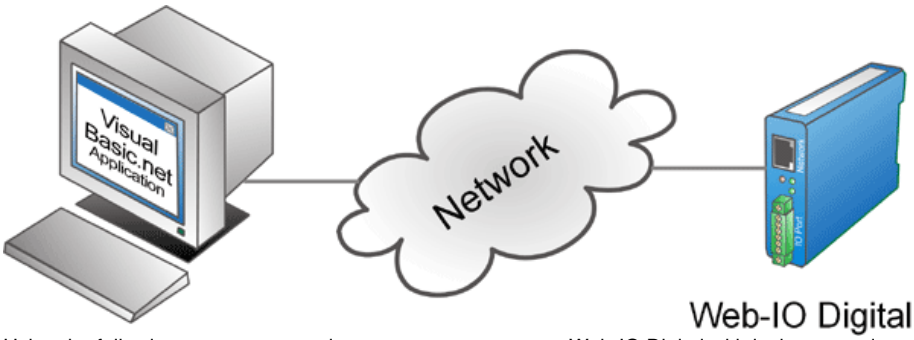
Application for the Web-IO Digital:

Control and monitor Web-IO Digital with VB.Net

VB.net has established itself as the successor to MS Visual Basic. Visual Basic.net offers everything needed for programming TCP/IP applications. This makes Visual Basic.net a favorite tool for creating applications which communicate with the Web-IO Digital. Additional drivers or DLLs are not required.



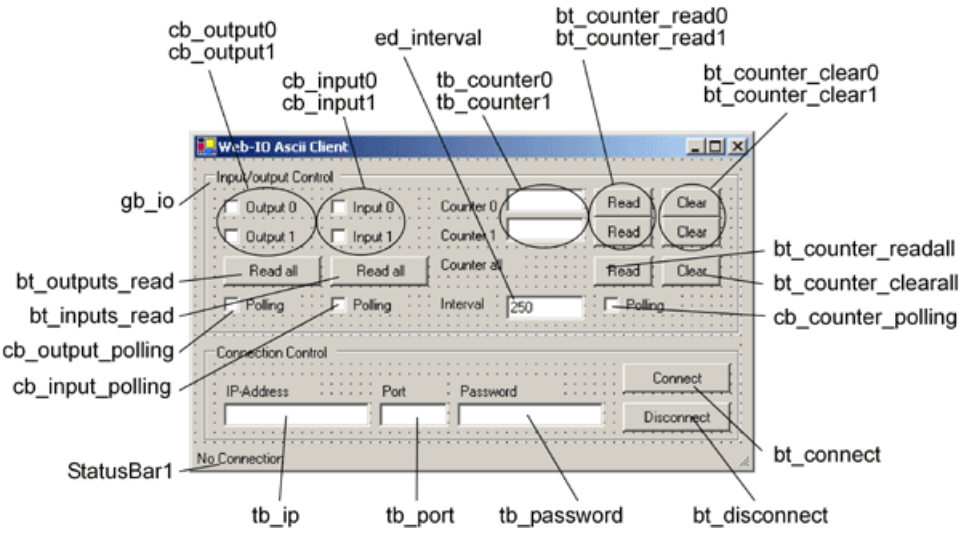
Using the following program example you can represent your Web-IO Digital with its inputs and outputs in a Windows application. You can also switch the Web-IO outputs.

Preparations

You have already provided your Web-IO Digital

- with power,
- connected inputs and outputs
- connected it to your network,
- assigned it an IP address - which with WuTility is no problem.

1. Combining the various operating elements and display objects in the VB.net form



In addition to the objects shown here, the program also needs a timer for polling (timer_polling).

When naming the individual objects it is helpful to use logical names. In this example the first part of the name describes the type of object and the second part the function.

