By Barry Seymour

Staying on Top

This is NOT Dr. Ruth's column; however I am about to give you some illicit advice this week. How To Stay On Top in Windows.

You've all used Windows Write at one time or another. Go on, admit it, you *have*. You've even gone so far as to use the search and/or replace feature. If not in Write, you've certainly done it in Word for Windows. I'm sure you've noticed a curious thing; the spell checker dialog box stays on top, even when it's not active!

How does it *do* that? This week I'll tell you.

That floating window is what's called an 'owned' window. The owner form which owns it can't block it! The owner can still receive keystrokes and mouse commands, but that owned window stays on top! The Visual Basic development environment is thick with the things. Run another app on top of VB, then activate any part of VB -- if you have a code window open it will obediently pop to the top as well. It's an owned window.

The reason I call this illicit is because I never say it documented anywhere until a friend mentioned the technique to me. After some research, I found it buried in the Window's 3.0 Programmer's reference. (OK, it was on page 1-22 -- not exactly buried, I admit.) Still the term 'owned window' was one I'd never heard before. Here's how the Programmer's Reference describes it...

"An owned window is a special type of overlapped window. Every owned window has an owner. This owner must also be an overlapped window. Being owned forces several constraints upon a window:

> An owned window will always be "above" it's owner when the windows are ordered. Attempting to move the owner above the owned window will cause the owned window to also change its position to ensure that it will always be above it's owner.
>
> Windows automatically destroys an owned window when it destroys the window's owner.
>
> An owned window is hidden when it's owner is minimized."

Okay, so then I had to look up 'overlapped window.'

"An overlapped window is always a top-level window. In other words, an overlapped

window never has a parent window."

Ah-hah.

This seemed like a neat trick, almost like 'taking over' a window and forcing it to do your bidding, keeping your form displayed while it still performed normally. I wondered how many of these characteristics I could expect out of a window a form of mine had owned.

My suspicion is that Microsoft never intended Visual Basic programmers to muck around in this kind of stuff. ("Hey Larry, pinch that fuel line between your fingers -- maybe she'll rev faster. Watch out for the fan blades!") The reason I say that, and the reason I refer to this as an 'illicit' tactic in VB is that you can crash Windows quite convincingly if you don't clean things up before you shut down your application. Bear in mind that whenever you start mucking around with API calls and the like you're skating on thin ice. With this technique, one misstep and BANG! UAE, GPF, FUBAR, what ever you want to call it, you're that. I've crashed Visual Basic (no surprise) on one machine; on another I got kicked out of Windows 3.1 so resoundingly I the SuperVGA monitor I was using pinged in agony. So BE CAREFUL. Save your work every time. Kids, don't try this at home without a spotter.

First (as always), the Function Declaration at the heart of the matter...

```
Declare Function SetWindowWord Lib "User" (ByVal hWnd As Integer, <+>
   ByVal nIndex As Integer, ByVal wNewWord As Integer) As Integer
```

The SetWindowWord function is the heart of the matter. When you make this call, set the hWnd value to the 'owned' window, set the nIndex value to -8 and set the wNewWord value to the handle of the 'owner' window!

In this week's example, VBEX016.MAK (note the new numbering sequence) Form1 'owns' Form2, which immediately starts acting like a pesky little brother you can't get rid of, no matter how hard you try. The code sample looks like this...

```
    GWW_HWNDPARENT% = (-8)
    X = SetWindowWord(Form2.hWnd, GWW_HWNDPARENT%, Form1.hWnd)
```

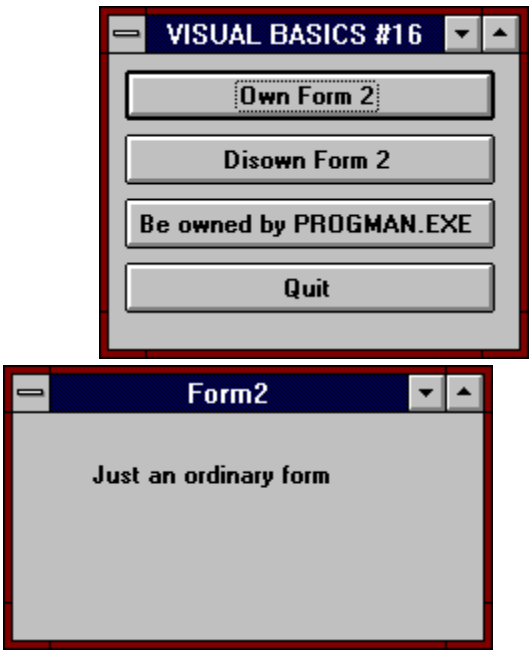
Simple, no?

Make sure you UNDO this before unloading these forms. You set things back to normal by making the window owned by your form and make it owned by ZERO. That looks like this...

```
GWW_HWNDPARENT% = (-8)
X = SetWindowWord(Form2.hWnd, GWW_HWNDPARENT%, 0)
```

Now Form2 is owned by window ZERO, which I assume is the 'mother of all windows' on your screen. Once this is done you can shut down your app gracefully.

