# It's About Time

Click & Retrieve Source CODE!

## *Create your own generic AboutBox to use in all your applications.*

by Chris Barlow

As you begin to write complete Visual Basic applications, you will start to develop your own common pieces of code that you use again and again. Most programmers find that one of their first common forms is an AboutBox form used to display an application's name, version, and copyright. This form, usually displayed at the startup of the program and from an About menu item on the Help menu, will be visible in every application you write.

I'll show you how to develop a generic AboutBox form that you will be able to use in all of your applications. You can develop your AboutBox to do more for you than just display a simple form—it's a convenient place to embed some of the routines you might want to run at the startup of your application, such as routines that test whether the operating system is Windows 95 or Windows NT, check the type of processor, and so forth. Because VB4 lets you create custom properties and methods for your forms, you can write some public procedures that will be useful to your entire application.

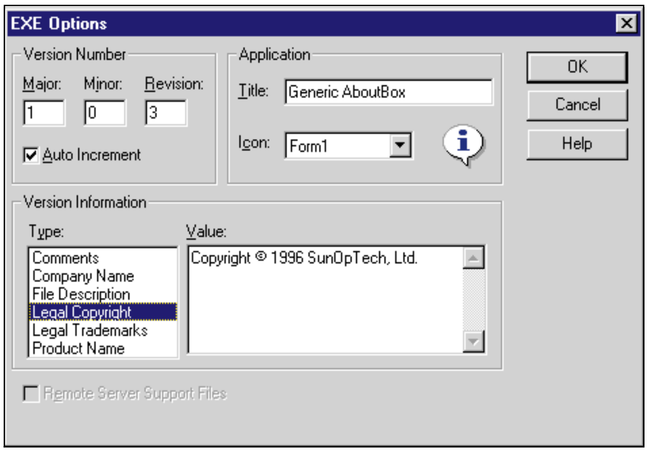
The other neat thing about VB4 is the version information that you can include in your EXE by using the EXE Options dialog (see Figure 1). You can add automatic version numbering, the product name, your company name, copyright and trademark information, and a general comment.

The code you will write from this column will work in any of the VB 32-bit editions. Create a new project, put a command button on Form1, and set the command button's Caption property to "About…" Because this application will call your generic AboutBox, save this project as "Demo AboutBox." Then insert another form into the project and change the Name to AboutBox. You will include this generic AboutBox form in each of your projects.

The AboutBox form will usually be displayed as a modal dialog, so you should change some of its default properties so it looks like a standard modal dialog. First, change the BackColor to a standard gray using the colors dialog. Then set the BorderStyle property to FixedSingle so there are no resize handles on the form. Then change the ControlBox property to False so no control box menu appears when the user clicks on the upper-left corner of the form. Also, set the MaxButton, MinButton, NegotiateMenus, and ShowInTaskbar properties to False.

*Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank and the ObjectJob Systems, where he and Ken Henderson hold a software patent related to decentralized distributed asynchronous object-oriented systems. Chris holds degrees from Harvard Business School and Dartmouth College, where he worked with Drs. Kemeny and Kurtz on the Basic language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.*

You'll need to draw some controls on your AboutBox form. Place an Image control in the upper-left corner for the application icon, and set the Name property to imgIcon. Next to the Image control, add a Label control to display the application title. Set the Name property to lbProgram and set the Font to Bold, 18 point. Then add four more Label controls to the form and set the Name properties to lbVersion, lbCopyright, lbInfo, and lbComments. Add a Line control to separate the sections of the form after the lbCopyright Label control and after the lbInfo Label control. Finally, add a command button in the lower-right corner and set the Name property to butOK (see Figure 2).


EXE Options

Version Number
Major: 1 Minor: 0 Revision: 3
[✓] Auto Increment

Application
Title: Generic AboutBox
Icon: Form1

Version Information
Type: Comments, Company Name, File Description, [✓ Legal Copyright], Legal Trademarks, Product Name
Value: Copyright © 1996 SunOpTech, Ltd.

[ ] Remote Server Support Files

FIGURE 1 ***The EXE Options Dialog.*** *Using Auto Increment is a great way to make sure that you are running the proper version of your EXE file.*

Double-click on the butOK and add code. Note that you should hide the form rather than unload it so that the properties and methods are available to the calling application:

```
Private Sub butOK_Click()
'Hides the form when OK clicked
AboutBox.Hide
End Sub
```

Now is a good time to set up the public properties of this form. The first property will be a public variable of type Form called fCallingForm. This variable will be used to hold a pointer to the form that launched the AboutBox. This way, you can center the AboutBox form over the calling form and display the proper icon:

```
'Calling app set this for icon/centering
Public fCallingForm As Form
```

