VEE 5.0 Introduction Notes (may 98)

last update 03 sep 98 / greg goebel / public domain / vwv_5_0

* This document provides introduction notes on the VEE 5.0 release in the spring of 1998.

* Contents:

[1] VEE 5.0 OVERVIEW
[2] VEE & ACTIVEX CONTROLS
[3] VEE & ACTIVEX AUTOMATION
[4] VEE & THE WORLD-WIDE WEB
[5] OTHER CHANGES
[6] BUGS FIXED IN VEE 5.0

[1] VEE 5.0 OVERVIEW

* The VEE 5.0 release contains a number of enhancements. The most important of them are the following three:

- VEE now supports the ability to integrate custom controls -- "ActiveX controls" -- into a VEE program.
- While VEE 4.0 could be controlled from other Windows applications using the VEE ActiveX control, it could not control other Windows ActiveX applications. VEE 5.0 adds this capability (which is now designated by MicroSoft as "ActiveX Automation").
- VEE now supports a number of mechanisms to allow the results of VEE programs to be accessed over the World Wide Web.

To support the new ActiveX features, VEE now has a third mode of operation. The modes are now:

- Standard
- VEE4
- VEE3

Other enhancments include a function to save panel images into graphics files; improved clipboard support; revised graphics display objects; updated sliders, knobs, and indicators; a multiline formula box; improved memory management; and a number of other small improvements.

The ActiveX features are not supported on HP-UX, as the operating system does not support them. The Web server capability is supported on HP-UX. As with all major revisions of VEE, programs created in VEE 5.0 cannot be loaded on earlier versions of VEE.

* VEE 5.0 is on the 1 May 1998 CPL and should be shipping presently. Bits will be available for HP support people as soon as they are validated. There are a few important changes in pricing from VEE 4.X:

- The price has been raised incrementally for new purchases.

- The upgrade price has been significantly lowered for upgrades, though there is a separate (and more expensive) upgrade product for those moving up from VEE 3.X than for those moving up from VEE 4.X.

  The catch for this is that VEE 5.0 has stronger serial-number protection than earlier versions of VEE ... you *must* already have a valid serial number to upgrade.

- There is a ten-seat site license product. Earlier versions of VEE had a 50-seat site license, but this was not a popular product and feedback indicated that we needed a smaller site license.

Ordering information is as follows:

```
VEE 5.0 for Windows95/NT:          E2120F (option AA8 for floppies)
   Upgrade from VEE 4.X:              option UP1
   Upgrade from VEE before 4.X:       option UP2

VEE 5.0 for HP-UX Series 700:      E2111F
   Upgrade from VEE 4.X:              option UP1
   Upgrade from VEE before 4.X:       option UP2

VEE 5.0 Automation Kit:            82345F (includes 82341C HPIB card)

VEE 5.0 10-User Site License:      E2119F (option AGE for upgrade)
VEE 5.0 50-User Site License:      E2117F (option AGE for upgrade)

VEE 5.0 Educational License:       H2327F option WNT
   Add HPIB Card:                  H2327F option PCN
   Lab Starter Kit:                H2326A
```

Please consult HP for local prices.

[2] VEE & ACTIVEX CONTROLS

* ActiveX Controls are essentially third-party "widgets" that can be integrated into any package that supports the protocol. Such widgets might include specialized graphics objects (such as 3D charts or waterfall diagrams), multimedia tools for video or audio presentation, statistical analysis packages, interface libraries for data-acquisition cards, and so on. Thousands are available as shareware or commercially.

ActiveX Controls are conceptually very similar to the Visual BASIC Extension (VBX) controls that were available for Windows 3, but are 32-bit implementations, not 16-bit implementations. Unlike VBX controls, they can be used with many applications (as long as they are ActiveX compliant), not just Visual BASIC.

As with ActiveX Automation, a VEE user can select among ActiveX Controls, and having selected one can use formulas to access the control. While the controls are part of the VEE program, they are not wired into the program as such, since they don't obey VEE control flow rules; but they can be easily controlled through formula boxes.

Note that if a user creates a VEE program using ActiveX Controls, the libraries for the controls have

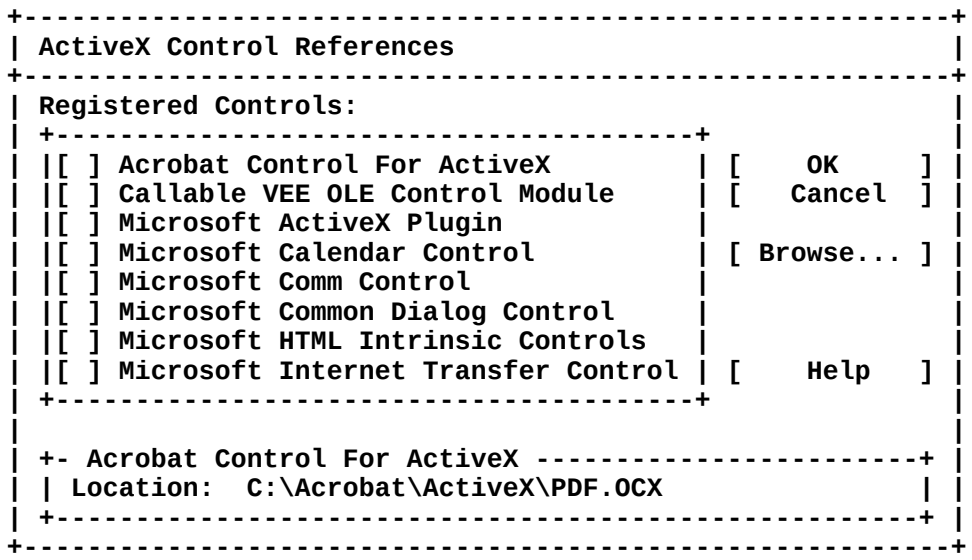
to be shipped along with the program, under the licensing arrangements defined by the vendor of the controls.

* For an example of how to use ActiveX Controls with VEE, I went out on the Web to the CNET (www.cnet.com) to find some useful ActiveX Controls that I could use for a demo.

I found an interesting shareware ActiveX control named Sysinfo, by a Jin Hui of China. This provided some useful status ("properties") about a PC like processor type, memory size and so on, and some useful actions ("methods") such as programmably rebooting the PC.

