# TI-Basic Programming Guide for the TI-84 Plus CE Graphing Calculator

To obtain the latest version of the documentation, go to *education.ti.com/guides*.

# Important Information

Except as otherwise expressly stated in the License that accompanies a program, 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 amount set forth in the license for the program. Moreover, Texas Instruments shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

.

## *Contents*

# What's New

## *What's New in TI-Basic Programming Guide for the TI-84 Plus CE Graphing Calculator v5.2*

**Programing Guide Menu Changes:**

All items in this list are new or updated entries in the TI-Basic Programming Guide for the TI-84 Plus CE Graphing Calculator v5.2.

**PRGM CNTRL Menu**

- Pause
- Wait

**PRGM IN/OUT Menu**

- eval(
- expr(
- Get(
- Send(
- toString(
- String>Equ(

**PRGM Hub Menu**

- Send("Set...
- Send("READ...
- Settings...
- Wait
- Get(
- eval(
- Send("CONNECT-OUTPUT...
- Send("CONNECT-INPUT...
- Ports...
- Send("RANGE...
- Send("AVERAGE...
- Send("DISCONNECT-OUTPUT...
- Send("DISCONNECT-INPUT...
- Manage...

# Introduction to TI-Basic on your TI-84 Plus CE

You can use TI-Basic to create a program on your graphing calculator. You can create a program that will calculate a desired output or control an experience, such as a game.

## *What Is a Program?*

A program is a set of one or more command lines, each containing one or more instructions. When you execute a program, the TI 84 Plus CE performs each instruction on each command line in the same order in which you entered them. The number and size of programs that the TI 84 Plus CE can store is limited only by available memory.

To create a program, simply enter command lines using the Program Editor. The program will run from the Home Screen. Use this guide to learn how to create, edit, and delete programs.

**Tip:** Use Catalog Help by pressing [ + ] on most commands to help you fill in the correct arguments for the commands before you paste them into the Program Editor.

As you progress in programming, a TI-Basic Program Editor is also available in TI Connect™ CE software. You can use the Program Editor workspace in TI Connect™ CE to create programs, to send programs to a connected calculator via USB, to test your programs, and to save programs to your computer. The Program Editor workspace in TI Connect™ CE allows copy, cut, paste, and undo commands.

**Note:** The Program Editor on the calculator does not contain editing features such as copy, cut, paste, or undo. When on the calculator, remember you cannot undo a [clear] or [del].

# Getting Started Activity:

## *Programming the Formula to find the Volume of a Cylinder given Radius and Height*

Given the Radius and Height of a cylinder, you can compute the Volume using this formula. This activity allows you to write a program to prompt for the values of the Radius and Height of a cylinder so that you can then compute the Volume.

The formula for the volume of a cylinder is

$V = \pi R^2 H$ cubic units

Where

V = Volume

R = Radius of the base

H = Height of the cylinder

This program could be useful for a variety of activities such as:

- Providing a table with many values of Radius and Height and having students fill out the Volume column
- Running a program to fill in the values for Volume in the table

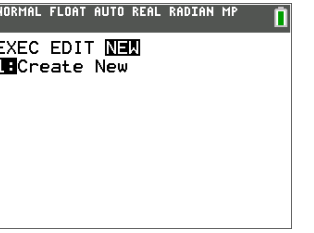
Some questions to investigate:

- (If formula is unknown to the student), what pattern do you see in the Volume numbers to make a good guess at the formula?
- What is the largest Volume found?
- How much does the Volume increase if the Height increases by one unit?
- How much does the Volume increase if the Radius increases by one unit?

Running a program repeatedly as a tool allows quick analysis for higher-level thinking problems.

## *Creating a NEW Program*

1. Press PRGM ▶ ▶ to display the **PRGM NEW** menu.

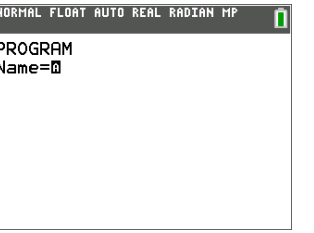


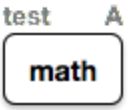
## *Naming the Program*

1. Press [ENTER] to select **1:Create New**.

   The **Name=** prompt is displayed, and [2nd] [A-lock] (alpha-lock) is on.

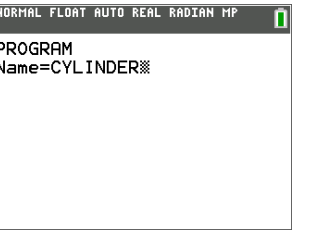
   **Tip**: The alpha characters are upper right above keys on the keypad and are pasted when [alpha] or [2nd] [A-lock] is pressed before pressing the primary key.




2. Press **C Y L I N D E R**, and then press [ENTER] to name the program **CYLINDER**.

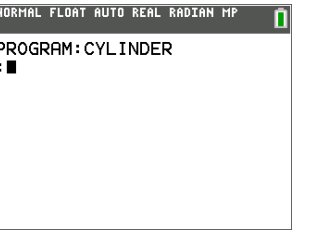
   **Tip**: Program names can have a maximum of eight characters. First character must be a letter. Notice the checkerboard cursor on the screen when the maximum is reached.




3. Press [ENTER] and you are now in the program editor.

   The colon ( : ) in the first column of the second line indicates the beginning of a command line.

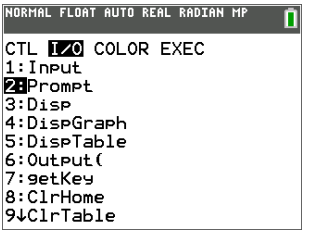
   **Note**: On the calculator, the command lines are not numbered as when using the TI Connect™ CE Program Editor.



## *Entering Commands*

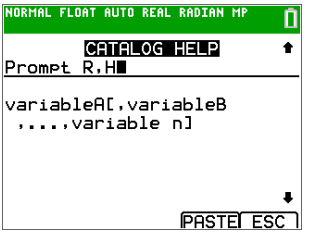
Whoever uses your program will have to input the Radius and Height values. You will use the **Prompt** command.

1. Press PRGM ▶ to access the I/O (Input/Output) command menu.
2. Press ▼ to highlight the **Prompt** command.

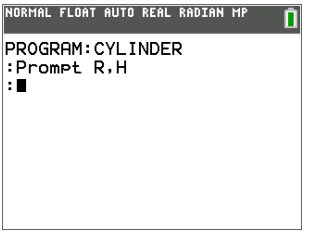



   **Note**: For this example, you will use the Catalog Help feature to illustrate this built-in argument syntax help in the calculator. If you already know the arguments for a command, you can select a menu item and paste them to the Program Editor without using Catalog Help.

3. The **Prompt** menu item number is highlighted so press +. Use the Catalog Help syntax editor (if needed). The syntax for the arguments of Prompt is shown below the editing line as variables separated by commas. Anything within a square bracket [ ] is an optional argument, so Prompt needs at least one variable name.

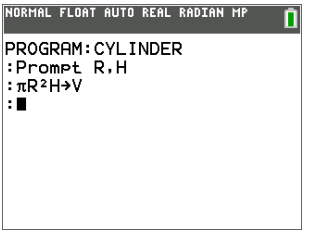



4. Press alpha **R** , alpha **H** to enter the variable names for Radius and Height.
5. Press [PASTE] (trace) to paste the command with the arguments back to the Program Editor. Press [ESC] ([graph]) to return to the last cursor location without pasting.
6. Back on the Program Editor, press enter to move the cursor to the next command line.



Store the formula for the volume of a cylinder:

7. To enter the expression [π] **R$^2$ H** and store value to the variable V, press [2nd] [π] [alpha] **R** [x²] [alpha] **H** [sto→] [alpha] **V** [enter].




## *Displaying the Calculated Volume.*

Create a command line to display the calculated volume:

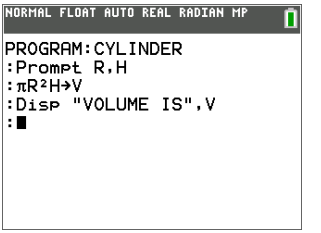
1. Press [prgm] [▸] **3** to select 3:Disp from the **PRGM I/O** menu.

   **Disp** is pasted to the command line.

   **Tip**: Remember you can press [+] on most commands to use the Catalog Help syntax editor to see the correct arguments for commands.

2. Press [2nd] [A-lock] ["] **VOLUME IS** ["] [alpha] [,] [alpha] **V** [enter]

   This will display the text **VOLUME IS** on one line and the calculated value of **V** on the next line of the Home Screen when you run the program.



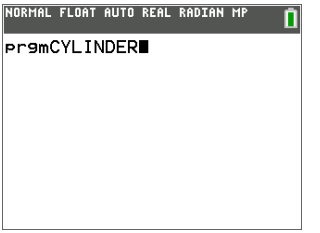

## *Running a Program*

Your program is complete! Now run the program from the Home Screen.

1. Press [2nd] [quit] to display the **Home Screen**.
2. Press [prgm] to display the **PRGM EXEC** menu.

   The items on this menu are the names of stored programs.

3. Press enter to paste **prgm CYLINDER** to the current cursor location. (If **CYLINDER** is not item 1 on your **PRGM EXEC** menu, move the cursor to **CYLINDER** before you press enter.)




## *Finding the Volume*

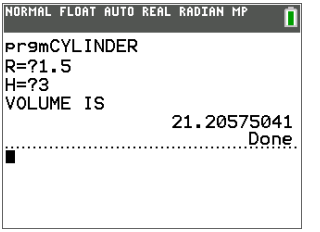
To find the volume of the cylinder with Radius 1.5 cm and Height 3 cm, complete the following steps.

1. Press enter to execute (run) the program.
2. When prompted for **R**, enter **1.5** and press enter
3. When prompted for **H**, enter **3** and press enter.

   The text **VOLUME IS**, the value of **V**, and **Done** are displayed.

   The volume of the cylinder is displayed to 8 decimal places as 21. 20575041 cubic cm.

