batteries. If BATT is displayed in reverse text (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-89 Titanium / Voyage 200 is waiting for input; press ENTER .
An error message is displayed.	Refer to the list of error messages in this module. Press ESC to clear.
The TI-89 Titanium / Voyage 200 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.

If:	Suggested action:
<p>The TI-89 Titanium appears to be “locked up” and will not respond to keyboard input.</p>	<p>The following action clears RAM. This erases all data, programs, and user-defined variables, functions, or folders. Press and hold , , and . Then press and release .</p> <p>The following action clears RAM <i>and</i> Flash ROM. This erases all data, programs, user-defined variables, functions, folders, Flash applications, and the user data archive.</p> <ol style="list-style-type: none"> 1. Remove one of the four AAA batteries. 2. Press and hold  and  as you reinstall the battery. 3. Continue holding  and  for five seconds before releasing.
<p>The Voyage™ 200 appears to be “locked up” and will not respond to keyboard input.</p>	<p>The following action clears RAM. This erases all data, programs, and user-defined variables, functions, or folders. Press and hold  and . Then press and release .</p> <p>The following action clears RAM <i>and</i> Flash ROM. This erases all data, programs, user-defined variables, functions, folders, Flash applications, and the user data archive.</p> <ol style="list-style-type: none"> 1. Remove one of the four AAA batteries. 2. Press and hold  and  as you reinstall the battery. 3. Continue holding  and  for five seconds before releasing.

Appendix C: Programmer's Guide

The parameter/mode strings used in the `setMode()`, `getMode()`, `setGraph()`, and `setTable()` functions do not translate into other languages when used in a program. For example, when you write a program in the French Language mode then switch to the Italian Language mode, the program will produce an error. To avoid this error, you must substitute digits for the alpha characters. These digits operate in all languages. This appendix contains the digit substitutions for these strings.

The following examples show how to substitute digits in the `setMode()` function.

Example 1: A program using alpha parameter/mode strings:

```
setMode("Graph","Sequence")
```

Example 2: The same program, substituting digits for those strings:

```
setMode("1","4")
```

setMode() and getMode()

Parameter/Mode Setting	Strings
ALL	0
Graph	1
FUNCTION	1
PARAMETRIC	2
POLAR	3
SEQUENCE	4
3D	5
DIFF EQUATIONS	6
DisplayDigits	2
FIX 0	1
FIX 1	2
FIX 2	3
FIX 3	4
FIX 4	5
FIX 5	6
FIX 6	7
FIX 7	8
FIX 8	9

Parameter/Mode Setting	Strings
FIX 9	10
FIX 10	11
FIX 11	12
FIX 12	13
FLOAT	14
FLOAT 1	15
FLOAT 2	16
FLOAT 3	17
FLOAT 4	18
FLOAT 5	19
FLOAT 6	20
FLOAT 7	21
FLOAT 8	22
FLOAT 9	23
FLOAT 10	24
FLOAT 11	25
FLOAT 12	26
Angle	3
RADIAN	1

Parameter/Mode Setting	Strings
DEGREE	2
GRADIAN	3
Exponential Format	4
NORMAL	1
SCIENTIFIC	2
ENGINEERING	3
Complex Format	5
REAL	1
RECTANGULAR	2
POLAR	3
Vector Format	6
RECTANGULAR	1
CYLINDRICAL	2
SPHERICAL	3
Pretty Print	7
OFF	1
ON	2
SplitScreen	8
FULL	1

Parameter/Mode Setting	Strings
TOP-BOTTOM	2
LEFT-RIGHT	3
Split1App	9
(applications are not numbered)	
Split2App	10
(applications are not numbered)	
Number of Graphs	11
1	1
2	2
Parameter/Mode Setting	Strings
Graph 2	12
FUNCTION	1
PARAMETRIC	2
POLAR	3
SEQUENCE	4
3D	5
DIFF_EQUATIONS	6
Split Screen Ratio	13
1:1	1
1:2	2

Parameter/Mode Setting	Strings
2:1	3
Exact/Approx	14
AUTO	1
EXACT	2
APPROXIMATE	3
Base	15
DEC	1
HEX	2
BIN	3

setGraph()

Parameter/Mode Setting	Strings
Coordinates	1
RECT	1
POLAR	2
OFF	3
Graph Order	2
SEQ	1
SIMUL	2
Grid	3
OFF	1
ON	2
Axes	4
In 3D Mode:	
OFF	1
AXES	2
BOX	3
Not in 3D Mode:	
OFF	1

ON	2
Leading Cursor	5
OFF	1
ON	2
Labels	6
OFF	1
ON	1
Seq Axes	7
TIME	1
WEB	2
Custom	3
Solution Method	8
RK	1
EULER	2
Fields	9
SLPFLD	1
DIRFLD	2
FLDOFF	3
DE Axes	10
TIME	1

Y1-VS-Y2	2
T-VS-Y'	3
Y-VS-Y'	4
Y1-VS-Y2'	5
Y1'-VS-Y2'	6

XR Style	11
WIRE FRAME	1
HIDDEN SRUFACE	2
CONTOUR LEVELS	3
WIRE AND CONTOUR	4
IMPLICIT PLOT	5

setTable()

Parameter/Mode Setting	Strings
Graph <->Table	1
OFF	1
ON	2
Independent	2
AUTO	1
ASK	2
Axes	4

Appendix D: Service and Warranty Information

Texas Instruments Support and Service

For general information

Home Page:	education.ti.com education.ti.com
KnowledgeBase and e-mail inquires:	education.ti.com/support education.ti.com/support
Phone:	(800) TI-CARES; (800) 842-2737 For U.S., Canada, Mexico, Puerto Rico, and Virgin Islands only
International information:	education.ti.com/international education.ti.com/international

For technical support

KnowledgeBase and support by e-mail:	education.ti.com/support education.ti.com/support
Phone (not toll-free):	(972) 917-8324

For product (hardware) service

Customers in the U.S., Canada, Mexico, Puerto Rico and Virgin Islands: Always contact Texas Instruments Customer Support before returning a product for service.

All other customers: Refer to the leaflet enclosed with this product (hardware) or contact your local Texas Instruments retailer/distributor.

Texas Instruments (TI) Warranty Information

Customers in the U.S. and Canada Only

One-Year Limited Warranty for Commercial Electronic Product

This Texas Instruments (“TI”) electronic product warranty extends only to the original purchaser and user of the product.

Warranty Duration. This TI electronic product is warranted to the original purchaser for a period of one (1) year from the original purchase date.

Warranty Coverage. This TI electronic product is warranted against defective materials and construction. **THIS WARRANTY IS VOID IF THE PRODUCT HAS BEEN DAMAGED BY ACCIDENT OR UNREASONABLE USE, NEGLIGENCE, 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.

Except as expressly provided in the One-Year Limited Warranty for this product, Texas Instruments does not promise that facilities for the repair of this product or parts for the repair of this product will be available.

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 Other Customers

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.

Battery Precautions

Take these precautions when replacing batteries:

- Do not leave batteries within the reach of children.
- Do not mix new and used batteries. Do not mix brands (or types within brands) of batteries.
- Do not mix rechargeable and non-rechargeable batteries.
- Install batteries according to polarity (+ and –) diagrams.
- Do not place non-rechargeable batteries in a battery recharger.
- Properly dispose of used batteries immediately.
- Do not incinerate or dismantle batteries.

TI-89 Titanium Shortcut Keys

General

- ◆ [APPS] List of Flash applications
- ◆ [2nd] [⇄] Toggle between last two chosen applications or split screens
- ◆ [−], [◆] [+] Lighten or darken contrast
- ◆ [ENTER] Calculate approximate answer
- ◆ [↶], [◆] [↷] Move cursor to top or bottom (in editors)
- ◆ [↑] [↶], [↑] [↷] Scroll tall objects in history
- ◆ [↑] [↶], [↑] [↷] Highlight left or right from cursor
- ◆ [2nd] [↶], [2nd] [↷] Page up or page down (in editors)
- ◆ [2nd] [↶], [2nd] [↷] Move cursor far left or far right

On-screen Keyboard Map (◆ [EE])

Press [ESC] to exit the map.



The keyboard map displays shortcuts that are not marked on the keyboard. As shown below, press ◆ and then the applicable key.

- ◆ [≠] ≠
- ◆ [G] Access Greek letters (see next column)
- ◆ [S] ● (comment)
- ◆ [D] Copy graph coordinates to sysdata
- ◆ [!] ! (factorial)
- ◆ [F] Display FORMATS dialog box
- ◆ [1-6] Run programs kbdprgm1() through kbdprgm6()
- ◆ [X] & (append)
- ◆ [EE] On-screen keyboard map
- ◆ [STO] @
- ◆ [ON] Turn off unit so that it returns to current application the next time you turn it on
- ◆ [0] (zero) ≤
- ◆ [.] ≥
- ◆ [↶] Copy graph coordinates to Home screen history

Alpha Rules

- [alpha] Type one lowercase letter
- [f] Type one uppercase letter
- [2nd] [a-lock] Lowercase alpha lock
- [f] [alpha] Uppercase alpha lock
- [alpha] Exit alpha lock

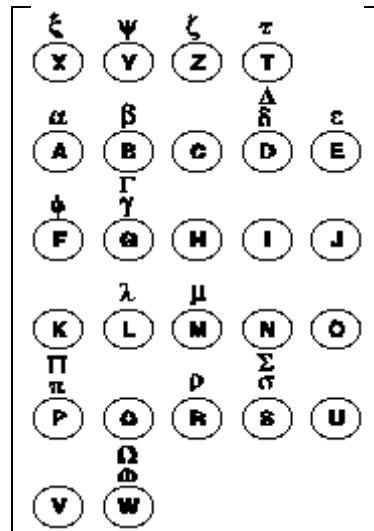
3D Graphing

- [↶], [↷], [↶], [↷] Animate graph
- [+], [−] Change animation speed
- X, Y, Z View along axis
- [0] Return to original view
- [f] Change graph format style
- [X] Expanded/normal view

Greek Letters

- ◆ [G] To access the Greek character set
- ◆ [G] [alpha] + letter To access lowercase Greek letters. Example: ◆ [G] [alpha] [W] displays ω
- ◆ [G] [f] + letter To access uppercase Greek letters. Example: ◆ [G] [f] [W] displays Ω

If you press a key combination that does not access a Greek letter, you get the normal letter for that key.



