Tech Note 652
Changing an ArchestrA Symbol's Custom Property Expression or Reference in Runtime

Topic#: 002416
OpsManage09 Session#: TS106
Created: November 2009
Updated April 2011

Introduction

This *Tech Note* describes changing the expression or reference of a Custom Property at runtime. This is done by calling the **SetCustomPropertyValue(string name, string value, IsConstant)** on the ArchestrA symbol using a custom script.

The goal of using the SetCustomPropertyValue() method is to help change the expression or reference of a Custom Property in runtime.

When using the method, consider the following:

- The symbol element exposes a method that allows you to change the reference of the custom property on the symbol at runtime.
- You can browse this method from client script editor.
- This method has three parameters – (name, value, type).
  a) **Name**: Name of the custom property to be modified on the symbol. This parameter is of type string, and it can be a reference or a constant.

  b) **Value**: The new value to be set. This parameter is of type string, and it can be an expression, reference, or constant.

  c) **Type**: The type of the value. This parameter is of type Boolean, which means that the second parameter is reference or constant only if the custom property (specified in the name parameter) is of string and time type. The type parameter has no meaning if the custom property is an integer, float, Boolean, or double type.
- This method is *only* supported for ArchestrA client scripts.
- **You can only change *public* custom properties on the symbol. The following scenario is an example of what this means in this context:**

  Suppose you create a symbol **S1** and create 5 custom properties: **CP1**, **CP2**, **CP3**, **CP4** and **CP5**.

  CP1 and CP2 are private and other three custom properties are public. In this case *all* custom properties are accessible for read/write by *all* animations and scripts configured inside symbol **S1**, regardless of custom properties visibility (private or public).

  This is because S1 is owner of all custom properties (CP1,..., CP5). A custom property is declared private or public only for the world outside the symbol, and not within the parent symbol itself.

  If you create another symbol **S2** and embed symbol S1 in it, only the public properties of S1 (**CP3**, **CP4** and **CP5**) are accessible for any animation or script configured in symbol S2. CP1 and CP2 of symbol S1 will not be accessible from S2 because they are declared private in S1.

Application Versions

- Wonderware Application Server 3.1 & later
- InTouch 10.1 & later

Procedure: Update the Custom Property Using a Constant

1. Create an ArchestrA Symbols (e.g AASymbol1)

2. Create a custom property **CP1** which has a string data type and assign a default value of **Test**.
3. Create a text object.
4. Add an animation value display link to the text object and point the reference to the Custom Property (CP1)
5. Create a button object called **SetCustomPropertyValue**.
6. Add the below animation action script on left click/key down:

```
SetCustomPropertyValue("CP1","Test SetCustomPropertyValue",1);
```

7. Create a derived InTouchViewApp.
8. Embed the above symbol on an InTouch Window (Figure 1 below).

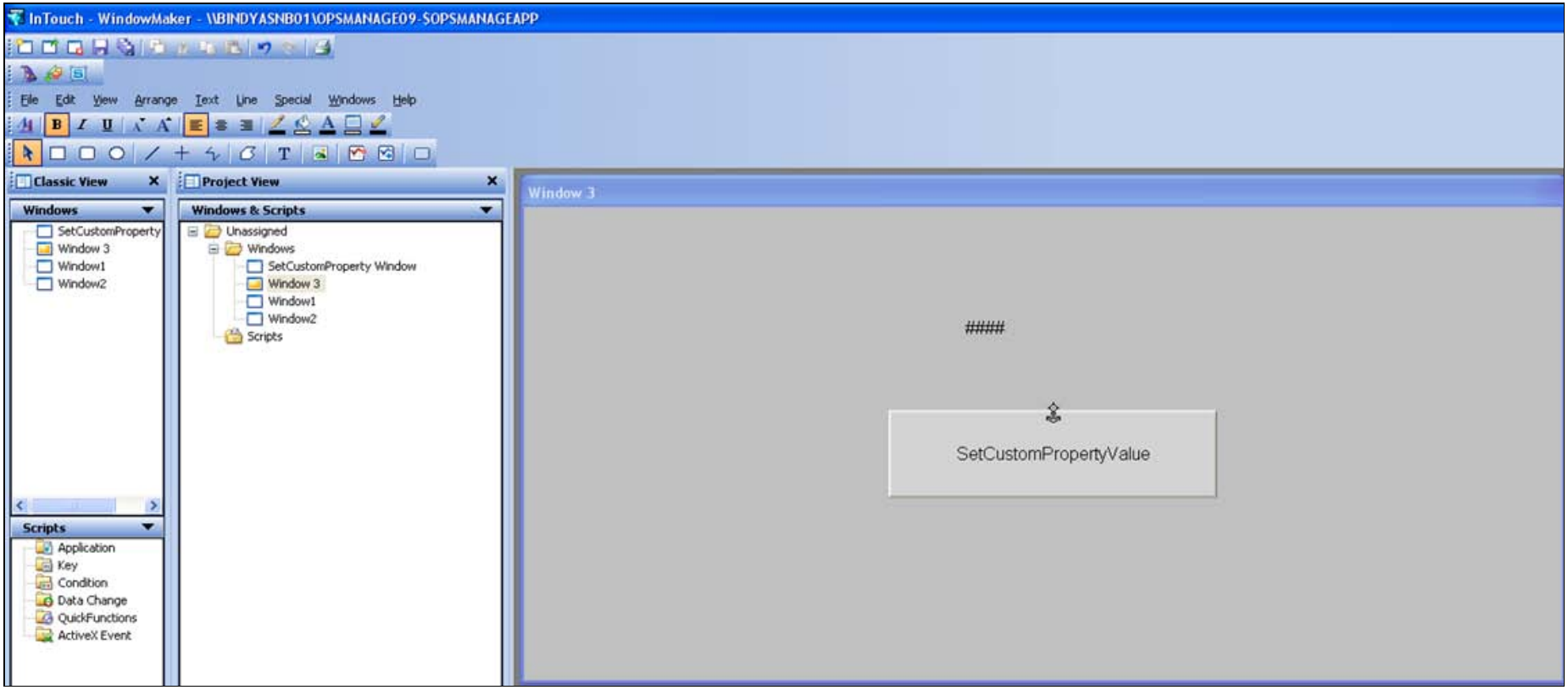

FIGURE 1: EMBEDDED SETCUSTOMPROPERTYVALUE SYMBOL IN INTOUCH

9. Switch to Runtime and notice that the text displays **Test**, which was the default string you added when you created the Custom Property **CP1**. as shown below:

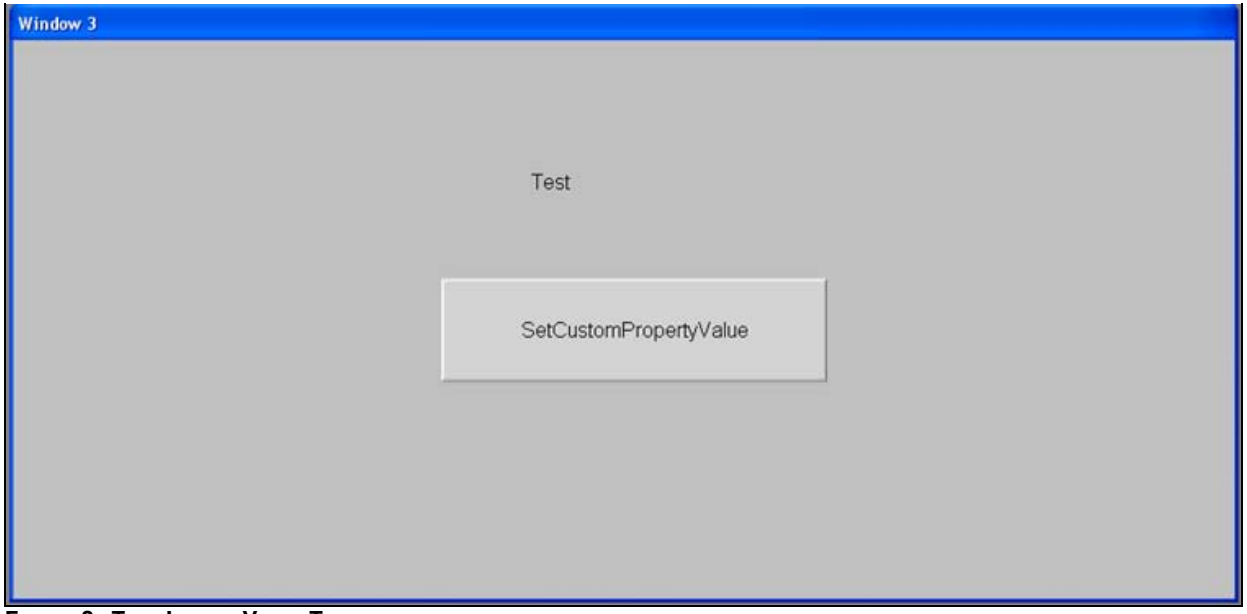

Figure 2: Test Initial Value Text

10. Click the **SetCustomPropertyValue** button.
11. The Action Script changes the value display string to **Test SetCustomPropertyValue** (Figure 3 below).

