TAKEBISHI Software Library

DeviceXPlorer Client Sample Guide

OPCUA/DA2.05A/3.0 対応

Users Manual
Revision P

はじめに

本書に対する注意事項
本製品の操作は、本書をよく読んで内容を理解した後に行ってください。
本書は、本製品の機能詳細を説明するものであり、お客様の特定目的に適合することを保証するものではありません。
本ソフトウェアと本書を運用した結果の影響について、株式会社たけびしは一切の責任を負いかねますので、ご了承下さい。
本書の一部または全部を、無断で転載、複製することは固くお断りします。
本書の内容については、将来予告なしに変更することがあります。

■目次

1 OPC DA クライアント

1.1 Prog.ID

当社製 OPC サーバーの Prog.ID は以下の通りです。

OPC サーバー名称	バージョン	Prog.ID
DeviceXPlorer OPC Server 6	Ver6	Takebishi.Dxp もしくは Takebishi.Dxp.6
DeviceXPlorer OPC Server 6	Ver5	Takebishi.Dxp もしくは Takebishi.Dxp.5
DeviceXPlorer OPC Server	Ver4	Takebishi.Dxp.1
MELSEC OPC Server	Ver3	Takebishi.Melsec.1

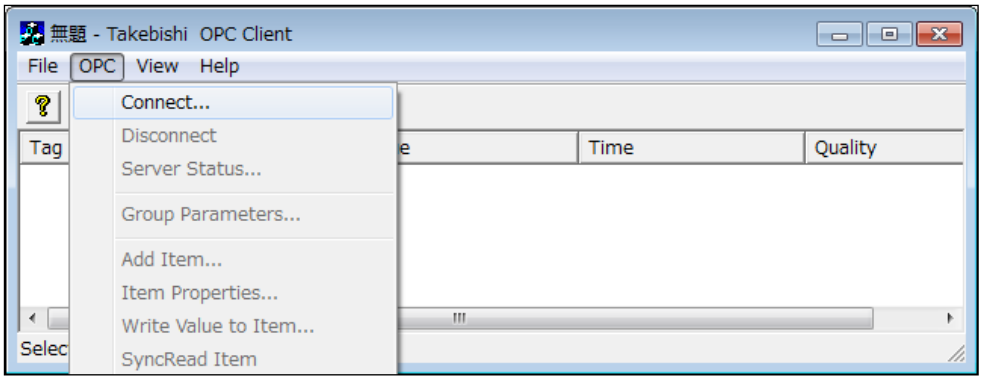
1.2 Visual C++テストクライアント（OPC Client.exe）

製品に添付されている OPC テストクライアントプログラムの使用方法について示します。

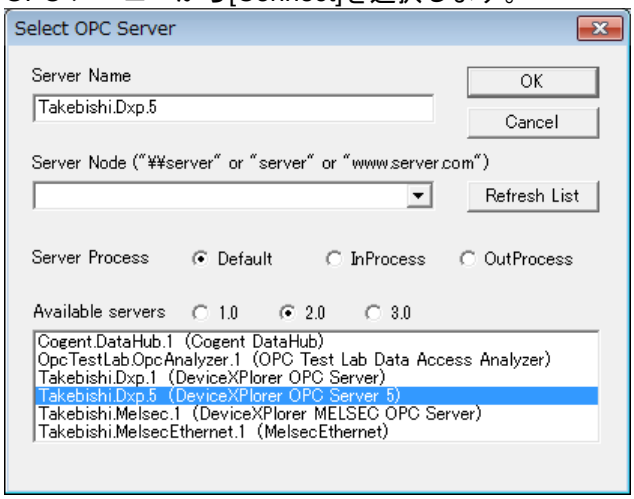
1.2.1 操作方法

OPC サーバーのインストールフォルダーの「Bin」フォルダー（C:¥Program Files¥TAKEBISHI¥DeviceXPlorer OPC Server 5¥Bin）から、「OPC Client.exe」を起動します。

【サーバーへの接続】



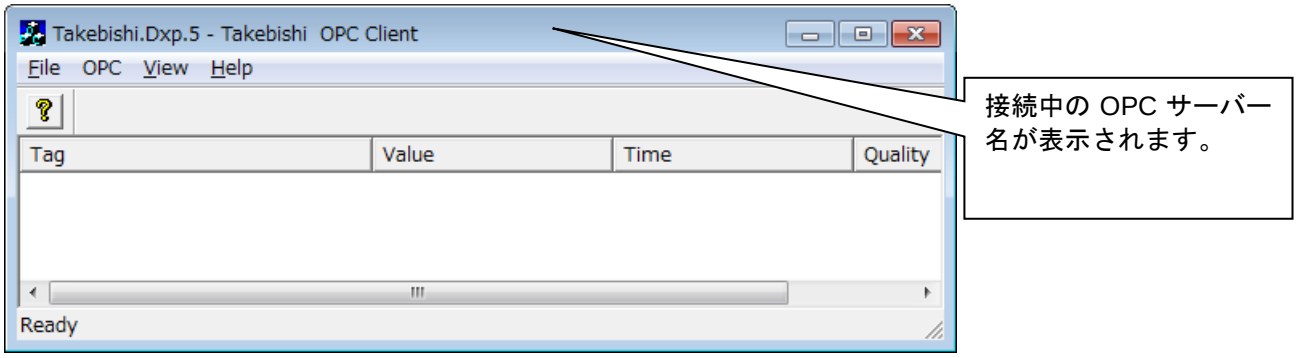
OPC メニューから[Connect]を選択します。



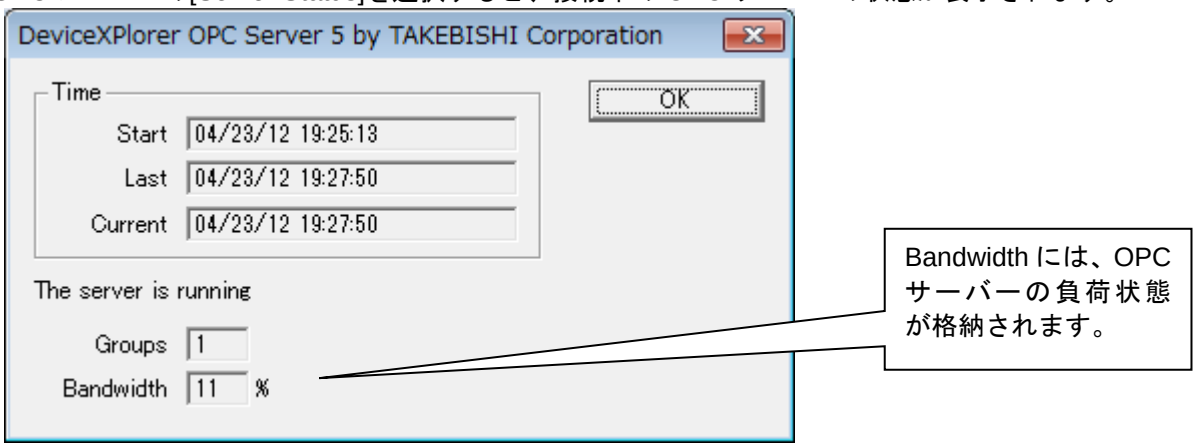
[Server Process]で接続する OPC サーバーのプロセス管理方法を選択します。[Default]を選択すると、接続する OPC サーバーの初期状態に従います。
[Available servers]で OPCDA のバージョンを選択すると、対応する OPC サーバーがリストアップされます。
デバイスエクスプローラ OPC サーバーの Prog.ID は「Takebishi.Dxp」です。その他の製品の Prog.ID は 1.1 項を参照してください。

OPC サーバーの Prog.ID が表示されない場合は、インストール先のフォルダーで、「register.bat」を実行し、OPC サーバーをレジストリに登録して下さい。

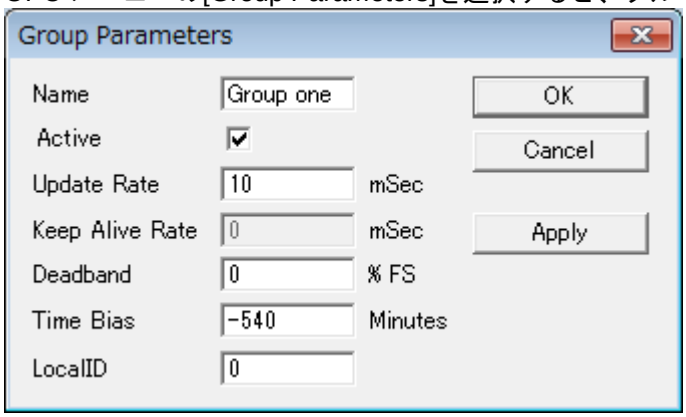
接続が完了すると、タイトルバーに接続中の OPC サーバー名が表示されます。



OPC メニューの[Server Status]を選択すると、接続中の OPC サーバーの状態が表示されます。



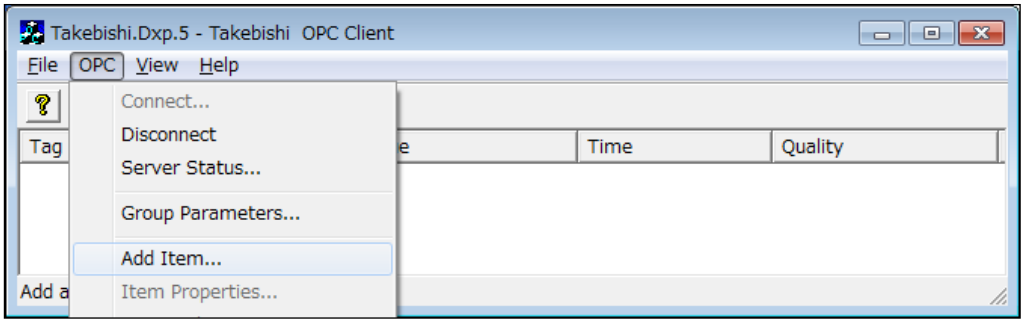
OPC メニューの[Group Parameters]を選択すると、グループのパラメーターが表示されます。



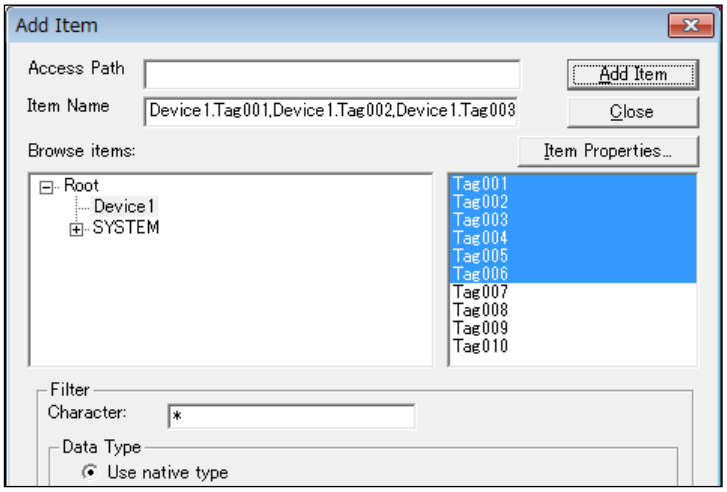
[Update Rate]は、このグループの読出周期を指定します。OPC サーバーは最小 10msec で設定できます。（ただし、ここで設定した周期を保証するものではありません）
[Active]のチェックボックスは、OPC クライアントとサーバー間の通信状態を指定します。[Active]がチェックされていない場合は通信を行いません。これらの設定は通信中も変更できます。
[Deadband]は、サーバーに登録されているタグの中でスケール変換されているタグの変化率を指定します。

接続終了する場合は、OPC メニューから[Disconnect]を選択して下さい。

【アイテムの追加】
OPC メニューの[Add Item]を選択します。

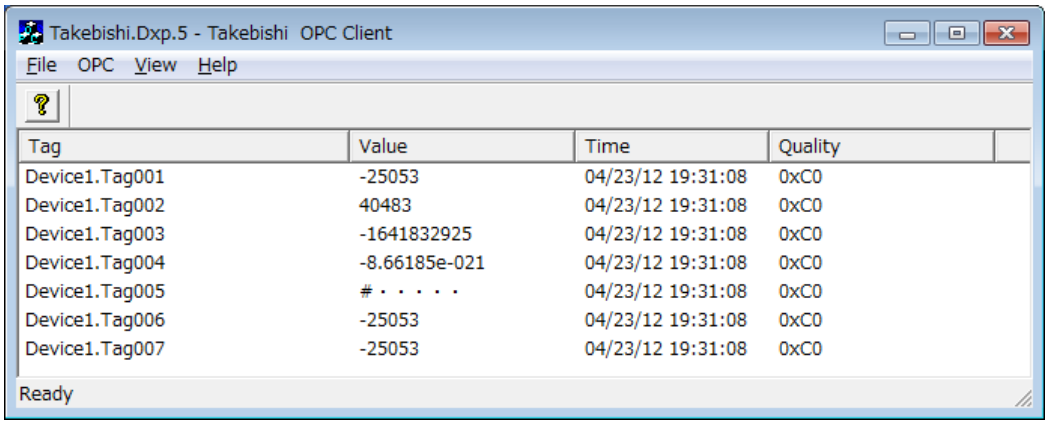


OPC サーバーで登録されているデバイス・グループ・タグが表示されます。
マウスで[Browse items]ビューから選択するか、[Item Name]にアクセスするデータを入力して、[Add Item]ボタンをクリックして下さい。アイテムが追加されます。

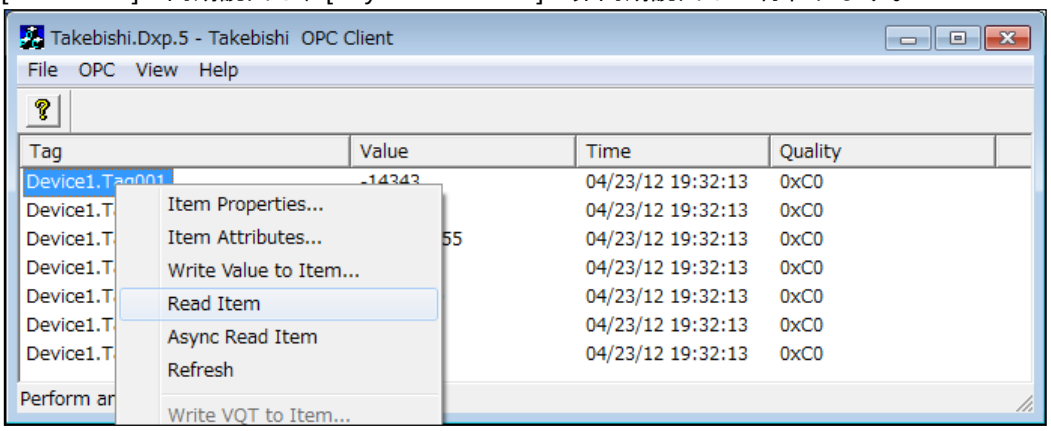


アイテムの追加が終了したら、[Close]ボタンをクリックして下さい。
追加されたアイテムがメインビューに表示されます。

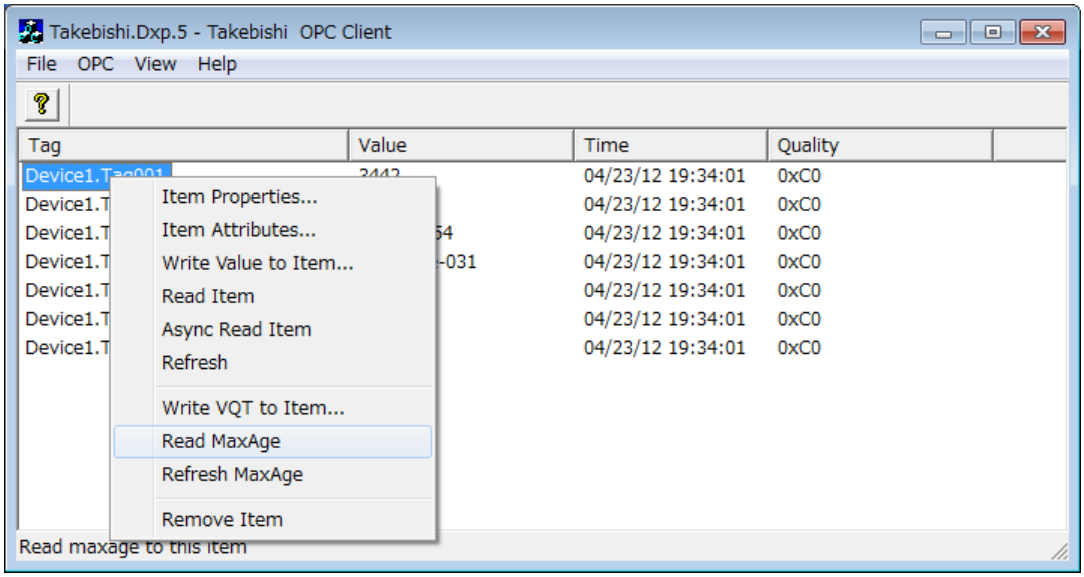
アイテムのデータが変化すると、ビュー上の Value と Time が更新されます。



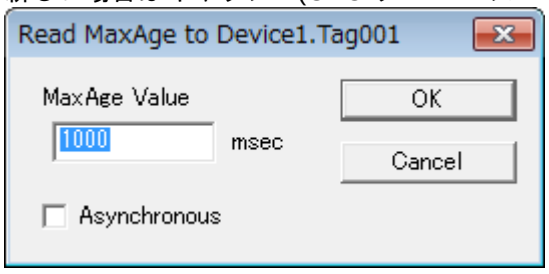
【アイテムの読出・書込】
強制的に値を読み出す場合は、アイテムを選択して OPC メニューまたはマウスの右クリックで表示されるポップアップメニューから[Read Item]もしくは[Async Read Item]を選択して下さい。
[Read Item]で同期読出し、[Async Read Item]で非同期読出しが行われます。



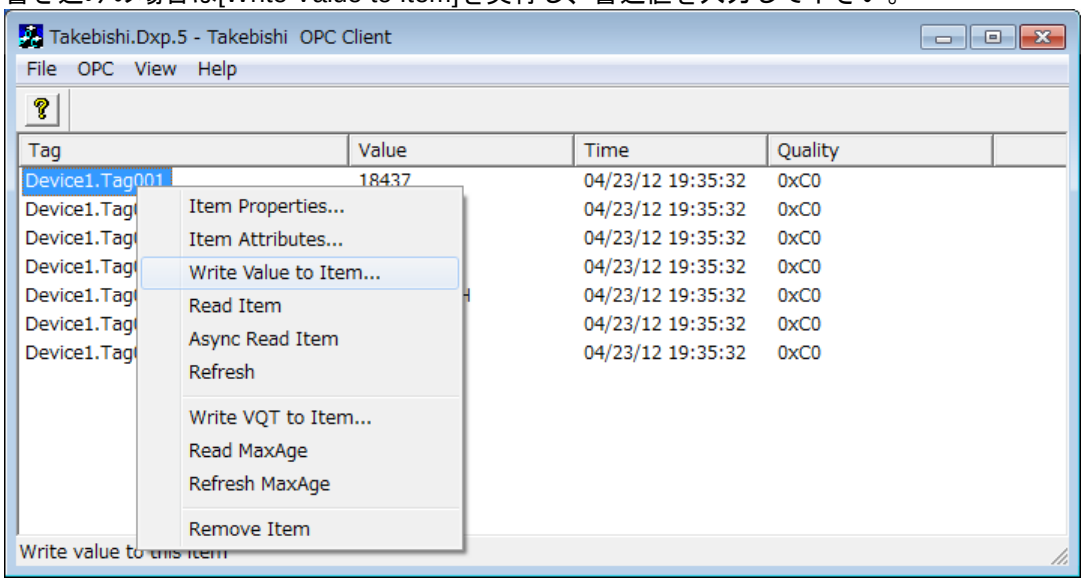
OPCDA3.0 対応の OPC サーバーでは、[Read MaxAge]によりデータの"古さ"を指定して値を読み出すことができます。



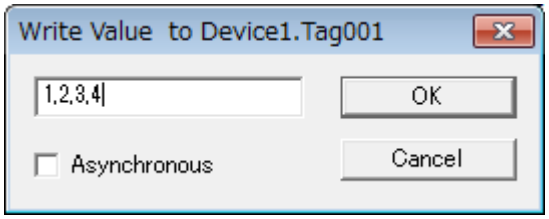
[MaxAge Value]に"古さ"の指定時間を設定して実行して下さい。1000[msec]を指定すると、OPC サーバー内に保持しているデータが 1000msec より古い場合はデバイス(PLC)から値を取得し、1000msec より新しい場合はキャッシュ(OPC サーバーのメモリ)の値を返します。



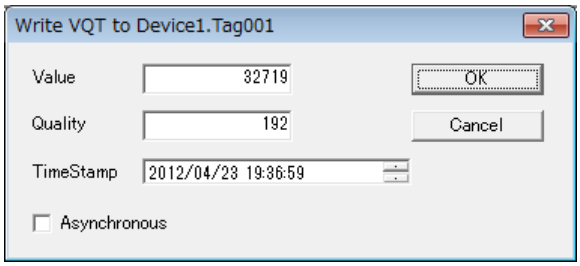
書き込みの場合は[Write Value to Item]を実行し、書込値を入力して下さい。



配列型のデータに書き込みを行う場合は値をカンマで区切って入力して下さい。
非同期処理を行う場合は、[Asynchronous]を選択して下さい。

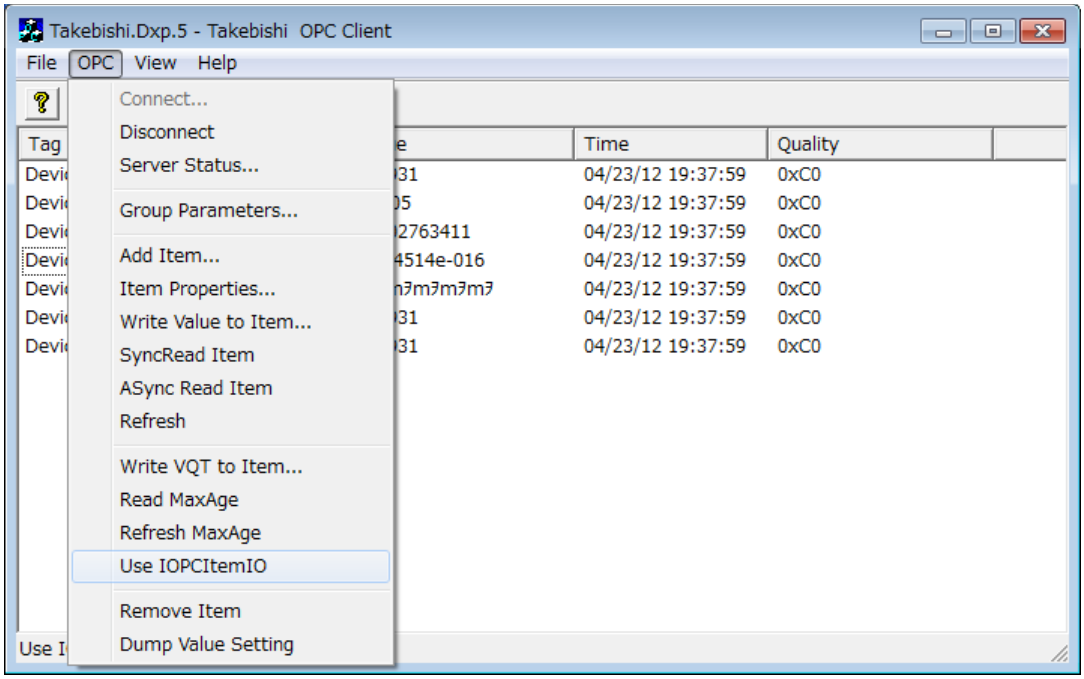


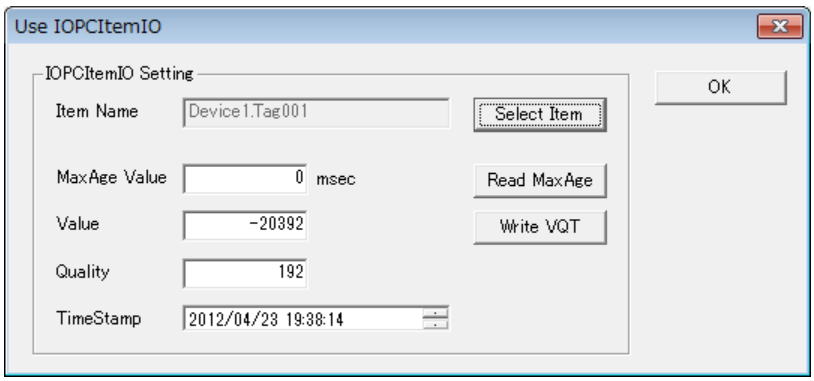
OPCDA3.0 対応の OPC サーバーでは、[WriteVQT]により OPC クライアントからデータの値、品質フラグ、タイムスタンプを書き込むことができます。
非同期処理を行う場合は、[Asynchronous]を選択して下さい。