4. At this point, to rerun the program, press enter and repeat for different values of **R** and **H**.



# Creating and Deleting Programs

This section describes how to create programs, and how to delete programs.

## *Operating Systems Versions and Programming*

- Programs created using the TI-84 Plus OS 2.55MP and earlier or the TI-83 Plus 1.19 OS or earlier will run on the TI-84 Plus CE; however, they may result in unexpected displays on the TI-84 Plus CE given the high resolution screen. You should test your existing programs on the TI-84 Plus CE and adjust command arguments as needed. In particular, any commands that display on the graph need to have the arguments adjusted to the desired pixel locations on the graph area. Programs displaying to the Home Screen should run as expected.
- Programs can run in Classic or MathPrint™ mode.
- Shortcut menus are available wherever the MATH menu can be accessed.
- MathPrint™ templates are not available for programs. All input and output is in Classic format.
- You can use fractions in programs, but you should test the program to make sure that you get the desired results.
- The spacing of the display may be slightly different in MathPrint™ mode than in Classic mode. If you prefer the spacing in Classic mode, set the mode using a command in your program. Screen shots for the examples in this chapter were taken in MathPrint™ mode.
- Syntax help is built in on the TI-84 Plus CE. When in program edit mode, press

**Note**: Press [+] when a command is highlighted in a menu to use the syntax help for your programming.

## *Creating a New Program*

To create a new program, follow these steps.

1. Press [PRGM] [◀] to display the **PRGM NEW** menu.


2. Press [ENTER] to select **1:Create New**. The **Name=** prompt is displayed, and alpha-lock is on.
3. Press a letter from A to Z or θ to enter the first character of the new program name.

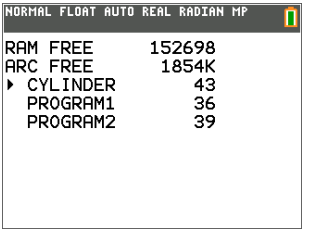
**Note:** A program name can be one to eight characters long. The first character must be a letter from A to Z or θ. The second through eighth characters can be letters, numbers, or θ.

4. Enter zero to seven letters, numbers, or θ to complete the new program name.
5. Press [ENTER]. The program editor is displayed.
6. Enter one or more program commands.
7. Press [2nd] [QUIT] to leave the program editor and return to the home screen.

## *Managing Memory and Deleting a Program*

To check whether adequate memory is available for a program you want to enter:

1. Press [2nd] [MEM] to display the **MEMORY** menu.
2. Select **2:Mem Management/Delete** to display the **MEMORY MANAGEMENT/DELETE** menu.
3. Select **7:Prgm** to display the **PRGM** editor.



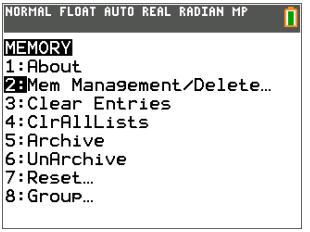

The TI-84 Plus CE expresses memory quantities in bytes.

## *Increase Available Memory*

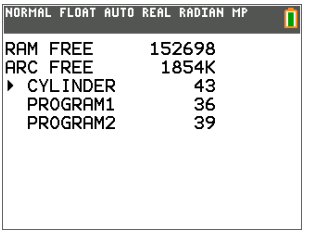
You can increase available memory in one of two ways. You can delete one or more programs or you can archive some programs.

**To increase available memory by deleting a specific program:**

1. Press [2nd] [MEM] and then select **2:Mem Management/Delete** from the **MEMORY** menu.




2. Select **7:Prgm** to display the program files.



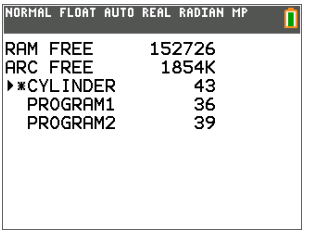

3. Press ▲ and ▼ ALPHA o move the selection cursor (▸) next to the program you want to delete, and then press DEL. The program is deleted from memory.

   **Note:** You will receive a message asking you to confirm this delete action. Select **2:yes** to continue.

   To leave the **PRGM** editor screen without deleting anything, press 2nd [QUIT], which displays the home screen.

**To increase available memory by archiving a program:**

1. Press 2nd [MEM] and then select **2:Mem Management/Delete** from the **MEMORY** menu.
2. Select **2:Mem Management/Delete** to display the **MEMORY MANAGEMENT/DELETE** menu.
3. Select **7:Prgm...** to display the program files.




4. Press ENTER to archive the program. An asterisk will appear to the left of the program to indicate it is an archived program.

   To unarchive a program in this screen, put the cursor next to the archived program and press ENTER. The asterisk will disappear.

   **Note:** Archive programs cannot be edited or executed. In order to edit or execute an archived program, you must first unarchive it.

# Entering Command Lines and Executing Programs

This section describes how to enter a command line and how to execute programs.

## *Entering a Program Command Line*

You can enter on a command line any command, instruction, or expression that you could execute from the home screen. In the program editor, each new command line begins with a colon. To enter more than one instruction or expression on a single command line, separate each with a colon.

**Note:** A command line can be longer than the screen is wide.

While in the program editor, you can display and select from menus. You can return to the program editor from a menu in either of two ways.

- Select a menu item, which pastes the item to the current command line.

  — or —

- Press [CLEAR].

When you complete a command line, press [ENTER]. The cursor moves to the next command line.

Programs can access variables, lists, matrices, and strings saved in memory. If a program stores a new value to a variable, list, matrix, or string, the program changes the value in memory during execution.

You can call another program as a subroutine.

## *Executing a Program*

To execute a program, begin on a blank line on the home screen and follow these steps.

1. Press [PRGM] to display the **PRGM EXEC** menu.
2. Select a program name from the **PRGM EXEC** menu. **prgm***name* is pasted to the home screen (for example, **prgmCYLINDER**).
3. Press [ENTER] to execute the program. While the program is executing, the busy indicator is on.

Last Answer (**Ans**) is updated during program execution. Last Entry is not updated as each command is executed.

The TI-84 Plus CE checks for errors during program execution. It does not check for errors as you enter a program.

## *Breaking a Program*

To stop program execution, press [ON]. The **ERR:BREAK** menu is displayed.

- To return to the home screen, select **1:Quit**.
- To go where the interruption occurred, select **2:Goto**.

# Editing Programs

In this section you will follow steps to edit a program. This section describes how to insert and delete command line.

## *Editing a Program*

To edit a stored program, follow these steps.

1. Press PRGM ▶ to display the **PRGM EDIT** menu.
2. Select a program name from the **PRGM EDIT** menu. Up to the first nine lines of the program are displayed.

   **Note:** The program editor does not display a ↓ to indicate that a program continues beyond the screen.
3. Edit the program command lines.
   - Move the cursor to the appropriate location, and then delete, overwrite, or insert.
   - Press CLEAR to clear all program commands on the command line (the leading colon remains), and then enter a new program command.

**Note:** To move the cursor to the beginning of a command line, press 2nd ◀; to move to the end, press 2nd ▶. To scroll the cursor down seven command lines, press ALPHA ▼. To scroll the cursor up seven command lines, press ALPHA ▲.

**Inserting and Deleting Command Lines**

To insert a new command line anywhere in the program, place the cursor where you want the new line, press 2nd [INS], and then press ENTER. A colon indicates a new line.

To delete a command line, place the cursor on the line, press CLEAR to clear all instructions and expressions on the line, and then press DEL to delete the command line, including the colon.

# Copying and Renaming Programs

This section describes how to copy and rename a program, and how to scroll the menus.

## *Copying and Renaming a Program*

To copy all command lines from one program into a new program, follow steps 1 through 5 for Creating a New Program, and then follow these steps.

1. Press [2nd] [RCL]. **Rcl** is displayed on the bottom line of the program editor in the new program.
2. Press [PRGM] to display the **PRGM EXEC** menu.
3. Select a name from the menu. **prgm***name* is pasted to the bottom line of the program editor.
4. Press [ENTER]. All command lines from the selected program are copied into the new program.

Copying programs has at least two convenient applications.

- You can create a template for groups of instructions that you use frequently.
- You can rename a program by copying its contents into a new program.

**Note:** You also can copy all the command lines from one existing program to another existing program using **RCL**.

## *Scrolling the PRGM EXEC and PRGM EDIT Menus*

The TI-84 Plus CE sorts **PRGM EXEC** and **PRGM EDIT** menu items automatically into alphanumerical order. Each menu only labels the first 10 items using 1 through 9, then 0.

To jump to the first program name that begins with a particular alpha character or θ, press [ALPHA] [letter from A to Z or θ].

**Note:** From the top of either the **PRGM EXEC** or **PRGM EDIT** menu, press [▲] to move to the bottom. From the bottom, press [▼] to move to the top. To scroll the cursor down the menu seven items, press [ALPHA] [▼]. To scroll the cursor up the menu seven items, press [ALPHA] [▲].

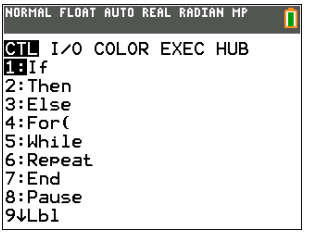
# PRGM CTL (Control) Instructions

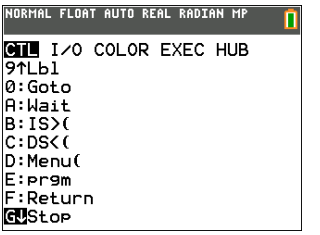
This section describes the **PRGM CTL** (Control) Instructions.

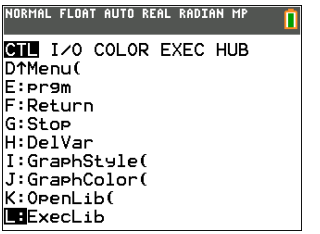
## *PRGM CTL Menu*

To display the **PRGM CTL** (program control) menu, press PRGM from the program editor only.

