



Application Note 114

Excel/OPC VBA Macro

Introduction

The purpose of this application note is to describe by example how one can use the OPC Foundation's OPC Automation 2.0 Interface (OPCDAAuto.DLL) and Visual Basic for Applications to retrieve data from an OPC server. A general understanding of OPC, Microsoft Visual Basic, Microsoft Excel, and Windows is assumed.

Microsoft Excel 97/2K and its included VBA were used to develop the described macro which communicates using the OPC Automation 2.0 Interface with the Hach ModIO Explorer Demo OPC Server. The software principals' outlined in this application note can be applied to communicate with any OPC Server using Visual Basic or Visual Basic for Applications and the OPC Automation DLL, OPCDAAuto.DLL.

System Requirements

- A Pentium class PC with Microsoft Excel 97 or 2000.
- Hach ModIO Explorer Demo software or Hach ModIO Explorer Version 2.0 software and the demo configuration file.
- The Excel file OPCEXcel.XLS
- OPCDAAuto.DLL Version 2.0.02.
- COMDLG32.OCX Version 6.084.18.
- Internet Explorer 4.01 minimum
- Windows 98 with DCOM or Windows NT4.0 SP6, or Windows 2000

Sources

Refer to the Data Access Automation Interface Standard for detailed information on the OPC Automation Interface. The Data Access Automation Interface Standard Version 2.02 can be obtained from the OPC Foundation by ordering their CD (free) from their web site:

<http://www.opcfoundation.org/>

The Excel Macro, OPCEXcel.xls with COMDLG32.OCX, can be downloaded from the Hach Company AquaTrend web site as OPCEXcel.exe, a self-extracting, self-installing zip file:

<http://www.aquatrend.com/Downloads/Software/OPCEXcel.exe>

The Hach ModIO Explorer Demo software can be downloaded from the Hach Company AquaTrend web site:

<http://www.aquatrend.com/Downloads/Software/ModIO Explorer Demo Install.exe>

Installing and registering files

After downloading the OPCEXcel.exe file from the AquaTrend web site, double click on the file in Windows Explorer and follow the instructions. The install program will install the OPCEXcel.xls file in the folder C:\PROGRAM FILES\OPCEXcel\VBEX\.

The install program will also install and register the CM Dialog ActiveX Control, COMDLG32.OCX, in your System32 folder. A shortcut to the OPCEXcel.xls file will be placed in your start menu i.e. **Start|Programs|OPCEXcel\VBEX.**

Installing the Hach ModIO Explorer Demo will install and register the OPCDAAuto.DLL on your system and insure that you have a suitable version installed of Internet Explorer and DCOM.

The ModIO Explorer Demo

Note: If you already have the ModIO Explorer Version 2.0 software installed on your computer, please refer to **Appendix A** for instructions.

Warning: If you have a version of the ModIO Explorer prior to **2.0**, **do not install** the ModIO Explorer Demo software. Select a different PC to install this demo on.

Install the Hach Mod-IO Explorer Demo by double clicking on the **ModIO Explorer Demo Install.exe** file in Windows Explorer, and follow the installation instructions.

Start the Mod-IO Explorer (OPC Server) by clicking **Start|Programs|Hach|Mod-IO Explorer.**

The demo opens a configuration file that simulates 8, 1720D turbidimeters. The sensor signals either continually ramp, or are random depending on the sensor.

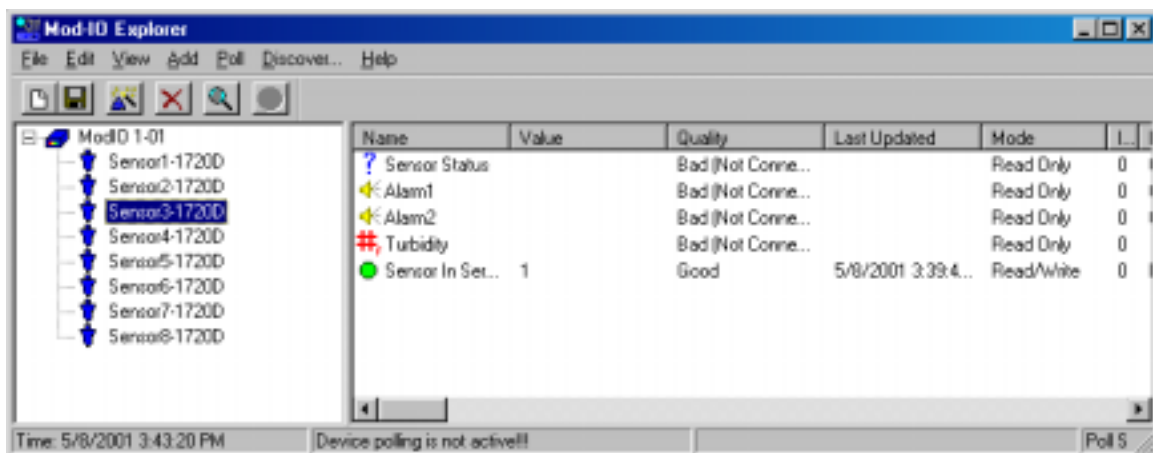


Figure 1: ModIO Explorer

Verify that the Mod-IO Explorer Demo is functioning properly by expanding the tree in the left hand pane and selecting Sensor3-1720D.

Select the **Spyglass** Icon on the Toolbar or **Poll|Start Polling** from the Mod-IO Explorer menu bar. The value of the Tag "Turbidity" in the right hand pane should ramp from 0.4 to 0.5 and its Quality should be "Good".

