W&T OPC Server for Network I/O Devices

Version 2.05

About OPC Servers

OPC (OLE for Process Control) is a software interface for accessing industrial process data, based on Microsoft's OLE technology. Application programs like HMI or SCADA software that use this interface are called OPC clients. On the other side of the interface we have OPC Servers. Those are device drivers that represent some particular hardware in an abstract way, as a set of OPC variables.

The present OPC server is intended to access Digital I/O Servers and serial Com Servers (W&T product families 50xxx and 58xxx respectively).

Installation

Start the setup program from the installation disk (for example by clicking "Run..." in the Start menu, then typing "A:\Setup"). This will install and register the OPC server on your computer.

The OLE server name that OPC clients will have to supply in order to connect to the server is: "Wiesemann-Theis.DigitalEA". The OPC server will start automatically when such a connection is attempted. For configuration you can also start it manually: You will find an entry "W&T OPC Server" under "Programs" in the Start menu.

Note: Installation on Windows 95 will possibly fail, with error messages about missing functions in OLE32.dll. An updated version of this DLL that will fix the problem is contained in the "DCOM for Windows 95" package, which is available as a free download from Microsoft, at http://www.microsoft.com/com/dcom/dcom95/dcom1_3.asp.

Uninstallation

To remove the OPC Server, use the "Software" option on the Control Panel. Look for "W&T OPC-Server for Digital IO" in the list.

Device Configuration

The "Com Server" dialog for device configuration appears every time you add a new I/O Server or serial Com Server, or want to change some settings for an existing device. The meaning of the individual items on it is as follows:

Host name or IP address: Must agree with the IP address (four numbers, separated by dots, e.g. 172.16.232.77) that was set on the device itself. In case the address has also been assigned a host name, using DNS or a similar name service, you may use this name instead of the IP address as well.

Server type: "With measuring/monitoring" refers to the I/O Server models 50211/50311. "Serial with 2+2 I/O" is intended for serial Com Servers equipped with the special interface module 18804.

There is *no* special type such as "serial with 12+12 I/O" to describe models 50310/50311. From the OPC server's point of view, the serial port of an I/O Server is a Com Server of its own. You could add two individual devices, sharing the same IP address, where one of them is an I/O Server, and the other's type is simply "serial".

TCP port number: Is 8000 for serial ports (or 8100, 8200, 8300 for the serial ports B, C, D on a 4-port server). The port number for I/O servers is usually 49153 (at least by default).

Note: Each serial port actually occupies two TCP connections: One data connection on the port number as it was specified, and one control connection on a port number which is greater by 1094, that means 9094, 9194, 9294 or 9394.

Unique device number: This number is automatically assigned when adding a server, and is intended to make sure that all devices end up with different names (Box1, Box2 and so on). Changing it by hand might make sense in order to let the I/O ports and the serial port on one server (see the notes on "server type") also share the same device number. The resulting device names could then be, for example, "Box1" and "Com1".

Update at least every ... seconds: I/O Servers and Com Servers report some events on their own initiative (incoming serial data, changed input lines on a 12+12 I/O Server), others require regular polling (changed input lines on a 2+2 I/O Server, for example). This value sets the interval between two such queries, where your input will be rounded to a multiple of 0.1 seconds.

Another purpose of regular polling is to quickly identify a disrupted TCP connection. A problem is assumed (and the connection is reset) when the device takes longer to reply than one of these intervals. Caution: Setting too small a value can therefore prevent any connections to be made at all. However, this problem should only occur in extreme situations, for example with a 0.1 second update interval, when at the same time the I/O server is not in same local network as your computer.

Disabled: No attempt is made to open a TCP connection to a disabled server. The OPC variables belonging to the server will be created, but reading them will always yield OPC_QUALITY_NOT_CONNECTED.

RS232 parameters: An initialization string like "19200,n,8,1,x", which sets the line parameters at the time the OPC server is started. Meaning of the individual parameters is: baud rate, parity bit (n = none, e = even, o = odd), number of data bits (5 - 8), number of stop bits (1 or 2), handshake protocol (x = software, XOn/XOff, h = hardware, RTS/CTS).

