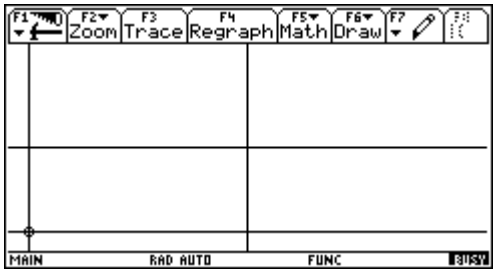
[1.16] Use your TI-89/TI-92 Plus as an on-screen ruler and protractor

The program in this tip, called *ruler()*, lets you measure small objects on the calculator screen. You must be careful not to damage the screen, but with some care it works fairly well. The program also shows some useful programming techniques such as automatically identifying the calculator type and doing some simple graphics. TI Basic is fast enough for this simple interactive program to be usable.

This screen shot shows the measurement screen.



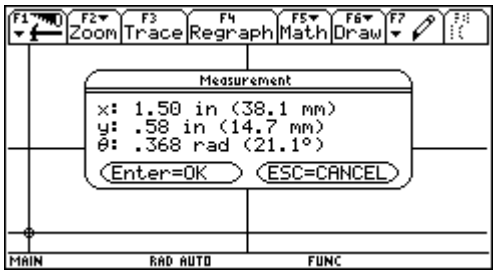
The object is measured between two sets of cross-hair cursors. The *origin* cursor at the lower left screen corner is identified by the small circle around the cross-hair intersection. The *target* cursor is at the center of the screen. Both cursors are moved with the blue cursor keys. When *ruler()* is first started, the cursor keys move the target cursor. The [O] key toggles origin and target movement.

These are the key functions, which can be displayed by pressing [H], for 'help'.

[O]	Switch between target movement and origin movement. When the program is started, the target can be moved. Push [O] to move the origin, then push [O] again to move the target. Repeat as necessary. Push [-] on the TI-89, don't use [alpha].
[UP]	Move cursor up one pixel
[2ND] [UP]	Move cursor up ten pixels
[DOWN]	Move cursor down one pixel
[2ND] [DOWN]	Move cursor down ten pixels
[LEFT]	Move cursor left one pixel
[2ND] [LEFT]	Move cursor left ten pixels
[RIGHT]	Move cursor right one pixel
[2ND] [RIGHT]	Move cursor right ten pixels
[M]	Display the measurement. Push [5] on the TI-89.
[STO]	Copy the measurement to the home screen
[H]	Display the help screen. Push [8] on the TI-89.
[HOME], [ESC], [QUIT]	Exit the program

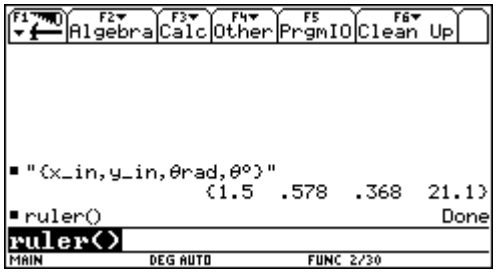
If the cursor moves beyond the edge of the screen, it will appear at the opposite screen edge. This is called wrap-around.

The measurement is shown by pressing [M], as this screen shot illustrates:



The *x* measurement is the horizontal distance between the two cursors. The *y* measurement is the vertical distance between the two cursors. The measurements are negative if the origin is above the target or to the right of it. The angle θ is measured from the origin cross-hair to the target. The angle is undefined (*undef*) if x = 0 and y = 0.

Push [STO] to copy the current measurement results to the home screen. This screen shot shows the result after exiting *ruler()*:



The string in the entry column shows that the list elements are the x-measurement, the y-measurement, and the angle measurement in radians and degrees. You can extract the individual elements with *list[n]*, where *n* is 1 to 4. For example, to extract the y-measurement, use the cursor keys to highlight the list, press [ENTER] to copy it to the entry line, then type [2] [ENTER], and .578 is entered in the history. This feature is accomplished with the *copyto_h()* utility, which must be installed in a folder named *util*\.

One version of *ruler()* runs on both the TI-89 and the TI-92 Plus. This table shows the measurement limits for the two calculators.

