Desaware

5 Town & Country Village, Suite 790 San Jose, CA 95128
(408) 377-4770 fax: (408) 371-3530

Online Catalog

We are pleased to present you with detailed information and demonstrations on our products.

SpyWorks-VB The ultimate low level programming toolkit for Visual Basic.

VersionStamper-VB Eliminate VBX and DLL incompatibilities when distributing your Visual Basic applications

The Common Dialog Toolkit - Tap into the full power of the Windows common dialog from Visual Basic.

CCF-Cursors - Create and use customer cursors (Mousepointers) in your Visual Basic applications.

The Custom Control Factory - The ultimate button factory: completely customizable including 3D effects, animation and virtually unlimited states.

Visual Basic Programmer's Guide - Learn to use the Windows API from Visual Basic.

How Computer Programming Works - A new book for non-programmers and beginners.

Technical Notes - A brief letter from the President of Desaware.

Ordering Information - The bottom line: prices and how to order.

SpyWorks-VB

SpyWorks-VB allows you to do virtually anything in Visual Basic that is possible with other languages. It does this by supporting powerful Windows programming techniques that are not part of Visual Basic. Subclassing allows you to intercept any Windows message going to a form or control. Callbacks allow you to provide function pointers to Windows and other DLL's which trigger VB events when called. Hooks give you system or task access to keyboard and mouse events.

- Philosophy of SpyWorks - An unusual approach to extending Visual Basic. progman.hlp
- Run the SpyWorks-VB Demo

Custom Controls:

- SBC.VBX - Generic subclassing custom control.
- CBK.VBX - Callback custom control.
- SBCKBD.VBX - Keyboard hook custom control.
- SBCHOOK - Windows hooks custom control.
- SBCEASY.VBX - Scrolling forms, rollups and more.
- DWSPYDLL.DLL - Extensive function library.

Debugging Tools:

- SpyMsg - View Windows messages and Visual Basic events.
- SpyParam - Detect Windows API parameter errors.
- SpyWin - Determine windows hierarchy.
- SpyMem - Analyze Windows memory, resource use, etc.
- SpyMenu - Read the menu structure for any window.
- SwIniEdt - Organize messages into groups.
- SpyNote #1 - Application notes

The Philosophy of SpyWorks-VB

One of the great advantages of Visual Basic over any other Windows development language is that it is extremely "safe" to use. In theory, it is impossible for a Visual Basic programmer to crash the system or to cause a General Protection Fault or other system error from within Visual Basic. This makes programming extremely efficient - you can quickly experiment and modify your code, examining intermediate values in the immediate window as needed. This differs markedly from other development systems where bugs frequently cause memory corruption, use of invalid pointers, and other errors that require sophisticated debuggers and many reboot/restart cycles.

Visual Basic protects you from your mistakes, but there is a price to pay for this safety. Visual Basic only implements a subset of the features that are available under Windows. As a result, there are many tasks that are difficult or impossible to do with Visual Basic directly. There are currently two methods for extending Visual Basic. You can directly access Windows API functions from VB, or you can write Dynamic Link Libraries or VB custom controls using C or C++ and traditional Windows programming techniques.

SpyWorks-VB provides a third approach to extending Visual Basic. It consists of a number of tools and programs to support extending the capabilities of Visual Basic from within the VB environment. No C or C++ language or other tools are required. These tools take two forms: Extension tools and Debugging tools.

The extension tools consist of five custom controls: SBC.VBX (Generic subclass) CBK.VBX (Generic Callback), SBCKBD.VBX (Keyboard hook), SBCHOOK.VBX (Windows hooks) and SBCEASY.VBX (subclassing solutions). SBC.VBX is a powerful custom control that allows you to intercept and manipulate the underlying Windows message stream for any Visual Basic form or control (including custom controls). CBK.VBX is a custom control that contains a pool of function addresses that can be uses as "callback" functions for Windows API functions. With these two controls and access to the Windows API functions, you can do virtually anything in Visual Basic that you could do using a dynamic link library. SBCKBD.VBX is a task or system keyboard event detector which allows you to intercept keyboard entries before they are received by Visual Basic or another application. SBCHOOK.VBX uses Windows "hooks" and is ideal for intercepting windows messages on a global basis. SBCEASY.VBX implements common subclassing tasks including Mouse tracking, scrolling and virtual forms, tiny captions and rollup windows.