1.2. Visual C++テストクライアント（OPC Client.exe）

OPCDA3.0 対応の OPC サーバーでは、IOPCItemIO インターフェースが使用できます。このインターフェースを使用すると、従来のグループ登録、アイテムの追加を行わずに直接アイテムにアクセスできます。





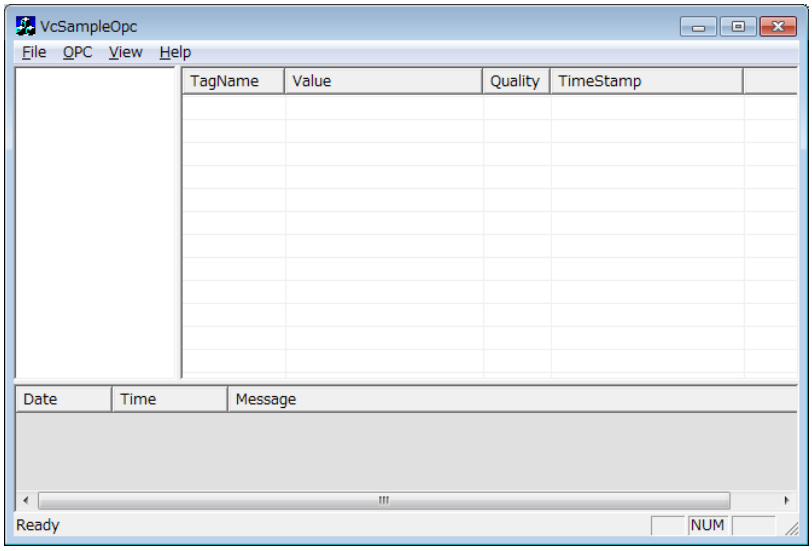
[Select Item]によりアイテムを選択します。
[Read MaxAge]によりデータの"古さ"を指定して値を読み出すことができます。
[WriteVQT]により OPC クライアントからデータの値、品質フラグ、タイムスタンプを書き込むことができます。

1.3 Visual C++サンプル（VcSampleOPC）

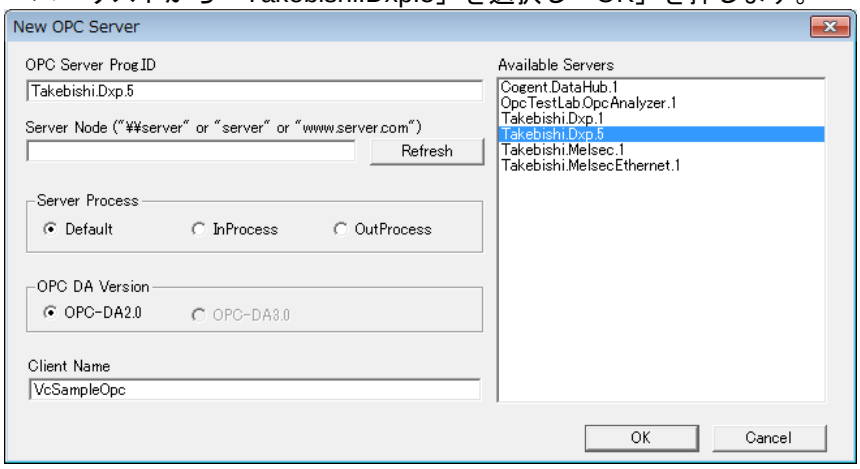
このプログラムは、Visual C++の開発環境で OPC カスタムインターフェースを使用して OPC クライアントを作成するためのサンプルプログラムです。

1.3.1　操作方法

インストール先の「¥Sample¥VcSampleOpc」フォルダー内に、Visual C++のサンプルファイルが格納されています。「Release」フォルダー内の「VcSampleOpc.exe」を実行できます。
スタートメニューから[プログラム]→[DeviceXPlorer OPC Server 5]→[OPCサンプルクライアント]を選択して起動することもできます。



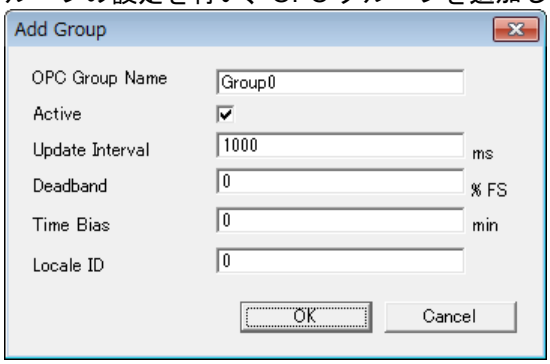
【OPC サーバーへの接続】
OPC メニューから「Connect」を選択すると下の画面が表示されます。「Available Servers」の OPC サーバーリストから「Takebishi.Dxp.5」を選択し「OK」を押します。



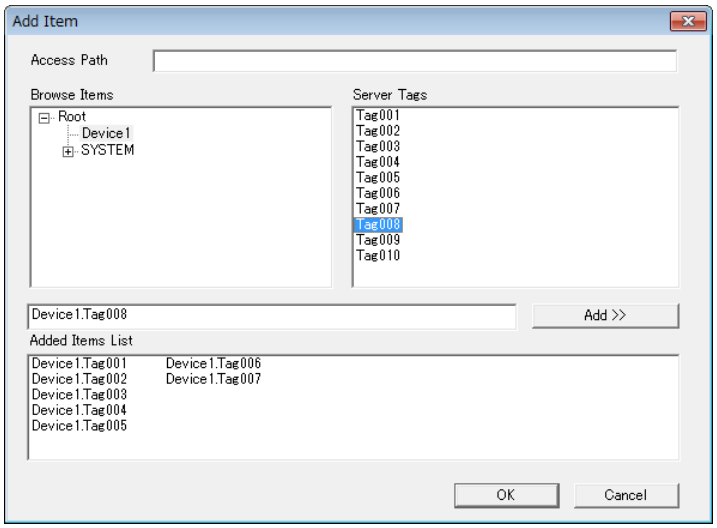
[Server Process]で接続する OPC サーバーのプロセス管理方法を選択します。[Default]を選択すると、接続する OPC サーバーの初期状態に従います。[Available servers]には OPC サーバーがリストアップされます。
デバイスエクスプローラ OPC サーバーの Prog.ID は「Takebishi.Dxp」です。その他の製品の Prog.ID は 1.1 項を参照してください。

OPC サーバーの Prog.ID が表示されない場合は、インストール先のフォルダーで、「register.bat」を実行し、OPC サーバーをレジストリに登録して下さい。

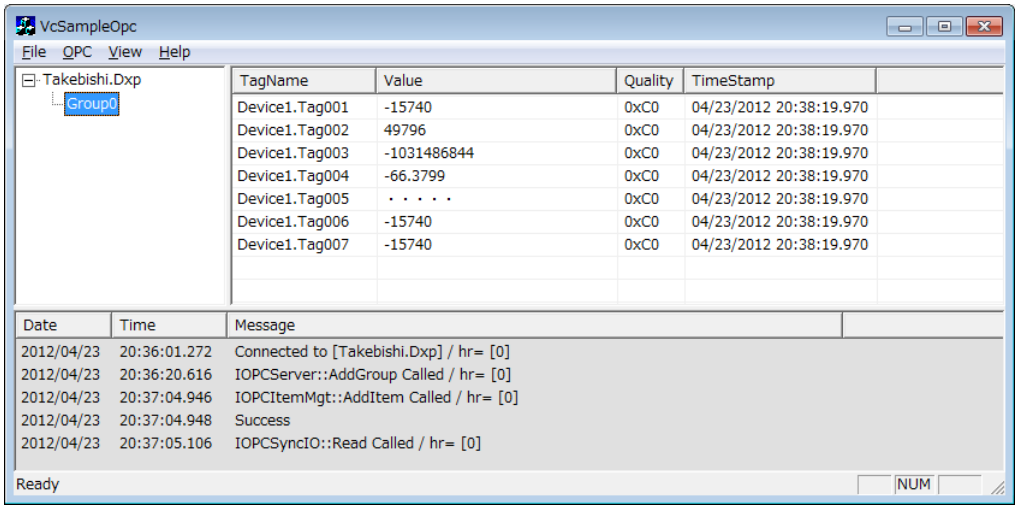
【OPC グループの追加】
接続した OPC サーバーを右クリックして「AddGroup」を選択すると下の画面が表示されます。OPC グループの設定を行い、OPC グループを追加します。



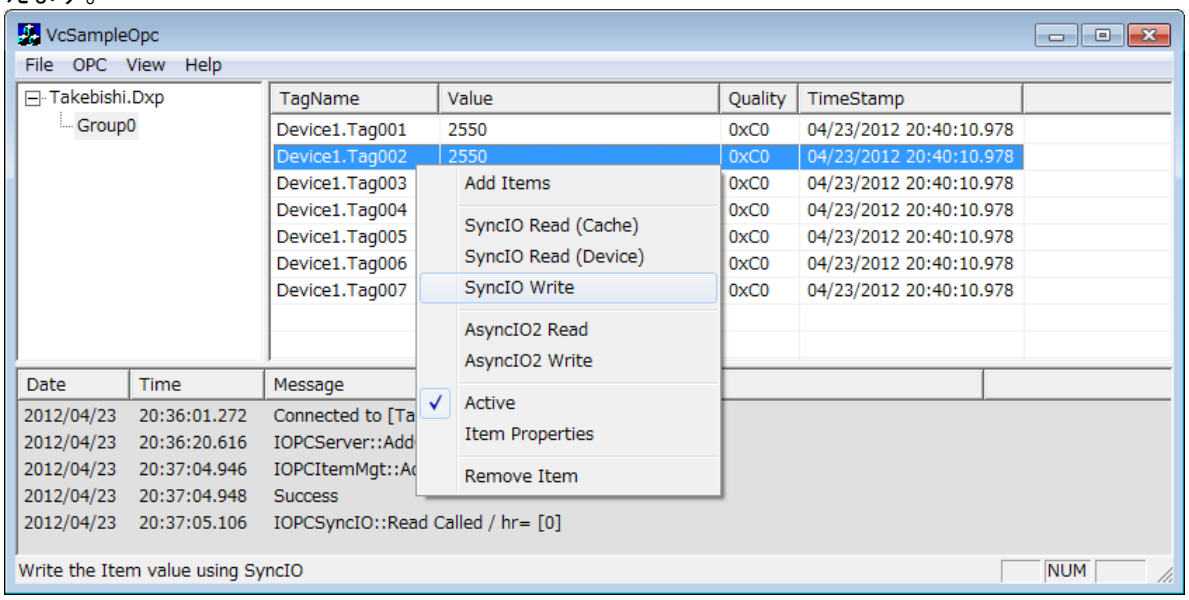
【タグの登録】
追加した OPC グループを右クリックして「Add Item」を選択すると下の画面が表示されます。「Browse Items」の階層ツリーを開き、その右側のリストに表示される定義済みタグから選択し、「Add >>」を押して、「Added Items List」に追加します。「Add >>」の左のエディットボックスに直接入力することで、ダイナミックタグの登録も可能です。
追加が完了したら、「OK」を押すと、監視対象のアイテムとして登録されます。



アイテムの追加が終了したら、[OK]ボタンをクリックして下さい。
追加されたアイテムがメインビューに表示されます。
アイテムのデータが変化すると、ビュー上の Value と Time が更新されます。



【値のモニタや書き込み】
画面上で登録タグがモニタできます。各タグを右クリックしポップアップメニューから書込操作などが行えます。

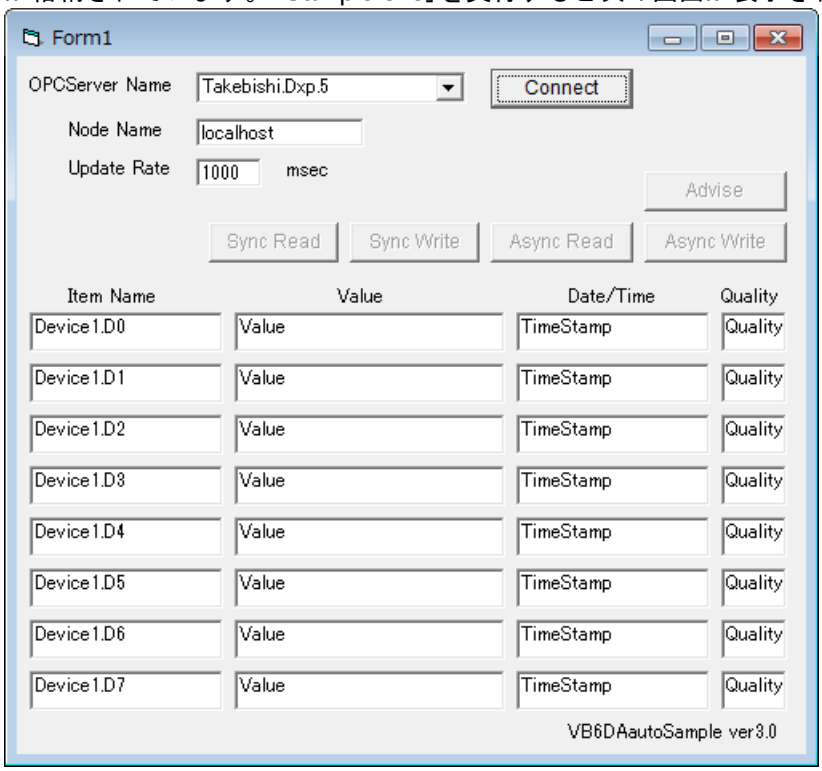


1.4 Visual Basic 6.0 サンプル（OPC オートメーションインターフェース）

このプログラムは、Visual Basic 6.0 の開発環境で OPC クライアントを作成するためのサンプルプログラムです。

1.4.1 操作方法

インストール先の「¥Sample¥VB6DAautoSample」フォルダー内に、Visual Basic 6.0 のサンプルファイルが格納されています。「Sample.exe」を実行すると次の画面が表示されます。



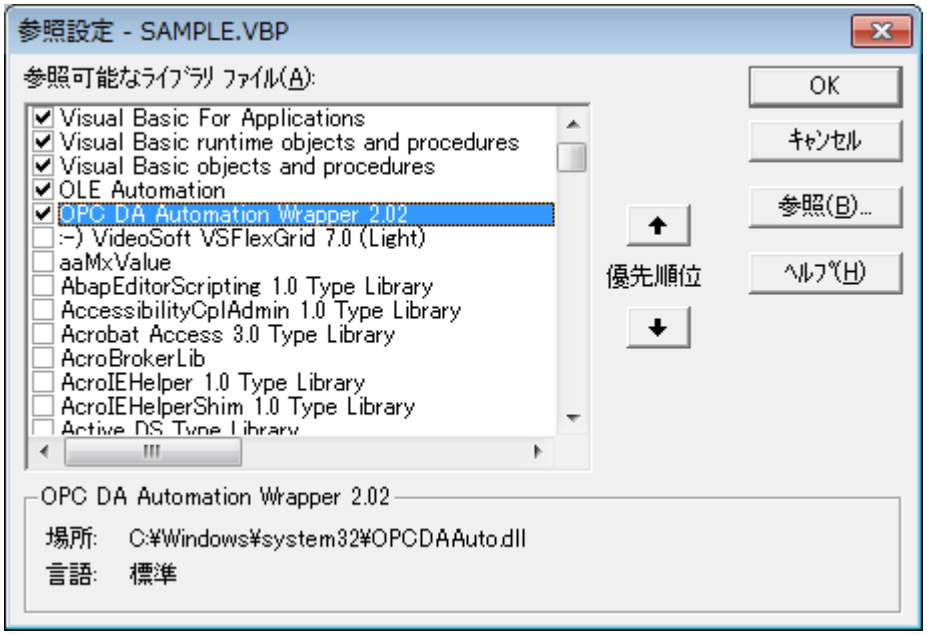
各部の機能は以下のとおりです。

機能名	内容
Node Name	接続先のノード（マシン名もしくは IP アドレス）を入力して下さい。
OPCServer Name	接続する OPC サーバー名を選択します。 デバイスエクスプローラ OPC サーバーの Prog.ID は「Takebishi.Dxp」です。その他の製品の Prog.ID は 1.1 項を参照してください。
Update Rate	PLC データの読出周期を入力して下さい。単位は「msec」です。
Item Name	OPC サーバー側で登録しているデバイス名.アイテム名を入力して下さい。デフォルトで「Device1.D0」～「Device1.D7」が入力されています。
Connect	OPC サーバーとの接続を行います。接続が成功するとボタン名称が[Disconnect]に変わります。
Read	OPC サーバーからデバイスデータを読み出し、結果（値、時刻、品質フラグ）を表示します。
Write	[Value]に入力した値を OPC サーバーに書き込みます。
Advise	OPC サーバーのグループ有効およびデータ通知が有効になります。OPC サーバーは[Update Rate]周期毎にデータを読み出し、変化があればクライアントに通知します。また、非同期読み出しや書き込みを行う場合も[Advise]を有効にしておく必要があります。
Async Read	OPC サーバーへ非同期によるデータ読み出しを要求します。OPC サーバーは非同期読み出しが完了するとクライアントに通知します。
Async Write	OPC サーバーへ非同期による Value 値の書き込みを要求します。OPC サーバーは非同期書き込みが完了するとクライアントに通知します。
Disconnect	OPC サーバーとの接続を切断します。

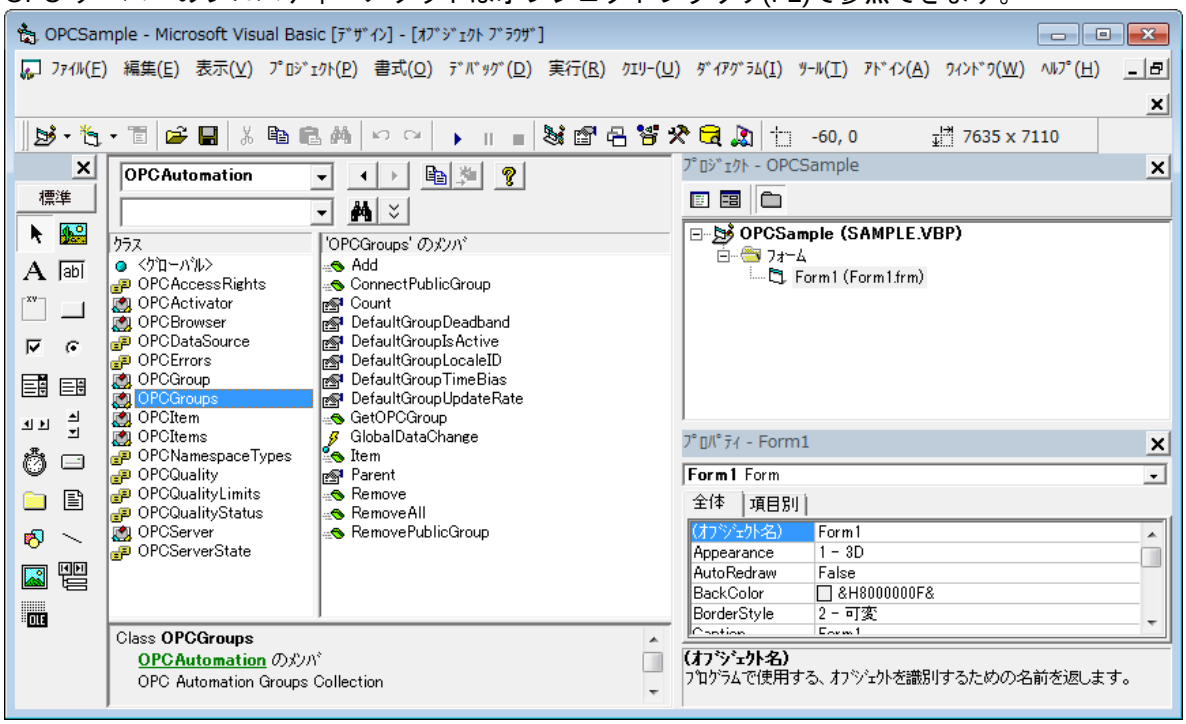
なお、接続中にOPCサーバーが終了した場合、”Server Shutdown”のメッセージボックスが表示されます。

1.4.2　開発環境の設定方法

Visual Basic 6.0 でプログラムを開発／修正する場合は、[プロジェクト]→[参照設定]で「OPC Automation 2.02」（ファイル名 OPCDAAuto.DLL）を有効にして下さい。



OPC サーバーのプロパティ・メソッドはオブジェクトブラウザ(F2)で参照できます。



1.4.3 プログラム例

Visual Basic 6.0 で OPC オートメーションインターフェースを使用したプログラムの作成概要を説明します。プログラミングの詳細および OPC オートメーションインターフェースの詳細に関しては、サンプルプログラムおよび OPCDA2.0 の仕様書を参照して下さい。
なお、OPC オートメーションインターフェースは OPCDA3.0 には対応していません。

❖ 定義部分

OPC サーバーで使用するオブジェクトを定義します。イベントを含むオブジェクト（OPC サーバーの場合、OPCServer/OPCGroups/ OPCGroup など）は WithEvents を指定し、イベントの通知を受けられるようにします。

```
Dim WithEvents OPCMyServer As OPCServer          ' OPC サーバーのオブジェクト
Dim WithEvents OPCMyGroups As OPCGroups          ' グループのコレクション
Dim WithEvents OPCMyGroup As OPCGroup            ' グループオブジェクト
Dim OPCMyItems As OPCItems                       ' アイテムのコレクション
Dim OPCMyItem As OPCItem                         ' アイテムオブジェクト
```

❖ サーバーとの接続

サーバーオブジェクトを作成し Connect メソッドを実行します。サーバー名は、OPC サーバーの Prog.ID を指定して下さい。この例では MELSEC OPC Server に接続しています。

```
Set OPCMyServer = New OPCServer                  ' サーバーオブジェクトの作成
OPCMyServer.Connect “Takebishi.Dxp.5”, ""        ' OPC サーバーの接続
```

❖ サーバーとの接続終了

OPC サーバーとの接続を終了するには、OPCGroups の Remove メソッドを実行し、OPCServer オブジェクトの Disconnect を実行して下さい。各オブジェクトは使用後、必ず削除して下さい。

```
OPCMyGroups.Remove OPCMyGroup.ServerHandle
Set OPCMyItems = Nothing                         ' アイテムのコレクション削除
Set OPCMyItem = Nothing                          ' アイテムオブジェクト削除
Set OPCMyGroups = Nothing                        ' グループのコレクション削除
Set OPCMyGroup = Nothing                         ' グループオブジェクト削除
OPCMyServer.Disconnect                           ' サーバーの切断
Set OPCMyServer = Nothing                        ' OPC サーバーのオブジェクト削除
```

❖ グループの作成

サーバーオブジェクトの OPCGroups プロパティを取得してグループを追加します。この例では、名称”Group1”で登録しています。ここで指定するグループ名は、OPC サーバー側で設定したグループとは関係ありません。

```
Set OPCMyGroups = OPCMyServer.OPCGroups          ' グループコレクション
Set OPCMyGroup = OPCMyGroups.Add("Group1")       ' グループの追加
```

❖ アイテムの追加

グループを作成した後にアイテムの追加を行います。この例ではアイテム名「Device1.D0」を 1 点 OPC サーバーに追加しています。アイテム名(OPCItemID)は、OPC サーバーで設定したデバイス・グループ・タグをピリオド(.)で区切って指定して下さい。
ClientHandle はクライアント側で識別出来るユニークな値を設定して下さい。ItemServerHandles および Errors は OPC サーバー側でセットされます。複数のアイテムを一括登録する事もできます。

```
Dim ItemServerHandles() As Long                  ‘ アイテムのハンドル
Dim ClientHandles(1) As Long                     ' クライアントハンドル
Dim OPCItemIDs(1) As String                      ' アイテム ID（デバイスの名称）
Dim Errors() As Long                             ' アイテムのエラー
Set OPCMyItems = OPCMyGroup.OPCItems             ' アイテムのコレクション
OPCItemIDs(1) = “Device1.D0”
ClientHandle(1) = 1
OPCMyItems.AddItems 1, OPCItemIDs, ClientHandles, ItemServerHandles, Errors
```

```
If Errors(1) <> 0 Then                                  ' アイテム登録でエラーがある場合
      MsgBox “Error”
End If
```

