# Understanding ActiveX Controls in Measurement Applications Using VEE 5.0


## Abstract

This white paper will briefly explain the technologies associated with the Microsoft technology called ActiveX Controls and the Component Object Model. It will also explain how these can be used to build better measurement solutions and their applicability to HP VEE 5.0, within the typical engineer's computer environment of Office97 running on Windows95 or NT. ActiveX Controls are compiled software components based on Microsoft's Component Object Model (COM) that encapsulate a set of functionality. ActiveX Controls from a wide variety of vendors can extend the functionality of a VEE program – both in terms of raw functionality AND user interface.


---

Engineers who need to make measurements as part of their job often end up struggling more with software issues than taking measurements. Writing good software is hard to do and time consuming. It is well documented that hardware design is aided by heavy reuse of existing components. This has resulted in unprecedented orders-of-magnitude improvement in performance over the last two decades. If similar standards could be defined to allow various software components from a variety of sources to be glued together to form a solution, then better software could be written in a shorter period of time.

Component (or object) software applies the hardware reuse concept to the creation of new software. Until recently, approaches to software reuse have not been sufficient, either technically or in their widespread market acceptance. Current, traditional technologies present three obstacles to creating a component software market: 1) Distributing objects with their source code; 2) Reusing objects across different languages; and 3) Relinking or recompiling an entire application when one object changes.

To solve these problems and to move towards fulfilling the goal of true software component reuse, Microsoft architected the ***Component Object Model*** (COM). COM is a foundation for interaction among all kinds of software. COM defines a common way to access software services. Without COM there are many different mechanisms are used to access the services provided by libraries, local processes, the operating system, and remote processes.

### *What is ActiveX?*

ActiveX is a marketing name for a set of technologies and services, all based on the Component Object Model (COM). It is also a set of *language-independent* interoperability technologies that enable software components that are written in different languages to work together in networked environments. Some of the benefits of ActiveX components are

- Versionable – there is a specification for determining which version of the object you are dealing with).
- Programmable – via a set of interfaces you can "discover" via one standard way.
- Lightweight and fast – although this is completely determined by the vendor writing the code.
- An open standard (though controlled by Microsoft) which allows them to be written in any language.
- Networked via DCOM – DCOM is simply the networked version of COM.

Today, most COM-based technologies are assigned the label ActiveX. However, ActiveX is really a stripped down version of a COM object to make it more readily usable over the Internet. Currently COM is available for Windows95 & NT, Macintosh, and Solaris operating systems.

### *What are ActiveX CONTROLS?*

ActiveX Controls are compiled software components based on the Component Object Model (COM) that encapsulates a set of business or user interface functions. These were formerly called OLE Controls. Optimizations in the technology resulted in smaller, faster software components and support for key features used to distribute components 'just in time' over networks such as the Internet. ActiveX Controls can be embedded in Web pages for use over the Internet as well as combined to create client/server applications that run over a corporate network. They can be created by a variety of programming languages from Microsoft or from third-party vendors. ActiveX Controls use the file extension .ocx

Taking things a bit further, an **ActiveX Control is a *stand-alone functional software component*** (but NOT a separate application) with a user interface that can plug into any container application. There are thousands of controls from 3rd parties. See some of these sites for a small sampling of some of them:

1. www.componentsource.com
2. www.vbextras.com
3. www.cnet.com/Resources/Swcentral

Applications that can use ActiveX include HP VEE, Microsoft Excel, Microsoft Word, LabVIEW and many others. Programming languages that can incorporate controls include Microsoft Visual Basic, Microsoft VC++, Borland Delphi and many others.

### *Some History*

The COM model was first introduced by providing the foundation technology behind Microsoft's second version of ***Object Linking and Embedding*** (OLE). OLE was originally designed to provide a way to create a document-centric approach to computing. This would allow one document to combine, for instance, an Excel spreadsheet and a Word text

document. However, there was a bigger problem of how interaction would occur between various software packages, including libraries, software applications, system software, and others.

OLE defines technologies for creating compound applications, which are increasingly common. Separate applications that support COM can cooperate to present one compound document to the user. Examples are word processing with graphical capabilities; spreadsheets with charting function. Consider a MS Word document that contains an Excel spreadsheet. When the user modifies the text, Word is in control. Double clicking on the spreadsheet silently starts Excel, allowing the user to manipulate the data in the spreadsheet. However, Word may not need to add Excel graphing functions if an existing graphing function within Word can be used.