The debugging tools are designed to help you to work with the extension tools and with the Windows API. The SPYPARAM.VBX custom control detects Windows API parameter errors to help track down elusive bugs in the API interface. SpyMsg.exe is a powerful detection program that allows you to detect and record both the Windows messages and Visual Basic events that are sent to a form or control. SpyMem.exe is a memory browsing tool to help track memory usage - indispensable for making sure that all memory blocks and GDI objects are properly released. SpyMenu.exe helps you to understand the menu structure for a window. SpyVBX.exe can be used to analyze any VB form or control.

There is a price to pay for this power. SpyWorks-VB is one of the few products you will ever buy that proudly and clearly claims to make it much easier to crash your program. This tradeoff is common in the programming world: with power and flexibility comes a reduction in the level of protection provided by the environment. SpyWorks-VB, by intercepting the Windows message stream and improving access to the Windows API makes it very easy to crash not only the application that uses it, but other applications as well.

SpyWorks-VB is designed for the intermediate to advanced Visual Basic programmer who has a knowledge of how to use the Windows Application Programmer's Interface (API). Some components of the package are useful to anyone (the SpyMsg.exe program, for example, by tracing all VB events that occur in an application, provides a useful debugging tool even for beginners). However, a good understanding of Windows is required to really use this package successfully.

If you already know Windows well, you will find SpyWorks-VB extremely easy to use. Simply consider the task you wish to perform and how you would do it in C or C++, then write it in Visual Basic. Any code that you would normally write in a Windows procedure in response to a Windows message, you can place in an event in an appropriately configured SBC.VBX custom control. Any time you need a

function pointer to use as a callback function, simply use a CBK.VBX custom control to obtain one. Where you would use Windows hooks, use SBCHOOK.VBX.

If you have never programmed in Windows, you must learn about it in order to use this package effectively. The professional version of Visual Basic comes with the help file for the Windows API - this can be used as a reference for all available Windows API functions and Windows messages.

There are other references available. Desaware strongly recommends "PC Magazine's Visual Basic Programmer's Guide to the Windows API" as a supplementary text for working with SpyWorks. This book was written by the author of SpyWorks, and in fact includes a subset of the SpyWorks package (including a limited subset of the CBK.VBX custom control). You can order this book at a 20% discount directly from Desaware.

SpyWorks-VB is a tool. Most add-on programs have a clearly defined set of operations that they can perform. Their documentation can, and often does, include extensive examples to show the capabilities of the product. A dozen books and manuals could not begin to do this with SpyWorks, because it has no clearly defined set of operations. It is a can-opener that enables you to tap the full power of Windows from within Visual Basic. This manual includes a number of examples of how the extension controls can be used, but we cannot even begin to guess at the potential of what can be accomplished. Try running the SpyDemo demonstration program to get a taste of some of the possibilities.

SBC.VBX - Generic Subclass Control

- Detect windows messages for any window, form or control in the system and trigger a VB event when it occurs.
- Detect messages before the Windows default processing, after the Windows default processing, or simply post it to yourself for later examination.
- When detecting messages before the default processing, you can change the message or its parameters, or cause the message to be completely ignored.
- Detect messages that are sent from Windows, or those sent directly by Visual Basic - this allows you to see the internal VB messages in VB forms and controls.
- Even detect messages for the new VB2/VB3 graphical controls.
- Subclass windows in other applications - you can effectively create VB programs that add their own menu commands to another application so that your VB program acts as an Add-on to that application.
- Specify exactly which messages to detect - this minimizes the overhead to provide the fastest possible performance.
- Detect registered windows messages.
- Delayed event processing - allows you to "post" an event to yourself without setting up a timer control.
- Each SBC.VBX control can subclass multiple windows or controls (limited only by memory).
- The SBC.VBX control can be configured to queue up messages received while a message box is visible (overcoming a limitation in VB).

