Subclassing & Hooking with Visual Basic

At its heart, Windows is a messaging system. Every second, thousands of messages that control much of the appearance and behavior of the Windows operating system are sent to individual windows and are processed by their window procedures. By exposing events, Visual Basic allows you to determine how your application responds to a small number of messages. But if you want finer control over your application or even over Windows as a whole, you have to work directly with Windows messages. Subclassing & Hooking with Visual Basic shows you how to do just that.

Two major families of techniques are available for intercepting and handling Windows messages: subclassing and hooking. Subclassing can take any of three forms:

- **Instance subclassing**, which allows you to intercept messages sent to a single window or control
- **Global subclassing**, which allows you to intercept messages for one or more windows or controls created from a single window class
- **Superclassing**, which allows you to intercept messages for one or more windows or controls created from a new window class that you define

Hooking, on the other hand, allows you to intercept particular kinds of messages on an application-wide or even a system-wide basis. Windows supports 15 different hooks that allow you to trap such things as mouse events, keyboard events, messages sent to window procedures, or notification that a thread is about to enter an idle state.

Using the techniques of subclassing and hooking, you can do such things as:

- Disallow a user to move or resize a window, or restrict the extent to which it can be resized or the area to which it can be moved
- Modify the look of a window or control
- Modify or disallow keystrokes sent to a window or control
- Create a “recorder” that records and plays back a user’s actions
- Develop computer-based training applications

Subclassing & Hooking with Visual Basic allows you to tap into the full power of Windows programming by showing you how to work directly with the Windows messaging system in ways that are safe and effective. It is the definitive resource for extending Visual Basic using subclassing and hooking techniques.

Visit O’Reilly on the Web at www.oreilly.com

ISBN: 978-0-596-00118-6
US $54.99  CAN $63.99
At its heart, Windows is a messaging system. Every second, thousands of messages that control much of the appearance and behavior of the Windows operating system are sent to individual windows and are processed by their window procedures. By exposing events, Visual Basic allows you to determine how your application responds to a small number of messages. But if you want finer control over your application or even over Windows as a whole, you have to work directly with Windows messages. Subclassing & Hooking with Visual Basic shows you how to do just that.

Two major families of techniques are available for intercepting and handling Windows messages: subclassing and hooking. Subclassing can take any of three forms:

- **Instance subclassing**, which allows you to intercept messages sent to a single window or control
- **Global subclassing**, which allows you to intercept messages for one or more windows or controls created from a single window class
- **Superclassing**, which allows you to intercept messages for one or more windows or controls created from a new window class that you define

Hooking, on the other hand, allows you to intercept particular kinds of messages on an application-wide or even a system-wide basis. Windows supports 15 different hooks that allow you to trap such things as mouse events, keyboard events, messages sent to window procedures, or notification that a thread is about to enter an idle state.

Using the techniques of subclassing and hooking, you can do such things as:

- Disallow a user to move or resize a window, or restrict the extent to which it can be resized or the area to which it can be moved
- Modify the look of a window or control
- Modify or disallow keystrokes sent to a window or control
- Create a "recorder" that records and plays back a user's actions
- Develop computer-based training applications

Subclassing & Hooking with Visual Basic allows you to tap into the full power of Windows programming by showing you how to work directly with the Windows messaging system in ways that are safe and effective. It is the definitive resource for extending Visual Basic using subclassing and hooking techniques.

Visit O'Reilly on the Web at [www.oreilly.com](http://www.oreilly.com)
Dedicated to Kandis, Patrick, and Nicholas
# Table of Contents

Preface ..................................................................................................................... xi

I. Introducing Subclassing and Hooking .............................................. 1

1. Introduction ......................................................................................... 3
   Subclassing ......................................................................................... 4
   The Window Hooking Mechanism ...................................................... 5
   Tools to Aid Us in Our Efforts ............................................................ 6
   A Word of Warning .......................................................................... 16

2. Windows System-Specific Information ........................................ 17
   Inside a Windows Application ............................................................ 18
   Inside the Windows Messaging System ............................................. 42
   Performance Considerations .............................................................. 47

3. The Basics of Subclassing and Hooks ............................................. 48
   What Is Subclassing? ....................................................................... 48
   What Are Hooks? ............................................................................. 67

II. Subclassing and Superclassing ......................................................... 81

4. Subclassing ....................................................................................... 83
   The AddressOf Operator .................................................................. 83
   Some Subclassing Tips .................................................................... 90
   Instance Subclassing: An Example .................................................. 91
   Global Subclassing ........................................................................ 112
   Summary of Key Points in Subclassing ........................................... 131
## Table of Contents

5. *Subclassing the Windows Common Dialog Boxes* .......................... 133
   How Common Dialog Box Subclassing Works ................................. 135
   Using the Resource ........................................................................ 150
   Creating the Subclassing Application ........................................... 154
   Subclassing Common Dialog Boxes Other Than Open and Save As ..... 175
   Problems Subclassing the Find and Replace Common Dialogs .......... 191

6. *ActiveX Controls and Subclassing* ............................................. 195
   Subclassing a Third-Party ActiveX Control .................................... 195
   Subclassing an ActiveX Control
   Created in VB .................................................................................. 196
   Subclassing a UserControl from Within a VB-Created ActiveX Control .... 202
   Creating an ActiveX Control That Subclasses Other Windows .......... 209

7. *Superclassing* ............................................................................ 218
   What Is Superclassing? .................................................................... 218
   How the Superclassing Example Works .......................................... 223
   Peering into the Superclassing Application with Spy++ .................... 243

8. *Debugging Techniques for Subclassing* ...................................... 247
   Where to Start .................................................................................. 247
   Microsoft Tools ................................................................................ 252
   NuMega’s SmartCheck ...................................................................... 254

III. *Hooking* .................................................................................... 267

9. *WH_CALLWNDPROC* ................................................................. 269
   How WH_CALLWNDPROC Works ..................................................... 269
   A Thread-Specific Example .............................................................. 279
   Caveats ............................................................................................ 285

10. *WH_CALLWNDPROCRET* ........................................................... 286
    How WH_CALLWNDPROCRET Works ............................................ 286
    A Thread-Specific Example ............................................................ 289
    Caveats .......................................................................................... 296

