# Understanding ActiveX Automation in Measurement Applications Using HP VEE 5.0

## Abstract

This paper explains ActiveX Automation technology (which supersedes OLE Automation) and Microsoft's underlying COM technology, how these apply to measurement applications, using HP VEE 5.0 with Microsoft's Office97 applications to illustrate. HP VEE and similar software generate large quantities of data from measurements. Microsoft's Access, Word and Excel are software applications used to manipulate data and present it to the end information consumer.

---

## Overview of ActiveX Automation Technology

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 that 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.

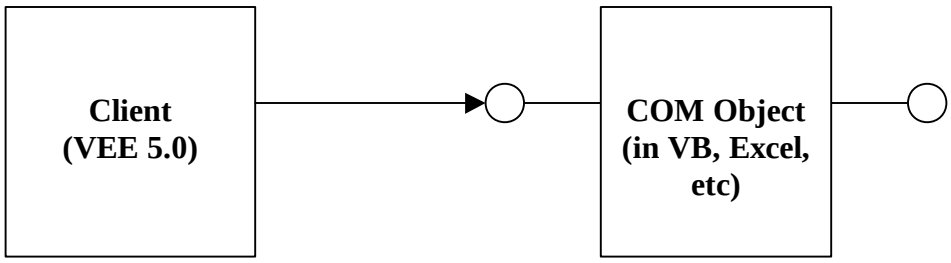
### Microsoft's Component Object Model (COM)

Component (or object) software applies this concept to the creation of new software. Until recently, approaches to software reuse have not been sufficient. Traditional technologies present three obstacles to creating a component software market:

- Distributing objects with their source code;
- Reusing objects across different languages; and
- 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 ***Components 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, different mechanisms are used to access the services provided by libraries, local processes, the operating system, and remote processes.

In the example below, HP VEE 5.0 is invoking the methods in the COM object's interface.




Today, most COM-based technologies are assigned the label ActiveX. However, ActiveX is a stripped down version of a COM object to make it more readily usable over the internet. COM offers the benefit of object orientation, provide consistency, and is language independent. Currently COM is available for Windows95 & NT, Macintosh, and Solaris operating systems.

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 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.

Originally, COM ran on only a single system. ***Distributed COM*** (DCOM) changes this. With DCOM, COM objects can provide their services across machine boundaries. To achieve this, DCOM relies on ***remote procedure calls*** (RPC). With RPC, you can call and execute an object across a network. In other words, DCOM can be used to build secure, distributed, COM-based applications.

## ActiveX Automation

Many desktop software applications have their own "macro" language that allows programmatic control within the application. However, the macro language is often cryptic and is not accessible by other programs. In addition, since there is no standard for macro languages among applications, each application macro language is distinct and not interchangeable. Within the application, the macro language is normally predetermined for the developer.

COM attempts to address these concerns. Now any language that can invoke the methods in the COM objects of an application can gain access to the services of that application. A program that accesses objects can programmatically do anything a user was doing manually. General programmability with COM is called ***ActiveX Automation***.

For example, a Microsoft Excel user can perform calculations on a range of cells by selecting menu picks and manual typing. What was accomplished through the user interface may be replicated through the use of Visual Basic or another tool to write code that does exactly what the user accomplished.

## The Simplified Interface Used by ActiveX Automation

ActiveX Automation is a COM-based technology that provides programmability of applications by other programs. However, most programs do not expose their services through ordinary COM interfaces, because it is too complex. Instead, the Visual Basic design group of Microsoft defined ***dispatch interfaces or dispinterfaces***. Dispinterface methods are easier to invoke from programs using simpler languages like Visual Basic. However, the parameters allowed with dispinterfaces can't be as complex as those using the full COM-based interfaces. One reason for this is Visual Basic doesn't support certain data types.

Dispinterfaces use a standard COM interface called ***IDispatch***. IDispatch is like other COM interfaces except it includes a method call Invoke, and Invoke can be used to invoke all other methods. By using Invoke, a client can invoke any of a group of methods, passing whatever parameters are required. Thus, the Visual Basic interpreter, for instance, can take advantage of IDispatch, by containing code that knows how to navigate through only one interface (IDispatch).

