PULSE
Automation programming in Visual Basic

From BK training lectures arranged by Jiří Tůma & Radovan Zadražil
OLE
Automation, general principles

Bruno S. Larsen
PULSE Software Development
Agenda

• Overview of Automation
  • Automation of PULSE
  • Why Expose Objects
  • What Is an ActiveX Object?
  • ActiveX Objects in Pulse
  • What Is an ActiveX Client?
  • Automation Summary
Overview of Automation

OLE
- **Object Linking and Embedding**
- Also known as OLE2
Overview of Automation

Automation

• Originally known as OLE Automation
• An application can expose its features to other programs
• Automation is based on a Client/Server architecture

Application (Program)
- PULSE
- MS Word
- MS Excel
Overview of Automation

The Client/Server architecture

- A Server exposes its functionality to other programs
- Clients are applications that access the services of a Server
Overview of Automation

- Automation is a Microsoft technology that implements a Client/Server architecture.

- It allows a software package (Server) to expose its unique features to other applications (Clients).

```
Automation Server
- PULSE
- MS Word
- MS Excel
```
```
Automation Client
Visual Basic
```
Agenda

• Overview of Automation
• Automation of PULSE
• Why Expose Objects
• What Is an ActiveX Object?
• ActiveX Objects in Pulse
• What Is an ActiveX Client?
• Automation Summary
Automation of PULSE

- Normally the Automation Server and the Client work tightly together in a one-to-one relation on the same PC.
Automation of PULSE

- Automation uses the Component Object Model (COM)
Automation of PULSE

• The Client and the Server program are able to run on separate PC’s connected by a LAN using Distributed COM (DCOM)
Agenda

• Overview of Automation
• Automation of PULSE
• Why Expose Objects
• What Is an ActiveX Object?
• ActiveX Objects in Pulse
• What Is an ActiveX Client?
• Automation Summary
Why Expose Objects?

• The Automation Server exposes its unique features to other applications
• The exposed features are called objects
• Using automation you can manipulate the exposed objects in one application from another application
• Exposing objects provides a way to manipulate an application’s tools programmatically
Why Expose Objects?

• With Automation solution providers can use general-purpose objects to build applications that target a specific task
Why Expose Objects?

• The PULSE customers can use a programming tool to automate repetitive tasks that might not have been anticipated.
Agenda

• Overview of Automation
• Automation of PULSE
• Why Expose Objects
• What Is an ActiveX Object?
• ActiveX Objects in Pulse
• What Is an ActiveX Client?
• Automation Summary
What Is an ActiveX Object?

- The exposed objects are called **ActiveX objects**
- The Automation Server is called an **ActiveX Server** or an **ActiveX component** acting in the role as server
What Is an ActiveX Object?

- An object class describes a group of objects with similar attributes and common behaviour.

- An ActiveX object is a concrete instance of that class.

Display

Class

Objects

Display

Display 1

Display 2

My Display
What Is an ActiveX Object?

- An ActiveX object exposes properties and methods.
- *Properties* are used to access information about the state of an object.
- For example, one of the objects in PULSE has a *Visible* property that determines whether pulse is visible.
- *Methods* are actions that an object can perform.
- For example, one of the objects in PULSE has an *OpenProject* method that opens a PULSE project.

<table>
<thead>
<tr>
<th>ActiveX Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Properties:</strong></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Visible</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td><strong>Methods:</strong></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>OpenProject</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Agenda

- Overview of Automation
- Automation of PULSE
- Why Expose Objects
- What Is an ActiveX Object?
- ActiveX Objects in Pulse
- What Is an ActiveX Client?
- Automation Summary
ActiveX Objects in Pulse

- PULSE exposes 60 different ActiveX objects
- All in all 1023 different properties and methods are exposed
- The objects are organised hierarchically, with an object named Application at the top of the hierarchy
- Most of the objects maps directly to a well known part of the PULSE user interface
ActiveX Objects in Pulse (From OLE2.HLP)
# ActiveX Objects in Pulse

<table>
<thead>
<tr>
<th>Application</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DemoMode</td>
<td>Enabled</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>PulseNotification</td>
<td>Visible</td>
<td></td>
</tr>
<tr>
<td>Methods:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActivateTemplate</td>
<td>Autorange</td>
<td>CalibrateMode</td>
<td></td>
</tr>
<tr>
<td>CloseProject</td>
<td>Exit</td>
<td>FilterExtension</td>
<td></td>
</tr>
<tr>
<td>FilterName</td>
<td>NewProject</td>
<td>OpenProject</td>
<td></td>
</tr>
<tr>
<td>Proceed</td>
<td>Reset</td>
<td>Save</td>
<td></td>
</tr>
<tr>
<td>SaveProject</td>
<td>SaveProjectAs</td>
<td>Start</td>
<td></td>
</tr>
<tr>
<td>StartGenerator</td>
<td>Stop</td>
<td>StopGenerator</td>
<td></td>
</tr>
<tr>
<td>Trigger1</td>
<td>Trigger2</td>
<td>Trigger3</td>
<td></td>
</tr>
</tbody>
</table>
Agenda

- Overview of Automation
- Automation of PULSE
- Why Expose Objects
- What Is an ActiveX Object?
- ActiveX Objects in Pulse
- What Is an ActiveX Client?
- Automation Summary
What Is an ActiveX Client?

