Are You Serial?!

Click & Retrieve Source CODE!

by Carl Franklin

Q **GET SERIAL!**
When attempting to read the volume serial number of a disk with Visual Basic, I can only get the volume label with the Dir$ function. Do you know how I can get the serial number?
—Gene Soohoo, Sacramento, California

A Visual Basic does not provide this ability. However, there is a shareware VBX that does low-level file functions, including reading and writing the serial number of a disk. The tool is VBIO.VBX, which was developed by SheAr software in The Netherlands. For more information, you can e-mail them at vbx_dev@shear.iaf.nl. The URL to download the file is: http://www.apexsc.com/vb/ftp/misc/iovb.zip.

VBIO.VBX enables you to determine and set disk serial number (number and string), disk volume label, file attributes (even hiding or un-hiding subdirectories), file date, and file time. You can also find out drive type (floppy, hard disk, CD-ROM, network string), drive file allocation table type (file system type), free and total disk space, and a file's size. VBIO.VBX enables you to determine if a drive or file exists, and you can find sectors per cluster, bytes per sector, free clusters, and total clusters for a drive. Errors are reported by the Error and ErrorMessage properties and/or as a trappable runtime error (you decide).

Q **OF MOVING AND MINIMIZING**
Is there a way to keep the user from being able to move a form around on the desktop? Secondly, how can I minimize the Program Manager when my program runs?
—David Baker, San Marcos, Texas

A You can prevent the user from moving the form with a subclassing control such as the Message Blaster. I've talked about MSGBLAST.VBX before, but there is a MSGBLAST.OCX as well.

The original Message Blaster was written at Microsoft and distributed as freeware. The only OCX version of such a control (that I've seen anyway) is a try-before-you-buy OCX from WareWithAll called MSGBLAST.OCX. You can download a trial version of this OCX from my Web site at http://www.apexsc.com/vb/ftp/misc/mblast32.zip.

It works exactly like the VBX. You give it an hWnd for the window (form) you want to trap messages from, and then you give it the message numbers you wish to trap. It fires an event upon the receipt of those messages.

A short example of MSGBLAST.OCX traps some of the WM_SYSCOMMAND Messages, including SC_MOVE, which is received when the user moves the window (see Listing 1). You can set wParam to zero to block the SC_MOVE message, which effectively prevents the user from moving the window. The program consists of a single form with a Message Blaster control on it. As you attempt to move, size, minimize, maximize, restore, or close the form, the caption will change to display a descriptive message of your actions.

Addressing your second question, you can minimize the Program Manager (or any window) by using the ShowWindow API call. ShowWindow takes an hWnd and a State parameter. The State parameter can be any one of these values:

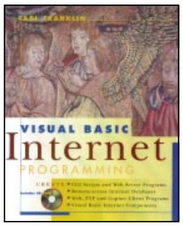
```
Const SW_HIDE = 0
Const SW_MINIMIZE = 6
Const SW_RESTORE = 9
Const SW_SHOW = 5
Const SW_SHOWDEFAULT = 10
Const SW_SHOWMAXIMIZED = 3
Const SW_SHOWMINIMIZED = 2
Const SW_SHOWMINNOACTIVE = 7
Const SW_SHOWNA = 8
Const SW_SHOWNOACTIVATE = 4
Const SW_SHOWNORMAL = 1
```

You can get the hWnd of the Program Manager window with the FindWindow function. Given its class name, window name, or both names, FindWindow returns the handle to any window. In the case of the Program Manager, the class name is PROGMAN. You can minimize the Program Manager using these two API calls (see Listing 2).

In Windows 3.x you can force the Program Manager to minimize whenever you launch *any* program just by selecting the "Minimize on Use" menu option under PM's Options menu. Just because this option exists, it does not mean the user has selected it.

Q **QUOTE, APOSTROPHE, END QUOTE**
I set up a list box to load records from the Namefield (text) in my database. When the user selects one of the names, I use the FindFirst method to

Carl Franklin is a software developer, and the co-owner of Carl & Gary's Visual Basic Home Page (http://www.apexsc.com/vb). Carl is also the author of Visual Basic Internet Programming *(John Wiley & Sons), available April 1996. In his spare time, Carl is researching a cure for feline pattern baldness. Reach Carl by e-mail at carlf@apexsc.com or through* Visual Basic Programmer's Journal. *CompuServe users, address your mail to internet:carlf@apexsc.com.*



This is your forum for addressing the intricacies of the Visual Basic language. Send your questions, clever tips, and techniques. Visual Basic Programmer's Journal *will pay $25 for any submission, tip, or question we print. If your submission includes code, please send a disk along with your hard copy. Send submissions along with your mailing address to Q&A Columnists, c/o Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA, USA, 94301-2500. CompuServe: 74774,305.*

update all my other bound controls. The problem is that some of the records in the Name field contain apostrophes that conflict with the syntax of the FindFirst method. For example, say a Name field equals "Charlie's Steakhouse" and my program tries to use the FindFirst method like this:

```
Data1.FindFirst "Name = 'Charlie's Steakhouse'"
```

VB interprets the apostrophe between the "e" and "s" as the end of the string. All fields without apostrophes, on the other hand, work fine. How can I get my apostrophe records to work without removing the apostrophes?
—Ken Kenny, St. Louis, Missouri

A Syntax sometimes interferes with data. On my last project, I ran up against this problem on a huge scale, and my team came up with a good solution.

First, know your tools. You can use either a quote or an apostrophe in most implementations of SQL, and Jet is no different. Because you can't *type* a quote character as data in a string (because VB uses the quote to encapsulate a string) you must use Chr$(34).