Another difference is dispinterface methods can explicitly be defined to either get or set ***properties***. A method that gets a property is a read-only: it merely reports the current value of the property. A method that sets a property sets the property to the new value supplied.

## Common Procedures for Using ActiveX Automation

The **first** step in using Automation is to declare an object. **Second**, it must be created. **Third**, the object's methods must then be invoked using the syntax of *object.method (parameters)*.

Underneath each application is a simplified form of Visual Basic called ***Visual Basic for Applications (VBA).*** VBA is specific to the application, and is simplified in its capabilities. VBA code may be used in Visual Basic. You can turn on the Macro Recorder under Tools → Macro → Record New Macro in many of the Microsoft applications to see the underlying VBA code. After stopping the Macro, go back into the same menu choices and select Visual Basic Editor. From there you can browse the objects and code structure. Below is an example of instantiating and using Excel's spellchecking object using VBA:

```
Sub SpellCheck ( )
     Dim Obj as Object
     Set Obj = CreateObject ("Excel.Application")
     Word = InputBox ("Enter Word")
     If Obj.CheckSpelling(Word) Then
          MsgBox ("Valid Word")
     Else
          MsgBox ("Word Not Found")
     End If
End Sub
```

This example prompts the user for a word, checks to see if it is in the SpellChecker's dictionary, and returns an appropriate comment depending on whether the word is found or not.

VBA uses the pre-registered "Excel" ***library***. At time of installation, the libraries of COM objects with all of their methods, properties and events are registered by class. The library has access to all classes of COM objects within it. Within the Excel library, the "Application" class was selected. Each COM object is an instance of a specific ***class***. Each class is organized around a specific function. For instance, one class of objects provide spell checking, another might provide thesaurus capabilities.

Excel supports an Application object that has a method called "CheckSpelling." A great source of information for the structure of an applications's classes and underlying methods, properties and events is the application's on-line HELP. For instance, when you can look at Excel's on-line HELP for more information. There are numerous books that give great detail as well as an overview for each application. A third option to use the Macro recorder in Microsoft applications, however not all macro recordings yield insight into what VBA is doing for a particular function.

The following Microsoft applications have ActiveX Automation capabilities.

| **Microsoft Applications Supporting ActiveX Automation** | | |
|---|---|---|
| **Description** | **Object Application** | **Controller Application** |
| **Access 97** | **X** | **X** |
| **Excel 97** | **X** | **X** |
| **Office Binder** | **X** | |
| **Outlook 97** | **X** | |
| **PowerPoint 97** | **X** | **X** |
| **Project 97** | **X** | **X** |
| **Team Manager 97** | **X** | |
| **Word 97** | **X** | **X** |

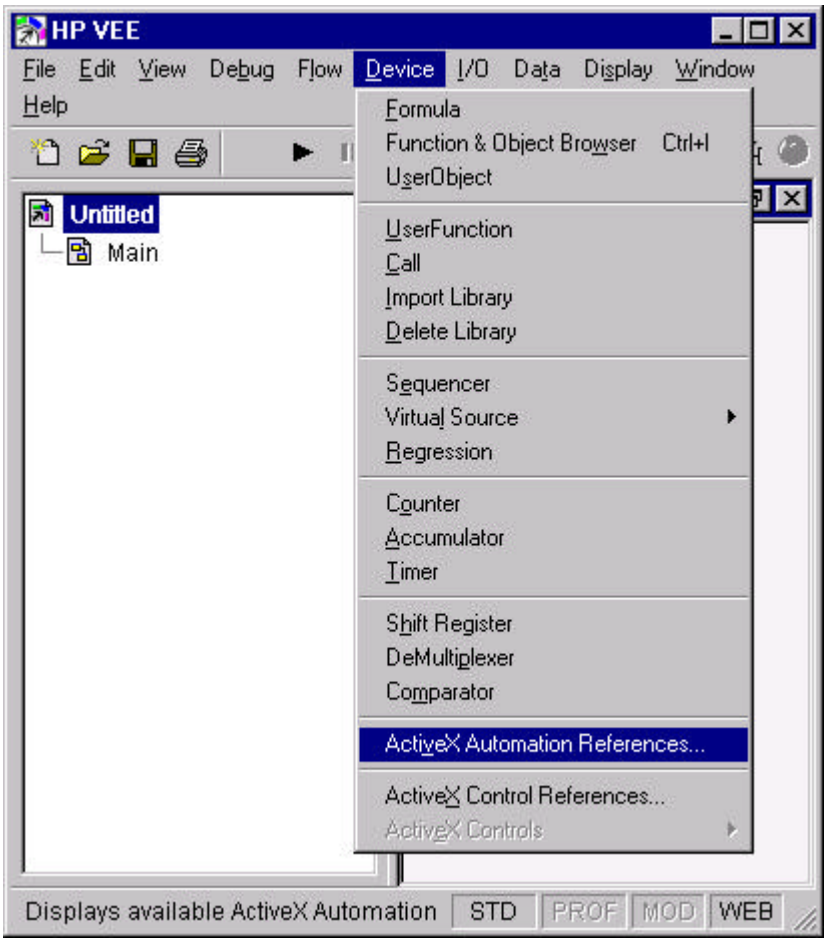
An Automation Object Application has its object model exposed to the outside world that allows it to be controlled by a Controller Application. It receives commands from a Controller

