OPC with Visual Studio

Hans-Petter Halvorsen, M.Sc.
Software

• MatrikonOPC Simulation Server
• Visual Studio
• Measurement Studio
  – Add-on package to Visual Studio created by National Instruments (same vendor as LabVIEW)
Measurement Studio

• Add-on package to Visual Studio created by National Instruments
• Same vendor as LabVIEW
• Makes it possible to communicate with an OPC DA Server from Visual Studio Code
• Uses the DataSocket Library (same as in LabVIEW)
What is OPC?

- A standard that defines the communication of data between devices from different manufactures
- Requires an **OPC server** that communicates with the **OPC clients**
- OPC allows “plug-and-play”, gives benefits as reduces installation time and the opportunity to choose products from different manufactures
- Different standards: “Real-time” data (**OPC DA**), Historical data (**OPC HDA**), Alarm & Event data (**OPC AE**), etc.
OPC

OPC Server
Data Storage

OPC Client

Read/Write Data
OPC Server

Data Storage

Write Data

OPC Client

Read Data

OPC Client

Monitoring and Analysis

Measurement Data
OPC - Server

Network

OPC-Client

PLC, PAC, DCS, SCADA

Data Acquisition

Sensors

Actuators

OPC-Server

Typical OPC Scenario

Process Data

Driver

Process

Typical OPC Scenario

OPC-Client

OPC-Client
OPC Specifications

“Classic” OPC

OPC DA
OPC HDA
OPC A&E

... (Many others)

“Next Generation” OPC

OPC UA
OPC Specifications

• **OPC DA** (Data Access)
The most common OPC specification is OPC DA, which is used to read and write “real-time” data. When vendors refer to OPC generically, they typically mean OPC DA.

• **OPC HDA** (Historical Data Access)

• **OPC A & E** (Alarms & Events)
  • ... (many others)

These OPC specification are based on the OLE, COM, and DCOM technologies developed by Microsoft for the Microsoft Windows operating system family. This makes it complicated to make it work in a modern Network! Typically you need a Tunneller Software in order to share the OPC data in a network (between OPC Servers and Clients)

• **OPC UA** (Unified Architecture)

OPC UA eliminating the need to use a Microsoft Windows based platform of earlier OPC versions. OPC UA combines the functionality of the existing OPC interfaces with new technologies such as XML and Web Services (HTTP, SOAP)
Next Generation OPC

OPC Classic
- COM/DCOM
- OPC DA
- OPC HDA
- OPC A&E
- Windows only

Next Generation OPC
- OPC UA
- Cross-platform (Windows, Linux, Mac, Embedded, VxWorks)
- All specifications collected in one (DA, HDA, A&E)
- Protocols: “UA Binary” or “UA XML”
- Simpler!!

OPC Client (DCOM)
- OPC DA Server
- OPC HDA Server
- OPC A&E Server

OPC UA Server
- OPC UA Client
- (everything built into one)
Next Generation OPC

To open DCOM through firewalls demanded a large hole in the firewall! Impossible to route over Internet!

No hole in firewall (UA XML) or just a simple needlestick (UA Binary) is necessary. Easy to route over Internet!
Classic OPC requires a Microsoft Windows operating system to implement COM/DCOM server functionality. By utilizing SOA and Web Services, OPC UA is a platform-independent system that eliminates the previous dependency on a Windows operating system. By utilizing SOAP/XML over HTTP, OPC UA can deploy on a variety of embedded systems regardless of whether the system is a general purpose operating system, such as Windows, or a deterministic real-time operating system.

Matrikon OPC Explorer – Connect to Server

1. Connect
2. Add Tags
Matrikon OPC Explorer - Add Tags

1. Double-click on the desired item to add tags.
2. Double-click to add tags.
3. Select the bucket Brigade Real4 in the list.
4. Click on the tag to add it.
5. Finished.
MatrikonOPC Explorer (OPC Client)

The MatrikonOPC Explorer is useful for testing. You can use it for writing and reading OPC Tags.
Measurement Studio

Hans-Petter Halvorsen, M.Sc.
Measurment Studio

• Add-on package to Visual Studio created by National Instruments
• Same vendor as LabVIEW
• Makes it possible to communicate with an OPC DA Server from Visual Studio Code
• Uses the DataSocket Library (same as in LabVIEW)
Measurement Studio

- Measurement Studio is an add-on to Visual Studio.
- Measurement Studio is used for development of measurement, control and monitoring applications using .NET and Visual Studio.
- Measurement Studio has a library (DataSocket library) that makes it possible to communicate with OPC DA servers that we will use is this lab work
- Download Software here: http://www.ni.com/academic/download
Visual Studio Editions

- **I have Visual Studio 2013**
  - You can Install and use Measurements Studio 2015 without problems