11. *WH_GETMESSAGE* ................................................................. 298
    How WH_GETMESSAGE Works ....................................................... 298
    A Thread-Specific Example ............................................................ 304
    Caveats .......................................................................................... 312
Table of Contents

12. **WH_KEYBOARD and WH_KEYBOARD_LL** .................................... 314
    How Keyboard Messaging Operates .............................................. 314
    Description of Hooks ..................................................................... 318
    A Thread-Specific Example ............................................................. 324
    A System-Wide Example ................................................................. 330
    A Low-Level Hook Example ........................................................... 335
    Caveats ............................................................................................ 339

13. **WH_MOUSE and WH_MOUSE_LL** ............................................ 341
    How Mouse Messaging Operates .................................................... 341
    Description of Hooks ..................................................................... 345
    Location of This Hook in the System .............................................. 351
    A Single-Thread Example ............................................................... 351
    A System-Wide Hook Example ........................................................ 355
    Caveats ............................................................................................ 365

14. **WH_FOREGROUNDIDLE** ......................................................... 366
    Description ..................................................................................... 366
    Location of This Hook in the System .............................................. 367
    Background Processing Example ................................................... 369
    Caveats ............................................................................................ 374

15. **WH_MSGFILTER** ................................................................... 376
    Description ..................................................................................... 376
    A Thread-Specific Example ............................................................. 384
    Caveats ............................................................................................ 389

16. **WH_SYSMSGFILTER** .............................................................. 391
    Description ..................................................................................... 391
    A System-Wide Example ............................................................... 393
    Caveats ............................................................................................ 399

17. **WH_SHELL** ........................................................................... 400
    Description ..................................................................................... 400
    Location of This Hook in the System .............................................. 408
    A Thread-Specific Example ............................................................. 409
    Caveats ............................................................................................ 416
# Table of Contents

18. **WH_CBT** ................................................................. 418  
   Description ......................................................................... 420  
   A Thread-Specific Example .................................................. 429  
   Caveats .............................................................................. 443  

19. **WH_JOURNALRECORD** ............................................. 444  
   Description ........................................................................ 444  
   The Macro Recorder Example ............................................ 450  
   Caveats .............................................................................. 457  

20. **WH_JOURNALPLAYBACK** ........................................ 458  
   Description ........................................................................ 458  
   The Macro Recorder/Playback Example .............................. 462  
   Caveats .............................................................................. 470  

21. **WH_DEBUG** ............................................................... 472  
   Description .......................................................................... 472  
   A Thread-Specific Example .................................................. 475  
   Some WH_DEBUG Scenarios ............................................... 485  
   Caveats .............................................................................. 487  

IV. **The .NET Platform** .................................................... 489  

22. **Subclassing .NET WinForms** ....................................... 491  
   What Is a WinForm? ............................................................ 491  
   Instance Subclassing .......................................................... 494  
   Subclassing a Control .......................................................... 511  
   Superclassing ...................................................................... 514  
   Superclassing a Control ...................................................... 519  
   Intercepting Keystrokes in a WinForm ................................. 522  
   Behind the Scenes with Spy++ ............................................ 529  

23. **Implementing Hooks in VB.NET** ................................. 534  
   Hooks and VB.NET .............................................................. 534  
   Delegates ........................................................................... 535  
   Implementing the WH_KEYBOARD Hook ......................... 540  
   Implementing the WH_MOUSE Hook ................................. 547
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>V. Appendixes</td>
<td>555</td>
</tr>
<tr>
<td>A. Messages</td>
<td>557</td>
</tr>
<tr>
<td>B. API Functions</td>
<td>601</td>
</tr>
<tr>
<td>C. Structures and Constants</td>
<td>630</td>
</tr>
<tr>
<td>Index</td>
<td>663</td>
</tr>
</tbody>
</table>
Preface

I came about the idea to write this book when I noticed that there was no one place you could go to learn about subclassing and hooking. Originally, these two subjects were geared toward more advanced C++ developers. Visual Basic (VB) developers were never able to use either of these techniques in a pure VB solution until the advent of Versions 5 and 6, when the language allowed developers to tap into subclassing and hooking techniques. Unfortunately, there remained a problem of insufficient information to help the VB developer understand how, when, where, and why to use these techniques.

In doing some research, I came to the realization that few VB developers had a solid grasp of these techniques. In fact, these techniques were being underused and even misused. Developers were creating incredibly complex and hard-to-maintain code, code that could be greatly simplified if the programmer had only taken advantage of subclassing, hooking, or even both techniques in tandem. This book is meant not only to teach subclassing and hooking techniques, but also to be a single resource from which developers can get information and answers to their questions about these techniques.

Who This Book Is For

This book is for the VB developer who wants a better understanding of subclassing and hooking, as well as knowledge of how to incorporate these techniques into his projects. You need not be an advanced VB developer to learn how to effectively use subclassing and hooking. The beginning of this book lays a solid foundation that will bring the beginning or intermediate developer up to speed.

In understanding subclassing and hooking, you must delve deep into the internals of the Windows operating system. A deep understanding of Windows enables
you not only to gain an understanding of subclassing and hooking, but also to
learn more about the messaging system, which is the heart of the Windows oper-
ating system. In the process, you’ll become a better programmer whether or not
you commonly use subclassing and hooking in your applications.

You will need to utilize many of the Win32 application programming interface
(API) functions to write subclassing and hooking applications. Therefore, a good
knowledge of the Win32 API functions and how they are used within VB is neces-
sary. You can find definitions for most of these API functions, constants, and struc-
tures using the API Text Viewer, which ships with Visual Studio 6. However,
several API functions, constants, and structure definitions are missing from this
viewer. You can find this missing information by searching through both the C++
Windows.h and WinUser.h header files. Additionally, I have provided definitions
for the various API functions, constants, and structures used within this book in
Appendixes A, B, and C.

I will focus on using the Visual Basic language in this book. All the subclassing
code and most of the hooking code will be written in VB. The limitations of VB
come into play when developers try to write system-wide hooks. System-wide
hooks require a true Win32 dynamic link library (DLL) to be created, something
that VB does not support (it enables you to generate only COM DLLs). Visual C++
will enter the picture at this point. The Visual C++ code will be kept to a min-
imum so that you can focus on using VB to implement hooks.