	TI-89	**TI-92 Plus**
Maximum x-measurement	2.18 in (55.4 mm)	3.29 in (83.6 mm)
Maximum y-measurement	1.06 in (27 mm)	1.42 in (36 mm)
Approximate x, y resolution	0.0138 in (0.4 mm)	Same

The accuracy is limited by my ability to accurately measure the pixel pitch, as well as screen parallax, LCD manufacturing consistency and the size of the pixel. Total error may be as much as two or three pixels, so the accuracy cannot be better than about 0.05 inches. The angle resolution depends on the distance between the target and the origin. If x = 5 pixels and y = 5 pixels, the resolution is about 5°. The best resolution is about 0.3° for either calculator.

Source code description

The source code for *ruler()* is shown on the following pages. I have added comments to describe the operation, but these comments are not in the program in the *tlcode.zip* file.

ruler() uses local programs and functions so it can be distributed as a single program. This does require using global variables to pass parameters, since the scope of local variables defined in *ruler()* does not extend to the other locally-defined programs. The global variables are stored in the *main*\ folder, since it is guaranteed to exist on every calculator, and they are deleted before exiting *ruler()*.

Since *ruler()* needs to change the Graph and Mode settings, they are saved in the beginning of the program and restored at the end. The Angle mode is set to Radian, so I know that the results of angle calculations are in radians. To change the Graph and Mode settings, I use the numeric string arguments for *setGraph()* and *setMode()*, instead of the text strings. This makes program operation independent of language localization.

I want the graph screen clear except for the cursors, so this code completely clears and displays the graph screen:

```
ClrDraw                     © Clear all plots, functions and drawings
ClrGraph
PlotsOff
FnOff
SetGraph("3","1")           © Turn grid off
SetGraph("4","1")           © Turn axes off
DispG                       © Display graph screen
```

To avoid writing, distributing and maintaining two versions of the program, I detect which calculator model (TI-89 or TI-92 Plus) on which the program is running. The model is determined by using *getConf()* to find the calculator screen width. If the width is 240, then the calculator is a TI-92 Plus, otherwise it is a TI-89. The expression to find the model is

```
if getconfg()[dim(getconfg())-14]=240 then
...
```

The test returns *true* if the calculator is a TI-92 Plus. *getConfg()* returns a list of configuration pairs. The location of the screen width configuration pair depends on whether or not certificates are installed. However, the screen width is 14 elements from the end of the list, regardless of certificate installation, so the expression gets the configuration with respect to the end of the list, that is, 14 elements from the end.

Three specific parameters depend on the calculator model: the screen size, the key codes for some keys, and the pixel pitch. I account for all these parameters by assigning the correct values to variables based on the calculator model, then the program logic uses these variables instead of hard-coded constants.

I use the letter keys [O], [M] and [H] for features in *ruler()*. The same keys are used for both the TI-89 and the TI-92 Plus. However, on the TI-89, I don't make the user enable alpha mode to press the right key. Instead, I test for the unmodified key code for that letter key. For example, the [O] key is the same as the [-] key, so I just test for the [-] key code (45), instead of the [O] key code (79). The TI-89 user presses the [-] key to perform the origin/target toggle function.

The user can press any of three keys to exit the program: [ESC], [QUIT] or [HOME]. This just makes it easier for the user, since she does not need to remember exactly which key stops the program. The cursor movement keys are consistent with the standard operation on the Graph screen. Pressing just

the cursor key moves one pixel, and pressing [2ND] with the cursor key moves the cursor a larger step.

There are a few dedicated function keys, and they are not necessarily obvious, so I include a 'help' screen, which is shown each time the program starts.

ruler() has two shortcomings which could be remedied. First, it is not obvious whether the origin or the target cursor will move when the cursor keys are pressed. However, it is obvious which cursor moves once the key is pressed, and the other cursor can be selected by pressing [O]. Second, the program would be easier to use if the measurement was continuously shown in the status line. This cannot be done with TI Basic, but can be done with a C program. I chose not to do that, in order to keep the program simple.