The calling form also uses the next property to determine whether or not the OK button will be visible on the AboutBox. The user can see the AboutBox at the start of an application because the application's first Form_Load event shows the AboutBox nonmodally, and then the end of the Form_Load event hides it. The program displays the AboutBox this way to give the user something to see while the application's initialization code is executing. Typically, you will not see the OK button during this process. When the user selects the About menu item on the Help menu after the application has started, however, the program displays the AboutBox modally and the OK button is visible:

```
Public ShowOK As Boolean
```

The other variables, listed here, will hold various public properties that will be available to the calling application as long as the AboutBox form remains loaded:

```
Public lPlatform As Long
Public sPlatform As String
Public sVersion As String
Public lProcessor As Long
Public lTotalMem As Long
Public lAvailMem As Long
```

### EXE OPTIONS

Now that you have designed the forms, you should set up your calling application with some information to be displayed in the AboutBox. Display Form1 and set the Icon property to the icon you want to use. Visual Basic provides a good selection of interesting icons in the Icons folder.

Select the Make EXE File menu item from the File menu and click on the Options button to display the EXE Options dialog (see Figure 1). The Version Number is made up of three long integers: Major, Minor, and Revision. If you check Auto Increment, the Revision will be incremented every time you make an EXE file. Then type a more meaningful name in the Application Title. In the Version Information section, add text for the Comment, Company Name, Legal Copyright, and Legal Trademarks sections. Note that you can insert the standard symbol for copyright (©) by holding the Alt key and using the numeric pad to type 0169. Similarly, you can insert the standard registered trademark symbol (®) by holding the Alt key and using the numeric pad to type 0174. Now click on the OK button to make an EXE file.

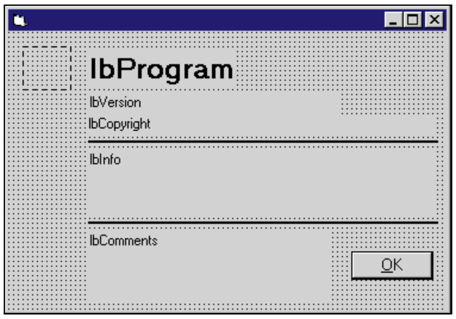


FIGURE 2 ***The AboutBox Form Controls.*** *You can easily change the location of these controls to design your own AboutBox format.*

You should find it easy to add the code in Form1 to drive the AboutBox demo. In the Form_Unload event, add the End statement to terminate the program:

```
Private Sub Form_Unload(Cancel As Integer)
End
End Sub
```

Then add the code in the CommandButton control's Click event to set two properties and show the form. First, set the fCallingForm property to the application's main form. Then set the ShowOK property to True so that the OK button will be displayed. Finally, show the AboutBox with the vbModal argument:

```
Private Sub Command1_Click()
Set AboutBox.fCallingForm = Me
AboutBox.ShowOK = True
AboutBox.Show vbModal
End Sub
```

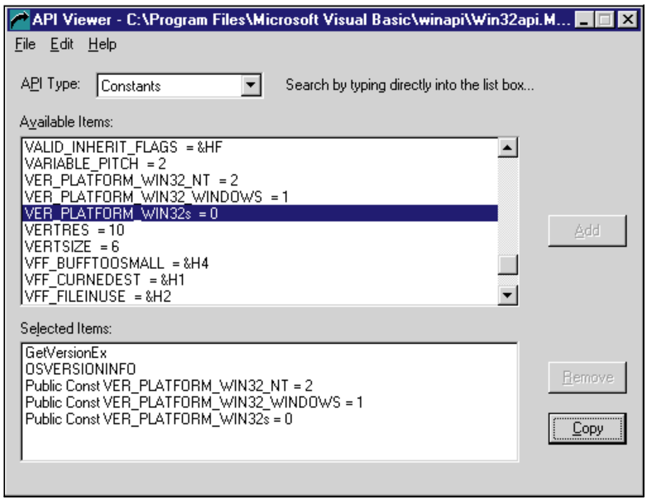
### ABOUTBOX CODE

You're ready to begin adding code to the AboutBox form. You can use your pointer to the calling form in the fCallingForm property to set the Image control's Picture property to the calling form's icon. That way, your generic AboutBox will properly display each application's icon. You can also use the calling form's dimensions to center the AboutBox over that form. This form is also a good place to set the Visible property of the butOK control to the value of the ShowOK property:

```
Private Sub Form_Load()
ImgIcon.Picture = fCallingForm.Icon
'Center the AboutBox over the calling form
Move fCallingForm.Left + _
   (fCallingForm.Width - Width) \ 2, fCallingForm.Top + _
   (fCallingForm.Height - Height) \ 2
butOK.Visible = ShowOK
```