❖ 自動読出の設定

OPCGroup の IsActive プロパティを True にすると OPC サーバーは登録されたアイテムの自動読出を開始します。IsSubscription プロパティを True にすると、読み出した値が変化すれば DataChange イベントをコールバックします。DataChange イベントには、グループ内で変化の発生したアイテムだけが通知されます。どのアイテムに変化が発生したかを識別するには、ClientHandles を参照して下さい。

```
OPCMyGroup.IsActive = True
OPCMyGroup.IsSubscribed = True

Private Sub OPCMyGroup_DataChange(ByVal TransactionID As Long, ByVal NumItems As Long, _
    ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date)

‘ NumItems には、通知されたアイテムの数がセットされている。
‘ ClientHandles には、変化のあったアイテムのハンドルがセットされている。

End Sub
```

❖ 同期読出

この例は、OPCGroup の OPCItems コレクションに登録されている全てのアイテムをデバイス指定で同期読出しています。OPC サーバーからデータ値の他に時刻、品質フラグが取得できます。

```
Dim anItem As OPCItem
For Each anItem In OPCMyGroup.OPCItems
      anItem.Read OPCDevice
      Debug.Print anItem.Value, anItem.TimeStamp, anItem.Quality
      Set anItem = Nothing
Next anItem

Dim Values() As Variant
Dim Errors() As Long
Dim Qualities As Variant
Dim TimeStamps As Variant

Dim i As Integer
Dim ItemServerHandles(ItemMax) As Long        ' Server Handle
For i = 1 To ItemMax
      ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle    ' Get Server Handle
Next i

Call OPCMygroup.SyncRead(OPCDataSource.OPCDevice, ItemMax, ItemServerHandles, Values, Errors,
Qualities, TimeStamps)

For i = 1 To ItemMax
      Debug.Print Values(i), Qualities(i), TimeStamps(i), Qualities(i)
Next i
```

❖ 同期書込

この例は、OPCGroup の OPCItems コレクションに登録されている全てのアイテムに 123 を同期書込しています。

```
Dim Values(ItemMax) As Variant
Dim Errors() As Long

Dim i As Integer

Dim ItemServerHandles(ItemMax) As Long        ' Server Handle
For i = 1 To ItemMax
      ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle    ' Get Server Handle
      Values(i) =123
Next i

Call OPCMygroup.SyncWrite(ItemMax, ItemServerHandles, Values, Errors)
```

❖ 非同期読出

この例は、OPCGroup の OPCItems コレクションに登録されている全てのアイテムをデバイス指定で非同期読出しています。AsyncRead をコールすると一旦要求の受付が完了します。結果は AsyncReadComplete イベントがコールバックされます。どのアイテムの結果なのかを識別するには、ClientHandles を参照して下さい。

```
Private Sub ASYNCREAD_Button_Click()
    Dim i As Integer
    Dim TransactionID As Long
    Dim CancelID As Long
    Dim Values() As Variant
    Dim Errors() As Long

    TransactionID = 100 ' set 100(an arbitrary number) in transactionID

    Dim ItemServerHandles(ItemMax) As Long        ' Server Handle
    For i = 1 To ItemMax
        ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle   ' Get Server Handle
    Next i

    Call OPCMygroup.AsyncRead(ItemMax, ItemServerHandles, Errors, TransactionID, CancelID)
End Sub

Private Sub OPCMygroup_AsyncReadComplete(ByVal TransactionID As Long, ByVal NumItems As Long,
ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date, Errors()
As Long)
    If TransactionID = 100 Then
        MsgBox "Async Read Complete"
    End If

‘ NumItems には、AsyncRead で受け付けられたアイテムの数がセットされている。
‘ ClientHandles には、アイテムのハンドルがセットされている。
End Sub
```

❖ 非同期書込

この例は、OPCGroup の OPCItems コレクションに登録されている全てのアイテムに 123 を非同期書込しています。AsyncWrite をコールすると一旦要求の受付が完了します。結果は AsyncWriteComplete イベントがコールバックされます。どのアイテムの結果なのかを識別するには、ClientHandles を参照して下さい。

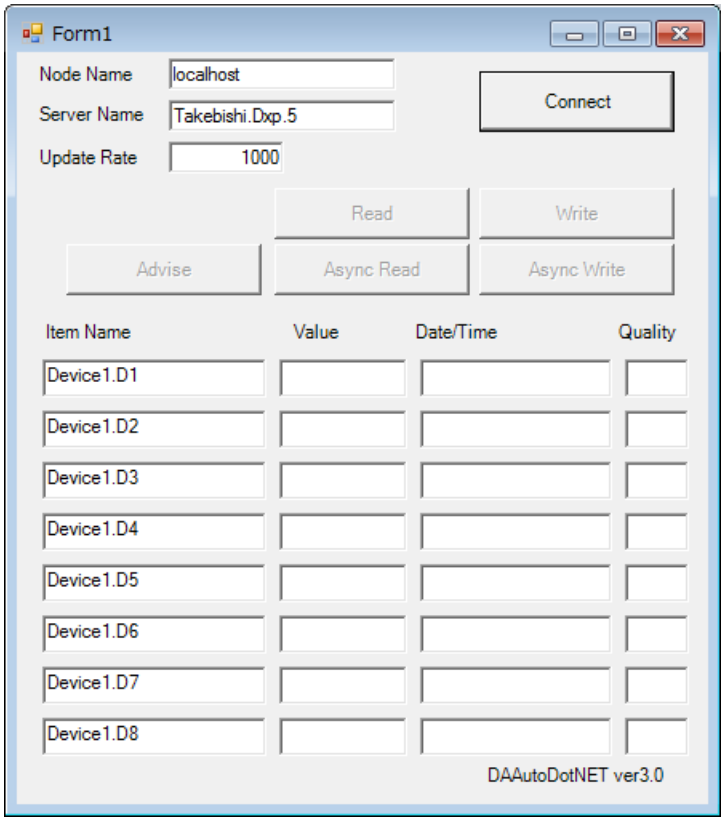
```
Private Sub ASYNCWRITE_Button_Click()
    Dim i As Integer
    Dim TransactionID As Long
    Dim CancelID As Long
    Dim Values(ItemMax) As Variant
    Dim Errors() As Long

    TransactionID = 101 ' set 101(an arbitrary number) in transactionID

    Dim ItemServerHandles(ItemMax) As Long        ' Server Handle
    For i = 1 To ItemMax
        ItemServerHandles(i) = OPCMygroup.OPCItems(i).ServerHandle   ' Get Server Handle
        Values(i) = 123
    Next i

    Call OPCMygroup.AsyncWrite(ItemMax, ItemServerHandles, Values, Errors, TransactionID,
CancelID)
End Sub

Private Sub OPCMygroup_AsyncWriteComplete(ByVal TransactionID As Long, ByVal NumItems As Long,
ClientHandles() As Long, Errors() As Long)
    If TransactionID = 101 Then
        MsgBox "Async Write Complete"
    End If

    ‘ NumItems には、AsyncRead で受け付けられたアイテムの数がセットされている。
    ‘ ClientHandles には、アイテムのハンドルがセットされている。
End Sub
```

1.5 Visual Basic .NET サンプル（OPC オートメーションインターフェース）

このプログラムは、Visual Basic .NET の開発環境で OPC オートメーションインターフェースを使用して OPC クライアントを作成するためのサンプルプログラムです。

1.5.1 操作方法

インストール先の「¥Sample¥DotNetDAautoSample」フォルダー内に、Visual Basic .NET のサンプルファイルが格納されています。「bin」フォルダー内の「DAAutoDotNET.exe」を実行すると次の画面が表示されます。



各部の機能は以下のとおりです。

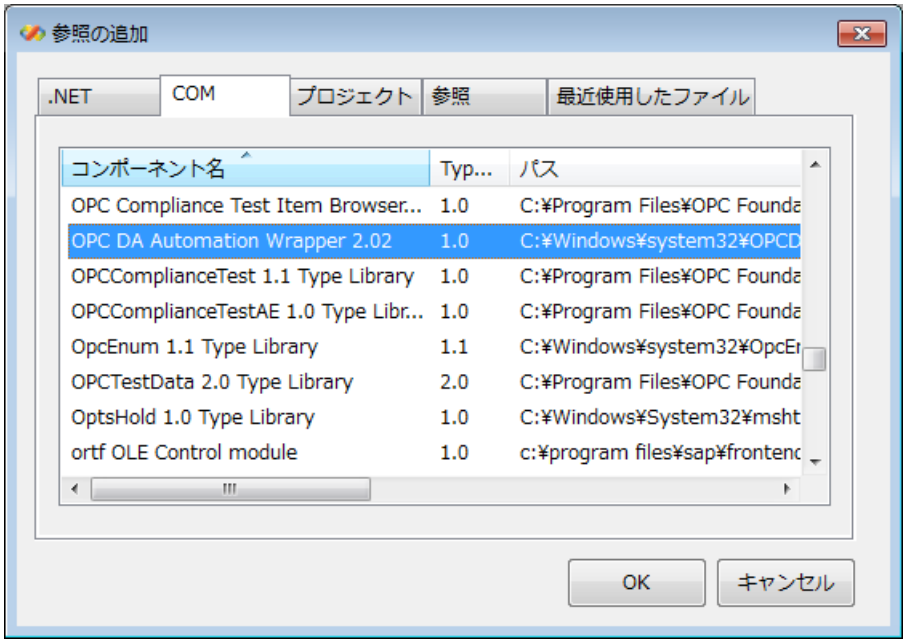
機能名	内容
Node Name	接続先のノード（マシン名もしくは IP アドレス）を入力して下さい。
Server Name	接続する OPC サーバー名を選択します。 デバイスエクスプローラ OPC サーバーの Prog.ID は「Takebishi.Dxp」です。その他の製品の Prog.ID は 1.1 項を参照してください。
Update Rate	PLC データの読出周期を入力して下さい。単位は「msec」です。
Item Name	OPC サーバー側で登録しているデバイス名.アイテム名を入力して下さい。デフォルトで「Device1.D1」～「Device1.D8」が入力されています。
Connect	OPC サーバーとの接続を行います。接続が成功するとボタン名称が[Disconnect]に変わります。
Read	OPC サーバーからデバイスデータを読み出し、結果（値、時刻、品質フラグ）を表示します。
Write	[Value]に入力した値を OPC サーバーに書き込みます。
Advise	OPC サーバーのグループ有効およびデータ通知が有効になります。OPC サーバーは[Update Rate]周期毎にデータを読み出し、変化があればクライアントに通知します。また、非同期読み出しや書き込みを行う場合も[Advise]を有効にしておく必要があります。
Async Read	OPC サーバーへ非同期によるデータ読み出しを要求します。OPC サーバーは非同期読み出しが完了するとクライアントに通知します。

Async Write	OPC サーバーへ非同期による Value 値の書き込みを要求します。OPC サーバーは非同期書き込みが完了するとクライアントに通知します。
Disconnect	OPC サーバーとの接続を切断します。

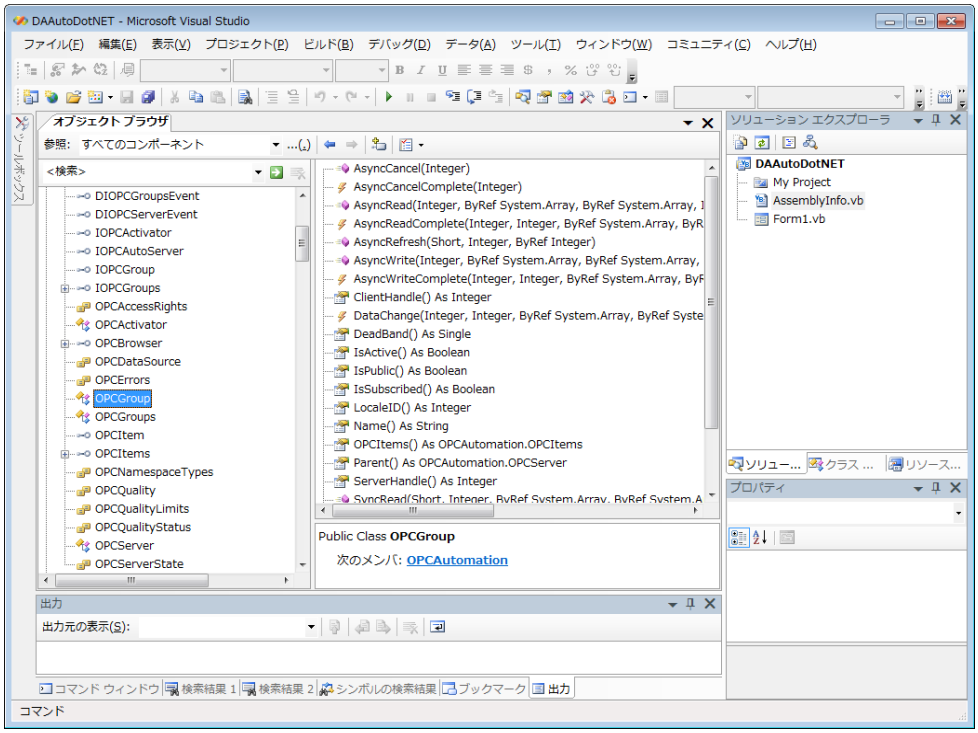
1.5.2　開発環境の設定方法

Visual Basic .NET でプログラムを開発／修正する場合は、[プロジェクト]→[参照の追加]で参照設定を行います。

[参照の追加]ダイアログが表示されますので、[COM]タブを選択して「OPCDA Automation Wrapper 2.02」（ファイル名 OPCDAAuto.DLL）を有効にして下さい。



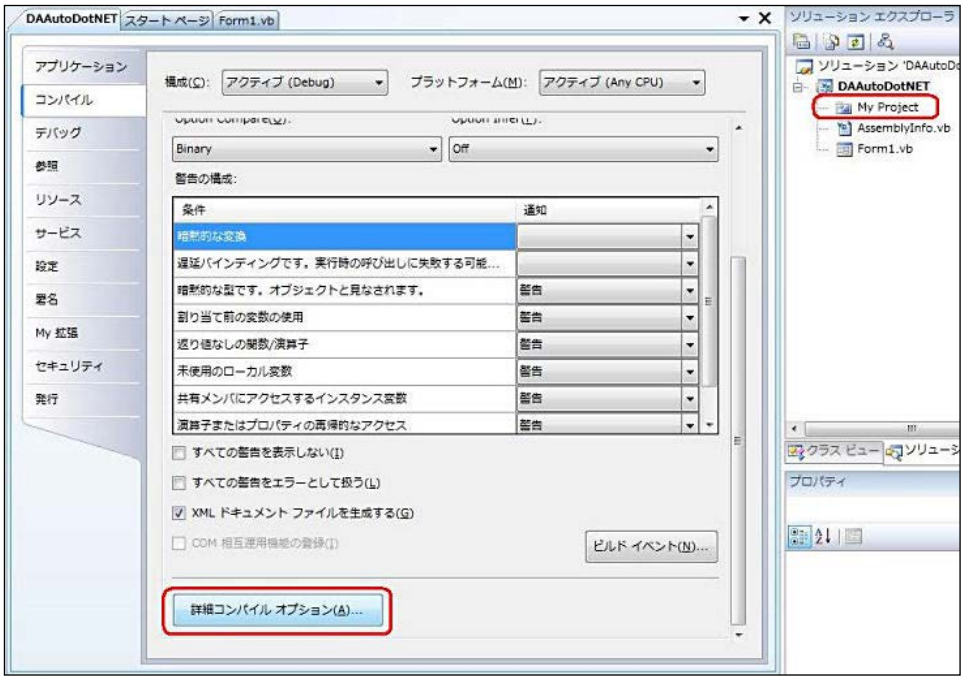
OPC サーバーのプロパティ・メソッドはオブジェクトブラウザで参照できます。



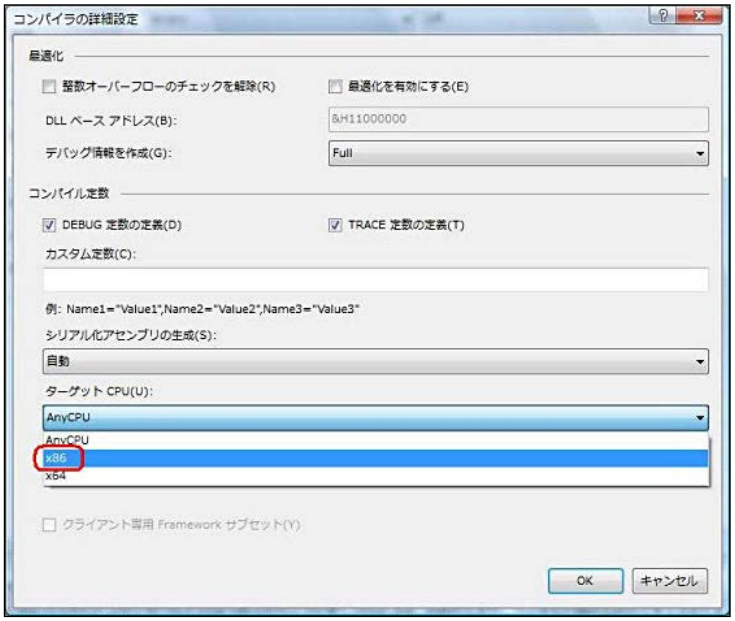
1.5.3　64 ビット Windows 上でビルドする際の注意事項

オートメーションラッパ DLL は､32 ビット用にビルドされています｡64 ビット Windows 上の VisualStudio で OPC クライアントのサンプルをビルドする際は、ターゲット CPU を x86 に設定下さい。ターゲット CPU を AnyCPU もしくは x64 に設定してビルドした場合、実行時に COM のクラス未登録の例外エラー（0x80040154）が発生します。

「プロジェクト設定 > コンパイル > 詳細コンパイルオプション」を開きます。



ターゲット CPU「x86」に設定して下さい。



1.5.4 プログラム例

Visual Basic .NET で OPC オートメーションインターフェースを使用したプログラムの作成概要を説明します。プログラミングの詳細および OPC オートメーションインターフェースの詳細に関しては、サンプルプログラムおよび OPCDA2.0 の仕様書を参照して下さい。
なお、OPC オートメーションインターフェースでは OPCDA3.0 はサポートされていません。

❖ 定義部分

OPC サーバーで使用するオブジェクトを定義します。イベントを含むオブジェクト（OPC サーバーの場合、OPCServer/OPCGroups/ OPCGroup など）は WithEvents を指定し、イベントの通知を受けられるようにします。

```
Dim WithEvents OPCMyServer As OPCAutomation.OPCServer  ' OPC サーバーのオブジェクト
Dim WithEvents OPCMyGroups As OPCAutomation.OPCGroups ' グループのコレクション
Dim WithEvents OPCMyGroup As OPCAutomation.OPCGroup   ' グループオブジェクト
Dim OPCMyItems As OPCAutomation.OPCItems              ' アイテムのコレクション
Dim OPCMyItem As OPCAutomation.OPCItem                ' アイテムオブジェクト
```

❖ サーバーとの接続

サーバーオブジェクトを作成し Connect メソッドを実行します。サーバー名は、OPC サーバーの Prog.ID を指定して下さい。この例では MELSEC OPC Server に接続しています。

```
Set OPCMyServer = New OPCAutomation.OPCServer ' サーバーオブジェクトの作成
OPCMyServer.Connect("Takebishi.Dxp.5", "")    ' OPC サーバーの接続
```

❖ グループの作成

サーバーオブジェクトの OPCGroups プロパティを取得し、グループを追加します。この例では、名称"Group1"で登録しています。ここで指定するグループ名は、OPC サーバー側で設定したグループとは関係ありません。

```
Set OPCMyGroups = OPCMyServer.OPCGroups      ' グループコレクション
Set OPCMyGroup = OPCMyGroups.Add("Group1")   ' グループの追加
```

❖ アイテムの追加

グループを作成した後にアイテムの追加を行います。この例では、アイテム名「Device1.D0」を 1 点 OPC サーバーに追加しています。アイテム名(OPCItemID)は、OPC サーバーで設定したデバイス・グループ・タグをピリオド(.)で区切って指定して下さい。
ClientHandle はクライアント側で識別出来るユニークな値を設定して下さい。ItemServerHandles および Errors は OPC サーバー側でセットされます。複数のアイテムを一括登録する事もできます。

```
Dim ItemServerHandles As Array                ' アイテムのハンドル
Dim ClientHandles(1) As Long                  ' クライアントハンドル
Dim OPCItemIDs(1) As String                   ' アイテム ID（デバイスの名称）
Dim Errors As Array                           ' アイテムのエラー
Set OPCMyItems = OPCMyGroup.OPCItems          ' アイテムのコレクション
OPCItemIDs(1) = "Device1.D0"
ClientHandle(1) = 1
OPCMyItems.AddItems(1, OPCItemIDs, ClientHandles, ItemServerHandles, Errors)
If Errors(1) <> 0 Then                        ' アイテム登録でエラーがある場合
      MsgBox "Error"
End If
```

❖ 自動読出の設定

OPCGroup の IsActive プロパティを True にすると OPC サーバーは登録されたアイテムの自動読出を開始します。IsSubscription プロパティを True にすると、読み出した値が変化すれば DataChange イベントをコールバックします。DataChange イベントには、グループ内で変化の発生したアイテムだけが通知されます。どのアイテムに変化が発生したかを識別するには、ClientHandles を参照して下さい。

```
OPCMyGroup.IsActive = True
OPCMyGroup.IsSubscribed = True

Private Sub OPCMyGroup_DataChange(ByVal TransactionID As Integer, _
                        ByVal NumItems As Integer, _
                        ByRef ClientHandles As System.Array, _
                        ByRef ItemValues As System.Array, _
                        ByRef Qualities As System.Array, _
                        ByRef TimeStamps As System.Array) Handles OPCMyGroup.DataChange

‘ NumItems には、通知されたアイテムの数がセットされている。
‘ ClientHandles には、変化のあったアイテムのハンドルがセットされている。

End Sub
```

❖ 読み出し

この例は、OPCGroup の OPCItems コレクションに登録されている全てのアイテムをデバイス指定で読み出しています。この他に OPCGroup には、SyncRead、AsyncRead の読み出しメソッドがあります。これらを使用すると、複数アイテムのデータ読み出し、非同期読み出しができます。OPC サーバーからデータ値の他に時刻、品質フラグが取得できます。

```
Dim anItem As OPCAutomation.OPCItem
For Each anItem In OPCMyGroup.OPCItems
    anItem.Read(OPCAutomation.OPCDataSource.OPCDevice)
    Set anItem = Nothing
Next anItem
```

❖ 書き込み

この例は、OPCGroup の OPCItems コレクションに登録されている全てのアイテムに 123 を書き込んでいます。この他に OPCGroup には、SyncWrite、AsyncWrite の書き込みメソッドがあります。これらを使用すると、複数アイテムのデータ書き込み、非同期書き込みができます。

```
Dim anItem As OPCAutomation.OPCItem
For Each anItem In OPCMyGroup.OPCItems
     anItem.Write(123)
     Set anItem = Nothing
Next anItem
```

❖ サーバーとの接続終了

OPC サーバーとの接続を終了するには、OPCGroups の Remove メソッドを実行し、OPCServer オブジェクトの Disconnect を実行して下さい。各オブジェクトは使用後、必ず削除して下さい。