Application. The focus of this white paper is viewing Excel, for instance, as mainly an Automation Object Application.

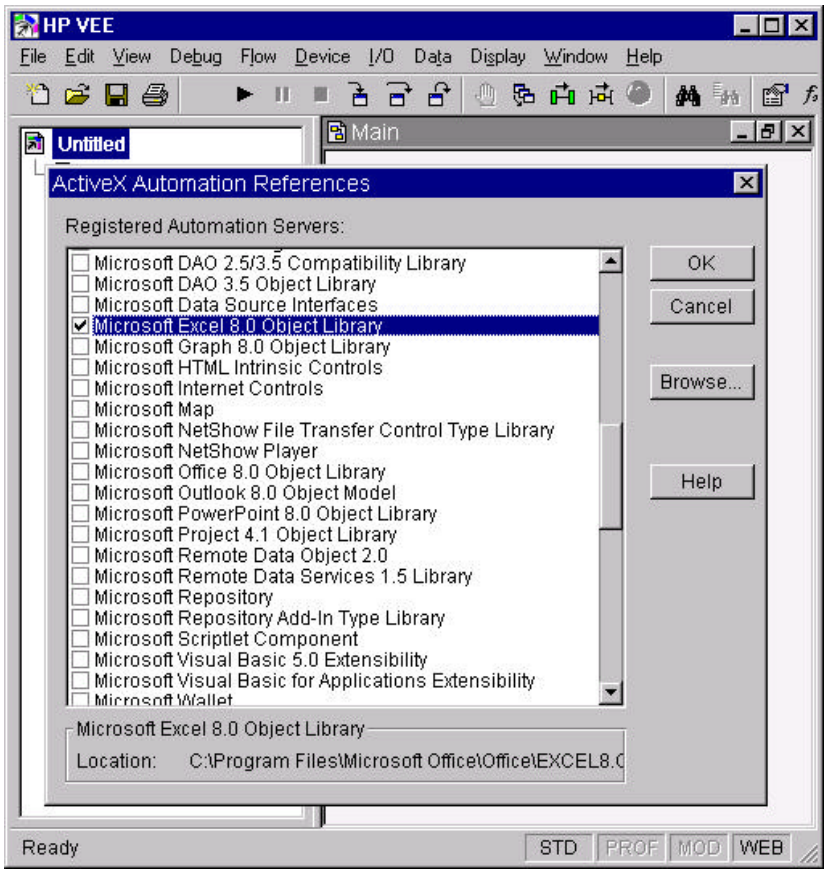
An Automation Controller (or sometimes called Container) Application controls Object Applications. It sends commands to an Object Application. The focus of this white paper is viewing HP VEE 5.0 as the main Controller Application

