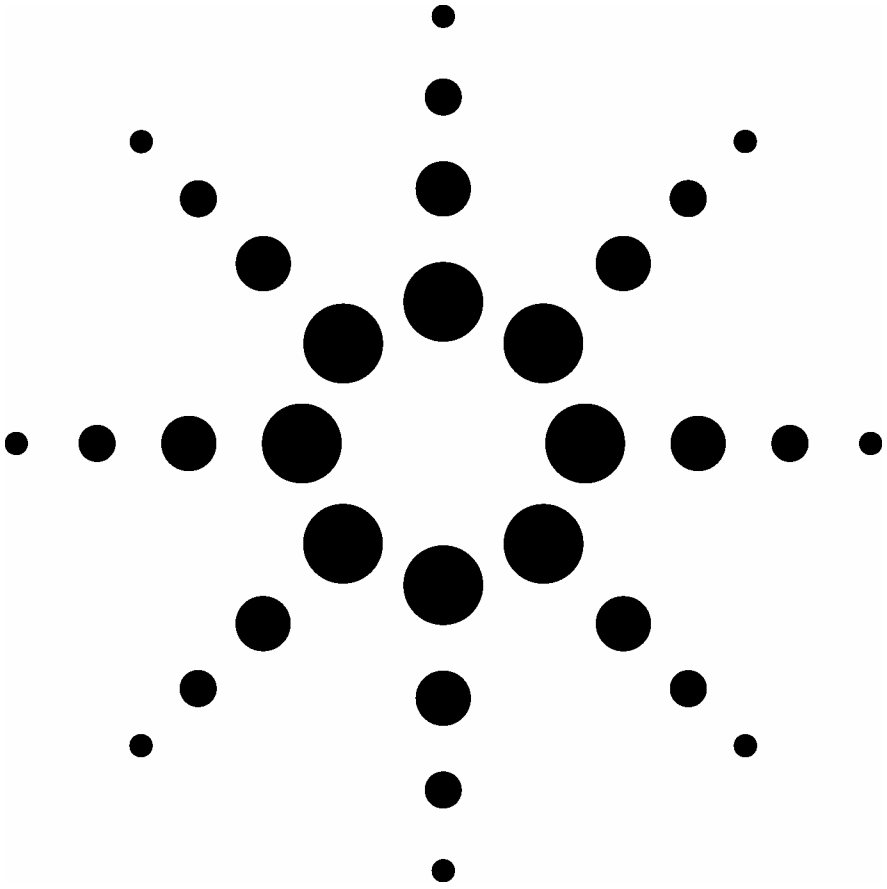
Enhancing Agilent VEE Pro Programs with Microsoft® .NET Controls

White Paper



Agilent Technologies

Introduction

Throughout the past decade, we at Agilent have dedicated significant development to continually make the VEE Pro Software (VEE) environment more "open" to the hardware and software that users choose. Over that time, we have added support for commonly used software tools, such as ActiveX, MATLAB, and Excel. In 2004, with VEE Pro 7.0, we introduced access to Microsoft's .NET Framework, and with Version 7.5, released in June 2005, we added support for .NET Controls. This article explores how VEE Pro users can leverage powerful, PC-standard .NET Controls to greatly enhance VEE Pro programs.

What are the .NET Framework and .NET Controls?

In simple terms, the .NET Framework is a development and execution environment that allows different programming languages to be used together to create Windows applications. It consists of the language-independent Common Language Runtime (CLR), the Framework Class Library (discussed below), and support for standard networking protocols, for various programming languages and libraries, as well as for different Windows platforms. (For more details, visit www.msdn.com.)

.NET Controls are user interface (UI) objects, such as progress bars, grids, and buttons, similar to VEE Pro's UI objects. However, as we will see in this article, .NET Controls' behavior and functionality are different from those of VEE Pro's native controls.

Why might a VEE programmer use .NET Functionality?