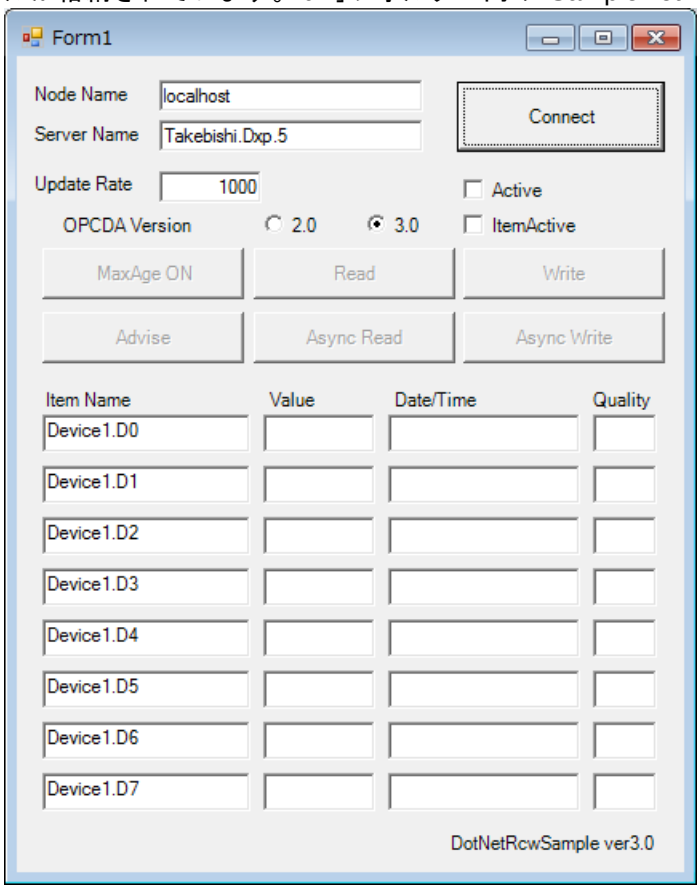
```
OPCMyGroups.Remove(OPCMyGroup.ServerHandle)
Set OPCMyItems = Nothing                    ' アイテムのコレクション削除
Set OPCMyItem = Nothing                     ' アイテムオブジェクト削除
Set OPCMyGroups = Nothing                   ' グループのコレクション削除
Set OPCMyGroup = Nothing                    ' グループオブジェクト削除
OPCMyServer.Disconnect()                    ' サーバーの切断
Set OPCMyServer = Nothing                   ' OPC サーバーのオブジェクト削除
```

1.6 Visual Basic .NET サンプル（RCW インターフェース）

このプログラムは、Visual Basic .NET の開発環境で OPC カスタムインターフェース(RCW)を使用して OPC クライアントを作成するためのサンプルプログラムです。

1.6.1 操作方法

インストール先の「¥ Sample¥DotNetRcwSample」フォルダー内に、Visual Basic .NET のサンプルファイルが格納されています。「bin」フォルダー内の「SampleDotNET.exe」を実行すると次の画面が表示されます。



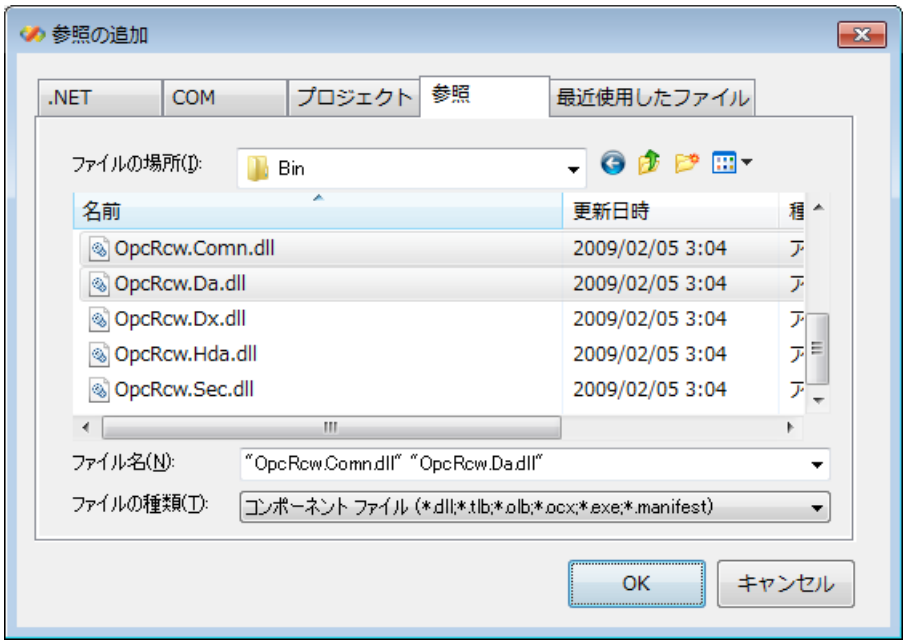
各部の機能は以下のとおりです。

機能名	内容
Node Name	接続先のノード（マシン名もしくは IP アドレス）を入力して下さい。
Server Name	接続する OPC サーバー名を選択します。 デバイスエクスプローラ OPC サーバーの Prog.ID は「Takebishi.Dxp」です。その他の製品の Prog.ID は 1.1 項を参照してください。
Update Rate	PLC データの読出周期を入力して下さい。単位は「msec」です。
Active	グループのアクティブ状態を指定します。
ItemActive	アイテムのアクティブ状態を指定します。
OPCDA Version	使用する OPC DA インターフェースのバージョンを選択します。
Item Name	OPC サーバー側で登録しているデバイス名.アイテム名を入力して下さい。デフォルトで「Device1.D0」～「Device1.D7」が入力されています。
Connect	OPC サーバーとの接続を行います。接続が成功するとボタン名称が[Disconnect]に変わります。
MaxAge On	アイテム毎にデータの"古さ"を指定した値の読み出しを 1 秒周期で行います。当サンプルでの MaxAge 設定時間("古さ"の指定時間)は 10000msec ですので、現在より 10000msec 前までに更新されているデータはキャッシュ読み出しを、それより古いデータ(10000msec 以上経過しているデータ)はデバイス読み出しを行います。
Read	OPC サーバーからデバイスデータを読み出し、結果（値、時刻、品質フラグ）を表示

	します。
Write	[Value]に入力した値を OPC サーバーに書き込みます。
Advise	OPC サーバーのグループ有効およびデータ通知が有効になります。OPC サーバーは[Update Rate]周期毎にデータを読み出し、変化があればクライアントに通知します。また、非同期読み出しや書き込みを行う場合も[Advise]を有効にしておく必要があります。
Async Read	OPC サーバーへ非同期によるデータ読み出しを要求します。OPC サーバーは非同期読み出しが完了するとクライアントに通知します。
Async Write	OPC サーバーへ非同期による Value 値の書き込みを要求します。OPC サーバーは非同期書き込みが完了するとクライアントに通知します。
Disconnect	OPC サーバーとの接続を切断します。

1.6.2　開発環境の設定方法

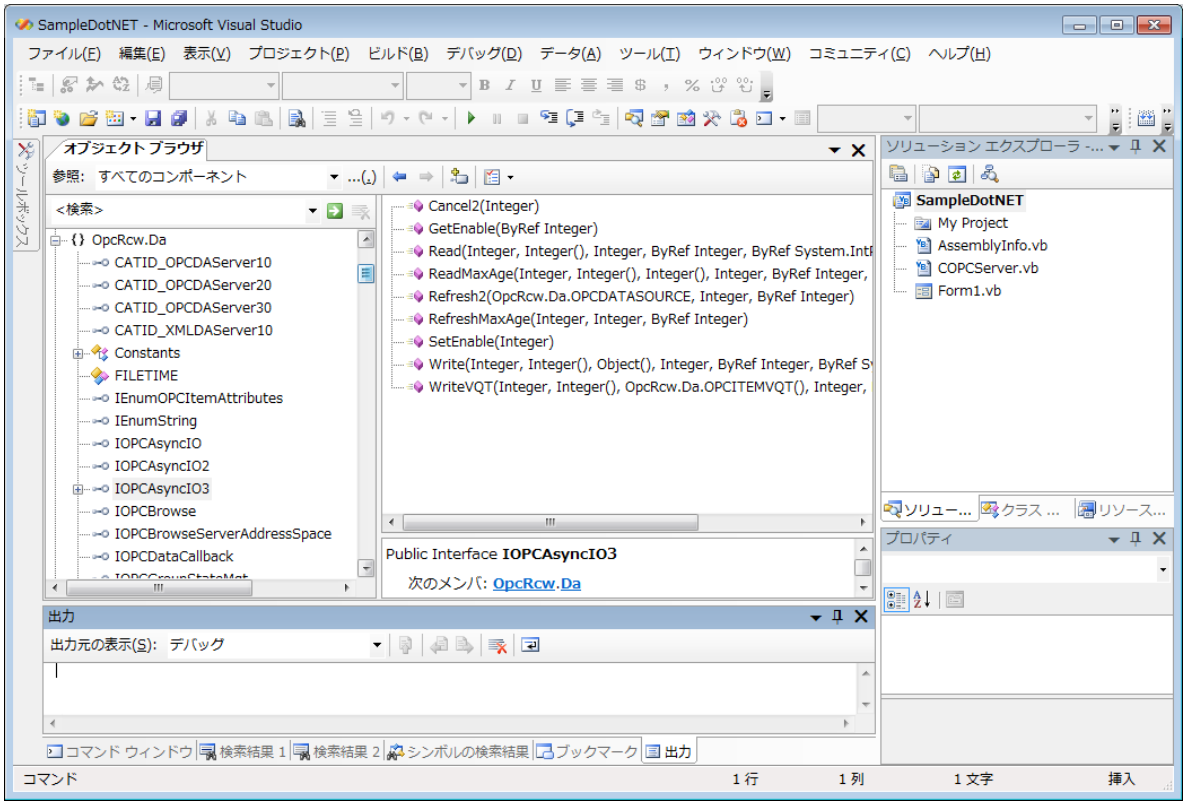
OPC サーバーは COM(Component Object Model)ベースで動作しますので、.NET 環境からアクセスする場合、COM と.NET の変換が必要になります。
予め OPC Core Components SDK をインストールして下さい。
OPC Core Components SDK は OPC Foundation のホームページからダウンロードできます。
http://www.opcfoundation.org/
Visual Basic .NET でプログラムを開発／修正する場合は、Visual Basic .NET の[プロジェクト]→[参照の追加]で参照設定を行います。
[参照の追加]ダイアログが表示されますので、[参照]ボタンをクリックして以下ファイルを選択して下さい。
＜OPC Foundation インストールフォルダー＞¥Bin¥OpcRcw.Comn.dll
＜OPC Foundation インストールフォルダー＞¥Bin¥OpcRcw.Da.dll



上記ファイルを選択すると、以下コンポーネントが有効になります。
OpcRcw.Comn
OpcRcw.Da

1.6 Visual Basic .NET サンプル（RCW インターフェース）

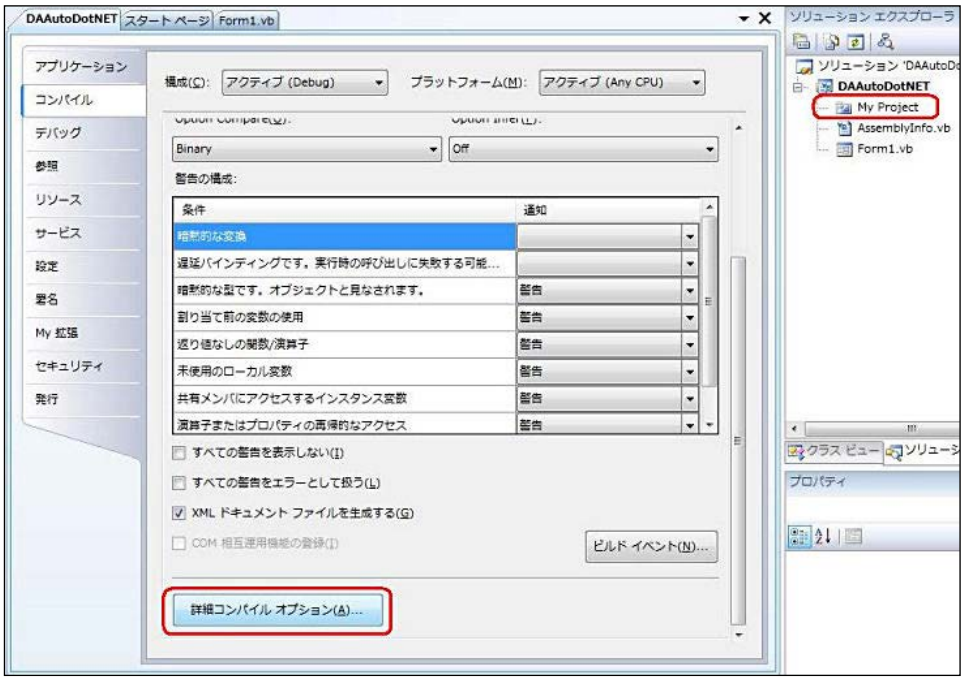
OPC サーバーのプロパティ・メソッドはオブジェクトブラウザで参照できます。



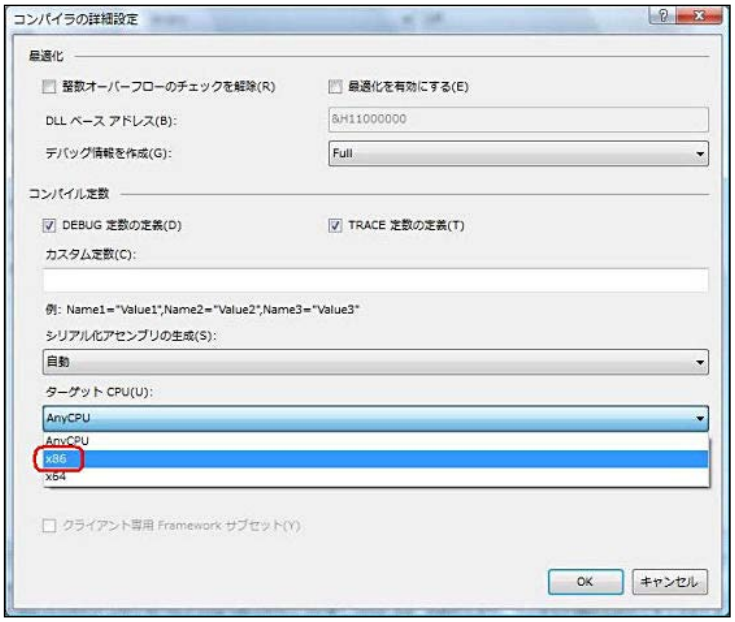
1.6.3　64 ビット Windows 上でビルドする際の注意事項

RCW の DLL は 32 ビット用にビルドされています。64 ビット Windows 上の VisualStudio で OPC クライアントのサンプルをビルドする際は、ターゲット CPU を x86 に設定下さい。ターゲット CPU を AnyCPU もしくは x64 に設定してビルドした場合、予期しない動作となります。

「プロジェクト設定 > コンパイル > 詳細コンパイルオプション」を開きます。



ターゲット CPU を「x86」に設定して下さい。



1.6.4　プログラム例

Visual Basic .NET で OPC カスタムインターフェースを使用したプログラムの作成概要を説明します。本項ではサンプルプログラムで使用する COPCServer クラスについて説明します。プログラミングの詳細および OPC カスタムインターフェースの詳細に関しては、サンプルプログラムおよび OPCDA3.0 の仕様書を参照して下さい。

❖ 概要

OPC サーバーとの通信処理を行うユーザークラスです。また、OPC サーバーからのイベントを取得するため、IOPCDataCallback インターフェースをインプリメントしています。

❖ 定義部分

OPC サーバーで使用するオブジェクトを定義します。

```
Private m_OPCServer As IOPCServer                    ' サーバーオブジェクト
Private m_OPCGroup As IOPCGroupStateMgt2             ' グループオブジェクト
Private m_OPCItem As IOPCItemMgt                     ' アイテムオブジェクト
Private m_OPCConnPointCntnr As IConnectionPointContainer
                                                     ' コネクションポイントコンテナオブジェクト
Private m_OPCConnPoint As IConnectionPoint            ' コネクションポイントオブジェクト
```

❖ サーバーとの接続

サーバーオブジェクトを作成しサーバーとの接続を行います。OPC サーバーの Prog.ID を指定して下さい。この例では MELSEC OPC Server に接続しています。

```
m_OPCServer = CreateObject("Takebishi.Dxp.5")
```

❖ グループの作成

サーバーオブジェクトの AddGroup メソッドを実行してグループを追加します。また、サーバーからのコールバックを有効にするため、COPCCallBack クラスにコネクションポイントを割り当てます。

```
guidGroupStateMgt = Marshal.GenerateGuidForType(GetType(IOPCGroupStateMgt2))
m_OPCServer.AddGroup("Group1", _                     ' グループ名
                     True, _                         ' True : アクティブ状態、False : 非アクティブ状態
                     1000, _                         ' 更新周期
                     1234, _                         ' クライアント側で指定するグループ ID
                     ptrTimeBias, _                  ' 時刻へのポインタ
                     ptrDeadBand, _                  ' デッドバンドへのポインタ
                     0, _                            ' 言語 ID
                     m_iServerGroup, _               ' サーバー側で指定するグループ ID
                     iRevisedUpdateRate, _           ' サーバーからの更新周期へのポインタ
                     guidGroupStateMgt, _            ' IOPCGroupStateMgt の GUID
                     m_OPCGroup)                     ' グループオブジェクト

m_OPCConnPointCntnr = CType(m_OPCGroup, IConnectionPointContainer)
guidDataCallback = Marshal.GenerateGuidForType(GetType(IOPCDataCallback))
m_OPCConnPointCntnr.FindConnectionPoint(guidDataCallback, m_OPCConnPoint)
```

❖ アイテムの追加

グループを作成した後にアイテムを追加します。この例では、アイテム名「Device1.D0」を 1 点 OPC サーバーに追加しています。

OPCITEMDEF 構造体には登録に必要な情報を格納します。szItemID は、OPC サーバーで設定したデバイス・グループ・タグをピリオド(.)で区切って指定して下さい。bActive はアイテム毎にアクティブ状態を設定して下さい。hClient にはクライアント側で識別できるユニークな値を設定して下さい。

ppResult および ppErrors は OPC サーバー側でセットされます。ppResult は OPCITEMRESULT 構造体のポインタで、hServer にはサーバー側でアイテムを識別する ID が格納されています。最後に ppResult および ppErrors のメモリを解放して下さい。

また、複数のアイテムを一括登録する事もできます。

```
itemDef(1).szItemID = "Device1.D0"                    ' アイテム名
itemDef(1).bActive = True                             ' True：アクティブ状態、False：非アクティブ状態
itemDef(1).hClient = 1                                ' クライアント側で指定するアイテム ID
m_OPCItem = CType(m_OPCGroup, IOPCItemMgt)
m_OPCItem.AddItems(1, itemDef, ppResult, ppErrors)

itemResult= CType(Marshal.PtrToStructure(ppResult, GetType(OPCITEMRESULT)), OPCITEMRESULT)
ServerHd = itemResult.hServer
Marshal.FreeCoTaskMem(ppResult)
Marshal.FreeCoTaskMem(ppErrors)
```

❖ コールバックの有効化

コネクションポイントオブジェクトの Advise メソッドでコールバックを有効にします。自動読み出しや非同期読み出し、非同期書き込みを行う場合はコールバックを有効にして下さい。

```
m_OPCConnPoint.Advise(Me, m_iCallBackConnection)
```

❖ 読み出し

この例は、グループオブジェクトを OPCSyncIO2 インターフェースにキャストして 1 アイテムをデバイス指定で同期読み出ししています。OPC サーバーからデータ値の他に時刻、品質フラグが取得できます。最後に ppItemVal および ppErrors のメモリを解放して下さい。
この他に OPCAsyncIO3 インターフェースにキャストして非同期読み出しする方法があります。

```
OPCSyncIO = CType(m_OPCGroup, IOPCSyncIO2)
OPCSyncIO.Read(OPCDATASOURCE.OPC_DS_DEVICE, 1, ServerHd, ppItemVal, ppErrors)
                                                      ' 同期読み出し
ItemState = CType(Marshal.PtrToStructure(ppItemVal, GetType(OPCITEMSTATE)), OPCITEMSTATE)
vValue = ItemState.vDataValue                         ' 値を取得
fTimeStamp = Date.FromFileTime(ItemState.ftTimeStamp.dwHighDateTime * 2 ^ 32 + _
                               ItemState.ftTimeStamp.dwLowDateTime)
                                                      ' 時刻を取得
wQuality = ItemState.wQuality                         ' 品質フラグを取得
Marshal.FreeCoTaskMem(ppItemVal)
Marshal.FreeCoTaskMem(ppErrors)
```

❖ 書き込み

この例は、グループオブジェクトを OPCSyncIO2 インターフェースにキャストして 1 アイテムを同期書き込みしています。最後に ppErrors のメモリを解放して下さい。
この他に OPCAsyncIO3 インターフェースにキャストして非同期書き込みする方法があります。

```
OPCSyncIO = CType(m_OPCGroup, IOPCSyncIO2)
vValue = 12345
OPCSyncIO.Write(1, ServerHd, vValue, ppErrors)        ' 同期書き込み
Marshal.FreeCoTaskMem(ppErrors)
```

❖ データ変化通知

OPC サーバーからデバイス内で変化の発生したアイテムが通知されます。どのアイテムに変化が発生したかを識別するには phClientItems を参照して下さい。

```
Sub OnDataChange(ByVal dwTransid As Integer, _
                 ByVal hGroup As Integer, _
                 ByVal hrMasterquality As Integer, _
                 ByVal hrMastererror As Integer, _
                 ByVal dwCount As Integer, _              ' 変化の発生したアイテム数
                 ByVal phClientItems() As Integer, _      ' クライアント側で設定したアイテム ID
                 ByVal pvValues() As Object, _            ' データ値
                 ByVal pwQualities() As Short, _          ' 品質フラグ
                 ByVal pftTimeStamps() As OpcRcw.Da.FILETIME, _        ' 時刻
                 ByVal pErrors() As Integer) _            ' エラー
                 Implements IOPCDataCallback.OnDataChange

    RaiseEvent DataChange(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, _
                          pwQualities, pErrors)           ' イベント発生
End Sub
```

❖ 非同期読み出し完了通知

OPC サーバーから非同期読み出しが完了すると通知されます。読み出しを行ったアイテムを識別するには phClientItems を参照して下さい。

```
Sub OnReadComplete(ByVal dwTransid As Integer, _
            ByVal hGroup As Integer, _
            ByVal hrMasterquality As Integer, _
            ByVal hrMastererror As Integer, _
            ByVal dwCount As Integer, _          ' 読み出しを行ったアイテム数
            ByVal phClientItems() As Integer, _  ' クライアント側で設定したアイテム ID
            ByVal pvValues() As Object, _        ' データ値
            ByVal pwQualities() As Short, _      ' 品質フラグ
            ByVal pftTimeStamps() As OpcRcw.Da.FILETIME, _      ' 時刻
            ByVal pErrors() As Integer) _        ' エラー
            Implements IOPCDataCallback.OnReadComplete

RaiseEvent ReadComplete(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, _
            pwQualities, pErrors)                ' イベント発生
End Sub
```

❖ 非同期書き込み完了通知

OPC サーバーから非同期書き込みが完了すると通知されます。書き込みを行ったアイテムを識別するには phClientItems を参照して下さい。

```
Sub OnWriteComplete(ByVal dwTransid As Integer, _
            ByVal hGroup As Integer, _
            ByVal hrMastererr As Integer, _
            ByVal dwCount As Integer, _          ' 書き込みを行ったアイテム数
            ByVal phClientItems() As Integer, _  ' クライアント側で設定したアイテム ID
            ByVal pErrors() As Integer) _        ' エラー
            Implements IOPCDataCallback.OnWriteComplete

RaiseEvent WriteComplete(dwTransid, dwCount, phClientItems, pErrors) ' イベント発生
End Sub
```

❖ 非同期処理のキャンセル完了通知

OPC サーバーから非同期処理のキャンセルが完了すると通知されます。

```
Sub OnCancelComplete(ByVal dwTransid As Integer, _
            ByVal hGroup As Integer) _
            Implements IOPCDataCallback.OnCancelComplete

RaiseEvent CancelComplete(dwTransid)    ' イベント発生
End Sub
```

