[7.46] 'Custom' command makes menu labels for trapped function keys

This tip shows how to use the *Custom* command to create labels for function keys used in interactive programs. An example might be a program which uses the Graph screen to display information, and the user can press one of several keys to perform program features. For example, suppose you have written a program which manipulates graphic objects on the screen. The objects can be moved and rotated, and there is a help feature. You want to provide quick, obvious access to these features. This program shows the basic idea.

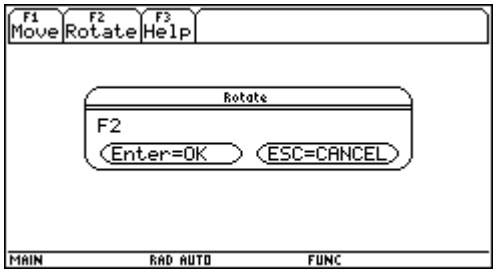
```
tooltab(m)
Prgm
© Custom toolbar with key trap demonstration
© (m) m is program name to restore menu, as a string
© 13janØ2/dburkett@infinet.com

local key             © Pressed key code

dispg                 © Display graph screen

custom                © Display toolbar tabs
 title "Move"
 title "Rotate"
 title "Help"
endcustm
custmon               © Display toolbar

© Loop to wait for key presses
loop
 getkey()→key         © Get pressed key code
 if key=264:exit      © Exit when [ESC] pressed

 if key=268 then      © [F1] key
  dialog
   title "Move"
   text "F1"
  enddlog

 elseif key=269 then  © [F2] key
  dialog
   title "Rotate"
   text "F2"
  enddlog

 elseif key=27Ø then  © [F3] key
  dialog
   title "Help"
   text "F3"
  enddlog

 endif

endloop

if dim(m)≠Ø:expr(m)   © Restore menu

disphome              © Display home screen

EndPrgm
```

The basic principles are:

1. Display the graph screen. You may need to clear the axes and make other settings, depending on your program.

2. Display a custom toolbar with only *Title* entries, no *Item* entries.
3. Execute a loop which includes a *getkey()* call get key press codes, and test the key code for function keys. The loops traps the *function* key presses.

When *tooltab()* is executed and [F2] is pressed, this screen is shown:



The dialog boxes are only examples; your program would instead perform the code needed to perform the desired function.

Another feature of *tooltab()* is restoring a custom menu. *tooltab()* replaces any custom menu with its own, so it would be considerate to restore the user's custom menu. No TI Basic program which changes the screen from Home to some other can restore a custom menu, but we can at least make it convenient to do by pressing [CUSTOM] after the program ends. *tooltab()* accomplishes this by accepting a program name as a string argument *m*, then executing

```
if dim(m)≠0:expr(m)    © Restore menu
```

before exiting. For example, suppose I have a program to set up a custom menu called *main\custom92()*, then I would use this call:

```
tooltab("main\custom92()")
```

After exiting *tooltab()*, I just press [CUSTOM], and my menu is restored. Note that I use the folder specifier *main*\, so *custom92()* is executed regardless of which folder from which *tooltab()* is executed. If you would rather not use this feature, just call *tooltab()* with an empty string:

```
tooltab("")
```