CBK.VBX - Generic Callback Control

- Provides a pool of function addresses which can be passed to the Windows API or any Dynamic Link Library for use as callback functions.
- Over a dozen standard parameter sequences (callback function types) are provided, along with Desaware's guarantee: If you need a callback type that is not provided we'll add it and send you and update - *fast*.
- Over 100 function addresses in the internal function pool (far more than you're ever likely to need, but we'll add more if you need them).
- Support for both the standard 'Pascal' and the 'C' calling convention.
- Cross task callbacks makes it possible to use callbacks that occur in the context of a different task.
- Posted callbacks allows use of the CBK control even with callbacks which restrict use of API functions to PostMessage only (low or interrupt level).
- Can be page locked for use with low level multimedia API callbacks.

SBCKBD.VBX - Keyboard Hook Control

- Receive all keystrokes sent to any task. Keystrokes are detected before they are sent to the application, thus you can even trap the enter key, control break, and tab key.
- Place a system wide keyboard hook. This allows your VB program to be a "hotkey" type application which is triggered by a specific key sequence regardless of which application is active.

SBCHOOK.VBX - Windows Hooks Control

- Implementation of Windows hooks including WH_MOUSE, WH_GETMESSAGE, WH_MSGFILTER, WH_SYSMSGFILTER and WH_CALLWNDPROC - Allows interception of messages going to many controls without subclassing each one.
- Full control over scope of message detection - per form, per task, or System-wide.
- Can be configured to queue up messages received while a message box is visible (overcoming a limitation in VB).
- Specify exactly which messages to detect - this minimizes overhead to provide the fastest possible performance.
- Detect registered windows messages.
- Detect messages as they occur, or post them for later processing.
- Ability to change or discard messages (depending on the hook and message)

SBCEASY.VBX - Windows Hooks Control

- True 'MouseEnter', 'MouseExit' event for every form or control in your application - including Visual Basic graphical controls. Ideal for status bars.
- Easy detection of menu select events - also ideal for status bars.
- Determine update area of a window during VB Paint events.
- Create 'Tiny' captioned windows similar to the VB control toolbar out of forms and most container controls.
- Ability to modify or draw your own window captions.
- Create rollup windows (drop down form) out of forms and most container controls.
- Intercept, block or modify system menu commands.
- Turn on scrollbars for any form, picture control and many container controls.
- Perform automatic scrolling on any form, picture control and many container controls.
- Turn forms, picture controls and most container controls into "virtual" controls where control acts as a window to a larger area than is visible.

DWSPYDLL.DLL - Function Library

- Dozens of functions that allow you to access Windows API functions easily and to manipulate data.
- Functions for obtaining the address of VB variables, performing huge memory calculations, splitting integer and long variables into component parts, and so on.
- Functions to provide access to the DeviceCapabilities, DeviceMode and ExtDeviceMode functions in any printer driver.
- A subset of the Visual Basic API including functions to obtain the name of a form or control, to determine the hierarchy of forms and controls, and so on.
- Indirect property access - read/write property values to any VB form or control in any application.
- Read/write picture properties. Allows easy combination of API picture functions with VB properties.
- Complete set of functions for analyzing the internal structure of any VB form or control.

SpyMsg

- View any or all of the messages and events going to a form, control or task.
- Your choice of detection technologies to use: subclassing, SendMessage hooks or GetMessage hooks.
- Set the scope of message and event detection: Any combination of one or more windows, any task, or system wide.
- Detect VB events for any form or control. Events are detected even if there is no code attached to the event.
- View detailed information on the form or control that received the message or event.
- View information on the parameters to the message or event.
- Easy selection of Windows to subclass - use a Windows hierarchy "tree" or point at a window and click.
- Detects messages and events for VB2 graphical controls.
- Write output to a file to create a complete history of the messages and events that occur in your application.
- Parameter descriptions for most standard windows messages.

SpyParam

- Detect API parameter errors as they occur (under Windows 3.1, many parameter errors are simply ignored).
- Trigger a VB runtime error when an API parameter error occurs. This can save hours in trying to track down an obscure problem.

SpyWin

- The first truly Visual Basic aware window/form/control browser.
- Windows are organized by task, making it easy to track down the windows for a given application.
- Provides a great deal of information about each window/form or control, including VB specific information such as control name and model information, styles, class information and so on.
- Allows you to select windows for further processing by your own programs.
- SpyWin includes complete Visual Basic source code!