❖ サーバーとの接続終了

OPC サーバーとの接続を終了するには、OPCServer の RemoveGroup メソッドを実行して下さい。各オブジェクトは使用後、必ず削除して下さい。

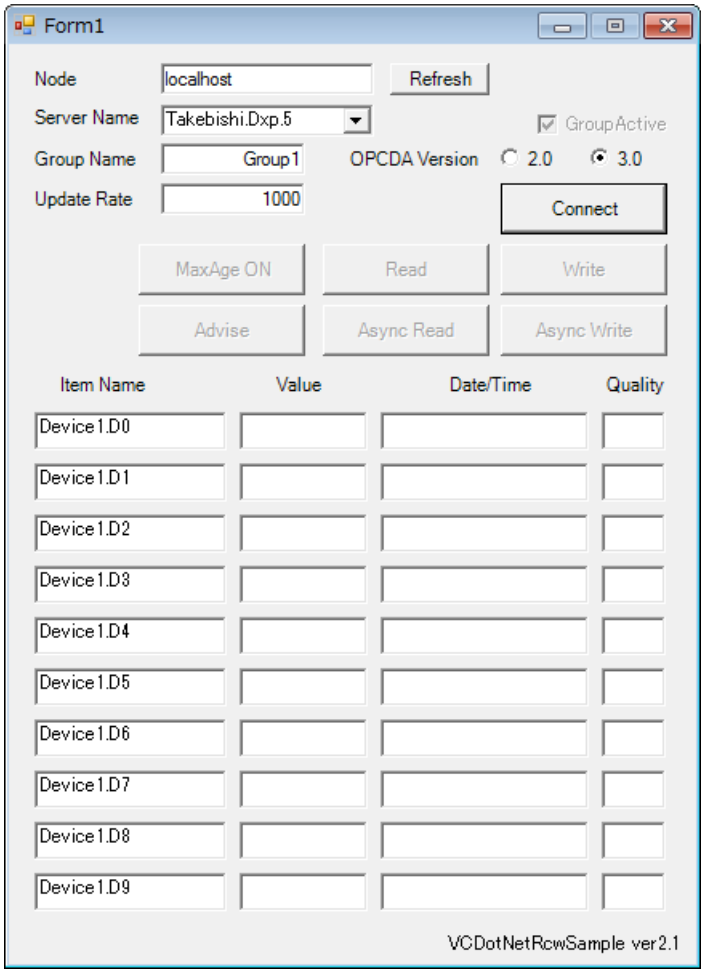
```
If m_iServerGroup <> 0 Then
    m_OPCServer.RemoveGroup(m_iServerGroup, False)
    ret = Marshal.ReleaseComObject(m_OPCGroup)
    m_iServerGroup = 0
End If

ret = Marshal.ReleaseComObject(m_OPCServer)
m_OPCGroup = Nothing
m_OPCServer = Nothing
```

1.7 Visual C# サンプル（OPC カスタムインターフェース）

このプログラムは、Visual C#の開発環境で OPC カスタムインターフェースを使用して OPC クライアントを作成するためのサンプルプログラムです。

1.7.1 操作方法

インストール先の「¥Sample¥VCDotNetRcwSample」フォルダー内に、Visual C#のサンプルファイルが格納されています。「¥bin¥Release」フォルダー内の「VCDotNetRcwSample.exe」を実行すると次の画面が表示されます。



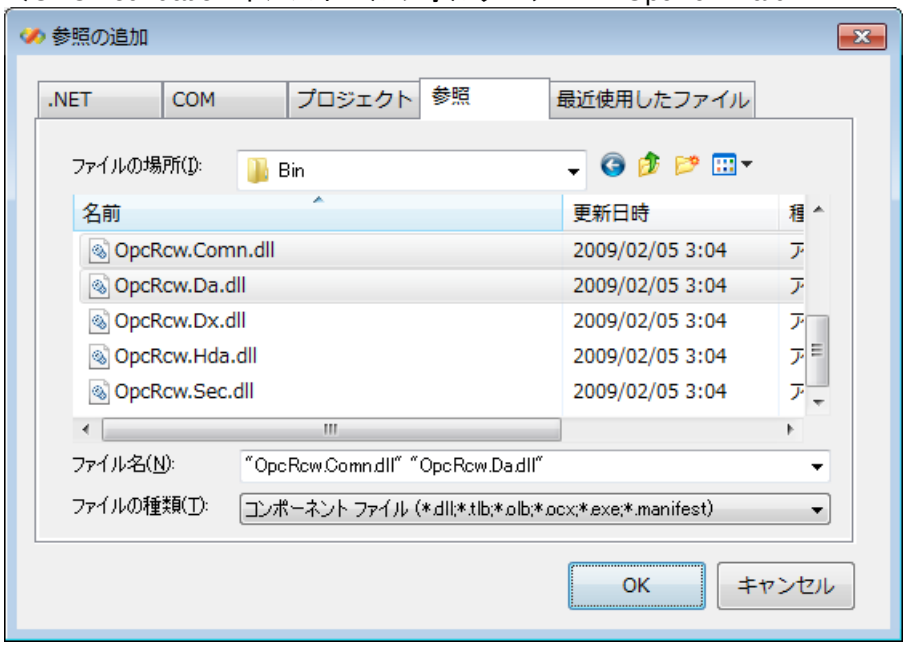
各部の機能は以下のとおりです。

機能名	内容
Node Name	接続先のノード（マシン名もしくは IP アドレス）を入力して下さい。
Server Name	接続する OPC サーバー名を選択します。 デバイスエクスプローラ OPC サーバーの Prog.ID は「Takebishi.Dxp」です。その他の製品の Prog.ID は 1.1 項を参照してください。
Update Rate	PLC データの読出周期を入力して下さい。単位は「msec」です。
GroupActive	グループのアクティブ状態を指定します。
OPCDA Version	使用する OPC DA インターフェースのバージョンを選択します。
Item Name	OPC サーバー側で登録しているデバイス名.アイテム名を入力して下さい。デフォルトで「Device1.D0」～「Device1.D9」が入力されています。
Connect	OPC サーバーとの接続を行います。接続が成功するとボタン名称が[Disconnect]に変わります。
MaxAge On	アイテム毎にデータの"古さ"を指定した値の読み出しを 1 秒周期で行います。当サンプルでの MaxAge 設定時間("古さ"の指定時間)は 10000msec ですので、現在より 10000msec 前までに更新されているデータはキャッシュ読み出しを、それより古いデ

	ータ(10000msec 以上経過しているデータ)はデバイス読み出しを行います。
Read	OPC サーバーからデバイスデータを読み出し、結果（値、時刻、品質フラグ）を表示します。
Write	[Value]に入力した値を OPC サーバーに書き込みます。
Advise	OPC サーバーのグループ有効およびデータ通知が有効になります。OPC サーバーは[Update Rate]周期毎にデータを読み出し、変化があればクライアントに通知します。また、非同期読み出しや書き込みを行う場合も[Advise]を有効にしておく必要があります。
Async Read	OPC サーバーへ非同期によるデータ読み出しを要求します。OPC サーバーは非同期読み出しが完了するとクライアントに通知します。
Async Write	OPC サーバーへ非同期による Value 値の書き込みを要求します。OPC サーバーは非同期書き込みが完了するとクライアントに通知します。
Disconnect	OPC サーバーとの接続を切断します。

1.7.2　開発環境の設定方法

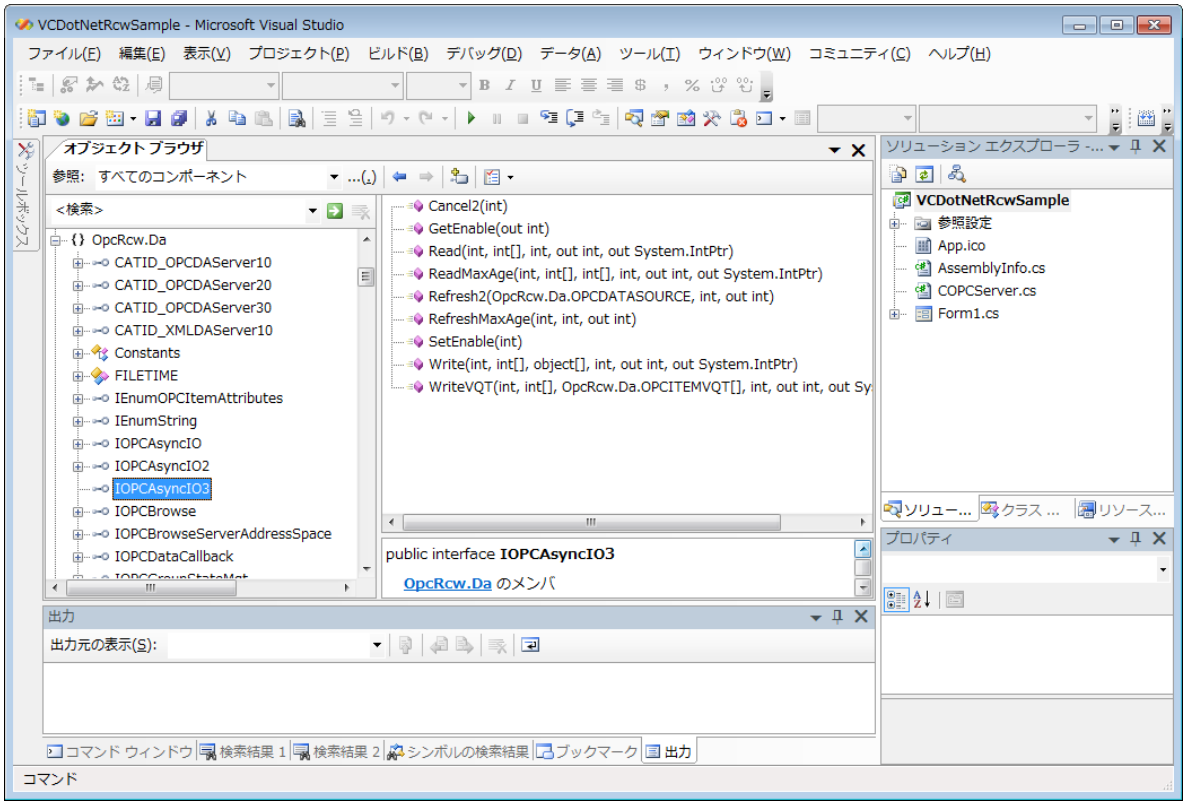
OPC サーバーは COM(Component Object Model)ベースで動作しますので、.NET 環境からアクセスする場合、COM と.NET の変換が必要になります。
予め OPC Core Components SDK をインストールして下さい。
OPC Core Components SDK は OPC Foundation のホームページからダウンロードできます。
http://www.opcfoundation.org/
Visual C#でプログラムを開発／修正する場合は、Visual C#の[プロジェクト]→[参照の追加]で参照設定を行います。
[参照の追加]ダイアログが表示されますので、[参照]ボタンをクリックして以下ファイルを選択して下さい。
＜OPC Foundation インストールフォルダー＞¥Bin¥OpcRcw.Comn.dll
＜OPC Foundation インストールフォルダー＞¥Bin¥OpcRcw.Da.dll



上記ファイルを選択すると、以下コンポーネントが有効になります。
OpcRcw.Comn
OpcRcw.Da

1.7 Visual C# サンプル（OPC カスタムインターフェース）

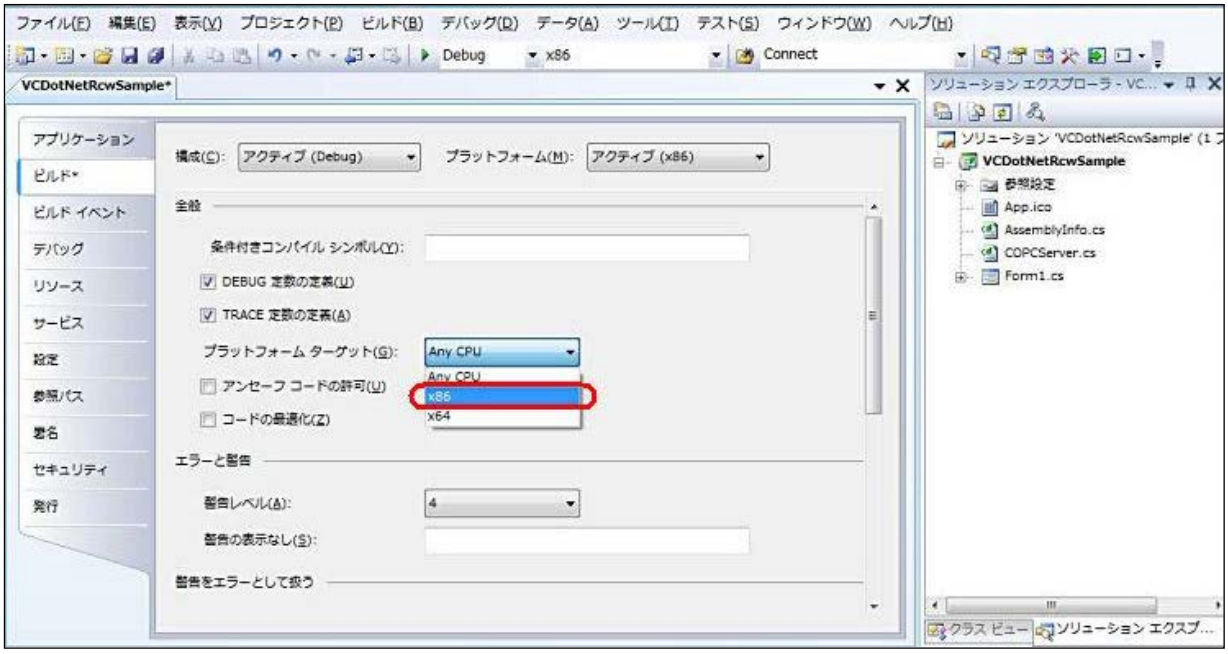
OPC サーバーのプロパティ・メソッドはオブジェクトブラウザで参照できます。



1.7.3　64 ビット Windows 上でビルドする際の注意事項

RCW の DLL は 32 ビット用にビルドされています。64 ビット Windows 上の VisualStudio で OPC クライアントのサンプルをビルドする際は、ターゲット CPU を x86 に設定下さい。ターゲット CPU を AnyCPU もしくは x64 に設定してビルドした場合、予期しない動作となります。

「プロジェクト設定 > コンパイル > 詳細コンパイルオプション」を開き、ターゲット CPU を「x86」に設定して下さい。



1.7.4 プログラム例

Visual C#で OPC カスタムインターフェースを使用したプログラムの作成概要を説明します。本項ではサンプルプログラムで使用する COPCServer クラスについて説明します。プログラミングの詳細および OPC カスタムインターフェースの詳細に関しては、サンプルプログラムおよび OPCDA3.0 の仕様書を参照して下さい。

❖ 概要

OPC サーバーとの通信処理を行うユーザークラスです。また、OPC サーバーからのイベントを取得するため、IOPCDataCallback インターフェースを継承しています。

❖ 定義部分

OPC サーバーで使用するオブジェクトを定義します。

```
private IOPCServer                  m_OPCServer;     // サーバーオブジェクト
private IOPCGroupStateMgt2          m_OPCGroup;      // グループオブジェクト
private IOPCItemMgt                 m_OPCItem;       // アイテムオブジェクト
private IConnectionPointContainer   m_OPCConnPointCntnr;
                                                     // コネクションポイントコンテナオブジェクト
private IConnectionPoint            m_OPCConnPoint;// コネクションポイントオブジェクト
```

❖ サーバーとの接続

まず、OPC サーバーリストを作成します。
次に、リストから OPC サーバーの Prog.ID をもとに OPC サーバーの CLSID を抽出し、CLSID をもとに OPC サーバーオブジェクトを作成してサーバーとの接続を行います。
この例では MELSEC OPC Server に接続しています。

```
IOPCServerList svrList = (IOPCServerList)CreateInstance(CLSID_SERVERLIST, null);
                                                     // OPC サーバーリストのインスタンス生成
Guid clsidList;                                      // OPC サーバーリストから抽出する CLSID
svrList.CLSIDFromProgID("Takebishi.Dxp.5", out clsidList);
                                                     // ProgID をもとに CLSID を抽出
m_OPCServer = (IOPCServer)CreateInstance(clsidList, null);
                                                     // OPC サーバーのインスタンス生成
```

❖ グループの作成

サーバーオブジェクトの AddGroup メソッドを実行してグループを追加します。また、サーバーからのコールバックを有効にするため、COPCCallBack クラスにコネクションポイントを割り当てます。

```
guidGroupStateMgt = Marshal.GenerateGuidForType(typeof(IOPCGroupStateMgt2));
m_OPCServer.AddGroup(sGrpName,                       // グループ名
                     1,                              // 1：アクティブ状態、：非アクティブ状態
                     1000,                           // 更新周期
                     1234,                           // クライアント側で指定するグループ ID
                     ptrTimeBias,                    // 時刻
                     ptrDeadBand,                    // デッドバンド
                     0,                              // 言語 ID
                     out m_iServerGroup,             // サーバー側で指定するグループ ID
                     out iRevisedUpdateRate,         // サーバーから返される更新周期
                     ref guidGroupStateMgt,          // IOPCGroupStateMgt の GUID
                     out group);                     // グループオブジェクト
m_OPCGroup = (IOPCGroupStateMgt2)group;
m_OPCGroup.SetKeepAlive(iKeepAliveTime, out iKeepAliveTime);

m_OPCConnPointCntnr = (IConnectionPointContainer)m_OPCGroup;
guidDataCallback = Marshal.GenerateGuidForType(typeof(IOPCDataCallback));
m_OPCConnPointCntnr.FindConnectionPoint(ref guidDataCallback, out m_OPCConnPoint);
                                                     // コネクションポイントの割り当て
```

❖ アイテムの追加

グループを作成した後にアイテムを追加します。この例では、アイテム名「Device1.D0」を 1 点 OPC サーバーに追加しています。
OPCITEMDEF 構造体には登録に必要な情報を格納します。szItemID は、OPC サーバーで設定したデバイス・グループ・タグをピリオド(.)で区切って指定して下さい。bActive はアイテム毎にアクティブ状態を設定して下さい。hClient にはクライアント側で識別できるユニークな値を設定して下さい。
ppResult および ppErrors は OPC サーバー側でセットされます。ppResult は OPCITEMRESULT 構造体のポインタで、hServer にはサーバー側でアイテムを識別する ID が格納されています。最後に ppResult および ppErrors のメモリを解放して下さい。
また、複数のアイテムを一括登録する事もできます。

```
itemDef[1].szItemID = "Device1.D0";                    // アイテム名
itemDef[1].bActive = 1;                                // 1：アクティブ状態、0：非アクティブ状態
itemDef[1].hClient = 1;                                // クライアント側で指定するアイテム ID
m_OPCItem = (IOPCItemMgt)m_OPCGroup;
m_OPCItem.AddItems(1, itemDef, out ppResult, out ppErrors);
                                                       // アイテム追加

itemResult = (OPCITEMRESULT)Marshal.PtrToStructure(ppResult, typeof(OPCITEMRESULT));
ServerHd = itemResult.hServer;                         // サーバー側で指定するアイテム ID を取得
Marshal.FreeCoTaskMem(ppResult);
Marshal.FreeCoTaskMem(ppErrors);
```

❖ コールバックの有効化

コネクションポイントオブジェクトの Advise メソッドでコールバックを有効にします。自動読み出しや非同期読み出し、非同期書き込みを行う場合はコールバックを有効にして下さい。

```
m_OPCConnPoint.Advise(this, out m_iCallBackConnection);
```

❖ 読み出し

この例は、グループオブジェクトを OPCSyncIO2 インターフェースにキャストして 1 アイテムをデバイス指定で同期読み出ししています。OPC サーバーからデータ値の他に時刻、品質フラグが取得できます。
最後に ppItemVal および ppErrors のメモリを解放して下さい。
この他に OPCAsyncIO3 インターフェースにキャストして非同期読み出しする方法があります。

```
OPCSyncIO2 = (IOPCSyncIO2)m_OPCGroup;              // IOPCSyncIO2 インターフェースにキャスト
OPCSyncIO2.Read(OPCDATASOURCE.OPC_DS_DEVICE, 1, ServerHd, out ppItemVal, out ppErrors);
                                                   // 同期読み出し

ItemState = (OPCITEMSTATE)Marshal.PtrToStructure(ppItemVal, typeof(OPCITEMSTATE));
vValue = ItemState.vDataValue;                     // 値を取得
fTime = ItemState.ftTimeStamp;                     // 時刻を取得
wQuality = ItemState.wQuality;                     // 品質フラグを取得
Marshal.FreeCoTaskMem(ppItemVal);
Marshal.FreeCoTaskMem(ppErrors);
```

❖ 書き込み

この例は、グループオブジェクトを OPCSyncIO2 インターフェースにキャストして 1 アイテムを同期書き込みしています。最後に ppErrors のメモリを解放して下さい。
この他に OPCAsyncIO3 インターフェースにキャストして非同期書き込みする方法があります。

```
OPCSyncIO2 = (IOPCSyncIO2)m_OPCGroup;              // IOPCSyncIO2 インターフェースにキャスト
vValue = 12345;
OPCSyncIO2.Write(1, ServerHd, vValue, out ppErrors);  // 同期書き込み
Marshal.FreeCoTaskMem(ppErrors);
```

1.7 Visual C# サンプル（OPC カスタムインターフェース）

❖ データ変化通知

OPC サーバーからデバイス内で変化の発生したアイテムが通知されます。どのアイテムに変化が発生したかを識別するには phClientItems を参照して下さい。

```
public void OnDataChange(
        int                         dwTransid,
        int                         hGroup,
        int                         hrMasterquality,
        int                         hrMastererror,
        int                         dwCount,        // 変化の発生したアイテム数
        int[]                       phClientItems,  // クライアント側で設定したアイテム ID
        object[]                    pvValues,       // データ値
        short[]                     pwQualities,    // 品質フラグ
        OpcRcw.Da.FILETIME[]        pftTimeStamps,  // 時刻
        int[]                       pErrors)        // エラー
{
        DataChange(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, pwQualities, pErrors);
                                                    // イベント発生
}
```

❖ 非同期読み出し完了通知

OPC サーバーから非同期読み出しが完了すると通知されます。読み出しを行ったアイテムを識別するには phClientItems を参照して下さい。

```
public void OnReadComplete(
        int                         dwTransid,
        int                         hGroup,
        int                         hrMasterquality,
        int                         hrMastererror,
        int                         dwCount,        // 読み出しを行ったアイテム数
        int[]                       phClientItems,  // クライアント側で設定したアイテム ID
        object[]                    pvValues,       // データ値
        short[]                     pwQualities,    // 品質フラグ
        OpcRcw.Da.FILETIME[]        pftTimeStamps,  // 時刻
        int[]                       pErrors)        // エラー
{
        ReadComplete(dwTransid, dwCount, phClientItems, pvValues, pftTimeStamps, pwQualities,pErrors);
                                                    // イベント発生
}
```

❖ 非同期書き込み完了通知

OPC サーバーから非同期書き込みが完了すると通知されます。書き込みを行ったアイテムを識別するには phClientItems を参照して下さい。

```
public void OnWriteComplete(
        int                         dwTransid,
        int                         hGroup,
        int                         hrMastererror,
        int                         dwCount,        // 書き込みを行ったアイテム数
        int[]                       phClientItems,  // クライアント側で設定したアイテム ID
        int[]                       pErrors)        // エラー
{
        WriteComplete(dwTransid, dwCount, phClientItems, pErrors);          // イベント発生
}
```

❖ 非同期処理のキャンセル完了通知

OPC サーバーから非同期処理のキャンセルが完了すると通知されます。

```
public void OnCancelComplete(
        int dwTransid,
        int hGroup)
{
        CancelComplete(dwTransid);                  // イベント発生
}
```

❖ サーバーとの接続終了

OPC サーバーとの接続を終了するには、OPCServer の RemoveGroup メソッドを実行して下さい。各オブジェクトは使用後、必ず削除して下さい。

```
if (m_OPCGroup != null)
{
        ret = Marshal.ReleaseComObject(m_OPCGroup);  // OPCGroup オブジェクトの参照を解放
        m_OPCGroup = null;
}
if (m_OPCConnPoint != null)
{
        ret = Marshal.ReleaseComObject(m_OPCConnPoint);
                                                    // OPCConnectionPoint オブジェクトの参照を解放

        m_OPCConnPoint = null;
}
if (m_iServerGroup != 0)
{
        m_OPCServer.RemoveGroup(m_iServerGroup, 0);// OPCGroup オブジェクトの削除
        m_iServerGroup = 0;
}
ret = Marshal.ReleaseComObject(m_OPCServer);          // OPCServer オブジェクトの参照を解放
```

