

# Define Preprocessor Symbols

Use preprocessor symbols for conditional compilation in C# and VB.NET, and add hotkey support in VB6.

by Juval Löwy and Karl E. Peterson

## Technology Toolbox

- VB.NET
- C#
- SQL Server 2000
- ASP.NET
- XML
- VB6

## Go Online!

Use these Locator+ codes at [www.visualstudiomagazine.com](http://www.visualstudiomagazine.com) to go directly to these related resources.

### Download

**VS0301QA** Download the code for this article, which includes the CHotKey class, which dispatches messages from a generic BAS module sink to a companion class that in turn notifies its associated form as needed.

### Discuss

**VS0301QA\_D** Discuss this article in the C# forum.

### Read More

**VS0301QA\_T** Read this article online. It includes an augmented code listing containing Initialize/Terminate events and hWnd property procedures.

**VSEP020308CS\_T** "6 Quick C# Bug Fixes" by Bill Wagner

**VB0002FB\_T** "Leverage COM Through the Registry" by Francesco Balena

## Q: Define Preprocessor Symbols

In C and C++, I could define macros and compiler directives using the #define keyword and related compiler instructions. Can I do the same in C#? What about VB.NET?

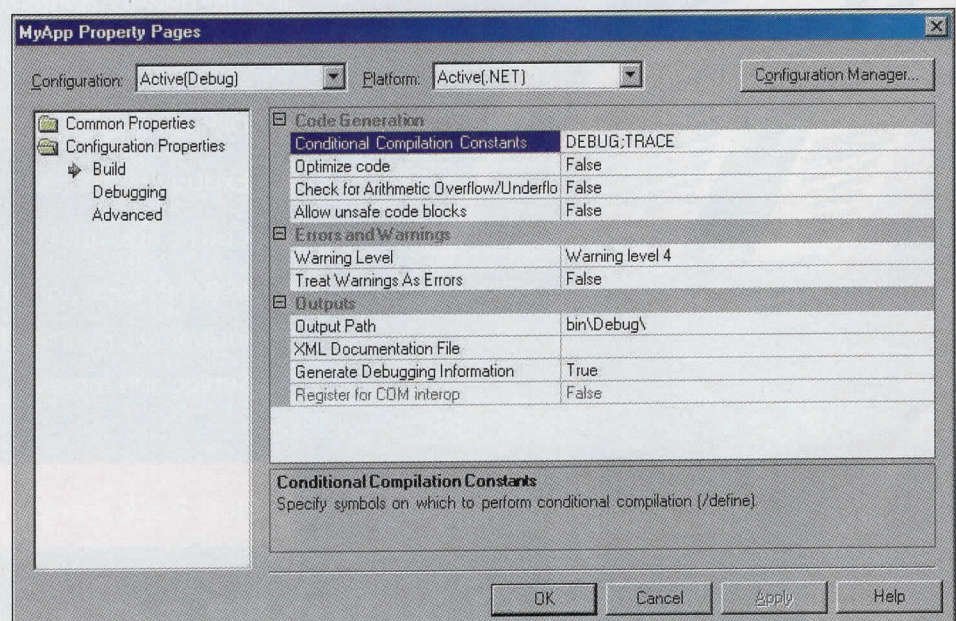
## A:

C# doesn't have macro support, because macros are a bad idea in general: You can't step through and debug them. In C++, you should use inline functions instead of macros, and constants instead of preprocessor definitions. This is why C# doesn't support macros or preprocessor defini-

tions of constants. C++ preprocessor directives were useful for conditional compilation in order to exclude code sections from the build if some preprocessor symbol wasn't defined. C# does support preprocessor directives and conditional compilation, although not in the same way C++ does. You use the #define directive to define a compilation symbol in C#:

```
#define Something
```

This looks similar to C++, but with a few restrictions. The defined symbol must be the first token in the file, and you can only define sym-



**Figure 1 Define New Preprocessor Symbols.** You can define new preprocessor symbols in a C# project's Property Pages instead of placing them in the code itself. By default, VS.NET defines DEBUG and TRACE for debug builds.

