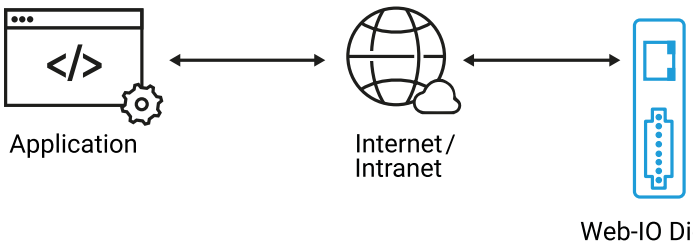
Application for the Web-IO Digital:

Control and monitor Web-IO Digital with Visual Basic

Visualizing, checking and controlling

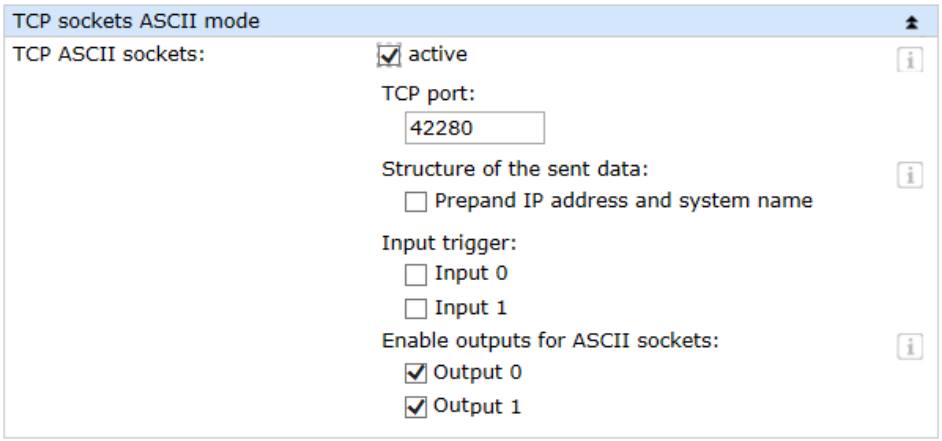
As an easy to learn higher language Visual Basic offers everything you need for programming TCP/IP applications. This also makes Visual Basic a favorite tool for creating applications that communicate with the Web-IO Digital. Additional drivers or DLLs are not required.



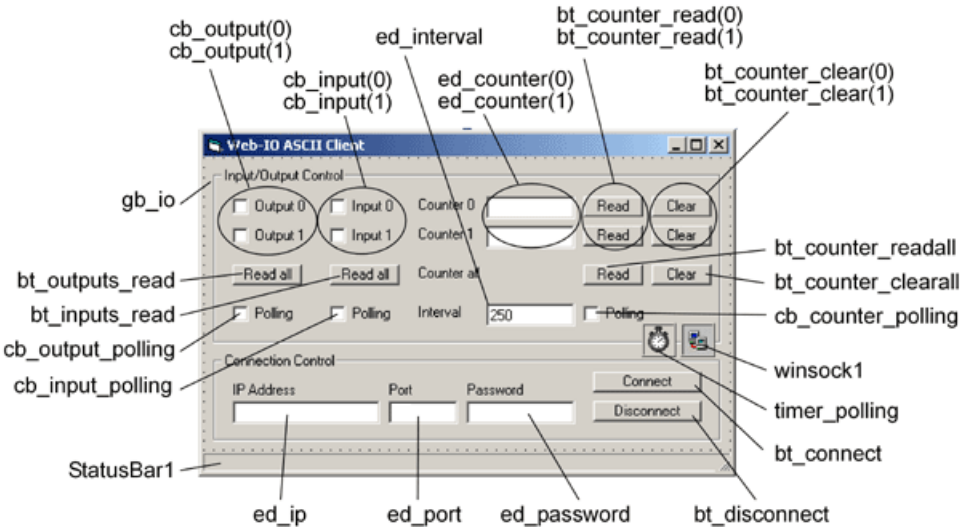
Using the following program example you can represent your Web-IO Digital with its inputs and outputs in a Windows application. You can also switch the Web-IO outputs.

Preparations

- Provide power to the Web-IO and connect the IOs
- Connect the Web-IO to the network
- Assign IP addresses
- On the Web-IO in *Communication channels >> Socket API* activate *TCP-ASCII sockets* and *enable outputs for switching*



Combining the various operating elements and display objects in the VB form



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.

Starting the program

Setting up the operating elements
At first the group with the operating elements for the Web-IO is blocked from use. The status line indicates that there is not yet a connection.

```
Private Sub Form_Load()
  fr_io.Enabled = False
  bt_disconnect.Enabled = False
  StatusBar1.SimpleText = "No Connection"
End Sub
```

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()
  If ed_ip.Text <> "" Then
    winsock1.RemoteHost = ed_ip.Text
    winsock1.RemotePort = Val(ed_port.Text)
    winsock1.Connect
  End If
End Sub
```

Opening the connection
The Winsock control of Visual Basic is used for TCP/IP handling. This control allows procedures to be specified and started when various connection states occur.

Connection is made
As soon as a connection is made with the Web-IO, the Winsock control carries out the corresponding procedure. The status line indicates that the connection has been established, the operating elements are enabled for use and the Disconnect button becomes operable again. The Connect button is disabled.