## Using ActiveX Automation in HP VEE 5.0

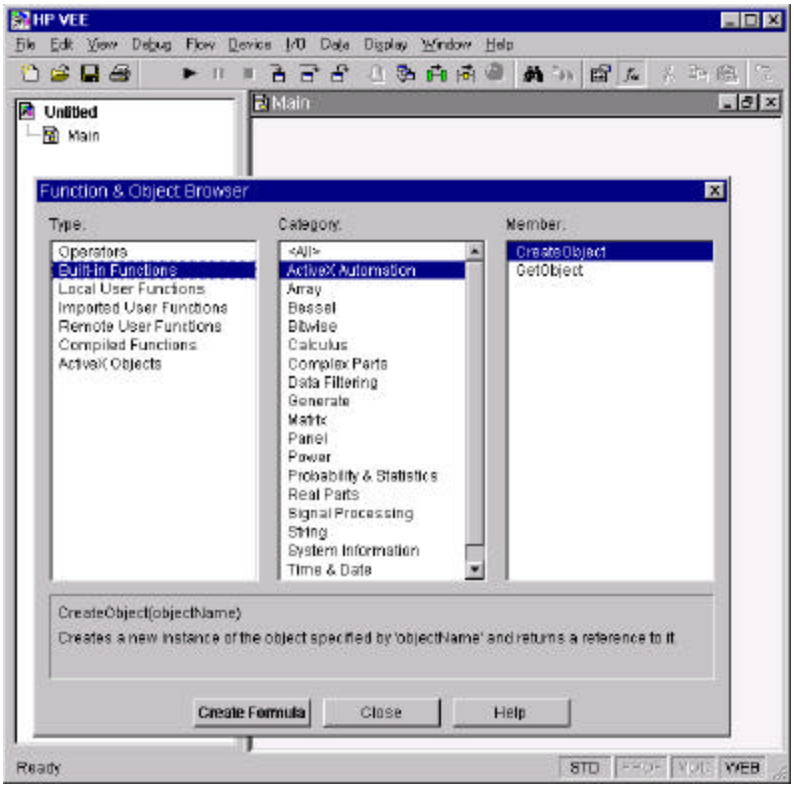
The first step in using ActiveX Automation is to reference the objects you want HP VEE to use, by selecting ActiveX Automation References in the menu as shown:



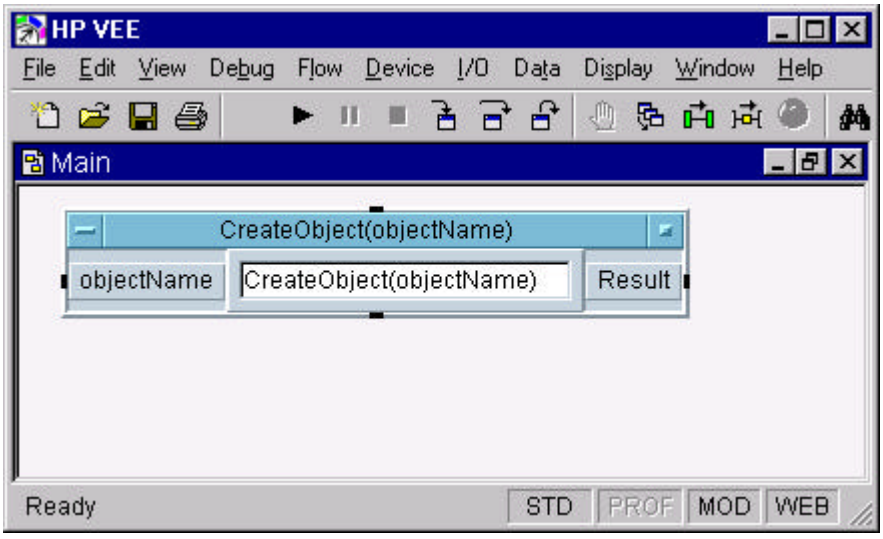

After referencing the objects, the next step is to select the appropriate library of objects. These were registered at time of installation on your system.