1.8 DXP 開発支援ツール

1.8.1　クライアントコンポーネント（.NET 専用）

1.8.1.1　概要

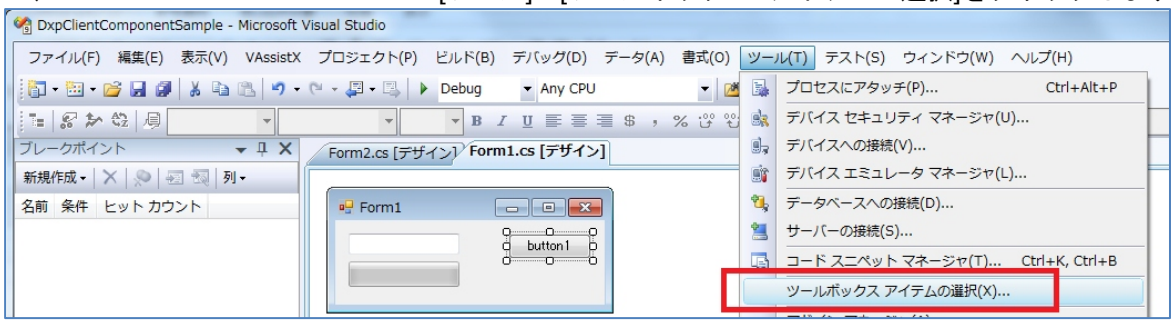
当社が提供する.NET 専用開発支援ツールを利用して、OPC クライアントを簡単に開発することができます。DxpClientComponent は、Visual Studio を使用したクライアント開発において、プログラムレスでフォーム上のコントロール（テキストボックスなど）にデバイスエクスプローラのアイテムを割り付け、自動的な値表示や、画面からの数値入力を可能とする.NET コンポーネントです。

重要
DxpClientComponent はデバイスエクスプローラのエンタープライズライセンスをご購入の方のみ使用できます。デモ版やスタンダードライセンスをご購入の方は使用されるとライセンス違反となります。

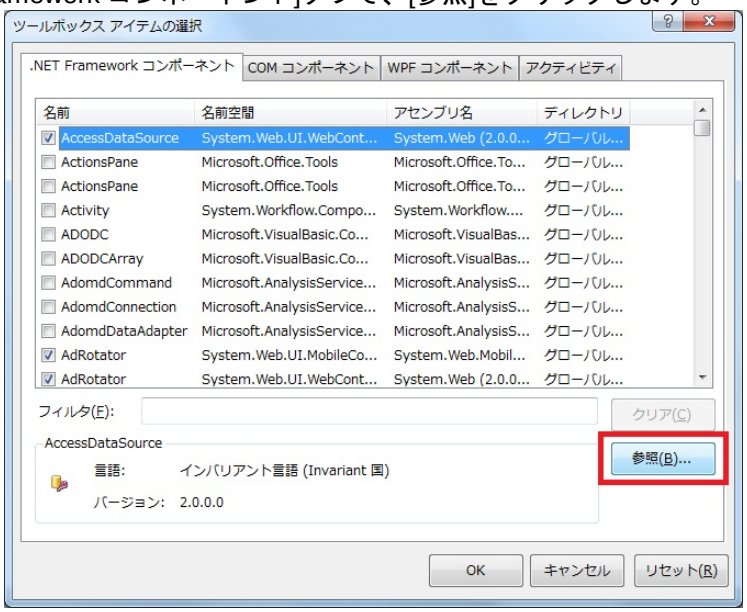
1.8.1.2　開発環境の設定方法

❖ Visual Studio のツールボックスにコンポーネントを表示する手順

Visual Studio 2008 のツールボックスに開発支援用のコンポーネントを登録する手順を説明します。最初に、Visual Studio のメニューバーの[ツール]→[ツールボックス アイテムの選択]をクリックします。

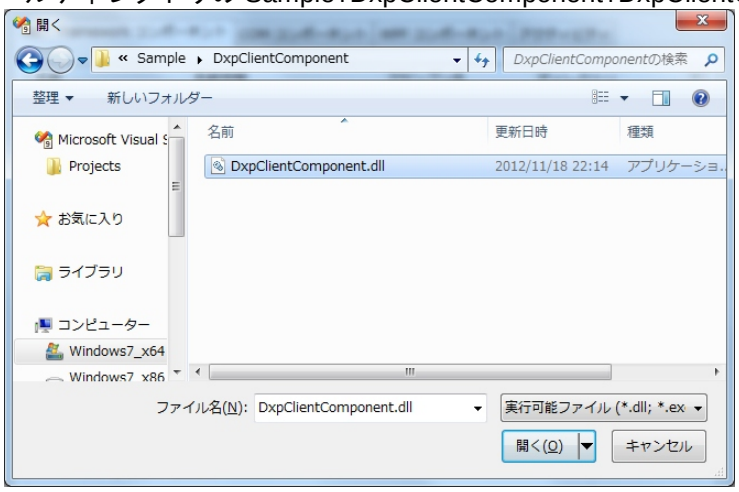


[.NET Framework コンポーネント]タブで、[参照]をクリックします。

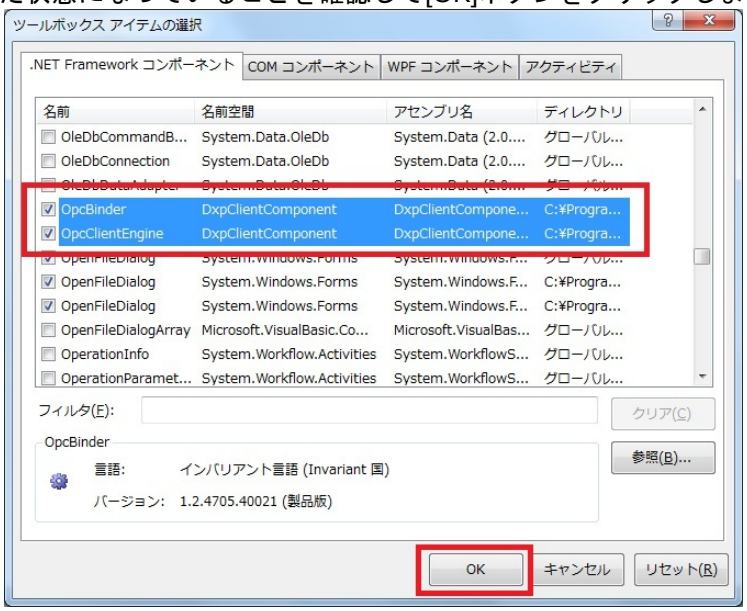


1.7 Visual C# サンプル（OPC カスタムインターフェース）

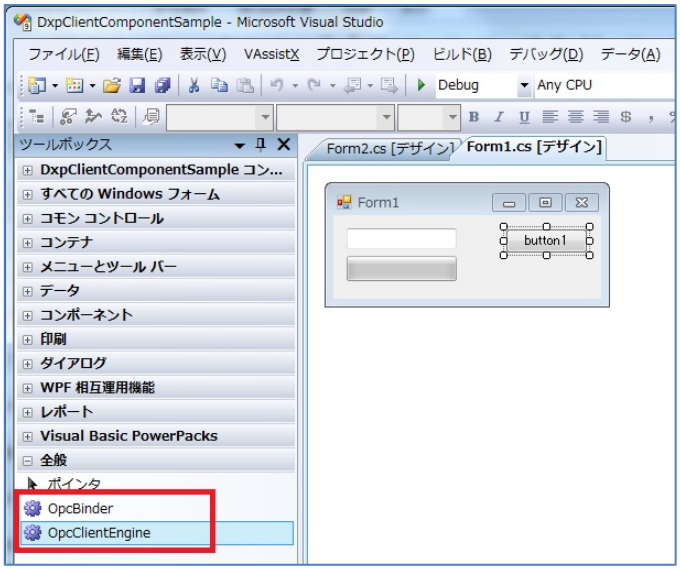
インストールディレクトリの Sample¥DxpClientComponent¥DxpClientComponent.dll を選択します。



一覧に、選択した DLL の名称が登録されます。OpcBinder と OpcClientEngine コンポーネントにチェックを入った状態になっていることを確認して[OK]ボタンをクリックします。

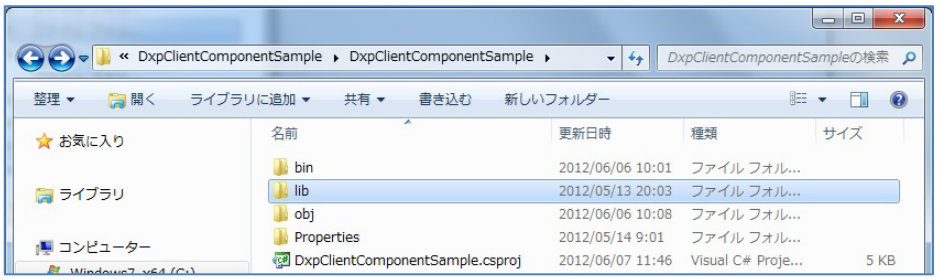


ツールボックスに、OpcBinder と OpcClientEngine が表示されます。

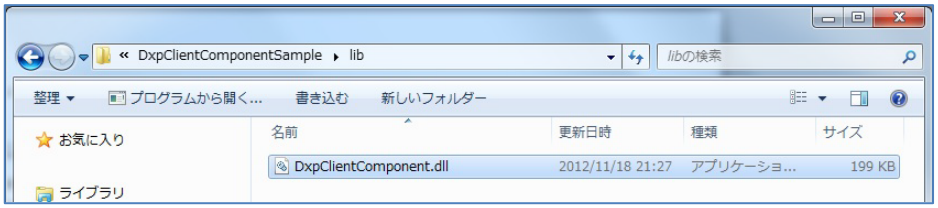


❖ プロジェクトの参照に登録する手順

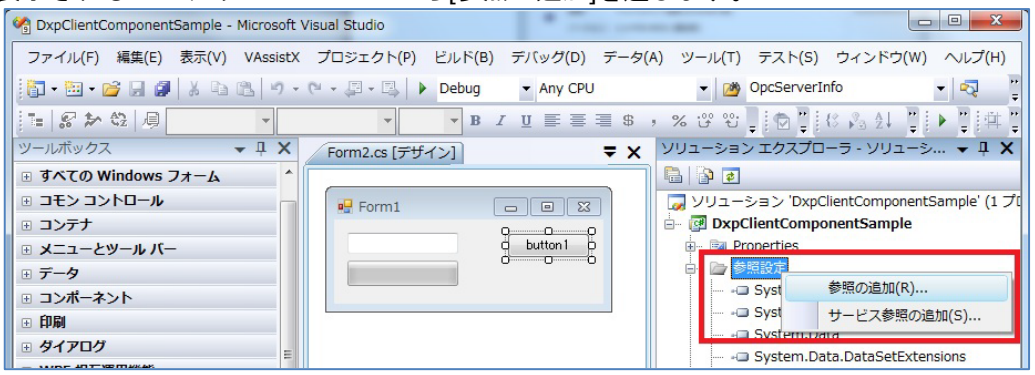
Visual C#もしくは Visual Basic.NET で、DxpClientComponent.dll をプロジェクトの参照に登録します。Visual C#のプロジェクトを例に、設定例を説明します。Visual Studio で新規にソリューション、プロジェクトを作成し、作成されたプロジェクトフォルダの下に「lib」フォルダーを作成します。



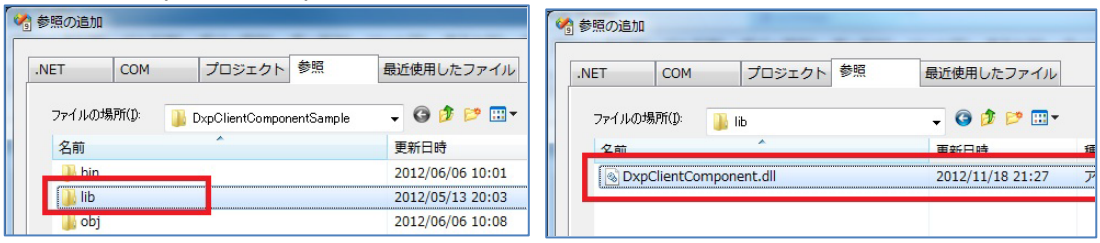
ここにインストールディレクトリの Sample¥DxpClientComponent¥DxpClientComponent.dll のファイルを、先ほど作成したプロジェクトの「lib」フォルダーにコピーします。



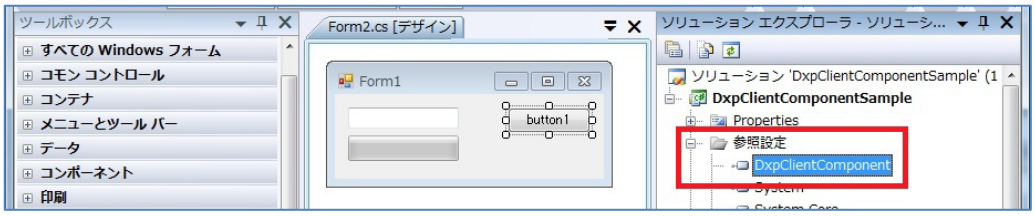
Visual Studio の[プロジェクト]→[参照の追加]を選ぶか、プロジェクトエクスプローラの参照設定の右クリックで表示されるコンテキストメニューから[参照の追加]を選びます。



[参照]タブから、lib¥DxpClientComponent.dll を選択します。



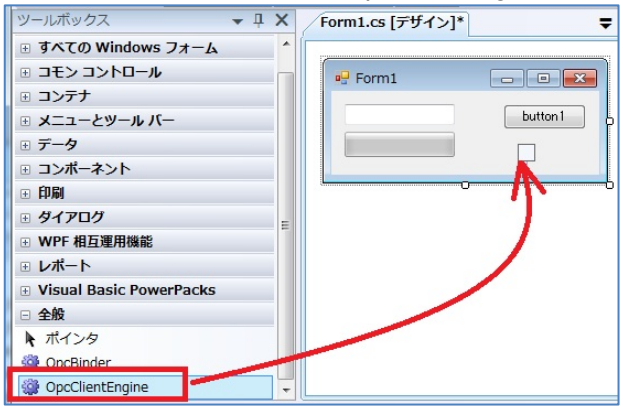
プロジェクトの参照設定に登録されたことを確認して完了です。



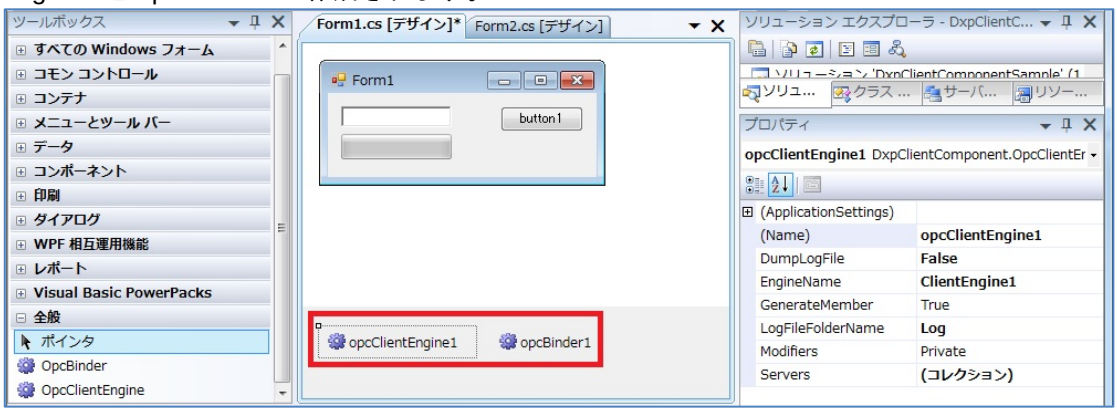
1.8.1.3 開発手順

❖ コンポーネントの利用手順

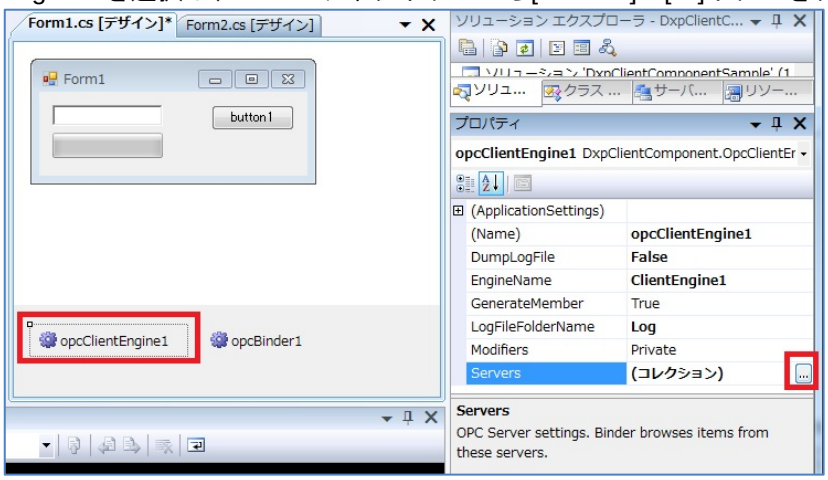
Visual Studio のツールボックスから、OpcClientEngine をフォームにドラッグ＆ドロップします。



同様の手順で、ツールボックスから OpcBinder をフォームにドラッグ＆ドロップします。
opcClientEngine1 と opcBinder1 が作成されます。

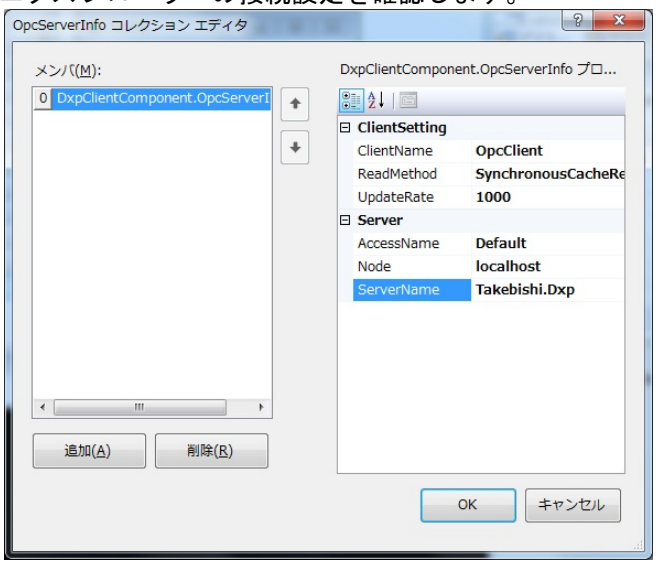


opcClientEngine1 を選択し、プロパティグリッドから[Servers]の[…]ボタンをクリックします。

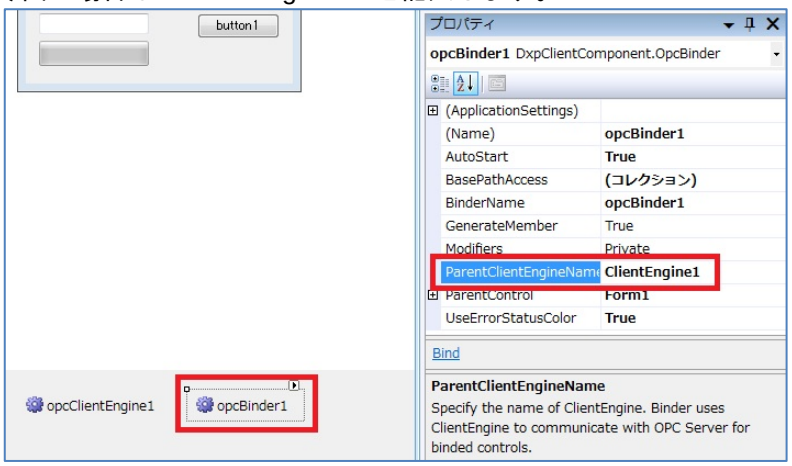


1.7 Visual C# サンプル（OPC カスタムインターフェース）

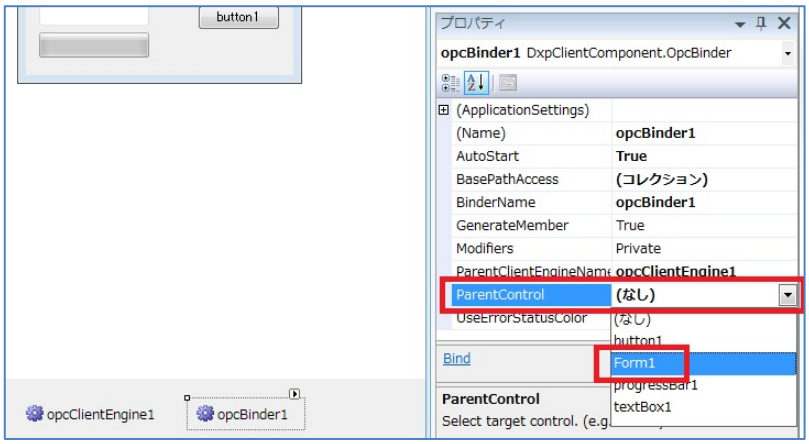
デバイスエクスプローラへの接続設定を確認します。



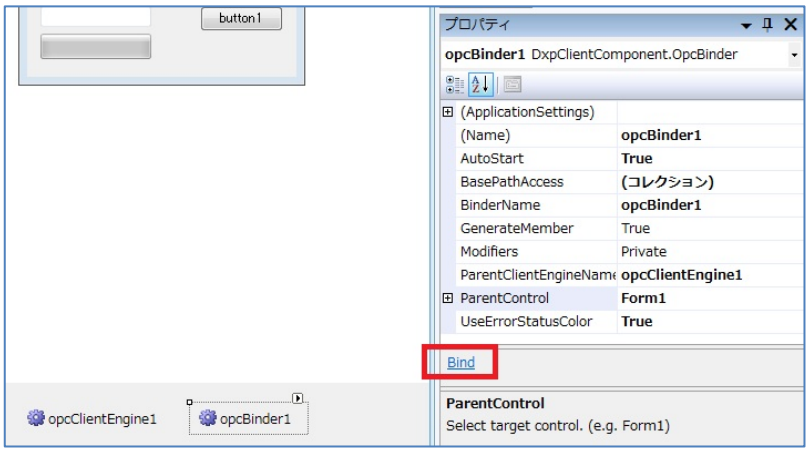
opcBinder1 の[ParentClientEngineName]に、opcClientEngine1 の[EngineName]に設定された文字を記入します。今回の場合は“ClientEngine1”と記入します。



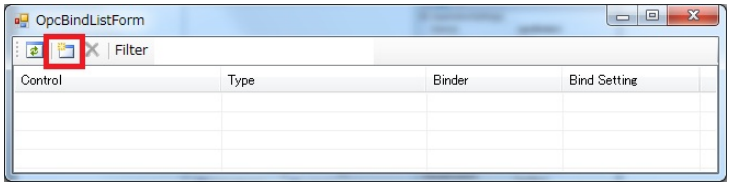
[ParentControl]で Form1 を選択します。これにより、Form1 上のコントロールに、OPC サーバーのアイテムを割り付けることができるようになります。



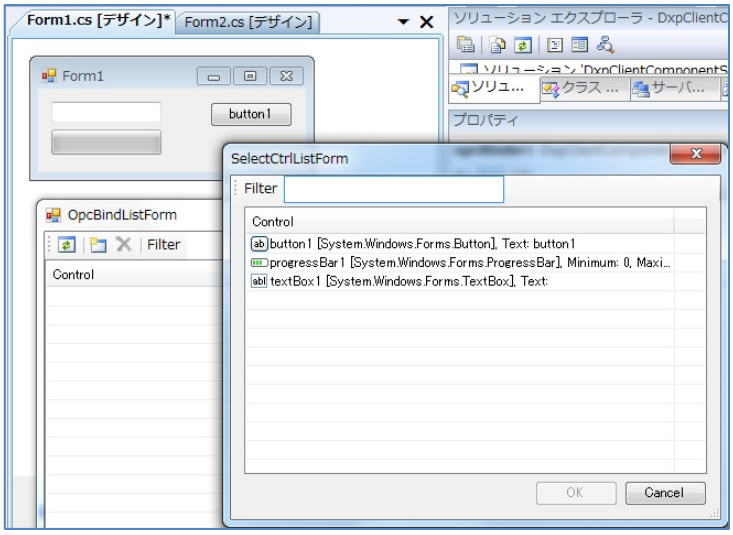
[Bind]ボタンをクリックします。



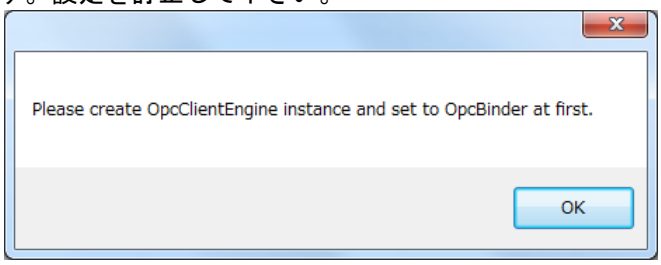
出現した設定画面から、新規作成のアイコンをクリックします。



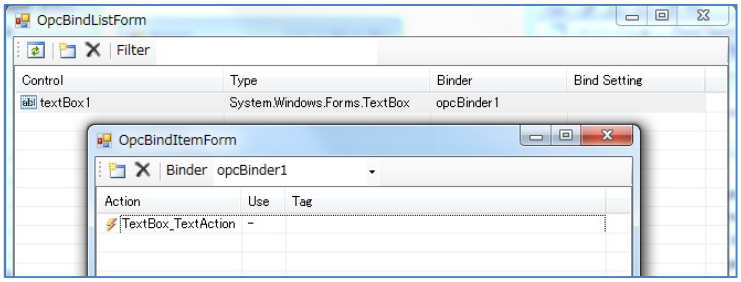
Form1 上のコントロールが一覧表示されます。値表示したいコントロールを選択して[OK]をクリックします。 本説明では、textBox1 を選択します。