- **I have Visual Studio 2015**
  - Measurement Studio 2015 is designed to work with Visual Studio 2013 (and older editions). Therefore, Measurement Studio 2015 does not install shipping examples and does not integrate with Visual Studio 2015.
  - This means, if you install Measurement Studio 2015 with Visual Studio 2015, the Measurement Studio .NET controls are not in the Toolbox, and you do not have a Measurement Studio menu item in the Visual Studio 2015 toolbar.
  - Follow these guidelines to do this manually:
Measurement Studio 2015 does not have integration features for Visual Studio 2015. If you install Measurement Studio 2015 with Visual Studio 2015, the Measurement Studio .NET controls are not in the Toolbox, you do not have a Measurement Studio menu item in the Visual Studio 2015 toolbar, and .licx will not be automatically generated. We have plans to make changes to Measurement Studio that will help us better keep up with new versions. Unfortunately, these changes are a quite a bit more costly than it would seem, particularly the Visual Studio Help integration, so these changes may not come soon.

This situation is the same as previous Measurement Studio software as in the following article 'Using Measurement Studio 2013 with Microsoft Visual Studio 2013': [http://digital.ni.com/public.nsf/allkb/C51E3B38578FAD2786257C070069F386](http://digital.ni.com/public.nsf/allkb/C51E3B38578FAD2786257C070069F386)

Visual Studio 2015 is not supported officially in Measurement Studio 2015; however, as in the above article, you can add the Measurement Studio .NET controls to the Toolbox manually and can create .licx files manually. I have attached a Help Document on this topic so you can refer to the Adding Measurement Studio 2015 User Interface Controls to the Toolbox section for more information on How to Add controls. This section also describes how these controls are licensed. The following sections discuss additional topics to consider when using Measurement Studio 2015 with Visual Studio 2015.

Rebecca Costin
National Instruments
Applications Engineering
www.ni.com/support
Select “New Project” in Visual Studio:

1. Make sure to select the DataSocket Library

2. Select the “NI Windows Application” Template

3. Note! You don’t need to use the “Measurement Studio Templates” – You can use an ordinary WinForm App also!!

Visual Studio 2015 + Measurement Studio

You can use an ordinary WinForm App

Then you need to:
1. Add References (Assemblies) Manually
2. Change/Update the License File (Licenses.licx)
Adding References to your Project

You need to add these Assemblies (.dll files) to your Visual Studio Project:

Locate the following Assemblies on your harddrive:

- NationalInstruments.Common.dll
- NationalInstruments.Net.dll

C:\Program Files...\National Instruments\MeasurementStudioVSXXXX\DotNET\Assemblies\Current\...
License File

You may need to update the License File with correct Version Number and PublicKeyToken according to the installed Assemblies you have on your harddrive.

OPC Read Example

Hans-Petter Halvorsen, M.Sc.
OPC Read

Matrikon OPC Server/OPC Explorer

Visual Studio App

Click Button to get latest Value from OPC Server
using NationalInstruments;
using NationalInstruments.Net;

string opcUrl;
double opcValue;

opcUrl = "opc://localhost/Matrikon.OPC.Simulation/Bucket Brigade.Real4";

DataSocket dataSocket = new DataSocket();

if (dataSocket.IsConnected)
    dataSocket.Disconnect();

dataSocket.Connect(opcUrl, AccessMode.Read);
dataSocket.Update();
opcValue = Convert.ToDouble(dataSocket.Data.Value);

Note! This Code Snippet reads only one value once, you can use e.g. a Timer in order to read values at specific intervals.
DEMO
using NationalInstruments;
using NationalInstruments.Net;
using System;
using System.Windows.Forms;

namespace OPC_Read
{
    public partial class Form1 : Form
    {
        DataSocket dataSocket = new DataSocket();

        public Form1()
        {
            InitializeComponent();

            string opcUrl;
            opcUrl = "opc://localhost/MATRIKON.OPC.Simulation/Bucket_Brigade.Real4";

            if (dataSocket.IsConnected)
            {
                dataSocket.Disconnect();
            }

            dataSocket.Connect(opcUrl, AccessMode.Read);
        }

        private void btnReadOpc_Click(object sender, EventArgs e)
        {
            dataSocket.Update();

        }
    }
}
OPC Write Example

Hans-Petter Halvorsen, M.Sc.
Write OPC Example

Matrikon OPC Server/OPC Explorer

Visual Studio App
using National Instruments;
using National Instruments.Net;
...
string opcUrl;
double opcValue;

opcUrl = "opc://localhost/Matrikon.OPC.Simulation/Bucket Brigade.Real4";

