January, 2002

Advisor Answers

Keyboard Access to Context Menus

VFP 7.0/6.0/5.0

Q: I've added a right-click menu to most of my forms, but I noticed that it doesn't work when the user hits the right-click menu button on the keyboard. How can I make that work, too?

–Advisor DevCon attendee

A: Most current applications use context menus (also known as shortcut or right-click menus) to provide users with a quick list of available options. No doubt you've implemented yours by adding code to the RightClick method of your forms and controls.

As you've discovered, however, pressing the context menu key on a Windows keyboard doesn't fire the RightClick method. It makes sense when you think about it, of course, but there are a few places where keyboard and mouse actions are mixed in VFP. (For example, clicking a button with the Cancel attribute puts CHR(27) into the keyboard buffer.) The key to solving this problem turns out to be the key combination that mimics that key. Pressing Shift-F10 has the same effect as pressing the context menu key. In most applications, that's opening a context menu.

How does this help? Once you know what keystrokes are involved, you can add code to the KeyPress method of the form or control to respond to it. KeyPress receives two parameters: a key code, and a value that indicates which, if any, of Ctrl, Shift and Alt are pressed. The parameter statement for the method is:

```
LPARAMETERS nKeyCode, nShiftAltCtrl
```

The second parameter is additive, since more than one of those keys may be pressed. A value of 1 indicates that the Shift key was pressed, 2 indicates that the Ctrl key was pressed, and 4 indicates that the Alt key was pressed. Any other value indicates that multiple special keys were pressed – for example, 3 means that both Shift and Ctrl were pressed.

The first parameter is interesting, because it's not as simple as the ASCII value for the specified key. The parameter passed varies with

the special keys. For example, KeyPress receives 97 for "a" and 65 for "A", as you'd expect, but it also receives 1 for Ctrl-a and 30 for Alt-a. You'll find the complete list of key codes in the VFP Help entry for INKEY().

A quick look at the chart there indicates that the key code for Shift-F10 is 93 and, of course, the nShiftAltCtrl value for that key is 1, since only Shift should be pressed. We can use this information in the KeyPress method to take action when Shift-F10 is pressed. More importantly, Windows converts the context menu key to Shift-F10 before passing it on to applications. So, code like the following in KeyPress gives you what you need:

```
LPARAMETERS nKeyCode, nShiftAltCtrl

IF nKeyCode = 93 and nShiftAltCtrl = 1
  DO MyContextMenu.MPR
ENDIF
```

Which KeyPress method should this code go in? That depends on the situation. If every control has a unique context menu or the controls have a different context menu than the form itself, you need to put code like this in the controls' KeyPress methods.

However, if the same context menu is called for all the controls and for the form itself, you can save a lot of work by putting the code in the form's KeyPress method and setting the form's KeyPreview property to .T. That setting indicates that any keystroke on the form should be sent first to the form-level KeyPress method. After that method is finished, the individual control's KeyPress method is called.

There's also a design issue here. Presumably, whatever you do when Shift-F10 or the context menu button is pressed is the same as when the user right-clicks. While you can put the same code (DO MyContextMenu.MPR, in the example above) in both RightClick and KeyPress, it's a better idea to have both of them call a method that handles right-clicks. That way, if the desired behavior changes, you only have to modify code in one place. Add a method to the form or control with a name like HandleRightClick and call it from both RightClick and KeyPress.

The desirability of this approach becomes more obvious when every control on the form and the form itself have the same context menu. Clearly, you don't want to DO SomeMenu.MPR in every RightClick method; maintenance of such a form is a nightmare. Your first instinct might be to have the RightClick methods of the controls call

ThisForm.RightClick and let the form-level method call the menu. But when you add the keyboard approach into the mix, it becomes clear that we're not just talking about RightClick anymore. In this case, having every RightClick method and the form-level KeyPress method call ThisForm.HandleRightClick results in much easier maintenance.

This month's Professional Resource CD contains an example form, Shortcut.SCX, that demonstrates the use of a single context menu for the form and its controls. It uses the menu EdtShort.MNX that's included with the Solutions Samples – you must have the samples installed for the example to work. (I used that menu just because it was available-you probably wouldn't make that set of options available at the form level in a real application.)

–Tamar