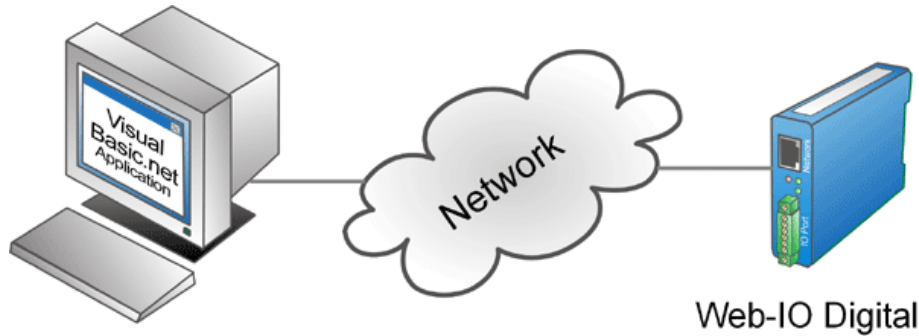


Aplicación al Web-IO digital:

Dirigir y supervisar Web-IO digital con VB.Net

- Resumen de productos
- Otras aplicaciones

Como sucesor de MS Visual Basic se ha establecido entre tanto VB.net. Visual Basic.net ofrece todo lo que es necesario para programar aplicaciones TCP/IP. Con ello Visual Basic.net es un auxiliar apreciado para crear aplicaciones que comunican con el [Web-IO digital](#). No se necesitan excitadores o DLLs adicionales.



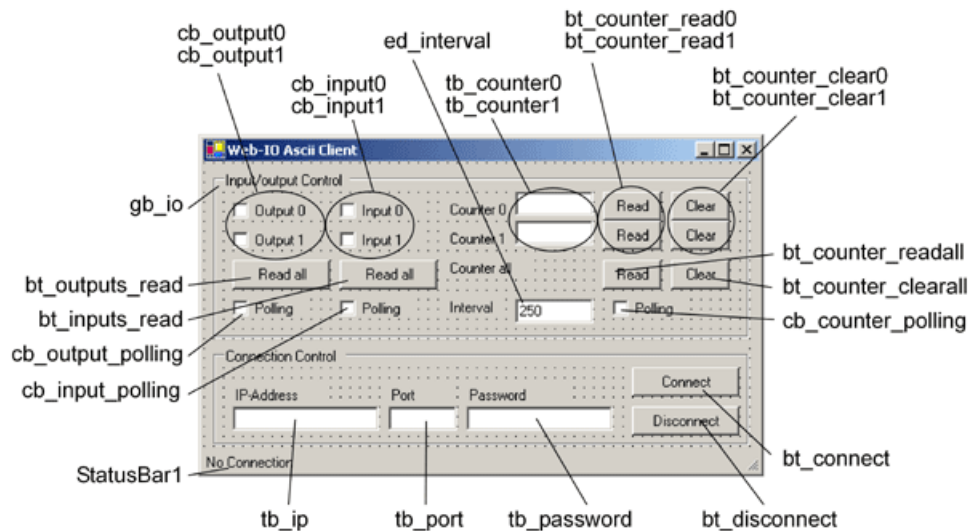
Con el ejemplo siguiente de programa puede representar su Web-IO Digital con sus Inputs y Outputs en una aplicación Windows. Además de ello puede conmutar los Outputs del Web-IO.

Preparativos

Ya ha abastecido su Web-IO Digital

- [con corriente](#)
- [entradas y salidas conectadas](#)
- [conectado a su red](#)
- dotado con una dirección IP - con [WuTility](#) no hay problemas

1. Recopilación de los diferentes elementos de manejo y objetos de visualización en formulario VB.net



Además de los objetos aquí mostrados el programa necesita además un Timer para el Polling (timer_polling).

Al denominar cada uno de los objetos, es de gran ayuda usar nombres con sentido. En este ejemplo la primera parte del nombre describe la clase de objeto y la segunda la función.

1. Arranque de programa

Mientras que la formación de superficies gráficas de usuario en VB.net es igual de fácil de hacer que con VB5 o VB6, resulta algo más difícil la comunicación de red en VB.net que es el predecesor. Esto reside en primera línea en que Microsoft en VB.net no pone más a disposición el elemento de mando Winsock. En lugar de eso se tiene que instanciar para el acceso de red a través del *Import* Statement el espacio del nombre para las clases usadas de zócalo en el cabezal del texto de origen.

Además de ello se tiene que instalar el zócalo en el que se debe tramitar la comunicación, y tiene que definirse un Buffer para los datos de entrada.

```
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. El control de conexión

Introducir la conexión

La conexión se arranca entrando la dirección IP del Web-IO en el campo de `textæd_ip` y chasqueando el botón `bt_connect`.

