October, 2000

Advisor Answers

Function keys and KeyPress Event

Visual FoxPro 6.0, 5.0, 3.0

Q: I'm trying to use F10 to do a method within my form. I don't want to use ON KEY LABEL to set my function keys for design reasons. I'm using the form's KeyPress event to capture which function key is pressed. I had to SET HELP OFF so that I could use F1. When F10 is pressed, the KeyPress Event doesn't fire and the focus goes to the first option on the menu. Is there something I can do to set F10 off? Thanks in advance for your help.

–Daniel Reber (via www.Advisor.com)

A: Before I offer a solution, I have to question the wisdom of what you're doing. F10 is universally defined in Windows as the menu activation key. That's why focus goes to the first option on the menu.

Try this in any application that conforms to the Windows Interface Guidelines and you'll find that pressing F10 highlights the first menu item. It's a way that a user without a mouse can get to the menu. While you might ask why someone would use Windows without a mouse, keep in mind first that there are disabled users who cannot use a mouse and second, that at one time or another, we've all run into situations where a mouse driver or the mouse itself fails and we're stuck without a mouse temporarily. In addition, some users simply prefer to work from the keyboard. That's why the Windows Interface Guidelines require that every mouse action have a keyboard equivalent – in fact, it's referred to as a "fundamental principle." (You can find the latest version of the Guidelines online beginning at http://msdn.microsoft.com/library/books/winguide/welcome.htm.)

For similar reasons, I'm concerned about your decision to override F1 as the Help key. F1 was established as the universal key for Help even before Windows became ubiquitous. To use it for any other action seems like a poor design choice.

If your application is aimed primarily at data-entry operators, you also may want to rethink the whole question of using the function keys at all. In order to hit the function keys, the user must remove her hands from the home row of the keyboard. In an application where the user

spends most of her time entering data, it's best if the hands never leave the keyboard (yet another reason to offer keyboard equivalents for all operations). This kind of application similarly shouldn't include complex controls – entry should use text boxes and edit boxes with nothing that requires the user to take her eyes off the source document that contains the input data.

By now, you're probably asking what I recommend rather than the function keys? The answer is menu shortcuts. Put the operations on a menu popup and give each a shortcut using the Options dialog. Ctrl+key combinations are traditional for menu items; Alt+key combinations are generally reserved for menu pads on the main menu bar. Regular users will quickly learn the shortcuts for the items they use all the time. You can also provide a hot key (that is, an underlined character) for each item to speed keyboard use for those who prefer to drop the menu popup open first. If you want to offer mouse users a shortcut as well, create a custom toolbar with a button for each frequently used item. The SYS(1500) function lets you tie buttons right into menu items. (See last month's ADVISOR Answers for more on that function.)

In case I still haven't convinced you not to use function keys, there is a solution, but it's not particularly attractive. You do have to use ON KEY LABEL (or the even older SET FUNCTION). In order to keep all your function key code together, though, my recommendation is to have the ON KEY LABEL change the output from F10 rather than calling the method directly. You can do that with code like:

```
ON KEY LABEL F10 KEYBOARD "{Shift+F11}" PLAIN
```

I chose Shift+F11 because it doesn't have any predefined meaning, but you can choose any key you want, with a few exceptions, discussed below. Be sure to add the PLAIN keyword in case the key combination has already been assigned a macro somewhere.

Once you redefine F10, you can then process it in the KeyPress method along with the other function keys. Just be sure to check for the keycode of the replacement keystroke, not the -9 that F10 generates. When using Shift+F11 you have to check the nShiftAltCtrl parameter, as well, because this particular combination has the ANSI code 135 that corresponds to the lowercase "ç", a common French character. Also, remember to set the form's KeyPreview property to .T. so that a keystroke on any control triggers the form's KeyPress event before the control's KeyPress event.

F1 and F10 aren't the only function keys that have meanings in Windows. Shift+F10 opens the context (right-click) menu. Ctrl+F10 maximizes the current window. Alt+F4 closes the current application. There are a number of others, as well. When you redefine F10, you'd be well-advised to stay away from those.

Since Visual FoxPro 5.0, KeyPress code doesn't fire for any key combination that includes ALT. While the KeyPress event itself fires in the event log, the code within is not executed. That's probably because ALT key combinations trigger a different Windows message; WM_SYSKEYUP/DOWN instead of WM_KEYUP/DOWN.

Overall, I'm less happy with this solution than using a menu shortcut because ON KEY LABELs are interrupts that can be executed at any time. If you feel you must go this route, set the OKL up as late as possible and turn it off as soon as you can.

—Tamar