SpyMem

- Browse the Windows global heap, and any local heap (including USER and GDI). View any block of memory.
- Browse the Windows task and module list. Free any module (ideal for cleaning up after a GPF).
- View the GDI resource list. View information about each resource (for example: if it is a brush, you will see an area painted by the brush - if it is a bitmap, you can view the bitmap).
- Automatic comparison of the current heap with a saved reference - makes it easy to track down global memory or resources that are not being freed.

SpyMenu

- Analyze the system menu and regular menu for any window in the system.
- Includes complete Visual Basic source code.

SpyVBX

- Analyze the model, property and event structures for any Visual Basic form or control including custom controls.
- Direct analysis of VBX files that export MODEL information using VBModelInfo.
- Dump complete analysis of a form, control or custom control to a text file.
- Supports version 1.0 through 3.0 of Visual Basic.
- Demonstrates how to read or write properties of any VB form or control.
- Includes complete Visual Basic source code.

SwIniEdt
Provides easy editing of the SpyWorks.ini initialization file. Used for creating groups of messages that are used by the other SpyWorks-VB applications and debugging tools.

SpyNotes #1

SpyNotes are a series of application paks that will deal with a particular subject and include both code and documentation in Windows Help file format. SpyNotes #1 is included free with SpyWorks-VB version 2.0. Refer to the On-line help for further information on this package.

Visual Basic Programmer's Guide to the Windows API

Build on your knowledge of Visual Basic to become a Windows expert by learning to take advantage of the hundreds of functions contained in the Windows API. No knowledge of 'C' or prior Windows knowledge is required. Written by the president of Desaware, it covers virtually every aspect of Windows and includes extensive sample code. Over 1000 pages plus disk. Published by Ziff-Davis Press. ISBN: 1-56276-073-4. List is $34.95, but we have it for only $27.96 (20% off).

How Computer Programming Works

This new book from Ziff-Davis Press is part of their best selling "How It Works" series. Written by Daniel Appleman, president of Desaware, this fully illustrated book introduces the fundamentals of programming and computer science in a way that is easy to understand for non-programmers and beginning programmers alike.

Ideal for helping friends and family to understand what it is that you actually do during all those hours that you sit in front of your computer!

Available from Desaware for 20% off list - only $19.96 (*available June 1994)*

VersionStamper-VB

You get a technical support call....

"Your program has stopped working!"

Did someone install an older version of a VBX or DLL required by your application?

Is another program using an incompatible version of a VBX or DLL that it loaded from a private directory, preventing yours from loading?

Is an incompatible version of a VBX or DLL present somewhere in the PATH and being loaded before the one that is required?

Did the user delete a component that is needed?

Did the user reinstall your application using an older release disk - copying over your most recent files because your VB application does not contain version information?

If only your application could tell you what was wrong....

Now it can....

Solving the Installation Problem.

Make no mistake - there is no simple "fix" to the problem of safely distributing Visual Basic applications. However, if you have complete information about the incompatibilities that occur and the versions of both components and VB executables, it is possible to quickly determine exactly what problems exist. Once you have this information, it becomes easy to find the correct solution.

Embedding version resources into VB applications.

The first, and most obvious need is to embed a standard Windows version resource into executables created by Visual Basic. This will make it possible for any standard Windows installation program to correctly handle your Visual Basic programs and prevent the accidental overwriting of later versions of your executables.

VersionStamper-VB contains the dwvstamp.vbx custom control which makes the process of embedding a version resource into your executable completely automatic. The version information is entered by setting the control's properties at design time just as you would set the properties for any custom control. This information will be converted into a standard version resource automatically during the compilation process.

Solving the Distribution Problem.

The key to eliminating problems relating to the distribution of Visual Basic applications is to be able to determine, quickly, whether all of the components required by the application are, in fact, present in the runtime environment.

Embedding component version information into your application.

The dwvstamp.vbx control makes it possible to embed into the executable a complete list of every DLL, VBX or other software component that is needed by your program. This embedded component information includes the required version information for each component. You can also specify the conditions under which a warning will be triggered.

Dwvstamp.vbx is able to automatically scan a project for a list of the custom controls used by the program and a list of all DLLs referenced by the Declare statements in the project.