Microsoft is also using COM to define extensions to Windows; applying it to many Microsoft applications; and using it to define standard interfaces for many kinds of services.

The predecessor to an ActiveX Control was an ***OLE Control***. (Visual Basic Extensions or ***VBXs*** were the predecessors to OLE Controls and work only in Visual Basic.) OLE Controls, however, had rigid specifications that required a COM object to implement a large set of interfaces. This was often overkill, especially when the OLE control was loaded from a web server.

Thus, ActiveX controls have a more relaxed specification that only requires the control to create its own entries in the system registry when requested to do so. This streamlined specification for ActiveX Controls was specifically created for interaction to the Internet or World Wide Web. The ActiveX Controls technology has been enhanced to allow a control's code and data to be intelligently downloaded as needed from a web server and executed inside a web browser.

OK. Let's get specific now on using ActiveX controls in VEE 5.0.

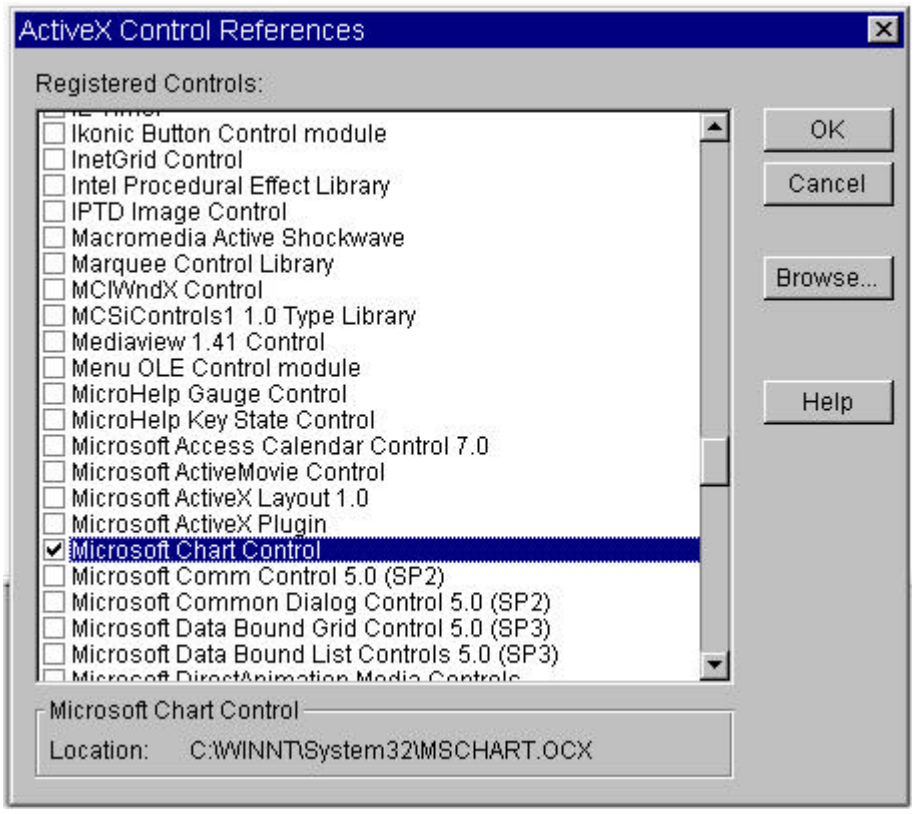
## Using ActiveX Controls in HP VEE 5.0

ActiveX Controls extend HP VEE's functionality by providing functionality that is not currently available in VEE. The functionality is accessed via the Properties, Methods and Events of the ActiveX Control (a.k.a. object). Take, for instance, a hypothetical HP-IB control object. A property of the object might be the address of the instrument (for example, 14) the object was communicating with. A method of the object might be to tell it to take an action like "writeData("*RST"). An event would get called when the object needed to tell the user something happened – one might get fired when the detected an SRQ on the bus.

## To make ActiveX Controls available in HP VEE

Click on *Device→ ActiveX Control References…* The ActiveX Control references dialog box that appears lists all the available objects that are installed on your system.

