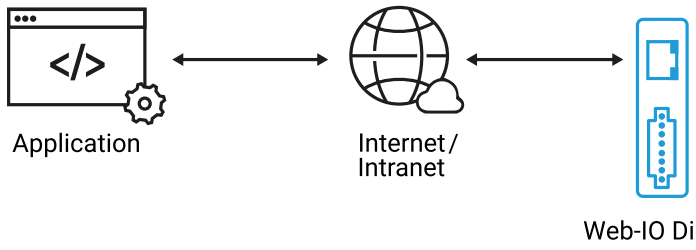
Aplicación al Web-IO digital:

Dirigir y controlar Web-IO digital con Visual C#

Resumen de productos
Sinopsis de aplicaciones

Para la creación de aplicaciones Windows, Visual C++ ha sido hasta hace poco una de las plataformas de desarrollo más utilizadas. Entre tanto trabajan cada vez más programadores con el .Net Framework y crean sus aplicaciones en C# (C Sharp).



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

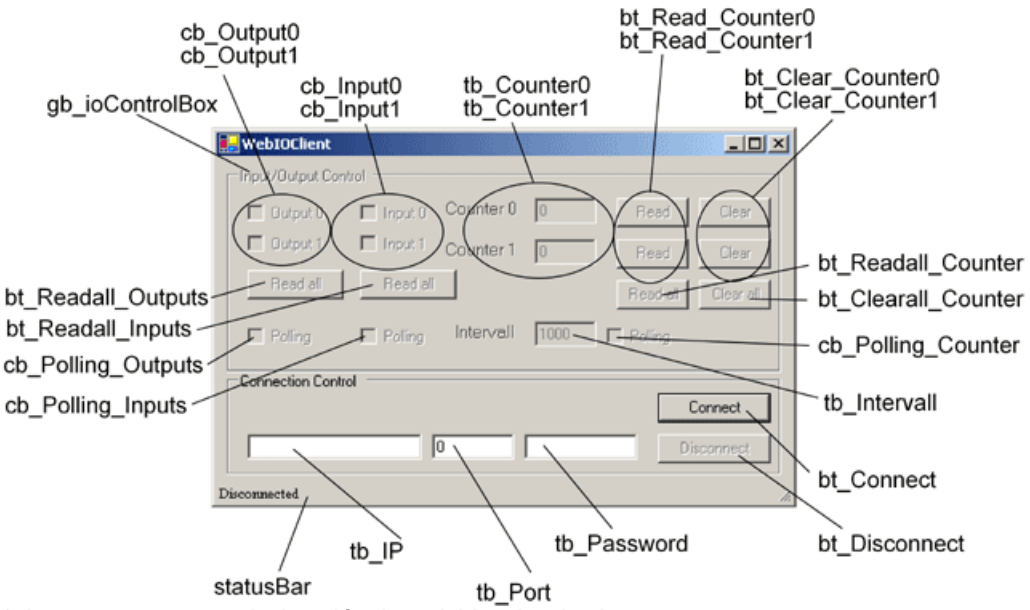
¿No tiene todavía un Web-IO® y quiere probar el ejemplo presentado?
No hay problema: Le ponemos a disposición el Web-IO Digital 2xInput, 2xOutput gratis durante 30 días. Rellene sencillamente un pedido muestra y le enviaremos el Web-IO para probar a cuenta abierta. Si nos devuelve el aparato dentro de los 30 días, le abonamos la factura completa.

Al pedido muestra

Preparativos

Ya ha abastecido su Web-IO Digital
- con corriente
- entradas y salidas conectadas
- conectado a su red
- dotado con una dirección IP - conWuTility no hay problemas

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



2. Importar recursos y declaración de variables de miembros

En primer lugar se importan todas las clases necesarias para la conexión de red y la GUI(Graphical User Interface).

```
        using System;
        using System.Drawing;
        using System.Collections;
        using System.ComponentModel;
        using System.Windows.Forms;
        using System.Data;
        using System.Net;
        using System.Net.Sockets;
        using System.Text;
```