I unpacked the archive containing the Sysinfo control and got the ActiveX Control for it, which was named "sysinfo.ocx", where ".ocx" was the extension designating an ActiveX control. I then ran VEE 5.0 so I could play with this control.

* The first thing to do in VEE was to "reference" the ActiveX control -- in essence, make VEE aware of it. I went to the "Device" menu and selected "ActiveX Control References". This gave me a dialogue box that listed the available ActiveX Controls:



There was a long set of ActiveX controls available ... apparently the list was derived from Windows Registry entries established when the applications that incorporated the controls (like Visual BASIC) were installed. A sample inspection of these controls showed some of them to be very specialized and cryptic, and of no interest to a VEE user.

In any case, I was after the Sysinfo control. I found a Microsoft Sysinfo control in the list, but this was different from and less useful than Jin Hui's Sysinfo control, and so I had to click on "Browse" to reference that item (as it had no installation program, obviously it was not going to be identified in the Windows Registry). A file browser popped up, and I used it to cruise through the file system to find and select "sysinfo.ocx".

The list now included "System Information Control". I set the flag to make sure it was referenced, and then clicked on OK to close the dialog box.

* Next, I had to create an "instance" of the control in my program to play with. I went to "Device" ->

"ActiveX Controls" and selected "Sysinfo". I got a VEE object labeled:

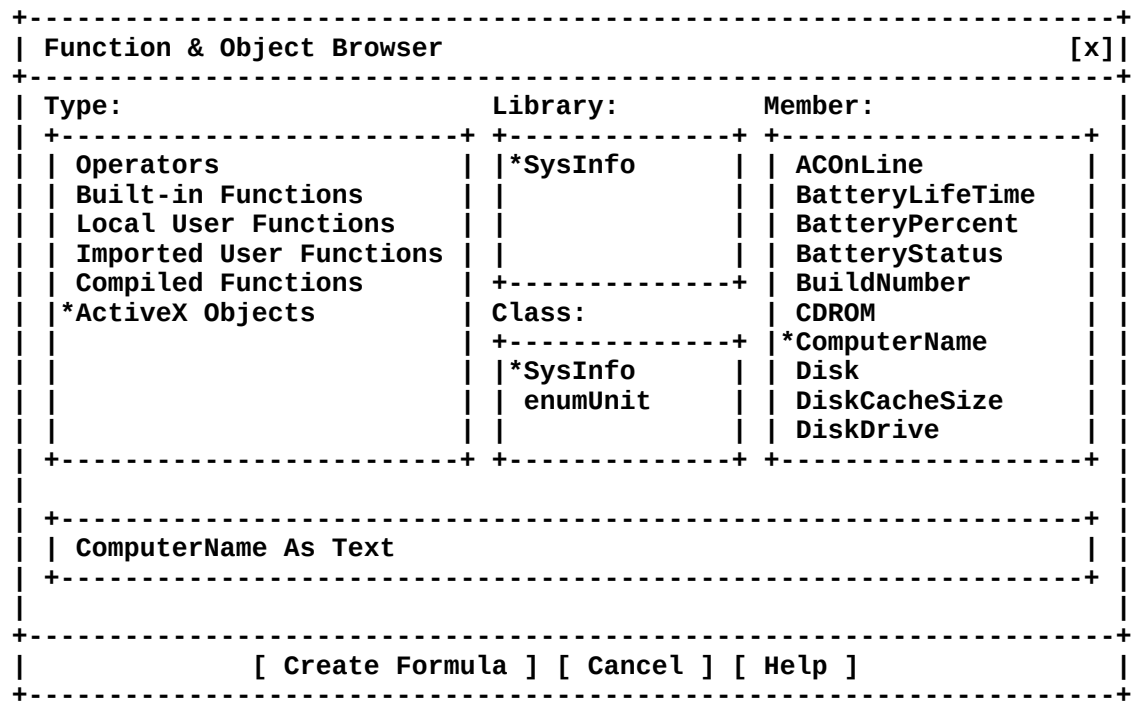
```
Sysinfo:  Sysinfo1
```

Since this particular control didn't have a visual interface worth thinking about, I turned it into its iconized form and left it that way. Essentially the control simply declared the type, "Sysinfo", and the name of this "instance" of the control in the program, "Sysinfo1", that I would need to make use of it.

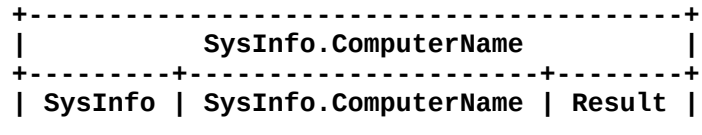
I could change this name to whatever I wanted, and if I created duplicate instances of the control they would be given different names such as "Sysinfo2", "Sysinfo3", and so on, that I could change as well. The names were simply "handles" to the particular instance of the control.

* Once I had an instance of the control in the program I could then start to make use of it by adding formula boxes containing the appropriate properties and methods.

I clicked on the "fx" button on the VEE Toolbar and got the "Function & Object Browser" dialog (which was named "Select Function" in its VEE 4.0 incarnation). I selected "ActiveX Objects" from "Type", then selected "Sysinfo" from "Library", and got a list of the Sysinfo properties and methods:



The list of properties and methods is at the right of the panel. Selecting one gives a little documentation on the property at the bottom of the display ... not very much in this case. Anyway, selecting the "ComputerName" property (which gives the network name of the computer) and clicking on "Create Formula" gave me the formula box:



```
+---------+----------------------+--------+
```

I deleted the input pin and type the name of the object instance into the formula field:

```
+--------------------------------+
|      SysInfo.ComputerName      |
+----------------------+---------+
| SysInfo1.ComputerName | Result |
+----------------------+---------+
```

Either way, running it gave my computer name, "HPLVLGVG". I then wrote a program that used the following properties:

```
SysInfo1.ComputerName               (HPLVLGVG)
SysInfo1.ProcessorType              (Pentium)
SysInfo1.UserName                   (gvg)
SysInfo1.PhysicalMemory             (32899072)
```

-- and then added a method, controlled by a Message Box, to reboot the computer:

```
SysInfo1.ShutdownSystem()
```

* The SysInfo control had properties and methods associated with it, but it had no need to interrupt the program using it, since a program user couldn't punch buttons or tweak widgets on it ... there weren't any.

That's not always the case, and many ActiveX Controls have "events" associated with them, where a mouse click (or whatever) interrupts the program. VEE has a mechanism for handling such events.