Detecting component incompatibilities at runtime.

Since an executable stamped by dwvstamp.vbx "knows" exactly what components it requires at runtime, it is an easy matter for it to check if all of those components are available and up to date. This comparison process is also built into the dwvstamp.vbx custom control.

VersionStamper-VB provides a great deal of flexibility in terms of how you handle the scanning and reporting process. Beginning programmers may simply choose to add the standard Vervrfy.frm form or vervrfy2.frm form and associated modules to their application to provide a default verification check at load time or under user control. More advanced programmers may choose to modify the code to create their own custom report. You might also provide the user with a step-by-step procedure on how to fix certain problems, or perhaps a request that they call your customer support line.

It is possible for the software components required by an application to be so old that the program will not run at all. For these cases VersionStamper-VB includes two sample "Rescue" program which are able to compare the component requirements of an executable with the runtime environment without actually loading that executable. These rescue programs include complete source code and can be fully customized to suit your own needs.

Fixing the VB setup kit.

There are many setup and installation programs available. The Visual Basic setup kit is actually quite good for many applications. However, it does contain one major error in the way it handles version comparisons.

VersionStamper-VB includes suggestions not only on how to fix this problem, but on how to add additional improvements. We use these techniques in the installation programs for all of Desaware's products.

Additional Tools and Information

VersionStamper-VB includes a program called VerInfo.exe which is able to report on the version information of any executable that contains a version resource. It is also able to report on the embedded component information added by the dwvstamp.vbx control.

This program also includes full source code. Which brings us to one of the final features of this package, a feature that we feel is one of the most important.

One of our goals at Desaware is to not just provide "canned solutions" for programmers but to also help educate programmers to take advantage of all of the technology that is available to Visual Basic programmers. Following this philosophy, you will find a fair amount of technical material in this manual and a generous amount of Visual Basic source code as well. We will also cover some of the reasons why certain things work the way they do and how they influenced the design of VersionStamper-VB.

But fear not - this product was designed to be used by absolute beginners as well. And for those who just need a quick solution to the distribution problems discussed earlier, feel free to skip forward to the Quick Start section which will walk you through the process of embedding a version resource, embedding component information and adding a runtime conflict report in a few easy steps.

The Common Dialog Toolkit (SpyNotes #2)

What are SpyNotes?

The release of SpyWorks-VB opened an entire new set of capabilities for the Visual Basic programmer. The primary goal of SpyWorks-VB is, in fact, to make it possible to accomplish virtually anything in Visual Basic that is possible to do using more traditional Windows programming techniques. As a result, SpyWorks-VB has been one of the most succesful add-on products for Visual Basic to date.

We have found that there is a real need among Visual Basic programmers for more advanced information than is commonly available - especially when it comes to dealing with the intricacies of Windows itself.

The Common Dialog Toolkit is the second of ongoing series of application paks for use with SpyWorks-VB. These application paks combine text in windows help file format along with plenty of sample code. Most of these paks will be on a single subject, for example: multimedia, or Database Access, and will cost about the same as a good book or newsletter might.

The Common Dialog Toolkit discusses advanced programming topics with an emphasis on using and modifying the Windows Common Dialog box library. It provides far more capability than the limited cmdialog.vbx custom control provided with the professional edition of Visual Basic. Many of the techniques described can apply towards other tasks and third party DLL's as well - so you should find the package worthwhile even if you never use the common dialogs themselves.

And as always, you'll find more than one technique demonstrated that is generally thought to be "impossible".

The following is a brief summary of the techniques that are demonstrated in this package:

- How to call the common dialog API functions directly.
- Finding and manipulating controls within a common dialog.
- Using the CBK.VBX control to implement a common dialog hook.
- Hiding command dialog controls.
- Positioning a common dialog before it is shown.
- Making common dialogs Modal or Modeless.
- Communicating with common dialogs and use of registered messages.
- Hooking the Visual Basic GetMessage loop.
- Embedding Visual Basic controls into common dialogs.
- Creating new controls on common dialogs.
- and much more....