bols, not code macros or constants. Like C++, C# supports conditional definitions through the #if, #elif, #else, #endif, and #undef directives. Use these directives as you would in C++:

```
#if Something
    Trace.WriteLine
        ("Something is defined");
#else
    Trace.WriteLine
        ("Something is not defined");
#endif
```

You can nest directive usage, and you can even use the Boolean operators =, !=, &&, and || to construct a compound condition:

```
#if Something && SomethingElse
    Trace.WriteLine("Something "
        "and SomethingElse are defined");
#endif
```

You can use the project's Property Pages instead of defining symbols in the source files. Go to the Build item under the Code Generation group to specify what symbols are defined for this build configura-

## VB6 • Add Hotkey Access

```
Option Explicit
' Win32 API declarations
Private Declare Function IsWindow Lib "user32" _
    (ByVal hWnd As Long) As Long
Private Declare Function RegisterHotKey Lib _
    "user32" (ByVal hWnd As Long, ByVal id As _
    Long, ByVal fsModifiers As Long, ByVal vk As _
    Long) As Long
Private Declare Function UnregisterHotKey Lib _
    "user32" (ByVal hWnd As Long, ByVal id As _
    Long) As Long
' Hotkey messages / modifiers
Private Const WM_HOTKEY As Long = &H312
Private Const MOD_ALT As Long = &H1
Private Const MOD_CONTROL As Long = &H2
Private Const MOD_SHIFT As Long = &H4
Private Const MOD_WIN As Long = &H8
' Error codes
Private Const ERROR_HOTKEY_ALREADY_REGISTERED _
    As Long = 1409&
' Member variables
Private m_hWnd As Long
Private m_HotkeyID As CGlobalAtom
Private m_HotkeyValue As KeyCodeConstants
Private m_HotkeyMods As Long
' Implemented interface(s)
Implements IHookSink
' Public Events
Public Event Hotkey()

' To conserve space, the Initialize/Terminate
' events and hWnd property procedures have been
' removed from this listing. See the Read More
' section of the Go Online box for details on how
' to view the full code listing.

' Public Methods
Public Function SetHotKey(ByVal KeyValue As _
    KeyCodeConstants, Optional ByVal Modifiers _
    As ShiftConstants) As Boolean
    Dim fsModifiers As Long
    ' Must be called after window creation!
    ' Bail if need be ...
    If m_hWnd = 0 Then Exit Function
    ' If the caller passed 0 (zero) as the
    ' KeyValue, unregister the hotkey and set the
    ' global atom to nothing to indicate we're not
    ' hooked.
    If KeyValue = 0 Then
        If Not (m_HotkeyID Is Nothing) Then
            Call UnregisterHotKey(m_hWnd, _
                m_HotkeyID.Value)
            Set m_HotkeyID = Nothing
            Exit Function
        End If
    End If
    ' Unfortunately, there's a conflict between
    ' the native key modifier codes and that used
    ' by the API, so we need to translate.
    If Modifiers And vbAltMask Then
        fsModifiers = fsModifiers Or MOD_ALT
    End If
    If Modifiers And vbCtrlMask Then
        fsModifiers = fsModifiers Or MOD_CONTROL
    End If
    If Modifiers And vbShiftMask Then
        fsModifiers = fsModifiers Or MOD_SHIFT
    End If
    ' Create an instance of CGlobalAtom if we
    ' don't already have one.
    If m_HotkeyID Is Nothing Then
        Set m_HotkeyID = New CGlobalAtom
    End If
    ' Register desired hotkey with system, and
    ' return success/failure.
    If RegisterHotKey(m_hWnd, m_HotkeyID.Value, _
        fsModifiers, KeyValue) Then
        SetHotKey = True
    Else
        ' 1409: Hot key is already registered.
        Debug.Print Err.LastDllError
    End If
End Function
' Subclassing
Friend Function IHookSink_WindowProc(hWnd As _
    Long, msg As Long, wp As Long, lp As Long) _
    As Long
    Dim Result As Long
    Select Case msg
        ' Add handlers here for each message you're
        ' interested in.
        Case WM_HOTKEY
            RaiseEvent Hotkey
        Case Else
            ' Pass along to default window
            ' procedure.
            Result = MHookMe.InvokeWindowProc( _
                hWnd, msg, wp, lp)
    End Select
    ' Return desired result code to Windows.
    IHookSink_WindowProc = Result
End Function
```