1. Starting the program

While the structure of graphical user interfaces in VB.net is just as easy to accomplish as with VB5 or VB6, the network communication under VB.net is somewhat more difficult to construct than with its predecessor. This is primarily because Microsoft no longer provides the Winsock control element with VB.net. Instead, network access requires that the *Import* Statement be used to instance the namespace for the socket classes used in the header of the source text.

In addition, the socket on which communication will be handled must be created and a buffer for the input data needs to be defined.

```
Imports System.Net
Imports System.Net.Sockets
Imports System.IO
Imports System.Windows.Forms
Public Class Form1
 Inherits System.Windows.Forms.Form
 Dim TCP_client As Socket
 Dim receivebuffer(511) As Byte
```

2. Connection control

Establishing the connection
The connection is opened by entering the IP address of the Web-IO in the text field *ed_ip* and clicking on the *bt_connect* button.

```
Private Sub bt_connect_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_connect.Click
 Dim webioep As New IPEndPoint(IPAddress.Parse(tb_ip.Text), Val(tb_port.Text))
 If tb_ip.Text <> "" And tb_port.Text <> "" Then
 TCP_client = New Socket(AddressFamily.InterNetwork, SocketType.Stream, _
 ProtocolType.Tcp)
 bt_connect.Enabled = False
  Try
   TCP_client.BeginConnect(webioep, New AsyncCallback(AddressOf callback_connect), _
   TCP_client)
  Catch ex As Exception
   closeconnections()
   MessageBox.Show(ex.Message, "Socket Error", MessageBoxButtons.OK, _
   MessageBoxIcon.Error)
  End Try
 End If
End Sub
```

Opening the connection
For TCP/IP handling first an IPEndPoint is defined from IP address and TCP port and used to initialize the TCP_client socket. As part of the connection request a reference to a callback procedure is created.

Connection is made
As soon as a connection is made with the Web-IO, the callback procedure is run. The status line shows that the connection has been opened, the operating elements are enabled for use and the Disconnect button becomes operable. The Connect button is blocked for operation. In addition a reference to a callback routine for data reception is created.

```
Private Sub callback_connect(ByVal ar As IAsyncResult)
 bt_connect.Enabled = False
 gb_io.Enabled = True
 bt_disconnect.Enabled = True
 StatusBar1.Text = "Connected to " + tb_ip.Text + " : " + tb_port.Text
 timer_polling.Enabled = True
  Try
  TCP_client.EndConnect(ar)
  TCP_client.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, New AsyncCallback(AddressOf callback_readdat
  Catch ex As Exception
  closeconnections()
  StatusBar1.Text = "error on connecting"
  End Try
End Sub
```

Disconnecting
The connection remains open until it is ended by the user clicking on the Disconnect button, or the Web-IO ends the connection.

```
Private Sub bt_disconnect_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_disconnect.Click
 bt_disconnect.Enabled = False
 closeconnections()
End Sub
```

In this case a corresponding procedure is invoked.

```
Private Sub closeconnections()
 Dim ar As IAsyncResult
 timer_polling.Enabled = False
 Try
  TCP_client.EndReceive(ar)
 Catch ex As Exception
 End Try
 Try
  TCP_client.Shutdown(SocketShutdown.Both)
 Catch ex As Exception
 End Try
 Try
  TCP_client.Close()
 Catch ex As Exception
 End Try
 bt_connect.Enabled = True
 gb_io.Enabled = False
 bt_disconnect.Enabled = False
 StatusBar1.Text = "no connection"
End Sub
```

Connection error
All the actions affecting TCP/IP communication are carried out within the Try-instruction. If errors occur, the CloseConnection procedure is also invoked.