Source code for ruler()

```
ruler()
Prgm
© On-screen ruler & protractor
© 15julØ2/dburkett@infinet.com
© Calls util\copyto_h()


© Global variables

© main\µox     Origin cursor x-coordinate
© main\µoy     Origin cursor y-coordinate
© main\µtx     Target cursor x-coordinate
© main\µty     Target cursor y-coordinate
© main\µx      Maximum x-coordinate
© main\µy      Maximum y-coordinate
© main\µscl    Scale factor, inches/pixel


© Local variables

local gdb,key,draw_all,mode,help,move_x,move_y,modedb,p,,q01,q02r,ko,km,kh,khm,dist

© gdb          graph database settings
© mode_db      Mode settings
© key          pressed-key code
© ko           [O] key code
© km           [M] key code
© kh           [H] key code
© khm          [HOME] key code
© mode         Cursor mode: 1 == move target cursor, Ø == move origin cursor
© p            Measurement results
© q01          Results label string for [STO] (copy results to home screen)
© q02          Results string for [STO] (copy results to home screen)
© r            Angle measurement string
© dist         Function: calculate measurements
© draw_all     Program: draw both cursors on Graph screen
© help         Program: display 'help' screen
© move_x       Program: move target or origin cursor in x-direction
© move_y       Program: move target or origin cursor in y-direction


©=======================================
© Local program and function definitions
©=======================================

© draw_all()
© Draw both cursors on LCD screen
```

```
Define draw_all()=Prgm
 PxlHorz main\μoy
 PxlVert main\μox
 PxlHorz main\μty
 PxlVert main\μtx
 PxlCrcl main\μoy,main\μox,2
EndPrgm

© help()
© Display help dialog box

Define help()=Prgm
 Dialog
  Title "Ruler Help"
  Text "[O] Set origin"
  Text "[M] Measure"
  Text "[STO▸] Copy measurement to Home"
  Text "[H] Help"
  Text "To quit:"
  Text "[HOME], [ESC] or [QUIT]"
 EndDlog
EndPrgm

© move_x(mode,distance)
© Move cursor (target or origin) in x-direction

Define move_x(m,i)=Prgm
 If m=1 then                          © Move target cursor
  PxlVert main\μtx,Ø                  © ... erase old cursor line
  mod(main\μtx+i,main\μx)→main\μtx     © ... find new coordinate with wrap-around
 Else                                 © Move origin cursor
  PxlVert main\μox,Ø                  © ... erase old cursor line
  PxlCrcl main\μoy,main\μox,2,Ø       © ... erase old origin circle
  mod(main\μox+i,main\μx)→main\μox     © ... find new coordinate with wrap-around
 Endif
EndPrgm

© move_y(mode, distance)
© Move cursor (target or origin) in y-direction

Define move_y(m,i)=Prgm
 If m=1 then                          © Move target cursor
  PxlHorz main\μty,Ø                  © ... erase old cursor line
  mod(main\μty+i,main\μy)→main\μty     © ... find new coordinate with wrap-around
 Else                                 © Move origin cursor
  PxlHorz main\μoy,Ø                  © ... erase old cursor line
  PxlCrcl main\μoy,main\μox,2,Ø       © ... erase old origin circle
  mod(main\μoy+i,main\μy)→main\μoy     © ... find new coordinate with wrap-around
 Endif
EndPrgm


© dist()
© Find distance and angle for current cursor locations

Define dist()=Func
 local dd,dr,xd,yd
 © dd  angle in degrees
 © dr  angle in radians
 © xd  x-axis distance between cursors
 © yd  y-axis distance between cursors

 (main\μtx-main\μox)*μscl→xd          © Find x-axis distance
 (main\μoy-main\μty)*μscl→yd          © Find y-axis distance

 If xd=Ø and yd=Ø Then                © Find angle
  undef→dr:dr→dd                      © ... angle is undef if x=Ø and y=Ø
 else
  R▸Pθ(xd,yd)→dr                      © ... else calculate angle in radians
  dr*18Ø/π→dd                         © ... and degrees
```

```
 EndIf
 return {xd,yd,dr,dd}
EndFunc


©=================
© Begin mainline
©=================

StoGDB gdb                    © Save graph database settings
GetMode("ALL")→modedb         © Save mode settings
setMode("3","1")              © Set Angle mode to radians

ClrDraw                       © Clear all plots, functions and drawings
ClrGraph
PlotsOff
FnOff
SetGraph("3","1")             © Turn grid off
SetGraph("4","1")             © Turn axes off
DispG                         © Display graph screen

© Initialize calculator-specific parameters

if getconfg()[dim(getconfg())-14]=24Ø then   © Determine calculator model by screen width