Then set both the AboutBox caption and some of the labels on the AboutBox that came from properties of the application's App object:

```
AboutBox.Caption = "About " $ App.EXEName
lbprogram = App.Title
lbCopyright = App.LegalCopyright
lbComment = App.Comments
```


API Viewer - C:\Program Files\Microsoft Visual Basic\winapi\Win32api.M...
File Edit Help
API Type: Constants
Search by typing directly into the list box...
Available Items:
Add
Selected Items:
Remove
Copy

```
VALID_INHERIT_FLAGS = &HF
VARIABLE_PITCH = 2
VER_PLATFORM_WIN32_NT = 2
VER_PLATFORM_WIN32_WINDOWS = 1
VER_PLATFORM_WIN32s = 0
VERTRES = 10
VERTSIZE = 6
VFF_BUFFTOOSMALL = &H4
VFF_CURNEDEST = &H1
VFF_FILEINUSE = &H2
```

```
GetVersionEx
OSVERSIONINFO
Public Const VER_PLATFORM_WIN32_NT = 2
Public Const VER_PLATFORM_WIN32_WINDOWS = 1
Public Const VER_PLATFORM_WIN32s = 0
```

FIGURE 3 ***The API Viewer.*** *Visual Basic provides most of the functionality you will need, but don't be afraid to use API calls when they can help.*

When the AboutBox is displayed, a decimal point usually separates the Major and Minor components of the application's Version Number, and sometimes a decimal point also separates the Revision. You can write a function that will properly format the version. The function can even use the new Optional argument in VB4. Notice how the IsMissing function formats the number differently if the application does not pass the Revision. If you define this function as public, then it can be used elsewhere by your calling application:

```
Public Function FormatVersion(Major&, _
   Minor&, Optional Revision) As String
If IsMissing(Revision) Then
   FormatVersion = Format(Major, "#0.") & Format(Minor, "00")
Else
   FormatVersion = Format(Major, _
      "#0.") & Format(Minor, "00.") & _
      Format(Revision, "0000")
End If
End Function
```

You can use this FormatVersion function in the AboutBox's Form_Load procedure to display the application's version number:

```
lbVersion = "Version: " & FormatVersion(App.Major, _
   App.Minor, App.Revision)
```

If you run your application now and click on the About button, you should see your generic AboutBox form appear with the information from the App object and the proper icon. While this may be fine for many applications, the AboutBox can also be a convenient place to encapsulate some other procedures. For example, some applications need to operate differently on Windows 95 than they do on Windows NT. You could write a procedure that uses an API call to check the operating system platform. If you save the result in a public property in addition to displaying it on the AboutBox, then you could write code in the calling application to branch to a different block of code based on the platform. Similarly, the version of Windows that is running can be important. The application might need to know, for example, whether it's running on version 3.50, 3.51, or 4.0 of Windows NT. You can add some procedures that call the Win32 API to detect, store, and display some of this information. You may want to add other information that is important to the applications you write.

When you install VB4, it creates a WinAPI folder that contains some handy tools to view the Declares, Types, and Constants for the Win32 API. The ApiLod32 program can load a text file and create an Access database that makes it easy to search for the proper function. Copy the Declare along with the associated Type and Constants to your Visual Basic program. For example, the GetVersionEx function will complete a special user-defined type variable, often called a structure, with information about the Windows version and platform. It uses the OSVERSIONINFO structure and three constants that begin with VER. You can use the ApiLod32 application to search the API database, locate these values, and copy them to your program (see Figure 3). Then you can define a variable, lpVerInfo, of type OSVERSIONINFO (see Listing 1). For more information on the GetVersionEx function, see the Programming Techniques column in this issue.

I suggest writing a separate public procedure called GetOSPlatform to make this API call (see Listing 2). That way, you can use GetOSPlatform for display purposes in your AboutBox, but the calling application can call the function directly if it needs just a portion of the information. Like many

VB4

```
Private Type OSVERSIONINFO
  dwOSVersionInfoSize As Long
  dwMajorVersion As Long
  dwMinorVersion As Long
  dwBuildNumber As Long
  dwPlatformId As Long
  szCSDVersion As String * 128
End Type
' dwPlatformId defines for OSVERSIONINFO structure
Const VER_PLATFORM_WIN32s = 0
Const VER_PLATFORM_WIN32_WINDOWS = 1
Const VER_PLATFORM_WIN32_NT = 2
Private Declare Function GetVersionEx Lib "kernel32" _
  Alias "GetVersionExA" (lpVersionInformation As _
  OSVERSIONINFO) As Long
Private lpVerInfo As OSVERSIONINFO
```