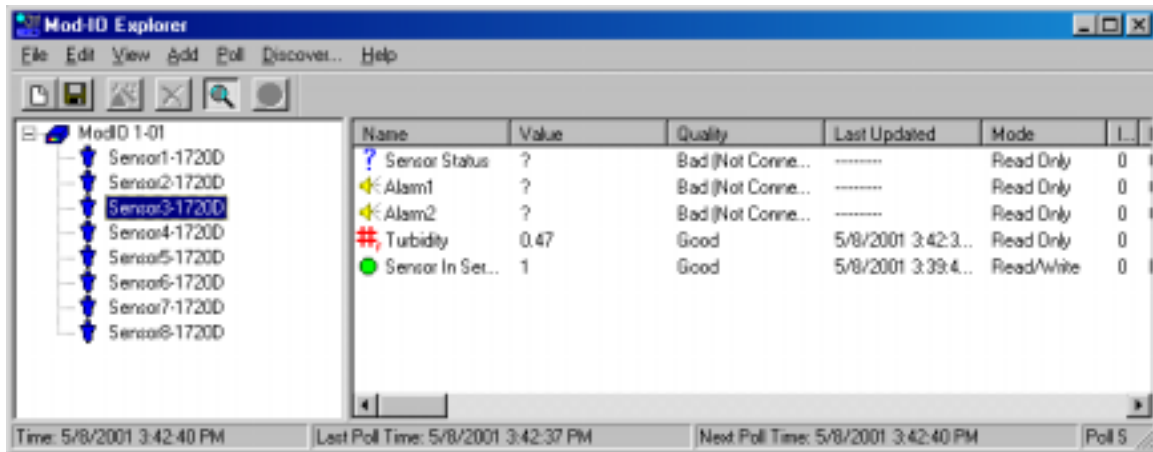


Figure 2: ModIO Explore Polling

Select **File|Exit** from the Mod-IO Explorer Demo menu bar to close the OPC server.

OPCExcel

Before opening the OPCExcel.xls Excel demo file, verify that your security settings for Macros will allow you to run an Excel Macro. For Excel 2000, open Excel and select **Tools|Macro|Security** from the menu bar.



Figure 3: Excel Security

On the Security Level tab, select Medium for the security level and click OK.

Open the OPCEXcel.xls file by selecting **Start|Programs|OPCEXcel|OPCEXcel.xls** or open it using **File|Open** from the Microsoft Excel menu bar, and open the file at **C:\PROGRAM FILES\OPCEXcel\OPCEXcel.xls**.

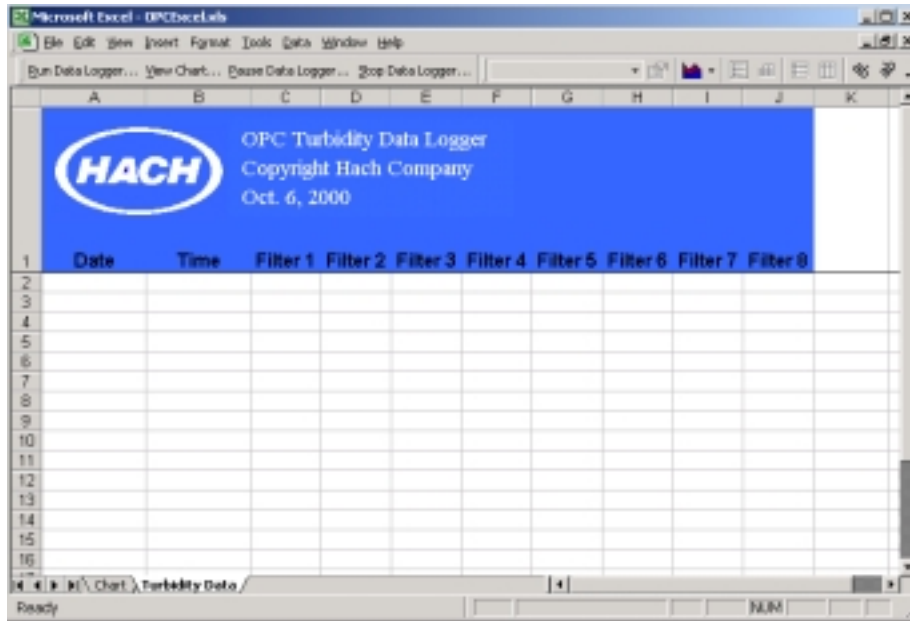


Figure 3: OPCEXcel.xls

The Excel workbook application uses a macro to communicate with the Mod-IO Explorer OPC server. This application can log turbidity data at a user-defined rate (sample interval in seconds). The number of turbidimeters can be 1 to 8 and the number of data records can be 1 to 65,000.

The application will log the requested turbidity data until either the Data Logger is stopped or the requested number of data records has been met. While data is being recorded, the Data Logger will display the data in tabular form on the Turbidity Data sheet and in graphical form on the Chart sheet. The turbidimeters that you want to view graphically are selected during the setup of the Data Logger application.

After the requested number of records has been met, the Data Logger will automatically calculate percentile statistics and report the second of two consecutive readings over 1 NTU for each turbidimeter. The reported statistics are displayed in the automatically created Report Sheet. The file is saved in a user-selected folder and has the file format of day-month-year (08-Nov-00.xls). Multiple files saved on the same day have an additional number (0 to 24) appended to the file name (08-Nov-00-0.xls).

To run the application, click on the Run Data Logger button in the OPC1720D tool bar.

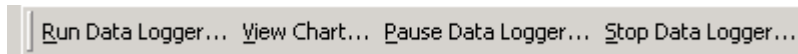


Figure 4: Tool Bar

Note: If the tool bar is not fully visible, maximize the Excel Application.

Next, click on the Log File Folder button.