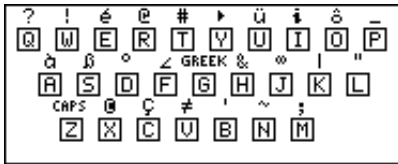
Voyage™ 200 Shortcut Keys

General

- ◆ [APPS] List of Flash applications (if desktop is off)
- ◆ [2nd] [⇧] Toggle between last two chosen applications or split screens
- ◆ D Copy graph coordinates to sysdata
- ◆ F Display FORMATS dialog box
- ◆ H Copy graph coordinates to Home screen history
- ◆ N Create new variable
- ◆ O Open existing variable
- ◆ S Save copy as
- ◆ [−], [⇧] [+] Lighten or darken contrast
- ◆ [ENTER] Calculate approximate answer
- ◆ [ON] Turn off unit so that it returns to current application the next time you turn it on
- ◆ 1 – 6 Run programs kbdprgm1() through kbdprgm6()

On-screen Keyboard Map (◆ [KEY])

Press [ESC] to exit the map.



See the table below for shortcuts that are not marked on the Voyage™ 200 keyboard. See the next column for accent marks and Greek letters.

- ◆ [2nd] X © (comment)
- ◆ [⇧] [⇧] ≠
- ◆ [⇧] [0] (zero) ≤
- ◆ [⇧] [⇧] ≥

Editing

- ◆ [⇧] [⇧] Move cursor to top
- ◆ [⇧] [⇧] Move cursor to bottom
- ◆ [2nd] [⇧] Move cursor to far left
- ◆ [2nd] [⇧] Move cursor to far right
- ◆ [⇧] [⇧], [⇧] [⇧] Scroll tall objects in history
- ◆ [2nd] [⇧], [2nd] [⇧] Page up and page down
- ◆ X Cut
- ◆ C Copy
- ◆ V Paste

3D Graphing

- ◆ [⇧] [⇧], [⇧] [⇧], [⇧] [⇧], [⇧] [⇧] Animate graph
- ◆ [⇧] [⇧], [⇧] [⇧] Change animation speed
- ◆ X, Y, Z View along axis
- ◆ [0] (zero) Return to original view
- ◆ F Change graph format style
- ◆ [X] Expanded/normal view

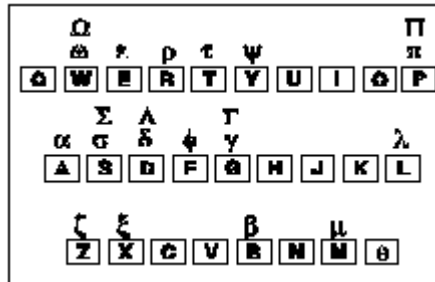
Accent Marks

- ◆ [2nd] A + letter à, è, ì, ò, ù, À, È, Ì, Ò, Ù
- ◆ [2nd] C + letter ç, Ç
- ◆ [2nd] E + letter á, é, í, ó, ú, ý, Á, É, Í, Ó, Ú, Ý
- ◆ [2nd] N + letter ã, ñ, õ, Ã, Ñ, Õ
- ◆ [2nd] O + letter â, ê, î, ô, û, Â, Ê, Î, Ô, Û
- ◆ [2nd] U + letter ä, ë, ï, ö, ü, ÿ, Ä, È, Ì, Ö, Ù

Greek Letters














































































- ◆ [2nd] G To access the Greek character set
- ◆ [2nd] G + letter To access lowercase Greek letters. Example: [2nd] G W displays ω
- ◆ [2nd] G [⇧] + letter To access uppercase Greek letters. Example: [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.



Keystroke Differences






There are certain differences in keystrokes using the TI-89 Titanium / Voyage™ 200 for various operations. The following table shows the keystrokes for major commands for the two calculators.

FUNCTION	 TI-89 Titanium	 Voyage 200
LETTERS		
One lowercase letter (a-s, u, v, w)	 A-S, U-W	A-S, U-W
One lowercase letter (t, x, y, z)	T, X, Y, Z	T, X, Y, Z
Several lowercase letters	 [a-lock]	
End several lowercase letters		
Several uppercase letters	 [a-lock]	 [CAPS]
End several uppercase letters		 [CAPS]
FUNCTION KEYS		
F6	 [F6]	
F7	 [F7]	
F8	 [F8]	
NAVIGATION		
Scroll tall objects up or down in history	   	   
Move cursor far left or far right on entry line	   	   
Diagonal movement	 and   and   and   and 	 and   and   and   and 
FUNCTIONS		
Display Home screen		 [CALC HOME]
Cut	 [CUT]	 X
Copy	 [COPY]	 C
Paste	 [PASTE]	 V
Catalog		 [CATALOG]
Display Units dialog box	 [UNITS]	 [UNITS]
Sin	 [SIN]	
Cos	 [COS]	
Tan	 [TAN]	
LN	 [LN]	
e^x	 [e^x]	 [e^x]
EE		 EE
FUNCTION	 TI-89 Titanium	 Voyage 200
SYMBOLS		
► (Conversion triangle)	 ►	 ►
_ (Underscore)	 [-]	 [-]

θ (Theta)	θ [θ]	θ
("With")		$\frac{1}{ }$
' (Prime)	' [$'$]	' [$'$]
° (Degree)	° [$^\circ$]	° [$^\circ$]
\sphericalangle (Angle)	\sphericalangle [\sphericalangle]	\sphericalangle [\sphericalangle]
Σ (Sigma)	Σ (Σ [Σ]
x[$_$] (Reciprocal)	\wedge^{-1}	x^{-1}
Space	α [$_$]	Space bar
HIDDEN SHORTCUTS		
Place data in sysdata variable	α ,	D
Greek characters	α or \uparrow	G or G \uparrow
Keyboard map	EE	[KEY]
Place data in Home screen history	(-)	H
Grave (à, è, ì, ò, ù)	[CHAR] 5	A a, e, i, o, u
Cedilla (ç)	[CHAR] 5 6	C c
Acute (á, é, í, ó, ú, ý)	[CHAR] 5	E a, e, i, o, u, y
Tilde (ã, ñ, õ)	[CHAR] 5 6	N a, n, o
Caret (â, ê, î, ô, û)	[CHAR] 5	O a, e, i, o, u
Umlaut (ä, ë, ï, ö, ü, ÿ)	[CHAR] 5	U a, e, i, o, u, y
? (Question mark)	[CHAR] 3	Q
β (Beta)	[CHAR] 5 6	S
# (Indirection)	[CHAR] 3	T
& (Append)	\times (times)	H
@ (Arbitrary)	STO	R
\neq (Not equal to symbol)	\neq	V
! (Factorial)	\div	W
Comment (Circle-C)] ●	X ©
New	F1 3	N
Open	F1 1	O
Save copy as	F1 2	S
Format dialog box	\uparrow	F

Index

Symbols

!, factorial	71, 904
→, store	596, 910
" , second notation	908
≠, ≠, not equal	604, 903
#, indirection	602, 906
$\sqrt{}$ (), square root	906
%, percent	902
&, append	602, 905
' , minute notation	908
' , prime	908
Σ (), sum	264, 906
\int (), integrate	77, 238, 240, 241, 245, 264, 265, 905
*, multiply	899
$\int f(x)dx$ (graph math tool)	329, 332
+, add	898
F1 – F8 (function keys)	
moving among toolbar menus	55
selecting categories	30, 32
selecting menus	49
uses	15
°, degree notation	760, 907, 908
-, negate	902
–, subtract	899
∠, angle	907
·, dot multiplication	902
·+, dot addition	901
–., dot subtraction	901
./, dot division	902
·^, dot power	902
/, divide	900
<, less than	604, 903
<<...>>, insufficient display memory	227
=, equal	604, 903
>, greater than	604, 904
 (negation key)	16
@, arbitrary integer	274, 275
Δ list (), list difference	834
Δ tbl, table increment	456
Δ tmpCnv (), temperature-range conversion	284, 888
Δx window variable	323
Δy window variable	323
^, power	901
_, underscore	908
≤, ≤, less than or equal	604, 904
 (subtraction key)	16
≥, ≥, greater than or equal	604, 904
●, comment	581, 911
, with	77, 80, 237, 247, 910
∞, infinity	276
 /  [DEL] (delete character)	19
 (hand modifier key)	
description	15
status	39

2nd (second modifier key)		►Dec, display as decimal integer	680, 802
description	14	►DMS, display as degree/minute/second	808
status	39	►Hex, display as hexadecimal	680, 826
2nd [MEM] (MEMORY)	19	►Polar, display as polar vector	850
2nd [EE] (exponent key)	16	►Rect, display as rectangular vector	860
2nd [►] (measurement conversions)	18	►Sphere, display as spherical vector	879
2nd [RCL] (recall)	19	↑ (shift modifier key)	
2nd [CATALOG] (Catalog)		description	14
commands	24	status	39
description	23	◆ (diamond modifier key)	
exiting	26	description	14
key command	19	status	39
2nd [CUSTOM] (Custom)		◆ F (FORMATS/GRAPH FORMATS)	18
description	55	◆ N (new file)	18
example	56	◆ O (open file)	18
key command	18	◆ S (SAVE COPY AS)	
2nd [CHAR] (Character)		description	18
description	50	dialog box	18
entering special characters	11	example	54
key command	19	STO► (store) key	19
selecting characters	11	⬅ (cursor keys)	
2nd [QUIT]		additional functions	16
calculator Home screen	26	entering commands	25
entering commands	24	function	16
exiting the split-screen mode	63	opening Apps	30
key command	19	selecting entry/answer pairs	28
turning off the calculator	8	used with the hand key	15
►, convert	281, 909	using the CHAR menu	12
►Bin, display as binary	680, 786	-, negate	153
►Cylind, display as cylindrical vector	800	Π(), product	264, 906
►DD, display as decimal angle	802		