The list contains only the controls that are registered on your system with existing files. Choose each control that you want to use. This loads the selected controls' ***type library*** which defines the properties, methods and events that the control has. We will use the MSChart object that is shipped with Microsoft Visual Basic for our example.

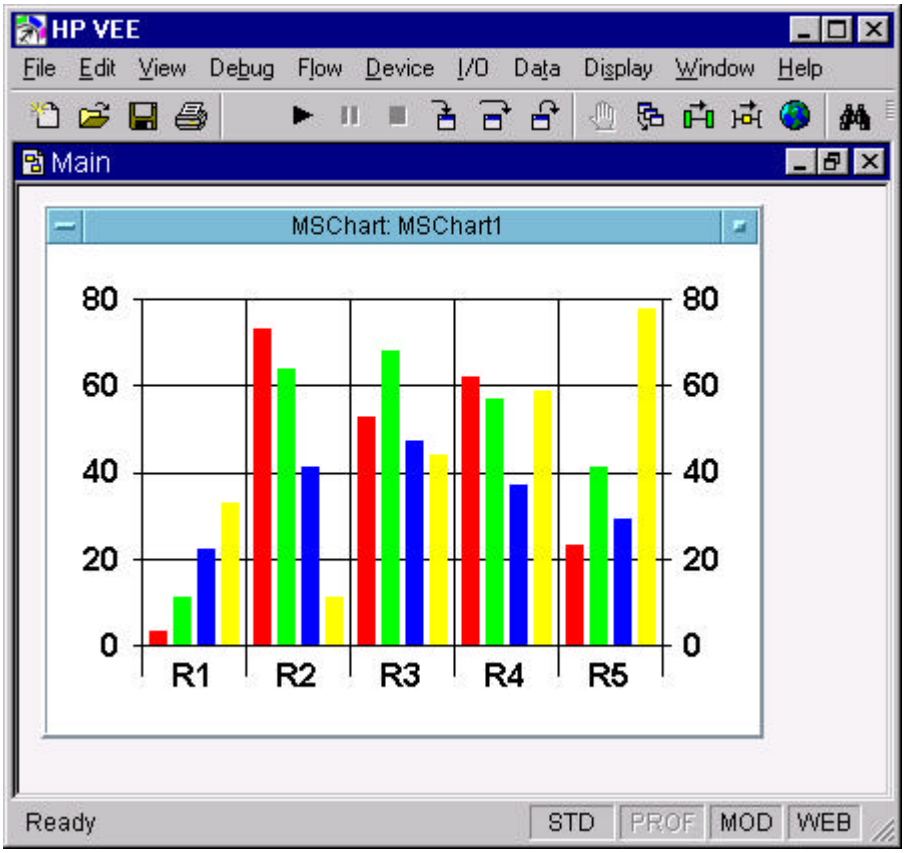



**NOTE:** Remember that the ActiveX control must *also* be installed on the machine it is going to run on. They are not automatically *"swept up"* by VEE when you distribute the VEE code. It is kind of the same thing as having the appropriate instrument drivers installed on your deployed systems.

## To load an Automation control into VEE

Now that you have told VEE that you want to use the Microsoft Chart Control you have to now put a control into VEE. There can be (and often are) more than one type of control contained in a single control library.

Click on *Device*→ *ActiveX Controls* → *MSChart.* Selecting the MSChart object that is placed in a VEE program will look something like this:

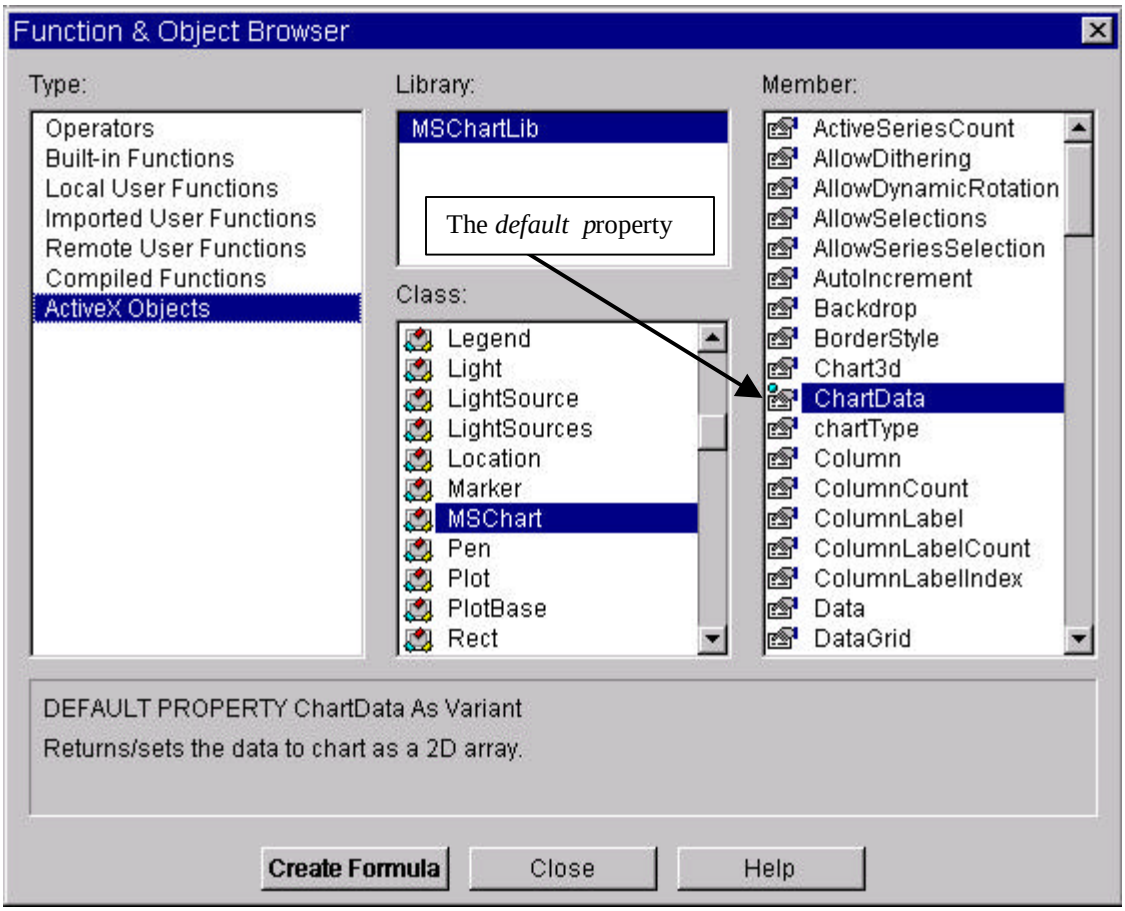



## To manipulate Automation controls

Manipulating Automation controls involves a few basic operations: getting properties, setting properties, calling methods and responding to events. To see the object's properties, methods and events you need to bring up the *Function and Object Browser* in VEE.