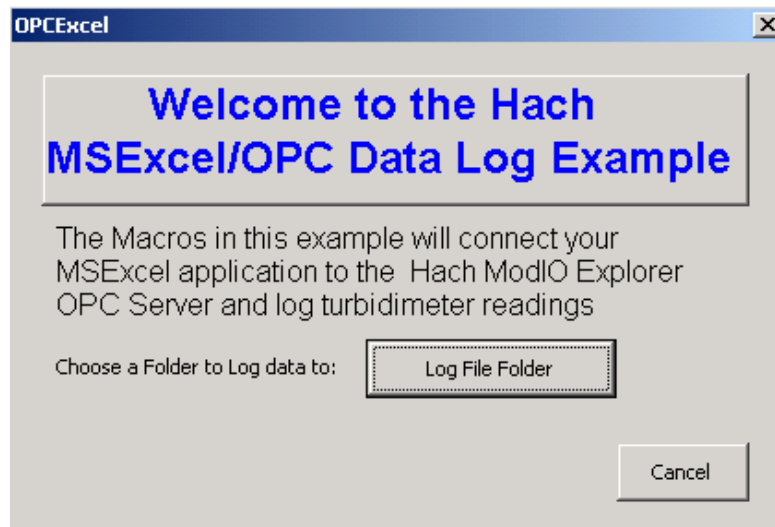


Figure 5: Splash Screen

Do not change the file name in the Save As dialog box. Select a folder to save your data log file too and click on the Save button.

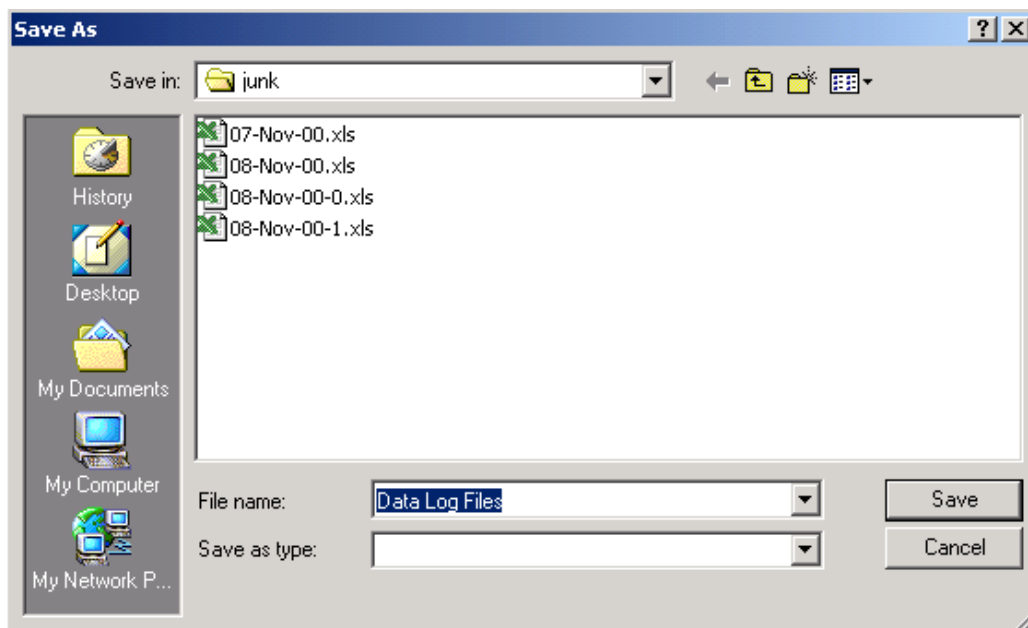


Figure 6: Save As Dialog

Verify that your path for the saved files is OK. If OK, click Yes.

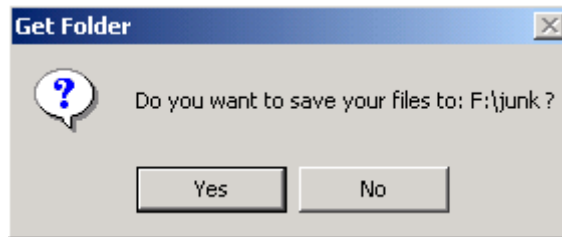


Figure 7: File Save Prompt

The Data Log Properties dialog allows one to choose the number of turbidimeters that they are acquiring data from. The range is from 1 to 8 for this macro example. You can also select the sample interval in whole number seconds and the maximum number of records you want to log. Since the maximum sample rate for a 1720D turbidimeter is 3 seconds, it is not practical to sample faster. Typical real world sample rates are in the range of 1 to 30 minutes (60 to 1800 seconds). The maximum number of records you can record is 65,000.

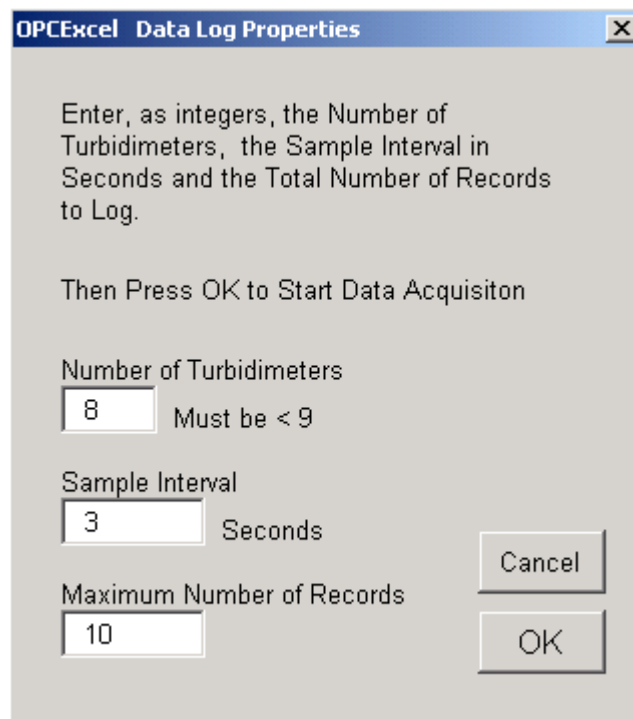


Figure 8: Data Log Properties

Click OK after making your selections (for quick demonstration purposes, leave selections as shown).

The Chart Properties dialog allows one to select which turbidimeters will show in the trend chart. You can select up to 8, but the chart will get "busy" if more than 4 are selected. You can also select the number of data points to be plotted on the trend chart.