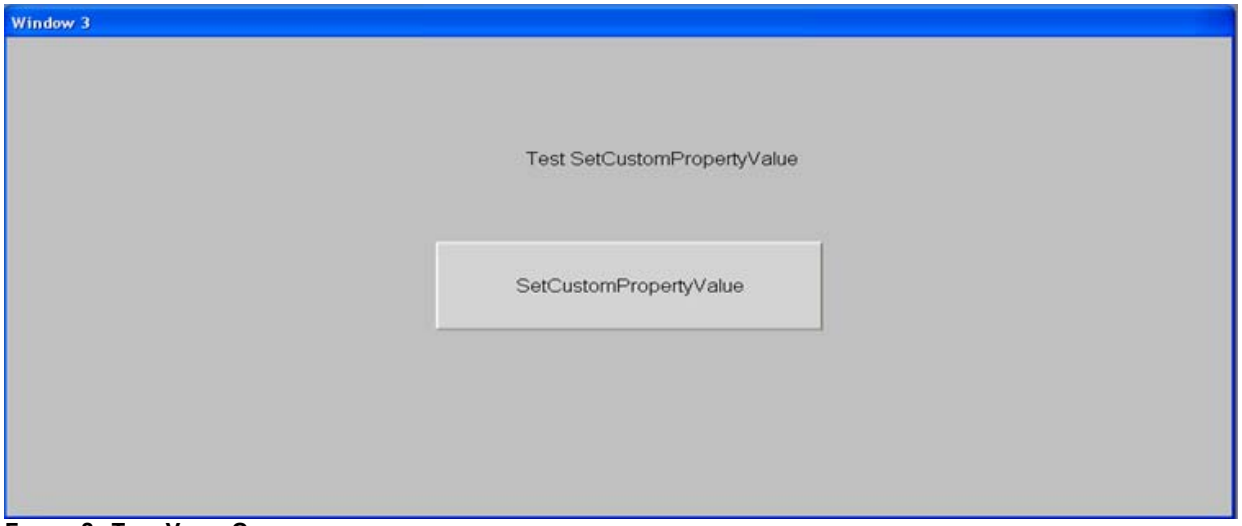

Figure 3: Text Value Change

Procedure: Update the Custom Property Using a Reference

1. Create an Automation Object from the $UserDefined template. This example is called **AO1**.
2. Create 2 UDAs: **UDA1**, **UDA2** in the **AO1** object.
3. Deploy **AO1**.

4. Create a symbol **S1** in the AO1 under the graphic tab with custom properties called **CP1**, **CP2**.
5. Create a 2 text objects **###** user input animations for the UDA1 and UDA2.
6. Create a 2 value display text objects **###** for CP1 and CP2.
7. Create the following action script on a button object and name it **SetCustomPropertyValue()** button.

```
SetCustomPropertyValue("CP1","AO1.UDA1",false);
SetCustomPropertyValue("CP2","AO1.UDA2",false);
```