θ max window variable	341
θ min window variable	341
θ step window variable	342
r, radian	907
T, transpose	883
(x window variable	937
x^{-1} , reciprocal	909
(y window variable	937

Numerics

009AppB, page = 574	945
0b, binary indicator	911
0h, hexadecimal indicator	911
$10^{\wedge}()$, power of ten	909
1Symbols	
#, indirection	940
\wedge , power	940
, with	940
3D (three-dimensional) mode	39
3D graphing	374–408
animation	99, 390
CONTOUR LEVELS	102, 394
HIDDEN SURFACE	102, 394
WIRE AND CONTOUR	102, 394
WIRE FRAME	102, 394

A

ABOUT screen	65
abs(), absolute value	763, 783
absolute value, abs()	763
accent marks	

CHAR menu	19
accented characters	655, 657, 658
accessory port	66
accuracy	937
add, +	898
Algebra menu	253, 256
algebra operations	779
All category	33
and (Boolean), and	247, 684, 783
and picture, AndPic	628
and, Boolean and	247, 605, 684
AndPic, and picture	784
Angle mode	185, 303, 919
angle mode	20
status	39
angle(), angle	784
angle, \angle	907
ans(), last answer	785
answer (last), ans()	217
APD (Automatic Power Down)	145
APD (Automatic Power Down) feature	
during calculation or program	9
turning on after	9
append, &	602, 905
APPLICATIONS menu	177
APPLICATIONS menu (<u>APPS</u>)	50, 58
approx(), approximate	785
Approximate mode	163, 186, 200, 239
approximate, approx()	254
Apps (calculator software applications)	
deleting	66

icon highlighted, last open	6	augment/concatenate, augment()	743
icons	5	Auto mode	163, 186, 200, 240, 921
names	30	AUTO mode status	39
opening	30, 58	Automatic Power Down (APD) feature	
shortcuts	34	during calculation or program	9
switching	63	in OS download mode	68
Apps desktop		turning on after	9
calculator Home screen and	26	automatic simplification	242
categories	30, 33	automatic tables	459
clock	42	auto-paste	28, 210, 218
date and time	44	avgRC(), average rate of change	786
initial startup	4, 5	axes (sequence), CUSTOM	362
mode	20, 40	Axes graph format	314, 414, 434, 435
parts of	6	Axes settings	385, 393
split-screen status	37		
turning off	40	B	
turning off the calculator	8	backspace (⌫)	19
arbitrary integer, @	274, 275	Base mode	20, 186, 922
Arc (graph math tool)	329, 335, 344, 352	BATT message	201, 945, 948
arc length, arcLen()	264	batteries	147, 201, 945, 946, 948
arccosine, $\cos^{-1}()$	793	precautions	68, 965
archive variables, Archive	596, 707, 708	prolonging life	9
Archive, archive variables	596, 707, 785	replacing	69
archiving variables	140	binary	
arcLen(), arc length	785	display, ►Bin	680, 786
arcsine, $\sin^{-1}()$	874	indicator, Ob	911
arctangent, $\tan^{-1}()$, arctangent	884	rotate, rotate()	685
assembly language	641, 644, 813	shift, shift()	686
asymptotes, faux, detecting	90	BldData, build data	439, 596, 787
augment(), augment/concatenate	743, 785	Bogacki-Shampine formula	944
		Boolean	

and, and	247, 605, 684, 783
exclusive or, xor	605, 684, 892
not, not	605, 683, 844
or, or	605, 684
Box Plot	556
build	
data, BldData	439, 596, 787
table, Table	883
web, Build Web	362
Build Web, build web	362, 364
build web, Build Web	364
BUSY	40
BUSY indicator	201, 573
Busy/Pause status	39

C

cables	64, 67, 715, 729, 733
Calc menu	263
calculator Home screen	
[2nd] [QUIT]	19
changing entry/answer pairs	29
custom menu	55
entering commands	24
function keys	15
key command	19
toolbar menus	49
turning off the calculator	8
calculator software applications (Apps)	6
icons	5
Calculator-Based Laboratory system	
connecting	67

Calculator-Based Laboratory. See CBL	
Calculator-Based Ranger See CBR	
Calculator-Based Ranger system	
connecting	67
calculus operations	779
Catalog ([2nd] [CATALOG])	
commands	24
description	23
exiting	26
key command	19
CATALOG menu	189
categories	
All	33
Apps desktop	33
customizing	34
English	33
example of editing	35
Graphing	33
Math	33
Organizr (organizer)	33
Science	33
selecting	32
selecting empty	33
SocialSt (social studies)	33
CBL	
get/return, Get	820
programs	632
send list variable, Send	865
statistical data	568, 569
CBL 2 system	
activity	757

connecting	67	special	11, 19, 50, 655, 657
programs	757	string, char()	602
CBR		symbols	657
get/return, Get	820	uppercase	11, 14
programs	632	uppercase/lowercase	152, 650
send list variable, Send	865	checkTmr(), check timer	788
statistical data	568, 569	circle	
CBR system		drawing	493
connecting	67	graphing	85, 88
programs	757	circle, Circle	631
ceiling(), ceiling	787	Circle, draw circle	631, 788
ceiling, ceiling()	745	Circular definition error	595
certificate 721, 727, 728, 729, 730, 731,		Clean Up menu	187
732,	733	clear	
Certificate revision (Cert. Rev.)	231, 232	drawing, ClrDraw	630
cFactor(), complex factor .	255, 769, 787	error, ClrErr	636
cFactor(†), complex factor	934	graph, ClrGraph	475, 627, 676
CHAR menu (<u>2nd</u> [CHAR])		I/O, ClrIO	575, 620
description	50	clipboard	210, 652
entering special characters	11	Clock	
key command	19	dialog box	42
char(), character string	788	operation	41
character string, char()	602	turning off	47
characters		turning on	49
accented	657, 658	ClockOff, turning clock off	788
deleting	19	ClockOn, turning clock on	789
Greek	11, 19, 50, 656, 659	ClrDraw, clear drawing	630, 790
international/accented ...	11, 19, 50	ClrErr, clear error	636, 790
math	11, 19, 50	ClrGraph, clear graph	627, 676, 790
numeric code, ord()	602	ClrHome, clear home	30, 791
punctuation	50	ClrIO, clear	

I/O	791	connecting	
ClrIO, clear I/O	575, 620, 791	computer	66
cobweb plot. See web plots		devices	66
colDim(), matrix column dimension	791	TI ViewScreen overhead panel	67
colNorm(), matrix column norm	791	TI-Presenter video adapter	67
combinations, nCr()	841	Voyage 200	66
comDenom(), common denominator	791	Constant Memory	145
command mark	660	Constant Memory feature	20
command scripts	208, 660, 663	constants	277
activity	748	predefined	290
commands		contact information	961
Flash Apps	23	contour plots	395, 399, 400
Key	11, 13	DrwCtour, draw contour	399
comment, ●	581, 911	contour-level graphing	102, 394, 943
common denominator, comDenom()	254, 255, 262	contrast	
complex		adjusting	4, 69
factor, cFactor()	255, 769, 787	initial startup	4
factor, cFactor(†)	934	contrast, adjusting	146
mode, Complex Format	185, 919	convert measurements	18
modulus surface	401	convert time, timeCnv()	886
numbers	70, 71	convert, ►	281, 909
solve, cSolve()	238, 796	Coordinates graph format	313, 342
solve, cSolve(†)	934	copy	210, 652
tables	463	copy variable, CopyVar	596, 702
zeros, cZeros()	238, 255, 800	CopyVar, copy variable	702, 792
zeros, cZeros(†)	934	cos(), cosine	793
Complex Format mode	185, 919	cos ⁻¹ (), arccosine	793
complex format mode	20	cosh(), hyperbolic cosine	794
Complex menu	255	cosh ⁻¹ (), hyperbolic arccosine	794
conj(), complex conjugate	792	cot(), cotangent	794
		cot ⁻¹ (), inverse cotangent	795

coth(), hyperbolic cotangent	795	off the curve	384
coth ⁻¹ (), inverse hyperbolic cotangent	795	Selecting a command	25
cover		trace	318
removing	6	Viewing entries	28
replacing	6	cursor keys (⏪⏩⏴⏵)	
stowing	7	additional functions	16
crossP(), cross product	795	entering commands	25
csc(), cosecant	795	function	16
csc ⁻¹ (), inverse cosecant	796	opening Apps	30
csch(), hyperbolic cosecant	796	selecting entry/answer pairs	28
csch ⁻¹ (), inverse hyperbolic cosecant	796	used with the hand key	15
cSolve(), complex solve	238, 796	using the CHAR menu	12
cSolve(†), complex solve	934	CustmOff, custom toolbar off	228, 799
cubic regression, CubicReg	546, 941	CustmOn, custom toolbar on	228, 799
CubicReg, cubic regression	546, 798, 941	CUSTOM (2nd [CUSTOM]) menu	56
cumSum(), cumulative sum	535, 798	descripton	55
Current folder mode	185, 918	key command	18
Current folder status	39	CUSTOM axes (sequence)	362
Current mode	20	CUSTOM custom plots	356, 434, 435
cursor		CUSTOM menu	228
3D graph	381	custom plots, CUSTOM	356, 434, 435
deleting characters	19	custom toolbar See toolbar	
free-moving	317, 343, 351, 360, 380, 419	Custom Units mode	186, 922
functionality	16	Custom, define toolbar	622, 799
hidden surface	383	customer support and service	961
in the history area	28	cut	210, 652
location following APD	9	cycle picture, CyclePic	500, 629, 800
moving	16, 148, 169	Cycle, cycle	799
		CyclePic, cycle picture	629, 800
		cylindrical vector display, ►Cylind	800
		cZeros(), complex zeros	238, 255, 800

cZeros(†), complex zeros 934

D

$d()$, first derivative 76, 245, 263, 265, 802

darker/lighter 146

data (new), NewData 841

data filtering 753

data plots 117

Data/Matrix Editor

cell width 527

column header . . 528, 531, 532, 533

filling 525

list variable 520, 522

new, NewData 597

scrolling 525

shift, shift() 872

sorting columns 535

statistical plots 550

values 523, 524

data/matrix editor 469

date

reset 49

setting 41

dayOfWk(), day of week 802

DE (differential equation) mode 39

decimal

angle display, ▶DD 802

integer display, ▶Dec 680, 802

define toolbar, Toolbar 621, 888

Define, define . . 356, 412, 446, 738, 803

define, Define . 222, 269, 306, 348, 356,
377, . . 412, 446, 472, 479, 591, 596,
627, 738, 803