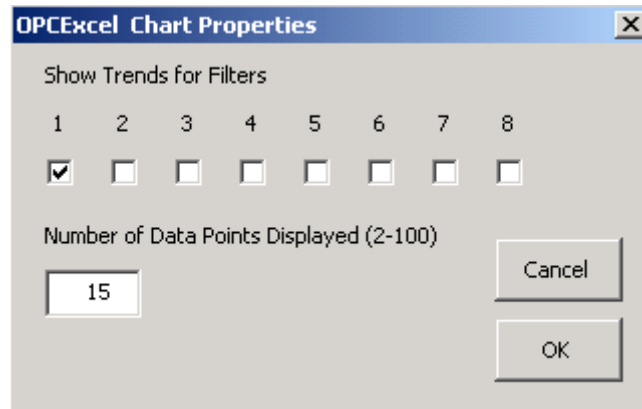


Figure 9: Chart Properties

Clicking OK in the Chart Properties dialog will start the data acquisition process. The process first fires the ModIO Explorer Demo application and its Icon will appear in the Task bar.

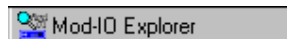


Figure 10: Task Bar Icon

You can view the data as it is acquired on the Turbidity Data sheet.


	A	B	C	D	E	F	G	H	I	J
	 OPC Turbidity Data Logger Copyright Hach Company Oct. 6, 2000									
1	Date	Time	Filter 1	Filter 2	Filter 3	Filter 4	Filter 5	Filter 6	Filter 7	Filter 8
2	08-Nov-00	09:02:25	0.260	0.360	0.460	0.560	0.660	0.760	0.860	0.960
3	08-Nov-00	09:02:28	0.270	0.370	0.470	0.570	0.670	0.770	0.870	0.970
4	08-Nov-00	09:02:31	0.280	0.380	0.480	0.580	0.680	0.780	0.880	0.980
5	08-Nov-00	09:02:34	0.280	0.380	0.480	0.580	0.680	0.780	0.880	0.980
6	08-Nov-00	09:02:37	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
7	08-Nov-00	09:02:40	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
8	08-Nov-00	09:02:43	0.200	0.300	0.400	0.500	0.600	0.700	0.800	0.900
9	08-Nov-00	09:02:46	0.210	0.310	0.410	0.510	0.610	0.710	0.810	0.910
10	08-Nov-00	09:02:49	0.230	0.330	0.430	0.530	0.630	0.730	0.830	0.930
11	08-Nov-00	09:02:52	0.240	0.340	0.440	0.540	0.640	0.740	0.840	0.940
12										

Figure 11: Turbidity Data View

You can also view the trend chart by clicking on the View Chart ... button in the OPC1720D tool bar.

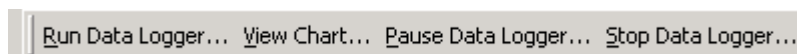


Figure 12: Tool Bar

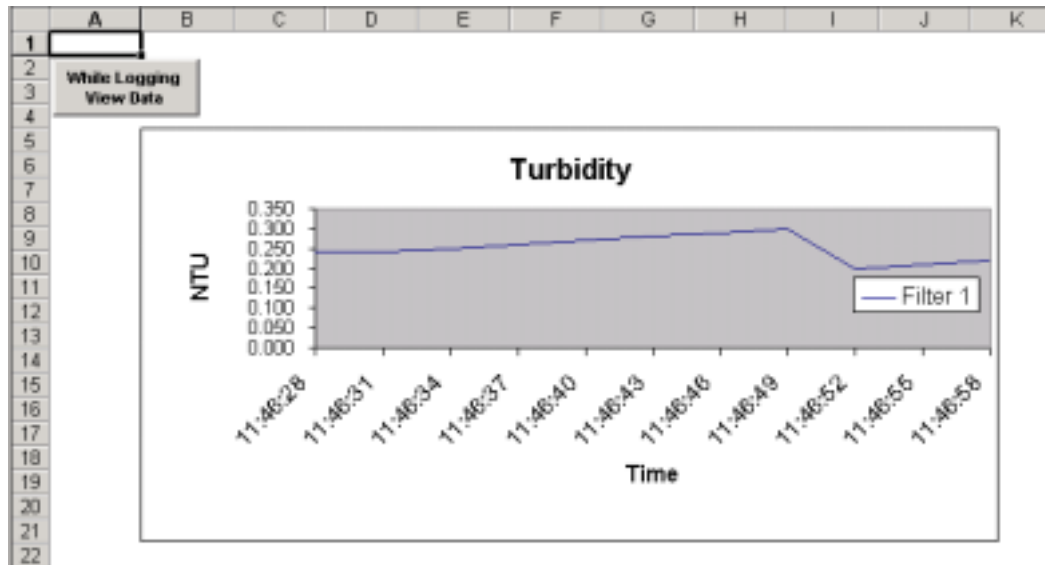


Figure 13: Chart View

To return to the Turbidity Data sheet, click on the While Logging View Data button on the Chart Sheet.

After all the data has been logged to the Turbidity Data spreadsheet, the Macro will calculate some statistics and create a Report sheet. Next it will save the file at the location you chose. While the macro is doing this, a Please Wait information box will be displayed if you are running Excel 2000. Excel 97 does not allow Modeless forms so you won't see the Please Wait ... form, but the Status Bar of Excel will indicate that it is calculating and saving.