 239→main\μx                  © For TI-92+:  set maximum x-dimension
 1Ø3→main\μy                  ©              set maximum y-dimension
 111→ko                       ©              set [O] key code
 1Ø9→km                       ©              set [M] key code
 1Ø4→kh                       ©              set [H] key code
 8273→khm                     ©              set [HOME] key code
 .Ø13772→main\μscl            ©              set scale factor in inches/pixel
else
 159→main\μx                  © For TI-89:   set maximum x-dimension
 77→main\μy                   ©              set maximum y-dimension
 45→ko                        ©              set [O] key code
 53→km                        ©              set [M] key code
 56→kh                        ©              set [H] key code
 277→khm                      ©              set [HOME] key code
 .Ø1381→main\μscl             ©              set scale factor in inches/pixel
endif

© Initialize cursor positions

1Ø→main\μox                   © Set origin cursor 'x' 1Ø pixels from screen left
main\μy-1Ø→main\μoy           © Set origin cursor 'y' 1Ø pixels above screen bottom
intdiv(main\μx,2)→main\μtx    © Set target cursor to center of screen
intdiv(main\μy,2)→main\μty

1→mode                        © Set mode to move target cursor

draw_all()                    © Redraw cursors
help()                        © Display 'help'

© Main loop

Loop
 GetKey()→key                              © Get pressed key code
 If key=264 or key=khm or key=436Ø:Exit    © Exit program

© Handle cursor movement keys

 If key=34Ø Then              © [RIGHT]
  move_x(mode,1)              © Move cursor 1 pixel right
  draw_all()                  © Update screen

 ElseIf key=337 Then          © [LEFT]
  move_x(mode,⁻1)             © Move cursor 1 pixel left
  draw_all()                  © Update screen

 ElseIf key=338 Then          © [UP]
```

```
  move_y(mode,⁻1)            © Move cursor 1 pixel up
  draw_all()                 © Update screen

 ElseIf key=344 Then         © [DOWN]
  move_y(mode,1)             © Move cursor 1 pixel down
  draw_all()                 © Update screen

 ElseIf key=4436 Then        © [2ND] [RIGHT]
  move_x(mode,1Ø)            © Move cursor 1Ø pixels right
  draw_all()                 © Update screen

 ElseIf key=4433 Then        © [2ND] [LEFT]
  move_x(mode,⁻1Ø)           © Move cursor 1Ø pixels left
  draw_all()                 © Update screen

 ElseIf key=4434 Then        © [2ND] [UP]
  move_y(mode,⁻1Ø)           © Move cursor 1Ø pixels up
  draw_all()                 © Update screen

 ElseIf key=444Ø Then        © [2ND] [DOWN]
  move_y(mode,1Ø)            © Move cursor 1Ø pixels down
  draw_all()                 © Update screen

© Handle feature keys

 ElseIf key=ko Then          © [O] Toggle origin/target adjustment mode
  when(mode=Ø,1,Ø)→mode      © If mode = Ø, toggle to 1 and vice versa

 ElseIf key=kh Then          © [H] Display 'help' screen
  help()

 ElseIf key=km Then          © [M] Display measurement results
  dist()→p                   © Calculate measurements
  if p[3]=undef then         © Set angle measurement display string
   "undef"→r                 © ... handle 'undef'
  else                       © ... else format radian and degree results
   format(p[3],"F3")&" rad ("&format(p[4],"F1")&"°)"→r
  EndIf

  Dialog                     © Display measurements
   Title "Measurement"
   text "x: "&format(p[1],"F2")&" in ("&format(25.4*p[1],"F1")&" mm)"
   text "y: "&format(p[2],"F2")&" in ("&format(25.4*p[2],"F1")&" mm)"
   text "θ: "&r
  EndDlog

 elseif key=258 then         © [STO] Copy measurements to Home screen
  "{x_in,y_in,θrad,θ°}"→qØ1  © Save label ...
  dist()→qØ2                 © ... and measurements
  util\copyto_h("qØ1","qØ2") © ... then copy them to the Home screen
  draw_all()                 © ... and redraw to clean up after copyto_h()
 EndIf

EndLoop

©================================
© Clean up before exiting program
©================================

                             © Delete global variables
delvar main\μox,main\μoy,main\μtx,main\μty,main\μx,main\μy,main\μscl
rclGDB gdb                   © Restore graph database settings
setMode(modedb)              © Restore mode settings
DispHome                     © Display Home screen

EndPrgm
```