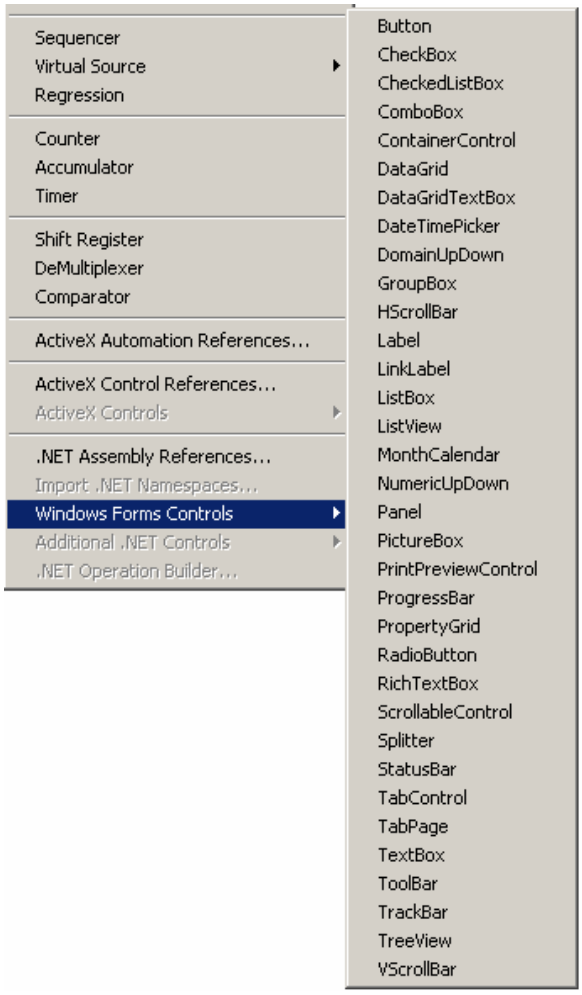
The key feature of the .NET Framework for VEE Pro is the Framework Class Library (FCL). The FCL contains hundreds of new functions (methods) and properties, which enable VEE users to add File and Directory management, simplify String management; quickly view the Operating System environment, manipulate Web pages, etc.

The .NET Framework also provides dozens of user interface control and display objects (.NET Controls), many which are not available as native VEE controls, such as the ComboBox, TreeView and DataGrid, or have more functionality than corresponding native VEE controls. VEE Pro also may incorporate third-party .NET controls into their programs in order to build even richer user interfaces. Another benefit of using .NET Controls is the ability to change properties during design time or during run time. In most cases, native VEE Pro control properties are settable only during design.

.NET Controls also give VEE users functionality similar to that provided by ActiveX controls in VEE: Events. In short, when a VEE program user triggers an event that corresponds to a .NET Control, such as clicking on a .NET button, a function, defined by the VEE user to handle that event, executes. Unlike with traditional VEE functions, this is an event-driven programming model rather than a data-flow programming model. In addition, a .NET control may have many different events defined, thus allowing VEE program to react differently according to different events. For example, imagine you have a TreeView control. You could let your VEE program perform different operations depending on whether a tree node is clicked or when a tree node is expanded.