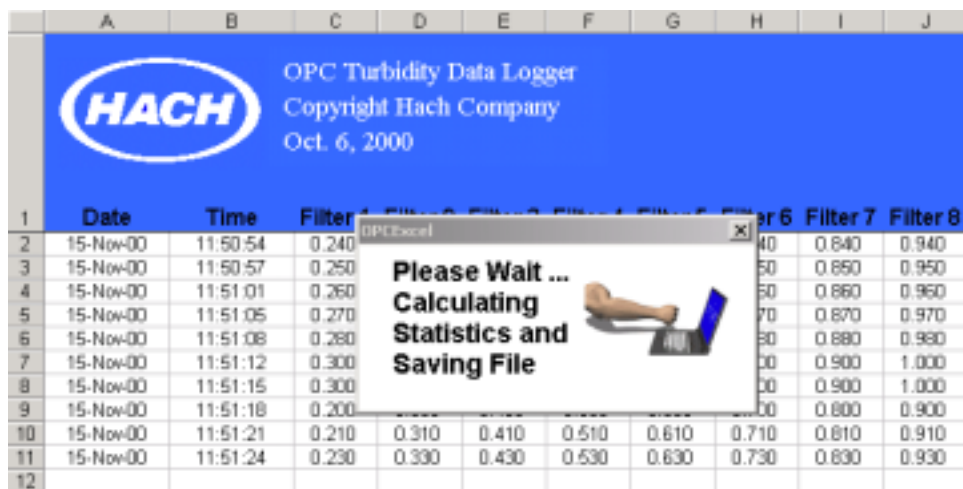


Figure 14: Calculating Statistics

The macro closes the original read only Workbook, OPCEXcel.xls and the ModIO Explorer Demo OPC server. The saved Workbook is displayed.

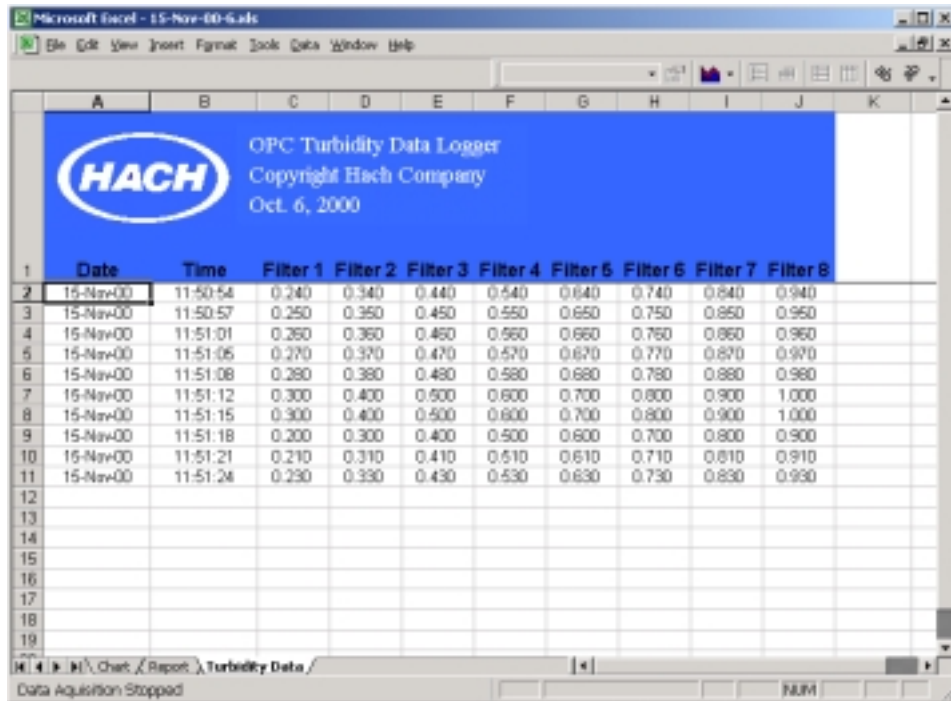


Figure 15: Saved Workbook

You can view the chart by clicking on the chart tab at the bottom of the Excel window.

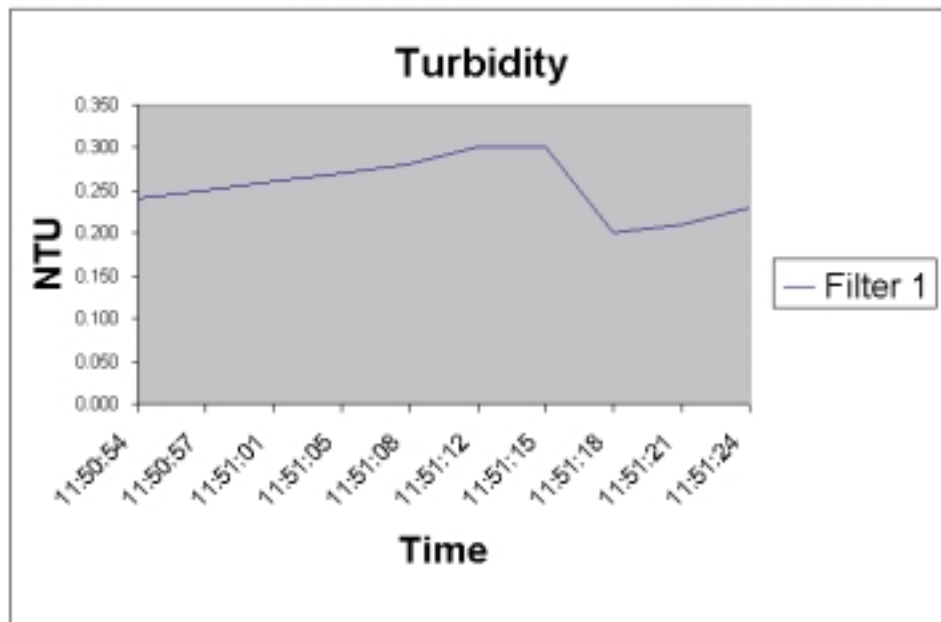


Figure 16: Chart

You can use the Chart Tool Bar at the top of the Excel Window to modify the chart.

The report can be viewed by clicking on the Report tab at the bottom of the Excel Window.

