# Supplemental Documentation for:
# TI-Nspire™ CAS Reference Guide
# TI-Nspire™ Reference Guide

Learn more about TI Technology through the online help at education.ti.com/eguide.

## *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*

# TI-Nspire™ CX II - Draw Commands

This is a supplemental document for the TI-Nspire™ Reference Guide and the TI-Nspire™ CAS Reference Guide. All TI-Nspire™ CX II commands will be incorporated and published in version 5.1 of the TI-Nspire™ Reference Guide and the TI-Nspire™ CAS Reference Guide.

## *Graphics Programming*

New commands have been added on TI-Nspire™ CX II Handhelds and TI-Nspire™ desktop applications for graphics programming.

The TI-Nspire™ CX II Handhelds will switch into this graphics mode while executing graphics commands and switch back to the context in which the program was executed after completion of the program.

The screen will display "Running…" in the top bar while the program is being executed. It will show "Finished" when the program completes. Any key-press will transition the system out of the graphics mode.

- The transition to graphics mode is triggered automatically when one of the Draw (graphics) commands is encountered during execution of the TI Basic program.
- This transition will only happen when executing a program from calculator; in a document or calculator in scratchpad.
- The transition out of graphics mode happens upon termination of the program.
- The graphics mode is only available on the TI-Nspire™ CX II Handhelds and the desktop TI-Nspire™ CX II Handhelds view. This means it is not available in the computer document view or PublishView (.tnsp) on the desktop nor on iOS.
    - If a graphics command is encountered while executing a TI Basic program from the incorrect context, an error message is displayed and the TI Basic program is terminated.

## *Graphics Screen*

The graphics screen will contain a header at the top of the screen that cannot be written to by graphics commands.

The graphics screen drawing area will be cleared (color = 255,255,255) when the graphics screen is initialized.

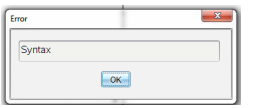
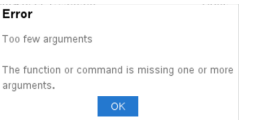
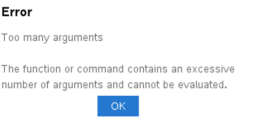
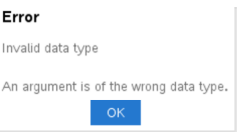
| **Graphics Screen** | **Default** |
|---|---|
| Height | 212 |
| Width | 318 |
| Color | white: 255,255,255 |

## *Default View and Settings*

- The status icons in the top bar (battery status, press-to-test status, network indicator etc.) will not be visible while a graphics program is running.
- Default drawing color: Black (0,0,0)
- Default pen style - normal, smooth
  - Thickness: 1 (thin), 2 (normal), 3 (thickest)
  - Style: 1 (smooth), 2 (dotted), 3 (dashed)
- All drawing commands will use the current color and pen settings; either default values or those which were set via TI-Basic commands.
- Text font is fixed and cannot be changed.
- Any output to the graphics screen will be drawn within a clipping window which is the size of the graphics screen drawing area. Any drawn output that extends outside of this clipped graphics screen drawing area will not be drawn. No error message will be displayed.
- All x,y coordinates specified for drawing commands are defined such that 0,0 is at the top left corner of the graphics screen drawing area.
  - **Exceptions:**
    - **DrawText** uses the coordinates as the bottom left corner of the bounding box for the text.
    - **SetWindow** uses the bottom left corner of the screen
- All parameters for the commands can be provided as expressions that evaluate to a number which is then rounded to the nearest integer.