The code in this book was written and tested using Visual Basic Version 6 Service
Pack 3, Visual Basic.NET Beta 1, and Visual C++ Version 6 Service Pack 3. The fol-
lowing operating systems were used while writing this book and its examples:
There are subtle differences in these operating systems that affect how subclassing
and hooking operate on each system. These differences are noted in this book
wherever they will affect the developer.

How This Book Is Structured

This book is grouped into four distinct sections. The first is the introductory sec-
tion, which contains information about the Windows operating system and the
basics of subclassing and hooking. This section lays the foundation that will
enable you to completely grasp the rest of the material in this book. This section
contains the following chapters:

Chapter 1, Introduction
This chapter defines subclassing and hooking and gives several examples of
how each can be used effectively. In addition, the tools used in this book are
explained. The tools include the following:
— Spy++, which enables us to view information about processes, windows, and messages. This tool comes as part of the Visual Studio Version 6 development environment.

— The Dbguproc.dll DLL, which enables us to more easily debug subclassing code within projects written in Versions 5 or 6 of VB. This tool can be downloaded from http://msdn.microsoft.com/vbasic/downloads/controls.asp.

— Microsoft System Information, which enables us to see the installed hooks in a Windows 98 system. This tool can be accessed from the Help → About Microsoft Visual Basic… menu item on the Visual Basic main menu bar. When the About dialog box appears, click the System Info… button to view the Microsoft System Information tool.

— SmartCheck, developed by Compuware Numega Labs, which enables us to watch under the hood as a VB project runs. A 14-day trial version of this software is available from Compuware Numega Labs at http://www.numega.com/evaluations/default.asp.

Chapter 2, Windows System-Specific Information
This chapter contains information on Windows' processes, threads, windows, and messaging system, as well as how they relate to subclassing and hooking.

Chapter 3, The Basics of Subclassing and Hooks
This chapter starts out with a discussion of how subclassing operates and the different types of subclassing at your disposal, and is followed by a discussion of how hooking operates. Finally, the pros and cons of using subclassing and hooks are laid out for you.

The second section covers the technique of subclassing and its variations. The process of subclassing VB forms is discussed, along with subclassing controls, common dialog boxes, and ActiveX controls that you create. Adding to this, global subclassing and superclassing are discussed at length, along with debugging your subclassing code. This section contains the following chapters:

Chapter 4, Subclassing
A discussion of the AddressOf operator is presented first, followed by an in-depth discussion of the various types of subclassing, along with examples of each type. Tips for using subclassing are presented throughout this chapter.

Chapter 5, Subclassing the Windows Common Dialog Boxes
A crash course on using the Windows Common Dialog APIs is presented first. Next, the technique of subclassing the Open and Save As common dialogs is discussed and examples are provided. Finally, the chapter finishes with a discussion of subclassing each of the other common dialogs.
Chapter 6, *ActiveX Controls and Subclassing*

This chapter discusses how to subclass a third-party ActiveX control and an ActiveX control that we create in VB. Next follows a discussion of subclassing a UserControl from within an ActiveX control that is created in VB. This chapter finishes by creating an ActiveX control that is used to subclass a VB form.

Chapter 7, *Superclassing*

The technique of superclassing a window is discussed and examples are provided, and the differences and similarities between superclassing and other types of subclassing are noted.

Chapter 8, *Debugging Techniques for Subclassing*

This chapter presents various techniques used to debug your subclassing application. Along with this discussion, several of the previously mentioned tools are discussed in more detail.

The third section covers the technique of hooking. Fifteen different types of hooks are discussed, each in its own chapter (except for the mouse and low-level keyboard hooks, which are grouped with the regular mouse and keyboard hooks, respectively). This section contains the following chapters:

Chapter 9, *WH_CALLWNDPROC*  
The *WH_CALLWNDPROC* hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 10, *WH_CALLWNDPROCRET*  
The *WH_CALLWNDPROCRET* hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 11, *WH_GETMESSAGE*  
The *WH_GETMESSAGE* hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 12, *WH_KEYBOARD* and *WH_KEYBOARD_LL*  
The *WH_KEYBOARD* and *WH_KEYBOARD_LL* hooks are presented in this chapter, along with details of their operation, examples, and things to watch out for when using them.

Chapter 13, *WH_MOUSE* and *WH_MOUSE_LL*  
The *WH_MOUSE* and *WH_MOUSE_LL* hooks are presented in this chapter, along with details of their operation, examples, and things to watch out for when using them.

Chapter 14, *WH_FOREGROUNDIDLE*  
The *WH_FOREGROUNDIDLE* hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.
Chapter 15, WH_MSGFILTER
The WH_MSGFILTER hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 16, WH_SYSMSGFILTER
The WH_SYSMSGFILTER hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 17, WH_SHELL
The WH_SHELL hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 18, WH_CBT
The WH_CBT hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 19, WH_JOURNALRECORD
The WH_JOURNALRECORD hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it.

Chapter 20, WH_JOURNALPLAYBACK
The WH_JOURNALPLAYBACK hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it. A discussion of how this hook is used in tandem with the WH_JOURNALRECORD hook also is provided.

Chapter 21, WH_DEBUG
The WH_DEBUG hook is presented in this chapter, along with details of its operation, examples, and things to watch out for when using it. Also discussed are ways to enhance the debugging of hooks in your applications.

The fourth and final section covers the techniques of subclassing and hooking as they apply to the new VB.NET language. This section contains the following chapters:

Chapter 22, Subclassing .NET WinForms
The various techniques of subclassing using the new VB.NET language are presented in this chapter, along with examples.

Chapter 23, Implementing Hooks in VB.NET
This chapter discusses how to use hooks with the new VB.NET language. Examples using various hooks also are provided.

Obtaining the Sample Code
The example VB source code from Subclassing and Hooking with Visual Basic is freely downloadable from the O'Reilly & Associates web site at vb.oreilly.com. Just
follow the link to the book’s title page and then click the *Examples* link. The downloadable code will be updated to reflect the most recent beta or production release of the VB.NET platform.