	A	B	C	D	E	F	G	H	I	J
1		The following are percentile values for the data set terminated on 11/15/2000 11:51:25 AM								
2										
3			Filter 1	Filter 2	Filter 3	Filter 4	Filter 5	Filter 6	Filter 7	Filter 8
4										
5		50th Percentile	0.255	0.355	0.455	0.555	0.655	0.755	0.855	0.955
6		90th Percentile	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
7		95th Percentile	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
8		98th Percentile	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
9		99th Percentile	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
10		0.30 NTU Percent Rank:	88.800%	0.000%	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
11										
12		The following readings represent the second of two consecutive readings \geq 1.0 NTU								
13										
14		Date	Time	Filter 1	Filter 2	Filter 3	Filter 4	Filter 5	Filter 6	Filter 7
15		15-Nov-00	11:51:15							1.000
16										
17										

Figure 17: Report

The Report displays the 50th through 99th percentile calculations and the 0.30 NTU percent rank for each of the turbidimeters. The calculations are based on standard Excel statistical functions. The second of any two consecutive readings greater than or equal to 1 NTU are also displayed in the report.

As with any software, many features could be added to this application to enhance its value in performing a specific function. Some enhancements that come to mind are:

- Periodic data backup while logging to prevent loss of data should the system have an upset during logging.
- The manual creation of an additional report from the data on the Turbidity Data sheet, thus allowing a report to be based on edited data (for example, deletion of Backwash data).
- The ability to easily change the time range that the data is displayed over on the trend chart.

These fun exercises are left to the readers of the application note.

The Macro

To view the Macro, press Alt + F11 to open the Visual Basic for Applications development environment.

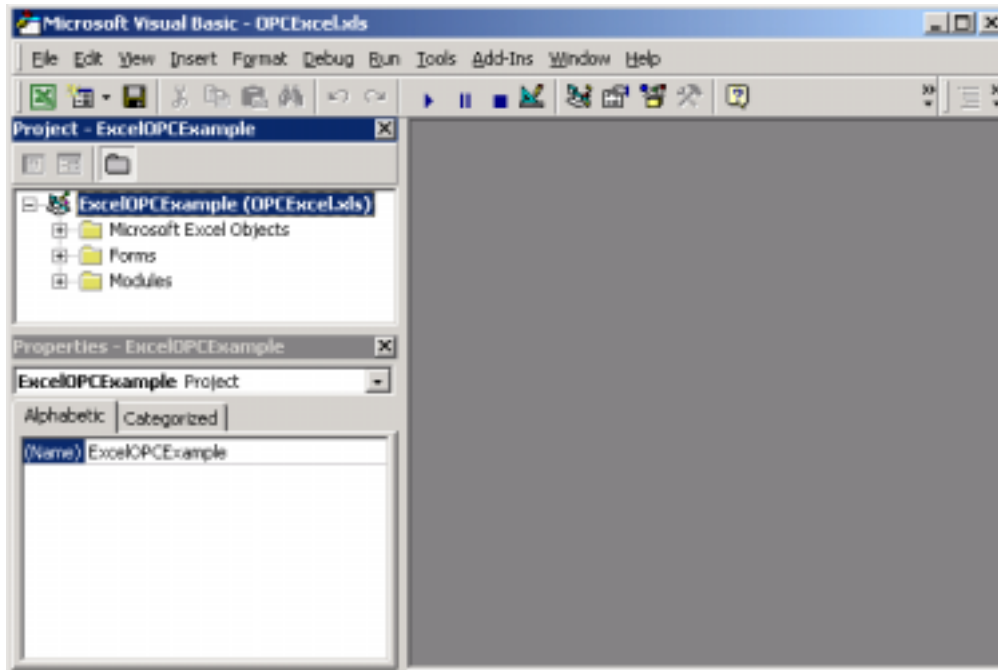


Figure 18: Visual Basic IDE

Expand the folders in the Project window.

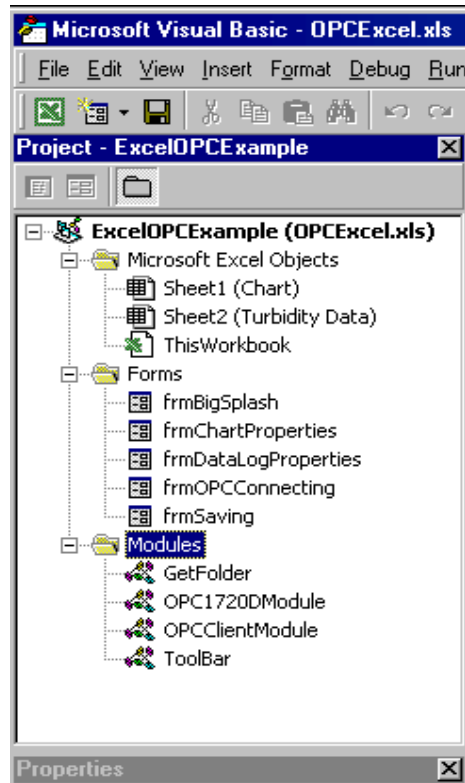


Figure 19: Project Folders

The Project window shows that this Excel Workbook, OPCExcel.xls, is made up of the following:

Microsoft Excel Objects

- **Sheet1 (Chart)**

This object contains one command button that allows you to return to the Turbidity Data sheet while you are logging data.

- **Sheet2 (Turbidity Data)**

There is no additional code associated with this object.

Forms

- *The five custom forms that are used with this macro are listed.*

Modules

- **GetFolder**

This module contains methods (functions) to browse for a folder to store the logged data too.

- **OPC1720DModule**

This module contains subroutines and functions that display the interface dialogs (Forms), acquire the configuration data, format the worksheets, log the data to the worksheet, calculate the statistics, generate the report and trend chart, and save the workbook to your selected folder. Key subroutines to review in this module are the AquireData and StartTimer. The StartTimer subroutine uses the Excel OnTime method to create the time delay between logged data records. The AquireData routine is recursively called via the StartTimer routine. A detailed discussion of OPC1720DModule module and the associated Forms is beyond the scope of this application note. The author hopes that the comments and format of the module will be self-explanatory.