Using .NET Controls

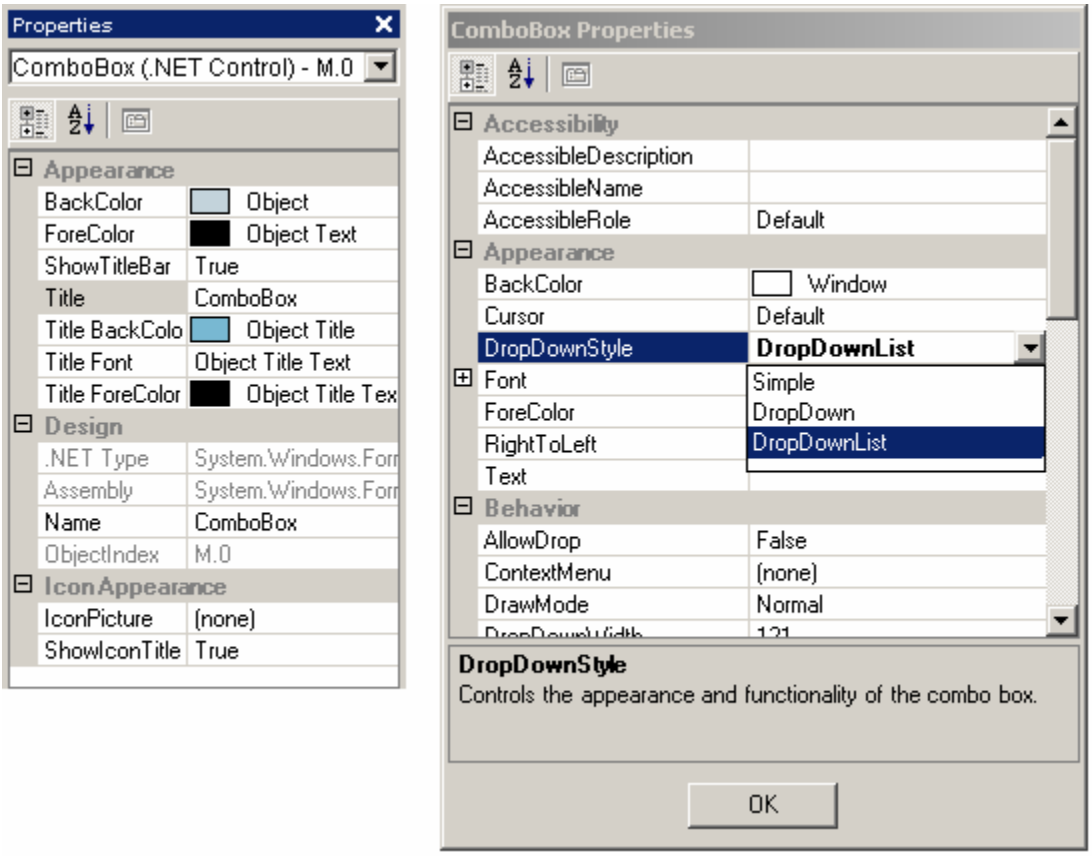
To use a standard control that ships with the .NET Framework, which is automatically installed with VEE Pro 7.5, go to VEE's *Device* > *Windows Forms Controls* menu and select a control from the list. (For a third party control, use the *Device* >*.NET Assembly References* menu to reference the .NET assembly that contains the third-party control. Then access them from *Device* > *Additional .NET Controls*.) You will see the following view of VEE Pro 7.5's Windows Forms Controls menu.



When you place a control in VEE's Detailed View (programming area), it has a default variable name in its title bar. However, it does not have pins that let you connect it to other objects using VEE's wires. This is in contrast to native VEE controls. Here we show a ComboBox as it appears in the Detailed View. Note the lack of terminals.