Run the Common Dialog Toolkit Demo

CCF-Cursors

The cursor's primary role in a Windows program is to indicate the position of the mouse on the screen. However, in many applications the cursor plays a key role in indicating the state or operating mode of a program. For example: The hourglass cursor indicates that the user should wait. The I beam cursor is often used to indicate a position in a text string. In paint programs, cursors often indicate the tool in use. A pencil, paint can and aerosol spray cursor can indicate the draw, fill and airbrush operations.

Visual Basic comes with only 12 built in cursors (or "mousepointers" as they are called), and could not easily be extended - until now. CCF-Cursors has been designed to provide this capability and much more.

CCF-Cursors Features:

- The ability to create cursors from scratch, or to convert any icon into a cursor.
- The ability to set the cursor for a form or control including VB 2.0/3.0 graphical controls.
- The ability to set the default cursor for all controls on a form.
- The ability to detect mouse events (MouseUp, MouseDown and MouseMove) for ANY control (including those that do not support mouse events). Also detects DblClick in most controls.
- Detection of menu selection events and characters entered when a menu is active.
- Support for automatic cursor animation.
- The ability to set and retrieve the cursor location, both in screen coordinates and control coordinates and to restrict the cursor to any control or part of a control or form.
- Full compatibility with Desaware's Custom Control Factory makes it easy to save cursors in your Visual Basic form or .exe file.
- A set of windows resource and coordinate transformation functions to simplify cursor operations.
- Icon to cursor conversion program icon2cur.exe and other demonstration programs with Visual Basic source.
- Application oriented documentation with detailed examples including cursor animation, cursor selection based on various criteria, etc.
- Over 50 Sample cursors including directional arrows and the rotating hourglass.

Run the CCF-Cursors Demo Program

The Custom Control Factory

Much of the power of Visual Basic derives from use of controls to build an application's user interface. The Custom Control Factory by Desaware has been developed to help bridge the gap between the standard controls included with Visual Basic, and custom controls which otherwise require advanced programming using the Visual Basic Control Development Kit, Windows System Development Kit and a 'C' compiler.

The Custom Control Factory allows you to create a wide variety of button controls. For example:

- **Animated Buttons** - Define a sequence of images to form an animated flic that will play each time you click on a button. Four different animation cycles are available.
- **Multistate Buttons** - A Mulitstate button can have any number of states, each with its own unique image.
- **2-State Animated Buttons** - A button that supports two states (like a checkbox) except that the transition between the two states can be fully animated.
- **Enhanced Standard Buttons** - Enhanced versions of the standard Visual Basic button, checkbox and option button controls. One can define the appearance of a button in both pressed and unpressed states.
- **Toolbar Buttons** - Create a toolbar similar to that used in Visual Basic and other Windows applications.
- **BONUS!** Now includes MLIST2.VBX - an enhanced listbox control that allows list box entries to contain bitmaps, to be colored individually, and more. Includes all source code!.

In addition, all Custom Control Factory buttons support the following features:

- **Flexible Text Placement** - The button caption can be placed on any side of the image or positioned anywhere on the image.
- **Multi-line Captions** - The button caption can contain multiple lines, and can be left, center or right aligned.
- **Use any Bitmap , Icon, Cursor or Metafile** - Any Windows compatible bitmap, icon, cursor or metafile may be used to define the image frames for a button.
- **Save/Load to File** - Image sequences and CCF button properties can be saved to files to create libraries of custom controls.
- **Automatic 3D Effects** - Build 3D style borders with user defined attributes such as color and width.
- **Synchronization and Owner Draw Capability** - Control every aspect of drawing for each state or frame in the control.
- **DDE Support** - Use Dynamic Data Exchange to allow images to be transferred to and from CCF Controls.
- **Image Compression** - Bitmaps can be compressed when saved to disk and may be compressed in memory. This helps reduce EXE file size.
- **256 Color Support** - with user specified palettes.
- **Runtime Configurable** - Most CCF button properties (including images) can be changed under program control at runtime.
- **No Special Programming** - Custom Control Factory buttons are created interactively at design time. No special programming is required to create or use these controls.
- **Adjustable Animation Speed**
- **Software Trigger** - CCF buttons can be triggered under program control. This allows additional effects, such as continuous animation, to be supported.

... and others (including image scaling, automatic control sizing, click position detect, and so on....