3. Operation and communication from the client side

As soon as a connection is made with the Web-IO, the user can use the corresponding program elements to send commands to the Web-IO.

```
Private Sub sendcommand(ByVal sendstring As String)
 Dim senddata As Byte() = System.Text.Encoding.ASCII.GetBytes(sendstring)
 Try
 TCP_client.Send(senddata)
 Catch ex As Exception
 closeconnections()
 End Try
End Sub
```

Setting the outputs
The user sets the outputs by using the two check boxes `cb_outputx`. The program uses the MouseUP event of this object. If a MouseUp, i.e. releasing the output check box, is used, the program carries out the corresponding procedure and - depending on whether the check box is set or not - passes the appropriate command to the Web-IO.

```
Private Sub cb_output0_MouseUp(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) Handles cb_output0.MouseUp
 If cb_output0.Checked Then
 sendcommand("GET /outputaccess0?PW=" + tb_password.Text + "&State=ON&")
 Else
 sendcommand("GET /outputaccess0?PW=" + tb_password.Text + "&State=OFF&")
 End If
End Sub
```

```
Private Sub cb_output1_MouseUp(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) Handles cb_output1.MouseUp
 If cb_output1.Checked Then
 sendcommand("GET /outputaccess1?PW=" + tb_password.Text + "&State=ON&")
 Else
 sendcommand("GET /outputaccess1?PW=" + tb_password.Text + "&State=OFF&")
 End If
End Sub
```

Query output/input status
The user can request the status of the outputs and inputs by clicking on the corresponding button.

```
Private Sub bt_outputs_read_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_outputs_read.Click
 sendcommand("GET /output?PW=" + tb_password.Text + "&")
End Sub



Private Sub bt_inputs_read_Click(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles bt_inputs_read.Click
 sendcommand("GET /input?PW=" + tb_password.Text + "&")
End Sub
```

Read/clear counters
Also the counter states of the input counters can be read or cleared.

```
Private Sub bt_counter_read0_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_read0.Click
 sendcommand("GET /counter0?PW=" + tb_password.Text + "&")
End Sub



Private Sub bt_counter_read1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_read0.Click
 sendcommand("GET /counter1?PW=" + tb_password.Text + "&")
End Sub



Private Sub bt_counter_clear0_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_clear0.Click
 sendcommand("GET /counterclear0?PW=" + tb_password.Text + "&")
End Sub



Private Sub bt_counter_clear1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_clear0.Click
 sendcommand("GET /counterclear1?PW=" + tb_password.Text + "&")
End Sub
```

Of course all the counters can be read or cleared at the same time.

```
Private Sub bt_counter_readall_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_readall.Click
 sendcommand("GET /counter?PW=" + tb_password.Text + "&")
End Sub
```

```
Private Sub bt_counter_clearall_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles bt_counter_clearall.Click
 sendcommand("GET /counterclear?PW=" + tb_password.Text + "&")
End Sub
```

3. Data reception from the Web-IO

- Process and display the received data
  All commands and requests to the Web-IO are acknowledged with a reply string. The replies have a specific structure depending on the type.
- For the outputs: output;<binary value of the output status in hexadecimal format>
- For the inputs: input;<binary value of the input status in hexadecimal format>
- For the counters: counterx;<decimal counter state>
- or counter;<decimal counter state 0 >; <decimal counter state 0 >;.....if you want to read all counters at the same time.
- All reply strings are finished off with a 0 byte.
- When data are received the corresponding Callback procedure is invoked

```
Private Sub callback_readdata(ByVal ar As IAsyncResult)
 If TCP_client.Connected Then
 Dim bytesread As Integer
 Try
 bytesread = TCP_client.EndReceive(ar)
 Catch ex As Exception
 End Try
 If bytesread = 0 Then
 closeconnections()
 Else
 Dim receivestring As String
 receivestring = System.Text.Encoding.ASCII.GetString(receivebuffer, 0, _
 receivebuffer.Length)
 receivebuffer.Clear(receivebuffer, 0, 512)
 Try
  TCP_client.BeginReceive(receivebuffer, 0, 512, SocketFlags.None, New AsyncCallback(AddressOf callback_readdat
 Catch ex As Exception
  closeconnections()
 End Try
 Select Case Mid(receivestring, 1, 1)
 Case "i"
  If (Val(Mid(receivestring, 7, 1)) And 1) = 1 Then
  cb_input0.Checked = True
  Else
  cb_input0.Checked = False
  End If
  If (Val(Mid(receivestring, 7, 1)) And 2) = 2 Then
  cb_input1.Checked = True
  Else
  cb_input1.Checked = False
  End If
 Case "o"
  If (Val(Mid(receivestring, 8, 1)) And 1) = 1 Then
  cb_output0.Checked = True
  Else
  cb_output0.Checked = False
  End If
  If (Val(Mid(receivestring, 8, 1)) And 2) = 2 Then
  cb_output1.Checked = True
  Else
  cb_output1.Checked = False
  End If
 Case "c"
  Dim tabpos
  If Mid(receivestring, 8, 1) = "0" Then _
   tb_counter0.Text = Mid(receivestring, 10)
  If Mid(receivestring, 8, 1) = "1" Then _
   tb_counter1.Text = Mid(receivestring, 10)
  If Mid(receivestring, 8, 1) = ";" Then
   tabpos = InStr(9, receivestring, ";")
   tb_counter0.Text = Mid(receivestring, 9, tabpos - 9)
   tb_counter1.Text = Mid(receivestring, tabpos + 1, _
   Len(receivestring) - tabpos - 1)
  End If
 End Select
 End If
 End If
End Sub
```