When no initialization string is supplied, the serial settings of the Com Server remain unchanged.

Incoming text complete after...: To represent the incoming serial data stream as an OPC variable, it somehow must be split into strings, or packets in other words. Either pauses or the occurrence of a series of special characters can be defined to end a packet.

When special characters are defined, pauses should be disabled (by setting a value of 0 seconds). Combination of both methods is possible, but not recommended.

Special characters are always entered as decimal numbers, even if they should be printable characters. For example, the correct input to have packets end at the occurrence of three plus signs ("+++") would be "43,43,43".

Filter from incoming data: Filtering special characters takes place after the serial data have already be split into packets, so filtering characters that are also packet delimiters is perfectly legal. Null bytes will be removed in any case, even if "0" is not expressly mentioned here, because they must not appear within text strings.

Append when sending: One reasonable use could be to append CR (13), LF (10) or both (13,10) to indicate end of line.

Direct Access by Control Panels

For each server on the list, you can open a control panel, which offers essentially the same level of access as the OPC interface. In particular, the individual controls are labeled with the same names as the corresponding OPC variables.

This feature is useful mainly to become familiar with the behavior of the devices, and for example to see how the identification of serial data packets is affected by the various options. But don't expect too much from it, because you cannot open more than one control panel at the same time.

OPC Variables for I/O Servers

For each I/O Server that is added to the configuration, the OPC server creates the following variables, where multiple I/O Servers are distinguished by the first part of the variable name: Box1, Box2, etc.

Box1.E.0 — Box1.E.11 (VT_BOOL, R): state of the inputs.

Box1.A.0 — Box1.A.11 (VT_BOOL, R/W): state of the outputs.

For I/O Servers with measuring and monitoring, there is also:

Box1.I.0 — Box1.I.11 (VT_R4, R): output currents (ampere).

Box1.T (VT_R4, R): internal temperature (Centigrade).

Box1.U (VT_R4, R): supply voltage (volt).

Box1.F (VT_BOOL, R/W): overload, outputs have been disabled. Reset the variable to enable them again.

Abbreviations

OLE data types:

- VT_BOOL: binary value
- VT_I2, VT_I4: integer number
- VT_R4: floating-point number
- VT_BSTR: string of characters

Access:

- R/W: read and write
- R: read only
- **W: write only**

OPC Variables for serial Com Servers

The device names for serial Com Servers are Com1, Com2, etc., and the individual variables are:

Com1.TxD (VT_BSTR, W): RS232 transmit data, assign a value to have it sent over the serial port.

Com1.RxD (VT_BSTR, R): RS232 received data (the most recent packet of text received from the serial port).

Com1.N (VT_I4, R/W): packet counter, is incremented every time a new packet of text arrives.

For ports configured as "serial with 2+2 I/O" (which makes most sense for ports with the special 18804 I/O module), there is also:

Com1.E.0 (VT_BOOL, R): digital input (DSR, or pin 4 on the I/O module).

Com1.E.1 (VT_BOOL, R): digital input (CTS, or pin 7 on the I/O module).

Com1.A.0 (VT_BOOL, R/W): digital output (DTR, or pin 6 on the I/O module).

Com1.A.1 (VT_BOOL, R/W): digital output (RTS, or pin 8 on the I/O module).

The following variables only grant access to the same serial line parameters that can also be set by the initialization string in the device configuration dialog:

Com1.Line.Baud (VT_I4, R/W): baud rate, in bits per second.

Com1.Line.Bits (VT_I2, R/W): number of data bits per character (5, 6, 7 or 8).

Com1.Line.Stop (VT_I2, R/W): number of stop bits (1 or 2).

Com1.Line.Parity (VT_I2, R/W): parity bit: 0 = none, 1 = odd, 2 = even parity

Com1.Line.Flow (VT_BOOL, R/W): use a handshake protocol for flow control?