- **OPCClientModule**

This module is the heart of the OPCExcel.xls macro example. All OPC "creations" and "calls" are made in this module. The subroutines and function that make up this module will be discussed in detail.

- **ToolBar**

The ToolBar module houses the code for the buttons on the OPC1720D tool bar. The RunDataLogger button starts the OPC1720D routine in the OPC1720DModule. The PauseDataLogger button pauses the logging process via a modal dialog box (message box) by setting a flag read in the OPC1720DModule. The dialog box must be cleared before logging will continue. The ViewChart button selects the Chart sheet for viewing and the StopDataLogger stops the logging process and saves the logged data, both do this by setting a flag.

OPCClientModule

The OPCClientModule consists of the following subroutines that are called by the OPC1720DModule:

- **ConnectOPC**

This routine is responsible for creating the OPC client connection to the OPC server (ModIO Explorer Demo in this example).

- **DisconnectOPC**

This routine is responsible for disconnecting the OPC client (the Excel Macro) from the OPC server and assuring that memory is cleared.

- **IsOPCConnected**

This function returns a Boolean value depending on whether the OPC client connection to the server is made or not. This is done indirectly by testing to see if a ConnectedServerName exists.

- **GetData**

This routine performs a synchronous read of the OPC tag values requested via the call GetData(nNumberTurbs), nNumberTurbs = number of turbidimeters. When periodic reads are required as with a data logger, the synchronous read method is preferred because it waits until the read is completed before continuing with the VBA code.

The OPC Data Access Automation DLL (OPCDAAuto.DLL) is used by the OPCClientModule to connect to the OPC server, read tags, and disconnect from the server. For detailed information on this DLL, its methods and properties refer to the Data Access Automation Interface Standard referenced in this document.

A reference to the OPCAAuto.DLL must be made for the macro. In the VBA development environment, select Tools | References on the Menu Bar and add OPC Automation 2.0 to the reference list.

The structure of an OPC Server object can be viewed as follows:

OPC Server Object
 OPC Groups Collection
 OPC Group Object(s)
 OPC Item Collection
 OPC Item Object(s)

Note: A collection is a set of same data type objects. OPC Item Object(s) are the tags we will be reading.

For a discussion of collection objects read *The Visual Basic Collection Object* at URL:

<http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbconthevisualbasiccollectionobject.htm>

Connecting to the OPC Server

A connection must be made to the OPC server before communication with the server can be consummated.

After declaring the objects and variables required for the OPCClientModule module, the connection process begins with creation of a new OPC Server Object.

```
Set MyOPCServer = New OPCServer
```

Because we want to connect to a specific OPC Server, the server must be defined.

```
ConnectedServerName = "Hach.ModIO.Demo"
```

Note: This OPC Server name was registered on the PC when the server software was installed.

The Connect method of the OPC Server Object is used to connect the server.

```
MyOPCServer.Connect (ConnectedServerName)
```

Each OPC Server has a specific interface for its Collection of Groups. To get the interface use:

```
Set MyServerGroup = MyOPCServer.OPCGroups
```

Next we activate the Group.

```
MyServerGroup.DefaultGroupsActive = True
```

The interface allows us to set a deadband by assigning an integer from 0 to 100 for the deadband. The integer is the percent of fullscale deadband.

```
MyServerGroup.DefaultGroupDeadband = 0
```

Now we add a Group to our Group Collection. You can use any Group name you want.

```
MyGroupName = "MyDataGroup"
```

```
Set MyOPCGroup = MyServerGroup.Add(MyGroupName)
```

Next initialize the update rate for the group. The update rate is a long integer value representing the millisecond group update rate.

```
MyOPCGroup.UpdateRate = 1000
```

We now define arrays of OPCItemIDs and ClientHandles using a For-Next loop. The OPC tags we want to read (see the ModIO Explorer Demo) have a structure:

```
ModIO1.Sensor1-1720D.Turbidity  
ModIO1.Sensor2-1720D.Turbidity  
*****  
ModIO1.Sensor8-1720D.Turbidity
```

Assigning the array:

```
For i = 1 To IItemCount  
    sMyOPCItemIDs(i) = "ModIO1.Sensor" & (i) & "-1720D.Turbidity"  
    IClientHandles(i) = i     ' Sets a reference pointer number for this point  
Next i
```

We get the item collection from the current OPC Group and set the collection to active.

```
Set MyOPCItemCollection = MyOPCGroup.OPCItems
```

```
MyOPCItemCollection.DefaultIsActive = True
```

Validation is recommended via the Validate method to ensure we haven't defined and set a non-existent tag.

```
MyOPCItemCollection.Validate IItemCount, sMyOPCItemIDs, IItemServerErrors
```

We then test the returned array of Item Server Errors for any reported errors.

```
For i = 1 To IItemCount  
    If IItemServerErrors(i) <> 0 Then  
        MsgBox "This Tag is not a valid OPC Item" & " " & sMyOPCItemIDs(i) & _
```

```

        """" , vbExclamation, "ConnectOPC"
    bTagError = True
End If
If bTagError Then
    ' Clear Memory Resources and End Program
    Set MyOPCItemCollection = Nothing
    ' Remove the group from the server
    MyServerGroup.Remove (MyGroupName)
    ' Release the Group system resources used
    Set MyServerGroup = Nothing
    Set MyOPCGroup = Nothing

    ' Disconnect from the OPC Server
    MyOPCServer.Disconnect
    ' Release the OPC Server object system resources
    Set MyOPCServer = Nothing
    ' End Program
    End
End If

Next i

```

The validation test loop contains code to disconnect the OPC Server and clear memory if an error is detected.

If we have no errors, the AddItems method is used to create OPC Item Objects and add them to our OPC Item Collection.