**Important Tip**: To quickly find a command, use alpha ▲ or alpha ▼ to page through screens.




**CTRL** **I/O** **COLOR** **EXEC** **HUB**

| | | Description |
|---|---|---|
| 1: | If | Creates a conditional test. |
| 2: | Then | Executes commands when **If** is true. |
| 3: | Else | Executes commands when **If** is false. |
| 4: | For( | Creates an incrementing loop. |
| 5: | While | Creates a conditional loop. |
| 6: | Repeat | Creates a conditional loop. |

| | | |
|---|---|---|
| 7: | End | Signifies the end of a block. |
| 8: | Pause | Pauses program execution. |
| 9: | Lbl | Defines a label. |
| 0: | Goto | Goes to a label. |
| A: | Wait | Suspends execution of a program for a given time. |
| B: | IS>( | Increments and skips if greater than. |
| C: | DS<( | Decrements and skips if less than. |
| D: | Menu( | Defines menu items and branches. |
| E: | prgm | Executes a program as a subroutine. |
| F: | Return | Returns from a subroutine. |
| G: | Stop | Stops execution. |
| H: | DelVar | Deletes a variable from within program. |
| I: | GraphStyle( | Designates the graph style to be drawn. |
| J: | GraphColor( | Designates the color of the graph to be drawn |
| K: | OpenLib( | Extends TI-Basic (not available) |
| L: | ExecLib( | Extends TI-Basic (not available) |

**Note**: Press [+] when a command is highlighted in a menu to use the syntax help for your programming.

These menu items direct the flow of an executing program. They make it easy to repeat or skip a group of commands during program execution. When you select an item from the menu, the name is pasted to the cursor location on a command line in the program.

To return to the program editor without selecting an item, press [CLEAR].

**Controlling Program Flow**

Program control instructions tell the TI-84 Plus CE which command to execute next in a program. **If**, **While**, and **Repeat** check a defined condition to determine which command to execute next. Conditions frequently use relational or Boolean tests, as in:

**If A<7:A+1→A**
or
**If N=1 and M=1:Goto Z**

**If**

Use **If** for testing and branching. If *condition* is false (zero), then the *command* immediately following **If** is skipped. If *condition* is true (nonzero), then the next *command* is executed. **If** instructions can be nested.

**:If** *condition*
:*command* (if true)
:*command*

**Program**

```
PROGRAM:COUNT
:0→A
:Lbl Z
:A+1→A
:Disp "A IS ",A
:If A≥2
:Stop
:Goto Z
:
```

**Output**

```
prgmCOUNT
A IS
                    1
A IS
                    2
                 Done
```

---

**If-Then**

**Then** following an **If** executes a group of *commands* if *condition* is true (nonzero). **End** identifies the end of the group of *commands*.

**:If** *condition*
**:Then**
:*command* (if true)
:*command* (if true)
**:End**
:*command*

**Program**

```
PROGRAM:TEST
:1→X:10→Y
:If X<10
:Then
:2X+3→X
:2Y-3→Y
:End
:Disp X,Y
:
```

**Output**

```
prgmTEST
                    5
                   17
                 Done
```

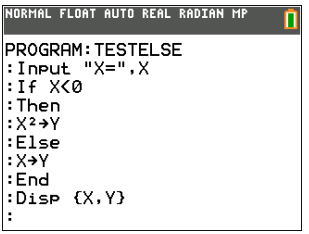
---

**If-Then-Else**

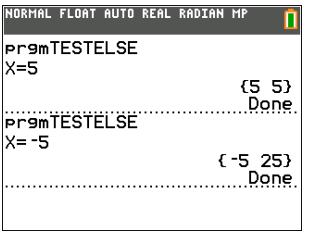
**Else** following **If-Then** executes a group of *commands* if *condition* is false (zero). **End** identifies the end of the group of *commands*.

**:If** *condition*
**:Then**
:*command* (if true)
:*command* (if true)
**:Else**
:*command* (if false)
:*command* (if false)
**:End**
:*command*

**Program**




**Output**



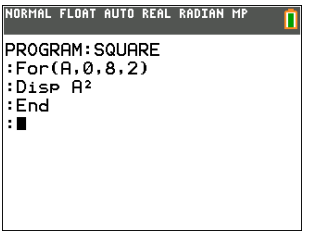

**Note**: Press ENTER to repeat the program.

**For(**

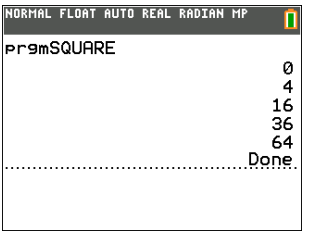
**For(** loops and increments. It increments *variable* from *begin* to *end* by *increment*. *increment* is optional (default is 1) and can be negative (*end*<*begin*). *end* is a maximum or minimum value not to be exceeded. **End** identifies the end of the loop. **For(** loops can be nested.

**:For(***variable*,*begin*,*end*[,*increment*]**)**
:*command* (while *end* not exceeded)
:*command* (while *end* not exceeded)
**:End**
:*command*

**Program**




**Output**

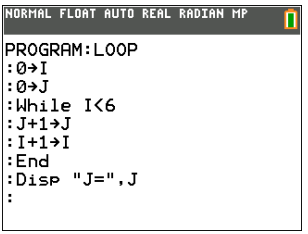


### While

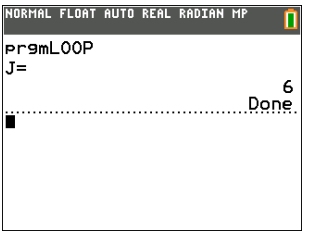
**While** performs a group of *commands* while *condition* is true. *condition* is frequently a relational test. *condition* is tested when **While** is encountered. If *condition* is true (nonzero), the program executes a group of *commands*. **End** signifies the end of the group. When *condition* is false (zero), the program executes each *command* following **End**. **While** instructions can be nested.

**:While** *condition*
:*command* (while *condition* is true)
:*command* (while *condition* is true)
**:End**
:*command*

**Program**




**Output**



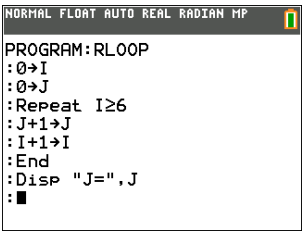

---

### Repeat

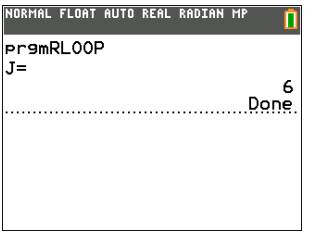
**Repeat** repeats a group of *commands* until *condition* is true (nonzero). It is similar to **While**, but *condition* is tested when **End** is encountered; therefore, the group of *commands* is always executed at least once. **Repeat** instructions can be nested.

**:Repeat** *condition*
:*command* (until *condition* is true)
:*command* (until *condition* is true)
**:End**
:*command*

**Program**




**Output**




---

**End**

**End** identifies the end of a group of *commands*. You must include an **End** instruction at the end of each **For(**, **While**, or **Repeat** loop. Also, you must paste an **End** instruction at the end of each **If-Then** group and each **If-Then-Else** group.

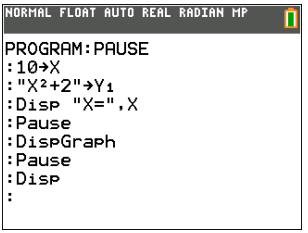
### Pause

**Pause** suspends execution of the program so that you can see answers or graphs. During the pause, the pause indicator is on in the top-right corner.

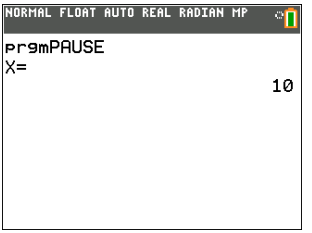
- **Pause** without an argument temporarily pauses the program. If the **DispGraph** or **Disp** instruction has been executed, the appropriate screen is displayed. Press ENTER to resume execution.
- **Pause** with *value* displays *value* on the current home screen. *value* can be scrolled. **Pause** *value*. Press ENTER to resume execution.
- **Pause** with *value* and *time* displays value on the current home screen and execution of the program continues for the time period specified. For time only, use Pause *"",time* where the value is a blank string. Time is in seconds. **Pause** *value,time*.

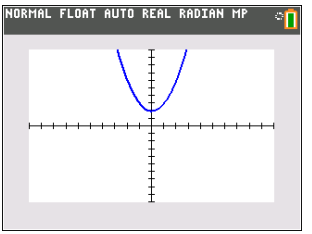
**Note:** When using TI Connect CE Program Editor, Pause must have a space after the command even if no argument is entered.

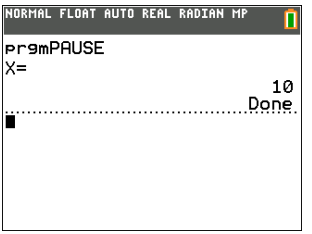
**Program**




**Output**



## Lbl, Goto

---

### Lbl

**Lbl** (label) and **Goto** (go to) are used together for branching.

**Lbl** specifies the *label* for a command. *label* can be one or two characters (A through Z, 0 through 99, or θ).

**Lbl** *label*

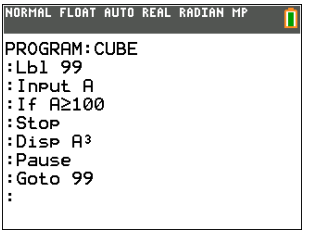
---

### Goto

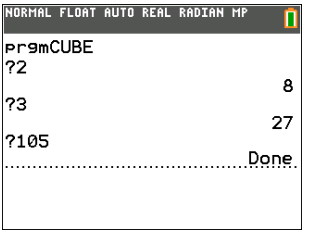
**Goto** causes the program to branch to *label* when **Goto** is encountered.