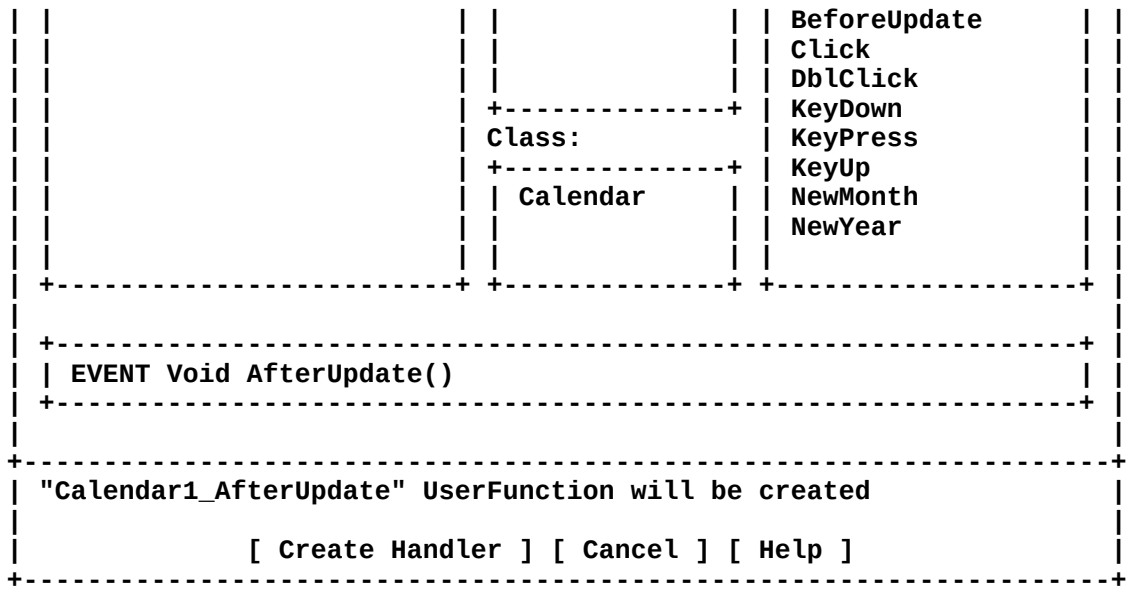
I was able to find a different control, the Microsoft Calendar control, that had events and would be useful for a demo. I referenced the Microsoft Calendar control, then got an instance of it, named:

```
Calendar.Calendar1
```

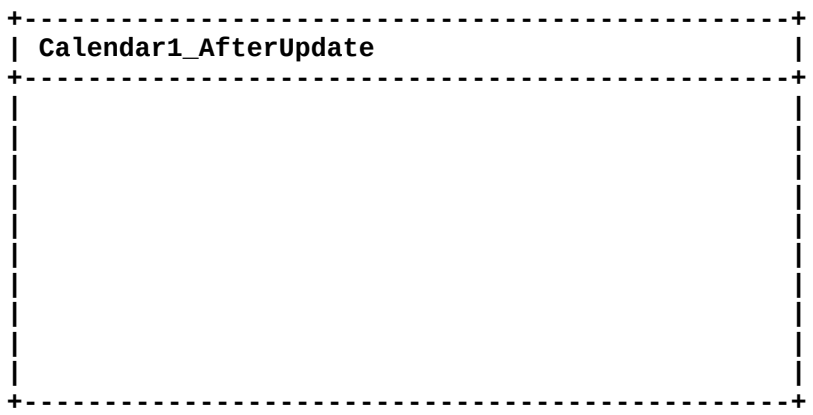
This provided a widget that looked like a little calendar. Note in passing that this object has properties of its own that are independent of VEE, and they can be accessed and controlled through the "Control Properties" entry in the object menu.

In any case, this object -- *not* the Function & Object browser -- allowed me to set up an event handler. I went to the object menu and selected the menu entry "CreateEventHandler", and got the dialog box:

```
+-------------------------------------------------------------------+
| Create Event Handler UserFunction                            [x]|
+-------------------------------------------------------------------+
| Type:                      Library:         Member:               |
| +------------------------+ +--------------+ +------------------+ |
| | ActiveX Objects        | | MSACAL       | | AfterUpdate      | |
```

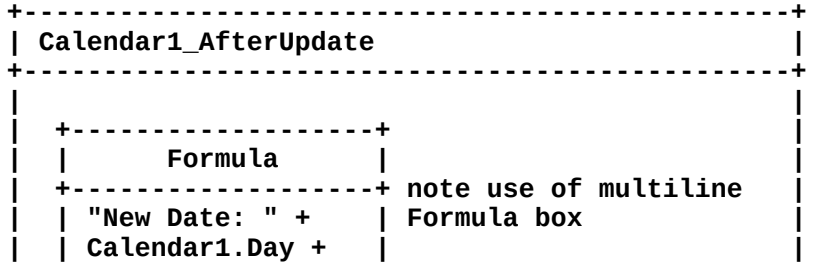


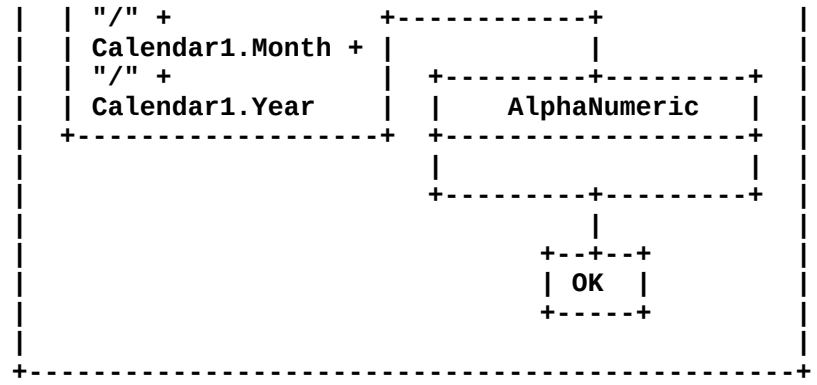
I selected "AfterUpdate" and indeed got an event handler:



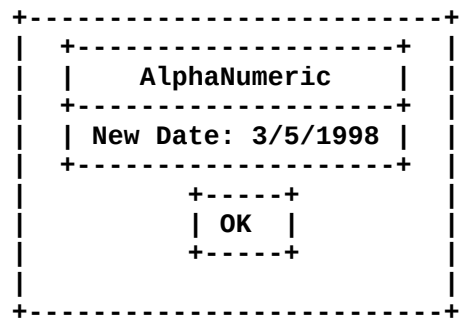
This handler is a UserFunction, but one with some odd restrictions. You can't put input or output pins on it, and in most cases it won't be wired into a program directly. The only way it can communicate with the rest of the program is through global variables. It is essentially a self-contained subprogram that is called when the event (in this case, any update to the Calendar object) occurs.