A continuación se declaran los componentes de la aplicación y las variables importantes para una conexión TCP como variables de miembros de la clase, teniendo acceso así a los métodos de la clase.

```
private System.Windows.Forms.CheckBox
    cb_Output0;
    private System.Windows.Forms.CheckBox
    cb_Output1;
    private System.Windows.Forms.CheckBox
    cb_Input0;
    private System.Windows.Forms.CheckBox
    cb_Input1;
    private System.Windows.Forms.CheckBox
    cb_Polling_Counter;
    private System.Windows.Forms.CheckBox
    cb_Polling_Outputs;
    private System.Windows.Forms.CheckBox
    cb_Polling_Inputs;
    private System.Windows.Forms.Button
    bt_Readall_Outputs;
    private System.Windows.Forms.Button
    bt_Readall_Inputs;
    private System.Windows.Forms.Button
    bt_Clear_Counter0;
    private System.Windows.Forms.Button
    bt_Clear_Counter1;
    private System.Windows.Forms.Button
    bt_Clearall_Counter;
    private System.Windows.Forms.Button
    bt_Read_Counter0;
    private System.Windows.Forms.Button
    bt_Read_Counter1;
    private System.Windows.Forms.Button
    bt_Readall_Counter;
    private System.Windows.Forms.Label
    lb_Counter0;
    private System.Windows.Forms.Label
    lb_Counter1;
    private System.Windows.Forms.Label
    lb_Intervall;
    private System.Windows.Forms.TextBox
    tb_Counter0;
    private System.Windows.Forms.TextBox
    tb_Counter1;
    private System.Windows.Forms.TextBox
    tb_Intervall;
    private System.Windows.Forms.Button
    bt_Connect;
    private System.Windows.Forms.Button
    bt_Disconnect;
    private System.Windows.Forms.TextBox
    tb_Password;
    private System.Windows.Forms.TextBox
    tb_Port;
    private System.Windows.Forms.TextBox
    tb_IP;
    private System.Windows.Forms.GroupBox
    gb_ioControlBox;
    private System.Windows.Forms.GroupBox
    gb_conControlBox;
    private System.Windows.Forms.StatusBar
    statusBar;
    private System.ComponentModel.Container
    components = null;
    private System.Windows.Forms.Timer
    counter;
    private System.Windows.Forms.Timer
    outputs;
    private System.Windows.Forms.Timer
    inputs;

    private Socket client;
    private string rcv;
    private int intervall;
    private byte[] buffer = new byte[256];
```

3. Arranque de programa

Instalar los elementos de manejo
El grupo con los elementos de manejo para el Web-IO se bloquea primero para el manejo. Tan pronto como se establezca una conexión, se da línea libre a todos los elementos, que poseen una versión conveniente.

El nombre del elemento respectivo de manejo se deriva del elemento mismo según el contexto. Los dos primeros signos del nombre significan el tipo del elemento (cb -> Checkbox, bt -> Button, gb -> Groupbox y tb-> TextBox).

```
public mainWindow()
    {
     InitializeComponent();

     gb_ioControlBox.Enabled = false;
     bt_Disconnect.Enabled = false;
     cb_Input0.Enabled = false;
     cb_Input1.Enabled = false;
     tb_Counter0.Enabled = false;
     tb_Counter1.Enabled = false;
    }
```

4. El control de conexión

Introducir la conexión

Entrando la dirección IP del Web-IO en el campo de texto tb_IP y en el puerto 80 en el campo de texto tb_Port se puede establecer una conexión activando el botón bt_Connect. Si no se entra ninguna dirección IP o ningún puerto, se da un mensaje por la aplicación en la barra de estado.

La conexión
Para poder establecer una conexión TCP se inicializa la variable ya declarada de zócalo. Aquí se le entrega un Stream y la clase de conexión. Además se crea una variable, que salva el puerto y la dirección IP. Para que el programa pueda trabajar asincrónicamente, no espera a sucesos sino que trabaja con rutinas Callback. Se inicializan métodos Callback, si se arranca un proceso y se solicita, si llega a darse el suceso correspondiente, p. ej. al establecer la conexión, al enviar o al recibir.

```
private void bt_Connect_Click(object sender, System.EventArgs e)
    {
     try
     {
     if((tb_IP.Text != "") && (tb_Port.Text == "80"))
     {
      client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
      IPEndPoint ipep = new IPEndPoint(IPAddress.Parse(tb_IP.Text), int.Parse(tb_Port.Text));
      client.BeginConnect(ipep, new AsyncCallback(connectCallback), client);
     }
     else
      statusBar.Text = "IP and Port needed!";
     }
     catch(SocketException)
     {
     statusBar.Text = "Connection not possible!";
     }
    }
```

A continuación se representa la rutina Callback, que se llama al darse la conexión. Tras una conexión positiva se da línea libre a todos los elementos de manejo útiles de la aplicación y se desactiva el botón Connect. Además la aplicación comienza inmediatamente a pasar a disposición de recepción. Para ello se inicializa la rutina Callback "receiveCallback", que se explicará más abajo.

```
private void connectCallback(IAsyncResult ar)
    {
     try
     {
     client = (Socket) ar.AsyncState;
     client.EndConnect(ar);
     statusBar.Text = "Connected!";

     gb_ioControlBox.Enabled = true;
     bt_Disconnect.Enabled = true;
     bt_Connect.Enabled = false;

     intervall = 1000;
     client.BeginReceive(buffer, 0, 255, SocketFlags.None, new AsyncCallback(receiveCallback), clie
     }
     catch(Exception)
     {
     disconnect(); statusBar.Text = "Error while connecting!";
     }
    }
```