- Applications that use Automation to access exposed ActiveX objects, are called **ActiveX clients**
- An ActiveX Client can also be described as an ActiveX **component** acting in the role as client

ActiveX Clients can:
- Use existing objects
- Create new instances of objects
- Get and set properties supported by the object
- Invoke methods supported by the object
What Is an ActiveX Client?

- You can use Visual Basic and similar programming tools as VC++, DELPHI and Word-VBA and Excel-VBA to create ActiveX Clients.
Agenda

• Overview of Automation
• Automation of PULSE
• Why Expose Objects
• What Is an ActiveX Object?
• ActiveX Objects in Pulse
• What Is an ActiveX Client?
• The Notification system in PULSE

• Automation Summary
Automation Summary

**ActiveX Server**

Expose features to other programs as ActiveX objects

- Expose properties and methods

**ActiveX Client**

Access exposed ActiveX objects
Get/set properties
Invoke methods
Getting Pulse Data
Getting Pulse Data

• Dataset objects can be accessed from the functions in the Function Organiser.
• Also the functions added to the displays in the Display Organiser gives access to datasets.
• In the Measurement Organiser datasets can be accessed from Input or Saved measurement data. Both the Input buffer and the Multi Buffers contains datasets.
Getting Data from the Function Organiser
Getting Data from the Function Organiser

Dim FunctionData As Object  ‘Using late binding
  or
Dim FunctionData As BKDataSet   ‘Using Type Library

Set FunctionData = Project.FunctionOrganiser.
                    FunctionGroups("FunctionSet").
                    Functions("Autospectrum(Signal 1)").
                    FunctionData
The **BKDataSet** object

The **BKDataSet** object has the following **properties**:  

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NumberOfXAxisEntries</strong></td>
<td>(Number of lines/frequencies etc.)</td>
</tr>
<tr>
<td><strong>NumberOfZAxisEntries</strong></td>
<td>(Number of spectra in multibuffer)</td>
</tr>
<tr>
<td><strong>XAxisUnit</strong></td>
<td>(Hz, RPM etc.)</td>
</tr>
<tr>
<td><strong>YaxisUnit</strong></td>
<td>( V^2 ), etc. Dependent to the type of spectrum. Independent of display setting!</td>
</tr>
<tr>
<td><strong>ZaxisUnit</strong></td>
<td>( )</td>
</tr>
<tr>
<td><strong>IsComplex</strong></td>
<td>(Is data complex ?)</td>
</tr>
<tr>
<td><strong>AcceptedInMultibuffer</strong></td>
<td>(Number)</td>
</tr>
<tr>
<td><strong>MultibufferStartIndex</strong></td>
<td>(Number)</td>
</tr>
</tbody>
</table>
The BKDataSet object

The **BKDataSet** object has the following **methods**:

- Frequencies (Get Frequencies)
- RealValues (Get real value part of a vector)
- ImaginaryValues (Get imaginary part of a vector)
- GetAllValues (Get real or imaginary multibuffer)
- GetAverageNumber (Look up the Average Number)
- GetAveragingTime (Look up the Averaging Time)
- GetOverloadRatio (Look up the Overload Ratio)
The BKDataSet object

Syntax

MyPulseData = Object.GetAllValues (Mode as Boolean)

The **GetAllValues** method syntax has three parts:

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyPulseData</td>
<td>Local Variable declared as Variant</td>
</tr>
<tr>
<td>Object</td>
<td>An object expression that evaluates to an BKDataSet object.</td>
</tr>
<tr>
<td>Mode</td>
<td>Selects real- (True) or imaginary- (False) part.</td>
</tr>
</tbody>
</table>
The Visual Basic Sample Program

Dim FunctionData As BKDataSet
Dim MyPulseData As Variant
Dim maxX As Long
Dim maxZ As Long

Set FunctionData = Project.FunctionOrganiser.
    FunctionGroups("FunctionSet").
    Functions("Autospectrum(Signal 1)").
    FunctionData

FunctionData.GetAllValues(True) ' Get real Y values
maxX = Ubound(MyPulseData,1)   ' Get max X Index (base 0)
maxZ = Ubound(MyPulseData,2)   ' Get max Z Index (base 0)
The Visual Basic Sample Program

'If the dataset is empty the GetAllValues method will throw an exception.
'We need to introduce some error handling.