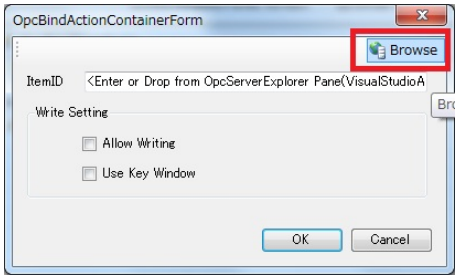
このとき、下記のメッセージが表示される場合は、opcBinder1 の[ParentClientEngineName]の設定が誤っています。設定を訂正して下さい。



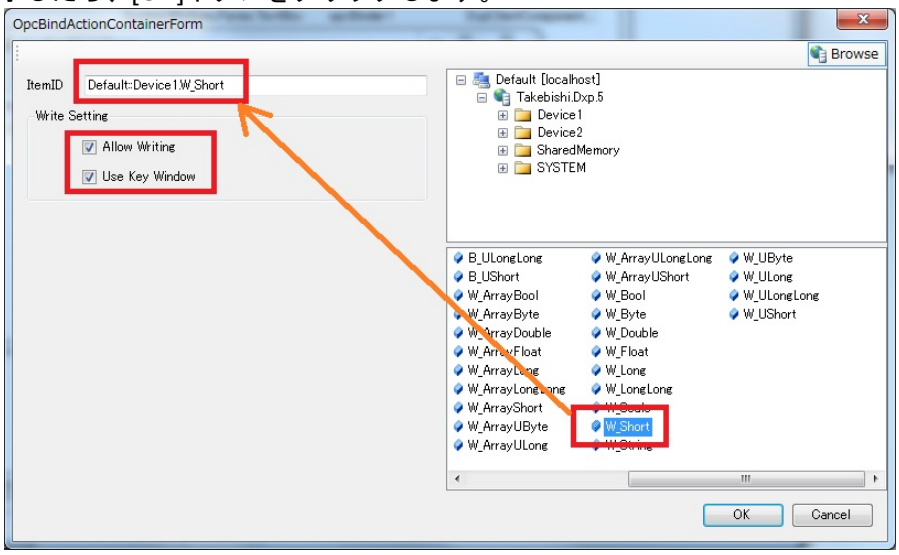
textBox1 に設定可能なアクションとして、TextBox_TextAction が表示されます。Use の列が“-”となっている場合、アクションを利用しないことを示しています。TextBox_TextAction をダブルクリックします。



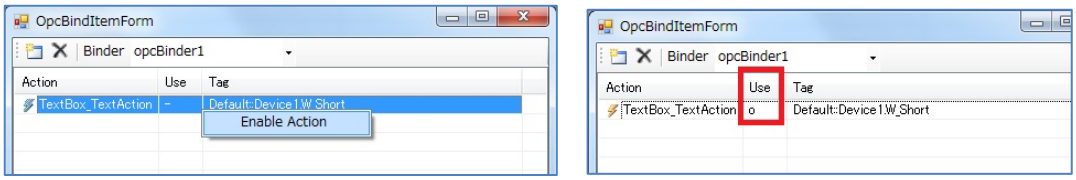
アクションの設定画面が出現します。画面右上の[Browse]ボタンをクリックすると、ブラウズビューが出現します。



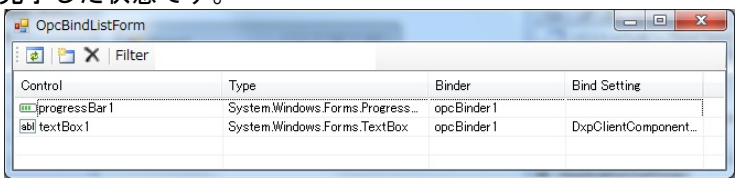
ブラウズビューで、「Takebishi.Dxp」ノードの[+]をクリックするとデバイスエクスプローラへ接続され、デバイスエクスプローラのアドレス空間がブラウジングできます。ブラウズビューの下部に表示されたアイテムをダブルクリックして[ItemID]に設定します。書き込みを許可する場合には[Allow Writing]にチェックを入れ、テンキーウィンドウを利用する場合は[Use Key Window]にもチェックを入れます。
設定が完了したら、[OK]ボタンをクリックします。



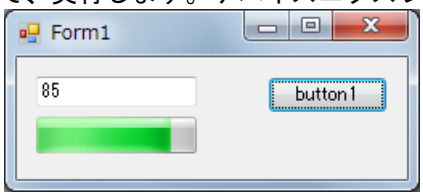
設定したアクションを有効にするために、アクションの上で右クリックし、[Enable Action]をクリックします。Use の列が“o”になることを確認します。



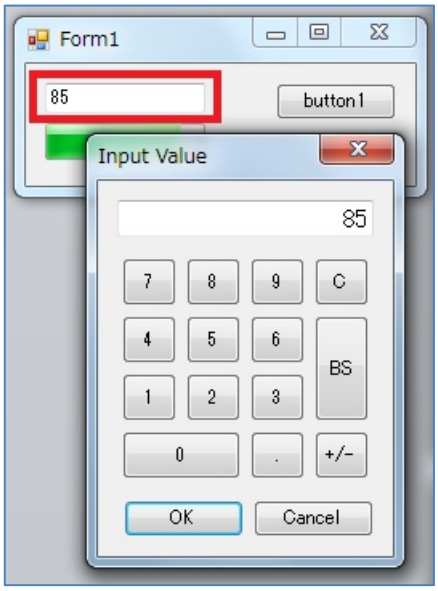
同様の手順で、Form1 上のプログレスバーにもアイテムを設定し、全ての設定画面を[×]ボタン、もしくはエスケープキーで閉じます。下記は、テキストボックス（textBox1）とプログレスバー（progressBar1）の設定が完了した状態です。



ビルドして、実行します。デバイスエクスプローラに接続され、自動的に値が表示されます。



テキストボックスをクリックするとキーウィンドウが表示され、値を書き込むことができます。



1.8.1.4 サンプル

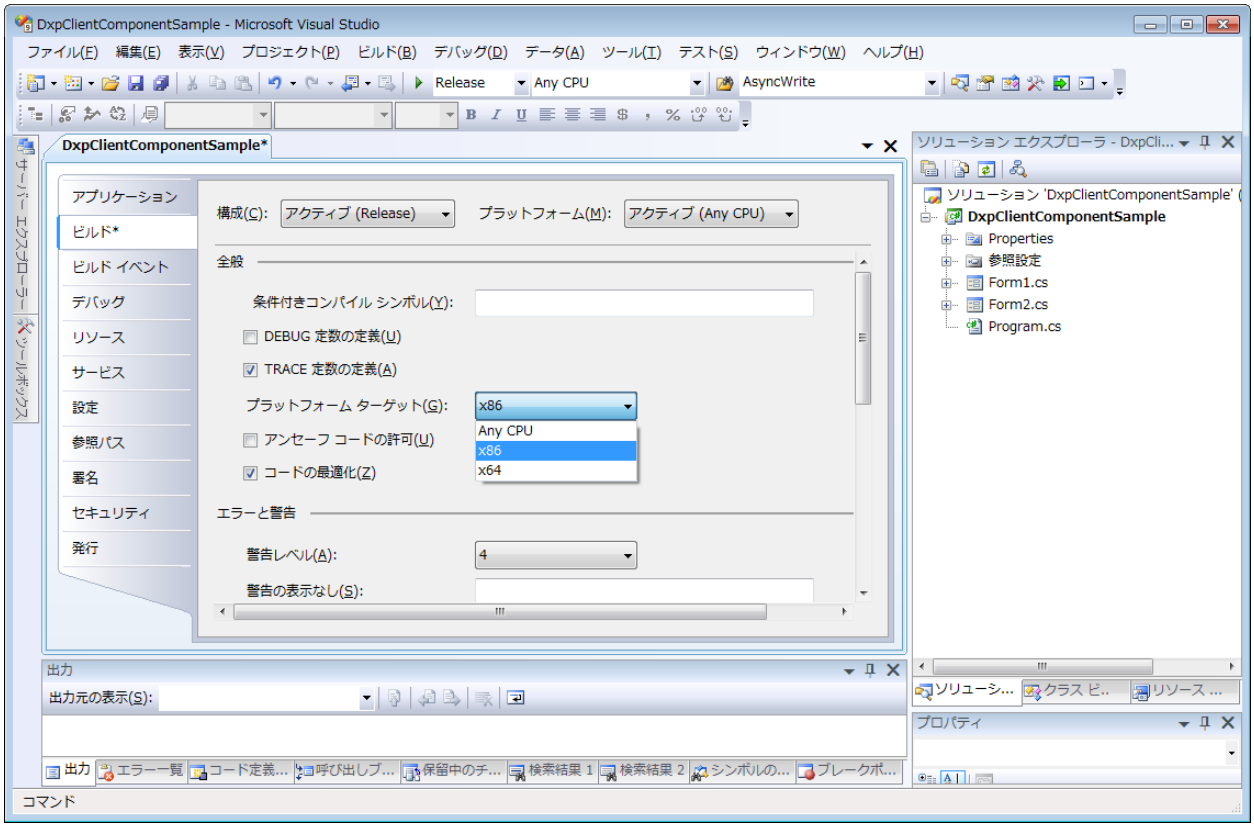
インストールディレクトリの Sample¥DxpClientComponent フォルダーの中に、VC#用のサンプルプログラム DxpClientComponentSample が同梱されています。

VC#サンプル	<Disk Directory>¥Samples¥DxpClientComponent¥DxpClientComponentSample
DLL	<Disk Directory>¥Samples¥DxpClientComponent¥DxpClientComponent.dll

1.8.1.5 64 ビット Windows 上でビルドする際の注意事項

DxpClientComponent の DLL は 32 ビット用にビルドされています。64 ビット Windows 上の VisualStudio で OPC クライアントのサンプルをビルドする際は、ターゲット CPU を x86 に設定下さい。ターゲット CPU を AnyCPU もしくは x64 に設定してビルドした場合、予期しない動作となります。

「プロジェクト設定 > ビルド > 全般」を開き、プラットフォームターゲットを「x86」に設定して下さい。



1.8.2 シンプル API（.NET 専用）

1.8.2.1 概要

当社が提供する.NET 専用開発支援ツールを利用して、OPC クライアントを簡単に開発することができます。DxpSimpleAPI は、Visual Studio を使用したクライアント開発において、単純なメソッド呼び出しで、デバイスエクスプローラからの値読み出しや書き込みを実現するヘルパ関数のライブラリです。

重要
DxpSimpleAPI はデバイスエクスプローラのエンタープライズライセンスをご購入の方のみ使用できます。デモ版やスタンダードライセンスをご購入の方は使用されるとライセンス違反となります。

1.8.2.2 開発環境の設定方法

Visual C#もしくは Visual Basic.NET の[プロジェクト]→[参照の追加]で[参照]ボタンをクリックして以下ファイルを選択して下さい。
＜インストールフォルダー＞¥Sample¥ DxpSimpleAPI¥ DxpSimpleAPI.dll

1.8.2.3 サンプル

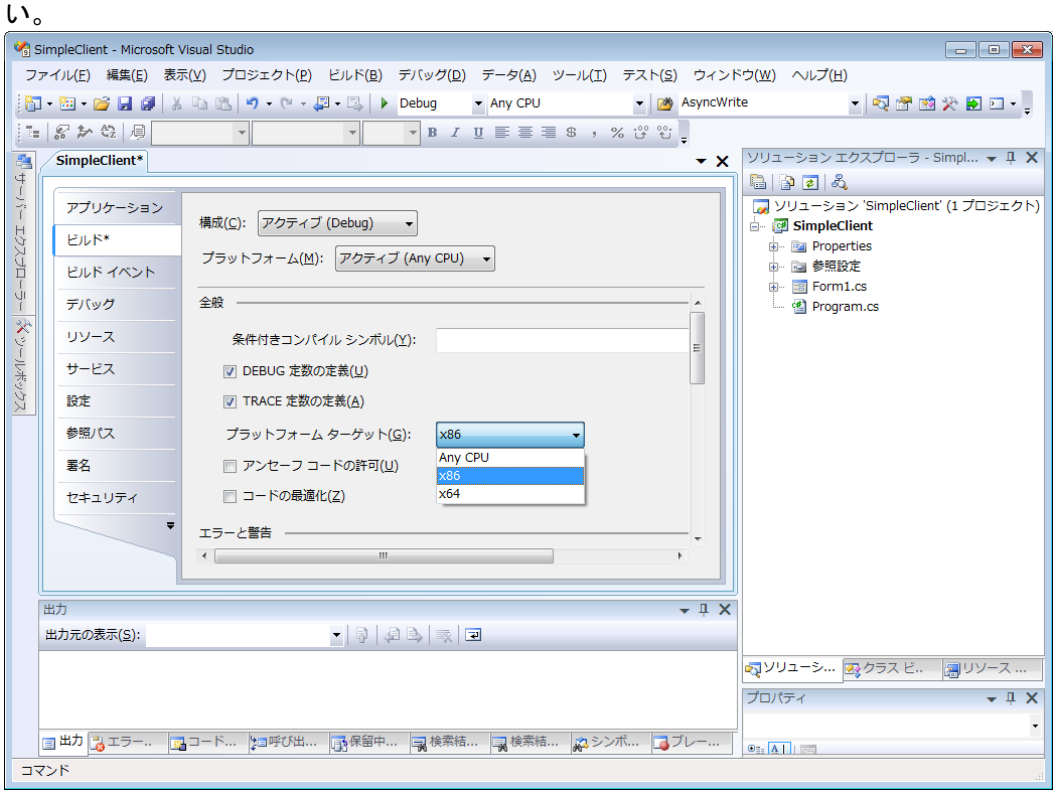
インストールメディアの Samples¥DxpSimpleAPI フォルダーの中に、VC#用のサンプルプログラム SimpleClient と、VB.NET 用のサンプルプログラム SimpleClientVB が同梱されています。

VC#サンプル	<Disk Directory>¥Samples¥DxpSimpleAPI¥SimpleClient
VB.NET サンプル	<Disk Directory>¥Samples¥DxpSimpleAPI¥SimpleClientVB
DLL	<Disk Directory>¥Samples¥DxpSimpleAPI¥DxpSimpleAPI.dll

1.8.2.4 64 ビット Windows 上でビルドする際の注意事項

DxpSimpleAPI の DLL は 32 ビット用にビルドされています。64 ビット Windows 上の VisualStudio で OPC クライアントのサンプルをビルドする際は、ターゲット CPU を x86 に設定下さい。ターゲット CPU を AnyCPU もしくは x64 に設定してビルドした場合、予期しない動作となります。

「プロジェクト設定 > ビルド > 全般」を開き、プラットフォームターゲットを「x86」に設定して下さい。



1.9 EXCEL 通信支援ツール（OPCFunction）

本通信支援ツールは EXCEL のアドインツールとして動作し、セルに数式を入力するだけで OPC サーバーにアクセスすることができます。

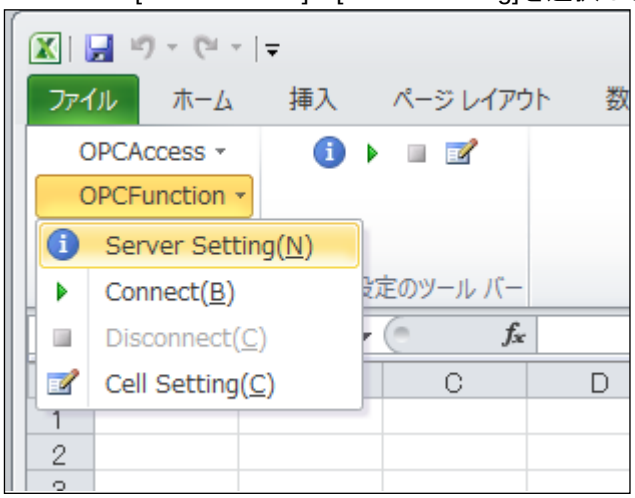
1.9.1 操作方法

❖ アドイン手順

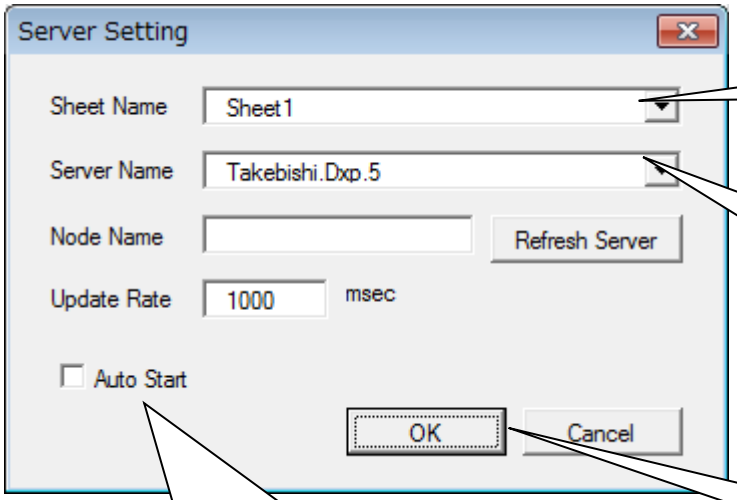
EXCEL を起動します。
メニューの[ツール]→[アドイン]を選択し、アドインダイアログで以下ファイルを選択して下さい。
＜OPC サーバーインストールフォルダー＞¥Sample¥EXCEL¥OPCFunction.xla
メニューに[OPCFunction]が追加されます。また、アイコンボタンも表示されます。

❖ 通信設定

メニューの[OPCFunction]→[Server Setting]を選択します。



OPC サーバー設定ダイアログが表示されますので、シート毎に設定を行って下さい。



通信設定を行うシート名を選択して下さい。

ノード名を指定して「Refersh Server」ボタンをクリックすると、OPC サーバー名が更新されます。空白時は自ノードで通信します。

xls ファイルを開くと同時に自動的に通信を開始します。

「OK」ボタンをクリックすると、設定が保存されます。

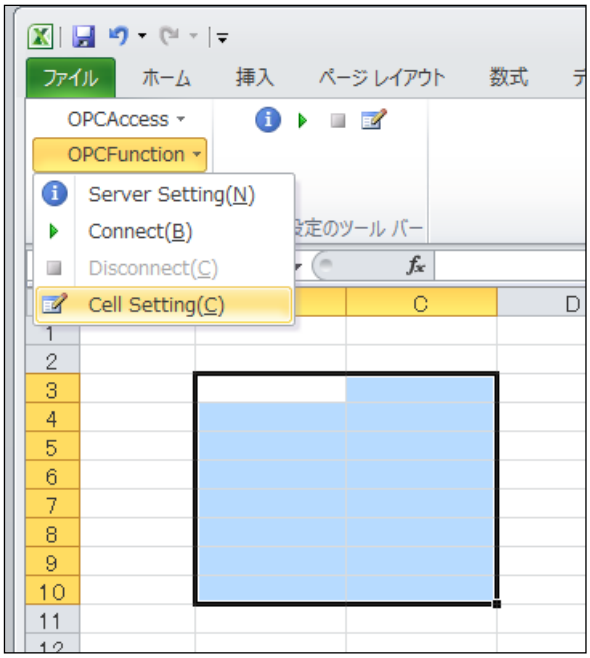
❖ 数式入力

セルに OPC アクセス関数（OPCValue）を埋め込むことにより数値モニタができます。

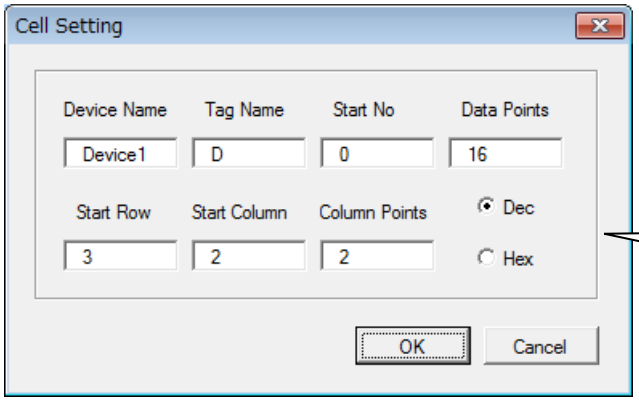
```
=OPCValue("OPC アイテム ID")
```

例）アイテム名"Device1.D0"の値をモニタする場合
=OPCValue("Device1.D0")
アイテム名の拡張指定もできます。ただし、配列型はサポートしていません。

また、セル設定機能によりあらかじめ選択したセル範囲に連続したデバイスの数式を埋め込むことができます。
セルを選択します。
メニューの[OPCFunction]→[Cell Setting]を選択します。またはアイコンボタンの　をクリックします。

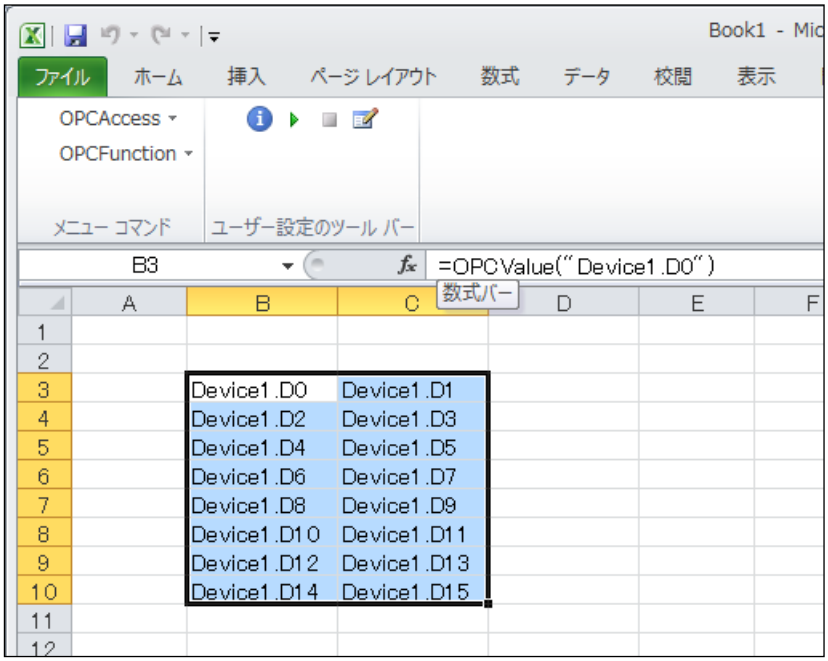


次のダイアログボックスが表示され、OPC サーバーのデバイス名、タグのデバイス先頭番号などを入力します。（デフォルトは”Device1.D0”）
セル設定では、アイテム名の拡張指定はできません。
一度に設定できる点数は 1000 点までです。