```
Private Sub winsock1_Connect()
  fr_io.Enabled = True
  bt_disconnect.Enabled = True
  bt_connect.Enabled = False
  StatusBar1.SimpleText = "Connected to " + ed_ip.Text
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()
  winsock1.Close
  fr_io.Enabled = False
  bt_disconnect.Enabled = False
  bt_connect.Enabled = True
  StatusBar1.SimpleText = "No Connection"
End Sub
```

Here again the Winsock control invokes a corresponding procedure

```
Private Sub winsock1_Close()
  fr_io.Enabled = False
  bt_disconnect.Enabled = False
  bt_connect.Enabled = True
  StatusBar1.SimpleText = "No Connection"
  winsock1.Close
End Sub
```

Connection error
Also in case of a connection error the Winsock control carries out a corresponding procedure which is essentially like the Disconnect procedure. The status line also indicates the Winsock error number.

```
Private Sub winsock1_Error(ByVal Number As Integer, Description As String, _
ByVal Scode As Long, ByVal Source As String, ByVal HelpFile As String, _
ByVal HelpContext As Long, CancelDisplay As Boolean)
  fr_io.Enabled = False
  bt_disconnect.Enabled = False
  bt_connect.Enabled = True
  StatusBar1.SimpleText = "ERROR " + Str(Number) + " : " + Description
  winsock1.Close
End Sub
```

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

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 Mouse Up, 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_output_MouseUp(Index As Integer, Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If cb_output(Index).Value Then
    winsock1.SendData ("GET /outputaccess" + Trim(Str(Index)) + _
    "?PW=" + ed_password.Text + "&State=ON&")
  Else
    winsock1.SendData ("GET /outputaccess" + Trim(Str(Index)) + _
    "?PW=" + ed_password.Text + "&State=OFF&")
  End If
End Sub
```

Querying 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()
  winsock1.SendData ("GET /output?PW=" + ed_password.Text + "&")
End Sub

Private Sub bt_inputs_read_Click()
  winsock1.SendData ("GET /input?PW=" + ed_password.Text + "&")
End Sub
```

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

```
Private Sub bt_counter_read_Click(Index As Integer)
  winsock1.SendData ("GET /counter" + Trim(Str(Index)) + _
  "?PW=" + ed_password.Text + "&")
End Sub

Private Sub ed_counter_clear_Click(Index As Integer)
  winsock1.SendData ("GET /counterclear" + Trim(Str(Index)) + _
  "?PW=" + ed_password.Text + "&")
End Sub
```

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

```
Private Sub cb_counter_readall_Click()
  winsock1.SendData ("GET /counter?PW=" + ed_password.Text + "&")
End Sub

Private Sub cb_counter_clearall_Click()
  winsock1.SendData ("GET /counterclear?PW=" + ed_password.Text + "&")
End Sub
```

Receiving data 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.

If data are received by the Winsock control, the latter invokes the corresponding procedure

```
Private Sub winsock1_DataArrival(ByVal bytesTotal As Long)
  Dim receivestring As String
  winsock1.GetData receivestring
  Select Case Left(receivestring, 1)
    Case "i"
      If (Val(Mid(receivestring, 7, 1)) And 1) = 1 Then
        cb_input(0).Value = 1
      Else
        cb_input(0).Value = 0
      End If
      If (Val(Mid(receivestring, 7, 1)) And 2) = 2 Then
        cb_input(1).Value = 1
      Else
        cb_input(1).Value = 0
      End If
    Case "o"
      If (Val(Mid(receivestring, 8, 1)) And 1) = 1 Then
        cb_output(0).Value = 1
      Else
       cb_output(0).Value = 0
      End If
      If (Val(Mid(receivestring, 8, 1)) And 2) = 2 Then
        cb_output(1).Value = 1
      Else
        cb_output(1).Value = 0
      End If
    Case "c"
      Dim tabpos
      If Mid(receivestring, 8, 1) = "0" Then
        ed_counter(0).Text = Mid(receivestring, 10, Len(receivestring) - 10)
      End If
      If Mid(receivestring, 8, 1) = "1" Then
       ed_counter(1).Text = Mid(receivestring, 10, Len(receivestring) - 10)
      End If
      If Mid(receivestring, 8, 1) = ";" Then
        tabpos = InStr(9,receivestring, ";")
        ed_counter(0).Text = Mid(receivestring,9, tabpos - 9)
        ed_counter(1).Text = Mid(receivestring,tabpos + 1, _
        Len(receivestring) - tabpos - 1)
      End If
  End Select
End Sub
```

The receiving procedure uses the first character of the receive data to check whether this is an input, output or counter message. Depending on this it is determined for example which output has which status. In the case of the counters it is also possible to read individual counter values or to read out all the counters at once. The individual counter states are then output in decimal format in a semicolon delimited string.

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_Timer()
  If winsock1.State = sckConnected Then
    If cb_output_polling.Value Then
      winsock1.SendData ("GET /output?PW="+ ed_password.Text + "&")
    End If
    If cb_input_polling.Value Then
      winsock1.SendData ("GET /input?PW="+ ed_password.Text + "&")
    End If
    If cb_counter_polling.Value Then
      winsock1.SendData ("GET /counter?PW="+ ed_password.Text + "&")
    End If
  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 ed_interval_Change()
  timer_polling.Interval = Val(ed_interval)
End Sub
```

Of course the Web-IO can also be configured to autonomously send messages from the program when something changes on one of the inputs.

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. 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

Products

Web-IO 4.0 Digital
2xIn, 2xOut

Power via PoE also when needed

Web-IO 4.0 Digital
12xIn, 12xOut

12x inputs,
12x outputs

Other Web-IOs

All W&T Web-IO Digital 24V

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