**Listing 1** Consider providing hotkey access to your app, especially if you're subclassing the main form's window already. This drop-in-ready class uses a generic hooking mechanism to alert a host form whenever users press the registered hotkey. You can extend this class to handle any number of other subclassing-related tasks as well, because the main hook module first routes messages directed to the main window through the implemented IHookSink\_WindowProc procedure.

tion, instead of placing the definitions in code (see Figure 1). By far, the most common use for conditional compilation is to modify the code's behavior when switching from debug to release builds. When creating a new C# project, Visual Studio .NET defines the DEBUG and TRACE symbols automatically for your use in debug builds, and only TRACE for release builds.

C# takes conditional compilation one step further than C++ by introducing the Conditional attribute defined in the System.Diagnostics namespace. This attribute makes the compiler exclude the method decorated with it from the build, if you don't define a condition:

```
#define MySpecialCondition

public class MyClass
{
    [Conditional("MySpecialCondition")]
    public void MyMethod()
    { ... }
}
```

You can use any symbol you like, but you would use the DEBUG symbol typically. If you don't define the condition, the compiler automatically excludes from the build both the method and any reference to it in the client's code:

```
MyClass obj = new MyClass();
```

```
//This line is conditional
obj.MyMethod();
```

Having the compiler do the method call exclusion automatically is a major improvement over C++. In the past, when C++ developers did code exclusion manually and wanted to put the method calls back in, they sometimes forgot to do it in one or two places and caused a defect. You can use the Conditional attribute only on methods with a void return type, so that you can build when the condition isn't met. In addition, although the compiler doesn't complain, make sure to apply the attribute only if the conditional method has no outgoing parameters, so your code behaves correctly when the symbol isn't defined.

VB.NET uses preprocessor symbols much like C# does. The only difference is that you can't define the symbols in your code; you must do so in the project's Property Pages. —J.L.

## Q: Add Hotkey Support

I want to create a program that usually runs in the background, but pops up and becomes the active task if the user presses (for example) Ctrl-I at any time. Someone pointed me to the RegisterHotKey API. Can you tell me more about it? Is a subclass required in this case?

## A:

You're on the right track. RegisterHotKey is the correct API—this call instructs the system to send you a notification if the user presses the indicated key or key combination. Classic VB doesn't give you

# FREE Subscription!

## XML & WEB SERVICES MAGAZINE

(Formerly XML Magazine)

## Broadens its Focus to Give You More:

**XML & Web Services Magazine**—critical information for technical managers making strategic decisions on interoperable information systems enterprise-wide!

- **More** Web Services and Systems Strategies
- **More** Deployment Tips
- **More** Management Tools
- **More** Resources
- **More** Business Strategies
- **Just Plain More**

**Are you ready for more XML and Web services?**

Subscribe today online at:

**[www.fawcette.com/XML/A28VW](http://www.fawcette.com/XML/A28VW)**

XML is a trademark of MIT. FREE XML & Web Services Magazine subscriptions available to qualified, IT professionals within the U.S. Publisher reserves the right to determine qualification. Please inquire for other subscription rates.

access to the main thread's message queue, so the easiest way of sinking this notification is by subclassing the desired window's message stream and reacting to a WM\_HOTKEY message.

My preference when subclassing is to have the messages dispatched from the generic BAS module sink (by necessity, due to AddressOf limitations) to a companion class that in turn notifies its associated form as needed. The details of this operation could fill an entire column, but luckily, various implementations abound on the Internet, and you can download my solution from the VSM Web site (see the Go Online box for details). I've wrapped this specific task up in a CHotKey class, which you set up like this:

```
' Member variables
Private WithEvents m_Hotkey As CHotKey

Private Sub Form_Load()
    ' Setup instance of hotkey tracking
    ' class, using Ctrl-I as the trigger.
    Set m_Hotkey = New CHotKey
    m_Hotkey.hWnd = Me.hWnd
    m_Hotkey.SetHotKey vbKeyI, _
        vbCtrlMask
End Sub
```

Each time the class sinks a WM\_HOTKEY message, it notifies the form by raising a Hotkey event, and the form then brings itself front and center:

```
Private Sub m_Hotkey_Hotkey()
    ' Reposition, and bring forward.
    With Me
        .WindowState = vbNormal
        .Move (Screen.Width - .Width) \ _
            2, (Screen.Height - .Height) \ _
            \ 2
        .Show
    End With
End Sub
```

You could've wrapped these response details into the companion class just as easily, but they're as likely as not to change from application to application, so I felt they were better left within the individual form.