## About the ActiveX Object Browser

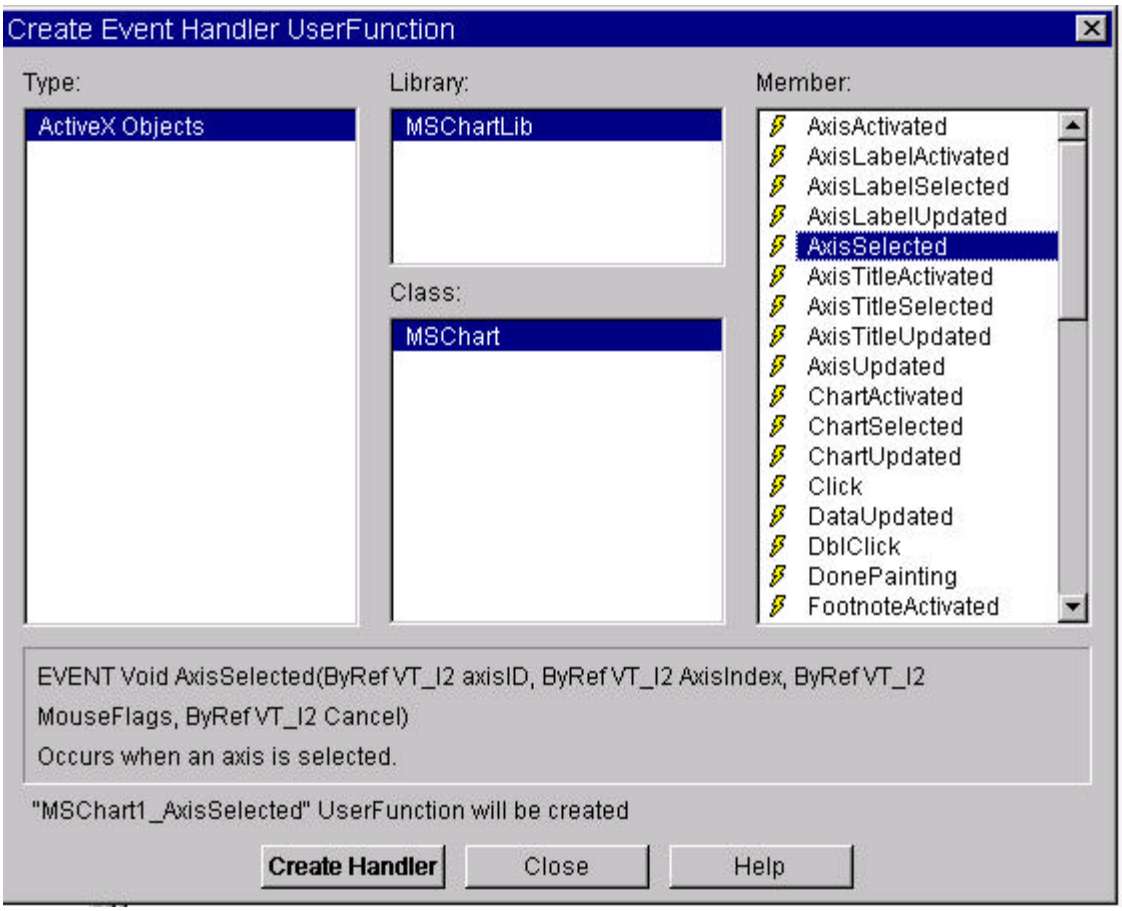
The ActiveX Object Browser is part of the Function & Object Browser that opens when you press $f_x$ on the toolbar (or select *Device* → *Function and Object Browser* from the menu). The browser configuration changes when you select Type: ActiveX Objects. The browser lets you explore what functionality an ActiveX object provides. When the Library name *MSChartLib* is selected, class names appear in the Class area. These classes are either 1) "sub-objects" that can be manipulated in the object – they all have their own set of properties, methods and events or 2) list of enumeration variables (constants). Not all objects are as complex (and big functionality-wise) as the MSChart control.




The information area of the dialog box displays the help string associated with the property, method, event or constant if the Automation object provides this information. Some applications, like Excel, for example, don't provide these help strings. The data type information provided for an ActiveX Object's Properties, Methods, and Events would be in VEE types where there is an exact match between a VEE type and an Automation type. For a property, the browser displays type information about the property, whether it is a read-only or write-only property, and whether it is the *default* property. When you create a Formula object from a property it is configured to do a read of that property as opposed to a write (set) of the property.

For a method call, the browser displays type information about each parameter in the parameter list and the return value. Methods can also be the *default* method, so the browser might indicate this also. When you create a Formula object for a method it is configured to call that method.

For events, the browser displays the same type information as for a method. The ActiveX control calls the event handler UserFunction. So, for events, the Create Formula button is grayed out. HP VEE does not let you to create anything from events. You can only view information about an event.