Run the Custom Control Factory Demo

Technical Notes

Desaware's products are designed to be used by professionals. This means that we do our best to make them efficient in terms of resource and memory use. Some of the most important features of these products are often "behind the scenes". The purpose of this paper is to introduce you to some of these hidden features.

SBC.VBX - Efficient Subclassing:

Subclassing a window by its very nature implies that the subclassing code will receive and process every message received by the window being subclassed. The performance impact of subclassing itself is very small - however we were concerned about several issues when developing this control. First, neither we, nor Microsoft guarantees that it is safe to fire a VB event during every Windows message. We know for a fact that there are many operations that are not allowed during certain types of message processing. Second, executing Visual Basic code for each message has higher overhead than processing it in C. In order to handle both situations, SBC.VBX only triggers VB events for those messages that you specify. This significantly reduces the chance of accidental errors, plus it reduces the amount of VB code required for message processing.

SBC.VBX is also able to safely subclass windows in other VB or non-VB applications, and to subclass VB3 graphical controls (that do not have a window handle). Each SBC control can subclass any number of controls, providing an efficient mechanism for subclassing multiple windows.

Subclassing is a simple idea in concept, but can be very tricky in practice - especially in an environment such as Visual Basic where a form or control may be subclassed several times by different 3rd party tools. SpyWorks-VB has been designed with this in mind, and uses a number of advanced techniques to cooperate with and protect itself from other tools and applications. These are detailed in the August 93 issue of Windows Tech Journal in an article titled "Rules of the Road".

dwVStamp.VBX - Automation or Full control + efficiency.

Visual Basic promotes the use of software components - VBX's, DLL's and so on. But while Windows provides good technical support for use of components, it is pretty clear that its designers never considered the real world problems that occur on systems with multiple versions of shared components. We ourselves have spent plenty of time on the phone chasing down bugs that turned out to be old versions of controls that somehow "reappeared" on our customer's systems.

VersionStamper-VB is our way of providing a tool to solve these problems when they occur. We had three major design criteria. First, we wanted it to be easy to use for those who wanted a quick and simple solution to these distribution problems. If you accept the control's defaults, all you need to do is select embedded files, type in your version numbers, and add a module and form to your application that contains our default report generator (assuming you want that capability).

But we also wanted to provide flexibility for serious programmers. As such, all conflict resolution is handled through Visual Basic events, allowing you to take complete control over the way any problems are handled. The package includes plenty of source code (all of the Visual Basic utilities include source code).

Finally, we wanted the control to be efficient. As such, we placed all code used at design time into a separate design time DLL that is not distributed with your application.

CBK.VBX - Different protocols, and a unique guarantee

Callbacks involve passing a function address to a DLL or Windows API function which it will call in order to trigger a VB event. Most callback functions use what is known as the PASCAL calling convention, however a few use the 'C' calling convention. CBK.VBX is designed to handle either type.

There are some callbacks under Windows that take place at a very low level (even at the interrupt level). The documentation for these callbacks warns that only the PostMessage API function may be safely called by the callback function. The low level multimedia functions are an example of functions that use

this type of callback. With SpyWorks-VB version 2.0, CBK.VBX is able to handle these callbacks by posting a message to itself which will later trigger the Visual Basic callback event. You can specify a default return value to for this type of callback (though in practice, few of these callbacks expect a return value).

There can be any number of callbacks types - depending on the parameters that are passed with the callback. In theory, it is possible to create a generic callback that simply points to an area on the stack, however we felt this would be extremely difficult for most people to use. Instead, we provide a VB event for each callback type with this unique guarantee: if you require a callback type that is not already implemented in CBK.VBX, we will add it for you at no charge. The new callback type will be incorporated into all future versions of the control, so you need not worry about incompatibility as long as your customer is using the version of the control that you distribute or a later one (and of course, all of our controls are properly version stamped). We've done this about 5 times so far, each time we've delivered the update within 72 hours (on one occasion on the same day).

SpyParam - Parameter error notification with a subtlety

We've wasted hours of time debugging some API calls. You see, Windows 3.1 often ignores API parameter errors such as invalid window handles (these errors used to cause UAE's under Windows 3.0). The SpyParam program detects these API parameter errors. It has two intriguing capabilities - first, the ability to trigger a Visual Basic runtime error on the exact line that an error occurs. Second, SpyParam is able to ignore parameter errors that are caused within Visual Basic or by custom controls (the kind that you have no control over, and that are often harmless).