**Conventions Used in This Book**

Throughout this book, we have used the following typographic conventions:

**Constant width**
Indicates a language construct such as a language statement, a constant, or an expression. Interface names appear in constant width. Lines of code also appear in constant width, as do function and method prototypes.

**Constant width bold**
Indicates user input in code sections.

**Italic**
Represents intrinsic and application-defined functions, the names of system elements such as directories and files, and Internet resources such as web documents. New terms also are italicized when they are first introduced.

**Constant width italic**
Indicates replaceable parameter names in prototypes or command syntax, and indicates variable and parameter names in body text.

The owl icon designates a note, which is an important aside to the nearby text.

The turkey icon designates a warning relating to the nearby text.

**How to Contact Us**

We have tested and verified all the information in this book to the best of our ability, but you might find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:
O'Reilly & Associates, Inc.
101 Morris St.
Sebastopol, CA 95472
(800) 998-9938 (in the U.S. or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (fax)

You also can send messages electronically. To be put on our mailing list or to request a catalog, send email to:

nuts@oreilly.com

To ask technical questions or comment on the book, send email to:

bookquestions@oreilly.com

For technical information on Visual Basic programming, to participate in Visual Basic discussion forums, or to acquaint yourself with O'Reilly's line of Visual Basic books, you can access the O'Reilly Visual Basic web site at:

http://vb.oreilly.com

Acknowledgments

I first want to thank Ron Petrusha, my editor, for taking a chance on an unproven author. This is the first book that I have written, and it has been a very fulfilling experience. Under Ron's constant guidance and direction, I have grown considerably in my skill and understanding of the intricacies of writing.

I cannot thank my wife enough; she has been there, enabling me in every way possible to complete this project. Without her, I might have dismissed the idea of writing this book early on, thinking that it would be too far out of my grasp. Hearing her enthusiasm while telling others about the book that I was writing always renewed my determination and energy to finish this book.

I want to thank Mom and Dad for their support and the many long-distance phone calls they made to make sure that I was still on schedule.

I would like to thank William J. Steele, a Microsoft consultant, for taking time out of his busy schedule to discuss some of the questions and theories that I came up with while in the process of writing this book.

The technical reviewers, Daniel Creeron and J.P. Hamilton, did a wonderful job of reviewing the book's technical material. This was no easy job, and I thank them for their valuable feedback.

Finally, I would like to thank Jessamyn Read for doing a great job in transforming the illustrations that I came up with into their final form.
I

Introducing Subclassing and Hooking

This section will lay the basis for developing applications that incorporate subclassing and hooking by examining the Windows family of operating systems as messaging systems. We'll also explore what subclassing and hooking involves, how they differ, and why you'd want to use one technique rather than another. Finally, we'll look at some software tools that you can use both to better understand the operation of Windows, as well as to gather information about how your subclassed windows or hooks are performing.
Chapter 1

Introduction

Windows is a message-based system. This means that every action you take while using the system creates one or more messages to carry out the action. These messages are passed between objects within the system. These messages also carry with them information that gives the recipient more detail on how to interpret and act upon the message.

Clicking a button control provides a good messaging example. This produces not only the message for the mouse button click, but also a wide array of other messages. These include messages to repaint the button in its depressed state, notification messages to inform other objects of the button’s change in state, messages to determine the state of the mouse cursor, as well as others. Even a simple act such as moving the mouse or pressing a key on the keyboard can produce an astonishing number of messages.

In addition to communicating user actions, Windows also uses messages internally to do housekeeping. Messages need to be sent to update the time and date, to notify other objects of a change in state, and even to notify applications when system resources are exhausted.

The Windows messaging system is the heart of the operating system. As a result, the messaging system is very complex.

Subclassing and the Windows hooking mechanism operate on messages within the messaging system. This makes subclassing and hooking two very powerful techniques. With them, we can manipulate, modify, or even discard messages bound for other objects within the operating system and, in the process, change the way in which the system behaves. As you can already have guessed, a thorough understanding of the messaging system is critical to mastering the techniques of subclassing and hooking.
Along with this power comes responsibility. It is up to the developer to make sure that he or she is using these techniques correctly. Windows is very unforgiving if these techniques are used incorrectly.

**Subclassing**

Subclassing techniques deal with intercepting messages bound for one or more windows or controls. These messages are intercepted before they can reach their destination window. The intercepted message can be left in its original state or modified. Afterward, the message can be sent to its original destination or discarded.

By intercepting messages in this manner, we can have a powerful influence on how the window or control will react to the messages it receives. Consider, for example, right-clicking the Visual Basic (VB) text box control. This action causes a default pop-up menu to be displayed containing the following menu items: Undo, Cut, Copy, Paste, Delete, and Select All. Replacing this menu with one of our own is a fairly simple task using subclassing. Subclassing has many other uses as well, such as:

- Determining when a window is being activated or deactivated and responding to this change
- Responding to new menu items that are manually added to the system menu of a window
- Displaying descriptions of menu items as the mouse moves across them
- Disallowing a user to move or resize a window
- Allowing a user to move or resize a window within specified boundaries
- Determining where the mouse cursor is and responding accordingly
- Modifying the look of a window or control
- Changing the way a combo box operates
- Determining when the display resolution has been changed
- Monitoring the system for a low system-resource condition
- Modifying or disallowing keystrokes sent to a window or control
- Modifying how a window or control is painted on the screen

Subclassing opens up a wealth of possibilities to the VB developer—possibilities that ordinarily are completely unavailable, or at least are not easy to implement.

There are three types of subclassing, any of which I will discuss. The first is *instance subclassing*, which enables the application to intercept messages for a single
instance of a window or control. This type of subclassing is the most commonly used. It is used to control, for example, the user’s ability to size a single instance of a window. The second is global subclassing, which makes it possible to intercept messages for one or more windows or controls that are all created from the same window class. All windows derive from some type of class; these classes describe the fundamental look and behavior of windows created from them. Take, for example, a standard button control; each instance of this control derives from a BUTTON class. Using global subclassing, we can change the behavior of the class. This in turn allows us to intercept messages from all window or control instances created from this class. Using global subclassing we can control the user’s ability to size any window created from a particular class. The third type of subclassing, superclassing, is a close relative of global subclassing. Superclassing also has the ability to intercept messages for one or more windows or controls. The difference is that a brand-new window class is created to facilitate this type of subclassing. Similar to global subclassing, superclassing allows users to size a window to be controlled.