## *Graphics Screen Errors Messages*

If the validation fails, an error message will display.

| Error Message | Description | View |
|---|---|---|
| Error<br>Syntax | If the syntax checker finds any syntax errors, it displays an error message and tries to position the cursor near the first error so you can correct it. |  |
| Error<br>Too few arguments | The function or command is missing one or more arguments |  |
| Error<br>Too many arguments | The function or command contains and excessive number of arguments and cannot be evaluated. |  |
| Error<br>Invalid data type | An argument is of the wrong data type. |  |

## *Invalid Commands While in Graphics Mode*

Some commands are not allowed once the program switches to graphics mode. If these commands are encountered while in graphics mode and error will be displayed and the program will be terminated.

| Disallowed Command | Error Message |
|---|---|
| **Request** | Request cannot be executed in graphics mode |
| **RequestStr** | RequestStr cannot be executed in graphics mode |
| **Text** | Text cannot be executed in graphics mode |

The commands that print text to the calculator - **disp** and **dispAt** - will be supported commands in the graphics context. The text from these commands will be sent to the Calculator screen (not on Graphics) and will be visible after the program exits and the system switches back to the Calculator app

## C

### Clear

Catalog > CXII

**Clear** *x, y, width, height*

Clears entire screen if no parameters are specified.

If *x*, *y*, *width* and *height* are specified, the rectangle defined by the parameters will be cleared.

```
Clear
```

Clears entire screen

```
Clear 10,10,100,50
```

Clears a rectangle area with top left corner on (10, 10) and with width 100, height 50

## D

### DrawArc

Catalog > CXII

**DrawArc** *x, y, width, height, startAngle, arcAngle*

Draw an arc within the defined bounding rectangle with the provided start and arc angles.

*x, y*: upper left coordinate of bounding rectangle

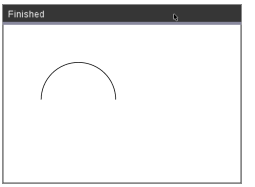
*width, height*: dimensions of bounding rectangle

The "arc angle" defines the sweep of the arc.

These parameters can be provided as expressions that evaluate to a number which is then rounded to the nearest integer.

```
DrawArc 20,20,100,100,0,90
```


```
DrawArc 50,50,100,100,0,180
```




**See Also:** FillArc

### DrawCircle

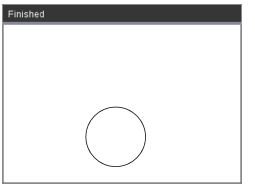
Catalog > CXII

**DrawCircle** *x, y, radius*

*x, y*: coordinate of center

*radius*: radius of the circle

```
DrawCircle 150,150,40
```




**See Also:** FillCircle

## DrawLine

Catalog > CXII

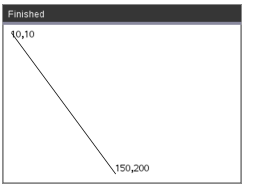
**DrawLine** *x1, y1, x2, y2*

Draw a line from *x1, y1, x2, y2*.

Expressions that evaluate to a number which is then rounded to the nearest integer.

**Screen bounds:** If the specified coordinates causes any part of the line to be drawn outside of the graphics screen, that part of the line will be clipped and no error message will be displayed.

```
DrawLine 10,10,150,200
```




## DrawPoly

Catalog > CXII

The commands have two variants:

**DrawPoly** *xlist, ylist*

or

**DrawPoly** *x1, y1, x2, y2, x3, y3...xn, yn*

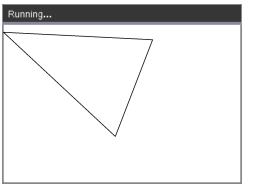
**Note:** DrawPoly *xlist, ylist*
Shape will connect *x1, y1* to *x2, y2*, *x2, y2* to *x3, y3* and so on.

**Note:** DrawPoly *x1, y1, x2, y2, x3, y3...xn, yn*
*xn, yn* will **NOT** be automatically connected to *x1, y1*.

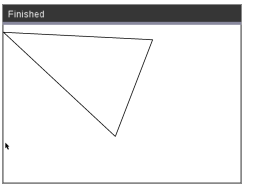
Expressions that evaluate to a list of real floats
*xlist, ylist*

Expressions that evaluate to a single real float
*x1, y1...xn, yn* = coordinates for vertices of polygon

```
xlist:={0,200,150,0}
ylist:={10,20,150,10}
DrawPoly xlist,ylist
```




```
DrawPoly 0,10,200,20,150,150,0,10
```



## DrawPoly

Catalog > CXII

**Note: DrawPoly**: Input size dimensions (width/height) relative to drawn lines. The lines are drawn in a bounding box around the specified coordinate and dimensions such that the actual size of the drawn polygon will be larger than the width and height.

