[9.14] Use keyboard programs to quickly change mode settings

The mode settings (shown with the [MODE] key) control fundamental aspects of calculator operation. Many of the tips in this list show that the proper mode setting is important for getting the right result. If you use your calculator for a variety problem types you will switch modes quite often, and this is tedious because several steps are needed to change a setting: press the [MODE] key, scroll down to the mode category, open the mode submenu, scroll to the right setting, change it, and press [ENTER] to finally make the change. One solution to this problem is to use the built-in keyboard program feature to change modes quickly with a few keystrokes. This is particularly useful with mode settings shown in the status line: angle units, auto/exact/approx and graph type. Since these are shown in the status line, you can quickly see that the correct mode is set. The keyboard programs are executed with [DIAMOND] [1] for *kbdprgm1()*, [DIAMOND] [2] for *kbdprgm2()*, and so on. You can have up to nine *kbdprgm()* programs.

In general, we will use a [DIAMOND] key combination to toggle each mode setting. For example, if I set up *kbdprgm1()* to cycle through the Auto/Exact/ mode, then repeatedly pressing [DIAMOND] [1] changes the setting through Auto, Exact, Approx, Auto and so on. If I use *kbdprgm2()* to change the angle mode, then pressing [DIAMOND] [2] changes the setting through RAD, DEG, RAD and so on.

We use *getMode()* to find the current mode, then *setMode()* to change the setting to the next option. The forms of *getMode()* and *setMode()* which we need are

```
getMode(modeString)

setMode(modeString,setString)
```

where *modeString* specifies the mode to get or set, and *setString* specifies the new setting. *modeString* and *setString* can be specified in one of two ways: either as text strings in the current language, or as numeric strings. I use numeric strings because this makes the programs independent of the current language, and it is easy to calculate the next mode setting.

We could used repeated *If ... then ... elseif ...* structures to change the mode, but any mode setting change can be done with one line of code:

```
setMode(modeString,string(exact(mod(expr(getMode(modeString)),n)+1)))
```

where *n* is the number of setting options. For example, if we want to change the Auto/Exact setting, then *modeString* = "14" and *n* = 3, since there are three choices. This statement changes the mode like this, working from the inside out:

Get the current mode as a string:
```
getMode(modeString)
```

Convert the mode string to a number:
```
expr(getMode(modeString))
```

Calculate the next mode setting number:
```
mod(expr(getMode(modeString)),n)+1
```

Use *exact()* to remove any decimal points or exponent characters:
```
exact(mod(expr(getMode(modeString)),n)+1)
```

Convert the next mode number to a string:
```
string(exact(mod(expr(getMode(modeString)),n)+1))
```

Change the mode setting:

```
setMode(modeString,string(exact(mod(expr(getMode(modeString)),n)+1)))
```

The next mode setting number is calculated with the *mod()* function which provides the wrap-around we need. For example, if there are three setting options, then we need to convert a 1 to 2, 2 to 3 and 3 to 1. The calculation works like this for n = 3:

mod(1,3)+1 = 2 *(setting 1 changes to setting 2)*
mod(2,3)+1 = 3 *(setting 2 changes to setting 3)*
mod(3,3)+1 = 1 *(setting 3 wraps around to setting 1)*

Applying *exact()* to the mode setting number calculation removes any decimal points or exponent characters, which may be present, depending on the current display format setting. This must be done because *setMode()* does not accept these characters in the *setString* argument. By including *exact()*, the expression will work with any display mode setting and in Approx mode.

This table shows the values of *modeString* and *n* for the modes which can be set with this method.

modeString and number of options for various modes

Mode category	modeString	n
"Graph"	"1"	6
"Display digits"	"2"	26
"Angle"	"3"	2
"Exponential Format"	"4"	3
"Complex Format"	"5"	3
"Vector Format"	"6"	3
"Pretty Print"	"7"	2
"Split screen"	"8"	3
"Number of Graphs"	"11"	2
"Graph 2"	"12"	6
"Split screen ratio"	"13"	2 for TI-89, 3 for TI-92+
"Exact/Approx"	"14"	3
"Base"	"15"	3

Refer to the *User's Guide* entry for the *setMode()* function for descriptions of the mode setting options.

Note that modes "Split 1 App" and "Split 2 App" do not use number codes for the settings, and the "Language" mode does not use a number code for the mode itself. These modes cannot be set with this method.

If you use several keyboard programs to change modes, you can save a little memory by using a program to change the mode settings:

```
modecycl(mode,n)
Prgm
©("mode",n) Toggle "mode" with n options
©3janØ2/dburkett@infine.com
setmode(mode,string(exact(mod(expr(getmode(mode)),n)+1)))
EndPrgm
```

then you could use these keyboard programs with [DIAMOND] [1], [DIAMOND] [2] and [DIAMOND] [3]:

```
kbdprgm1()
Prgm
©Change Angle setting
modecycl("3",2)
EndPrgm


kbdprgm2()
Prgm
©Change Exact/Auto/Approx setting
modecycl("14",3)
EndPrgm

kbdprgm3()
Prgm
©Change Graph setting
modecycl("1",6)
EndPrgm
```

Note that the [DIAMOND] keys are defined in the same order that the Mode indicators appear in the display status line. This makes it easier to remember the key which changes a particular mode.

For the Angle units, Auto/Exact and the Graph modes, these programs work well because the setting is shown in the status line. However, the other mode settings are not shown, so we don't know what the current setting is, nor what we have changed it to. In addition, some mode settings have so many options that it is not as helpful to cycle through them: it takes just as much time to do that as it does to use the built-in Mode screen.

With nine keyboard programs, it can be difficult to remember which programs are assigned to which keys. It helps to assign the keyboard programs in groups, as we did here, with some logical ordering to the key assignment.