The Window Hooking Mechanism

The window hooking mechanism, or hooks, also deals with intercepting messages, but at a much broader scope than subclassing. Hooking allows us to intercept messages at various set points within the operating system. For example, we can intercept a message before and after a window has processed it.

There are several different kinds of hooks, each with their own special purpose and location within the operating system. They are:

- WH_CALLWNDPROC
- WH_CALLWNDPROCRET
- WH_CBP
- WH_DEBUG
- WH_FOREGROUNDIDLE
- WH_GETMESSAGE
- WH_JOURNALPLAYBACK
- WH_JOURNALRECORD
- WH_KEYBOARD
- WH_KEYBOARD_LL
- WH_MOUSE
- WH_MOUSE_LL
- WH_MSGFILTER
- WH_SYSMSGFILTER
- WH_SHELL
Hooks, unlike subclassing, can have an application scope or a system-wide scope. By this, I mean a single hook can intercept specific messages within a single application, or it can be set up to intercept those same messages for all applications running in the system. Hooks give us control over the system, which cannot be achieved with subclassing. The following are just a few of the uses for hooks:

- Modifying messages sent to dialog boxes, scroll bars, menus, or message boxes
- Subclassing a window that resides in a separate process
- Creating a macro recorder that can play back the recorded macro as well
- Developing computer-based training (CBT) applications
- Capturing and modifying mouse or keyboard messages at a system level
- Providing a help function key for menu items and message boxes
- Creating a utility similar to Spy++
- Creating an automated testing application
- Determining when an application is idle
- Modifying mouse buttons and keystrokes for a particular application, or for all applications
- Modifying ALT+TAB and ALT+ESC key functionality

I will discuss all the hooks listed here, as well as show how to apply them to a single application or to all applications running in the system, in Chapters 9 through 21.

**Tools to Aid Us in Our Efforts**

Along with using these advanced techniques, effectively implementing subclassing and hooking in our development work requires that we employ debugging tools beyond the capabilities of the VB debugger.

While developing the projects for this book, I used several software utilities as well as other professional applications that I built. Although you can successfully build applications that subclass various windows or that hook into certain message streams without these utilities, I do not suggest doing so. These utilities give you, the developer, a valuable insight into what is happening inside the system while running your projects in the VB integrated development environment (IDE) and especially at runtime. You will be able to see things operate in a way that is unavailable to you by just using the Visual Basic or Visual C++ development environments.
I would go as far as saying that some of these utilities are necessary to understand how subclassing and hooks work. Otherwise, you will only be blindly plugging code into an application, not fully understanding why you are doing it and what is happening behind the scenes. When the application locks up, debugging it will be frustrating and possibly futile. What I am stressing here is that we, as programmers, must aspire to have an understanding of what we are doing. Without this understanding we cannot hope to reach the more advanced areas of our discipline. Having an understanding of how subclassing and hooks work and interact with the rest of the Windows system will allow you to build successful applications.

I will describe the utilities that I use in the following sections. Although this book will not include a tutorial for operating these utilities, there is some very good documentation in the Microsoft Developer Network Library (MSDN) for Spy++ and PView. The NuMega tools come with their own documentation. Note that some of these tools display different information depending on which operating system you are using (e.g., Windows 9x, NT, or 2000).

**Spy++**

The Spy++ utility is included in the Win32 Software Development Kit (SDK) as well as in Microsoft Visual Studio.

I have used this utility the most, except maybe for NuMega’s SmartCheck utility. Spy++ is one of the most valuable tools when implementing subclassing and superclassing. Spy++ can provide you with all the information you need to verify the state of the application before and after a subclassing operation, as well as all the message information being passed to and from a window. Spy++ is described as a tool for “spying” on different parts of the operating system. This means you can watch as processes, threads, and windows contained within the threads are created and destroyed. Also, you can get valuable information on these objects, some of which is contained within the structures used to create them. But even more useful is the ability to watch in real time as messages flow throughout the system. This, in my opinion, is the most powerful feature of this tool.

Spy++ is a Multiple Document Interface (MDI) application. Let’s start up Spy++ and quickly run through the windows and menus, just to become familiar with getting around in the tool. Each child window within Spy++ displays information on processes, threads, top-level windows, or messages. Let’s start with the Processes window and work our way down.

When Spy++ is started, it will take a snapshot of the system at that point in time. (This does not apply to spying on messages; messages are displayed as soon as they arrive at the window.) Any time an application is started or ended, or its state
changes, you should refresh the display by pressing the F5 key. This will allow you to view the most current state of the system.

The Processes window, which is shown in Figure 1-1, is opened automatically when Spy++ is launched, and displays a list of currently running processes in the system in a tree hierarchy. You can drill down through the processes, which are displayed with a two-gear icon, into the threads within a process, displayed with a single-gear icon. If a thread contains any top-level windows, you can drill down into these windows as well. Top-level windows are windows that have the desktop window as their only window. The top-level windows are displayed with a rectangular window icon. Each item in the tree can be double-clicked to display a dialog box that displays its properties. Within each properties dialog box, except for the Processes Properties dialog, there are hyperlinks to facilitate the process of drilling up and down through processes, threads, windows, and messages. Each item can be right-clicked as well to display a pop-up menu for that item. A separate pop-up menu is displayed for processes, threads, windows, and messages.

![Figure 1-1. Using Spy++ to view the processes currently running](image)

If your primary interest is examining the running threads rather than the processes, you can open the Threads window by selecting the Threads option from the Spy menu. The Threads window, which is shown in Figure 1-2, displays a list of currently running threads in all processes in the system, sorted by thread ID. The display is similar to the Processes window, except that the Processes level has been removed and the running threads are now at the top of the hierarchy. You can double-click and right-click the items in the list, just as in the Processes window.
Finally, if you’re interested in the windows handled by the system and its applications, you can use the Windows window, which also is opened when Spy++ starts. The Windows window, which is shown in Figure 1-3, operates like the Processes and Threads windows and displays a list of all currently running top-level windows and their child windows. With this information, you can see how an application’s designer arranged the user interface (UI) for each application.