LISTING 1 ***Of Versions and Variables.*** *Once you have used ApiLod32 to search the API database, locate the OSVERSIONINFO structure and three constants that begin with VER, and copy them to your program, you are ready to define a variable, lpVerInfo, of type OSVERSIONINFO.*

VB4

```
Public Function GetOSPlatform() As Long
lpVerInfo.dwOSVersionInfoSize = Len(lpVerInfo)
GetVersionEx lpVerInfo
lPlatform = lpVerInfo.dwPlatformId
If lPlatform = VER_PLATFORM_WIN32_NT Then
  sPlatform = "Microsoft Windows NT"
Else
  sPlatform = "Microsoft Windows 95"
End If
sVersion = FormatVersion(lpVerInfo.dwMajorVersion, _
  lpVerInfo.dwMinorVersion)
GetOSPlatform = lPlatform
End Function
```

LISTING 2 ***Platform on Display.*** *This separate public procedure allows you to use the GetOSPlatform API call for display purposes in your AboutBox. The calling application can call the function directly, however, if it needs only a portion of the information.*

structures, the first member contains the length of the structure, which must be filled out before you call the function. In this case, you set the dwOSVersionInfoSize to Len(lpVerInfo) and then call the GetVersionEx function. The resulting dwPlatformID is saved in the 1Platform property of the form and the text version is saved in the sPlatform property. The sVersion property is set to the formatted Windows version. Finally, the 1Platform is returned in the function call.

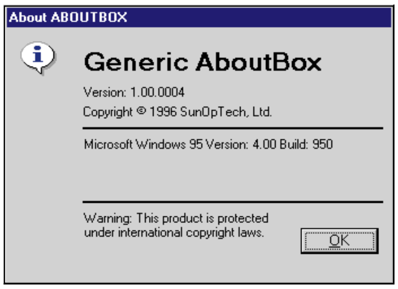
Notice that VB4 lets you call a function procedure as if it were a subprocedure—that is, a procedure with no return value. If the return value does not interest you, let the compiler "swallow it." If you add this single line of code to the Form_Load procedure, the version and platform information will be available in your form's properties to place in the lblnfo control:

```
GetOSPlatform
lblnfo = sPlatform & " Version: "
lblnfo = lblnfo & sVersion
```

Now run the application and you should see your generic AboutBox (see Figure 4). This neat piece of reusable code will fit in most of your applications. You can probably think of several ways to expand its functionality. In the sample code available to Registered members of The Development Exchange Web site (see the "Code Online" box at the end of this column for details), I have added several other API function calls that show the type of processor and current state of virtual memory on the AboutBox form.

## EASTER EGG

Just for fun, add one more bit of code to your generic AboutBox. Many developers hide their "Easter egg" in the AboutBox. An "Easter egg" is that fun piece of code that displays some information about the development team when you press a specific series of keys.


About ABOUTBOX

Generic AboutBox

Version: 1.00.0004
Copyright © 1996 SunOpTech, Ltd.

Microsoft Windows 95 Version: 4.00 Build: 950

Warning: This product is protected under international copyright laws.

FIGURE 4 ***The Completed AboutBox.*** *Get this from the The Development Exchange Web site to find the Easter egg.*

First, add another Image control to the form and change the Name property to imgEgg. Then insert any picture file—perhaps a picture of the development team. You will display this picture only when the user first displays the AboutBox, then double-clicks on the icon in the AboutBox, then clicks on the OK button, then displays the AboutBox again and double-clicks on the icon again. If all of these events happen in this order, you can set the imgIcon control's Picture property to the imgEgg control's Picture property to display your Easter egg.

Add a private integer property to your form called iEasterEgg to hold a counter. Then add this code to the imgIcon control's DblClick event:

```
Private Sub ImgIcon_DblClick()
If iEasterEgg = 0 Then iEasterEgg = 1
If iEasterEgg = 2 Then ImgIcon.Picture = imgEgg.Picture
End Sub
```

Then add this code to the butOK_Click event:

```
If iEasterEgg = 1 Then
   iEasterEgg = 2
Else
   iEasterEgg = 0
End If
```

Try to display the Easter egg with the sample code on the Registered Level of The Development Exchange Web site and see what you get. ☒

**Code Online**

***For all the current issue's listings in one file, go to the Registered Level of The Development Exchange (http://www.windx.com), The Microsoft Network (GO WINDX), or CompuServe* VBPJ *Forum's Magazine Library (GO VBPJ). Development Exchange Premier Level subscribers ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* VBPJ *and* Microsoft Interactive Developer *magazines.***

***It's About Time***

**Locator+ Codes**

EXTRA CODE ONLINE

*Listings ZIP file (free Registered Level): VBPJ0996*
*Listings for this article plus EXE, VBP, FRM, and FRX files associated with the AboutBox application (subscriber Premier Level): GS0996P*