This week's example plays with forms taking over each other. VBEX016.MAK has two forms, 1 and 2, with some command buttons. When running, they look like this...



When you click on Form1's 'Own Form 2' button, Form1 'owns' Form2. Set the focus to Form2 and move it partly over Form1. Then click on Form1. It activates, but Form2 still overlaps it!

When you click on 'Be owned by PROGMAN.EXE' you're in for some real fun. The app restores the Program Manager and gets PROGMAN's handle using the GetActiveWindow API call. It then uses that information to make Program Manager take possession of Form1. At this point Form1 will *not* go away, even though

ProgMan continues to function normally. Click on 'Be FREE' to undo this.

This is way cool.

I've put code in the Form_Unload statement of this week's example which makes all affected forms owned by zero. This avoids a crash upon shutdown. If you're in the mood for some fireworks, go ahead and comment it out, then run your app and shut it down. Please e-mail me your results on Windows Online or CompuServe -- they should make for some very entertaining stories. I am not responsible for lost code or broken glass.

To create this example, create a project with two forms that have the following controls and properties...

Formname *FORM1*
BUTTONS...

CtlName	**Caption**
Command1	Loaded in the code
Command2	Loaded in the code
Command3	Loaded in the code
Command4	Loaded in the code

Formname *FORM2*
LABELS..

CtlName	**Caption**
Label1	Loaded in the code

Copy and paste the code from the example into Form1. There is no code in Form2, and no global module. Save and run the program.

As always, this column plus sample code is available on the Windows Online BBS in Danville, California, phone 1 510 736-8343. This column in Windows HELP format (HLP) plus the Visual Basic source code is in VB016EX.ZIP, and may be distributed as freeware.

Barry Seymour
Marquette Computer Consultants
San Rafael, CA 415/459-0835
for Windows Online "the Weekly"

'SAMPLE CODE

'NOTE TO PROGRAMMERS: Watch for the LINE JOIN character, <+>. Remove all these
'symbols and join the lines to make this code work.

```
Declare Function SetWindowWord Lib "User" (ByVal hWnd As Integer, <+>
   ByVal nIndex As Integer, ByVal wNewWord As Integer) As Integer
Declare Function SetFocusAPI Lib "User" Alias "SetFocus" (ByVal hWnd As
Integer) As Integer
Declare Function GetFocus Lib "User" () As Integer
Declare Function GetActiveWindow Lib "User" () As Integer

Sub makeOwnedWindow (ownedHwnd%, ownerHwnd%)
    '  offset into window structure's parent handle element
    GWW_HWNDPARENT% = (-8)
    X = SetWindowWord(ownedHwnd%, GWW_HWNDPARENT%, ownerHwnd%)
End Sub

Sub Command1_Click ()
    GWW_HWNDPARENT% = (-8)
    X = SetWindowWord(Form2.hWnd, GWW_HWNDPARENT%, Form1.hWnd)
    Form2.Label1.Caption = "Now I'm owned by Form1!  He can't block me!"
End Sub

Sub Command2_Click ()
    GWW_HWNDPARENT% = (-8)
    X = SetWindowWord(Form2.hWnd, GWW_HWNDPARENT%, 0)
    Form2.Label1.Caption = "Just an ordinary form again."
End Sub

Sub Form_Load ()
    Form2.Show
    Command1.Caption = "Own Form 2"
    Command2.Caption = "Disown Form 2"
    Command3.Caption = "Be owned by PROGMAN.EXE"
    Command4.Caption = "Quit"
    Form2.Label1.Caption = "Just an ordinary form"
End Sub

Sub Command3_Click ()
    GWW_HWNDPARENT% = (-8)
    If Command3.Caption = "Be owned by PROGMAN.EXE" Then
        AppActivate "Program Manager"
        SendKeys "%{ }R", -1
        PMhWnd% = GetFocus()
        X% = SetWindowWord(Form1.hWnd, GWW_HWNDPARENT%, PMhWnd%)
        Form1.Caption = "PROGMAN OWNED"
        Command3.Caption = "Be FREE"
        X% = SetFocusAPI(Form1.hWnd)
    Else
```

```
        X% = SetWindowWord(Form1.hWnd, GWW_HWNDPARENT%, 0)
        Form1.Caption = "VISUAL BASICS #16"
        Command3.Caption = "Be owned by PROGMAN.EXE"
    End If
End Sub

Sub Command4_Click ()
    Unload Form2
    Unload Form1 ' see FORM1.FORM_UNLOAD
    End
End Sub

Sub Form_Unload (Cancel As Integer)
    'WHEN UNLOADING, SET OWNERS OF ALL WINDOWS TO 0
    'THIS AVOIDS A CRASH!
    GWW_HWNDPARENT% = (-8)
    X% = SetWindowWord(Form1.hWnd, GWW_HWNDPARENT%, 0)
    X% = SetWindowWord(Form2.hWnd, GWW_HWNDPARENT%, 0)
End Sub
```