You can change the Properties of the control at design time. In the case of .NET Controls, VEE's Property Window lets you modify the properties of the .NET Control's host (container). For complete control of the .NET Control's specific attributes, use the "Control Properties…" dialog, accessible through the control's context menu.

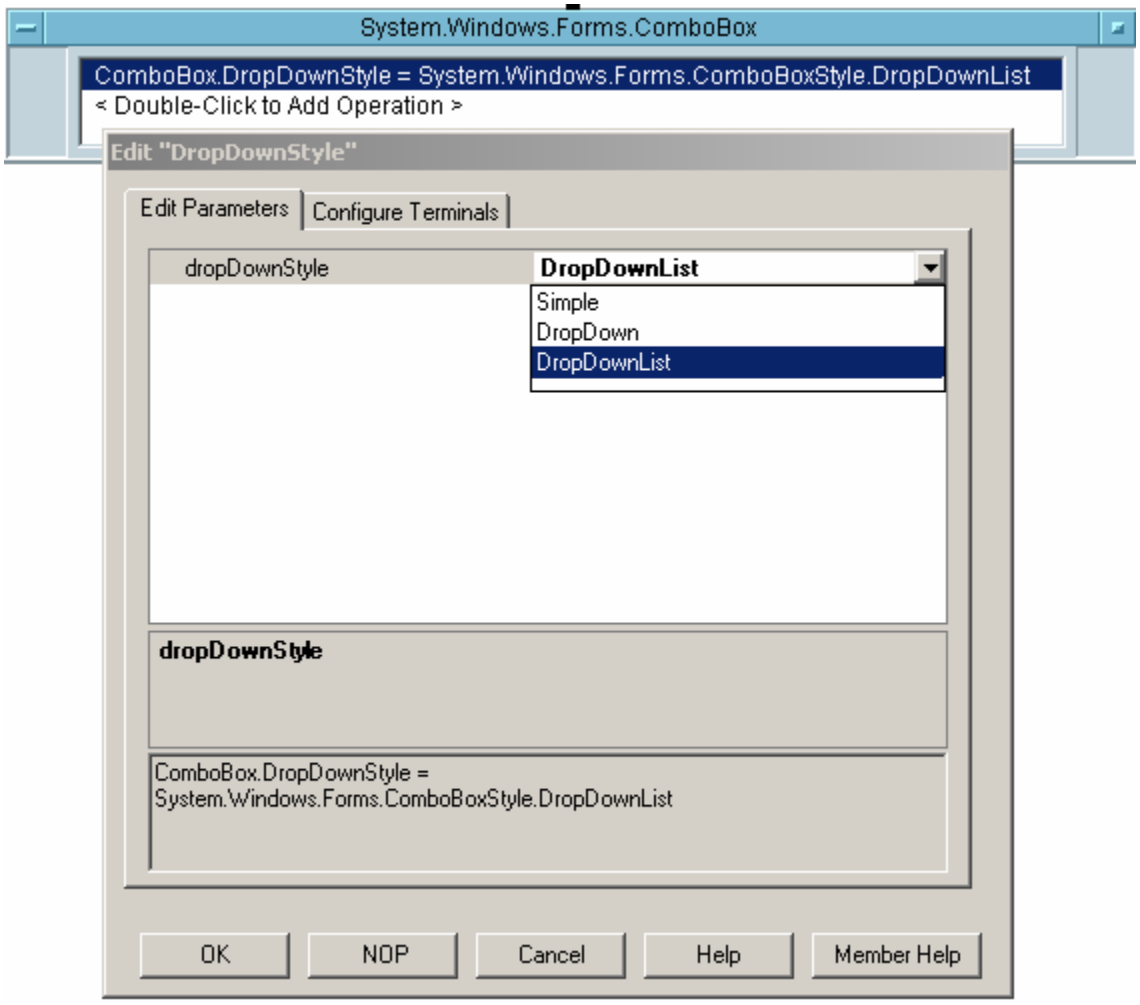
The figures below refer to the properties for a .NET ComboBox, which essentially is a ListBox and user input field. Note some of the differences between the properties accessible through VEE's Properties Window (on the left) and the object's "Control Properties…" dialog on the right. The Properties Window lets the user set properties such as the control's Title and Name. "Control Properties…" allows the user to set attributes of the ComboBox itself, such as the DropDownStyle.