**Goto** *label*

**Program**




**Output**



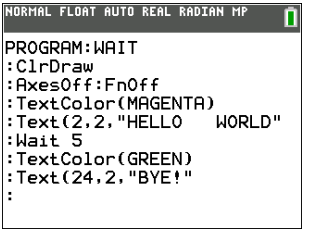

---

## Wait

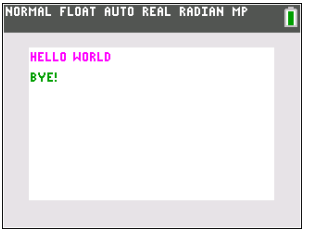
**Wait** suspends execution of a program for a given time. Maximum time is 100 seconds. During the wait time, the busy indicator is on in the top-right corner of the screen.

Wait *time*

**Program**




**Output: “Bye!” displays after 5 seconds.**

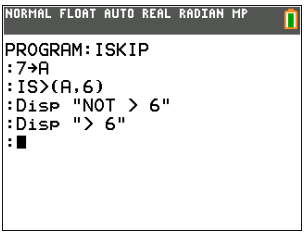


### IS>(

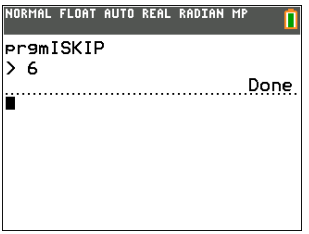
**IS>(** (increment and skip) adds 1 to *variable*. If the answer is > *value* (which can be an expression), the next *command* is skipped; if the answer is ≤ *value*, the next *command* is executed. *variable* cannot be a system variable.

**:IS>(***variable*,*value***)**
:*command* (if answer *value*)
:*command* (if answer > *value*)

**Program**




**Output**



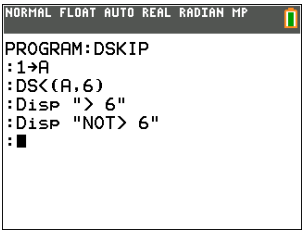

**Note: IS>(** is not a looping instruction.

### DS<(

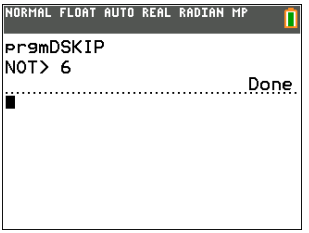
**DS<(** (decrement and skip) subtracts 1 from *variable*. If the answer is < *value* (which can be an expression), the next *command* is skipped; if the answer is ≥ *value*, the next *command* is executed. *variable* cannot be a system variable.

**:DS<(***variable*,*value***)**
:*command* (if answer *value*)
:*command* (if answer < *value*)

**Program**




**Output**




**Note: DS<(** is not a looping instruction.

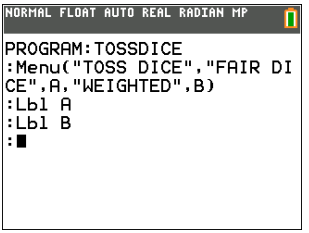
**Menu(**

**Menu(** sets up branching within a program. If **Menu(** is encountered during program execution, the menu screen is displayed with the specified menu items, the pause indicator is on, and execution pauses until you select a menu item.

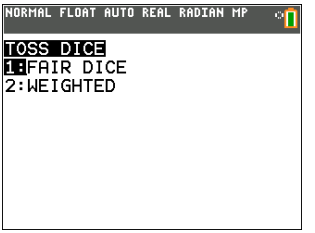
The menu *title* is enclosed in quotation marks ( " ). Up to nine pairs of menu items are allowed. Each pair comprises a *text* item (also enclosed in quotation marks) to be displayed as a menu selection, and a *label* item to which to branch if you select the corresponding menu selection.

**Menu("***title***","***text1***",***label1***,"***text2***",***label2***, . . .)**

**Program**




**Output**




The program above pauses until you select **1** or **2**. If you select **2**, for example, the menu disappears and the program continues execution at **Lbl B**.

---

**prgm**

Use **prgm** to execute other programs as subroutines. When you select **prgm**, it is pasted to the cursor location. Enter characters to spell a program *name*. Using **prgm** is equivalent to selecting existing programs from the **PRGM EXEC** menu; however, it allows you to enter the name of a program that you have not yet created.

**prgm***name*

**Note:** You cannot directly enter the subroutine name when using **RCL**. You must paste the name from the **PRGM EXEC** menu.

---

**Return**

**Return** quits the subroutine and returns execution to the calling program, even if encountered within nested loops. Any loops are ended. An implied **Return** exists at the end of any program that is called as a subroutine. Within the main program, **Return** stops execution and returns to the home screen.

---

### Stop

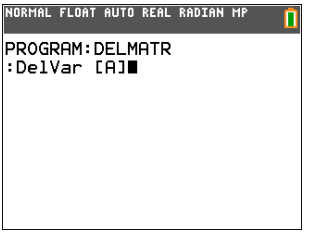
**Stop** stops execution of a program and returns to the home screen. **Stop** is optional at the end of a program.

---

### DelVar

**DelVar** deletes from memory the contents of *variable*.

**DelVar** *variable*




---

### GraphStyle(

**GraphStyle(** designates the style of the graph to be drawn. *function#* is the number of the Y= function name in the current graphing mode. *graphstyle* is a number from 1 to 7 that corresponds to the graph style, as shown below.

**1** = (Thin)
**2** = (Thick)
**3** = (Shade above)
**4** = (Shade below)
**5** = (Path)
**6** = (Animate)
**7** = (Dot-Thick)
**8** = (Dot-Thin)

**GraphStyle(***function#*,*graphstyle*)

For example, **GraphStyle(1,5)** in **Func** mode sets the graph style for Y1 to (path; 5).

Not all graph styles are available in all graphing modes.

---

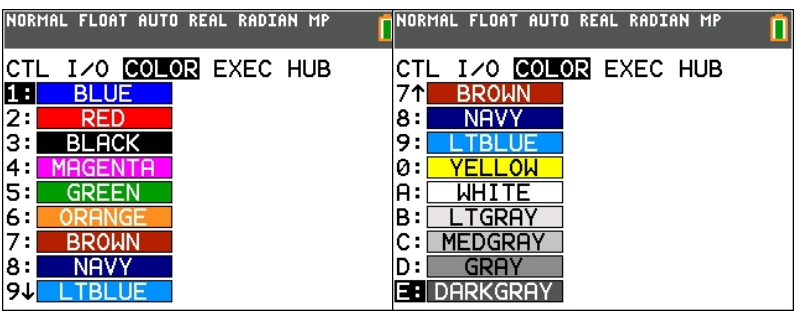
### GraphColor

**GraphColor(** designates the color of the graph to be drawn. *function#* is the number of the Y= function name in the current graphing mode. *color#* is a number from 10 to 24 that corresponds to the graph color, as shown in the table below:

| Color Number | Color Name |
|---|---|
| 10 | BLUE |

| | |
|---|---|
| 11 | RED |
| 12 | BLACK |
| 13 | MAGENTA |
| 14 | GREEN |
| 15 | ORANGE |
| 16 | BROWN |
| 17 | NAVY |
| 18 | LTBLUE |
| 19 | YELLOW |
| 20 | WHITE |
| 21 | LTGRAY |
| 22 | MEDGRAY |
| 23 | GRAY |
| 24 | DARKGRAY |

You can also choose a color name in the VARS menu (**color** sub-menu).




**GraphColor(***function#,color#***)**

For example,**GraphColor(2, 4) or GraphColor(2, MAGENTA).**

---

**OpenLib(**

Extends TI-Basic (not available)

---

**ExecLib(**

Extends TI-Basic (not available)

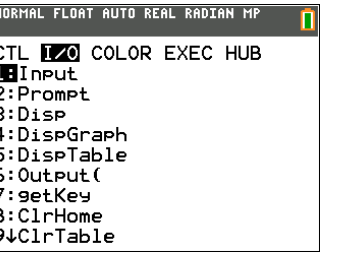
# PRGM I/O (Input/Output) Instructions

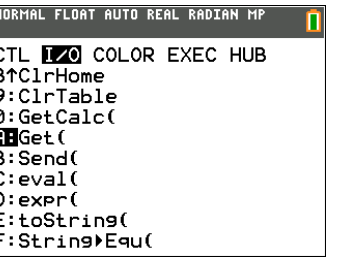
This section describes the **PRGM I/O** (Input/Output) Instructions.

## *PRGM I/O Menu*

To display the **PRGM I/O** (program input/output) menu, press [PRGM] [▶] from within the program editor only.

**Important Tip**: To quickly find a command, use [alpha] [▲] or [alpha] [▼] to page through screens.




**CTRL** **I/O** **COLOR** **EXEC** **HUB**

| | | Description |
|---|---|---|
| 1: | Input | Enters a value or uses the cursor. |
| 2: | Prompt | Prompts for entry of variable values. |
| 3: | Disp | Displays text, value, or the home screen. |
| 4: | DispGraph | Displays the current graph. |
| 5: | DispTable | Displays the current table. |
| 6: | Output( | Displays text at a specified position. |
| 7: | getKey | Checks the keyboard for a keystroke. |
| 8: | ClrHome | Clears the display. |
| 9: | ClrTable | Clears the current table. |
| 0: | GetCalc( | Gets a variable from another TI-84 Plus CE. |
| A: | Get( | Gets a variable from a USB connected device. The device being used governs how to use this functionality. |
| B: | Send( | Sends a variable to a USB connected device. The device being |

| | | |
|---|---|---|
| | | used governs how to use this functionality. |
| C: | eval( | Returns an evaluated expression as a string with 8 significant digits. |
| D: | expr( | Converts the character string contained in *string* to an expression and executes it. |
| E: | toString( | Converts value to a string where *value* can be real, complex, an evaluated expression, list, or matrix. |
| F: | String▸Equ( | .NEW |

**Note**: Press + when a command is highlighted in a menu to use the syntax help for your programming.

These instructions control input to and output from a program during execution. They allow you to enter values and display answers during program execution.

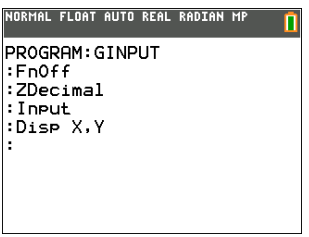
To return to the program editor without selecting an item, press CLEAR.