Note: IClientHandles are passed here, but not used in the GetData routine.

```

MyOPCItemCollection.AddItems lItemCount, sMyOPCItemIDs, _
    IClientHandles, lItemServerHandles, lItemServerErrors

```

This completes the connection to the OPC Server.

Getting the OPC Tag Data

Once we have connected to the OPC Server we can use the GetData routine to read the tag values. This routine fills a string array (ValString()) with the OPC tag values. If the OPC Quality of the tag is not good, it places the string "Bad" in the array. The same holds true if an OPC Error is returned and the string "Error" is placed in the array. The GetData(nNumTags as Integer) routine is passed an integer value equal to the number of turbidimeters to be read. The SyncRead method is used to read OPC data from the server.

```

MyOPCGroup.SyncRead OPCCache, lItemCount, lItemServerHandles(), _
    Values(), Errors(), Qualities

```

Next we test for Bad Quality and Errors.

```

For li = 1 To lItemCount
    If Qualities(li) <> &HC0 Then

```



```

        Values(li) = "Bad"
    End If
    If Errors(li) <> 0 Then
        Values(li) = "Error"
    End If
    ValString(li) = Values(li)
Next li

```

If no errors or bad quality is detected the ValString() array will contain our requested turbidity readings. Since ValString() is a Public Global String Array Object, the OPC1720DModule routines can have access to the data.

Disconnecting the OPC Server

The DisconnectOPC routine provides a clean shutdown of the OPC connection. It removes the items and groups and their associated collections from the OPC Server and then disconnects from the server.

To disconnect we first create an array of Item Server Handles. These are the items we wish to remove. ReDim is used to set the size of the array to be equal to our number of turbidimeters.

```

Dim RemoveItemServerHandles() As Long
ReDim RemoveItemServerHandles(ItemCount) As Long

```

We also create an array of Item Server Errors and since this array is established in the OPC Server, we do not need to dimension its size.

```

Dim RemoveItemServerErrors() As Long

```

We then get the handles for the valid items.

```

For i = 1 To ItemCount
    ' A none zero ItemServerHandle is valid
    If ItemServerHandles(i) <> 0 Then
        RemoveItemServerHandles(ITempCount) = ItemServerHandles(i)
        ITempCount = ITempCount + 1
    End If
Next i

```

The Remove method is used to remove the items.

```

MyOPCItemCollection.Remove ITempCount, RemoveItemServerHandles, _
    RemoveItemServerErrors

```

Next clear the memory of the collection.

```

Set MyOPCItemCollection = Nothing

```

Remove the Group from the server.

```

MyServerGroup.Remove (MyGroupName)

```

Release the Group system resources.

```
Set MyServerGroup = Nothing  
Set MyOPCGroup = Nothing
```

Disconnect the OPC Server.

```
MyOPCServer.Disconnect
```

Release the OPC Server object system resources.

```
Set MyOPCServer = Nothing
```

On exit of this routine we use the VB keyword "End" to reset all module level variables, stop code execution, destroy all objects, free memory, and invalidate object references held by other routines. This assures that if there are no other connections from other programs to the OPC Server, the server will shut down.

Key Items That Define the OPC Server and Desired Tag

To communicate with an OPC Server you need to define the Server Name. In the case of this example:

```
ConnectedServerName = "Hach.ModIO.1"
```

The Server Name is "Hach.ModIO.1". This Server Name is registered in the system registry when the OPC Server is installed.

The specific OPC tag also needs to be defined. In this example a For – Next loop was used to iterate through a series of tag definitions:

```
sMyOPCItemIDs(i) = "ModIO 1-01.Sensor" & (i) & "-1720D.Turbidity"
```

The basic tag for sensor 1 being:

```
ModIO 1-01.Sensor1-1720D.Turbidity
```

To connect to other OPC Servers you will need to refer to the target Server documentation to obtain the Server Name and the structure of the specific tags you are interested in. There are several "Browser" ActiveX controls on the market now that you could incorporate in your VBA interface. Browser controls would allow you to dynamically set your Servers and tags that you want to poll. A search of the Internet should provide you with a list of available OPC ActiveX Browser controls.

Appendix A

If the ModIO Explorer Version 2.0 is Already Installed

Warning: If you have a version of the ModIO Explorer prior to **2.0**, **do not install** the ModIO Explorer Demo software. Select a different PC to install this demo on.

If you already have the functional ModIO Explorer OPC server software version 2.0 installed you do not need to install the ModIO Explore Demo software. You can download a demo configuration file from:

<http://www.aquatrend.com/Downloads/Software/MIOConfig.osf>

You will need to move or rename your current ModIO Explorer configuration file. Via Windows Explorer, go to **C:\PROGRAM FILES\HACH\OPC** and rename your current **MIOConfig.osf** file to **OldMIOConfig.osf**. Place the downloaded configuration file in the **C:\PROGRAM FILES\HACH\OPC** folder. You can delete the demo configuration file when you are finished with the demo and rename your old configuration file back to **MIOConfig.osf** to restore your previous configuration.

Removing the OPCEXcelVBEx Demo

You can remove the demo by clicking the Add-Remove Programs Icon on the Control panel and selecting **OPCEXcelVBEx** to remove.

Reference:

<http://www.opcfoundation.org/>

<http://www.aquatrend.com/>

<http://msdn.microsoft.com/library/devprods/vs6/vbasic/vbcon98/vbconthevisualbasiccollectionobject.htm>

Data Access Automation Interface Standard Version 2.02 February 4, 1999, OPC Foundation

For more information, visit our AquaTrend Technical Information site at:

<http://www.aquatrend.com/>



FOR TECHNICAL ASSISTANCE, PRICE INFORMATION AND ORDERING:
In the U.S.A. – Call toll-free 800-227-4224
Outside the U.S.A. – Contact the HACH office or distributor serving you.
On the Worldwide Web – www.hach.com; E-mail – techhelp@hach.com

HACH COMPANY
WORLD HEADQUARTERS
Telephone: (970) 669-3050
FAX: (970) 669-2932