Com1.Line.Hard (VT_BOOL, R/W): use hardware or software handshake? (This variable is not present on ports configured as "serial with 2+2 I/O", which only support software handshake.)

One I/O Server, several Clients

The OPC server gains exclusive access to the I/O Servers and serial Com Servers that have been assigned to it, at the moment it successfully establishes a TCP connection to them. That is why a configuration with OPC servers on several computers accessing the same I/O Server makes no sense or at least would not work reliably.

To achieve a similar effect, use a single OPC server instead, with clients from several computers accessing it by DCOM (distributed COM). Please note that you have to grant explicit access rights before using DCOM. This is done using the program "DcomCnfg.exe", which is a standard component on Windows NT.

In order to use DCOM on Windows 95, you first have to install the update packets DCOM95.EXE and DCM95CFG.EXE. (To see if this is still required, try to invoke DcomCnfg.exe from the command prompt.) Then change the Registry value HKEY_LOCAL_MACHINE\Software\Microsoft\OLE\EnableRemoteConnect to "Y". And be aware that Windows 95 does not allow an OLE server to be started in response to an external DCOM request. So on a Win95 computer, the OPC server would need to be started manually before OPC clients on other computers can connect to it.

More detailed information on DCOM is available in the Microsoft Knowledge Base. Some of the relevant articles are:

Q176799, INFO: Using DCOM Config (DCOMCNFG.EXE) on Windows NT
Q165101, HOWTO: Use Win95 as a DCOM Server
Q182248, HOWTO: Use DCOM Config (DCOMCNFG.EXE) with Windows 95
Q158508, FAQ: COM Security Frequently Asked Questions
Q174024, FAQ: DCOM95 Frequently Asked Questions

You can find these articles on the Microsoft web site: Either use the search engine, http://search.microsoft.com, make sure that category "Support & Knowledge Base" is selected, and enter an article number such as "Q165101". Or use a direct URL such as http://support.microsoft.com/support/kb/articles/q165/1/01.asp.

Access from Visual Basic or VBA

Other than the basic Custom Interface, this OPC server only offers an Automation Interface according to OPC version 2.0. In case you are familiar with the old 1.0 Automation Interface: There are severe differences between the two versions. Besides that, at least Visual Basic version 5.0 or Office 97 is required to use the new interface. For a complete specification of the interface, see the OPC Foundation home page, http://www.opcfoundation.org. For a first impression, however, the following examples (Excel macros) may be helpful as well.

Example 1 uses the namespace browser to find all the variables on the OPC server, and places their names in column 1 of the current Excel worksheet.

Example 2 takes the variable names from column 1 of the current Excel worksheet, and for each of them reads the properties "value", "unit" and "description", and places them in the adjacent columns.
There is a special significance to the property "signal quality", which essentially says if the OPC server can offer a valid value for a particular variable. One possible reason for such problems could be that the OPC server has only just been started (maybe automatically in response to the macro invocation, so this is not an unlikely scenario), and has not yet established the TCP connections to its I/O devices.

Example 3 takes a variable name from column 1 in the current worksheet, a value from column 2, and assigns the value to the name.
Unlike reading values, writing cannot be done by using the "item properties" of the OPC server, but only by directly accessing the appropriate OPCItem object. By the way, reading could also be done this way, and it even would be more efficient than the method shown in example 2. (And speaking of efficiency: In a real VB application, as opposed to just an Excel macro, one would try to keep the OPCItem object around as long as possible, rather than creating and deleting it again for each and every variable access.)

Note: Visual Basic can use the OPC interface only after the entry "OPC Automation 2.0" in the list of references has been activated. (In Visual Basic 6.0 this list is found in the "Project" menu, for Excel 97 see the "Extras" menu in the Visual Basic editor.)