**See Also:** FillPoly

## DrawRect

Catalog > CXII

**DrawRect** *x, y, width, height*

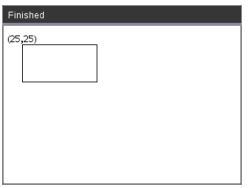
*x*, y: upper left coordinate of rectangle

*width, height*: width and height of rectangle (rectangle drawn down and right from starting coordinate).

**Note:** The lines are drawn in a bounding box around the specified coordinate and dimensions such that the actual size of the drawn rectangle will be larger than the width and height indicate.

**See Also:** FillRect

```
DrawRect 25,25,100,50
```




## DrawText

Catalog > CXII

**DrawText** *x, y, exprOrString1 [,exprOrString2]...*

*x*, *y*: coordinate of text output

Draws the text in *exprOrString* at the specified *x*, *y* coordinate location.

The rules for *exprOrString* are the same as for **Disp** – **DrawText** can take multiple arguments.

```
DrawText 50,50,"Hello World"
```

## FillArc

Catalog > CXII

**FillArc** *x, y, width, height startAngle, arcAngle*

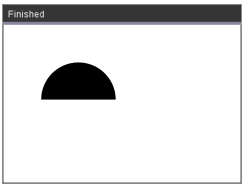
*x, y*: upper left coordinate of bounding rectangle

Draw and fill an arc within the defined bounding rectangle with the provided start and arc angles.

Default fill color is black. The fill color can be set by the SetColor command

The "arc angle" defines the sweep of the arc

```
FillArc 50,50,100,100,0,180
```




## FillCircle

Catalog > CXII

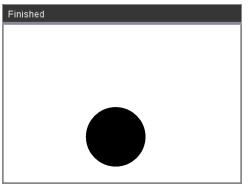
**FillCircle** *x, y, radius*

*x, y*: coordinate of center

Draw and fill a circle at the specified center with the specified radius.

Default fill color is black. The fill color can be set by the SetColor command.

```
FillCircle 150,150,40
```




```
Here!
```

## FillPoly

Catalog > CXII

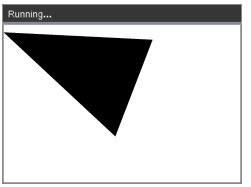
**FillPoly** *xlist, ylist*

or

**FillPoly** *x1, y1, x2, y2, x3, y3...xn, yn*

**Note:** The line and color are specified by SetColor and SetPen

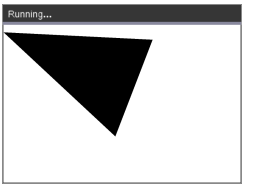
```
xlist:={0,200,150,0}
ylist:={10,20,150,10}
FillPoly xlist,ylist
```



## FillPoly

Catalog > CXII

```
FillPoly 0,10,200,20,150,150,0,10
```




## FillRect

Catalog > CXII

**FillRect** *x, y, width, height*

*x, y*: upper left coordinate of rectangle

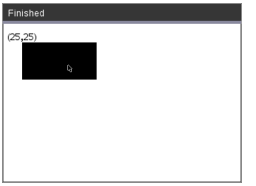
*width, height*: width and height of rectangle

Draw and fill a rectangle with the top left corner at the coordinate specified by (*x,y*)

Default fill color is black. The fill color can be set by the SetColor command

**Note:** The line and color are specified by SetColor and SetPen

```
FillRect 25,25,100,50
```



**getPlatform()** **Catalog > CXII**

**getPlatform()**

getPlatform() "dt"

Returns:
“dt” on desktop software applications
“hh” on TI-Nspire™ CX handhelds
“ios” on TI-Nspire™ CX iPad® app