**Displaying a Graph with Input**

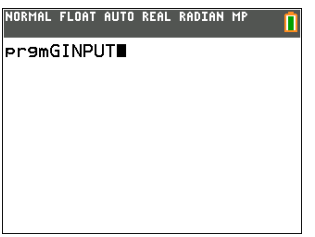
**Input** without a variable displays the current graph. You can move the free-moving cursor, which updates X and Y (and R and θ for **PolarGC** format). The pause indicator is on. Press ENTER to resume program execution.

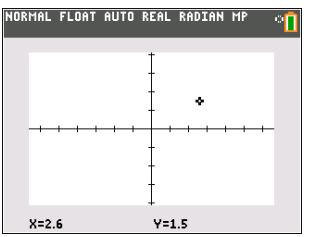
**Input**

**Program**



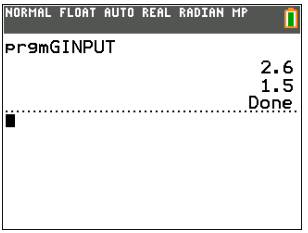

**Output**



**Program**

**Output**




---

### Storing a Variable Value with Input

**Input** with *variable* displays a ? (question mark) prompt during execution. *variable* may be a real number, complex number, list, matrix, string, or Y= function. During program execution, enter a value, which can be an expression, and then press ENTER. The value is evaluated and stored to *variable*, and the program resumes execution.

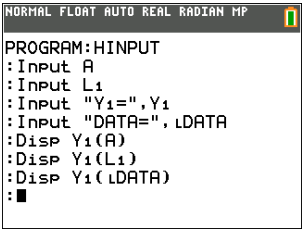
---

### Input [*variable*]

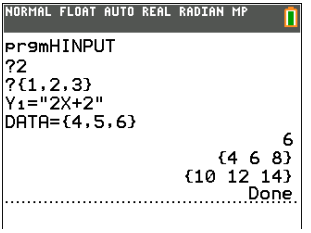
You can display *text* or the contents of **Str***n* (a string variable) of up to 26 characters as a prompt. During program execution, enter a value after the prompt and then press ENTER. The value is stored to *variable*, and the program resumes execution.

**Input** ["*text*",*variable*]
**Input** [**Str***n*,*variable*]

**Program**




**Output**




**Note:** When a program prompts for input of lists and **Y***n* functions during execution, you must include the braces ( **{ }** ) around the list elements and quotation marks ( " ) around the expressions.

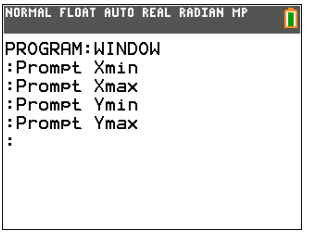
---

**Prompt**

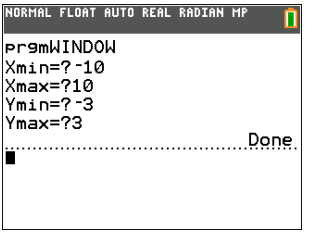
During program execution, **Prompt** displays each *variable*, one at a time, followed by =?. At each prompt, enter a value or expression for each *variable*, and then press ENTER. The values are stored, and the program resumes execution.

**Prompt** *variableA*[,*variableB*,...,*variable n*]

**Program**




**Output**




**Note:** Y= functions are not valid with **Prompt**.

---

**Disp**

***Displaying the Home Screen***

**Disp** (display) without a value displays the home screen. To view the home screen during program execution, follow the **Disp** instruction with a **Pause** instruction.

***Displaying Values and Messages***

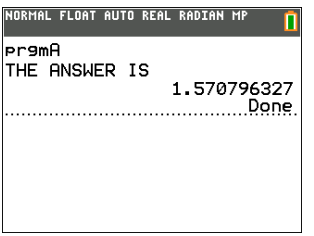
**Disp** with one or more *values* displays the value of each.

**Disp** [*valueA*,*valueB*,*valueC*,...,*value n*]

- If *value* is a variable, the current value is displayed.
- If *value* is an expression, it is evaluated and the result is displayed on the right side of the next line.
- If *value* is text within quotation marks, it is displayed on the left side of the current display line. → is not valid as text.

**Program**


**Output**



If **Pause** is encountered after **Disp**, the program halts temporarily so you can examine the screen. To resume execution, press ENTER.

**Note:** If a matrix or list is too large to display in its entirety, ellipses (**…**) are displayed in the last column, but the matrix or list cannot be scrolled. To scroll, use **Pause** *value*.

---

**DispGraph**

**DispGraph** (display graph) displays the current graph. If **Pause** is encountered after **DispGraph**, the program halts temporarily so you can examine the screen. Press ENTER to resume execution.

---

**DispTable**

**DispTable** (display table) displays the current table. The program halts temporarily so you can examine the screen. Press ENTER to resume execution.

---

**Output(**

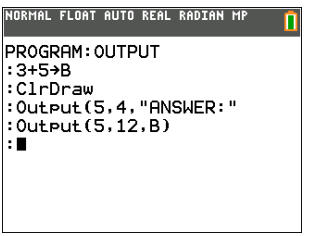
**Output(** displays *text* or *value* on the current home screen beginning at *row* (1 through 10) and *column* (1 through 26), overwriting any existing characters.

**Note:** You may want to precede **Output(** with **ClrHome**.

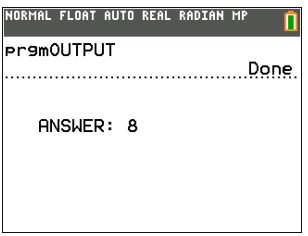
Expressions are evaluated and values are displayed according to the current mode settings. Matrices are displayed in entry format and wrap to the next line. → is not valid as text.

**Output(***row*,*column*,"*text*"**)**
**Output(***row*,*column*,*value***)**

**Program**




**Output**




For **Output(** on a **Horiz** split screen, the maximum value for *row* is 4.

---

### getKey

**getKey** returns a number corresponding to the last key pressed, according to the key code diagram below. If no key has been pressed, **getKey** returns 0. Use **getKey** inside loops to transfer control, for example, when creating video games.

**Program**

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:GETKEY
:While 1
:getKey→K
:While K=0
:getKey→K
:End
:Disp K
:If K=105
:Stop
:End
```

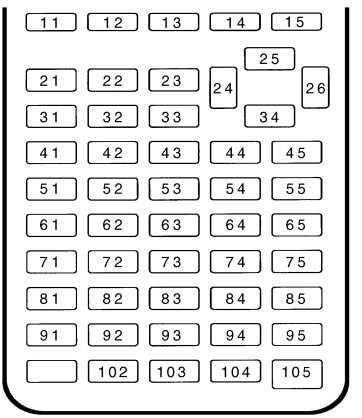
**Output**