Example 1

```
Option Base 1

Sub OpcGetNames()
    ’ Retrieve the available variable names and place them in column 1.
    ’ We are dealing with a hierarchical namespace, with a maximum depth
    ’ of two levels.
    Dim TheOpcServer As OPCServer
    Dim MyBrowser As OPCBrowser
    Set TheOpcServer = New OPCServer
    TheOpcServer.Connect ("Wiesemann-Theis.DigitalEA")
    Set MyBrowser = TheOpcServer.CreateBrowser

    Dim RootName, BranchName As String
    Dim i, j, k, n As Integer
    ’ First clear the contents of column 1.
    Columns(1).ClearContents
    n = 0   ’ number of variables found
    MyBrowser.ShowBranches
    For i = 1 To MyBrowser.Count
        RootName = MyBrowser.Item(i)
        MyBrowser.MoveDown (RootName)
        ’ leafs on the stem itself
        MyBrowser.ShowLeafs
        For j = 1 To MyBrowser.Count
            n = n + 1
            Cells(n, 1) = RootName + "." + MyBrowser.Item(j)
        Next j
        ’ dive into the branches
        MyBrowser.ShowBranches
        For j = 1 To MyBrowser.Count
            BranchName = MyBrowser.Item(j)
            MyBrowser.MoveDown (BranchName)
            ’ leafs at the lowest level
            MyBrowser.ShowLeafs
            For k = 1 To MyBrowser.Count
                n = n + 1
                Cells(n, 1) = RootName + "." + BranchName + "." +
MyBrowser.Item(k)
            Next k
            MyBrowser.MoveUp
            MyBrowser.ShowBranches
        Next j
        MyBrowser.MoveUp
        MyBrowser.ShowBranches
    Next i

    Set MyBrowser = Nothing
    TheOpcServer.Disconnect
    Set TheOpcServer = Nothing
End Sub
```

Example 2

```
Option Base 1

Sub OpcUpdate()
    ’ Query all variables found in column 1 for description
    ’ and current contents.
    Dim TheOpcServer As OPCServer
    Set TheOpcServer = New OPCServer
    TheOpcServer.Connect ("Wiesemann-Theis.DigitalEA")

    Dim PropertyIDs(5) As Long
    Dim Data() As Variant
    Dim Errors() As Long
    Dim i, j As Integer
    PropertyIDs(1) = 3      ’ OPC_PROP_QUALITY
    PropertyIDs(2) = 2      ’ OPC_PROP_VALUE
    PropertyIDs(3) = 100    ’ OPC_PROP_UNIT
    PropertyIDs(4) = 101    ’ OPC_PROP_DESC
    PropertyIDs(5) = 4      ’ OPC_PROP_TIME
    Columns("B").ClearContents
    Columns("E:F").ClearContents
    i = 1
    While Cells(i, 1) <> ""
        TheOpcServer.GetItemProperties Cells(i, 1), 5, PropertyIDs, Data,
Errors
        For j = 2 To 5
            Cells(i, j) = Data(j)
        Next j
        If Data(1) = 20 Then        ’ OPC_QUALITY_LAST_KNOWN
            Cells(i, 6) = "STALE"
        ElseIf Data(1) <> 192 Then  ’ OPC_QUALITY_GOOD
            Cells(i, 6) = "ERROR"
            Range(Cells(i, 2), Cells(i, 3)).ClearContents
        End If
    n = n + 1
    Wend

    TheOpcServer.Disconnect
    Set TheOpcServer = Nothing
End Sub
```

Example 3

```
Option Base 1

Sub OpcWrite()
    ’ Assign the variable (found in column 1) on the current row a
    ’ value (from column 2).
    Dim TheOpcServer As OPCServer
    Dim MyGroup As OPCGroup
    Dim MyItem As OPCItem
    Set TheOpcServer = New OPCServer
    TheOpcServer.Connect ("Wiesemann-Theis.DigitalEA")
    Set MyGroup = TheOpcServer.OPCGroups.Add("group")

    Set MyItem = MyGroup.OPCItems.AddItem(Cells(ActiveCell.Row, 1), 1234)
    MyItem.Write (Cells(ActiveCell.Row, 2))

    TheOpcServer.OPCGroups.Remove (MyGroup.ServerHandle)
    TheOpcServer.Disconnect
    Set TheOpcServer = Nothing
End Sub
```