DEG (degree) mode 39

degree notation, ° 760, 907, 908

degree/minute/second display, ▶DMS 808

delete character (← / → [DEL]) 19

deleting

folder, DelFold 596, 804

variable, DelVar 236, 268, 596, 600,
804

variables of type 704

deleting variables 142

DelFold, delete folder 596, 804

DelType command 704

DelVar, delete variable . . . 596, 600, 804

denominator 791

derivatives 76

first derivative, $d()$ 802

first derivative, $d()$. 76, 245, 263, 265

numeric derivative, nDeriv() 264, 841

Derivatives (graph math tool) . 329, 332,

344, 352

deSolve(), solution 264, 444, 804

det(), matrix determinant 806

diag(), matrix diagonal 806

dialog box

(S) S (SAVE COPY AS) 18

CLOCK 42

edit categories 34

FORMATS/GRAPH FORMATS (◀)	
F)	18
menu indicator	53
MODE	20
to open Apps	30
dialog box, define, Dialog	621, 806
dialog boxes	175
Dialog, define dialog box	621, 806
Diamond modifier key (◀)	
description	14
status	39
differential equations	
DIRFLD, direction field	414, 423
first order	425, 444
FLDOFF, field off	414, 424, 451
graphing	409–452
initial conditions	420
second order	427, 444
SLPFLD, slope field	414, 423, 448
solution methods	413, 439, 944
third order	431
troubleshooting	447
diftol window variable	417
dim(), dimension	807
dimension, dim()	602
direction field, DIRFLD	414, 423, 450
DIRFLD, direction field	414, 423, 450
discontinuities	
detecting	90
Disp, display I/O screen	129, 582, 620,
635,	807
DispG, display graph	620, 627, 807
DispHome, display Home screen ..	620,
807	
display	
graph, DispG	620, 627, 807
Home screen, DispHome ..	620, 807
I/O screen, Disp	129, 582, 620, 635,
807	
table, DispTbl	620, 627, 808
display as	
binary, ▶Bin	680, 786
cylindrical vector, ▶Cylind	800
decimal angle, ▶DD	802
decimal integer, ▶Dec	680, 802
degree/minute/second, ▶DMS ..	808
hexadecimal, ▶Hex	680, 826
polar vector, ▶Polar	850
rectangular vector, ▶Rect	860
spherical vector, ▶Sphere	879
Display Digits mode	166, 185, 918
display digits mode	20
DispTbl, display table	808
Distance (graph math tool) 329, 333, 344,	
.....	352
divide, /	900
domain constraints	251
dot	
addition, .+	901
division, ./	902
multiplication, .*	902
power, .^	902

subtraction, - 901
 dotP(), dot product 808
 DrawFunc, draw function . 488, 632, 808
 drawings and drawing
 circle, Circle 631, 788
 circles 493
 clearing, ClrDraw 630, 790
 contour, DrwCtour 632, 810
 erasing 492
 freehand 491
 function, DrawFunc . . 488, 632, 808
 horizontal line, LineHorz . . . 632, 833
 inverse, DrawInv 489, 632, 808
 line, Line 631, 833
 lines 492, 494
 on a graph 629
 parametric, DrawParm 488, 632, 809
 Pencil 491
 polar, DrawPol 488, 632, 809
 slope, DrawSlp 495, 631, 809
 tangent line, LineTan 632, 833
 vertical line, LineVert 632, 833
 DrawInv, draw inverse . . . 489, 632, 808
 DrawParm, draw parametric . . 488, 632,
 809
 DrawPol, draw polar 488, 632, 809
 DrawSlp, draw slope 495, 631, 809
 drop-down menu, DropDown 622
 DropDown, drop-down menu . . 622, 809
 DrwCtour, draw contour . . 399, 632, 810
 dtime window variable 417





E

E, exponent 16, 810
 e, natural log base 276
 e^(), e to a power 810
 editing 169
 eigVc(), eigenvector 811
 eigVl(), eigenvalue 811
 else if, ElseIf 478, 607
 Else, else 827
 else, Else 607
 ElseIf, else if 607, 811
 end
 custom, EndCustm 622, 799
 dialog, EndDlog 621, 806
 for, EndFor 582, 611, 819
 function, EndFunc 587, 820
 if, EndIf 582, 606, 827
 loop, EndLoop 614, 837
 program, EndPrgm . . . 127, 591, 851
 toolbar, EndTBar 621, 888
 try, EndTry 636, 889
 while, EndWhile 613, 891
 EndCustm, end custom 622, 799
 EndDlog, end dialog 621, 806
 EndFor, end for 582, 611, 819
 EndFunc, end function 587, 820
 EndIf, end if 582, 606, 827
 EndLoop, end loop 614, 837
 EndPrgm, end program . . . 127, 591, 851
 EndTBar, end toolbar 621, 888
 EndTry, end try 636, 889

EndWhile, end while	613, 891	Exact/Approx mode	163, 186, 200, 238, 239, 240, 921
English category	33	exact/approx mode	20
entry line		example	
clearing the history area	30	changing mode settings	21
cursor rests on	28	editing categories	35
inserting commands	24	restoring the default custom menu	56
recalling	28	selecting menu options	51
entry(), entry	812	turning off the clock	48
entry, entry()	217	turning on/off the custom menu	56
entry/answer pairs	29	using dialog boxes	54
status	39	using the CHAR menu	12
EOS (Equation Operating System)	939	using the keyboard map	13
equal, =	604, 903	examples, previews, activities	
Equation Operating System (EOS)	939	3D graphing	99, 746
equations, solving	665, 669, 673, 674	additional graphing topics	108
error conditions after APD	9	angle modes	78
errors and troubleshooting	948	baseball	759
Circular definition	595	CBL 2 program	757
clear error, ClrErr	636, 790	complex factors	769
Memory error	712, 713	complex modulus surface	401
Out-of-memory	271	complex numbers	70
pass error, PassErr	636, 849	complex zeroes	762
programs	634	constants and measurement units	81
transmission	721, 731	converging web plots	365
warnings	918	cos(x)=sin(x) activity	744
Estep window variable	417	cubic polynomial	762
Euler method	413, 439	data filtering	753
evaluate polynomial, polyEval()	850	data/matrix editor	115
EXACT mode status	39	decomposing a rational function	751
exact(), exact	812	derivatives	76

detecting discontinuities	90	sampling	771
differential equations	103	second-order differential equation	427, 444
diverging web plots	367	sequence graphing	96
expanding expressions	73	solving inequalities	75
factorial	70	solving linear equations	74, 258
factoring polynomials	257	split screen	112, 760
Fibonacci sequence	372	standard annuity	765
finding roots	72	statistics	117
function graphing	85, 88	symbolic manipulation	79
graphing functions	85	tables	110
implicit derivatives	76	text operations	130
implicit plots	406	third-order differential equation	431
integrals	77	time value of money	767
log to any base	78	trees and forest	96
memory management	136	tutorial script with the text editor	748
number bases	134	variable management	136
numeric solver	132	vectors	77
oscillating web plots	368	examples, previews, activities	
parametric graphing	92, 760	log to any base	78
path of a ball	92	exclusive or (Boolean), xor	605, 684, 892
polar rose	94	exclusive or picture, XorPic	629, 892
pole-corner problem	737	Exec, execute assembly language	644, 813
population	117	execute assembly language, Exec	644, 813
predator-prey model	370, 435	execute program, Prgm	127, 591, 851
prime factors	70	Exit, exit	813
programming	126, 129, 636	exp▶list(), expression to list	813
Pythagorean theorem	737	expand(), expand	73, 741, 762, 813
quadratic formula	739	expand, expand()	73, 254, 257, 741, 762
rational factors	769		
real factors	769		
reducing expressions	73		


exponent key (2nd [EE])	16	false message	273
exponent, ϵ	16, 810	family of curves	479
Exponential Format mode	167, 185, 919	FCC statement	2
exponential Format mode	20	Fibonacci sequence	372
exponential regression, ExpReg	546, 815, 941	field off, FLDOFF	414, 424, 451
expr(), string to expression	601, 602, 620, 814	field picture, fldpic	418
ExpReg, exponential regression	546, 815, 941	File, new (\blacktriangledown N)	18
expressions	26, 156, 157, 158, 169	File, open (\blacktriangledown O)	18
expanding	73	Fill, matrix fill	816
expression to list, exp(list())	813	Flash applications	178, 192, 271, 688, 689, 692
reducing	73	deleting	721
string to expression, expr()	601, 602, 620, 814	FLASH APPLICATIONS (\blacktriangledown APPS)	
Extract menu	255	accessing Apps not listed	58
eye ϕ z-axis window variable	377, 386, 387	description	50
eye θ x-axis window variable	377, 386, 387	key command	18
eye ψ rotation window variable	377, 386, 388	Flash, upgrading operating system	727, 728, 729
F		FLDOFF, field off	414, 424, 451
factor(), factor	72, 741, 769, 815	fldpic, field picture	418
factor, factor()	72, 238, 254, 257, 741, 769	fldres window variable	417
factorial, !	71, 904	floor(), floor	745, 816
factoring	257	floor, floor()	745
activity	769	fMax(), function maximum	817
		fMin(), function minimum	817
		FnOff, function off	627, 818
		FnOn, function on	627, 818
		folders	185, 918
		delete, DelFold	596, 804
		get/return, getFold()	822
		locking/unlocking	702

new, NewFold 597, 842
 pasting name 704, 705
 renaming 699, 702
 setting, setFold() 617, 867
 transmitting . 716, 717, 718, 719, 720
 VARLINK . 692, 693, 694, 695, 697,
 698, 699, 700
 For, for 819
 for, For 582, 611, 819
 format string, format() ... 602, 620, 627
 format(), format string 819
 FORMATS ( F)
 dialog box 18
 key command 18
 FORMATS dialog box ... 102, 103, 313,
 392, 394, 396
 fpart(), function part 820
 fractions 254, 262, 751, 853
 free-moving cursor . 317, 343, 351, 360,
 380, 419
 Frobenius norm, norm() 844
 full-screen mode
  [QUIT] 19
 Apps desktop 37
 changing from split-screen 64
 displaying Apps in 63
 FUNC (function) mode 39
 Func, program function 587, 820
 function keys (–)
 moving among toolbar menus ... 55
 selecting categories 30, 32

selecting menus 49
 functions 23, 156
 delayed simplification 245
 graphing 300
 maximum, fMax() ... 238, 264, 817
 minimum, fMin() ... 238, 264, 817
 multistatement 477
 off, FnOff 308, 627, 818
 on, FnOn 308, 627, 818
 part, fPart() 820
 program function, Func ... 587, 820
 user-defined 193, 220, 269, 377, 475,
 477, 585, 587, 803