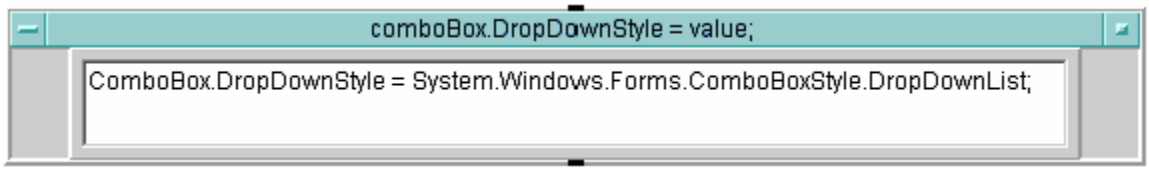
During run time, you must use expressions that reference the control's variable name in order to read and set the control's Properties, and call its associated Methods. .NET Control Properties are values that you can "get" and "set," whereas Methods are similar to functions. The variable name essentially is the declared variable name of the control and is very similar to a variable created using VEE's Declare Variable object, except that the control's variable is already initialized.

VEE Pro 7.5's new .NET Operation Builder is the quickest way to generate expressions that access the control. To do so, select *Generate .NET Operation Builder* from the control's context menu. VEE creates a transaction box, a commonly used VEE object that allows the user to send as a single block multiple commands for a particular task, such as file I/O and instrument I/O. Below we use the .NET Operation Builder to change

the ComboBox's DropDownStyle, which you might do depending on the type of operator who is using the test program.