As a first step, I implemented the handler to simply pop up a panel to show off the new date provided by the user, using the Calendar's properties as obtained through the Function & Object Browser:



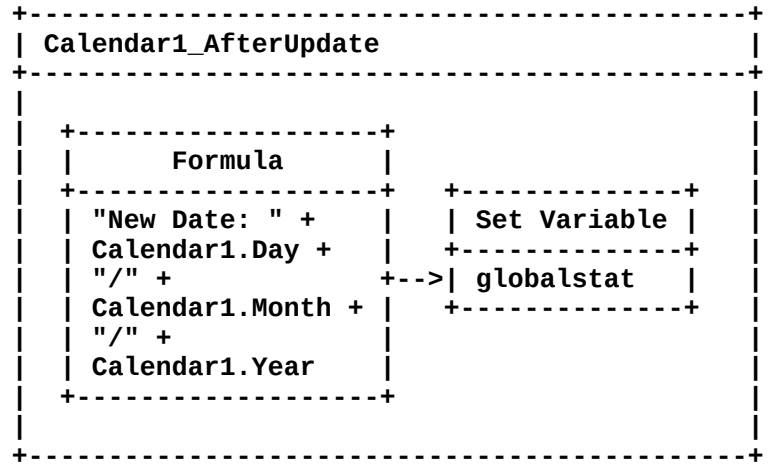


I then set up a Panel View as follows:

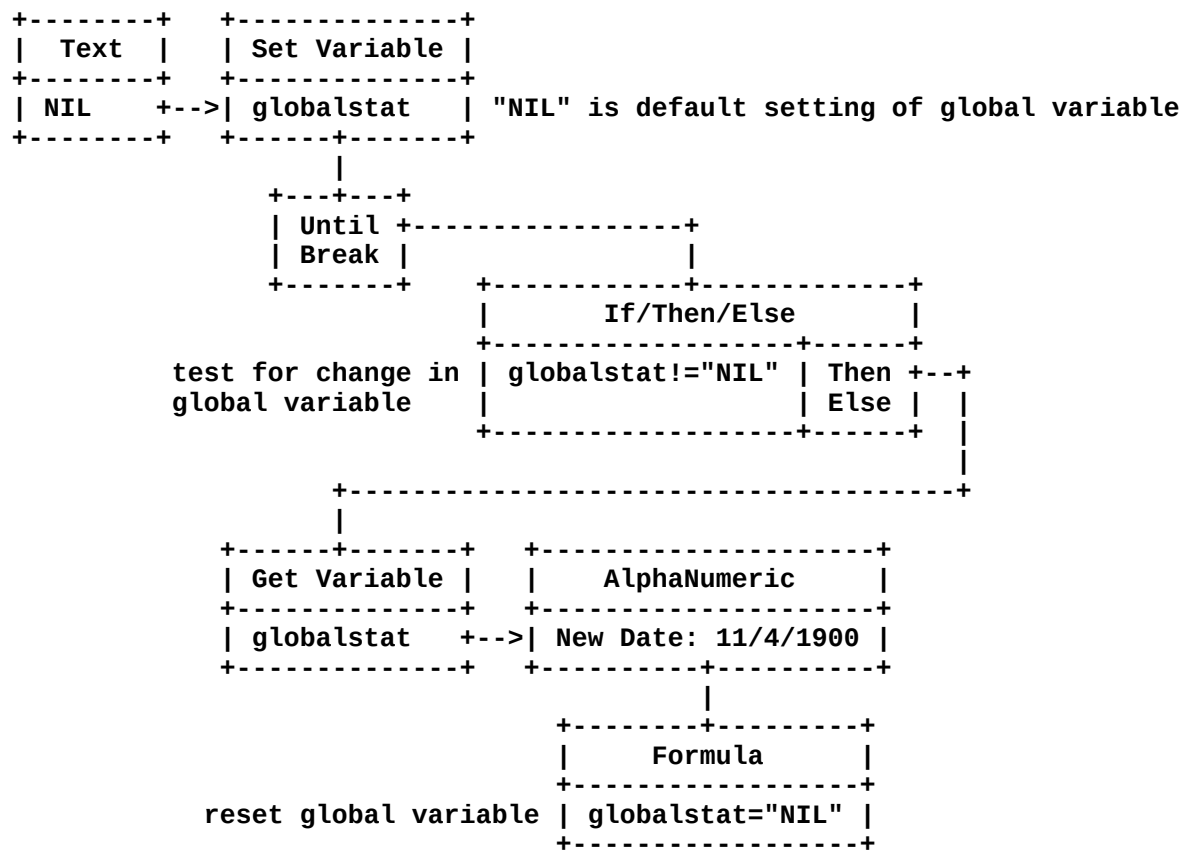


I set this UserFunction to "Show Panel On Execute". I created a Main Program consisting of an Until Break and ran that, then played with the Calendar object. When I updated the day of the month, I got the pop-up.

* It was easy to modify this program to allow the Event Handler to communicate back to the main program. I eliminated the Panel View in the handler UserFunction and changed it to put its text into a Global Variable named "globalstat":



I then extended the main program as follows:



This scheme could be extended to handle other events.

[3] VEE & ACTIVEX AUTOMATION

* ActiveX Automation is a relatively new scheme by which different Windows programs can communicate with each other. It is conceptually similar to the old DDE scheme, but is much better implemented.