Finally, the Messages window, which is shown in Figure 1-4, is the window that we will be most interested in for the applications that we will be building throughout this book.
Clicking the Messages ➝ Options… menu item displays the Message Options dialog box. This dialog contains the following three tabs.

**The Windows tab**

Determines which windows will be watched. Dragging the Finder icon and dropping it onto a window will select that window’s messages to be displayed in Spy++. The checkboxes in the Additional Windows frame allow you to view messages for additional windows.

**The Messages tab**

Because watching every message for every window in the system would produce far too much information to digest, this tab allows you to choose which messages to display. The Messages to View list box displays every message that can be watched. The checkboxes in the Message Groups frame, to the right of the list box, correspond to separate groups of related messages. Most of the checkboxes are self-explanatory; for example, mouse messages correlate with the Mouse checkbox. The Non-Client checkbox relates to messages that usually have the letters NC in them. NC stands for non-client. These messages describe actions originating from the non-client area of a window, such as the title bars and/or a border that is being resized. The General checkbox relates to the messages commonly used in a window, such as WM_COMMAND, WM_TIMER, or WM_PAINT. The Registered checkbox watches for messages defined by the developer using the `RegisterWindowMessage` application programming interface (API) function. The Unknown checkbox watches for messages that are defined to be in the range zero to one less than the WM_USER constant (`&H400`). These are message identifiers that are reserved for the system to use. The Registered checkbox watches for messages that are defined
to be equal to or greater than the \texttt{WM\_USER} constant. These are application-defined messages. After you select all the messages you want to watch, it is a good idea to check the Save Settings as Default checkbox; this way, you will not have to go back every time and re-select the appropriate messages.

\textit{The Output tab}

This tab allows you some control over the message information displayed by Spy++. For this tab, I usually check all the checkboxes grouped in the Show in Message Log frame, and increase the value in the Lines Maximum text box to an appropriate value (somewhere around 3,000). Checking all these checkboxes will display the maximum amount of information about a message. We will not need the Message Origin Time and Message Mouse Position checkboxes until later, when we look into using journaling hooks. Checking the Save Settings as Default checkbox is a good idea here as well.

After selecting which messages to view for which windows, clicking Messages \textlongrightarrow\ Start Logging will allow Spy++ to start displaying the messages that you have selected in the Messages window. The first column of the Messages window will display a line number to denote the order of the messages. The next column is the window handle that the message was directed to. The third column is for message codes. A message code could be displayed as a P, S, s, or R. A message code of P means that this message was posted to the window’s message queue, and that the posting application has continued to execute code and it is not waiting for a return code to be sent back. An S means that the message was sent to this window using the \texttt{SendMessage} API function or one of its derivatives, such as \texttt{SendMessageCallback}. \texttt{SendMessage} will wait for a return value to be passed back to it before continuing to execute code in the calling application. Every message with a code of S is followed by that same message with a code of R. A message code of R means that a return value has been passed back to the caller. A message code of s means that the return value cannot be accessed due to a security restriction. The next column in the Messages window displays the actual message name. This name might be preceded by one or more periods. Each period is a nesting level. This means that a message could be received by the window procedure that, in turn, might fire off several other messages, each of which could be handled before the original message completes processing. If you watched for every message for a particular window, you would notice that some messages having a code of S (sent messages) are not immediately followed by the returned message. Instead several other nested messages might be fired off, each being preceded by one or more periods. It would look something like this in Spy++:

```
000301B0  S  WM\_NCACTIVATE
000301B0  S  .WM\_GETTEXT
000301B0  R  .WM\_GETTEXT
000301B0  R  WM\_NCACTIVATE
```
Values following the message will describe in detail the `wParam` and `lParam` parameters. This description depends on the type of message. By checking the Decoded Message Parameters and Decoded Return Values checkboxes in the Message Options dialog box, you will be able to see a useful description of the `wParam`, `lParam`, and return values of each message.

If you double-click a message, the Message Properties dialog box appears. This dialog basically displays the same information that is present in the Messages child window, but it adds two useful features. The first is the Window Handle field in the dialog box, which is a hypertext link to the window properties. Clicking this field will take you to the Window Properties dialog box for the window with that particular handle. The second is that, if a message contains a pointer to a string or a structure, the Message Properties dialog box displays the actual text in the case of a string or the members and their values in the case of a structure.

Clicking Messages ➝ Stop Logging will stop Spy++ from displaying messages.

**Using Spy++ to examine a VB application**

The parent of all windows is the Desktop window; this window will always be at the top of the hierarchy. Below that window are all the parent windows within each running application. To view the window information for your VB application, search through the list for any parent windows containing the text ThunderRT6FormDC. The caption of your window should be to the left of ThunderRT6FormDC (e.g., “Chapter 4—Subclassing Example” ThunderRT6FormDC). ThunderRT6FormDC is the name of the class from which this form was created. Any form that your application creates will be created from this class and will be considered a parent window. The next level below the parent is the child window. Child windows are usually controls contained within a VB form. Any VB-intrinsic control class will be prepended with the word ThunderRT#. Hence, a command button would be called ThunderRT6CommandButton in Version 6 of VB. Child windows can be parents to other child windows, as happens when controls (such as a PictureBox control) contain controls. Each container control is the parent to the child control(s) that it contains.

If you run a simple VB executable (EXE), you will notice that several different hidden windows are running within the same process as your EXE. These hidden windows are VBBubbleRT6, OleMainThreadWndClass 0x############, VBMsOStdCompMgr, ThunderRT6Main, OleDdeWndClass 0x############, and VBFocusRT6.

Every VB application has a hidden top-level window to which all messages and events are initially sent. This window is derived from the class called ThunderRT6Main. This window owns all other VB forms in the application.
OleMainThreadWndName is a hidden window derived from the OleMainThreadWndClass 0x##### class. It is created by COM to handle message marshaling between COM components.

The VBMsoStdCompMgr class is the basis for several controls developed for Microsoft Office. For example, the Microsoft Office development team created the default toolbar that all windows now use. The letters “Mso” contained in VBMsoStdCompMgr stand for Microsoft Office. While running an application in the VB IDE, this class will drop the VB in its name and be displayed as MsoStdCompMgr instead.