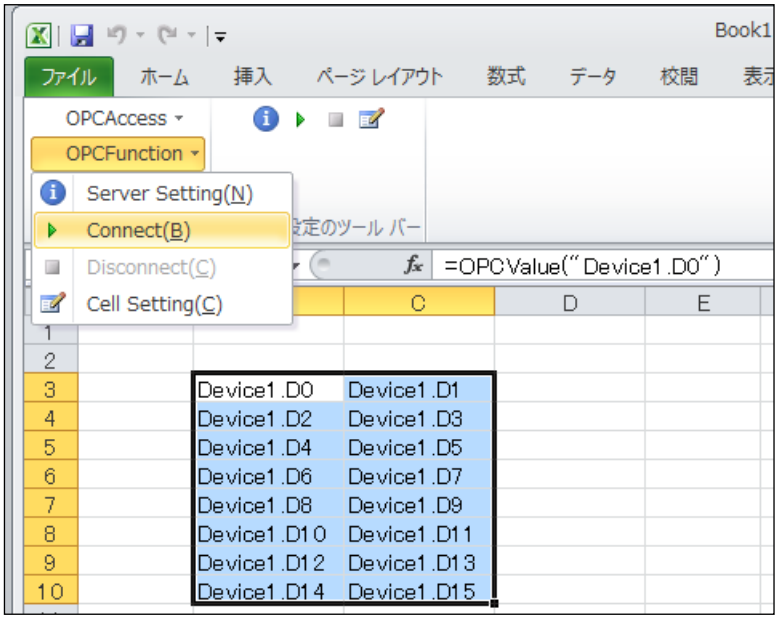
設定する Tag Name によりデバイス番号が 10 進数か 16 進数かを選択して下さい。

「OK」ボタンをクリックすると、選択した範囲に数式が書き込まれます。



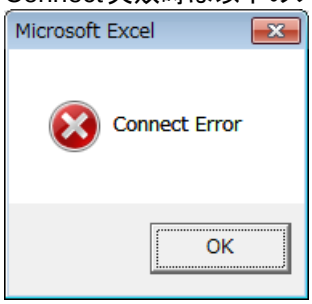
❖ 通信の開始・終了（デバイス値の読み出し）

通信を開始するには、メニューの[OPCFunction]→[Connect]を選択します。または、アイコンボタンの をクリックします。

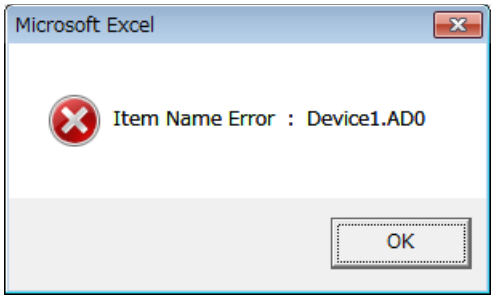


OPCValue の数式が設定されているセルにデバイス値が格納されます。

Connect 失敗時は以下のメッセージが表示されますので、通信設定の OPC サーバー名を確認して下さい。



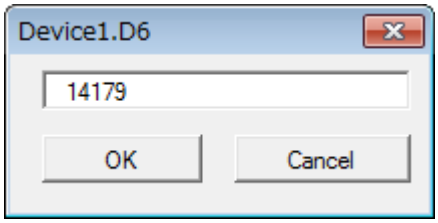
また、OPC アクセス関数（OPCValue）のアイテム ID が不正な場合（OPC サーバーの AddItem に失敗した場合）は以下のメッセージが表示されますので、アイテム ID を確認して下さい。



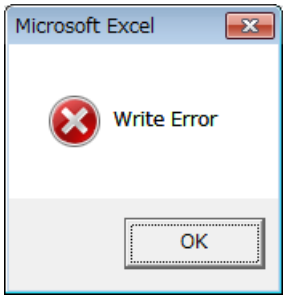
通信を終了するには、メニューの[OPCFunction]→[Disconnect]を選択します。または、アイコンボタンのをクリックします。

❖ デバイス値の書き込み

OPCValue の数式が設定されているセル上で通信中に右クリックを行うと、下記のウィンドウが表示されます。数値をボックスに入力し、「OK」ボタンにて書き込みを行います。



書き込み失敗時は以下のメッセージが表示されます。



1.10　EXCEL サンプル（OPCDataAccess）

本サンプルプログラムは EXCEL のアドインツールとして動作し、選択したセルで容易にデバイス値をモニタすることができます。

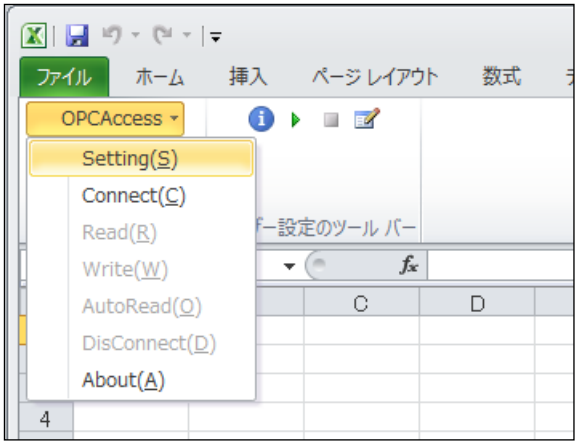
1.10.1 操作方法

❖ アドイン手順

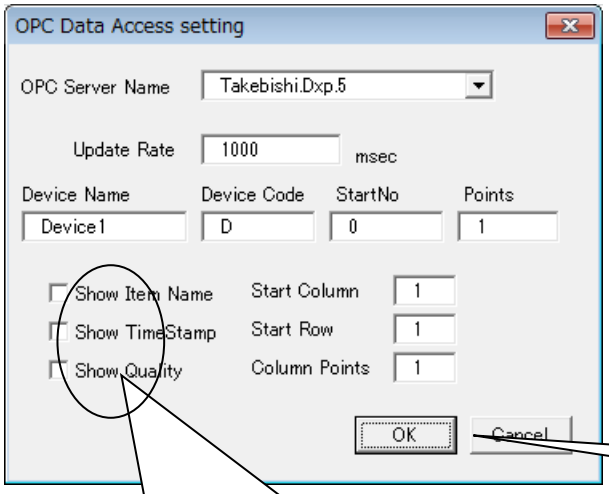
EXCEL を起動します。
メニューの[ツール]→[アドイン]を選択し、アドインダイアログで以下ファイルを選択して下さい。
＜OPC サーバーインストールフォルダー＞¥Sample¥EXCEL¥OPCDataAccess.xla
メニューに[OPCAccess]が追加されます。

❖ 通信設定

セルを選択します。
メニューバーの[OPCAcceess]→[Setting]を選択します。



次のダイアログボックスが表示され、OPC サーバー名・デバイス名・先頭デバイス番号・点数などを設定します。



「OK」ボタンをクリックすると設定が保存されます。

・Show Item Name：アイテム名を表示します
・Show Time Stamp：タイムスタンプを表示します
・Show Quality：品質フラグを表示します

❖ 通信の開始・終了

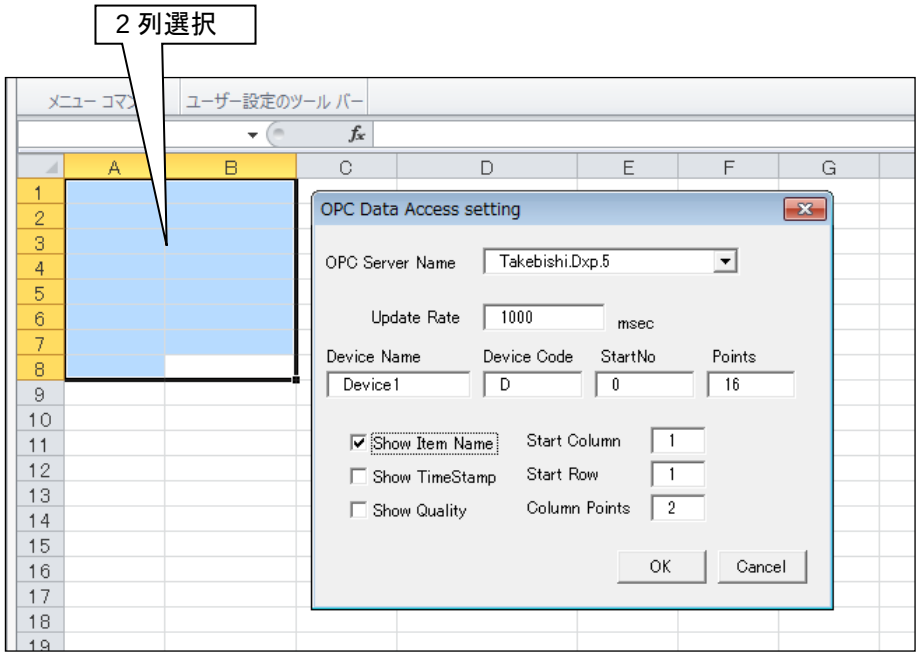
通信を開始するには、メニューの[OPCAcceess]→[Connect]を選択します。
接続が成功すると、[Setting]、[Connect]メニューが無効になり、[Read]、[Write]、[AutoRead]、[Disconnect]メニューが有効になります。
接続状態で[Disconnect]を選択すると通信を終了します。

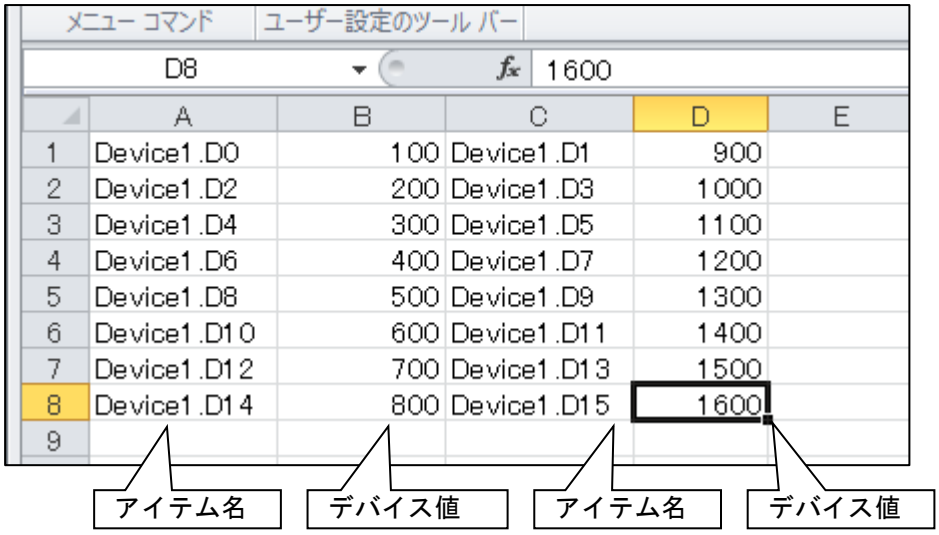
❖ アイテムの読み出し・書き込み

接続が確立している状態で[Read]を選択すると、一度だけデバイスを読み出します。
接続が確立している状態で[AutoRead]を選択すると、自動的にデバイスを読み出します。
接続が確立している状態で選択したセルに値を書き込み[Write]を選択すると、一括で値を書き込みます。

❖ サンプル画面

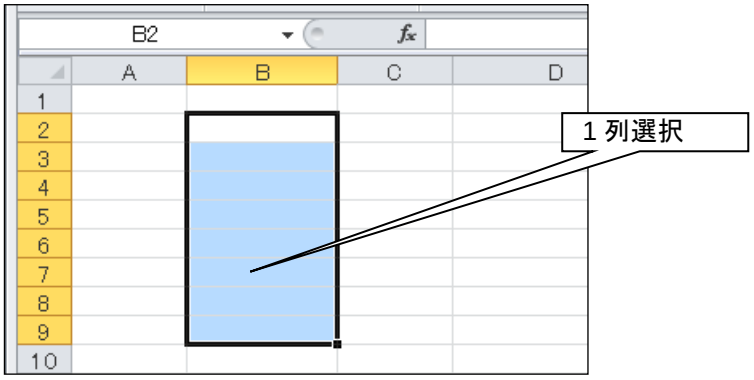
アイテム名の表示を有効にして Device1.D0 から 16 点を読み出す場合

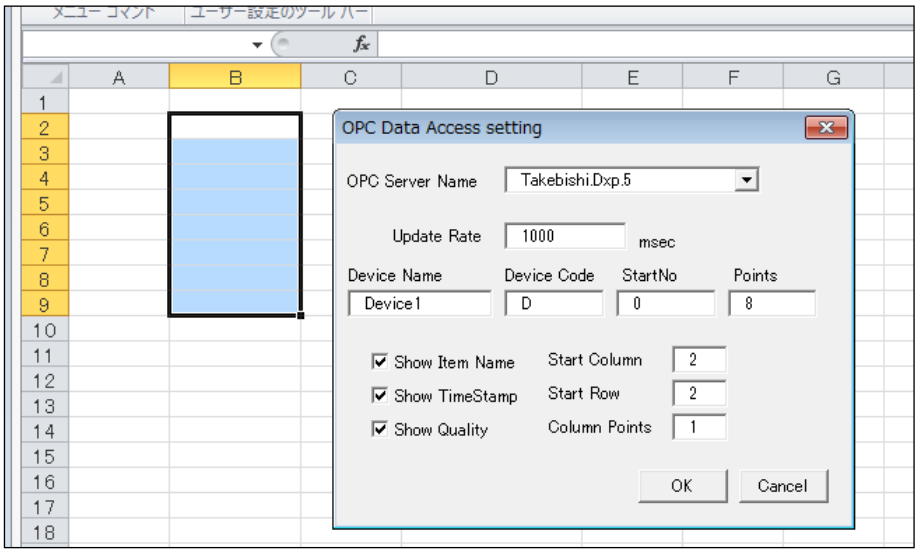


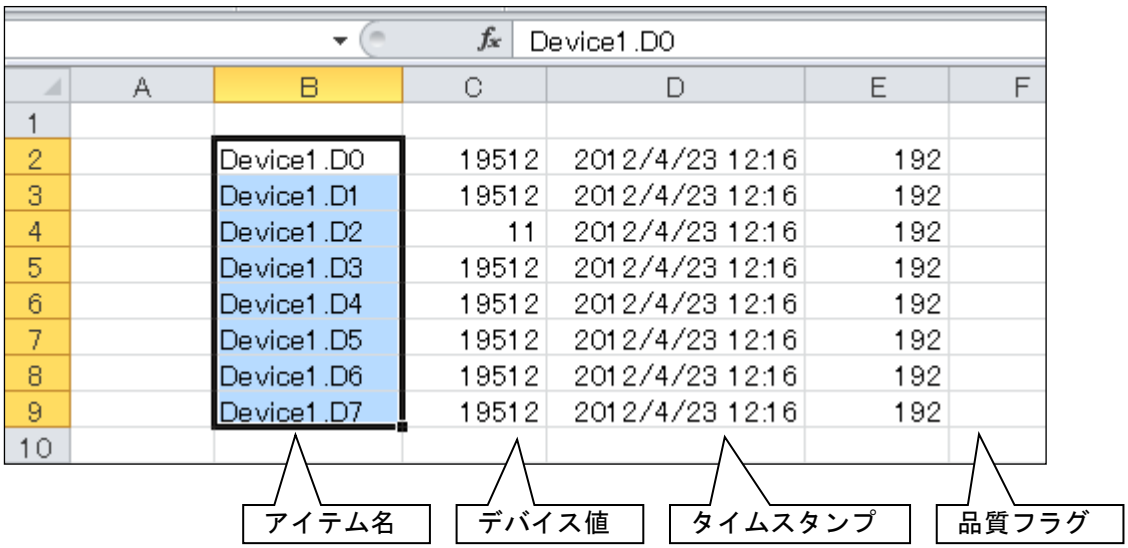


アイテム名・デバイス値の組合せの合計 4 列（2 列×2 組）で表示されます。

アイテム名・タイムスタンプ・品質フラグの表示を有効にして Device1.D0 から 9 点を読み出す場合



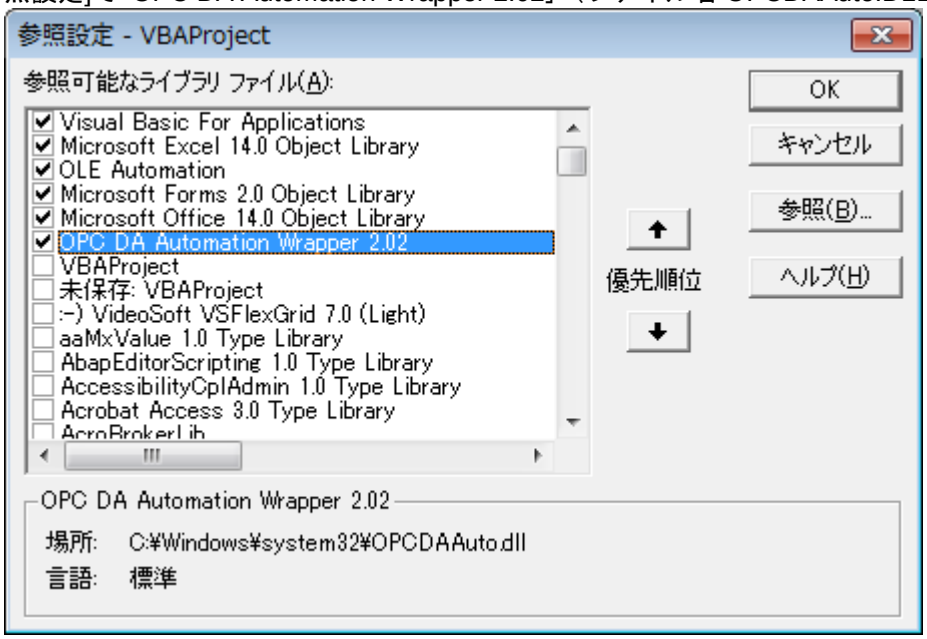




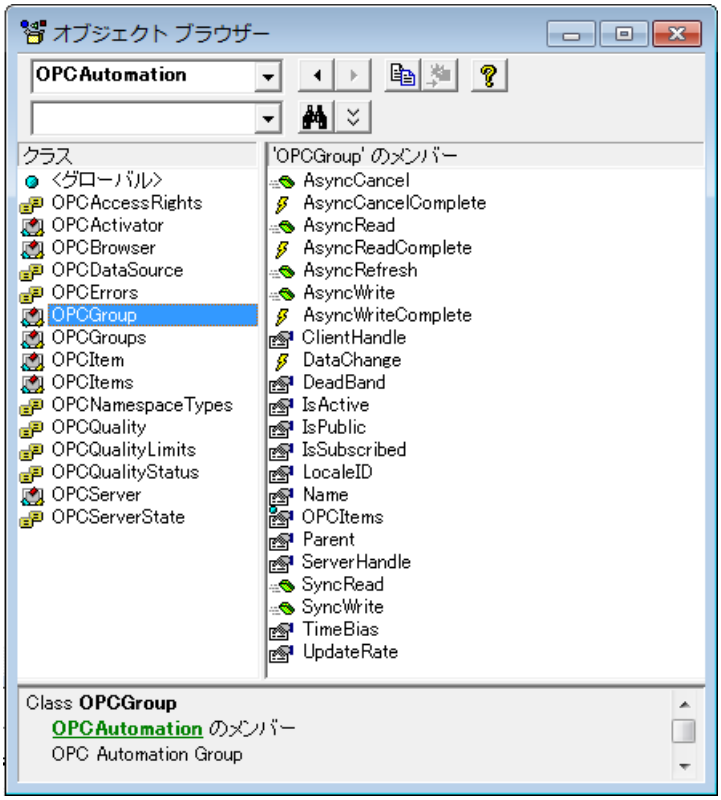
アイテム名・デバイス値・タイムスタンプ・品質フラグの組合せの合計 4 列（4 列×1 組）で表示されます。

1.10.2 開発環境の設定方法

EXCEL でプログラムを開発／修正する場合は、EXCEL メニューの[ツール]→[マクロ] →[Visual Basic Editor]を選択して Visual Basic Editor を起動して下さい。
OPC オートメーションインターフェースを使用する場合は、Visual Basic Editor メニューの[ツール]→[参照設定]で「OPC DA Automation Wrapper 2.02」（ファイル名 OPCDAAuto.DLL）を有効にして下さい。



OPC サーバーのプロパティ・メソッドはオブジェクトブラウザ(F2)で参照できます。



1.10.3 プログラム例

EXCEL で OPC オートメーションインターフェースを使用したプログラム例については、1.4.3 章を参照して下さい

1.10.4 アンインストール方法

■アドインファイルの削除
＜OPC サーバーインストールフォルダー＞¥Sample¥Excel¥OPCDataAccess.xla を削除してください。
（DeviceXPlorer OPC Server をアンインストールすると削除されます。）

■EXCEL メニューからの削除
（EXCEL2003 の場合）
1. EXCEL のメニューバーを右クリックし、[ユーザー設定]を開きます。
2. [コマンド]タブを選択し、右下の[コマンドの配置の変更]を開きます。
3. [ツールバー]のラジオボタンにチェックを入れ、その横のコンボボックスから[ワークシート メニューバー]を選択します。
4. 表示されるメニューから[OPC Access]を選択した状態で、[削除]を行います。

（EXCEL2007 の場合）
1. [アドイン]のリボンメニューにある[OPC Access]を右クリックします。
2. [ユーザー設定のコマンドの削除]を行います。

以上で、EXCEL のメニューバーから削除されます。

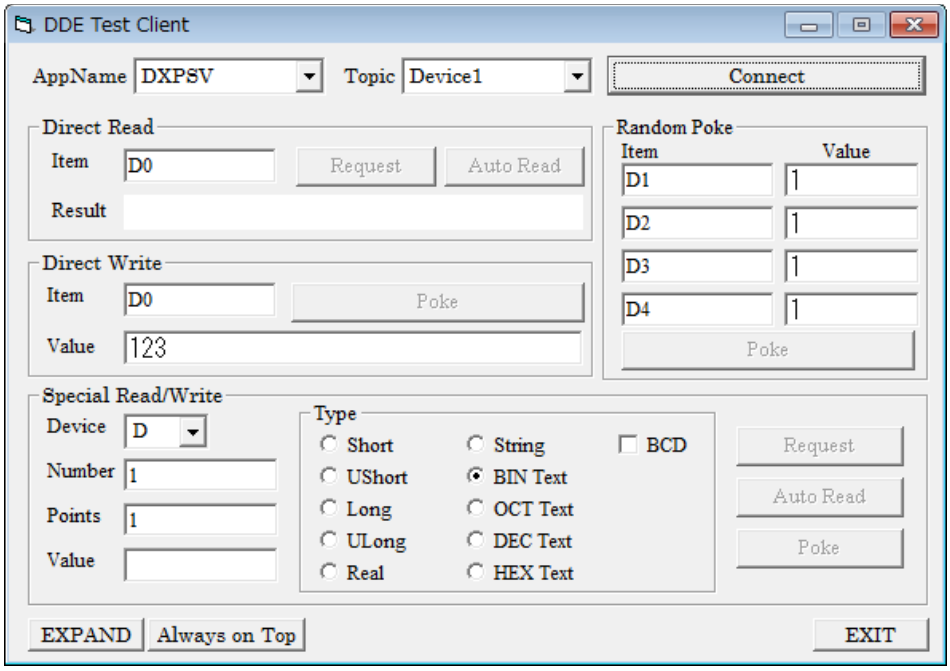
2 DDE クライアント

2.1 テストクライアント（DDE Client）

製品に添付されている DDE テストクライアントプログラムの使用方法について示します。

2.1.1 操作方法

スタートメニューから[プログラム]→[DeviceXPlorer OPC Server 5]→[DDE クライアント]を選択して DDE クライアントを起動します。



AppName（アプリケーション名）、TopicName（トピック名）を選択し、「Connect」ボタンを押します。アプリケーション名コンボボックスは、「DXPSV」を、トピック名は、デバイスエクスプローラで定義した文字列を選択します。

通常読出処理は「Direct Read」の「Item」にアイテム名（デバイス名）を入力し、「Request」ボタンを押すと行えます。また、自動読出「AutoRead」ボタンを押すと常時読み出しになります。

書込処理は「Direct Write」の「Item」にアイテム名（デバイス名）を入力し、「Value」に値を入力し、「Poke」ボタンを押すと、PLC に書き込まれます。

アスキー形式、ダブルワード形式などでアクセスする場合は、デバイス種を選択し、「Number」にデバイス番号、「Points」にアクセス点数、「Type」にアクセス形式を選択することで、通常読出、自動読出、書き込みが行えます。

コネクションを切断する場合は、「DisConnect」ボタンを押してください。

TAKEBISHI Software Library
DeviceXPlorer
Client Sample Guide
ユーザーズマニュアル

株式会社たけびし
〒615-8501 京都市右京区西京極豆田町 29
TEL (075) 325-2172
Email fa-support@takebishi.co.jp

DxpSERVER

41094-C104-Rev P May 2021