8. Embed S1 into InTouchViewApp window (Figure 4 below).

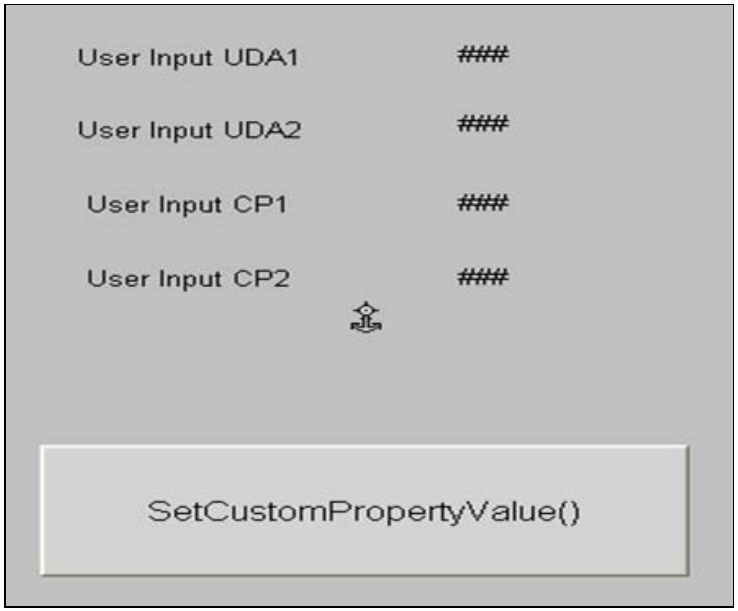

FIGURE 4: SET UP THE REFERENCES

9. Switch to Runtime and notice all the values are **0** (zero) (Figure 5 below).

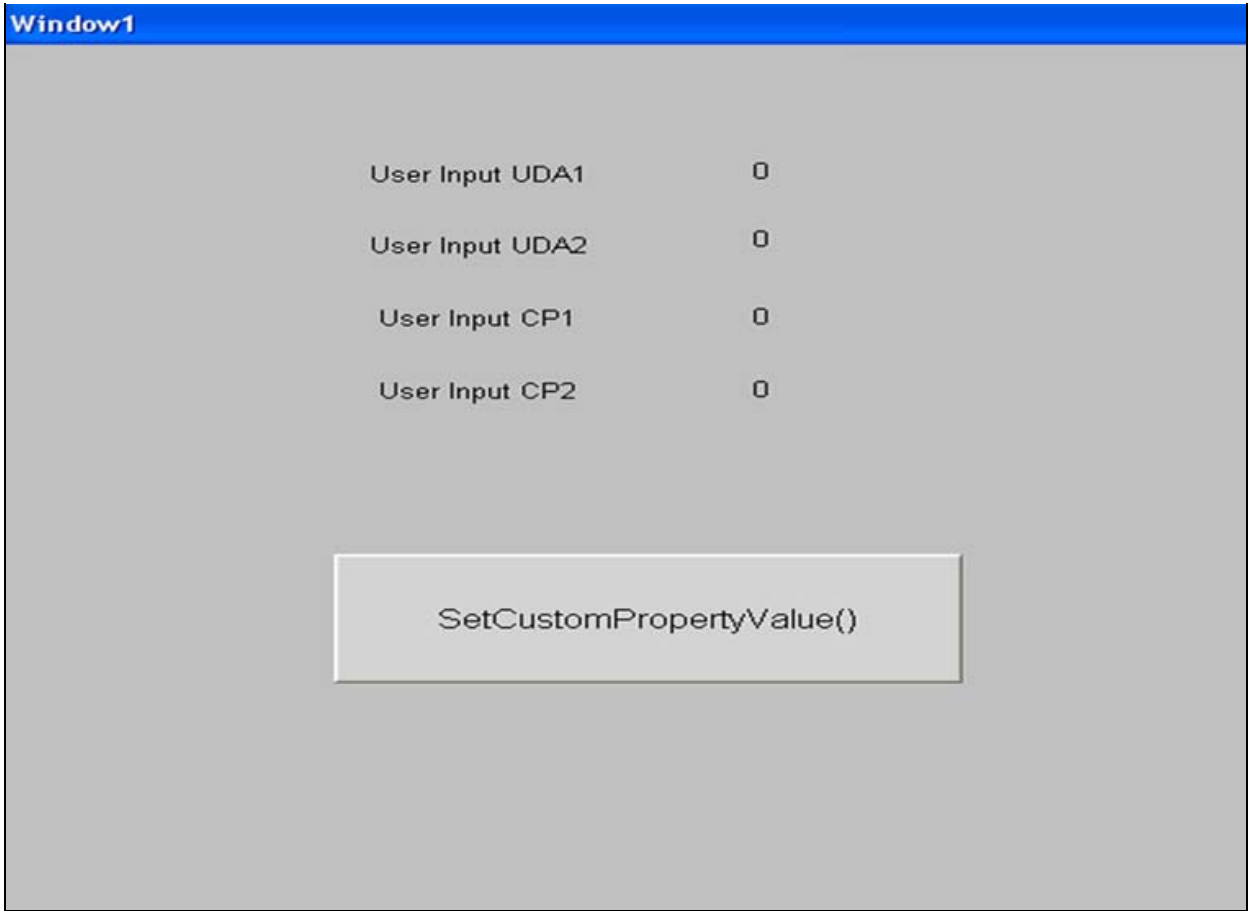

FIGURE 5: INITIAL VALUES ARE ZEROS

10. Type inputs for UDA1 = **10** and UDA2 = **40** as (Figure 6 below).

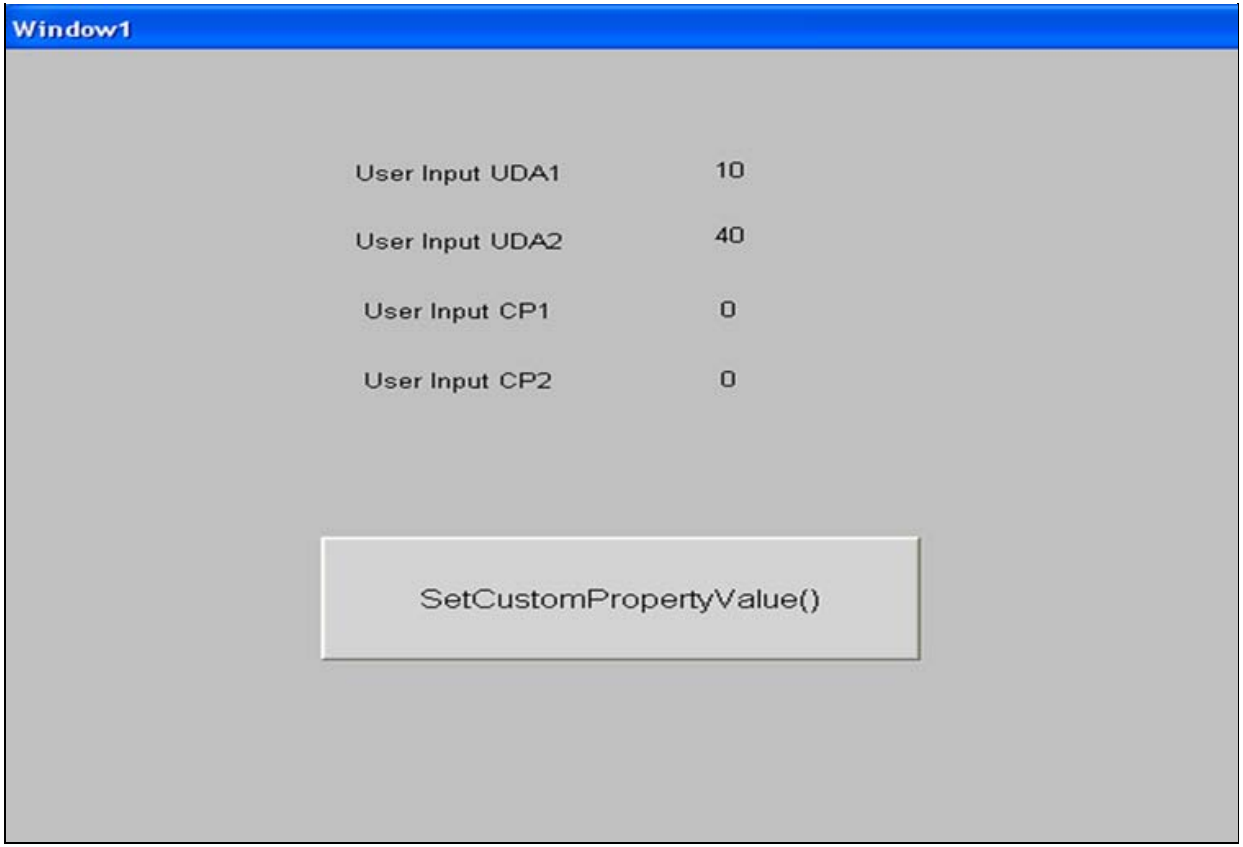

FIGURE 6: TYPE UDA VALUES

11. Click the **SetCustomPropertyValue()** button.
12. Notice that the CP1 and CP2 value change as per the script as shown below:

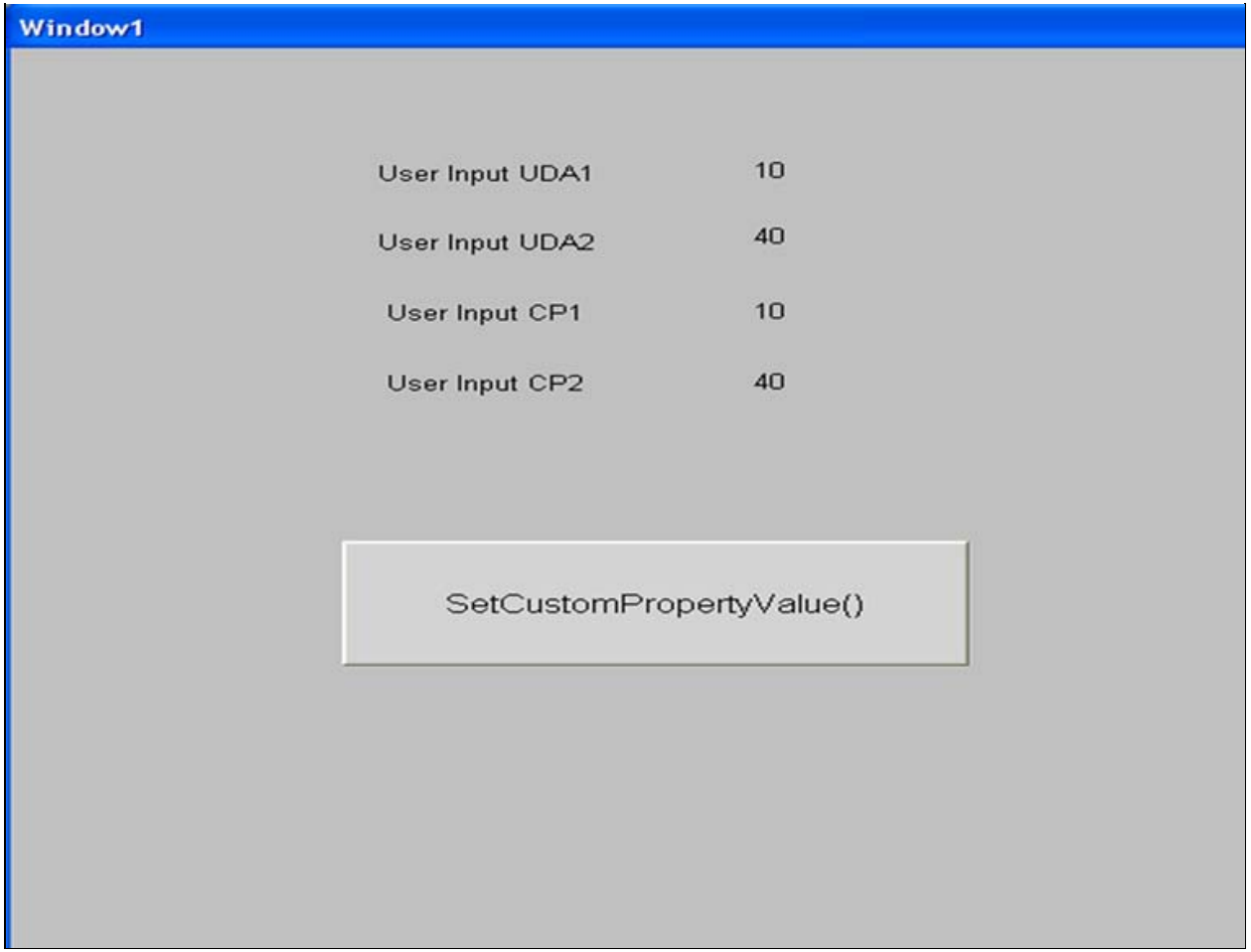

Figure 7: Set Custom Property Values for User Input Fields

Procedure: Update the Custom Property Using an Expression

1. Create two User defined templates called **$Reactor1** and **$Tank1**.
2. Create two symbols: **T_S1** in $Tank1, and **R_S1** in $Reactor1.
3. Create two integer UDAs called **U1** and **U2** in **$Tank1** and one integer UDA called **U1** in $Reactor1.
4. In T_S1, configure two value display animation referring to **me.U1** and **me.U2**.
5. Create two integer custom properties called **CP1** and **CP2** with default values of **me.U1** and **me.U2** respectively.
6. Create a button called **SetCustomPropertyValue()** and configure action script as:

```
SetCustomPropertyValue("CP1", "CP2", false);
```

7. Configure a user input animation referring to CP1 and CP2.

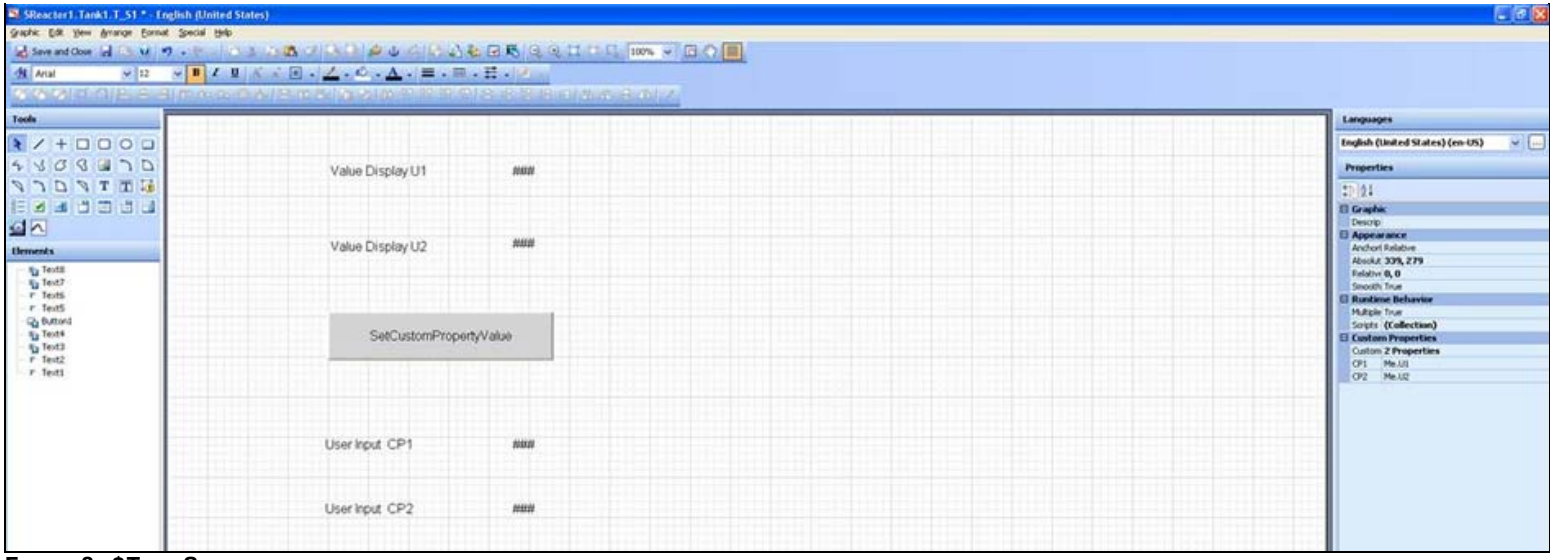

FIGURE 8: $TANK SYMBOL

8. Assign $Tank1 to $Reactor1 and create two instances of $Reactor1.
9. In R_S1, embed T_S1 and configure following action script to a button called **Hierarchical Name**.

```
Tank1_001_T_S11.OwningObject = Tank1_002.HierarchicalName;
```