DataSocket dataSocket = new DataSocket();

if (dataSocket.IsConnected)
    dataSocket.Disconnect();

dataSocket.Connect(opcUrl, AccessMode.Write);

opcValue = Convert.ToDouble(txtWriteOpcValue.Text);
dataSocket.Data.Value = opcValue;
dataSocket.Update();
DEMO
using NationalInstruments;
using NationalInstruments.Net;
using System;
using System.Windows.Forms;

namespace OPC_Write
{
    public partial class Form1 : Form
    {
        DataSocket dataSocket = new DataSocket();

        public Form1()
        {
            InitializeComponent();

            string opcUrl;
            opcUrl = "opc://localhost/MATRIKON.OPC.Simulation/Bucket Brigade.Real4";

            if (dataSocket.IsConnected)
                dataSocket.Disconnect();

            dataSocket.Connect(opcUrl, AccessMode.Write);
        }

        private void btnWriteOpc_Click(object sender, EventArgs e)
        {
            double opcValue = 0;

            opcValue = Convert.ToDouble(txtWriteOpcValue.Text);

            dataSocket.Data.Value = opcValue;

            dataSocket.Update();
        }
    }
}
OPC Write/Read Example

Hans-Petter Halvorsen, M.Sc.
Write/Read Example

OPC Example

22
Write OPC

22
Read OPC

Contents of ‘Group0’

Item ID | Access Path | Value | Quality | Timestamp | Status
--- | --- | --- | --- | --- | ---
Bucket Brigade.Real4 | 22 | Good, non-specific | 01.29.2016 3:48:04.904 PM | Active

Server Info

Server: Matrikon.OPC.Simulation.1
Connected: Yes
State: Running
Groups: 1
Total Items: 1
Current Local Time: 01.29.2016 3:48:04.904 PM
Update Local Time: 01.29.2016 3:48:04.995 PM

Did you know?
Explorer Tip #2
You can configure any MatrikonOPC Server from OPC Explorer.

MatrikonOPC
using System;
using System.Windows.Forms;
using NationalInstruments;
using NationalInstruments.Net;

namespace OPCExample
{
    public partial class Form1 : Form
    {
        DataSocket dataSocketRead = new DataSocket();
        DataSocket dataSocketWrite = new DataSocket();

        public Form1()
        {
            InitializeComponent();
            string opcUrl;
            opcUrl = "opc://localhost/MATRIKON.OPC.Simulation/Bucket Brigade.Real4";

            if (dataSocketRead.IsConnected)
                dataSocketRead.Disconnect();

            dataSocketRead.Connect(opcUrl, AccessMode.Read);

            if (dataSocketWrite.IsConnected)
                dataSocketWrite.Disconnect();

            dataSocketWrite.Connect(opcUrl, AccessMode.Write);
        }

        private void btnReadOpc_Click(object sender, EventArgs e)
        {
            dataSocketRead.Update();
            txtReadOpcValue.Text = dataSocketRead.Data.Value.ToString();
        }

        private void btnWriteOpc_Click(object sender, EventArgs e)
        {
            double opcValue = 0;
            opcValue = Convert.ToDouble(txtWriteOpcValue.Text);
            dataSocketWrite.Data.Value = opcValue;
            dataSocketWrite.Update();
        }
    }
}
Additional Features

Hans-Petter Halvorsen, M.Sc.
Additional Features

• Using a Timer in order to read/write Data from/to the OPC Server at specific Intervals

• Trending/Plotting Data
  – Using the ”WaveformGraph” Control included with Measurement Studio

• Using OOP, i.e., Create and Use Classes in your Code
In Visual Studio you may want to use a Timer instead of a While Loop in order to read values at specific intervals.

1. Select the “Timer” component in the Toolbox

2. Initialization:

   ```csharp
   public Form1()
   {
       InitializeComponent();
       timer1.Start();
   }
   ```

3. Properties:

   - Name: timer1
   - Enabled: False
   - GenerateMember: True
   - Interval: 100
   - Modifiers: Private

   You may specify the Timer Interval in the Properties Window.

4. Timer Event:

   ```csharp
   private void timer1_Tick(object sender, EventArgs e)
   {
       ... //Read from OPC
       ... //Scaling
       ... //Plot Data
   }
   ```

   Structure your Code properly!! Define Classes and Methods which you can use here.
Trending Data with Measurement Studio

You may use the “WaveformGraph” Control included with Measurement Studio

You only need one line of code, e.g. in the Timer Event:

```plaintext
... 
{
  ...
  waveformGraph.PlotYAppend(analogDataIn);
}
```

Name of your WaveformGraph Control  Name of the Method to use

Name of the variable with Temperature data
Example:

```csharp
class Car //Class
{
    public string color; //Field

    //Method
    public void ShowCarColor()
    {
        MessageBox.Show("My Car is: " + color);
    }
}

Car myCar = new Car(); //We create an Instance of the Class
myCar.color = "blue"; //We set a value for the color Field
myCar.ShowCarColor(); //We call the Method
```

You should use OOP techniques in your application! Create your own classes, fields and Methods.
Hans-Petter Halvorsen, M.Sc.

University College of Southeast Norway

www.usn.no

E-mail: hans.p.halvorsen@hit.no
Blog: http://home.hit.no/~hansha/