## PaintBuffer

Catalog > CXII

**PaintBuffer**

Paint graphics buffer to screen

This command is used in conjunction with UseBuffer to increase the speed of display on the screen when the program generates multiple graphical objects.

```
UseBuffer
For n,1,10
x:=randInt(0,300)
y:=randInt(0,200)
radius:=randInt(10,50)
Wait 0.5
DrawCircle x,y,radius
EndFor
PaintBuffer
This program will display all the
10 circles at once.
If the “UseBuffer” command is
removed, each circle will be
displayed as it is drawn.
```

**See Also:** UseBuffer

**PlotXY** *x, y, shape*

*x*, *y*: coordinate to plot shape

*shape* : a number between 1 and 13 specifying the shape

1 - Filled circle

2 - Empty circle

3 - Filled square

4 - Empty square

5 - Cross

6 - Plus

7 - Thin

8 - medium point, solid

9 - medium point, empty

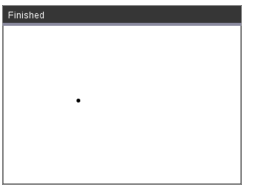
10 - larger point, solid

11 - larger point, empty

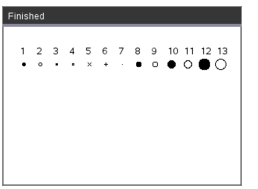
12 - largest point, solid

13 - largest point, empty

```
PlotXY 100,100,1
```




```
For n,1,13
DrawText 1+22*n,40,n
PlotXY 5+22*n,50,n
EndFor
```



## SetColor

Catalog > CXII

**SetColor**

Red-value, Green-value, Blue-value

Valid values for red, green and blue are between 0 and 255

Sets the color for subsequent Draw commands

```
SetColor 255,0,0
DrawCircle 150,150,100
```


## SetPen

Catalog > CXII

**SetPen**

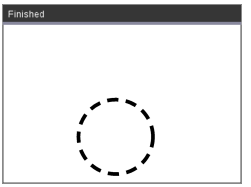
thickness, style

thickness: 1 <= thickness <= 3|1 is thinnest, 3 is thickest

style: 1 = Smooth, 2 = Dotted, 3 = Dashed

Sets the pen style for subsequent Draw commands

```
SetPen 3,3
DrawCircle 150,150,50
```




## SetWindow

Catalog > CXII

**SetWindow**

xMin, xMax, yMin, yMax

Establishes a logical window that maps to the graphics drawing area. All parameters are required.

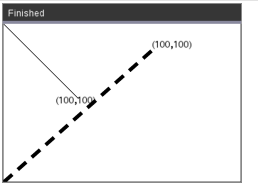
If the part of drawn object is outside the window, the output will be clipped (not shown) and no error message is displayed.

```
SetWindow 0,160,0,120
```

will set the output window to have 0,0 in the bottom left corner with a width of 160 and a height of 120

```
DrawLine 0,0,100,100
SetWindow 0,160,0,120
SetPen 3,3
DrawLine 0,0,100,100
```

If xmin is greater than or equal to xmax or ymin is greater than or equal to ymax, an error message is shown.




Any objects drawn before a SetWindow command will not be re-drawn in the new configuration.

To reset the window parameters to the default, use:

SetWindow 0,0,0,0

## UseBuffer

**Catalog > CXII**

**UseBuffer**

Draw to an off screen graphics buffer instead of screen (to increase performance)

This command is used in conjunction with PaintBuffer to increase the speed of display on the screen when the program generates multiple graphical objects.

With UseBuffer, all the graphics are displayed only after the next PaintBuffer command is executed.

UseBuffer only needs to be called once in the program i.e. every use of PaintBuffer does not need a corresponding UseBuffer

**See Also:** PaintBuffer

```
UseBuffer
For n,1,10
x:=randInt(0,300)
y:=randInt(0,200)
radius:=randInt(10,50)
Wait 0.5
DrawCircle x,y,radius
EndFor
PaintBuffer
```

This program will display all the 10 circles at once.

If the “UseBuffer” command is removed, each circle will be displayed as it is drawn.

# TI-Nspire CX II - TI-Basic Programming Features

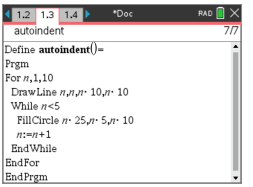
## *Auto-indentation in Programming Editor*

The TI-Nspire™ program editor now auto-indents statements inside a block command.