The window created from the VBFocusRT6 class is an invisible proxy form for windowless or lightweight controls. This proxy form is used to receive keyboard, mouse, and system messages for these controls.

All VB form and control classes are superclasses of the VBBubbleRT6 class, which is also a superclass of the ThunderRT6Main class. The VBBubbleRT6 class is responsible for forwarding messages to the appropriate window.

If you bring up a VB application (EXE) in Spy++, you might notice that all windows beginning with the word ThunderRT6 have the same window procedure and class window procedure, except for ThunderRT6Main. To see this, look at the Window Proc field in the General and Class tabs of the Window Properties dialog box. The reason all window procedures use the same function pointer is that all classes beginning with the word ThunderRT6, except for ThunderRT6Main, are derived from the same base class, VBBubbleRT6. In other words, ThunderRT6CommandButton, ThunderRT6FormDC, etc., are all superclasses of the base class, VBBubbleRT6. Other third-party controls will have different window procedures because these controls were created from different window classes.

Another interesting thing to notice is the construction of the ThunderRT6ComboBox control. When viewing a control created from this class, you can drill down one level deeper to discover a standard Windows edit control. (This is not a VB control because the class name is not prefixed with the word Thunder). This shows that a ThunderRT6ComboBox control consists of a VB-defined combo box and a standard Windows edit control. This information will come in handy if you ever need to subclass a VB combo box.

If you double-click a window in the hierarchy, Spy++ opens the Window Properties dialog, which contains information about that particular window. There are five tabs in this dialog box: General, Styles, Windows, Class, and Process. The General tab has several items of interest. The Window Caption displays the caption of a form or button. Some windows, such as the text box control, do not have captions. The Window Handle is the unique, system-wide handle of the selected window. This value is always passed on to the window’s window procedure to
identify which window the message was directed to. When watching messages with Spy++, it is sometimes helpful, especially when debugging, to match up the window handle in the message to an actual window. This will show you which window is receiving the message. The Window Proc is another very valuable field on this tab; this is the field we change to subclass a window. Do not confuse it with the class window procedure, which is contained only within the class and not the window instance. Before subclassing a window, check out the value of its Window Proc. After subclassing a window, recheck this value (don’t forget to hit F5 to refresh the view). You’ll notice that it has changed and is now pointing to the window procedure we defined in our code (BAS) module. After removing the subclass, check this value again and notice that it has been returned to its original number.

The Styles tab has two list boxes, one containing Window Styles and the other containing Extended Styles. These styles have an indirect effect on the messages that a window sends and receives; for details, see the documentation on the MSDN CD-ROM. The Windows tab contains the window handles to the selected window’s parent, the first child, and any owner windows of that particular window. The parent window handle is useful to determine to which window a notification message will be sent. The owner window of any VB application is the ThunderRT6Main class. The handles listed on this tab are hyperlinks to the windows that they reference. Clicking the hyperlink displays the properties of that window in the Window Properties dialog box.

The Class tab is also very useful for determining subclassing information. The Class Name is the name given to this class when it was registered with the system. The Class Atom is the unique 16-bit integer value that identifies this class; it is returned from the call to RegisterClassEx. Every class also has styles similar to a window style. The CD_DBLCLKS, CS_HREDRAW, CS_VREDRAW, and CS_SAVEBITS class styles can have an effect on the messages sent and received by a window. There is one more bit of critical information on this tab: the Window Proc field, which translates to the window procedure of the class, not the window. Using this field, we can determine when a window has been subclassed. All we have to do is see whether the Window Proc values on the General and Class tabs match. If they do, the window has not been subclassed. If they are different, the window is subclassed. Global subclassing and superclassing are two techniques that will modify the class window procedure directly.

This completes the whirlwind tour of Spy++. If you do nothing else, familiarize yourself with this utility. Even if you never use subclassing or hooks, you can still use this utility to debug application messages. Spy++, as well as watch and learn how an application is set up and operates within the system.
NuMega SmartCheck

SmartCheck, developed by NuMega (http://www.numega.com), is a tool designed to help developers track down bugs and correct them. As an additional bonus, with this tool you can take a look under the hood of a compiled VB application and watch how it works. To tell you how to use it would require more than just one chapter. I will leave that subject to the documentation provided with the tool.

SmartCheck really shines when your application throws a General Protection Fault (GPF). If you have ever had the pleasure of tracking down a GPF in VB without any tools, you will understand what I mean. Because VB hides many lower-level system operations from the developer, it is difficult to determine why a piece of code will produce a GPF, and if it happens at random intervals, it is nearly impossible to figure out. SmartCheck goes over each executing line of code with a fine-toothed comb. That way, it is possible to see a problem such as a string returned from an API call that is overwriting its bounds and setting up a time bomb that will eventually blow up in your face. Try finding that problem yourself without any tools.

SmartCheck tracks all sorts of items such as API calls, bad calls to intrinsic VB functions, value coercion problems, messages, memory leaks, hooks, and much more. We will be paying close attention to the hooks, messages, and API calls that SmartCheck will be watching throughout the book.

Dbgwproc.dll

This is a dynamic link library (DLL) that you can get from the Microsoft web site that helps with troubleshooting VB applications that use subclassing. As we shall see later, debugging a VB application that uses the subclassing and hooking techniques described in this book is difficult because, having circumvented VB's own protective mechanisms, you cannot go very far with the debugging tools VB provides. Dbgwproc.dll is a tool that, when your application is running in debug mode, will allow you to trace through your application without crashing. You can find this tool on the Microsoft web site at http://msdn.microsoft.com/vbasic/downloads/controls.asp. I will talk about this tool in Chapter 8.

Microsoft System Information

This utility can be run under Windows 98 by selecting Start → Programs → Accessories → System Tools → System Information. Sadly, the System Information application provided with Windows NT/2000 does not show this information.