RegisterHotKey requires four parameters. The first parameter tells the system which window to notify when the user presses a hotkey. The second parameter is an identification value unique to your application. The third and fourth parameters identify the modifier key(s) and virtual key codes for your desired hotkey. The ID value and the modifier key value(s) are slightly counterintuitive in some aspects.

You can generate a system-wide unique value to use as your hotkey's ID value by calling the GlobalAddAtom API and passing a string representation of the current time. GlobalAddAtom returns a value in the range &hC000-&hFFFF, which is the documented range shared DLLs (as opposed to standalone EXEs) should use with RegisterHotKey. Fortunately, no ill effect seems to result from using this convenient number-generation scheme. Be sure to match a call to GlobalDeleteAtom for each call to GlobalAddAtom to avoid overfilling the global atom table.

Another potential problem exists in that the system-defined key modifiers—MOD\_ALT, MOD\_CONTROL, and MOD\_SHIFT—have values that don't match their VB counterparts (ShiftConstants) directly. The Shift and Alt modifier values are flip-flopped in these two constant schemes, so you need to recombine the modifiers for the API if you want to offer the class consumer IntelliSense support for your Modifiers parameter (see Listing 1).

Hotkeys work on a first-come, first-served basis. If another application has registered your desired hotkey already, RegisterHotKey fails and Err.LastDllError contains 1409—the error code for ERROR\_HOTKEY\_ALREADY\_REGISTERED. Your only option should that occur is to fall back to second, or even third, choices and alert your user. On NT-class systems, don't use F12 for your hotkey, because NT reserves this particular key for use by debuggers. —K.E.P.

**Juval Löwy** is a software architect and the principal of IDesign, a consulting and training company focused on .NET design and .NET migration. Juval is a Microsoft regional director for the Silicon Valley, working with Microsoft on helping the industry adopt .NET. He's the author of *COM and .NET Component Services* (O'Reilly). Contact him at [www.idesign.net](http://www.idesign.net).

**Karl E. Peterson** is a GIS analyst with a regional transportation planning agency and serves as a member of the VSM Editorial Advisory Board. Online, he's a Microsoft MVP and a section leader on several DevX forums. Find more of Karl's VB samples at [www.mvps.org/vb](http://www.mvps.org/vb).



Add unique PDF features to all your applications

### Amyni PDF CONVERTER

- Convert all your reports and documents to PDF format.
- Concatenate, merge, compress, embed fonts, add bookmarks, hyperlinks and watermarks...
- Create encrypted, secure documents\*\*
- Optimize documents for the Web through PDF Linearization\*\*

**Other formats:**  
DHTML, RTF, TXT

**Available versions**

- Single-user license
- Server license
- Royalty-free Developer license
- Unlimited Site license
- OEM license

### Amyni PDF CREATOR

- View, Edit and Print PDF\* documents
- Create complex documents, forms and reports
- Create encrypted, secure documents\*\*
- Optimize documents for the Web through PDF Linearization\*\*
- ActiveX component for easy integration into your applications
- Link directly to the database...

[www.amyni.com](http://www.amyni.com)  
Info: [sales@amyni.com](mailto:sales@amyni.com)  
Evaluation: [www.amyni.com](http://www.amyni.com)

**Americas**  
Toll Free: 1-866-926-9864  
Support: (514) 868-9227

### Amyni Clients/Server PDF printing solutions

- Web or application servers delivering reports in PDF format
- Server used to archive or print documents
- Integration with .NET platform

**Europe**  
Sales: (+33) 1 30 61 07 97  
Support: (+33) 1 30 61 07 98

### NEW

For a quick start visit our Technical notes at:  
<http://www.amyni.com/en/support/technotes.html>

All trademarks are property of their respective owners  
2002 AMYUNI Technologies  
All rights reserved. \*Portable Document Format \*\*Professional versions only

