[7.28] Use long strings with *setMode()* in custom menus

The Custom menu structure lets you customize the 89/92+ for more efficient operation. You can replace the function key menu tabs at the top of the screen with more useful operations. However, there is a limit to the length of the strings you can use in the Item argument. This limit prevents the use of the long strings needed for some of the *setMode()* function arguments. The solution to this dilemma is to use the number codes for the *setMode()* arguments, instead of the strings themselves.

Suppose we want to make a custom menu to set the Exact/Approx modes to Auto, Exact or Approximate. You might try this program, but it will not work:

```
custom1()
Prgm

custom
 title "Modes"
 item "setmode(""Exact/Approx"",""Auto"")"
 item "setmode(""Exact/Approx"",""Exact"")"             <- FAILS HERE
 item "setmode(""Exact/Approx"",""Approximate"")"
endcustm

custmon

EndPrgm
```

The string for the second Item line, to set the mode to Exact, causes a "Dimension" error because the string is too long. The solution is to create shorter strings by using the number string codes for *setMode(),* instead of the strings themselves. These codes are listed on page 584 of the online *89/92+ Guidebook*.

Note also the use of the double quote marks around the item strings. These are used to embed double quotes in the Item arguments, which are themselves strings.

This is how the program looks using the codes instead of the strings:

```
custom1()
Prgm

custom
 title "Modes"
 item "setmode(""14"",""1"")"
 item "setmode(""14"",""2"")"
 item "setmode(""14"",""3"")"
endcustm

custmon

EndPrgm
```

"14" is the code for Exact/Approx. "1", "2" and "3" are the codes for Auto, Exact and Approximate, respectively. This program works, but you can't really identify tell from the displayed menu which item is which. A solution to this problem is to use extra Item commands, just as labels. With this solution, the program now looks like

```
custom1()
Prgm

custom
 title "Modes"
 item "Set Auto:"
 item "setmode(""14"",""1"")"
 item "Set Exact:"
 item "setmode(""14"",""2"")"
 item "Set Approx:"
 item "setmode(""14"",""3"")"
endcustm

custmon

EndPrgm
```

This program will create a menu for function key F1 that looks like this:

```
1:Set Auto:
2:setmode("14","1")
3:Set Exact:
4:setmode("14","2")
5:Set Approx:
6:setmode("14","3")
```

There are a few problems with this method. First, it makes the menu twice as long as it really needs to be. Second, it is an awkward user interface, because the user's natural inclination is to press [1] for Auto mode, while it is really key [2] that needs to be pressed.

These objections can be overcome by using both codes and strings in the Item arguments. Use the "14" code to shorten strings, but use the strings themselves for Auto, Exact and Approximate:

```
custom1()
Prgm

custom
 title "Modes"
 item "setmode(""14"",""Auto"")"
 item "setmode(""14"",""Exact"")"
 item "setmode(""14"",""Approximate"")"
endcustm

custmon

EndPrgm
```

This program makes a menu that looks like this:

```
1:setmode("14","Auto")
2:setmode("14","Exact")
3:setmode("14","Approximate")
```

If you switch modes often, this type of menu is faster than using the MODE menu. Using the MODE menu takes at least 7 keystrokes. The Custom menu method requires 3 keystrokes. However, the Custom menu method unnecessarily leaves the commands in the history display. You need to press [ENTER] after selecting the menu item, which is just one more keystroke. Finally, the appearance of the menu items is rather poor in that all the *SetMode()* details are displayed. Still, it is an improvement over using the [MODE] key and cursor control keys to make a simple mode change.

(Credit for use of code strings declined)