10. Deploy the two instances.
11. Embed R_S1 in a Window W2 of WindowMaker as shown below:

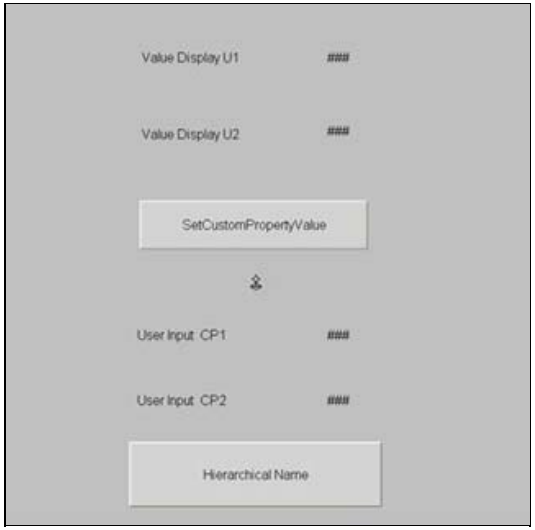

FIGURE 9: EMBEDDED R_S1 SYMBOL

12. Switch to Runtime and note the default values (10 and 20 respectively).

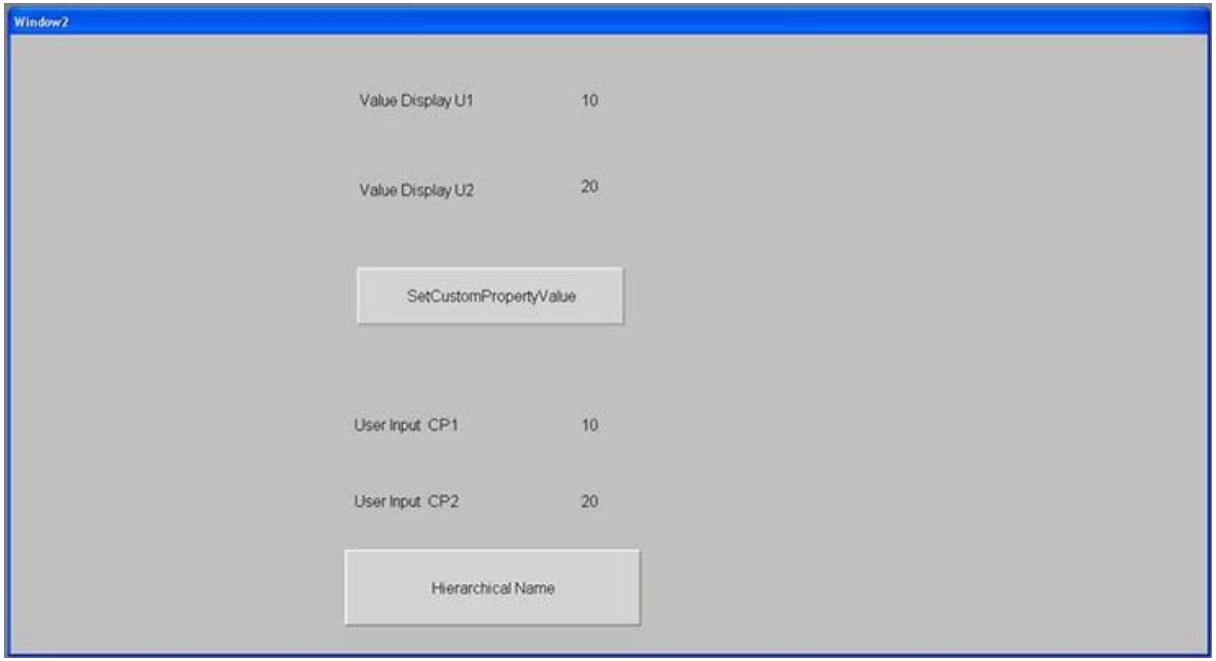

FIGURE 10: DEFAULT VALUES ASSIGNED

13. Click the **SetCustomPropertyValue** button. This action assigns the value of CP2 to **CP1** from Tank1_001 (Figure 11 below).

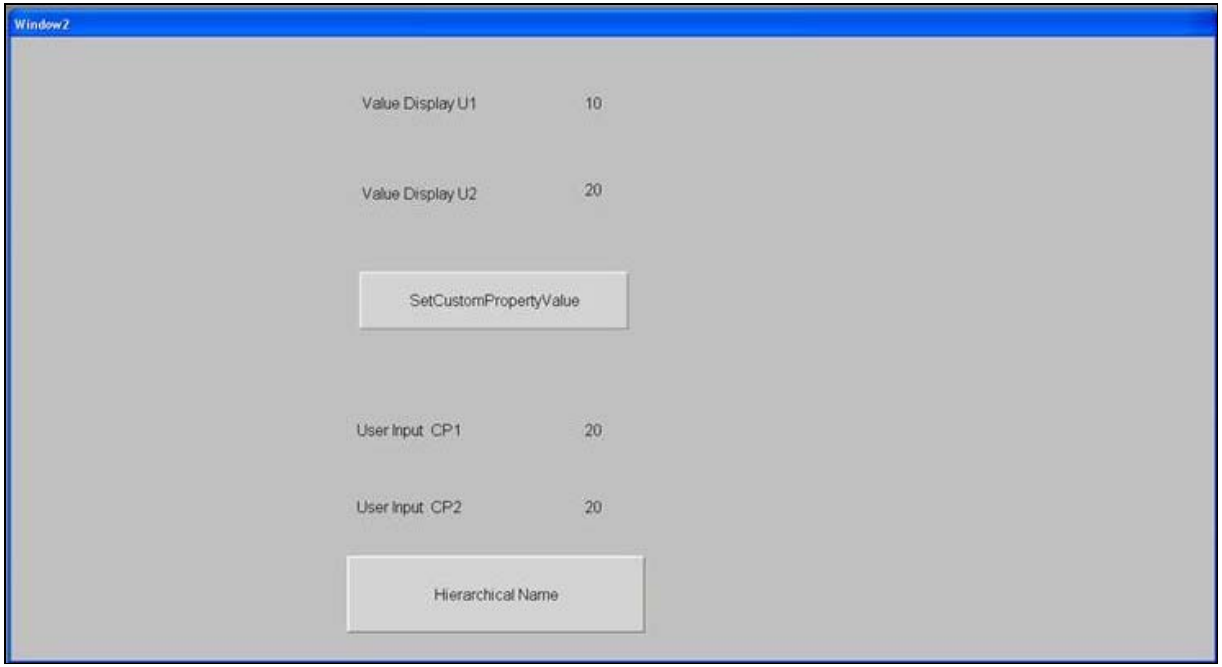

FIGURE 11: VALUES ASSIGNED TO INT1, INT2

14. Open ObjectViewer and change the value of **me.U2** in Tank1_002 to **200**. The value changes in WindowViewer for the CP1 and CP2 (Figure 12 below) because

now the Tank1_001 is changed to point to Tank1_002s as it is using the functionality of the OwningObject.

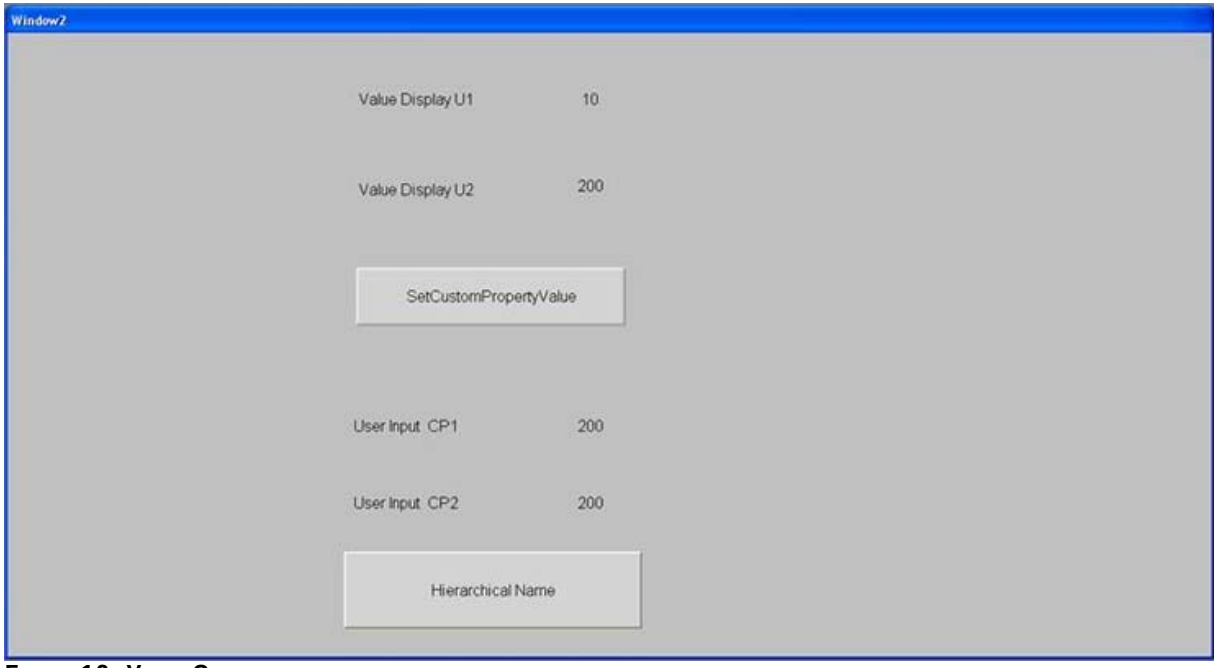

FIGURE 12: VALUE CHANGES

Advanced Techniques for Using SetCustomPropertyValue()

This section shows how to successfully implement the new AASymbols property called SetCustomPropertyValue into an InTouch Managed Application which can be then be made a Deployed or Published application.