This tool can display all the system-wide hooks that are currently installed in the system. To do this, expand the System Information → Software Environment
WM_GETMINMAXINFO message, 562
WM_HOTKEY message, 589
WM_INITDIALOG message, 166, 575
WM_KEYDOWN message, 314, 571
WM_KEYUP flag, constants, 660
WM_KEYUP message, 572
WM_KILLFOCUS message, 559–560
WM_LBUTTONDOWN message, 344, 579
WM_LBUTTONUP message, 344, 580
WM_MBUTTONDOWN message, 582
WM_MBUTTONUP message, 583
WM_MOUSE hook, filter functions and, 347–351
WM_MOUSEMOVE message, 103, 579
WM_MOVE message, 558
WM_MOVING message, 588
WM_NCACTIVATE message, 565
WM_NCCALCSIZE message, 564
WM_NCCREATE message, 112, 563
capturing, superclassing, 221
WM_NCDESTROY message, 563
WM_NCHITTEST codes, constants, 660–661
WM_NCHITTEST message, 103, 564–565
WM_NCLBUTTONDBLCLK message, 567
WM_NCLBUTTONDOWN message, 105, 212, 567
modal loops and, 377
WM_NCLBUTTONUP message, 212, 567
WM_NCMBUTTONDOWN message, 569
WM_NCMBUTTONDOWN message, 569
WM_NCMBUTTONUP message, 569
WM_NCMOUSEMOVE message, 566–567
WM_NCPAINT message, 565
WM_NCRBUTTONDOWN message, 568
WM_NCRBUTTONDOWN message, 568
WM_NCRBUTTONUP message, 568
WM_NCXBUTTONDBLCLK message, 571
WM_NCXBUTTONDOWN message, 570
WM_NCXBUTTONDOWN message, 570
WM_NCXBUTTONUP message, 570
WM_NOTIFY message, 189
WM_NULL message, 557
WM_PAINT message, 560
WM_PARENTHIT message, 586–587
WM_PSD_ENVSTAMPRECT message, 599
WM_PSD_FULLPAGERECT message, 598, 600
WM_PSD_MARGINRECT message, 599
WM_PSD_MINPAGERECT message, 599
WM_PSD_PAGESETUPDLG message, 598
WM_QUEUESYNC message, 432
WM_QUIT message, 46, 227, 560
posting, 617
WM_RBUTTONDOWN message, 582
WM_RBUTTONDOWN message, 581
control superclassing, 522
WM_SETFOCUS message, 581
WM_SETTEXT message, 103, 561
WM_SETFOCUS message, 559
WM_SETTEXT message, 560
WM_SHOWWINDOW message, 561
WM_SIZE message, 558–559
wParam values, 661
WM_SIZING message, 106, 587
WM_SYSCHAR message, 575
WM_SYSKEYDOWN message, 105, 576–577
wParam constants, 662
WM_SYSKEYDOWN message, 314, 573–574
WM_SYSKEYUP message, 574
WM_SYSCOMMAND message, 105, 576
WM_WINDOWPOSCHANGED message, 562
WM_WINDOWPOSCHANGING message, 562
WM_XBUTTONDOWN message, 585–586
WM_XBUTTONDOWN message, 584
WM_XBUTTONDOWN message, 585
WNDCLASSEX structure, 28–30, 649–650
base class and, 218
lpfnWndProc window class element, 33–34
WndProc method, overriding, WinForms and, 496–500
WndProc window, switch statement, 38
wParam parameter, 12
high-order words, 173
low-order words, 173
WS_CHILD style, 147
WS_CLIPSIBLINGS style, 147
X
XButton, constants, 662
Z
Z-order, windows, 18, 20
ALT-ESC key combination, 315
ALT-TAB key combination, 315
ignoring, 20
modifying, 625–627
stacked windows, 18–19
About the Author

Stephen Teilhet earned a degree in Electrical Engineering, but soon afterwards began writing software for the Windows platform. For the last eight years, he has worked for several consulting firms on a wide range of projects, specializing in Visual Basic, Visual C++, MTS, COM, MSMQ, and SQL Server. Stephen currently works for Compuware Numega Labs in Nashua, New Hampshire, where he is immersed in the Microsoft .NET technologies.

His interest in understanding the underlying mechanisms of software, rather than just using the software, is what led him to write this book. In addition to this book he has also written for the Visual Basic Programmers Journal. Currently, his two main interests involve studying the inner workings of compilers and understanding the intricacies and operation of the .NET framework and languages.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animals on the cover of Subclassing and Hooking with Visual Basic are common brush-tail possums (Trichosurus vulpecula). These small Australian marsupials are about one-sixth the size of a housecat. They have big ears, long whiskers, and a pointy snout tipped with a pink nose; their fur color ranges from light gray to dark brown. A brushtail possum pregnancy lasts only about 17 days, but, after birth, the single infant spends up to 5 months in his mother's pouch and then is dependent on mom for another 1 to 2 months longer.

Ordinarily, brushtail possums are tree-dwellers; their long, prehensile tails and opposable digits are ideal for grasping tree branches. They dine on leaves, fruits, and flowers and were once common all over Australia. But now they're largely absent from the interior of the country, and they can too often be found living in the eaves and attics of suburban houses, raiding trash cans for their dinners. Introduced to New Zealand in 1840, they're a major pest species in that country as well, and population control efforts include attempts to popularize possum-fur garments.

Leanne Soylemez and Matt Hutchinson were the production editors for Subclassing and Hooking with Visual Basic. Leanne Soylemez was the proofreader. Audrey Doyle was the copyeditor. Audrey Doyle was the copyeditor. Linley Dolby provided quality control. Johnna VanHoose Dinse and Brenda Miller wrote the index. Ellie Volckhausen designed the cover of this book, based on a series design by Edie Freedman. The cover image is from The Illustrated Natural History: Mammalia. Emma Colby produced the cover layout with QuarkXPress 4.1 using Adobe's ITC Garamond font. David Futato designed the interior layout based on a series design by Nancy Priest. Anne-Marie Vaduva converted the files from Microsoft Word to FrameMaker 5.5.6 using tools created by Mike Sierra. The text and heading fonts are ITC Garamond Light and Garamond Book; the code font is Constant Willison. The illustrations that appear in the book were produced by Robert Romano and Jessamyn Read using Macromedia FreeHand 9 and Adobe Photoshop 6. This colophon was written by Leanne Soylemez.

Whenever possible, our books use a durable and flexible lay-flat binding. If the page count exceeds this binding's limit we use perfect binding.