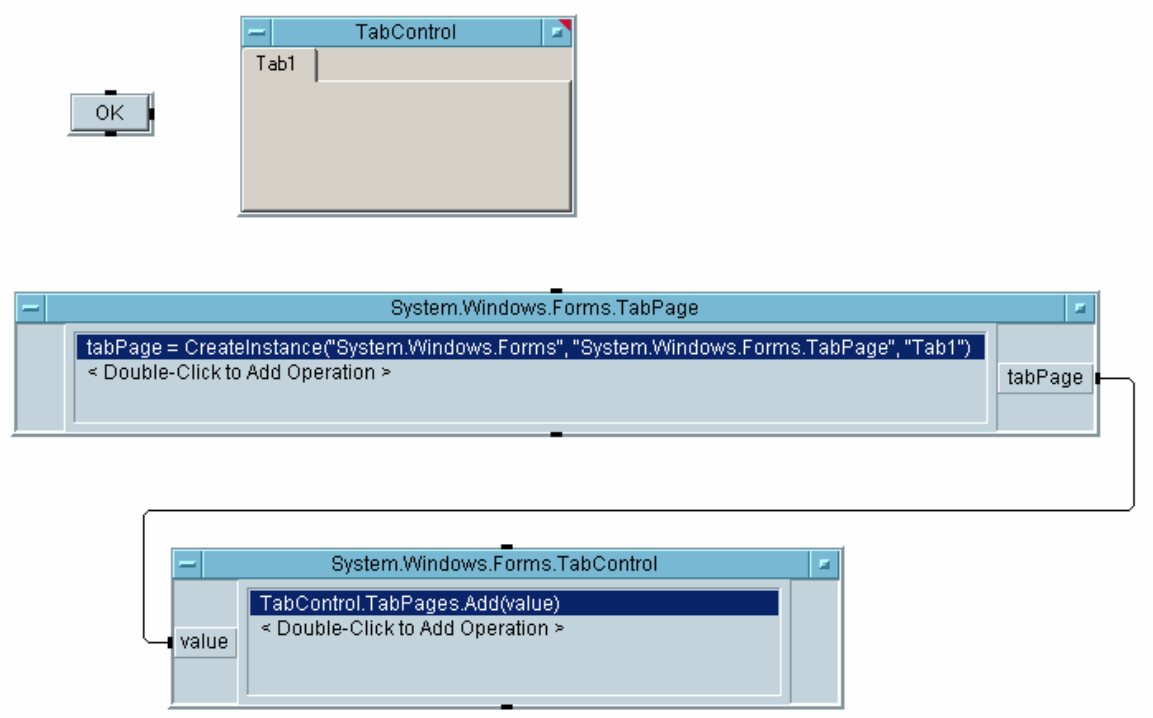
We can achieve the same thing by using the Function & Object Browser (*Device > Function & Object Browser*) to look up and generate the formula template. In that case, we reference the control through its name then type in the right side of the formula, as shown below. Though both approaches achieve the same goal, the latter is prone to syntax errors.



VEE also supports more complicated, composite controls, made of multiple .NET Controls. For instance, VEE can combine a TreeView and a ListView to make a control akin to Windows Explorer. Two other controls are needed to accomplish this: a Panel control, which acts as a parent control, or container, to house the two controls, plus a

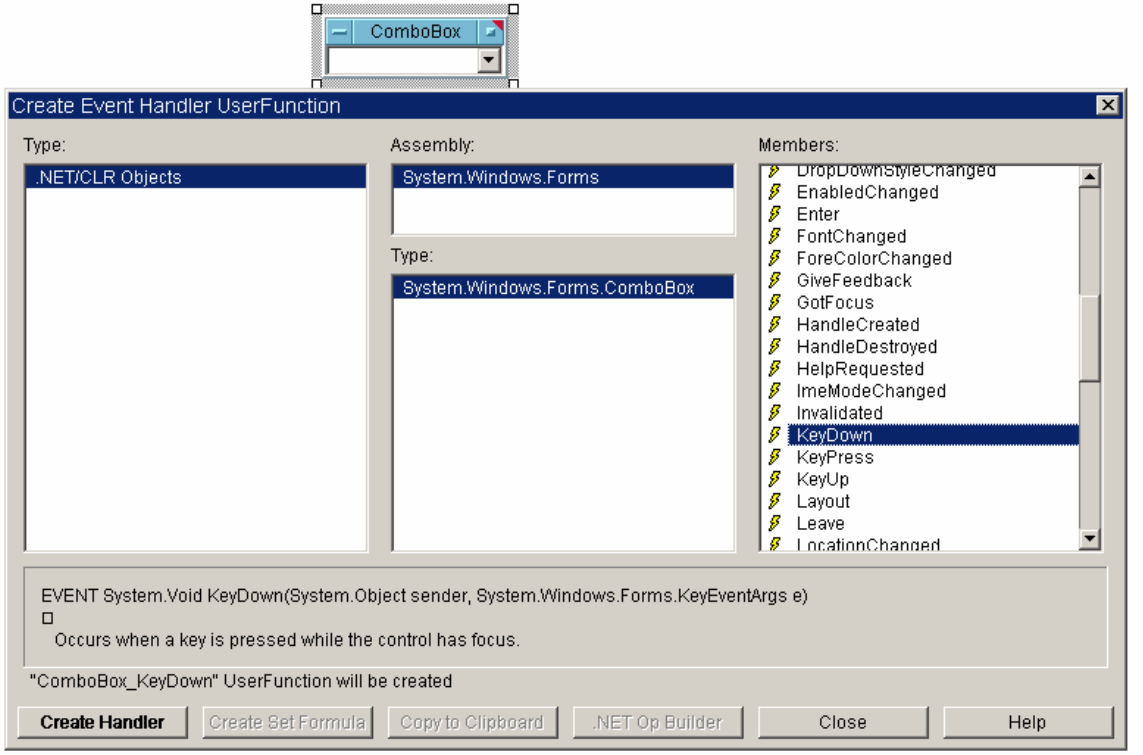
Splitter control, which separates the Tree and List within the Panel control. When building a composite control, it is necessary to place only the parent control directly on the VEE workspace, and add the child controls programmatically. This is due to the fact that some .NET Controls, such as the Splitter, are designed only for use within parent controls, and VEE Pro does not yet have the same visual designer capability as Microsoft Visual Studio® .NET in which you can drag and drop a child control into its parent control. For a detailed illustration of how to do this, refer to VEE Pro's "splitter.vee" sample program, one of many .NET-related sample programs that ship with VEE Pro.