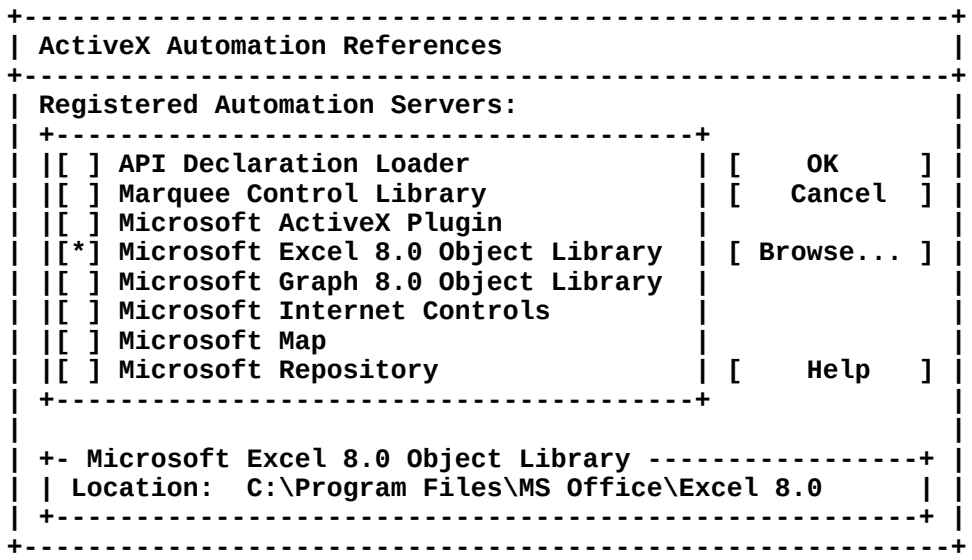
ActiveX Automation is much more robust, faster, and better documented than DDE. When applications are installed under Windows, they are "registered" as ActiveX applications. A user wishing to program VEE as a Windows "client" application that makes use of another Windows "server" application can bring up a list of available server applications, and enable an application for use in VEE.

Then VEE has access to a list of functions for dealing with the server application. The VEE programmer can create formulas containing these commands to access the other application.

Note that the new VEE 5.0 ActiveX Automation features allow it to act as a client. As for server capabilities, those were available in VEE 4.0 through the VEE ActiveX control. These server capabilities were limited to allowing other applications to call VEE userfunctions.

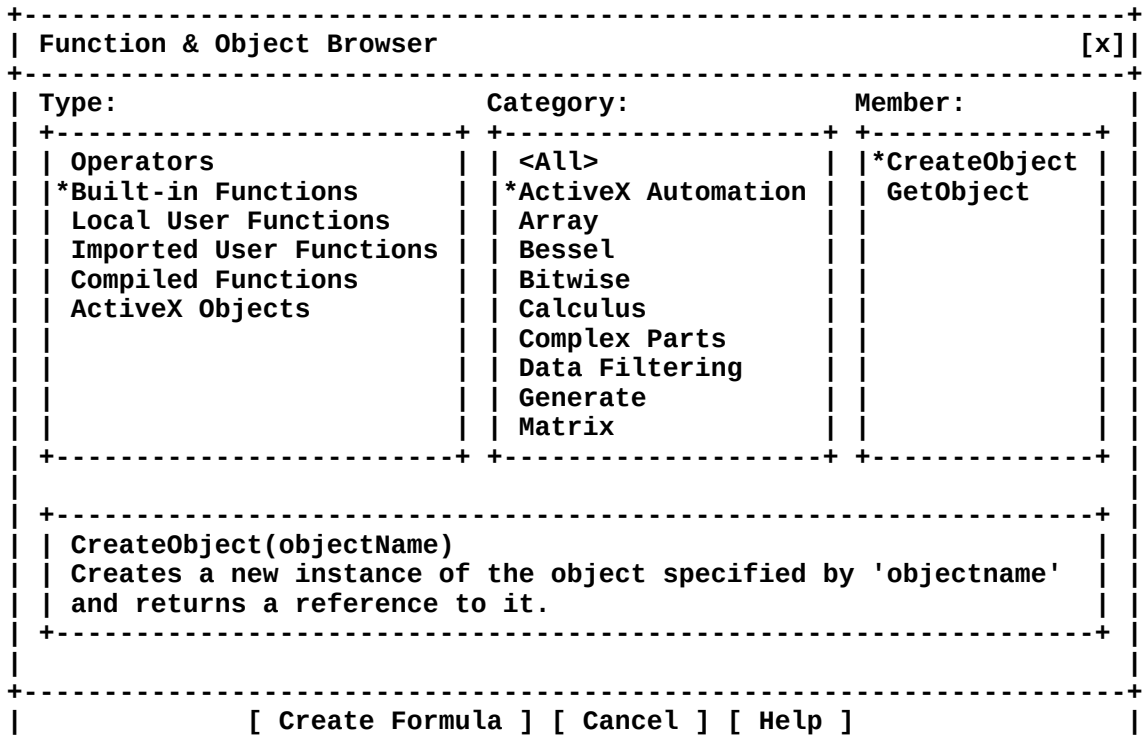
* The ActiveX Automation scheme works very much like the ActiveX Control scheme, and in fact both share the same "object oriented" design philosophy (under a Microsoft spec known as "COM", for "Component Object Model").

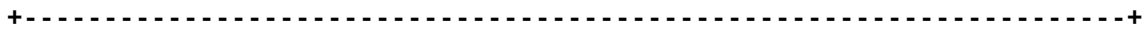
As with ActiveX Controls, the first thing to do is "reference" the application and create an "instance" of an "object" for it. I went to the "Device" menu and selected "ActiveX Automation References". This gave me a dialog box that listed the available applications:



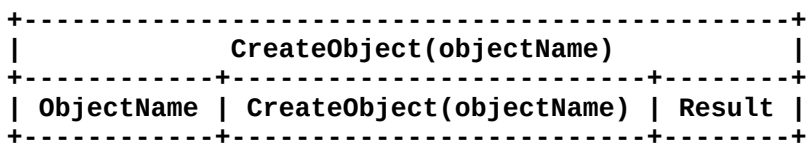
Having referenced the Excel 8.0 object library, I was then able to create an instance of an object to deal with Excel. This was done somewhat differently from the way an ActiveX Control instance is created; in this case I went directly to the Function & Object Browser ... and selected "Built-in Functions" (*not* "ActiveX Objects").