This is how you are currently using FindFirst:

```
Data1.FindFirst "Name = '" & txtName & "'"
```

Here is a better way:

```
Data1.FindFirst "Name = " & Chr$(34) & _
```

VB4

```
Option Explicit
Const WM_SYSCOMMAND = &H112&
Const SC_CLOSE = &HF060&
Const SC_MAXIMIZE = &HF030&
Const SC_MINIMIZE = &HF020&
Const SC_MOVE = &HF012&
Const SC_RESTORE = &HF120&
Const SC_SIZE = &HF008&
Private Sub Form_Load()
  '-- Trap messages for this form
  MsgBlaster1.hWndTarget = Me.hWnd

  '-- Trap the system command message
  MsgBlaster1.AddMessage WM_SYSCOMMAND, 1
End Sub

Private Sub MsgBlaster1_Message(ByVal hWnd As Long, _
  ByVal Msg As Long, wParam As Long, _
  lParam As Long, nPassage As Integer, _
  lReturnValue As Long)
  Dim szMsg      As String
  Select Case Msg
     Case WM_SYSCOMMAND
        Select Case wParam
           '-- Create a descriptive message based
           '   on the system command message
           Case SC_CLOSE
              szMsg = "Closing Window"
              MsgBox szMsg
              Exit Sub
           Case SC_MAXIMIZE
              szMsg = "Maximizing Window"
           Case SC_MINIMIZE
              szMsg = "Minimizing Window"
           Case SC_MOVE
              szMsg = "Attempted Move"
              '-- Disable the moving
              wParam = 0
           Case SC_RESTORE
              szMsg = "Restoring Window"
           Case SC_SIZE
              szMsg = "Sizing Window"
        End Select
        '-- Display the descriptive message
        '   in the form's caption
        Caption = szMsg
  End Select
End Sub
```

LISTING 1 ***I'm Not Moving!*** *Using WareWithAll's 32-bit Message Blaster OLE control, you can easily subclass a Visual Basic form to prevent the user from moving the form.*

VB4

```
Option Explicit

Private Declare Function FindWindow _
  Lib "user32" Alias _
  "FindWindowA" ByVal lpClassName As String, ByVal _
  lpWindowName As Any) As Long

Private Declare Function ShowWindow Lib "user32" _
  (ByVal hwnd As Long, ByVal nCmdShow As Long) As Long

Private Const SW_SHOWMINIMIZED = 2

Private Sub Command1_Click()

  Dim hW As Long

  hW = FindWindow("PROGMAN", 0&)
  If hW Then
     ShowWindow hW, SW_SHOWMINIMIZED
  Else
     MsgBox "Cannot Find Program Manager", _
        vbInformation
  End If

End Sub
```

LISTING 2 ***Minimize the Program Manager.*** *This sample code gets the Program Manager's window handle and then minimizes it with the ShowWindow API call.*

VB4

```
Function QuoteToApostrophe(szText As String)
  Dim szTmp   As String
  Dim szQuote As String
  Dim nPos As Integer
  szQuote = Chr$(34)
  '-- Copy the passed string so
  '   we don't overwrite it.
  szTmp = szText
  Do
     '-- Is there a quote?
     nPos = InStr(szTmp, szQuote)
     If nPos Then
        '-- Yes. replace it with a '
        szTmp = Left$(szTmp, nPos - 1) _
          & "'" & Mid$(szTmp, nPos + 1)
     Else
        '-- No more quotes.. exit.
        Exit Do
     End If
  Loop
  '-- Return the new string.
  QuoteToApostrophe = szTmp
End Function
```

LISTING 3 ***Filter Your Data.*** *The QuoteToApostrophe function helps deal with the issue of data that interferes with Visual Basic syntax. When saving textual data that is going to be retrieved through FindFirst, or another SQL string, first pass it through this filter.*

```
    txtName & Chr$(34)
```

However, there is one gotcha in this case. If the name field contains a quote character, you're in trouble. But, this is probably unlikely. The apostrophe is much more common in language than the quote. What's more, you can get away with substituting an apostrophe for a quote if need be, but you cannot substitute a quote for an apostrophe.

To fix this problem, my team wrote a preprocessor for any text data to be retrieved with Visual Basic. The preprocessor substitutes quote characters with apostrophes. The QuoteToApostrophe Function replaces quotes with apostrophes in a given string (see Listing 3).

The best solution is to use something like SQL Server where your queries are done in stored procedures. In a procedure, you simply use passed variables to represent the text that you are either processing or returning.

EXITUS MAXIMUS

Q I am a beginner with VB Pro 4.0, and I've been trying to locate code on shutting down Windows 95, and even restarting Windows. Can you help?
—Bernie Wilt, Aloha, Oregon

A In Windows 95, the Win32 API function ExitWindowsEx handles this nicely. ExitWindowsEx lets you restart, shut down, force log-off, or log off and log on as another user (see Listing 4).

The first parameter is one of the listed constants, which specify what action to take. The second parameter is reserved and must always be zero. ☒

VB4

```
Option Explicit

Private Declare Function ExitWindowsEx Lib _
  "user32" (ByVal uFlags As Long, _
  ByVal dwReserved As Long) As Long

Private Const EWX_FORCE = 4       '— Force logoff
Private Const EWX_LOGOFF = 0      '- Log off
Private Const EWX_REBOOT = 2      '- Restart
Private Const EWX_SHUTDOWN = 1    '- Shut down

Sub Command1_Click ()

  Dim nRet As Long

  '-- Try to reboot
  nRet = ExitWindowsEx(EWX_REBOOT, 0&)

  If nRet = False Then
     MsgBox "Could not Exit Windows"
  End If

End Sub
```

LISTING 4 ***Exit Windows at Will.*** *This sample code shows how to call the ExitWindowsEx function in 32-bit Windows to restart the computer.*