```
NORMAL FLOAT AUTO REAL RADIAN MP
prgmGETKEY
                              41
                              42
                              43
                             105
                            Done
```

**Note:** MATH, APPS, PRGM, and ENTER were pressed during program execution.

**Note:** You can press ON at any time during execution to break the program.

**TI-84 Plus CE Key Code Diagram**




---

### ClrHome, ClrTable

**ClrHome** (clear home screen) clears the home screen during program execution.

**ClrTable** (clear table) clears the values in the table during program execution.

**GetCalc(**

**GetCalc(** gets the contents of *variable* on another TI-84 Plus CE and stores it to *variable* on the receiving TI-84 Plus CE. *variable* can be a real or complex number, list element, list name, matrix element, matrix name, string, Y= variable, graph database, or picture.

**GetCalc(***variable*[,*portflag*]**)**

By default, the TI-84 Plus CE uses the USB port if it is connected. If the USB cable is not connected, it uses the I/O port. If you want to specify either the USB or I/O port, use the following portflag numbers:

*portflag*=0 use USB port if connected;
*portflag*=1 use USB port;
*portflag*=2 use I/O port (Ignored when program runs on the TI-84 Plus CE.)

**Note: GetCalc(** does not work between TI-82 and TI-83 Plus or a TI-82, TI-84 Plus and TI-84 Plus CE calculators.

---

**Get(, Send(**

---

**Get(**

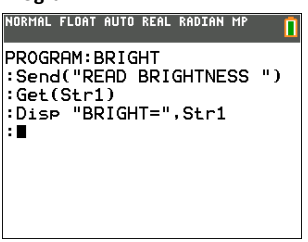
**Get(** Retrieves a value from a connected TI-Innovator™ Hub and stores the data to a variable on the receiving CE calculator.

**Get(***variable***)**

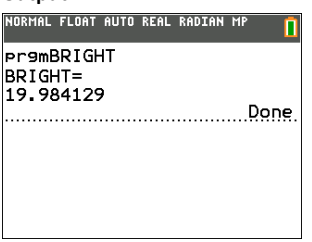
**Notes:**

- Use **GetCalc(** to get data from another CE calculator.
- You can access **Get(**, **Send(** and **GetCalc(** from the CATALOG to execute them from the home screen.

**Program**




**Output**




**TI-Innovator™ Hub Tips:**

**Get(** command definition is specific to the TI-8x calculator and the cable connection via DBus or USB. The CE calculator is USB connectivity only and here, **Get(** is designed for communication with the TI-Innovator™ Hub.

See also **Send(** and **eval(**.

See the HUB menu for TI-Innovator™ Hub details.

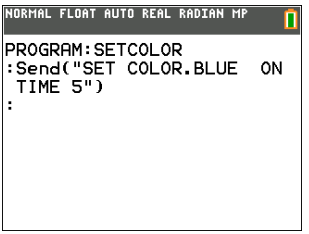
---

**Send(**

Sends one or more TI-Innovator™ Hub commands to a connected hub.

**Send(***string***)**

**Program**




**Output**

Turns blue LED on for 5 seconds when sent to connected TI-Innovator™ Hub.

**TI-Innovator™ Hub Tips:**

See also **eval(** and **Get(** commands related to the **Send(** command.

TI-Innovator™ Hub commands are supported in the HUB submenu in the CE OS v.5.2 program editor.

See the HUB menu for TI-Innovator™ Hub details.

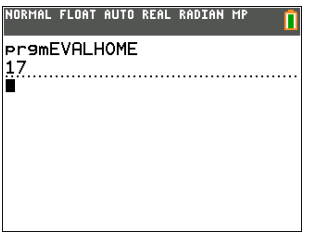
**eval(**

eval( returns an evaluated expression as a string with 8 significant digits. The expression must simplify to a real expression.

eval(*expression*)

**Program**


**Output**



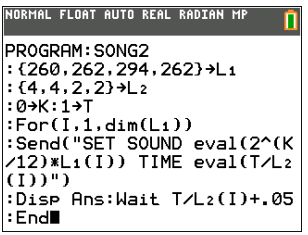

**TI-Innovator™ Hub Tips:**

**eval(** may be used within a string in the **Send(** command. The evaluated *expression* replaces *eval*(*expression*) with the result as characters within the string.

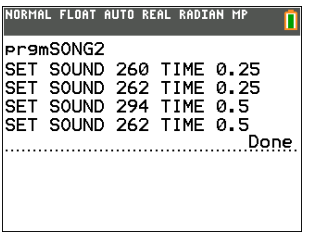
For debugging purposes, using the command line Disp Ans immediately after a command line using **Send(** displays the complete string being sent.

See the HUB menu for TI-Innovator™ Hub details.

**Program**




**Output: Using Disp Ans after Send( command line.**

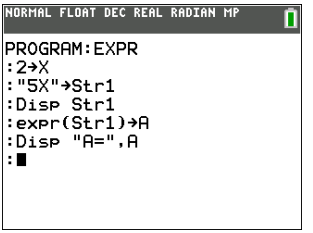


**expr(**

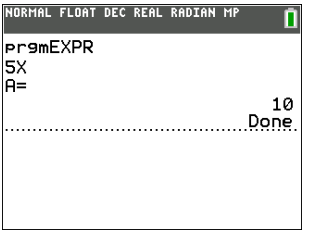
Converts the character string contained in *string* to an expression and executes the expression. *string* can be a string or a string variable.

**expr(***string***)**

**Program**




**Output**



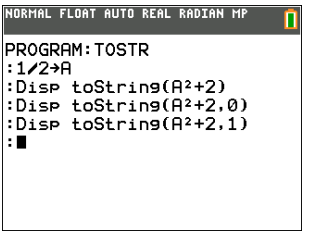

---

**toString(**

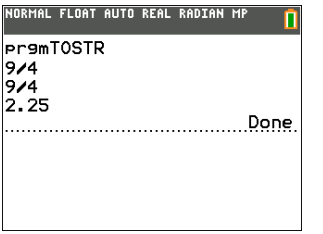
Converts value to a string where *value* can be real, complex, an evaluated expression, list, or matrix. String *value* displays in classic *format (0)* following the mode setting AUTO/DEC or in decimal *format (1)*.

**toString(***value*[,*format*]**)**

**Program**




**Output**



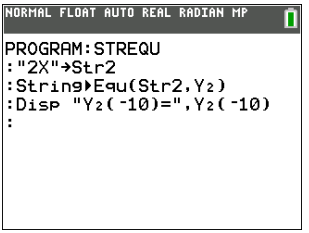

---

**String▶Equ(**

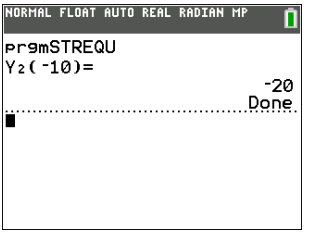
**String▶Equ(** converts *string* into an equation and stores the equation to **Y***n*. string can be a string or string variable. **String▶Equ(** is the inverse of **Equ▶String(**.

**String▶Equ(***string*,**Y***n***)**

**Program**




**Output**



# PRGM COLOR Instructions

This section describes the **COLOR** menu and the color numbers to use as arguments where setting color is an option such as **GraphColor(**.

You can paste the color token, such as **BLUE**, or use the color number, such as **10**, shown in the table below.

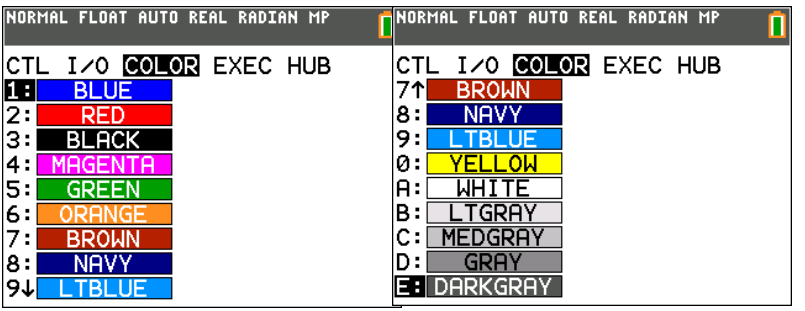
## *PRGM COLOR Menu*

To display the **PRGM COLOR** menu, press PRGM ▸ from within the program editor only.

**CTRL** **I/O** **COLOR** **EXEC** **HUB**

| | | Description |
|---|---|---|
| 1: | BLUE | #color = 10 |
| 2: | RED | #color = 11 |
| 3: | BLACK | #color = 12 |
| 4: | MAGENTA | #color = 13 |
| 5: | GREEN | #color = 14 |
| 6: | ORANGE | #color = 15 |
| 7: | BROWN | #color = 16 |
| 8: | NAVY | #color = 17 |
| 9: | LTBLUE | #color = 18 |
| 0: | YELLOW | #color = 19 |
| A: | WHITE | #color = 20 |
| B: | LTGRAY | #color = 21 |
| C: | MEDGRAY | #color = 22 |
| D: | GRAY | #color = 23 |
| E: | DARKGRAY | #color = 24 |

**Note:** You can also choose a color name in the vars menu (COLOR sub-menu).



# PRGM EXEC Instructions

## *Calling Other Programs as Subroutines*

On the TI-84 Plus CE, any stored program can be called from another program as a subroutine. Enter the name of the program to use as a subroutine on a line by itself.

## *Calling a Program from Another Program*

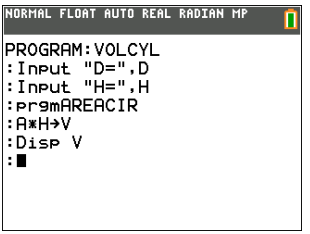
You can enter a program name on a command line in either of two ways.

- Press [PRGM] [◂] to display the **PRGM EXEC** menu and select the name of the program **prgm***name* is pasted to the current cursor location on a command line.
- Select **prgm** from the **PRGM CTL** menu, and then enter the program name.

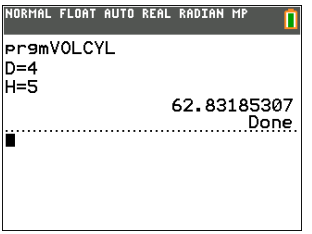
  **prgm***name*

When **prgm***name* is encountered during execution, the next command that the program executes is the first command in the second program. It returns to the subsequent command in the first program when it encounters either **Return** or the implied **Return** at the end of the second program.

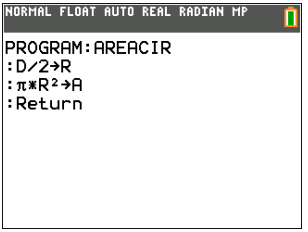
**Program**




**Output**




Subroutine ↓↑




**Notes about Calling Programs**

Variables are global.

*label* used with **Goto** and **Lbl** is local to the program where it is located. *label* in one program is not recognized by another program. You cannot use **Goto** to branch to a *label* in another program.

**Return** exits a subroutine and returns to the calling program, even if it is encountered within nested loops.

# PRGM HUB Instructions

## *TI-Innovator™ HUB Menu Instructions*

This section describes the TI-Innovator™ **HUB** Menu Instructions.

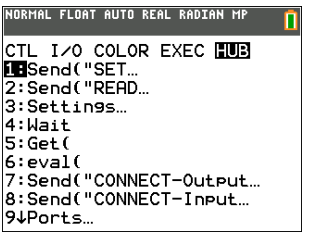