The Receive procedure uses the first character of the receive data to check whether these are input, output or counter messages. Depending on this, a determination is made for example as to which output has which status. With the counters it is possible both to query individual counter values as well as to read out all the counters at one time. The individual counter states are then output in decimal format in a semicolon delimited string.

4. Polling

- Cyclical polling of particular values
  In order to enable automatic refreshing of the display, a timer is used.
- Depending on the check boxes for output, input and counter polling, the corresponding information is obtained from the Web-IO at a set interval.

```
Private Sub timer_polling_Elapsed(ByVal sender As System.Object, _
ByVal e As System.Timers.ElapsedEventArgs) Handles timer_polling.Elapsed
 If (cb_input_polling.Checked And TCP_client.Connected) Then
 sendcommand("GET /input?PW=" + tb_password.Text + "&")
 End If
 If (cb_output_polling.Checked And TCP_client.Connected) Then
 sendcommand("GET /output?PW=" + tb_password.Text + "&")
 End If
 If (cb_output_polling.Checked And TCP_client.Connected) Then
 sendcommand("GET /output?PW=" + tb_password.Text + "&")
 End If
 If (cb_counter_polling.Checked And TCP_client.Connected) Then
 sendcommand("GET /counter?PW=" + tb_password.Text + "&")
 End If
End Sub
```

The desired interval can be entered in the corresponding text field. When changes are made the timer interval is automatically adjusted.

```
Private Sub tb_interval_TextChanged(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles tb_interval.TextChanged
 timer_polling.Interval = Val(tb_interval.Text)
End Sub
```

The sample program supports all common functions of the Web-IO in command string mode, optimized for the Web-IO 2x Digital Input, 2x Digital Output PoE For the other Web-IO models you may have to adapt the program. Additional program examples for socket programming can be found on the Tool pages for the Web-IO. A detailed description for the socket interface of the Web-IO Digital models can be found in the reference manual.

Download program example

You don’t have a Web-IO® yet but would like to try the example out sometime?
No problem: We will be glad to send you the Web-IO Digital 2xInput, 2xOutput at no charge for 30 days. Simply fill out a sample ordering form, and we will ship the Web-IO for testing on an open invoice. If you return the unit within 30 days, we will simply mark the invoice as paid.

To sample orders

W&T
www.WuT.de

We are available to you in person:

Wiesemann & Theis GmbH
Porschestr. 12
42279 Wuppertal
Phone: +49 202/2680-110 (Mon.-Fri. 8 a.m. to 5 p.m.)
Fax: +49 202/2680-265
info@wut.de