I then selected the "ActiveX Automation" category and the "CreateObject" function:



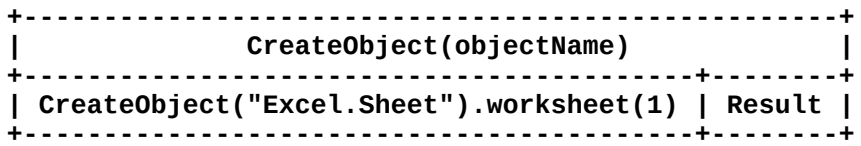


This gave me the function:

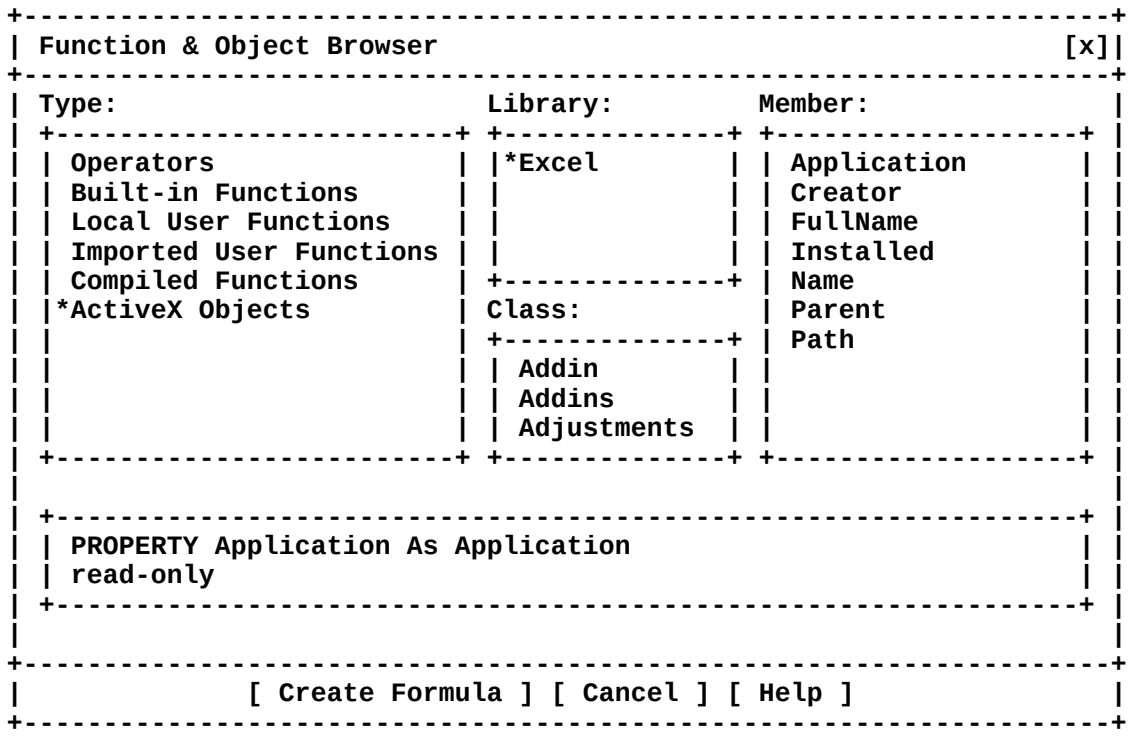


What the exact syntax for this object is must be determined by examination of the target application's documentation ... it will generally reference Visual BASIC as the controlling software, but the syntax will be the same for VEE.

Supposing we want to get an Excel spreadsheet, we would then modify this as follows:

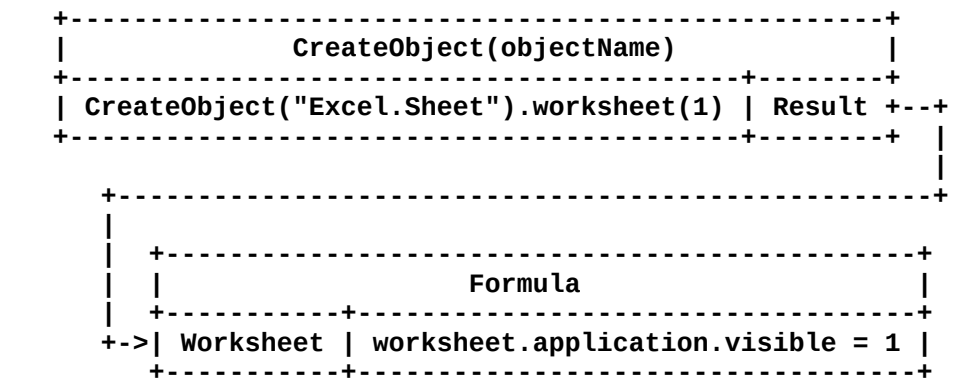


The result of this function is a handle that will be used by functions that set properties or execute methods for that spreadsheet. As with ActiveX Controls, these are obtained with the Function & Object Browser:



For example, an elementary property would be to make the spreadsheet visible. This can be done

with:



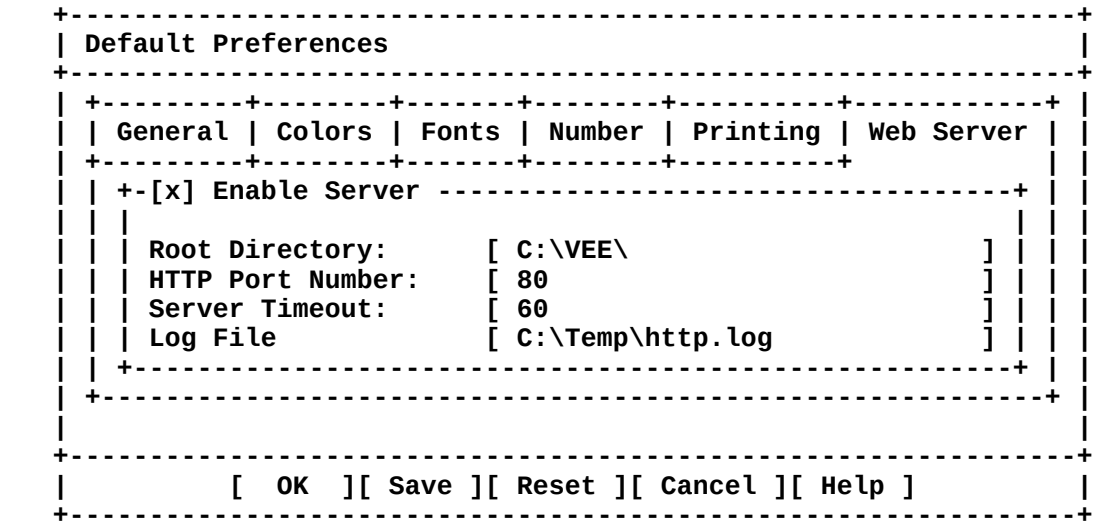
Further properties and methods can be implemented in the same way. However, more details will require a more extensive document than this one.