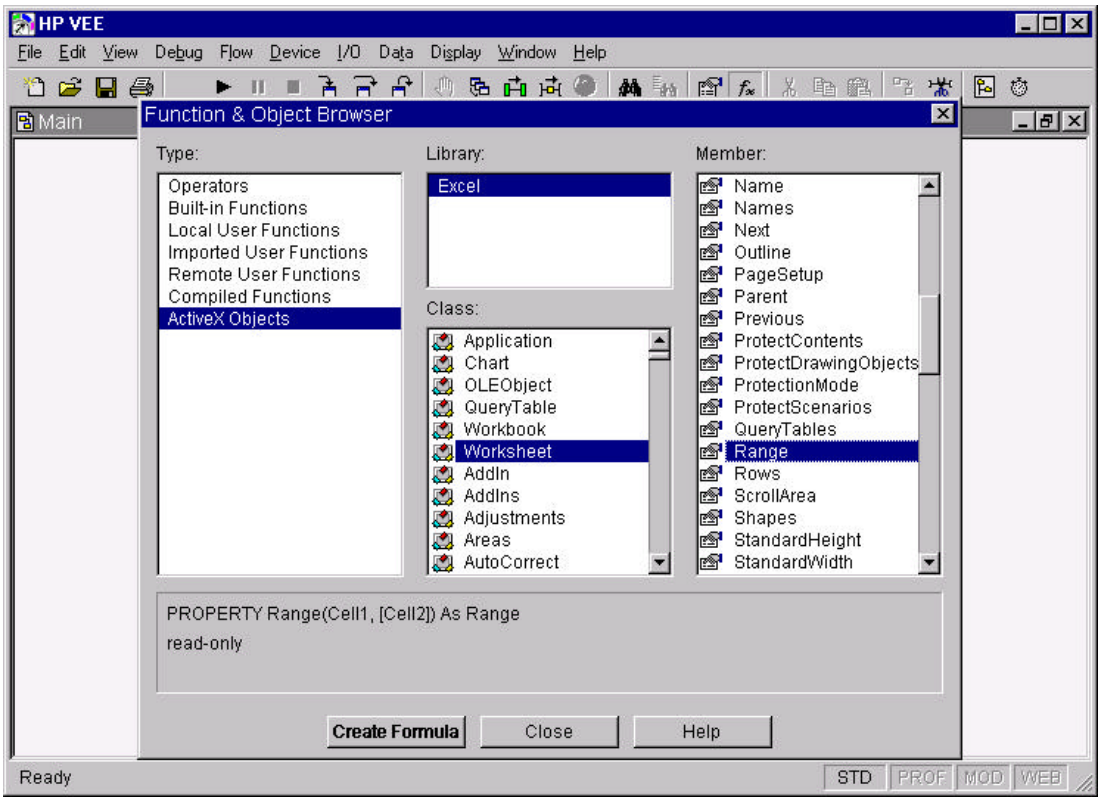

The next step, just like in the VBA program, is to create an object:



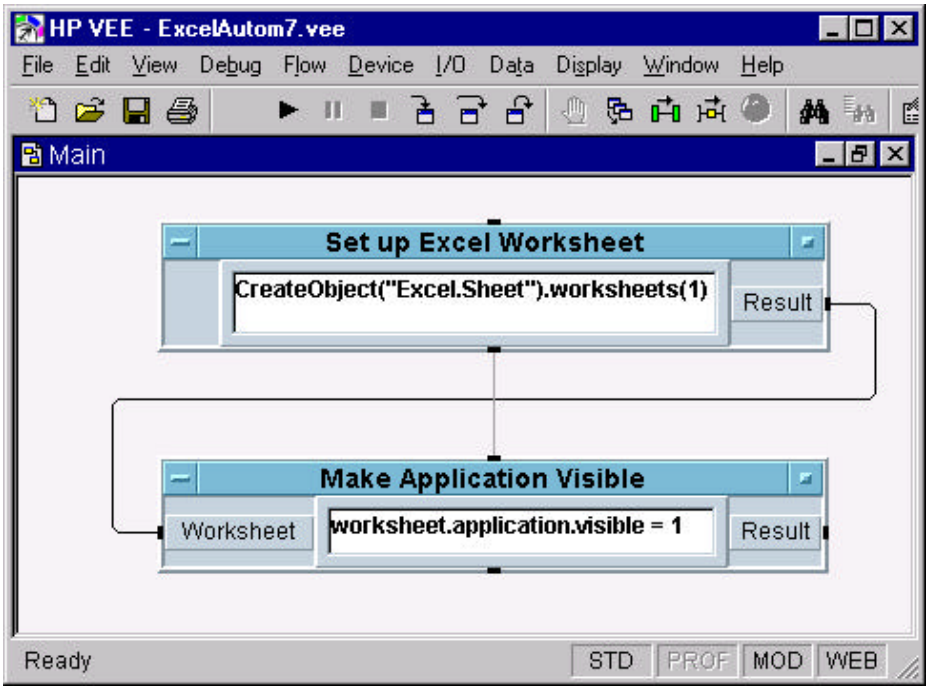
Once chosen, the object will be placed in your work environment. Then you modify the text to state the exact object definition (here it is left in a generic form).