The goal is to build a dynamic object that calls another AA symbol (show Symbol animation) in Runtime, and set the custom properties to point to different InTouch tags.

Using this technique will save development time during Application design, and dramatically reduce the number of pop up windows needed to achieve the same result using old methods. You get extra value when you need to change a particular element in the symbol, because a single change is propagated to every symbol instance.

Building the Application

1. Create a new derived InTouchViewApp and add the following tags to the Tagname Dictionary:

Memory Discrete For Status of each Valve	**Memory Discrete** Sets AUTO mode for each Valve	**Memory Discrete** Sets MAN mode for each Valve	**Memory Message** Pass the name of the Valve selected in Runtime
			Valve_Name
Valve1_Status	Valve1_Auto	Valve1_Man	Valve1
Valve2_Status	Valve2_Auto	Valve2_Man	Valve2
Valve3_Status	Valve3_Auto	Valve3_Man	Valve3
Valve4_Status	Valve4_Auto	Valve4_Man	Valve4

2. For each valve name above e.g Valve1 add the initial value as its tagname.
3. Create a new toolset under the Graphic ToolBox called **SetCustomProperty_Symbols**.

Create the Application Symbols

Valve Symbol

1. Create a Valve Symbol called **Valve**.
2. Configure its custom properties as shown below:

Custom Property Name	Data Type
Auto	Boolean
Man	Boolean
Valve_Name	String
Valve_Status	Boolean

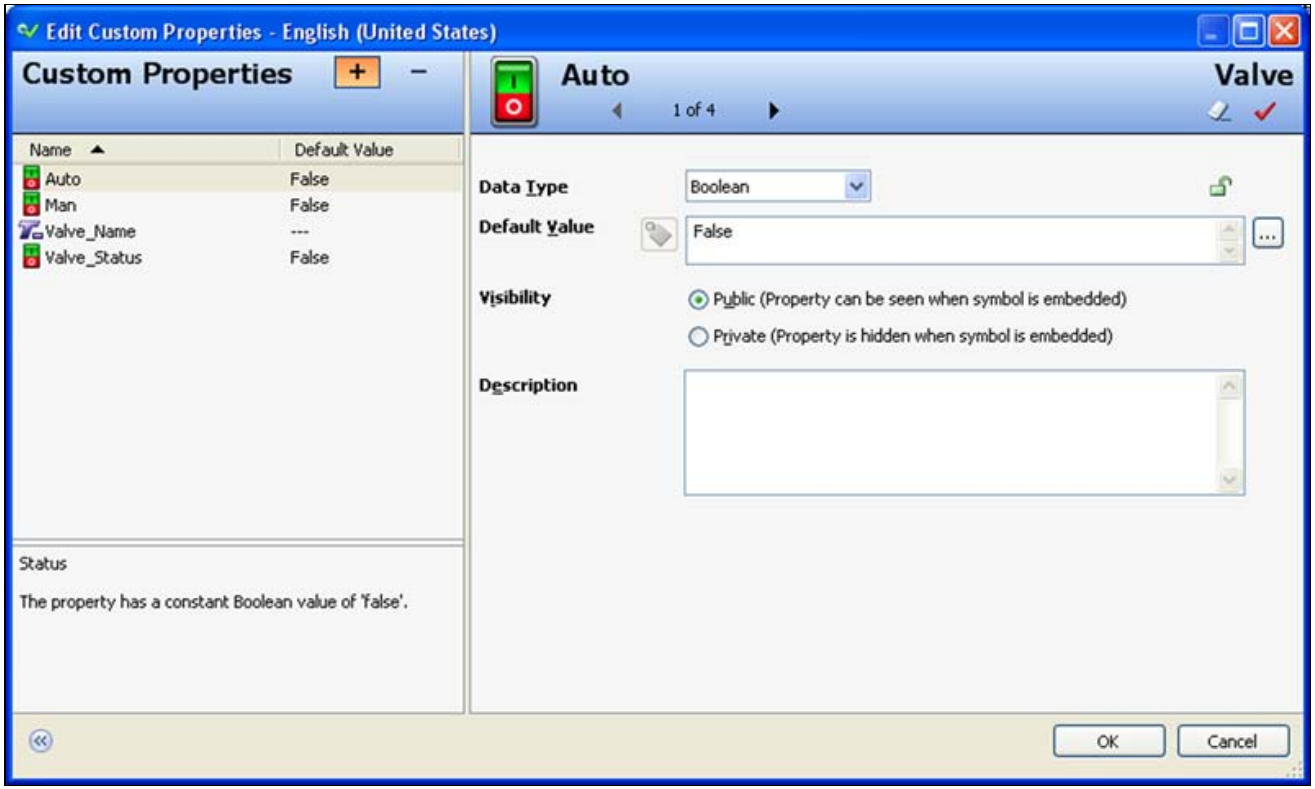

Figure 12: Edit Valve Symbol Custom Properties

6. Click **OK**.

   Use Figure 13 (below) for orientation in steps 7 - 11.

7. Draw a Rectangle.
8. Draw two horizontal lines.
9. Draw one vertical line that connects the two horizontal lines.
10. Draw one more vertical line at the below center.
11. Draw three text boxes (two at the bottom and one on top).

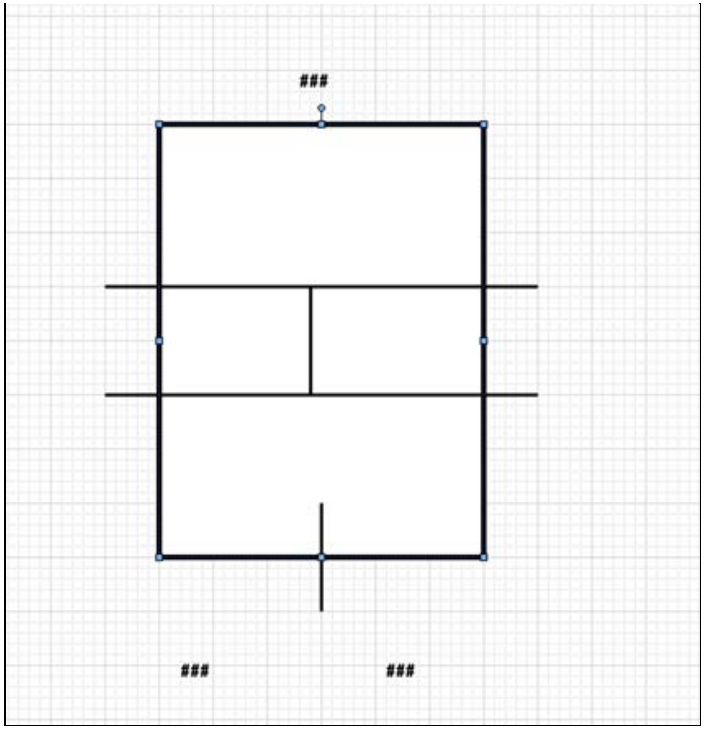

FIGURE 13: VALVE GRAPHIC SYMBOL

12. Save and Close the Valve graphic symbol.

Man_Auto Symbol

1. Duplicate the **Switch3PositionBlack** switch from the ArchestrA Symbol Library. You have a symbol called **Switch3PositionBlack_Copy1**.
2. Move **Switch3PositionBlack_Copy1** to the **SetCustomProperty Symbols** toolset.
3. Rename **Switch3PositionBlack_Copy1** to **Man_Auto**.
4. Add three more custom properties by right-clicking the embedded **Man_Auto** switch as shown in the followng table.
5. Click **OK** when you are finished.