```
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
```

La conexión

Para la tramitación del manejo TCP/IP se define primero un `IPEndPoint` de la dirección Ip y puerto TCP y así inicializar el zócalo `TCP_client`. En el transcurso de la solicitud de conexión se crea una referencia a un procedimiento `Callback`.

Se establece la conexión

Tan pronto como el Web-IO acepta la conexión, se ejecuta el procedimiento `Callback`. En la línea de estado aparece la realización de la conexión, los elementos de manejo se liberan para el uso y puede manejarse el botón `Disconnect`. El botón `Connect` se bloquea para el manejo. Además de ello se crea una referencia a una rutina `Callback` para la recepción de datos.

```
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_readdata), TCP_client)
Catch ex As Exception
closeconnections()
StatusBar1.Text = "error on connecting"
End Try
End Sub
```

Separar la conexión

La conexión permanece tanto tiempo hasta que el usuario la finalice chasqueando el botón `Disconnect` o el Web-IO finalice la conexión.

```
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
```

En este caso se llama un procedimiento correspondiente.

```
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
```

Error de conexión

Todas las acciones correspondientes a la comunicación TCP/Ip se ejecutan dentro de la orden `Try`. Si aparecen errores, se llama igualmente el procedimiento `CloseConnection`.

3. Manejo y comunicación por parte del cliente

Tan pronto como se ha establecido la conexión con el Web-IO, el usuario puede enviar comandos al Web-IO manejando los correspondientes elementos de programa

```
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
```

Poner Outputs

El usuario puede poner los Outputs a través de dos casillas de verificación *cb_outputx*. El programa utiliza para ello el suceso de MouseUP de este objeto. Si se registra un Mouse Up, es decir soltar la casilla de verificación Output, el programa realiza el correspondiente procedimiento y transmite el adecuado comando al Web-IO - según si está puesta la casilla o no.

```
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
```

Solicitar estado de Output/Input

El usuario puede solicitar el estado de los Outputs e Inputs chasqueando el botón correspondiente.

```
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
```

Contadores preguntar/borrar

También se pueden preguntar o borrar los estados de contador de los Input-Counter.

```
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
```

Naturalmente también se pueden leer o borrar al mismo tiempo todos los contadores.

```

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. Recepción de datos del Web-IO

Evaluar e indicar los datos recibidos

- Todos los comandos y solicitudes al Web-IO se confirman con un String de respuesta. Aquí las respuestas presentan una estructura específica según el tipo.
- Para los Outputs: output;<valor binario del estado de salida en formato hexadecimal>
- Para los Inputs: input;<valor binario del estado de salida en formato hexadecimal>
- Para los contadores: counterx;<estado decimal de contador >
- o counter;<estado decimal de contador 0 >; <estado decimal de contador 0 >; si todos los contadores se deben leer de una sola vez.
- Todos los Strings de respuesta se finalizan con un Byte 0.
- Al recibir datos se llama el procedimiento correspondiente Callback.

```

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_readdata), TCP_client)
            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

```

El procedimiento de recepción verifica según el primer signo de los datos de recepción, si se trata de mensajes de Input, Output o del contador. Dependiendo de ello se constata p. ej. qué Output tiene qué estado. En los contadores es posible tanto preguntar valores de contador separados como también leer todos los contadores de un viaje. Cada uno de los estados de contador se emiten decimalmente separados por punto y coma en una cadena (String).

4. Polling

Solicitud cíclica de determinados valores

A fin de posibilitar una actualización automática de la indicación, se utiliza un Timer.

Dependiendo de las casillas de verificación para el Polling de Output, Input y Counter se llaman las informaciones correspondientes a un intervalo ajustado del Web-IO.

```
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
```

El intervalo deseado puede entrarse en el campo correspondiente de texto. En caso de un cambio el intervalo del Timer se adapta automáticamente.

```
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
```

El [programa ejemplo](#) asiste todas las funciones corrientes del Web-IO en el modo String de comando, optimado para el [Web-IO 2x Entradas digitales, 2x Salidas digitales](#). Para los otros modelos Web-IO tienen que realizarse en caso necesario adaptaciones en el programa. Otros ejemplos de programa para la programación del zócalo los encontrarán en las [páginas de herramientas](#) al Web-IO. Una descripción detallada de la interfaz del zócalo de los modelos Web-IO digitales la encontrarán en el [manual de referencia](#).



Le atendemos personalmente:

Wiesemann & Theis
GmbH
Porschestra. 12
42279 Wuppertal
Tel: +49 202/2680-110 (lu-vi de 8-17
horas)
Fax: +49-202/2680-265
info@wut.de

© Wiesemann & Theis GmbH, salvo errores y modificaciones: como podemos cometer errores, no se debe utilizar nuestros enunciados sin verificarlos. Por favor, notifíquenos todas las erratas y malentendidos que detecte, para que podamos localizarlo y solucionarlo lo antes posible.

[Protección de datos](#)