See TI-Innovator™ System activities for details and parameter values for specific sensors and controls. This section describes the instructions or commands contained in the TI-Innovator™ HUB menu and how the commands paste to the program editor.

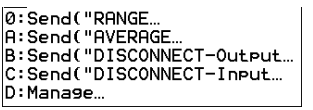
## *TI-Innovator™ HUB Menu*

To display the TI-Innovator™ **HUB** menu, press PRGM from the program editor only.

**Important Tip**: To quickly find a command, use alpha ▲ or alpha ▼ to page through screens.

- If A-lock is **on**, then ▲ and ▼ will page through screens in menus and the program edit screen.
- After entering alpha characters, remember to **turn off** A-lock to avoid unexpected paging of screens.




**Note:** All TI-Innovator™ Hub command can be entered character by character as well.

TI-Basic commands such as **Send(**, **Get(**, **Wait**, and **eval(** must be pasted as tokens from the menus.

**CTRL** **I/O** **COLOR** **EXEC** **HUB**

| | | Description |
|---|---|---|
| 1: | Send("SET… | Builds out a **Send(** command to paste to editor |
| 2: | Send("READ… | Builds out a **Send(** command to paste to editor |
| 3: | Settings… | Pastes a TI-Innovator™ Hub command to editor |
| 4: | Wait | Pastes a TI-Basic command to editor |
| 5: | Get( | Pastes a TI-Basic command to editor |
| 6: | eval( | Pastes a TI-Basic command to editor |
| 7: | Send("CONNECT-OUTPUT | Builds out a **Send(** command to paste to editor |
| 8: | Send("CONNECT-INPUT | Builds out a **Send(** command to paste to editor |
| 9: | Ports… | Pastes a TI-Innovator™ Hub command to editor |
| 0: | Send("RANGE… | Builds out a **Send(** command to paste to editor |

| | | |
|---|---|---|
| A: | Send("AVERAGE... | Builds out a **Send(** command to paste to editor |
| B: | Send("DISCONNECT-OUTPUT | Builds out a **Send(** command to paste to editor |
| C: | Send("DISCONNECT-INPUT | Builds out a **Send(** command to paste to editor |
| D: | Manage... | Pastes several commands (:) to editor |

For Catalog Help when using the commands **eval(**, **Get(**, or **Wait**, press +.

To return to the program editor without selecting an item, press CLEAR until the cursor returns to the program editor.

**Warning:** Do not press CLEAR repeatedly unless you are viewing the screen navigation. Once the menus are cleared, pressing clear again may clear an entire line of your program. There is no undo in the program editor.

---

**Before you begin**

This section provides descriptions of how each menu item pastes to the program editor. For specific TI-Innovator™ System information dealing with syntax and parameters for each sensor, see specific information in the TI-Innovator™ System activities and kits. You may also type the TI-Innovator™ Hub command (only) letter by letter using the [alpha] key, [ " ], [ _ ], etc.

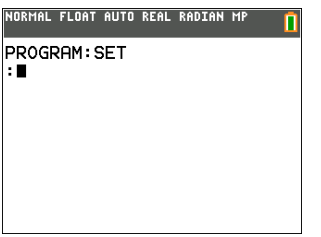
Extra spaces are pasted for your convenience. TI-Innovator™ Hub sketch will ignore extra spaces within quotation marks in a **Send(** command. However, when you run your program, extra spaces cannot be at the end of command lines and will give you a syntax error. If you get a syntax error at the end of a line, check for extra spaces and delete.

***How does the Send( command build out a TI-Innovator™ Hub command from the HUB menu?***

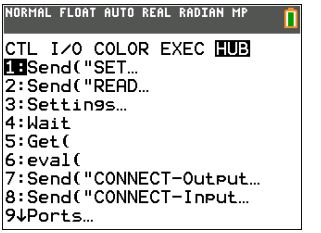
From the **HUB** menu, select a **Send(** command. The next screen will give you options for that format of **Send(**.

**Example:** To paste **Send("SET COLOR.RED** to the program editor, follow these steps.

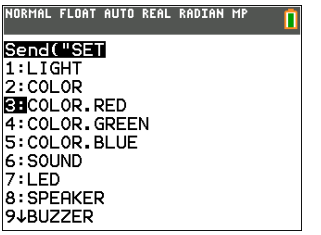
1. With cursor on a command line in the program editor, press prgm to get to the programming command menus.



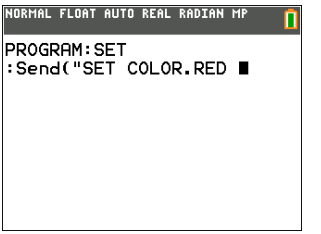
2. Press ◂ to select the **HUB** menu.
   Select **1:Send("SET…**
   The **"…"** indicates there is another menu of options.




3. Select **3:COLOR.RED**.




4. The entire **Send(** command line pastes to the program editor.
   Repeat to select more TI-Innovator™ Hub commands.
   Use alpha **["]** and **[ ) ]** to complete the **Send(** command when appropriate.

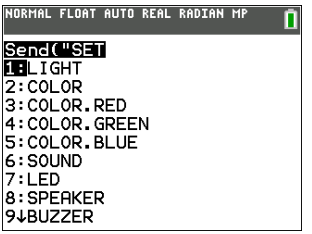


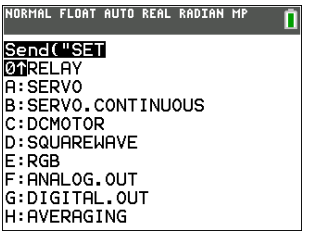

**Note:** All TI-Innovator™ Hub commands using **Send(** within quote marks can be typed in using the [alpha] keys on the keypad.

For colors, do not use the COLOR token command from the COLOR menu when communicating with TI-Innovator.™ Hub.

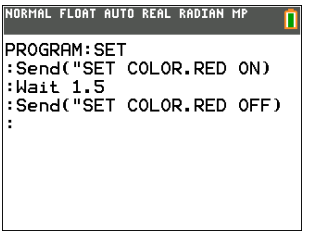
**Send("Set...**

The **SET** command instructs the "TI-Innovator™ Hub sketch to **SET** the value of the specified object. It supports all of the 'named' objects.




**Program**




**Output**

**Example:** This switches a red **LED** on for 1.5 seconds and then switches it off.

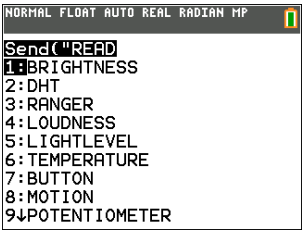
**Note:** The power **LED** is green.

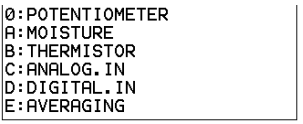
The **ON** and **OFF** command can be typed in or are found in the **Settings...** menu item in the **HUB** menu.

Use [alpha] [ _ ] for space as needed.

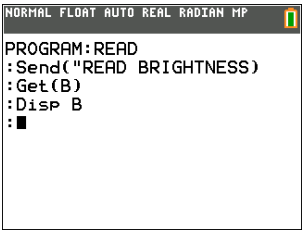
**Send("READ...**

The '**READ**' command is to instruct the TI-Innovator™ Hub sketch to read the value from the specified port/pin/object. It supports all of the 'named' objects. It can also be used with 'raw' pin addresses. It needs to be followed by a '**Get(** command to actually transfer the information to a variable to use or display the variable value.

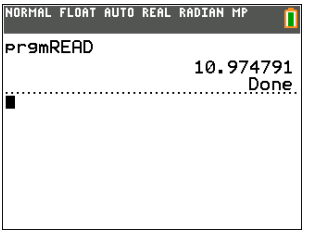



**Program**

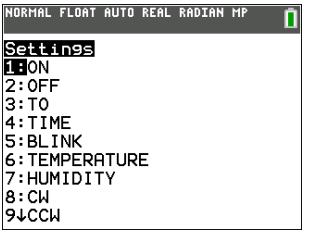



**Output**



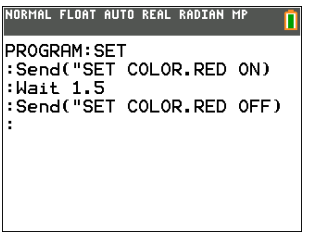
**Settings...**

**Settings** menu contains operations to set the state of digital and analog pin operations such as the **LED** in the TI-Innovator™ Hub or a connected servo motor movement to states such as **ON**, **OFF**, **CW** (clockwise), and **CCW** (counterclockwise).
See TI-Innovator™ System activity kits for more details.





**Program**




**Output**

**Example:** This switches a red **LED** on for1.5 seconds and then switches it off.

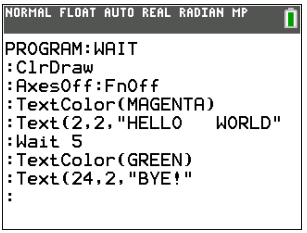
**Reminder:** The power **LED** is green.

### Wait

**Wait** suspends execution of a program for a given time. Maximum time is 100 seconds. During the wait time, the busy indicator is on in the top-right corner of the screen.

Wait *time*

**Program**




**Output: "Bye!" displays after 5 seconds.**


**TI-Innovator™ Hub Tips:**

**Wait** may be used in TI-Innovator™ Hub programs to allow time for sensor or control communications prior to the program executing the next command line.

**Get(**

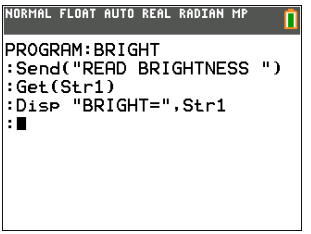
**Get(** Retrieves a value from a connected TI-Innovator™ Hub and stores the data to a variable on the receiving CE calculator.

**Get(***variable***)**

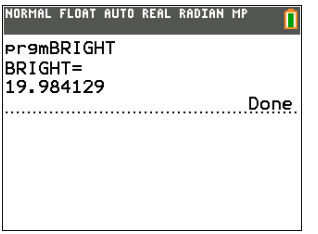
**Notes:**

- Use **GetCalc(** to get data from another CE calculator.
- You can access **Get(**, **Send(** and **GetCalc(** from the CATALOG to execute them from the home screen.

**Program**




**Output**




**TI-Innovator™ Hub Tips:**

**Get(** command definition is specific to the TI-8x calculator and the cable connection via DBus or USB. The CE calculator is USB connectivity only and here, **Get(** is designed for communication with the TI-Innovator™ Hub.

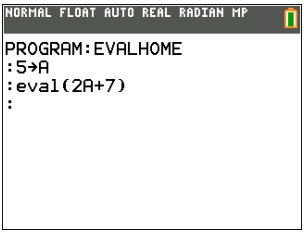
See also **Send(** and **eval(**.

**eval(**

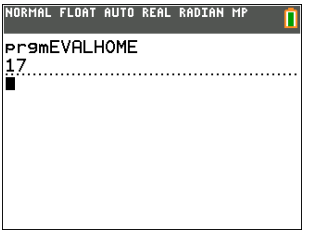