Custom Property Name	Data Type
MyReferenceAuto	String
MyReferenceMan	String
Counter	Integer

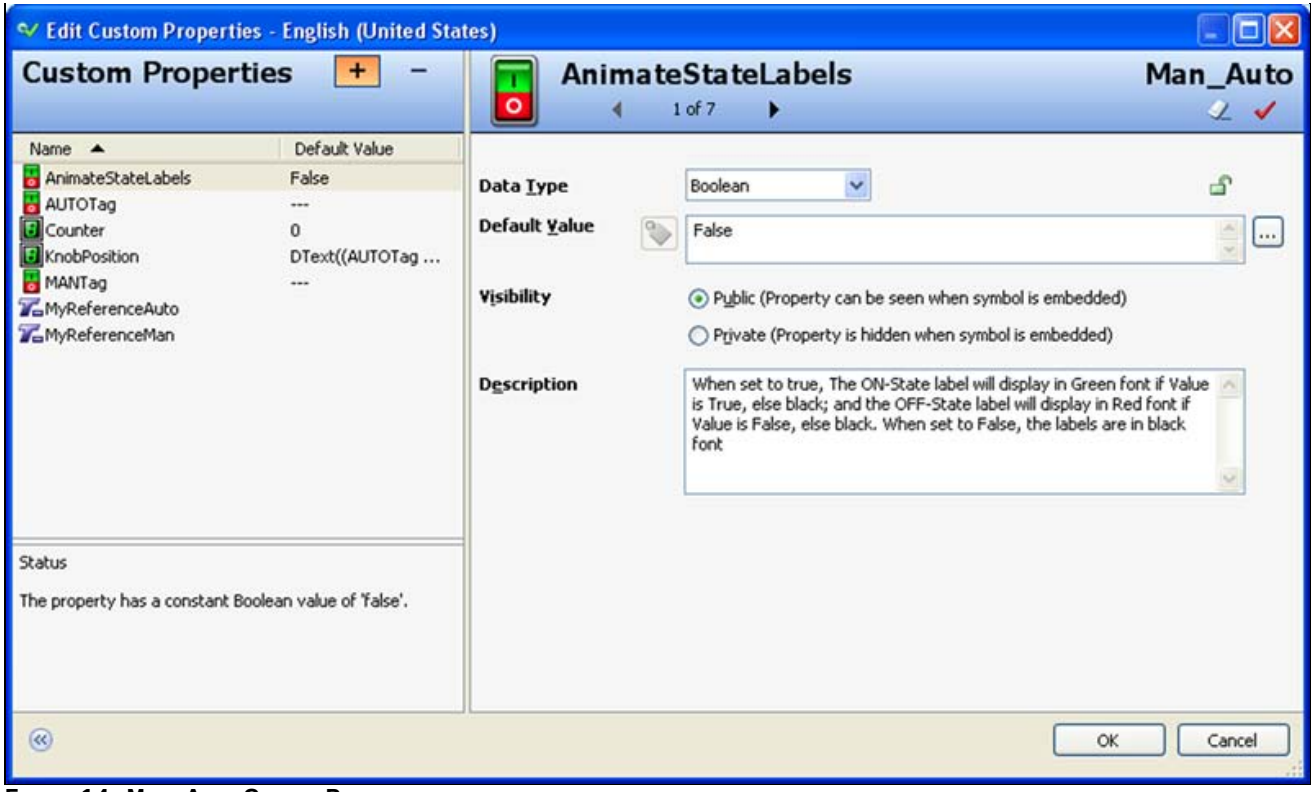

FIGURE 14: MAN_AUTO CUSTOM PROPERTIES

6. Add a **While Showing** script as shown below.

   The Script builds the referenced tag, then changes the corresponding custom property using the SetCustomPropertyValue function.
   You can use this technique build a Symbol that can open another symbol. The open symbol then refers to the opening symbol automatically.

```
if Counter < 2
then
   MyReferenceAuto = "Intouch:" + Intouch:Valve_Name.value + "_AUTO";
   SetCustomPropertyValue("AUTOTag", MyReferenceAuto, 0 );
   MyReferenceMan = "Intouch:" + Intouch:Valve_Name.value + "_MAN";
   SetCustomPropertyValue("MANTag", MyReferenceMan, 0 );
   Counter = Counter + 1;
Endif;
```

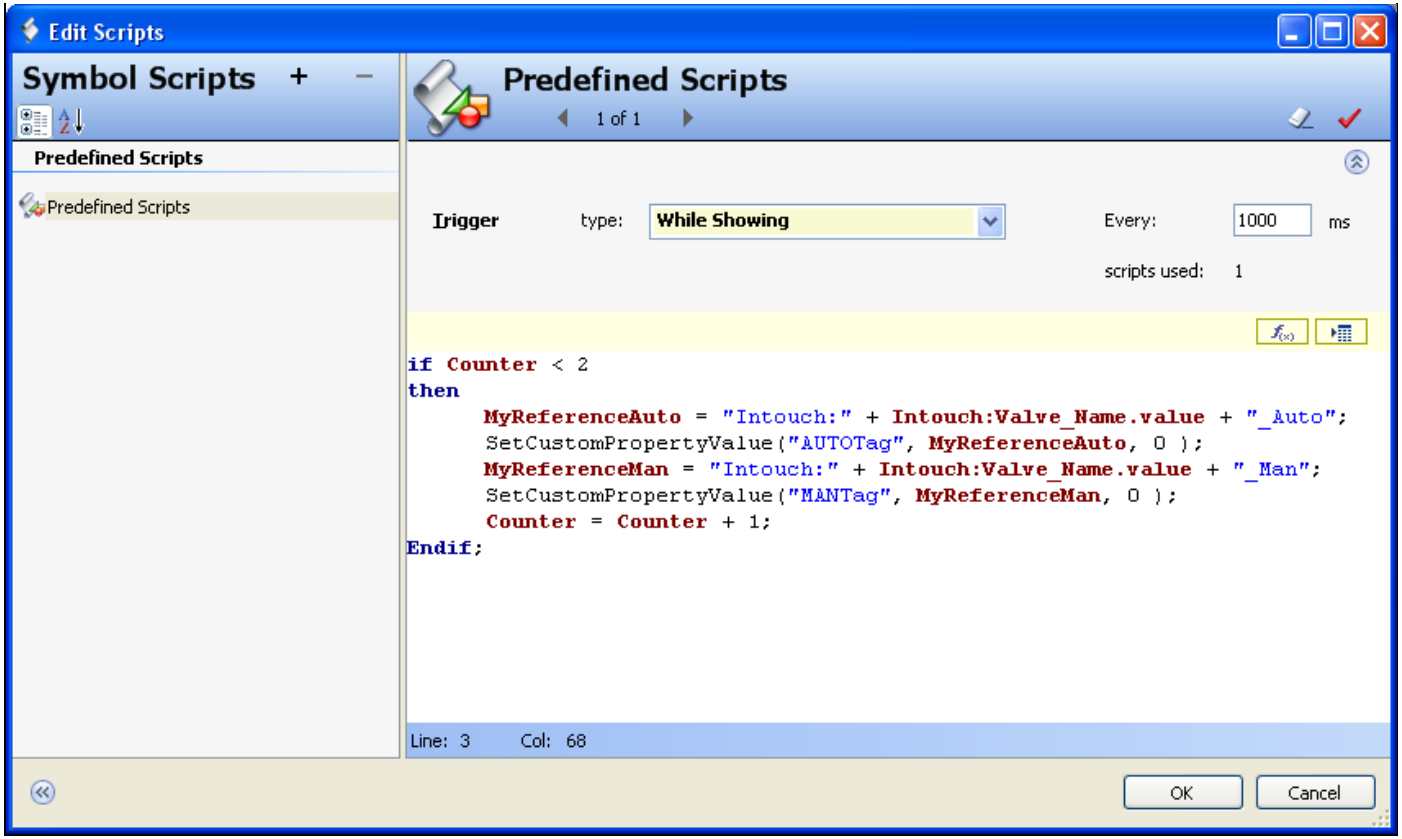

FIGURE 15: SYMBOL SCRIPTS

7. Click **OK** and return to our Valve Symbol and add the following:
8. Double-click on the Rectangle and add the following animations:

   • **Action Script:**

   ```
   Intouch:Valve_Name.value = Valve_Name;
   ```

   When the Valve is clicked the name is written into an InTouch tag called **Valve_Name**.

   • **Show Symbol**: Point to the **Man_Auto** symbol created above, and under the Title Bar assign the custom property called **Valve_Name**.
   • **Value Display** animation to left text box and point it to **Automatic**.
   • **Value Display** animation to left text box and point it to **Manual**.
   • **Value Display** animation to left text box and point it to **Value_Name**.