[4] VEE & THE WORLD-WIDE WEB

* VEE 5.0 has a simple means of interfacing to the World-Wide Web. You can enable a simple HP VEE HTTP server that will allow inspection (though not control) of a VEE program over the Web.

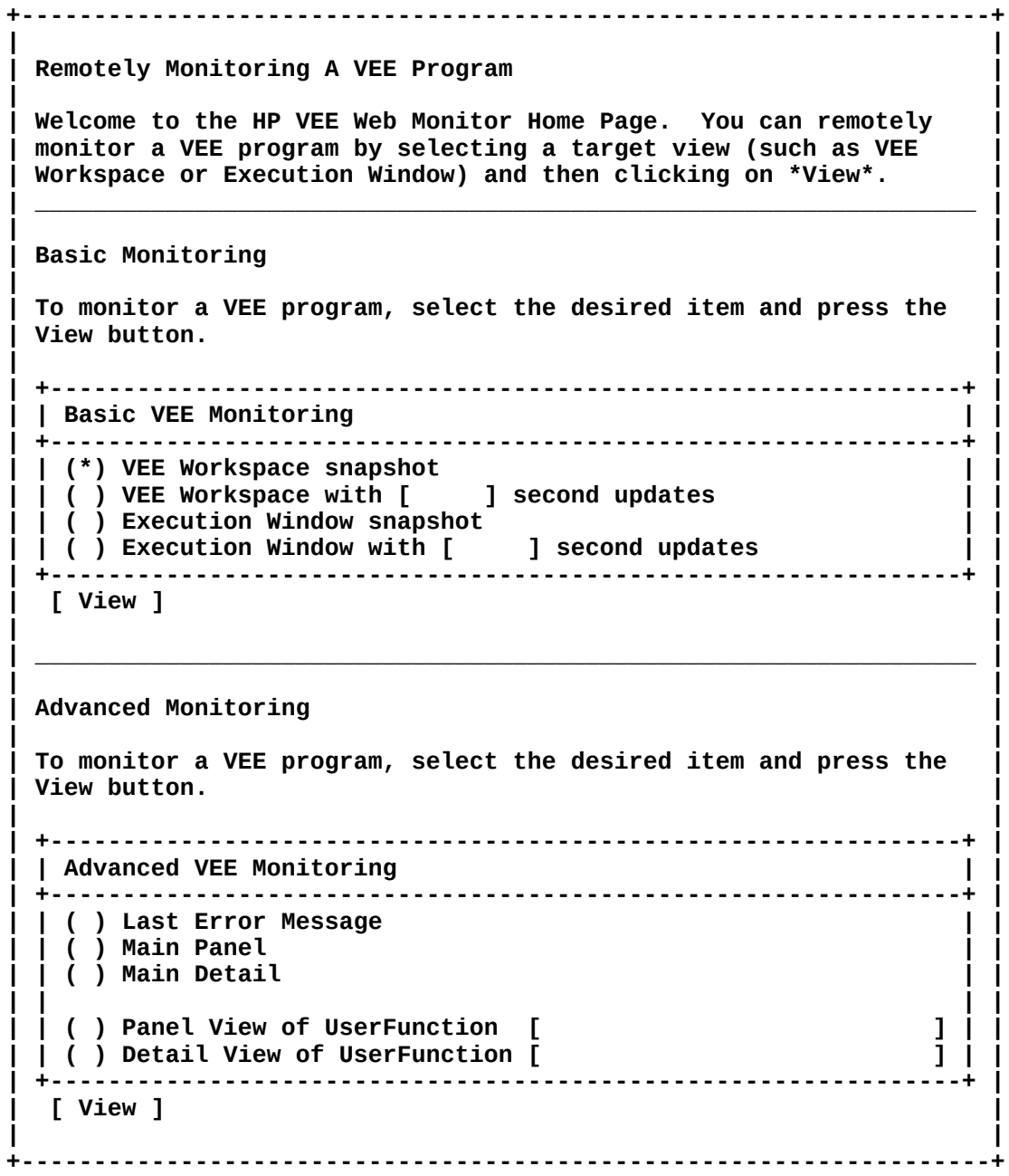
The VEE HTTP server supports a number of commands that are defined as Web addresses (URLs). These commands take a "snapshot" of some element of the VEE display (such as the Runtime window or a function panel) and send a bitmap of it back to the Web surfer. Since as far as the Web is concerned, these commands look like addresses, they are transparent to the Web and no special knowledge of the VEE HTTP server is required.

* All you have to do to enable the VEE HTTP server is go to the VEE Default Preferences configuration dialog box, and click on "Enable Server":



Once this was done, I could then use the name of my PC -- "http://hplvlgvg" -- as an address for my

Web browser to access the VEE Web server that was then installed. This default Web page looked like this:



This default Web page shows off the capabilities of the VEE Web server, which allows you to take bitmap snapshots of different VEE views, such as the VEE Workspace, the Execution Window, error messages, main panel, the panel view of a specified UserFunction, and so on.

This page is built with standard Hypertext Markup Language (HTML) tags, with the various commands used to access VEE implemented as extensions to the Web server's URL:

```
see entire VEE window:     http://<server>/ViewVEE
see main VEE panel:        http://<server>/ViewMainPanel
```

```
see main program:          http://<server>/ViewMainDetail
see Execution Window       http://<server>/ViewExecWindow
see UserFunction panel:    http://<server>/ViewPanel?<userfunction_name>
see UserFunction program:  http://<server>/ViewDetail?<userfunction_name>
see last error message:    http://<server>/ViewError
```

For example, to view the Execution Window of VEE on my pc, I could type the following URL into my browser:

```
http://hplvlgvg/ViewExecWindow
```

If I wanted to get the Panel View of a UserFunction named "DataGen", I would enter:

```
http://hplvlgvg/ViewPanel?DataGen
```

A user could build his or her own web page using these commands and standard HTML codes. There are additional techniques for building more sophisticated VEE web servers using other VEE tools that will eventually be detailed in application notes.