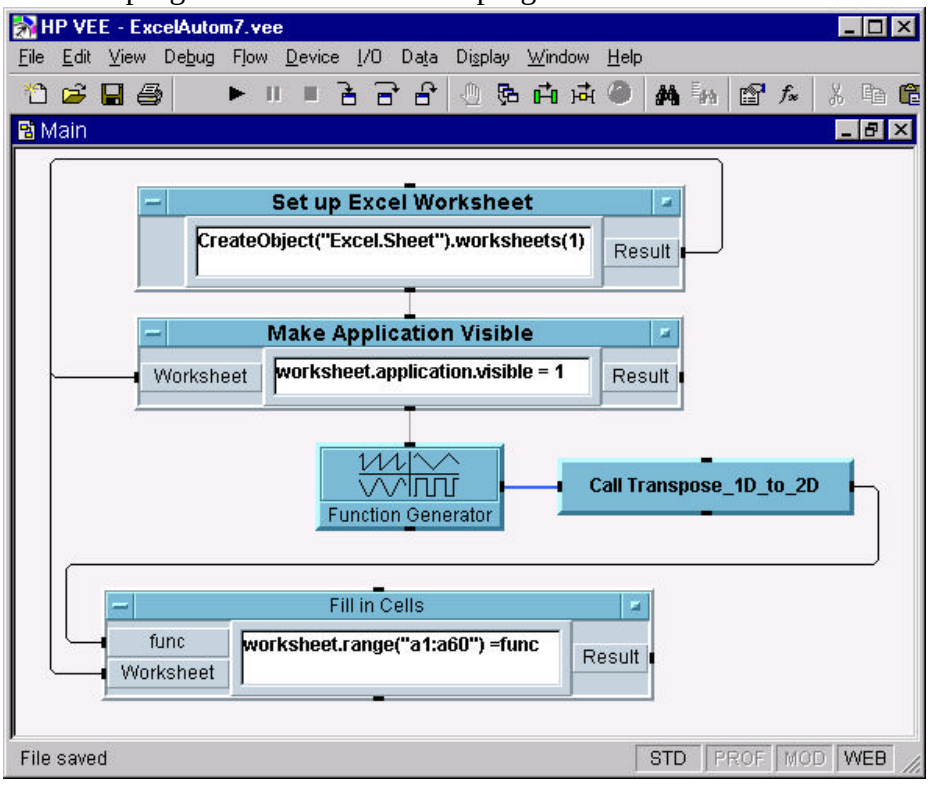

In the example below, you next select the particular property you want to set. After the Excel object is created, the Excel program and worksheet is made visible. Note the HELP button: this directly connects to VBA's help file on this particular property.



By clicking on the Range property shown above, it adds a second object to the HP VEE program. Following the syntax guide in the VBA HELP, it is modified to make both the worksheet and the application visible. Now this program in HP VEE will initialize an Excel program, and display Excel as well as the worksheet. Note the wire that passes the "worksheet" variable, so when you want to make the worksheet visible, the program knows what you're referring to.




Now the entire program can be seen. This program starts Excel from HP VEE, creates a virtual



source cosine wave, transforms to 2 dimensions (and from row to column) so Excel can recognize the data, and dumps the data into the first 60 rows.

The functions can also be made aware of the "worksheet" variable by using the Data → Variable → Declare Variable menu selection. This is also a requirement if you want to use Events, but is beyond the scope of this paper

## Summary

ActiveX Automation, generally speaking, provides programmability of ActiveX-compliant applications. This allows engineers to use the correct software tools for each job by joining them together to develop a complete measurement solution. Engineers may now choose the correct part of each software package for the best solution possible.