SpyMsg - Event spying

We feel that SpyMsg is one of the most flexible message spy programs available. It allows you to specify the method to use for spying (subclassing or windows hooks), and to precisely control the task or window(s) that are being monitored.

But what we're really proud of is the fact that SpyMsg is able to spy on actual Visual Basic events for any VB 2.0 or 3.0 application. This means that you can literally create a history of the events in a program (including the parameters to those events). It spies on events regardless of whether or not there is VB code attached to the event. This is ideal for tracking down many types of bugs, especially those relating to reentry problems.

Our Technical Vision - a note from the president:

It is my belief that Visual Basic represents the future paradigm for programming. Visual Basic itself is an impressive implementation of that paradigm, and I expect that it will continue to be one of the dominant platforms for Visual programming.

One reason for this is that it is possible to use Visual Basic to write Windows applications without compromising on the power of traditional C and SDK based techniques. I've seen a great many VB programmers get frustrated by their inability to do something in VB and blame the environment itself. My primary goal with Desaware (aside from keeping a roof over my head), is to help VB programmers break through these walls both through education, and by providing innovative development tools.

I think we've made a good first step in this direction. My book, "PC Magazine's Visual Basic Programmer's Guide to the Windows API" is being very well received as a "Windows SDK" for the Visual Basic programmer. SpyWorks-VB provides the low level hooks and debugging tools required to do the heretofore "impossible" in VB. VersionStamper-VB is the first product to seriously address the problems of distributing applications made from a large number of components.

In the future Desaware will continue with this approach. You won't be seeing large libraries of controls or high end solutions - there are many vendors providing excellent tools of that type. Instead watch for development tools that enable you to develop your own solutions, for the debugging tools needed to learn enough to create those solutions, and for additional information in the form of application notes such as our new **Common Dialog Toolkit**, or newsletters that will build on the knowledge base of the Visual Basic community at large (and of course, Desaware's customers in particular).

Thank you for your support in the past, and I look forward to working with you in the future.

Ordering Information

We at Desaware strive to provide top quality products at reasonable prices. You can print out the enclosed order form, or call or fax for further information.

Ordering information:

SpyWorks-VB 2.0	_____	**X $129 each**	_____
VersionStamper-VB 1.0	_____	**X $129 each**	_____
SpyNotes #2 - Common Dialog Toolkit	_____	**X $35 each**	_____
CCF-Cursors	_____	**X $35 each**	_____
The Custom Control Factory	_____	**X $48 each**	_____
Visual Basic Programmer's Guide to the Windows API	_____	**X $27.96 each**	_____
How Computer Programming Works	_____	**X $19.96 each**	_____
SpyWorks-VB T-Shirt Circle Size M L XL	_____	**X $12 each**	_____
Package Discount (see below)			_____
CA residents, add sales tax:	**X 8.25%**		_____
Shipping	**$5 U.S. / Canada, $15 International $15 FedX 2Day. $20 UPS Red or FedX Standard.**		_____
	Total:		_____

Ship To:

Name: _____

Company: _____

Address: _____

_____

City/State/Zip _____

Phone _____

Payment: Visa/MC/Amex_____**exp:** _____

Package Discounts: Order two of Desaware's software products at one time and take $10 off your total. Order all five and take $40 off. Discount does not apply to non-software products (books or T-shirt)

For Visa/MC/Amex orders, fax this order form to (408) 371-3530, or mail to Desaware. Checks accepted via mail. Company purchase orders at Net 30 on credit approval.

Shipment on book orders outside of the U.S. and Canada is an extra $10 to $30 depending on country. Contact

Desaware for pricing.

30 day satisfaction guarantee on software products.

Allow 4 weeks for book and T-Shirt deliveries. T-Shirt back shows "You Can't do that in Visual Basic", with *Can't* crossed out and ***Can*** substituted, "with a little help from Desaware" (in white & red on black). Front shows "SpyWorks-VB by Desaware" on pocket area". Shirt availability is limited.

Call Desaware at (408) 377-4770 for further information.