G

Garbage collection message .. 708, 709,
 710, 711
 gcd(), greatest common divisor ... 820
 get time zone, GetTmZn() 825
 Get, get/return CBL/CBR value 568, 634,
 820
 get/return
 calculator, GetCalc .. 633, 724, 725,
 820
 CBL/CBR value, Get .. 568, 634, 820
 configuration, getConfig() .. 617, 821
 denominator, getDenom() . 255, 821
 folder, getFold() 596, 617, 822
 key, getKey() 619, 822
 key, getKey(†) 925, 927
 mode, getMode() 617, 823

number, getNum()	255, 823	GRAPH FORMATS ( F)	18
type, getType()	235, 596, 825	dialog box	18
units, getUnits()	617, 825	Graph mode	185, 200, 301, 340, 347, 355,
GetCalc, get/return calculator	633, 724, 725,	376, 411, 918	
getConfig(), get/return configuration	617, 821	graph mode status	39
getDate(), get date	821	graph number mode status	39
getDenom(), get/return denominator	821	Graph Order graph format	314, 413
getDtFmt(), get date format	822	Graph, graph	826
getDtStr(), get date string	822	graph, Graph	306, 473, 480, 628
getFold(), get/return folder	596, 617	Graph<->Table, table-graph	456
getKey(), get/return key	619, 822	graphing category	33
getKey(+), get/return key	925, 927	graphs	
getMode(), get/return mode	617, 823	drawing on	15
getNum(), get/return number	823	number of	61, 63
getTime(), get time	823	graphs and graphing	
getTmFmt(), get time format	824	∫f(x)dx	329, 332
getTmStr(), get time string	824	3D	374–408
getTmZn(), get time zone	825	animation	500
getType(), get/return type	596, 825	Arc	329, 335, 344, 352
getUnits(), get/return units	617, 825	clearing, ClrGraph	475, 627, 676, 790
global variables	600	contour plots	395, 399, 400
go to, Goto	593, 609, 616	coordinates	85, 317
Goto, go to	826	custom axes	362
GRAD(gradian) mode	78	custom plots	356, 434, 435
Gradian angle mode	78	Derivatives	329, 332, 344, 352
graph		differential equations	409–452
mode	20, 39	Distance	329, 333, 344, 352
number mode	39	drawing 490, 491, 492, 493, 494, 495,	496, 629
Graph 2 mode	186, 921	family of curves	479
		formats	313, 342, 413

functions	300	QuickCenter	321
functions off, FnOff	627, 818	recall graph database, RclGDB	629, 860
functions on, FnOn	627, 818	selecting functions	307, 348, 357, 412
graph databases	503	sequence	353–373
graph, Graph	473, 628, 826	setting, setGraph()	617, 628, 867
Home screen	471, 473	Shade	330, 335
implicit plots	403, 406	shading, Shade	632, 871
independent variable	471	simultaneous graphs	481
Inflection	329, 333	split screen	482, 486, 506
Intersection	329, 331	store graph database, StoGDB	629, 880
inverse functions	489	style, Style	628, 881
line styles	309, 341, 349, 357, 377	Tangent	329, 334, 344, 352
math functions	329	text	496
matrix data	469	time plots	356, 362, 434, 435
Maximum	329, 331	trace, Trace	318, 628, 747, 757, 759, 762, 888
Minimum	85, 329, 331	tracing	85, 318, 322, 344, 351, 361, 380, 419
modes	185, 200, 301, 340, 347, 355, 376, 411, 918	two-graph mode	482, 484, 485, 506
native independent variable	471	Value	329, 330, 344, 352, 361, 381, 420
nested functions	476	viewing window	311, 341, 349, 358, 378
operations	779	web plots	356, 362, 363
overview	300, 338, 345, 353, 374, 409	window variables	311, 341, 349, 358, 378
panning	321	Y= editor	85, 88, 303, 341, 347, 355, 376, 411, 471
parametric	345		
pausing	316		
pictures	497, 499		
piecewise functions	475		
pixels	937		
polar	338		
programs	627		

Zero	329, 331
zoom	322, 343, 351, 361, 380, 628
zoom factors	324, 326
zoom Memory	324, 327
greater than or equal, \geq , \geq	604, 904
greater than, $>$	604, 904
greatest common divisor, gcd()	820
Greek characters	11, 656, 659
Grid graph format	314


H

Hand modifier key (☞)	
description	15
status	39
Hardware version	231, 232
hexadecimal	
display, ►Hex	680, 826
indicator, 0h	911
hidden surface	102, 383, 394
highlighting	
characters when editing	14
to view full name of App	5
highlighting text	651
Histogram	557
History area	
status	39
history area	205, 206, 663
History indicator	29
Home icon	26
Home screen	203
Home screen. See calculator home screen	

hyperbolic	
arccosine cosh ⁻¹ ()	794
arcsine, sinh ⁻¹ ()	875
arctangent, tanh ⁻¹ ()	885
cosecant, csch()	796
cosine, cosh()	794
cotangent, coth()	795
secant, sech()	865
sine, sinh()	875
tangent, tanh()	884





I

I/O port	66
ID list	732, 733, 734
ID number	230, 727, 728, 729, 732, 733, 734
identity matrix, identity()	827
identity(), identity matrix	827
If, if	827
if, If	478, 582, 606, 607
imag(), imaginary part	828
implicit derivatives	76
implicit plots	403, 406, 943
implied multiplication	157, 348
Independent AUTO/ASK, independent	
auto/ask	459, 464
independent auto/ask, Independent	
AUTO/ASK	457, 459, 464
indirection, #	602, 906, 940
inequalities	75
infinity, ∞	276

Inflection (graph math tool)	329, 333
initial conditions	420
initial startup	4
input string, InputSt	601, 619, 725, 829
Input, input	828
input, Input	619, 628
InputSt, input string	601, 619, 725, 829
insert mode ( [INS])	19
inString(), within string	602, 829
instructions	156
calculator Home screen	26
Catalog	23
insufficient display memory, <<...>>	227
int(), integer	829
intDiv(), integer divide	682, 829
integer divide, intDiv()	682
integer part, iPart()	97, 829
integer, int()	829
integrate, ∫() 77, 238, 240, 241, 245, 264,	265, 905
international/accented characters	11
Intersection (graph math tool)	329, 331
inverse cosecant, csc ⁻¹ ()	796
inverse cotangent, cot ⁻¹ ()	795
inverse hyperbolic	
cosecant, csch ⁻¹ ()	796
cotangent, coth ⁻¹ ()	795
secant, sech ⁻¹ ()	865
inverse, x ⁻¹	909
iPart(), integer part	829
isArchiv(), is archived	693

isClkOn(), is clock on	830
isLocked(), is locked	693
isPrime(), prime test	830
isVAR(), is variable	693
Item, menu item	622, 624, 831

K

key commands	
keyboard map	13
special characters	11
keyboard	147
 (hand) key	149
 (second) key	149
 (shift) key	149
 (diamond) key	149
key codes	619
map	11, 13, 656
QWERTY	11
shortcuts	656
keys	
cursor	16
function	15
modifier	14

L

label, Lbl	593, 609, 616
Labels graph format	314
Language mode	187, 922
language mode	
changing mode setting	21
viewing	20

last answer	152, 161, 214, 217	folders	716, 717, 719, 720, 721
last entry	152, 214, 215	get/return CBL/CBR value, Get	568, 634, 820
Lbl, label	831	program	632, 724, 725
lcm, least common multiple	831	send chat, SendChat	724, 725
Leading Cursor graph format	314	send list variable, Send	634, 865
least common multiple, lcm	831	send to calculator, SendCalc	633, 724, 725
left(), left	832	variables	716, 717, 718, 719, 720
left, left()	255, 602	LinReg, linear regression	546, 834, 941
left-right split screen		list difference, Δ list()	834
setting	59	list to matrix, list \blacktriangleright mat()	834
setting initial Apps	61	list \blacktriangleright mat(), list to matrix	532, 834
status	37	lists	
less than or equal, \neq ,	904	augment/concatenate, augment()	785
less than or equal, \leq , $<=$	604	cross product, crossP()	795
less than, $<$	604, 903	cumulative sum, cumSum()	798
lighter/darker	146	difference, Δ list()	834
limit(), limit	832	dot product, dotP()	808
limit, limit()	245, 264, 266	expression to list, exp \blacktriangleright list()	813
Line, draw line	631, 833	list to matrix, list \blacktriangleright mat()	834
linear regression, LinReg	546, 834, 941	matrix to list, mat \blacktriangleright list()	838
LineHorz, draw horizontal line	632, 833	maximum, max()	838
LineTan, draw tangent line	632, 833	mid-string, mid()	839
LineVert, draw vertical line	632, 833	minimum, min()	839
Link transmission table	736	new data, NewData	597, 841
linking and transmitting	865, 866	new, newList()	842
calculator to calculator	632, 715, 716, 718, 724, 725, 726, 727	operations	779
cancelling	721	product, product()	851
errors	721, 731	sort ascending, SortA	879
Flash applications	716, 717, 718, 723, 724		

sort descending, SortD	879
summation, sum()	862, 882
table variables	466
variables	516
ln(), natural logarithm	834
LnReg, logarithmic regression	546, 835, 942
local variable, Local	589, 594, 596, 597
Local, local variable	835
lock variable, Lock	597
Lock, lock variable	835
locked/archived variable status	39
log to any base	78
log(), logarithm	836
logarithm, log()	836
logarithmic regression, LnReg	546, 835, 942
logarithms	834, 836
logistic regression, Logistic	546, 942
Logistic, logistic regression	546, 836, 942
Loop, loop	837
loop, Loop	614
LU, matrix lower-upper decomposition	837

