January, 1999

Advisor Answers

Changing Text Case

Visual FoxPro 6.0/5.0

Q: In Microsoft Word, if you highlight a block of text and press Shift-F3, it changes the case of the text. It's an easy way to fix text that was typed in with the wrong case. Is there any way to make VFP do the same thing?

–Name withheld by request (Elkins Park, PA)

A: I use Shift-F3 all the time in Word, both in my own writing and while editing other people's work. It uses a fairly complex algorithm that cycles between upper, lower and "proper" (initial caps) cases. It knows how to handle text that's mixed to begin with and even behaves differently depending on the first character of the mixed text.

At first glance, getting VFP to do the same things seems simple. After all, we have LOWER(), UPPER() and PROPER() functions. In fact, that part of the problem is simple. The hard part is getting your hands on the highlighted text and putting it back. Hardest of all is leaving it highlighted when you're done.

Actually, there are two different cases, and one of those is also very easy. To do this for an object that has SelStart, SelLength and SelText properties, the only twist is figuring out the current condition (all upper, all lower, mixed, etc.) of the text, as long as we make the object available to the routine.

However, getting the same behavior in a Browse or in a code editing window or another place where we're not dealing with an object is trickier. The secret is to use the _CLIPTEXT variable that contains VFP's clipboard text, along with SYS(1500) that lets you call on VFP's menu items. To re-highlight the text afterward, the only solution seems to be to KEYBOARD the necessary keystrokes.

Here's a program called ChgCase.PRG that handles all the variations. It's available on this month's Professional Resource CD:

```
* ChgCase.Prg
* Change the case of the highlighted text. As in Word,
* cycle among UPPER, lower and Proper.
* Leave the changed text highlighted

LPARAMETERS oSource
   * oSource = Container for this text. If passed,
   *           can manipulate highlight using
   *           SelStart and SelLength

LOCAL cInText, nCurrentState, nSelStart, nSelLength, ;
      lFromObject, cClipBoard
   *
   * cInText = highlighted text
   *
   * nCurrentState = current capitalization condition
```

```
   *   1 = all upper -> change to all lower
   *   2 = all lower -> change to proper
   *   3 = proper -> change to all upper
   *   4 = mixed - upper case first ->
   *               change to all upper
   *   5 = mixed - lower case first ->
   *               change to all lower
   *   6 = text begins with a
   *       non-alphabetic character ->
   *       change to all lower
   *
   * nSelStart, nSelLength - hold
   *    SelStart and SelLength values from
   *    calling object
   *
   * lFromObject - Indicates whether this
   *    routine was called from an object
   *    that has SelStart and SelLength properties.
   *
   * cClipBoard - the text on the clipboard before this

* Figure out whether we have an object and SelStart/SelLength
* properties to work with. Get the text.

IF     VarType(oSource) = "O" ;
   AND VarType(oSource.SelStart) = "N"
   * Safe to assume if one is there, so's the other
   lFromObject = .T.
ELSE
   lFromObject = .F.
ENDIF

* Preserve the original contents of the clipboard
cClipboard = _CLIPTEXT

IF lFromObject
   nSelStart = oSource.SelStart
   nSelLength = oSource.SelLength
   cInText = oSource.SelText
ELSE
   * Grab the highlighted text
   SYS(1500,"_med_copy","_medit")
   cInText = _CLIPTEXT
   nSelLength = LEN(_CLIPTEXT)
ENDIF

* Now figure out the current configuration
DO CASE
CASE UPPER(cInText) == cInText
   nState = 1
CASE LOWER(cInText) == cInText
   nState = 2
CASE PROPER(cInText) == cInText
   nState = 3
CASE ISUPPER(cInText)
   * ISUPPER() tests only first character
   nState = 4
CASE ISLOWER(cInText)
   * ISLOWER() tests only first character
   nState = 5
OTHERWISE
```

```
   * First character is not alpha.
   nState = 6
ENDCASE

* Change text based on current state
* This is done separately from the case above to
* make it easier to change the cycle order, if desired.
DO CASE
CASE INLIST(nState, 1, 5, 6)
   cInText = LOWER(cInText)
CASE nState = 2
   cInText = PROPER(cInText)
CASE INLIST(nState, 3, 4)
   cInText = UPPER(cInText)
OTHERWISE
   ASSERT .F. ;
      MESSAGE "Can't figure out original capitalization"
   * do nothing - we're confused
ENDCASE

* Now replace with changed text
* and re-highlight. If possible, use
* properties because keyboarding is risky.

IF lFromObject
   oSource.SelText = cInText
   oSource.SelStart = nSelStart
   oSource.SelLength = nSelLength
ELSE
   * Replace highlighted text
   _CLIPTEXT = cInText
   SYS(1500,"_med_paste","_medit")
   _CLIPTEXT = lcClipboard
   * Have to use brute force for highlighting
   CLEAR TYPEAHEAD
   LOCAL nCount
   FOR nCount = 1 TO nSelLength
      KEYBOARD "{Shift-LeftArrow}" PLAIN
   ENDFOR
ENDIF

RETURN
```

The code can be broken into several steps. First, we determine whether we're dealing with an object that has the properties for selected text. If so, we save SelText as the text to manipulate. If not, we call **Edit**>**Copy** from the menu to get the highlighted text.

Next, we determine the current condition of the selected text. A number of built-in VFP functions make the task easier. There are six states it could be in, as listed in the comments. Once we know its current state, we can change the text appropriately.

Next, we replace the original text with the changed text. If we're dealing with an object, we use SelText. If not, we call **Edit**>**Paste** from the menu.

The final step is to highlight the replaced text so we can call the function again to cycle through other options. For an object, it's easy. Just reset SelStart and SelLength. In other places, the only solution is to KEYBOARD the appropriate number of **Shift-LeftArrow** characters.

Using this code depends on the situation. To make it available in your development environment, you can define an ON KEY LABEL. (Shift-F3 seems appropriate.)

However, a better choice is to create a popup with the shortcut definition. Even if the popup is not attached to a menu and is never seen, the shortcut key will catch the keystroke. In addition, the popup is only active during wait states, unlike an ON KEY LABEL, so it is less likely to interfere with your existing applications.

Be aware that, for some reason, the Menu Designer doesn't let you assign Shift-F3 as a shortcut. However, you can do it in hand-written menu code.

Clearly, the best place for this item is on the Edit menu. Unfortunately, various actions in VFP (like opening a method window) seem to disable the item and leave it disabled when it's on the Edit menu, even though no Skip For condition is specified. Putting it under a custom menu pad solved the problem. Here's the code to create a Text menu pad and put a Change Case item on that pad. (This month's PRD contains this program. In your applications, you'll probably want to use the BEFORE or AFTER clause of DEFINE PAD to position the menu pad where you want it.)

```
DEFINE PAD Text OF _MSYSMENU PROMPT "Te\<xt" COLOR SCHEME 3 ;
       KEY ALT+X, ""
ON PAD Text OF _MSYSMENU ACTIVATE POPUP _mpopText

DEFINE POPUP _mpopText MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE BAR 1 OF _mpoptext PROMPT "Change Case" ;
       KEY Shift+F3, "Shift+F3"
ON SELECTION BAR 1 OF _mpoptext do ChgCase with ;
   IIF(TYPE("_Screen.ActiveForm.ActiveControl") = "O", ;
              _Screen.ActiveForm.ActiveControl, .F.)
```

The popup code checks for an active form and control and passes an appropriate reference if it exists. If not, the routine is called with a dummy .F. parameter.

If you add this menu item to your applications, users familiar with Word will probably kiss you.

–Tamar