On Error Resume Next
For I = 1 to 100
   MyPulseData = FunctionData.GetAllValues (True)
   If Err.Number = 0 Then
      Exit For
   End If
Next I
If I > 100 Then
   MsgBox Err.Description
   Exit Sub
End If
On Error GoTo 0 'Restablish default error handling
Getting PULSE data using GetAllValues
The BKDataSet object

Syntax

Object.**RealValues** DataArray [,InxZ]

The **RealValues** method syntax has three parts:

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>An object expression that evaluates to an BKDataSet object.</td>
</tr>
<tr>
<td>DataArray</td>
<td>An array of strings or an array of values</td>
</tr>
<tr>
<td>InxZ</td>
<td>A numerical expression (optional).</td>
</tr>
</tbody>
</table>
The BKDataSet object

Object.**RealValues** DataArray [,InxZ]

The settings for the **RealValues** method are:

<table>
<thead>
<tr>
<th><strong>DataArray</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Array of strings</td>
<td>A reference to an array of strings.</td>
</tr>
<tr>
<td>Array of reals</td>
<td>A reference to an OLE Automation SafeArray of real values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>InxZ</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Default - InxZ is set equal to 0. (This is the first data vector in the BKDataSet structure.)</td>
<td></td>
</tr>
<tr>
<td>0 to NumberOfZAxisEntries – 1 (The index of the data vector containing the RealValues data structure to be copied).</td>
<td></td>
</tr>
</tbody>
</table>
The BKDataSet object

Object.RealValues DataArray [,InxZ]

• The DataArray structure must be declared in the application that is controlling PULSE and the size of the array must be equal to the number of x-axis entries.
• In Visual Basic, use the ReDim statement to declare and allocate storage space for an array variable of the type String, Double or Single.
• In other languages use a safearray of strings (VT_BSTR) or reals (VT_R8 for double or VT_R4 for float/single).
The Visual Basic sample program Readdata

Dim FunctionData As Object ' BKDataSet
Dim Entries As Long

Set FunctionData = Project.FunctionOrganiser. FunctionGroups("FunctionSet"). Functions("Autospectrum(Signal 1)").FunctionData

Entries = FunctionData.NumberOfXAxisEntries

ReDim ValueArr(Entries - 1) As Double' Dimension array variable

FunctionData.RealValues ValueArr ' Get Y values
Getting Data from the Display Organiser
Getting Data from the Display Organiser
Getting Data from the Display Organiser

Dim FunctionData As Object ' BKDataSet

‘Get the dataset form one of the functions
Set FunctionData = Project.DisplayOrganiser.
    DisplayGroups("Display Group").
    Displays("Autospectrum(Signal 1)").
    FunctionData

‘Or get the dataset directly from the active function:
Set FunctionData = Project.DisplayOrganiser.
    DisplayGroups("Display Group").
    FunctionData
Getting Data from the Measurement Organiser
Getting Data from the Measurement Organiser
Introduction to Collections & Navigating the Pulse Tree
Introduction to Collections

A Collection is a container able to contain everything that’s an Object
Introduction to Collections

The defining characteristics of an Object

An Object is something that can be accessed through an OLE Automation interface. Examples of Objects could be: a FunctionGroup, a Functions, a SignalGroup, a Signal, a DisplayGroup, and a Display.
Introduction to Collections

The defining characteristics of a Collection

• A (unordered) group of Objects.
• The ability to iterate over the Objects contained in the Collection

Example:

Set Instruments = … Templates(“XXX”).Setup.
    Instruments
For Each Instrument In Instruments
    MsgBox Instrument.Name
Next Instrument
Introduction to Collections

Collection capabilities:

• You can add Objects to a collection
• You can remove Objects from a collection
• You can count Objects in a collection
• You can retrieve an Object from a collection
Introduction to Collections

Collection drawbacks (advantages)

- Collections are not Arrays.
- An Object’s position in a collection is unpredictable.
- Collections can change at any time.
Navigating the Pulse Tree

- Where can you get information about the Pulse Tree:
- The Organisers in Pulse
- The Ole2.hlp help file:
  C:\Program Files\Bruel and Kjaer\Pulse\Exe\Ole2.hlp
  (Not always updated)
- The Pulse “Type Library”, using the Visual Basic Object Browser.
  (Always updated to the used Pulse version)
Working with Notifications
Pulse Notifications

- The Pulse application is able to transmit status information from its objects, called notifications.
- Pulse 5.x introduces a new and easier notification system
- Two types of notification objects are available
  - Notify uses the previously used notifier concept.
  - Notify2 uses a distributed notification system that makes notification filtering easier
Pulse Notify2

• Many of the Objects used in Pulse includes a Notify2 Event
• The Interface to notify2 is simply made by a change in the declaration of the Pulse Object

Example:
Public WithEvents PulseApplication as PulseLabshop.Application

Add a PulseApplication_Notify2 sub, and the notification data is ready for use
Where do You Find Notify2

- List of the Pulse Objects that uses Notify2 Event

Adapter  OrderOnalyzer
Application * OverallAnalyzer
ConfigurationOrganiser Project
CPBAnalyzer * PulseFunction
FFTAnalyzer Recorder
FreeSpanFFTAnalyzer Slice
FunctionOrganiser TachoMeter
Generator Template *
GeneratorSignal TimeCaptureAnaluzer
LoudnessAnalyzer
Thank for paying attention!