M

mat►list(), matrix to list	838
math category	33
MATH menu	329
MATH menu ($\frac{2nd}{MATH}$)	50
math operations	26, 780

matrices	
augment/concatenate, augment()	743, 785
column dimension, colDim()	791
column norm, colNorm()	791
copying	537
cumulative sum, cumSum()	798
data from a graph	469
determinant, det()	806
diagonal, diag()	806
dimension, dim()	807
dot addition, .+	901
dot division, ./	902
dot multiplication, .(902
dot power, .^	902
dot subtraction, .-	901
eigenvalue, eigVl()	811
eigenvector, eigVc()	811
filling, Fill	816
identity, identity()	827
list to matrix, list►mat()	834
locking	530
lower-upper decomposition, LU	837
matrix to list, mat►list()	838
maximum, max()	838
minimum, min()	839
new data, NewData	597, 841
new, newMat()	842
operations	780
pretty print	519
product, product()	852


QR factorization, QR	856	resetting	688, 689
random, randMat()	743, 858	unarchive, Unarchiv	597, 707, 708, 889
reduced row echelon form, rref()	260, 864	VARLINK screen	690, 692, 693, 694, 695, 697, 698, 699, 700, 707
row addition, rowAdd()	863	MEMORY (2nd [MEM])	19
row dimension, rowDim()	863	Memory (zoom)	324, 327
row echelon form, ref()	861	Memory error	712
row multiplication and addition, mRowAdd()	840	menu item, Item	622, 624, 831
row norm, rowNorm()	864	Menus	
row operation, mRow()	840	APPLICATIONS (APPS)	50, 58
row swap, rowSwap()	864	menus	172
submatrix, subMat()	882	Algebra	253, 256
summation, sum()	862, 882	APPLICATIONS	177
transpose, T	883	Calc	263
matrix to list, mat►list()	838	canceling	55
max(), maximum	838	CATALOG	189
Maximum (graph math tool)	329, 331	CHAR	11, 19, 50
mean(), mean	838	Clean Up	187
measurement		Complex	255
conversions (2nd [►])	18	CUSTOM	228
median(), median	838	custom	623, 626
medium-medium line regression, MedMed	546, 839, 942	CUSTOM (2nd [CUSTOM])	18, 55, 56
MedMed, medium-medium line regression	546, 839, 942	Extract	255
memory		FLASH APPLICATIONS (♦ [APPS])	18, 50, 58
archiving, Archive	596, 707, 708, 785	MATH	329
checking	688, 689	options	14
insufficient display memory, <<...>>	227	selecting options	50
		submenu options	52
		toolbar	172, 228


Trig	255	Current folder	185, 918
using	172	Custom Units	922
messages		custom units	20
BATT	201, 945, 948	DE (differential equation)	39
false	273	DEG (degree)	39
Garbage collection ...	708, 709, 710, 711	Display Digits	166, 185, 918
insufficient display memory, <<...>> .	227	display digits	20
true	273	EXACT	39
undef (undefined)	276	Exact/Approx ...	163, 186, 200, 238, 239,
messages See also errors and troubleshooting		240, 921	
mid(), mid-string	839	exact/approx	20
mid-string, mid()	602	Exponential Format ...	167, 185, 919
min(), minimum	839	exponential format	20
Minimum (graph math tool) .	85, 329, 331	fullscreen ...	19, 32, 37, 61, 63, 64
minute notation, '	908	FUNC (function)	39
mod(), modulo	840	get/return, getMode()	617, 823
modes	183	Graph	185, 200, 301, 340, 347, 355, 376,
3D (three-dimensional)	39	graph	20
Angle	20, 39, 185, 303, 919	Graph 2	186, 921
APPROX	39	graph number	39
Approximate ...	163, 186, 200, 921	graph type	39
Apps desktop	20	grayed out	20
AUTO	39	insert (2nd [INS])	19
Auto	163, 186, 200, 240, 921	Language	187, 922
Base	20, 186, 922	language	20, 21
Complex Format	185, 919	Number of Graphs	186, 921
complex format	20	overwrite (2nd [INS])	19
current	20	PAR (parametric)	39
		POL (polar)	39
		Pretty Print	20, 163, 185, 920

RAD (radian)	39	negation key ([-])	16
SEQ (sequence)	39	negative numbers	16
setting in programs	616	new	
setting, setMode()	617, 628, 868	data, NewData	569, 597, 841
settings	20	folder, NewFold	597, 842
Split App	186, 921	list, newList()	842
Split Screen	186, 920	matrix, newMat()	842
split screen 5, 20, 32, 37, 39, 59, 61, 63		picture, NewPic	597, 629, 842
Unit System	186, 922	plot, NewPlot	556, 628, 843
unit system	20	problem, NewProb	188, 843
Vector Format	185, 920	New file (◀ N)	18
vector format	20	NewData, new data . 518, 531, 569, 597, 841	
modifier keys (2nd ◀ ▶ ↵ ↻)	14	NewFold, new folder	597, 842
status	39	newList(), new list	842
move variable, MoveVar	597	newMat(), new matrix	842
MoveVar, move variable	840	NewPic, new picture	597, 629, 842
mRow(), matrix row operation	840	NewPlot, new plot	556, 628, 842
mRowAdd(), matrix row multiplication and addition	840	NewProb, new problem	188, 843
multiply, *	899	nInt(), numeric integral	264, 843
multistatement functions	477	nmax window variable	358
N		nmin window variable	358
natural log base, e	276	norm(), Frobenius norm	844
natural logarithm, ln()	834	not (Boolean), not	683, 844
ncontour window variable	378	not equal, ≠, /=	604, 903
nCr(), combinations	841	not, Boolean not	605, 683, 844
ncurves window variable	416	nPr(), permutations	844
nDeriv(), numeric derivative	264, 841	nSolve(), numeric solution	254, 845
negate, -	153, 902	number bases	679
		Boolean operations	683
		conversions	680

math operations	681
Number of Graphs mode	186, 921
numbers	
irrational	238, 239
negative	153
rational	238, 239, 240
numeric	
derivative, nDeriv()	264, 841
integral, nInt()	264, 843
solution, nSolve()	254, 845
numeric keypad	16
numeric solver	665, 669, 673, 674
equations	665, 667, 668
graphing	674, 675, 676, 677
split screens	675, 677
variables	669, 673

O

on/off	144
OneVar, one-variable statistic	845
Open file ( O)	18
operating system	729, 730, 731
operating system (OS)	
downloading	68
Operating System (OS) version	231, 232
operating system, upgrading	727, 728, 729
operators	156
or (Boolean), or	684
or, Boolean or	605, 684, 846
ord(), numeric character code	602, 846

Organizr (organizer) category	33
OS	727, 728, 729
OS (Operating System) version	231, 232
Out-of-memory error	271
Output, output	846
output, Output	620, 627
overwrite mode ( [INS])	19

P

P►Rx(), rectangular x coordinate	847
P►Ry(), rectangular y coordinate	847
panning	321
PAR (parametric) mode	39
parallelepiped activity	746
parametric graphing	345
parentheses, brackets, and braces	157, 939
part(), part	847
pass error, PassErr	636
PassErr, pass error	849
paste	210, 652
PAUSE	40
PAUSE indicator	201
Pause, pause	849
pause, Pause	621, 635
percent, %	902
permutations, nPr()	844
pictures	497, 499
and, AndPic	628, 784
cycle, CyclePic	629, 800
deleting	500

exclusive or, XorPic	629, 892	change, PtChg	631, 853
new, NewPic	597, 629, 842	off, PtOff	631, 853
recall, RclPic	629, 860	on, PtOn	631, 853
replace, RplcPic	629, 864	test, ptTest()	631, 853
storing, StoPic	629, 881	text, PtText	631, 854
piecewise functions	475	POL (polar) mode	39
pixel		polar	
change, PxlChg	631, 854	coordinate, R►Pθ()	858
circle, PxlCrcl	631, 854	coordinate, R►Pr()	858
horizontal line, PxlHorz	632, 854	graphing	338
line, PxlLine	496, 631, 854	vector display, ►Polar	850
off, PxlOff	631, 855	polyEval(), evaluate polynomial	850
on, PxlOn	496, 631, 855	polynomials	257, 265
test, pxlTest()	631, 855	activity	762
text, PxlText	631, 855	evaluate, polyEval()	850
vertical line, PxlVert	632, 855	random, randPoly()	859
plots		popup menu, PopUp	620
clearing	554	PopUp, popup menu	620, 850
data	117	port	
new, NewPlot	556, 628, 843	accessory	66
off, PlotsOff	308, 628, 850	power of ten, 10^()	909
on, PlotsOn	308, 628, 850	power regression, PowerReg	547, 851,
selecting	553, 560	942	
tracing	562	power, ^	901, 940
viewing window	561	PowerReg, power regression	547, 851,
Y= Editor	559	942	
PlotsOff, plots off	308, 850	pretty print	85, 163, 204
PlotsOn, plots on	308, 850	Pretty Print mode	20, 163, 185, 920
plotStep window variable	358	Prgm, execute program	127, 591, 851
plotStrt window variable	358	prime number test, isPrime()	830
point		prime numbers	72