[5] OTHER CHANGES

* One of the other important changes in VEE 5.0 has been the introduction of bitmap handling features. There is a new function named SavePanelImage() that allows saving the image of a UserFunction panel. It has the syntax:

```
SavePanelImage( "SomePanel", "SomeBitMapFile.bmp", 256 )
```

The first parameter is the name of the UserFunction panel. It *won't* work on UserObjects, but you can get it to work on the Main panel by specifying "Execution Window" as the text.

The second parameter is the name of a bitmap file that will store the image; a full path can be specified. If this parameter is not specified, the bitmap file will be given the name of the UserFunction panel and stored in the VEE installation directory. Supported bitmap formats are .BMP (not supported on HP-UX) or .JPEG (.JPG).

The third parameter is the number of colors, and this must be a power of 2: 2,4,8,16, 256, or whatever. This defaults to 256 if not specified. It only applies to .BMP files, as JPEGs are alway 24-bit color.

A related improvement is better clipboard support. While in VEE 4.X, a user could cut and paste VEE objects between different copies of VEE running on the same PC, in VEE 5.0 he or she can also cut and paste VEE objects to any Windows application (such as Word or Paint) that supports "bitmap rendering".

UserObject and UserFunction object menus now have a new function, "Copy", to support clipboard transfers.

* Other changes in VEE 5.0 include:

- The graphics display objects (XY Trace and so on) have been largely re-engineered and have a different appearance. Graph configuration has been rationalized, number formatting is supported, and configuration has been improved by providing all configuration settings in the "tabfolder" configuration.

- Sliders, knobs, and indicators -- meters, thermometers, tank, fillbar
- have received a number of small updates, such as log scales, user-defined tic label spacing, and numeric formatting.

- Formula boxes now support a multiline format. This is just a typing convenience, however; stepping occurs through an entire function at a time. Multiple formulas can be accommodated in a single formula box, as in VEE 4.0, but they can now be written as multiple lines.

- Memory management is more intelligent: while older versions of VEE didn't release memory allocated for large arrays when program contexts will do so, VEE 5.0 will.

- The secured runtime VEE programs (VXE files) are now in a compressed format. This ensures that they take up less disk space, load faster, and are more difficult to decipher (not that it was easy before).

- There were a few cosmetic changes. For example, the Toolbar Buttons are flat rather than raised, and the tabs in the tabfolders are smaller. The buttons in the Instrument Manager are more informative -- "Find Instruments" instead of "Refresh", for example. Finally, the "Select Function" dialog box has been renamed to the "Function & Object Browser".

- UserFunctions in VEE 4.X had a menu pick to allow a user to create a call to the UserFunction. This menu pick was named "Generate Call". VEE 5.0 turns this menu entry into a cascade menu to allow a user to generate a number of different objects that incorporate the UserFunction: Call, Formula, If/Then/Else, ShowPanel, HidePanel.

  This enhancement is also encorporated in the Program Explorer menu associated with UserFunctions.

- The F1 key now brings up Help for the selected object.

- The VEE Help menu has a web hyperlink to VEE online services.

- VEE now refuses to let a user wire from a Sequence-Out pin to a Data Input pin.

[6] BUGS FIXED IN VEE 5.0

* The following bugs were fixed in VEE 5.0:

- Memory Hoard With Direct I/O Address Pins: You can bring out a pin on the Direct I/O object to allow programmatically setting the address of a instrument. Unfortunately ... every time you did this, VEE allocated a little more memory.

  You can also bring out a pin on Direct I/O to allow you to specify an I/O configuration name ... it had the same problem as the address pin.

- VEE Execute Program Causes Memory Leak: It turns out that if you performed EXECUTE

PROGRAM from HP VEE, it drops a little memory each time a program is executed.

- ID Compiler Won't Work: In principle, you should be able to configure a classic VEE ID through the Instrument Manager and it will be automatically compiled. In practice, however, there was a "long file names" problem that prohibited it.
- Print Program Truncations: If you use PRINT PROGRAM to print a function that's bigger than the screen, sometimes you get multiple pages that give the entire function in parts, sometimes you just get one screen. The problem occurs if a window in the program is maximized.
- Pop-Up Dialog Placement: We had a number of complaints from users that when they ran programs, they had no control over the placement of pop-up dialogs when they appear on the screen.
- Line Probe Reveals Partial Arrays: The line probe feature under VEE 4.01 can demonstrate very odd behavior in displaying text arrays where the elements only have a few characters, with the effect that only part of the display can be shown.
- No RS-232 Support For PNP Drivers On VEE: Instrument Manager didn't allow you to configure a driver if it was connected to a serial interface.
- Hostname Pin Causes BRB: If you brrought out the Hostname pin on a Direct I/O object and tried to shove a LAN address into it, you got a Big Red Box.
- Load Library Causes GPF IN VEE: We had programs that loaded UserLibraries with Global Variables and obtained a GPF.
- VEE RPCS Don't Like Indirect Function Calls: If VEE tried to load and use a set of UserFunctions as Remote Procedure Calls (RPCs) on remote systems, and one of these UserFunctions called another UserFunction in the UserLibrary that VEE was not accessing, a segmentation error occured.

  A segmentation violation would also occur if VEE tried to read back the output of an RPC UserFunction that had no data on the output.

Many other obscure bugs were fixed:

- Serious errors caused by setting a For Count to 0, or sorting on an Integer array, or cutting out a DropDown List box (under certain conditions), or assigning values to record fields containing multidimensional arrays.
- An odd problem where a program running a For Range counting up gave peculiarly different results from one where it counted down.
- Inconsistent handling of trace scales in Log mode for an XY Trace object.
- A lockup due to running out of memory while accessing Datasets, and using EXECUTE REWIND through a From File object.
- Creating a Constant Record with a 2D array of type Coordinate led to an array of fewer elements than specified.

- A very funny bug with Message Boxes, where if you tried to change the output pin configuration *twice*, it would reverse the triggering of the output pins (meaning the user would always get the wrong output).
- There was an obscure problem with Drop-Down Lists that were set to Auto-Execute and led to a lockup. There was also a subtle List object problem where if it were embedded in panels inside UserFunctions, it would not respond properly to inputs.
- Under some circumstances, comparisons involving arrays would not work properly.
- One user accidentally put double-quotes ("") in the output pin of a Direct I/O object. He was still able to save the program, but it was corrupted.
- There was an obscure bug where a set of Radio Buttons built into a Record could be compared with a larger set of Radio Buttons and not yield an error.

[<>]