Here we show a very simple example to demonstrate the same concept. We first create a TabPage (child) control then add it to the (parent) TabControl that we placed at the top of the VEE workspace.

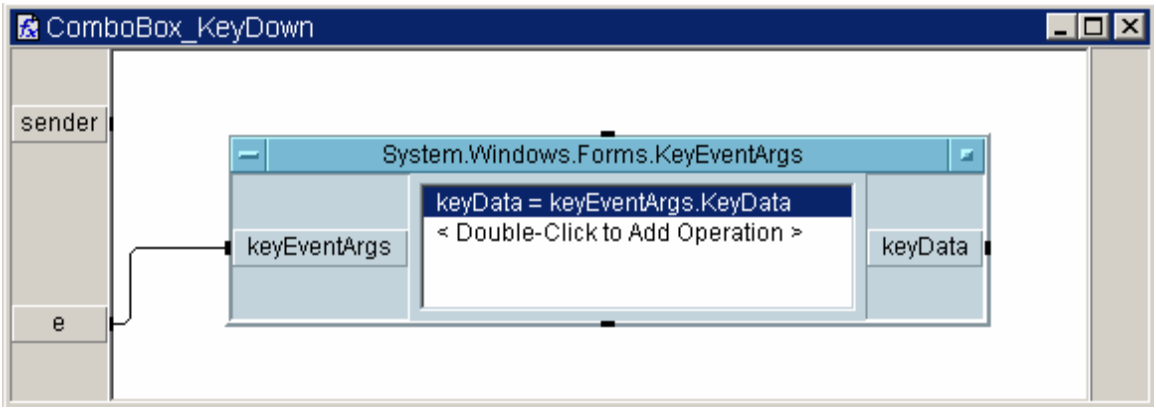


.NET Control Events

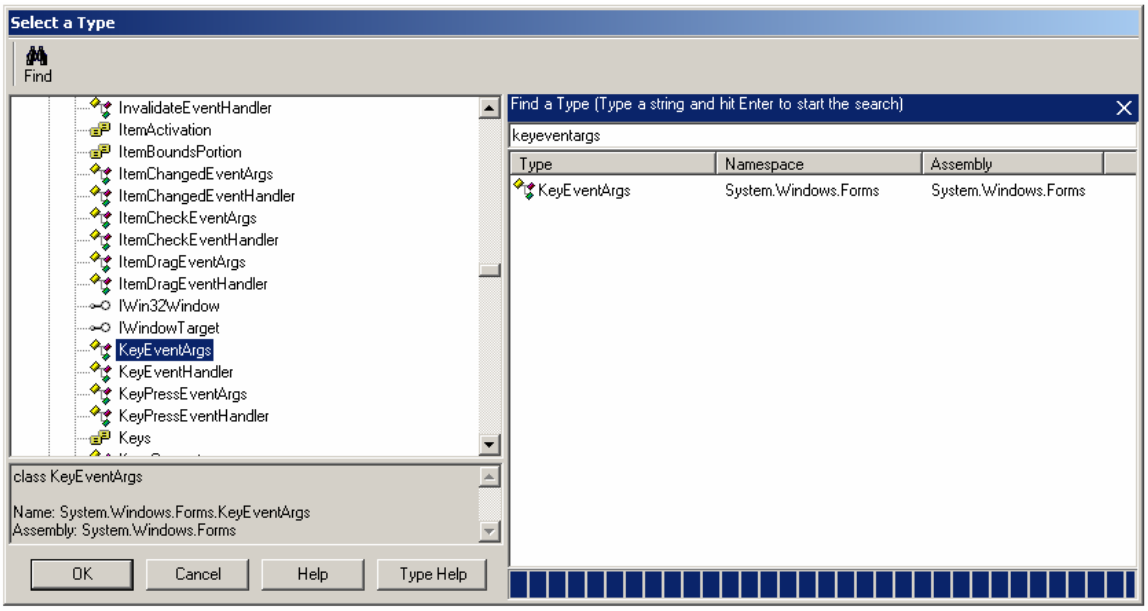
Another key difference between .NET and native VEE controls is a .NET Control's use of events. Visual Basic and Visual C++/C# users should be very familiar with this concept because it is the mechanism that Visual Studio uses for responding to user interface actions. If, for instance, you want your VEE program to respond when the user types a key into a ComboBox, simply define the event and action to be taken by using "Create Event Handler…" on the ComboBox control's context menu.



We have selected the "KeyDown" event. VEE automatically creates an event handler Local UserFunction, with the name format <control name>_<event>, as shown below. Enter your code in the Event Handler window and/or use the .NET Operation Builder to generate code.



Note that the Event Handler input variable “sender” refers to the control that issued the event. You can take advantage of this parameter, for example, to get the name property of the control. The “e” variable provides additional information about the specific event. In this case, this variable is of type KeyEventArgs. Launch *Device >.NET Operation Builder* and use the Find toolbar button to locate and place a KeyEventArgs .NET Operation Builder in your user function as done below. Then select any of the properties and wire “e” to the automatically generated keyEventArgs input pin of the .Net Op Builder. In this case, we have done so in order to retrieve information about the specific key that was pressed to trigger the event.



When a .NET Control’s event is triggered, the control takes over the VEE thread and calls each of its pre-defined event handler functions to handle the event (Typically, your VEE UserFunction is the only event handler, but there may be event handlers in external code or even in the control itself). When the VEE UserFunction is called by the control, both VEE and the .NET control wait for the function to finish. At that time the .NET control and then the VEE Pro program return to their previous states, before the event was triggered. Therefore, programmers should not do lengthy operations inside an event handler function as both VEE and the .NET control wait for the event handler function to finish.

This behavior is in contrast to VEE’s data flow programming approach, which dictates that an object (or UserFunction or UserObject) will not execute until all of the data is present on its input pins and, if there is an input sequence pin dependency, the sequence pin is pinged by the completion of the preceding action.

Conclusion

In this article, we have seen how VEE Pro 7.5's new .NET Control support allows users to add functionality that significantly enhances user interface design and behavior as well as program execution.

To try this new functionality, download a copy of the VEE Pro 7.5 evaluation software or request a CD at www.agilent.com/find/adnevalvee.

Agilent Technologies' Test and Measurement Support, Services, and Assistance
Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise
Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you receive your new Agilent equipment, we can help verify that it works properly and help with initial product operation.

Your Advantage
Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Direct

www.agilent.com/find/agilentdirect
Quickly choose and use your test equipment solutions with confidence.

www.agilent.com

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Phone or Fax
United States:
(tel) 800 829 4444
(fax) 800 829 4433

Canada:
(tel) 877 894 4414
(fax) 800 746 4866

China:
(tel) 800 810 0189
(fax) 800 820 2816

Europe:
(tel) 31 20 547 2111

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Korea:
(tel) (080) 769 0800
(fax) (080) 769 0900

Latin America:
(tel) (305) 269 7500

Taiwan:
(tel) 0800 047 866
(fax) 0800 286 331

Other Asia Pacific Countries:
(tel) (65) 6375 8100
(fax) (65) 6755 0042
Email: tm_ap@agilent.com

Contacts revised: 05/27/05

Printed in USA, March 6, 2006
5989-4912EN

Agilent Technologies