prime, '	908	define toolbar, Toolbar	621, 888
problems (new), NewProb	188, 843	define, Define	591, 627, 738, 803
product ID	230	deleting	579
Product ID (identifier)	231, 232	display graph, DispG	620, 627, 807
product(), product	851	display Home screen, DispHome	620, 807
product, Π ()	264, 906	display I/O screen, Disp	129, 582, 620, 635, 807
Program Editor	31	display table, DispTbl	620, 627, 808
programs and programming 23, 572–644		drop-down menu, DropDown	622, 809
arguments	584	else if, Elseif	478, 607, 811
assembly language	641, 644	else, Else	607, 827
branching	582, 606, 609	end custom, EndCustm	622, 799
calling another program	591	end dialog, EndDlg	621, 806
CBL	632	end for, EndFor	582, 611, 819
CBL 2 system	757	end function, EndFunc	587, 820
CBR	632	end if, EndIf	582, 606, 607, 827
CBR system	757	end loop, EndLoop	614, 837
clear error, ClrErr	636, 790	end program, EndPrgm 127, 591, 851	
clear graph, ClrGraph	475, 627, 790	end toolbar, EndTBar	621, 888
clear home, ClrHome	791	end try, EndTry	636, 889
clear I/O, ClrIO	575, 620, 791	end while, EndWhile	613, 891
clear table, ClrTable	791	entering	576, 579
comment, ●	581, 911	execute assembly language, Exec	644, 813
conditional tests	603	execute program, Prgm	127, 591, 851
copying	579	exit, Exit	813
custom toolbar off, CustmOff	228, 621, 799	for, For	582, 611, 819
custom toolbar on, CustmOn	228, 621, 799	format string, format() 620, 627, 819	
debugging	635		
define dialog box Dialog	621, 806		
define toolbar, Custom	622, 799		

function, Func	587, 820	request, Request	620, 622, 861
functions	576, 585, 587	return, Return	589, 592, 862
get/return configuration, getConfig()		running	572
617,	821	stop, Stop	579, 880
get/return folder, getFold()	617, 822	stopping	573
get/return from calculator, GetCalc		subroutines	591
633,	724, 725, 820	tables	627
get/return key, getKey()	619, 822	text, Text	621, 622, 886
get/return key, getKey(†)	925, 927	Then, Then	606, 607, 827
get/return mode, getMode()	617, 823	title, Title	622, 887
get/return units, getUnits()	825	try, Try	636, 889
go to, Goto	593, 609, 616, 826	variables	593
graphs	627	while, While	613, 891
if, If	478, 582, 606, 607, 827	Prompt(), prompt	620, 852
input	574, 582, 619	prompt, Prompt()	620
input, Input	619, 628, 828	proper fraction, propFrac	254, 262, 751
label, Lbl	593, 609, 616, 831	propFrac, proper fraction	254, 262, 751, 853
local, Local	589, 594, 596, 597, 835	PtChg, point change	631, 853
loop, Loop	614, 837	PtOff, point off	631, 853
looping	582, 611, 613	PtOn, point on	631, 853
menu item, Item	622, 624, 831	ptTest(), point test	631, 853
menus	622, 626	PtText, point text	631, 854
multicommand lines	580	PxlChg, pixel change	631, 854
operations	781	PxlCrcl, pixel circle	631, 854
output	574, 582, 620	PxlHorz, pixel horizontal line	632, 854
output, Output	620, 627, 846	PxlLine, pixel line	496, 631, 854
pass error, PassErr	636, 849	PxlOff, pixel off	631, 855
passing values	584	PxlOn, pixel on	496, 631, 855
pause, Pause	621, 635, 849	pxlTest(), pixel test	631, 855
popup menu, PopUp	620, 850	PxlText, pixel text	631, 855
prompt, Prompt()	620, 852		

PxlVert, pixel vertical line 632, 855

Q

QR factorization, QR 856

QR, QR factorization 856

quadratic regression, QuadReg 547, 856,
. 942

QuadReg, quadratic regression 547, 856,
. 942

quartic regression, QuartReg . .547, 857,
942

QuartReg, quartic regression . .547, 857,
942

QuickCenter 321

Quit (2nd [QUIT]) 19

QWERTY Keyboard 11

R

r, radian 907

R►Pθ(), polar coordinate 857

R►Pr(), polar coordinate 858

RAD (radian) mode 39

radian, r 907

rand(), random number 858

randMat(), random matrix 743, 858

randNorm(), random norm 858

random

matrix, randMat() 743, 858

norm, randNorm() 858

number seed, RandSeed . . 743, 859

number, rand() 858

polynomial, randPoly() 859

randPoly(), random polynomial 859

RandSeed, random number seed . . 743,
859

rational functions activity 751

RclGDB, recall graph database 505, 629,
860

RclPic, recall picture 629, 860

real(), real 860

recall

graph database, RclGDB . 505, 629,
860

picture, RclPic 629, 860

Recall (2nd [RCL]) 19

reciprocal, x^{-1} 909

rectangular x coordinate, P►Rx() . . . 847

rectangular y coordinate, P►Ry() . . . 847

rectangular-vector display, ►Rect 860

reduced row echelon form, rref() . . 260,
743

ref(), row echelon form 861

regressions 834

cubic, CubicReg 546, 798, 941

exponential, ExpReg . .546, 815, 941

formulas 941, 943

linear regression, LinReg . 546, 834,
941

logarithmic, LnReg . . .546, 835, 942



logistic, Logistic546, 836, 942

medium-medium line, MedMed . 546,
839, 942

power regression, PowerReg . . .	547, 851,	942
quadratic formula activity		739
quadratic, QuadReg . .	547, 856, 942	
quartic, QuartReg	547, 857, 942	
selecting		545
sinusoidal, SinReg . . .	547, 876, 942	
remain(), remainder		861
Rename, rename		861
rename, Rename		597
replace picture, RplcPic	629, 864	
Request, request		861
request, Request	620, 622	
reserved names		939
results		26
return See get/return		
Return, return		862
return, Return	478, 589, 592	
right(), right		862
right, right()	255, 603	
roots		72
rotate(), rotate	685, 862	
rotate, rotate()	603, 685	
round(), round		863
row echelon form, ref()		861
rowAdd(), matrix row addition		863
rowDim(), matrix row dimension		863
rowNorm(), matrix row norm		864
rowSwap(), matrix row swap		864
RplcPic, replace picture	629, 864	

rref(), reduced row echelon form . .	260, 743,	864
Runge-Kutta method	413, 435, 944	

S

sampling activity		771
SAVE COPY AS ( S)		
description		18
dialog box		18
example		54
Scatter plots		555
scientific notation	16, 154	
scripts	208, 660, 663	
activity		748
tutorial		748
scrolling	28, 226, 463	
sec(), secant		865
sec ⁻¹ (), inverse secant		865
secant, sec(),		865
sech(), hyperbolic secant		865
sech ⁻¹ (), inverse hyperbolic secant . .		865
Second modifier key ()		
description		14
status		39
second notation,		908
selecting categories		32
send chat, SendChat	724, 725, 866	
send list variable, Send	634, 865	
send to calculator, SendCalc . .	633, 724, 725,	865
Send, send list variable		634, 865

SendCalc, send to calculator . . .	633, 724, 725,	865	shift, shift()	603, 686
SendChat, send chat	724, 725, 866		show statistical results, ShowStat . .	548, 873
SEQ (sequence) mode	39		ShowStat, show statistical results . .	548, 873
seq(), sequence	866		sign(), sign	873
sequence graphing	353–373		simplification	
serial number	230		delayed	245
set			rules	242
folder, setFold()	617, 867		stopping	244
graph, setGraph()	617, 628		simult(), simultaneous equations . .	873
mode, setMode()	617, 628		simultaneous equations, simult() . .	260
table, setTable()	459, 617		sin(), sine	874
units, setUnits()	617		sin ⁻¹ (), arcsine	874
Set factors (zoom)	324, 326		sinh(), hyperbolic sine	875
setDate(), set date	867		sinh ⁻¹ (), hyperbolic arcsine	875
setDtFmt(), set date format	867		SinReg, sinusoidal regression . .	547, 876, 942
setFold(), set folder	617, 867		sinusoidal regression, SinReg . .	547, 876, 942
setGraph(), set graph	617, 628, 867		slope field, SLPFLD	414, 423, 448
setMode(), set mode	617, 628, 868		SLPFLD, slope field	414, 423, 448
setTable(), set table	459, 617, 869		Smart Graph	316
setTime(), set time	869		SocialSt (social studies) category	33
setTmFmt(), set time format	869		software version	230
setTmZn(), set time zone	870		Solution Method graph format	413
setUnits(), set units	617, 870		solution, deSolve()	264, 444, 804
Shade (graph math tool)	330, 335		solve(), solve	74, 80, 446, 876
Shade, shade	632, 871		solve, solve()	74, 80, 238, 240, 241, 245, 251, 254, 258, 446
shade, Shade	632		solving linear equations	74, 258
Shift modifier key (⇧)				
description	14			
status	39			
shift(), shift	534, 603, 686, 872			

SortA, sort ascending	879	startTmr(), start timer	879
SortD, sort descending	879	statistics	540–571
special characters	655, 657	Box Plot	556
spherical vector display, (Sphere	879	Calculation Type	541, 545
Split App mode	186, 921	categories	565
split screen	482, 486, 662, 675, 677	Category	541, 543
entry line	512, 514	combinations, nCr()	841
exiting	509	factorial, !	71, 904
setting	506	Freq	541, 543
switch, switch()	617, 882	frequency	563
switching	513	Histogram plots	557
Split Screen mode	186, 920	mean, mean()	838
split-screen mode		median, median()	838
active graph	39	new plot, NewPlot	556, 843
exiting	63	operations	782
Number of graphs	61	overview	540
Ratio	61	permutations, nPr()	844
returning from within an App	32	plots	550, 553, 554, 558, 560
selecting active App	63	plots off, PlotsOff	308, 628, 850
setting	59	plots on, PlotsOn	308, 628, 850
setting initial Apps	61	random norm, randNorm()	858
specifying Apps displayed	61	random number seed, RandSeed	743, 859
Split 1 App	61	random number, rand()	858
Split 2 App	61	Scatter plots	555
status	37	show results, ShowStat	548, 873
status and open Apps	5	standard deviation, stdDev()	880
viewing	20	two-variable results, TwoVar	546, 889
square root, $\sqrt{}$ ()	906	variables	548
standard annuity activity	765	variance, variance()	890
standard deviation, stdDev()	880		
start timer, startTmr()	879		