## To handle events

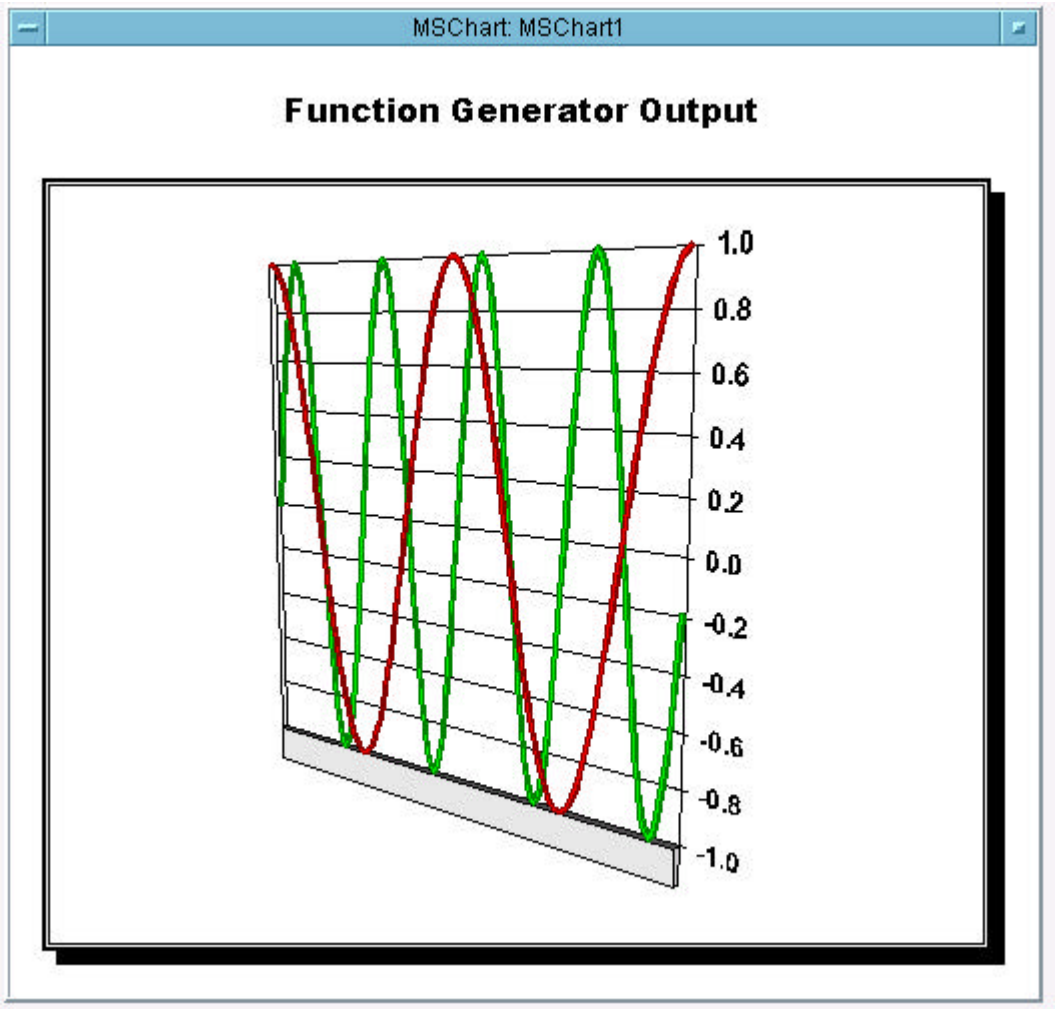
Automation controls can generate events. HP VEE lets you handle events via UserFunctions. You can create an event-handler UserFunction for each event an Automation control can generate.

In the MSChart object's object menu, you can select *Create Event Handler*. Select an event name when the dialog box appears that lists all of the events. Then select Create Handler. A new UserFunction window appears. This new UserFunction is empty except for any required inputs or outputs. You must program it to handle the event appropriately.

Events are tied to the declared variable's name. The UserFunction title combines the controls name with the event name. For instance our chart was created with the name "MSChart1" you might create event-handler UserFunctions with the following names:

MSChart1_AxisSelected,
MSChart1_AxisTitleSelected and so on.

In essence, events are nothing more than *callback* functions. You must program the generated UserFunctions (the callback functions) to handle each event appropriately. Automation controls sometimes expect a return value from HP VEE when they fire an event. If so, you must program the UserFunction to return a value. When the control expects a return value, it waits until HP VEE provides this return value. You should write event-handler




UserFunctions to work quickly, since both HP VEE and the Automation server wait until the event-handler UserFunction returns.

## Using it in VEE

Once you have the object configured the way you want it – in this case as a 3D Line Chart you can send data to it as shown below. I modified the properties of the object by selecting the *Control Properties* menu pick on the chart object and changing the items I wanted to change. HP VEE will write the object information out (called Object Persistence) when you save the program and read it back in when reading the program in.

## Data Type Compatibility

ActiveX Automation supports only certain data types. This section describes what type coercion takes place between HP VEE data types and ActiveX Automation data types. In any event data type coercion between ActiveX Automation data types and HP VEE data types occurs automatically.

## Summary

The ability for HP VEE 5.0 to use ActiveX controls brings its openness to a new high level. As time goes by, more and more controls are created which gives the developer opportunities to develop more and more power programs much easier.