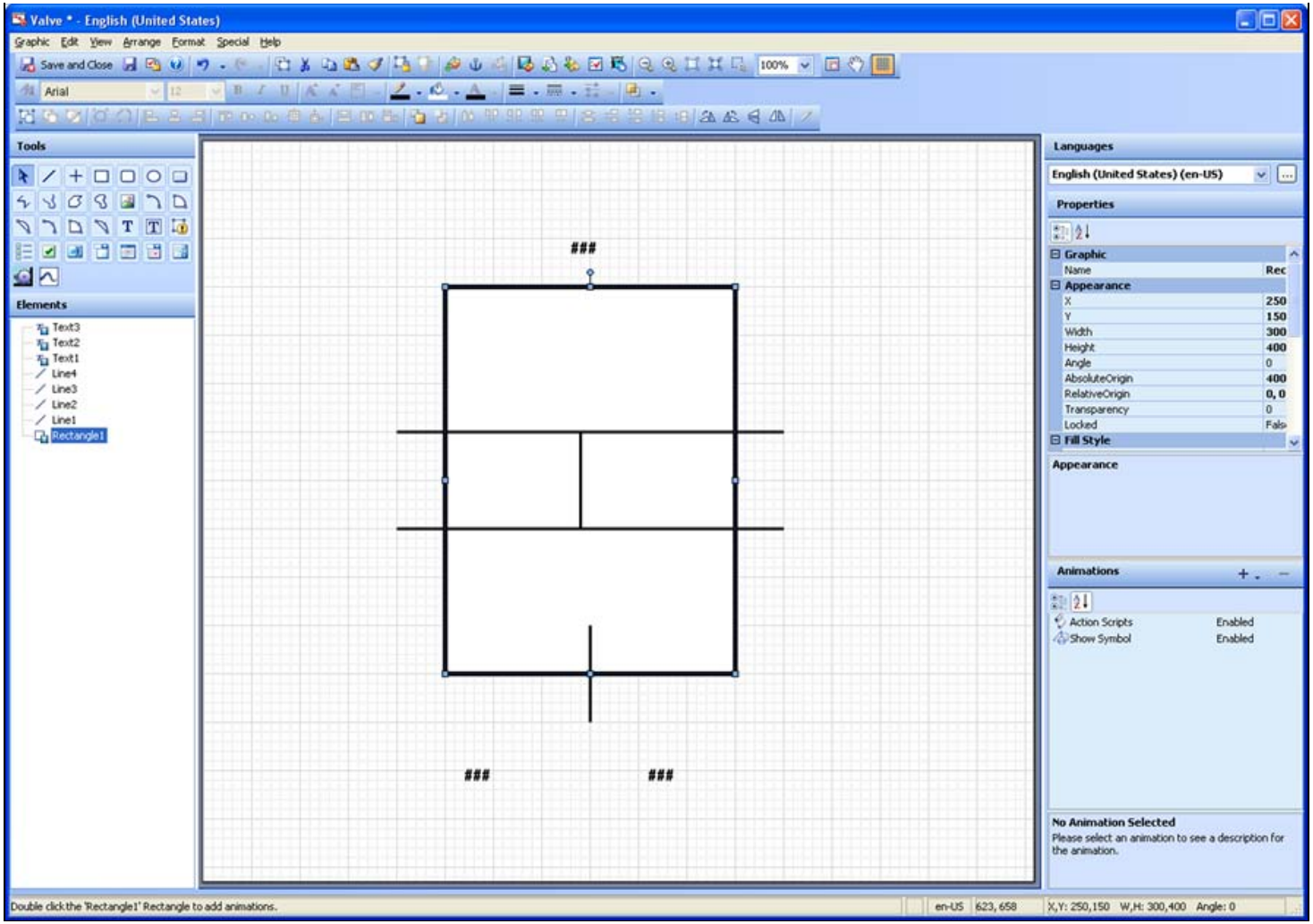

Figure 16: Add Custom Properties to Symbols

Label Symbol

1. Duplicate the **ButtonGlossyOrange** under switches from the ArchestrA Symbol Library. You have a symbol called **ButtonGlossyOrange_Copy1**.
2. Move **ButtonGlossyOrange_Copy1** to the **SetCustomProperty Symbols** toolset.
3. Change **ButtonGlossyOrange_Copy1** to **Label**.
4. Add a custom property called **LabelName** as a String data type (Figure 17 below).

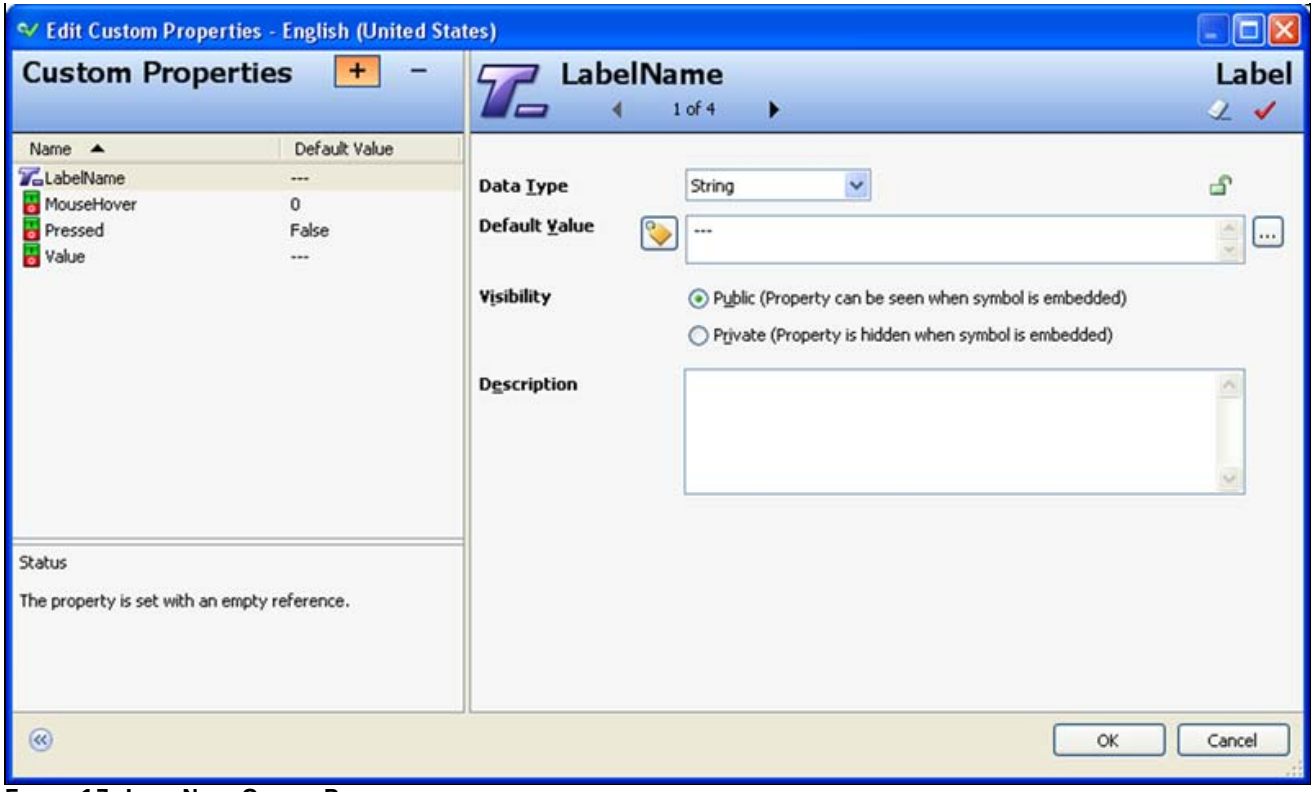

FIGURE 17: LABELNAME CUSTOM PROPERTY

All ArchestrA Symbols are now ready to use.

1. Open the InTouchViewApplication created above.
2. Create a new Window called **SetCustomProperty Window**.
3. Embed the Valve symbol 4 times on the above window.
4. Assign the InTouch tags created above to each Valve symbol (Figure 18 below).