xyline plots	555	inputting, InputSt	601, 619, 725
statistics. <i>See also</i> regressions		left, left()	602, 832
status		mid-string, mid()	602, 839
battery low	69	operations	600, 602, 782
on Apps desktop	5	right, right()	603, 862
split-screen	37	rotate, rotate()	603, 862
status line	199, 303	shift, shift()	603, 872
command parameters	25	string to expression, expr()	601, 602,
history information	29		620, 814
stdDev(), standard deviation	880	within, lnString	602, 829
StoGDB, store graph database	505, 629,	Style, style	881
880		style, Style	310, 628
Stop, stop	880	subMat(), submatrix	882
stop, Stop	579	submenus	174
StoPic, store picture	629, 881	substitutions	247, 249, 251
stopping a calculation	162	subtract, -	899
Store (STO) key	19	subtraction key (\square)	16
storing		sum(), summation	862, 882
graph database, StoGDB	629, 880	sum, Σ ()	264, 906
picture, StoPic	629, 881	support and service	961
symbol, \rightarrow	596, 910	switch(), switch	882
string(), expression to string	603, 881	switch, switch()	617
strings		symbolic manipulation	267
append, &	602, 905	system data, sysData	469, 470
character code, ord()	602	system variables	938, 939
character string, char()	602, 788		
dimension, dim()	602	T	
expression to string, string()	603,	τ , transpose	883
881		t0 window variable	415
format, format()	602, 620, 627, 819	TABLE SETUP, table setup	455
indirection, #	602, 906, 940	Table, build table	883

table-graph, Graph<->Table	456	Taylor polynomial, taylor()	264, 267, 885
tables		taylor(), Taylor polynomial	885
Δ tbl	456	tblStart, table start	456
automatic	459	tCollect(), trigonometric collection	886
cell width	461, 467	temperature conversion, tmpCnv()	284, 887
clearing, ClrTable	791	temperature-range conversion, Δ tmpCnv()	284
complex numbers	463	temperature-range conversion, Δ tmpCnv()	888
differential equations	452	tExpand(), trigonometric expansion	886
displaying, DispTbl	620, 627, 808	text editing	645, 655, 660
functions	463	cut, copy, paste	210, 652
generating with sequence	372	find	653
graphing, Graph<->Table	456	highlighting	651
incrementing, Δ tbl	456	Text, text	886
Independent AUTO/ASK	457, 459, 464	text, Text	621, 622
manual	464	Then, Then	606, 607, 827
overview	454	three-dimensional graphing	374–408
programs	627	animation	99, 390
setTable()	459	CONTOUR LEVELS	102, 394
setting, setTable()	617, 869	HIDDEN SURFACE	102, 394
setup	459	WIRE AND CONTOUR	102, 394
setup, TABLE SETUP	455	WIRE FRAME	102, 394
starting, tblStart	456	TI Connectivity Cable	64, 715, 729, 733
tblStart	456	TI ViewScreen overhead panel	
tan(), tangent	884	connecting	67
\tan^{-1} (), arctangent	884	TI Connect software	64, 729
Tangent (graph math tool)	329, 334, 344, 352	time	
tangent, tan()	884	reset	49
tanh(), hyperbolic tangent	884	setting	41
\tanh^{-1} (), hyperbolic arctangent	885		

time plots, TIME	356, 362, 434, 435	transpose, T	883
time value of money activity	767	Trig menu	255
TIME, time plots	356, 434, 435	trigonometric	
timeCnv(), convert time	886	collection, tCollect()	255
TI-Presenter video adapter		expansion, tExpand()	255
connecting	67	trigonometric collection, tCollect()	886
Title, title	887	trigonometric expansion, tExpand()	886
tmax window variable	349, 415	true message	273
tmin window variable	349	Try, try	889
tmpCnv(), temperature conversion	284, 887	try, Try	636
toolbar		tstep window variable	350, 415
define, Custom	622, 799	turning clock off, ClockOff	788
off, CustmOff	228, 799	turning clock on, ClockOn	789
on, CustmOn	228, 799	turning off	8
Toolbar menus		after APD	9
calculator Home screen	49	following inactivity	9
moving among	55	turning on	
replaced by custom menu	55	initial startup	4
selecting math operations	15	TwoVar, two-variable results	546
Toolbar, toolbar	621, 888	two-variable results, TwoVar	546, 889
top-bottom split screen		typing	
setting	59	file name	31
setting initial Apps	61	to scroll through Catalog	24
status	37	uppercase characters	11
tplot window variable	416	U	
Trace, trace	747, 757, 759, 762, 888	Unarchiv, unarchive variables	597, 707, 708,
trace, Trace 318, 628, 747, 757, 759, 762		889
tracing 85, 318, 322, 344, 351, 361, 380, 419		unarchive variables, Unarchiv	597, 707, 708,
transmitting. See linking and transmitting		889
		undef (undefined) message	276

underscore, _ 908
 Unit ID (identifier) 231, 232
 Unit System mode 186, 922
 unit System mode 20
 unit vector, unitV() 890
 units
 converting 281
 defaults 285, 290
 displaying 285
 get/return, getUnits() 825
 measurement 277
 modes 186, 922
 setting, setUnits() 617, 870
 user-defined 288
 unit-to-unit cable 68
 connecting 67
 unitV(), unit vector 890
 Unlock, unlock 890
 unlock, Unlock 597
 upgrading operating system (OS) ... 728,
 729
 uppercase characters 11
 user-defined functions ... 193, 220, 269,
 377, 475, 477, 585, 587, 803
 user-defined units 288

V

Value (graph math tool) ... 329, 330, 352,
 381, 420
 variables 39, 195, 197
 archiving and unarchiving . 706, 707

archiving, Archive 596, 707, 708, 785
 clearing 678
 copy, CopyVar 596, 702, 792
 copying 702
 data 517
 defined 233, 670
 delayed simplification 245
 delete, DelVar .. 236, 268, 596, 600,
 804
 deleting 721
 DelType 704
 in applications 704, 705
 local, Local .589, 594, 596, 597, 835
 locking, Lock 597
 locking/unlocking 201, 702
 matrix 519
 moving, MoveVar 597
 overriding 237
 pasting name 704, 705
 recall 19
 referring to App files 30
 renaming 699
 reserved names 938, 939
 statistical 544, 548
 store 19
 system 938, 939
 text 208
 transmitting 715, 716, 719, 720
 unarchive, Unarchiv . 597, 707, 708,
 889
 undefined 233, 670

unknown, solving for	669, 673
unlocking, Unlock	597
VARLINK	690, 692, 693, 694, 695, 697, 698, 699, 700, 707
variance(), variance	890
Vector Format mode	185, 920
vector format mode	20
vectors	77
cross product, crossP()	795
cylindrical vector display, ►Cylind	800
dot product, dotP()	808
unit, unitV()	890
Vector Format mode	185, 920
viewing angle	385
viewing orbit	390

W

web plots	
convergence	365
divergence	367
oscillation	368
WEB	356, 362, 363
WEB, web plots	356, 362, 363
when(), when	890
when, when()	475
While, while	891
while, While	613
Window Editor	58
window variables	
(x	937
(y	937

diftol	417
dtime	417
Estep	417
eye ϕ (z axis)	377, 386, 388
eye θ (x axis)	377, 386, 388
eye ψ (rotation)	377, 386, 388
fldres	417
ncontour	378
ncurves	416
nmax	358
nmin	358
plotStep	358
plotStrt	358
θ max	341
θ min	341
θ step	342
t0	415
tmax	349, 415
tplot	416
tstep	350, 415
xgrid	378
xmax	311, 342, 350, 358, 378, 416, 937
xmin	311, 342, 350, 358, 378, 416, 937
xres	312
xscl	311, 342, 350, 358, 416
ygrid	378
ymax	311, 342, 350, 358, 378, 416, 937

ymin . 311, 342, 350, 358, 378, 416,
 937
 ysc1 311, 342, 350, 358, 416
 zmax 378
 zmin 378
 wire-and-contour graphing 102, 394
 wire-frame graphing 102, 394
 with, | 77, 80, 237, 247, 910, 940
 within string, inString() 602, 829

X

xgrid window variable 378
 xmax window variable . . 311, 342, 350,
 358, 378, 416, 937
 xmin window variable 311, 342, 350, 358,
 378, 416, 937
 xor, Boolean exclusive or . 605, 684, 892
 XorPic, exclusive or picture . . . 629, 892
 xres window variable 312
 xscl window variable 311, 342, 350, 358,
 416, 937
 xyline plots 555

Y

Y= editor 85, 88, 303, 341, 347, 355, 376,
 411, 471
 ygrid window variable 378
 ymax window variable . . 311, 342, 358,
 378, 416, 937
 ymin window variable 311, 342, 350, 358,
 378, 416, 937

ysc1 window variable 311, 342, 350, 358,
 416

Z

Zero (graph math tool) 329, 331
 zeroes
 activity 762
 zeroes, zeroes() 238, 254, 261
 zeroes, zeros() 739, 893
 zeros(), zeroes 739, 893
 zmax window variable 378
 zmin window variable 378
 zoom
 box, ZoomBox 323, 325, 894
 data, ZoomData 324, 895
 decimal, ZoomDec 323, 895
 factors 324, 326
 fit, ZoomFit 324, 896
 in, ZoomIn 323, 326, 896
 integer, ZoomInt 324, 896
 Memory 324, 327
 out, ZoomOut 323, 326, 897
 previous, ZoomPrev 328, 897
 recall, ZoomRcl 328, 897
 square, ZoomSqr 323, 897
 standard, ZoomStd 323, 898
 store, ZoomSto 327, 328, 898
 trig, ZoomTrig 324, 898
 Zoom menu 322
 ZoomBox, zoom box 894
 ZoomData, zoom data 895

ZoomDec, zoom decimal	895
ZoomFit, zoom fit	896
ZoomIn, zoom in	896
ZoomInt, zoom integer	896
ZoomOut, zoom out	897
ZoomPrev, zoom previous	897
ZoomRcl, zoom recall	897
ZoomSqr, zoom square	897
ZoomStd, zoom standard	898
ZoomSto, zoom store	898
ZoomTrig, zoom trig	898