GraphWorX32 – Local Aliasing and Tag Indirection

APPLICATIONS NOTE

April 2010

Description: Guide to set local aliasing and tag indirection in GraphWorX32.
OS Requirement: Win 2000, XP Pro, Server 2003, Vista, Server 2008, Windows 7
General Requirement: GraphWorX32 must be installed

Introduction

Aliasing is the ability to define a string to represent another string or number. This feature is very powerful and it will significantly reduce development time and increase reusability.

Because OPC tags are often very similar in syntax, and only differ in one or two places or numbers, it would drastically reduce development time if we could create just one process point and have it switch which tag it displays with the push of a button. This is where GraphWorX32 local aliasing and tag indirection is very useful.

For example, on the factory floor there are several actuators, all of the same type with similar I/O. If a detail window were to be created for each actuator, the system integrator would have to create one display per actuator, resulting in many files that essentially contain the same information, but have different OPC Tags. GraphWorX32 allows you to re-use one display and associate it with different OPC Tags using Aliasing. This feature reduces the cost of maintenance on your application; you will have fewer files to maintain and update when changes have to be made.

Displaying Local Aliases as Strings

The "**Edit Local Aliases**" command can be found on the Dynamics Menu in GraphWorX32. When including a local alias name in a data connection, the alias name should be surrounded by the special tokens << and >>. These tokens are used to identify the beginning and end of the alias name.

For this applications note we will use the following OPC Tags as an example:

ICONICS.Simulator\SimulatePLC.Ramp
ICONICS.Simulator\SimulatePLC.Sine
ICONICS.Simulator\SimulatePLC.Random

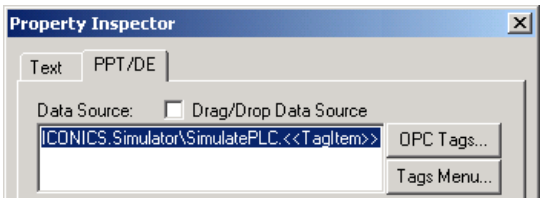

Figure 1 - Example GraphWorX32 Aliased Process Point

In the example in Figure 1, the alias name is TagItem. In order to resolve the alias:

1. Create a process point and type in **$"<<TagItem>>"$** as the Data Source
2. Select String for the client display Type and click OK to exit the properties inspector.

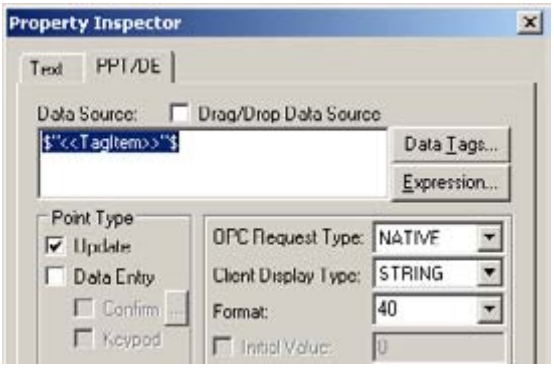

Figure 2 - Process Point Property Inspector

3. Create a push button and select "**Set Local Aliases**" as the Action.
4. Click on the "**Set Aliases…**" button.
5. This will bring up the Local Aliases Configuration.
6. If you do not remember the name of the alias, click on the "**Aliases…**" button in the dialog, and a list of aliases in the display will be shown.
7. Select an alias name and hit OK.
8. Specify "Ramp" in the Alias Definition field.
9. Click on "**Add**" to create the alias.

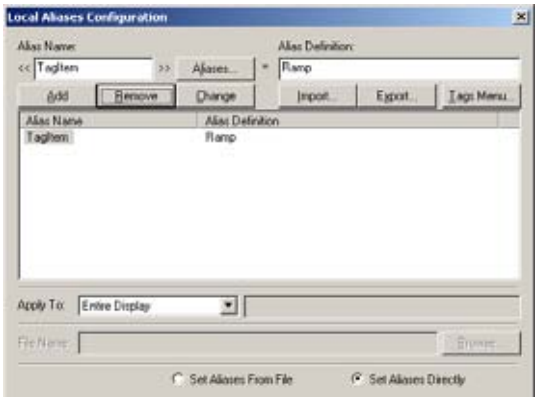

Figure 3 - Local Aliases Configuration Window

10. Repeat steps 4-8 setting the alias value to be "Sine" and "Random" tag.
11. Go into runtime and press the buttons one by one and you will notice the values changing accordingly.

Displaying Local Aliases as OPC Values

Displaying an OPC value using local alias has the same set up as displaying a text string with as local alias value except for one tiny difference. Notice that in the Data Source we have entered **$"<<TagItem>>"$** instead of just the local alias **<<TagItme>>**; this is because in order to resolve string, we needed to enclose the special tokens **$"** and **"$**. To display an OPC value, all we need is to enter the alias without the separators.

In the case that you want to display the simulated Ramp tag, you would set the alias value to be: "ICONICS.Simulator\SimulatePLC.Ramp". In runtime, the alias would resolve to display the value of the Ramp Tag.

Aliases can be used in any combination for an OPC Tag. For example:

<<Vendor>>.<<OPCServer>>\<<Device>>.<<TagItem>>

In this example all four aliases need to be resolved before the process point will show any data. GraphWorX32 can assign many aliases to a single object. You could set the aliases for the example above to be:

<<Vendor>>	= ICONICS
<<OPCServer>>	= Simulator
<<Device>>	= SimulatePLC
<<TagItem>>	= Ramp

Defining Alias Values using a Text File

GraphWorX32 supports aliasing with the use of alias files to even further reduce the amount of development time. These files can be used for different displays. Instead of configuring the aliases in each display, you can specify an alias file that will automatically resolve the aliases for that display.

1. In GraphWorX32 configuration, go to Tools →Local Alias File Editor and specify the alias involved.
2. Click on the Export button to export the Alias information to a standard text file.
3. Using a text editor like Notepad, the file needs to comply with the following syntax:

   aliasName<TAB>aliasDefinition<CR>

4. Specify the aliasName then press the Tab key, followed by the aliasDefinition, and then the enter key. Repeat this for all aliases you want to include in the alias file. Figure 4 shows an example of how the text file would look like. Note that there is exactly one tab between the aliasName and the aliasDefinition, which is why the columns may or may not line up

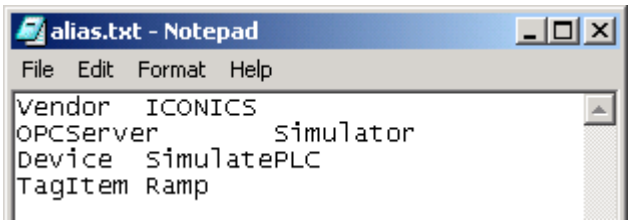

Figure 4 - Sample Alias File

Local Aliasing Using VBA

GraphWorX32 also has a Visual Basic automation interface that supports aliasing. Use the following automation calls in your GraphWorX32 display to create alias files.

Table 1 - Automation Calls for Aliases

Function	Description
SetAliasDefinition (aliasName As String, newDefinition As String) As Boolean	Sets the alias definition of the specified alias name for all dynamic objects in the display
SetAliases (commandString As String) as Boolean	Sets multiple alias definitions for the entire display. The commandString parameter has the following format: aliasName1 Chr(9) aliasDef1 Chr(13) Chr(10) aliasName2 Chr(9) aliasDef2 Chr(13) Chr(10) … etc.
Function SetAliasesFromFile (filename As String) As Boolean	Sets multiple alias definitions for the entire display. Using the same file as format as specified above