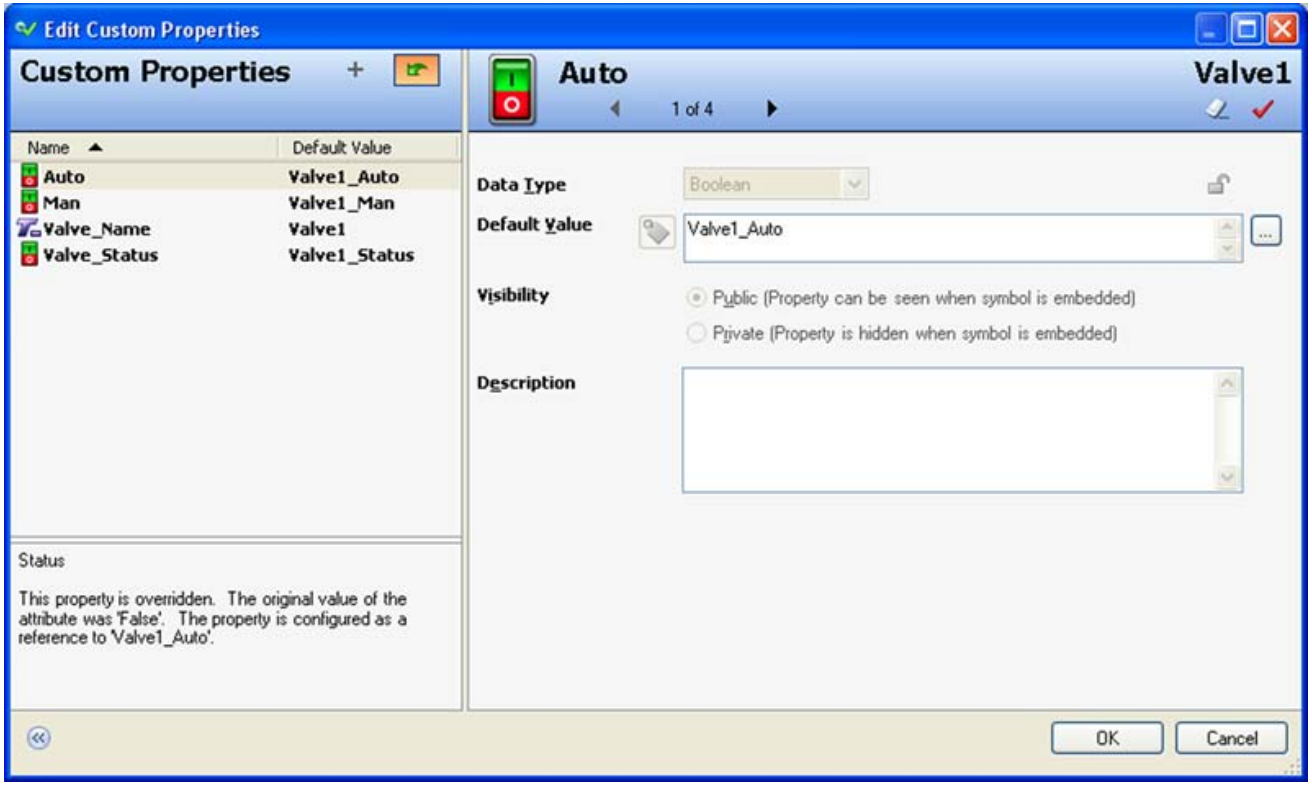

FIGURE 18: INTOUCH TAGS IN EACH SYMBOL

5. Embed the **Label** symbol.
6. Figure 19 (below) shows the WindowMaker™ display.

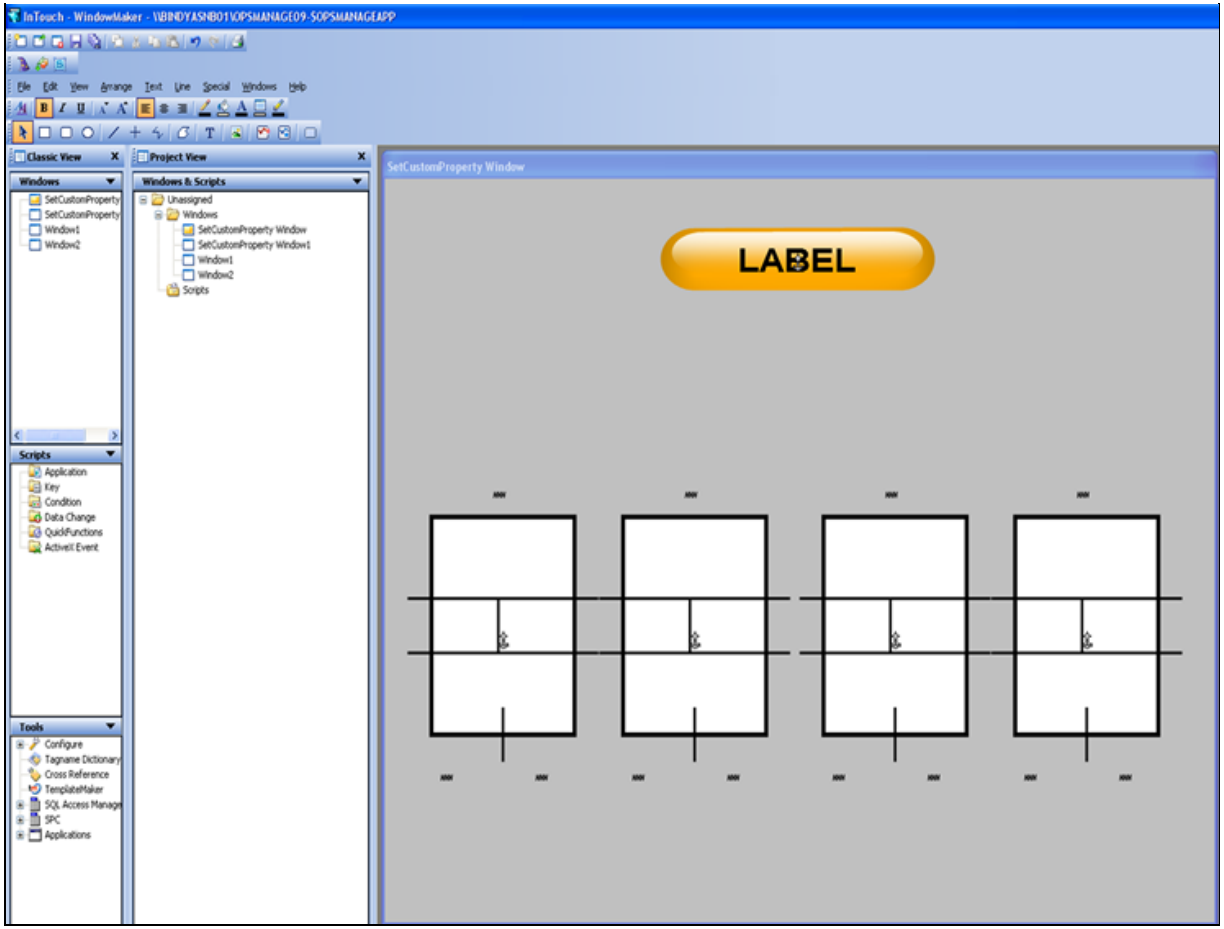

FIGURE 19: FOUR SYMBOLS IN WINDOWMAKER

7. Switch to runtime and click **Valve1** as (Figure 20 below).

   Notice the **Man_Auto** symbol appears since we have used Show Symbol.

8. Click **Automatic** or **Manual**. The Automatic display is shown in Figure 20 (below). **SetCustomProperty** value is working.

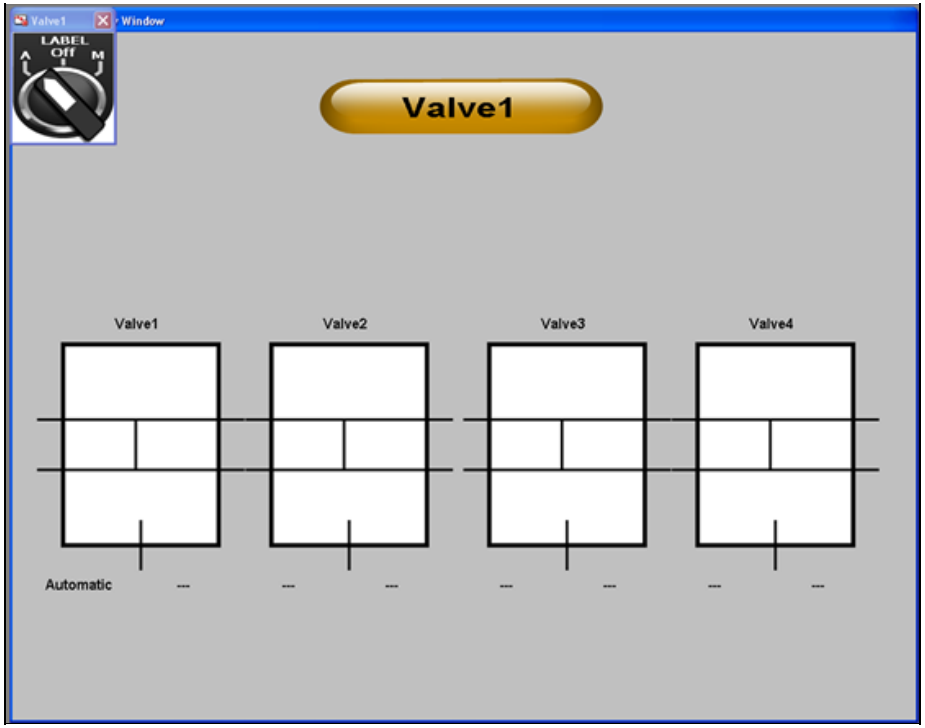

FIGURE 20: AUTOMATIC MODE IN RUNTIME

9. Click Valve2 and you see the following display:

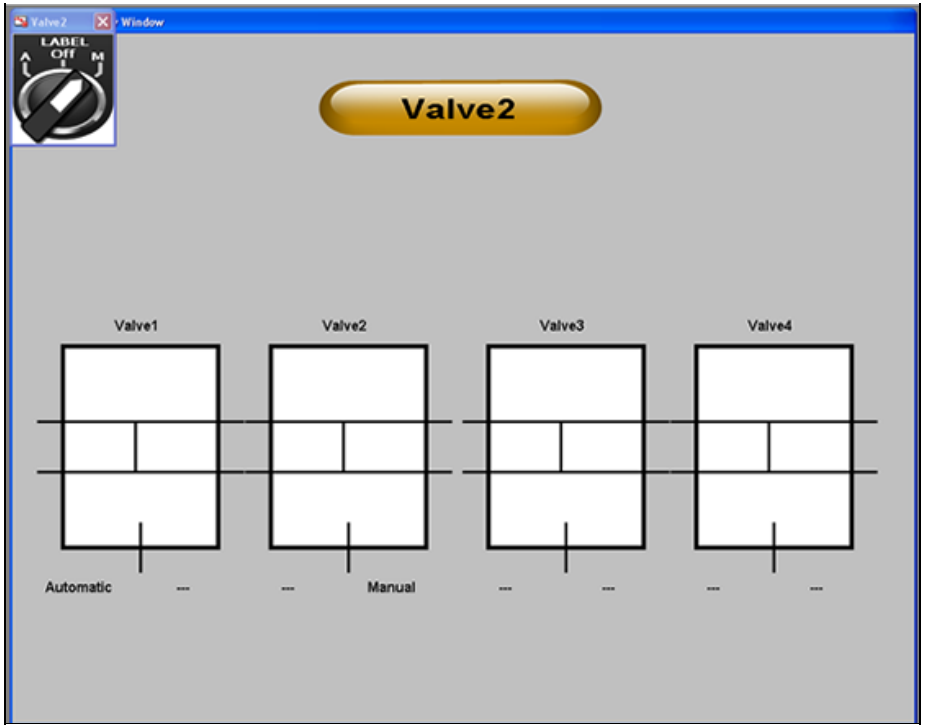

FIGURE 21: VALVE2 WITH MANUAL SETTING

B. Shah & S. Vatrano

Tech Notes are published occasionally by Wonderware Technical Support. Publisher: Invensys Systems, Inc., 26561 Rancho Parkway South, Lake Forest, CA 92630. There is also technical information on our software products at **Wonderware Technical Support.**

For technical support questions, send an e-mail to **support@wonderware.com**.

Back to top