**eval(** returns an evaluated expression as a string with 8 significant digits. The expression must simplify to a real expression.

**eval(***expression***)**

**Program**




**Output**

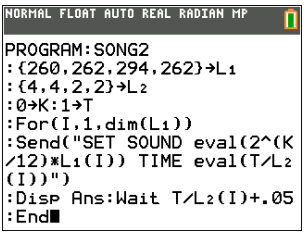



**TI-Innovator™ Hub Tips:**

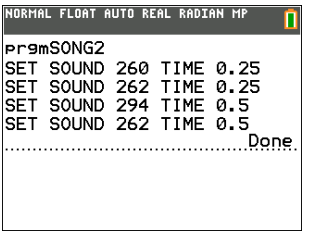
**eval(** may be used within a string in the **Send(** command. The evaluated *expression* replaces **eval(***expression*) with the result as characters within the string

For debugging purposes, using the command line Disp Ans immediately after a command line using **Send(** displays the complete string being sent.

**Program**

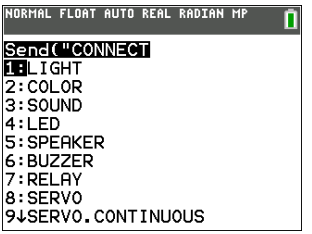


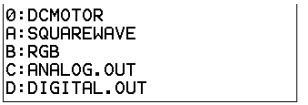

**Output: Using Disp Ans after Send( command line.**



**Send("CONNECT-OUTPUT...**

CONNECT (Output) associates a given control or sensor with a pin or port on the TI-Innovator™.



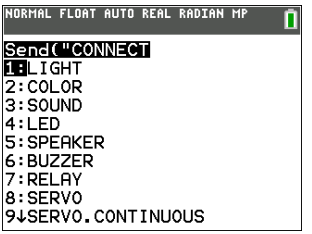

**Program**

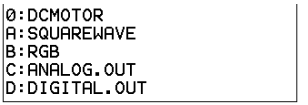

**Output**

Connects servo motor to OUT3.

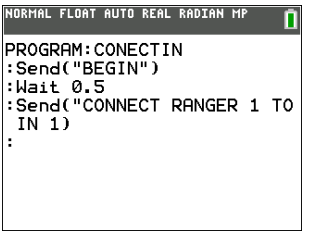
**Send("CONNECT-INPUT...**

CONNECT (Input) associates a given control or sensor with a pin or port on the TI-Innovator™ Hub.




**Program**

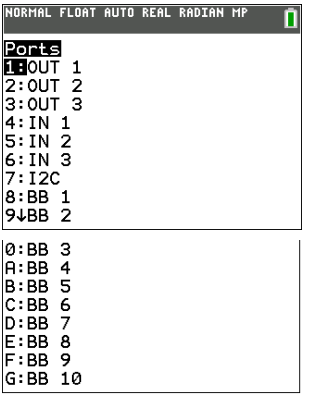



**Output**

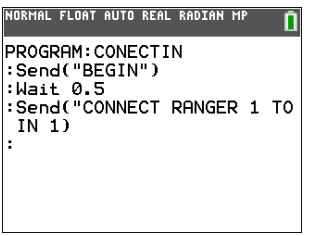
Connects an external range finder to IN 1.

**Ports…**

Ports menu lists available ports to connect such as input, output or to a breadboard.




**Program**



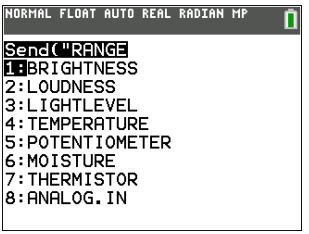

**Output**

Connects an external range finder to IN 1.

**Note:** Extra spaces paste such as the space in “IN 1.” “IN1” is also accepted by the TI-Innovator™ Hub sketch on TI-Innovator™.

**Send("RANGE…**

Changes or sets the range to a user-selected range from minimum to a maximum value.




**Syntax Examples:**

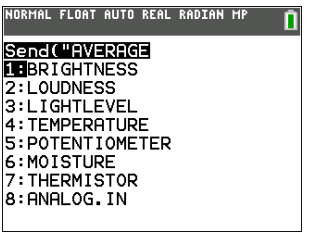
Send(“RANGE BRIGHTNESS minimum maximum”)

Send(“RANGE LIGHTLEVEL # minimum maximum”)

---

**Send("AVERAGE…**

The AVERAGE command is used to set the number of samples taken to represent an average single sensor reading.




**Syntax Examples:**

Send(“AVERAGE BRIGHTNESS number”)

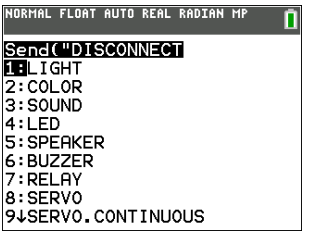
Send(“AVERAGE LIGHTLEVEL # number”)

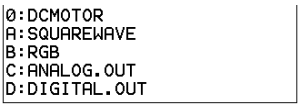
Where “number” is the number of readings to average.

---

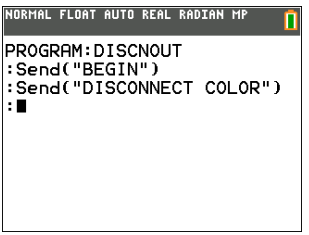
**Send("DISCONNECT-OUTPUT…**

DISCONNECT (Output) breaks the association between a specific control or sensor from a pin or port on the TI-Innovator™.




**Program**

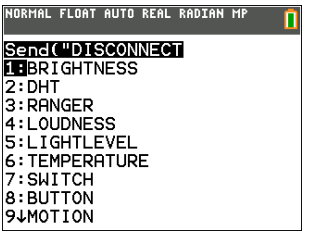


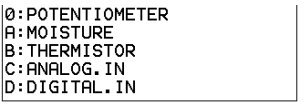

**Output**

Disconnects the on-board RGB LED from use.

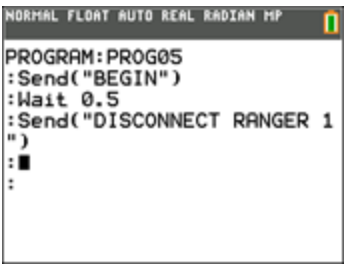
**Send("DISCONNECT-Input…**

DISCONNECT (Input) breaks the association between a specific control or sensor from a pin or port on the TI-Innovator™.




**Program**



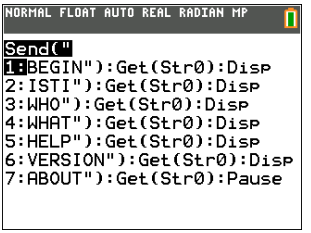

**Output**

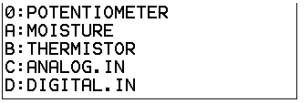
Disconnects range sensor from use.

**Manage…**

The Manage menu pastes a **Send(** command with the following management items. Str0 is displayed on Home Screen with information if requested in the command.

- **BEGIN** – Disconnects all connected sensors and controls. **Send("BEGIN")** may be needed in a TI-Innovator™ Hub program to re-initialize a sensor or control prior to sending a command to that sensor or control.
- **ISTI** – Responds with TI STEM
- **WHO** – Responds with TI-Innovator™ Hub ON MSP432
- **WHAT** – Responds with TI-Innovator™ Hub
- **HELP** – Responds with USE HELP COMMAND FOR DETAILS
- **VERSION** – Responds with TI-Innovator™ Hub version number
- **ABOUT** – Responds with TI-Innovator™ Hub ©2016 Texas Instruments




**Note:** The [ **:** ] is used to sequence command lines on one command line. The **Manage…** menu pastes a convenient set of commands to then display the information in **Str0** on the home screen.

# Running an Assembly Language Program

You can run programs written for the TI-84 Plus CE 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.

**Note:** Because an assembly language program has greater control over the calculator, if your assembly language program has error(s), it may cause your calculator to reset and lose all data, programs, and applications stored in memory.

When you download an assembly language program, it is stored among the other programs as a **PRGM** menu item. You can:

- Transmit it using the TI-84 Plus CE communication link
- Delete it using the Memory Management/ Delete screen

To run an assembly Program, the syntax is: **Asm(***assemblyprgmname***)**

If you write an assembly language program, use the two instructions below from the CATALOG to identify and compile the program.

| Instructions | Comments |
|---|---|
| **AsmComp (***prgmASM1, prgmASM2***)** | Compiles an assembly language program written in ASCII and stores the hex version |
| **Asm84CEPrgm** | Identifies an assembly language program; must be entered as the first line of an assembly language program |

To compile an assembly program that you have written:

1. Follow the steps for writing a program (16-4) but be sure to include **Asm84CEPrgm** as the first line of your program.
2. From the home screen, press [2nd] [CATALOG] and then select **AsmComp(** to paste it to the screen.
3. Press [PRGM] to display the **PRGM EXEC** menu.
4. Select the program you want to compile. It will be pasted to the home screen.
5. Press [,] and then select **prgm** from the **CATALOG**.
6. Key in the name you have chosen for the output program.

   **Note:** This name must be unique — not a copy of an existing program name.
7. Press [)] to complete the sequence.

   The sequence of the arguments should be as follows:

   **AsmComp(***prgmASM1, prgmASM2*)
8. Press [ENTER] to compile your program and generate the output program.

**Note**: The TI-84 Plus or TI-84 Plus C Silver Edition **AsmPrgm transfers to the TI-84 Plus CE but fails upon execution.**

# General Information

## *Texas Instruments Support and Service*

**General Information: North and South America**

| | |
|---|---|
| **Home Page:** | education.ti.com |
| **Knowledge Base and e-mail inquiries:** | education.ti.com/support |
| **Phone:** | (800) TI-CARES / (800) 842-2737<br>For North and South America and U.S. Territories |
| **International contact information:** | education.ti.com/support/worldwide |

**For Technical Support**

| | |
|---|---|
| **Knowledge Base and support by e-mail:** | education.ti.com/support or ti-cares@ti.com |
| **Phone (not toll-free):** | (972) 917-8324 |

**For Product (Hardware) Service**

**Customers in the U.S., Canada, Mexico, and U.S. territories:** Always contact Texas Instruments Customer Support before returning a product for service.

**For All Other Countries:**

**For general information**

For more information about TI products and services, contact TI by e-mail or visit the TI Internet address.

| | |
|---|---|
| **E-mail inquiries:** | ti-cares@ti.com |
| **Home Page:** | education.ti.com |

## *Service and Warranty Information*

For information about the length and terms of the warranty or about product service, refer to the warranty statement enclosed with this product or contact your local Texas Instruments retailer/distributor.

# Index

**R**



**S**



**T**



**W**