Block commands are If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry

The editor will automatically prepend spaces to program commands inside a block command. The closing command of the block will be aligned with the opening command.

The example below shows auto-indentation in nested block commands.




Code fragments that are copied and pasted will retain the original indentation.

Opening a program created in an earlier version of the software will retain the original indentation.

---

## *Improved Error Messages for TI-Basic*

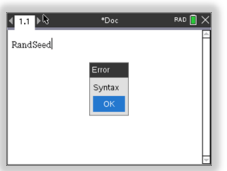
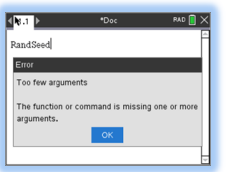
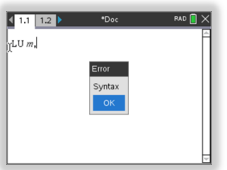
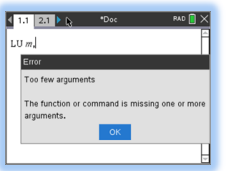
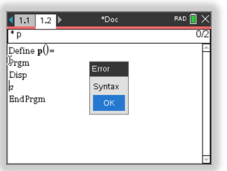
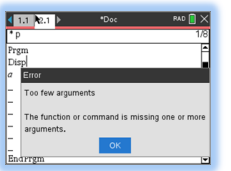
**Errors**

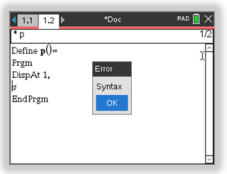
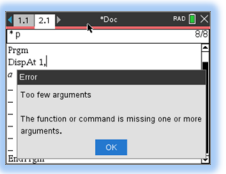
| **Error Condition** | **New message** |
|---|---|
| Error in condition statement (**If/While**) | A conditional statement did not resolve to **TRUE** or **FALSE**<br>**NOTE:** With the change to place the cursor on the line with the error, we no longer need to specify if the error is in an "**If**" statement or a "**While**" statement. |
| Missing **EndIf** | Expected **EndIf** but found a different end statement |
| Missing **EndFor** | Expected **EndFor** but found a different end statement |
| Missing **EndWhile** | Expected **EndWhile** but found a different end statement |
| Missing **EndLoop** | Expected **EndLoop** but found a different end statement |

| Error Condition | New message |
|---|---|
| Missing **EndTry** | Expected **EndTry** but found a different end statement |
| "**Then**" omitted after **If** <condition> | Missing **If..Then** |
| "**Then**" omitted after **ElseIf** <condition> | **Then** missing in block: **ElseIf**. |
| When "**Then**", "**Else**" and "**ElseIf**" were encountered outside of control blocks | **Else** invalid outside of blocks: **If..Then..EndIf** or **Try..EndTry** |
| "**ElseIf**" appears outside of "**If..Then..EndIf**" block | **ElseIf** invalid outside of block: **If..Then..EndIf** |
| "**Then**" appears outside of "**If....EndIf**" block | **Then** invalid outside of block: **If..EndIf** |

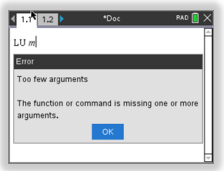
### Syntax Errors

In case commands that expect one or more arguments are called with an incomplete list of arguments, a "**Too few argument error**" will be issued instead of "**syntax**" error

| Current behavior | New CX II behavior |
|---|---|
|  |  |
|  |  |
|  |  |

| Current behavior | New CX II behavior |
| --- | --- |
|  |  |

**Note:** When an incomplete list of arguments is not followed by a comma, the error message is: “too few arguments”. This is the same as previous releases.



# General Information

## *Online Help*

education.ti.com/eguide

Select your country for more product information.

## *Contact TI Support*

education.ti.com/ti-cares

Select your country for technical and other support resources.

## *Service and Warranty Information*

education.ti.com/warranty

Select your country for information about the length and terms of the warranty or about product service.

Limited Warranty. This warranty does not affect your statutory rights.