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. Después de pulsar el botón se emite un mensaje de que se va a finalizar la conexión.

```
private void bt_Disconnect_Click(object
    sender, System.EventArgs e)
    {
     disconnect();
    }
```

Si se separa la conexión, tienen que colocarse de nuevo todos los elementos en la posición de partida. Entonces debe ser imposible pulsar el botón Disconnect.

```
private void disconnect()
    {
     try
     {
     client.Shutdown(SocketShutdown.Both);
     client.Close();
     statusBar.Text = "Disconnected!";

     gb_ioControlBox.Enabled = false;
     bt_Disconnect.Enabled = false;
     bt_Connect.Enabled = true;
     }
     catch(Exception)
     {
     statusBar.Text = "Not able to disconnect!";
     }
    }
```

Ahora se ha finalizado la conexión y la aplicación ha retrocedido a su estado de partida.

5. 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.

Al enviar una noticia al Web-IO se solicita una rutina Callback igual como al recibirla.

```
private void sendCallback(IAsyncResult ar)
    {
     try
     {
     Socket tmp_client = (Socket) ar.AsyncState;
     int bytessend = tmp_client.EndSend(ar);
     }
     catch(Exception)
     {
     statusBar.Text = "Error while sending";
     }
    }
```

Poner Outputs

Los Outputs del Web-IO pueden conectarse con ayuda de las dos cajas de chequeo cb_Output0 y cb_Output1. La caja de chequeo dispara una acción si se chasquea. Dependiendo de si la caja de chequeo ya está puesta, el Output se pone a On o a OFF.

```
private void cb_Output0_CheckedChanged(object sender, System.EventArgs e)
    {
     if(cb_Output0.Checked)
     send("GET /outputaccess0?PW=" + tb_Password.Text + "&State=ON&");
     else
     send("GET /outputaccess0?PW=" + tb_Password.Text + "&State=OFF&");
    }
private void cb_Output1_CheckedChanged(object sender, System.EventArgs e)
    {
     if(cb_Output1.Checked)
     send("GET /outputaccess1?PW=" + tb_Password.Text + "&State=ON&");
     else
     send("GET /outputaccess1?PW=" + tb_Password.Text + "&State=OFF&");
    }
```

Solicitar estado de Output/Input

```
private void bt_Readall_Outputs_Click(object sender, System.EventArgs e)
    {
     send("GET /output?PW=" + tb_Password.Text + "&");
    }
private void bt_Readall_Inputs_Click(object sender, System.EventArgs e)
    {
     send("GET /input?PW=" + tb_Password.Text + "&");
    }
```

Solicitar Contadores

El método siguiente envía una solicitud a un contador determinado y exige una respuesta con el estado actual del contador.

```
private void bt_Read_Counter0_Click(object sender, System.EventArgs e)
    {
     send("GET /counter0?PW=" + tb_Password.Text + "&");
    }
private void bt_Read_Counter1_Click(object sender, System.EventArgs e)
    {
     send("GET /counter1?PW=" + tb_Password.Text + "&");
    }
```

Naturalmente también se pueden solicitar todos los estados de contador con un solo comando.

```
private void bt_Readall_Counter_Click(object sender, System.EventArgs e)
    {
     send("GET /counter?PW=" + tb_Password.Text + "&");
    }
```

Reposición a cero de contadores

```
private void bt_Clear_Counter0_Click(object sender, System.EventArgs e)
    {
     send("GET /counterclear0?PW=" + tb_Password.Text + "&");
    }
private void bt_Clear_Counter1_Click(object sender, System.EventArgs e)
    {
     send("GET /counterclear1?PW=" + tb_Password.Text + "&");
    }
```

Naturalmente también se pueden reponer a cero todos los contadores con un sólo comando.

```
private void bt_Clearall_Counter_Click(object sender, System.EventArgs e)
    {
     send("GET /counterclear?PW=" + tb_Password.Text + "&");
    }
```

Puesto que todos los estados de contador se pueden leer o reponer a cero con un sólo comando, tiene que implementarse todavía un método, que procese el string de respuesta del Web-IO y que asigne su estado específico a cada contador en la aplicación.

```
private void readAndClearCounter(string data)
    {
     int j = 0;
     string[] counter = new string[12];
     for(int i = 0; i < data.Length-1; i++)
     {
     if((data[i].CompareTo(';')) == 0)
      j++;
     else
      counter[j] += data[i].ToString();
     }
     tb_Counter0.Text = counter[0];
     tb_Counter1.Text = counter[1];
    }
```

6. 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. Las respuestas presentan una estructura

específica según el tipo:

- Para los Outputs: output;<valor binario del estado de salida en formato hexadecimal>
- Para un Output especial: outputx;<ON u OFF>
- Para los Inputs: input;<valor binario del estado de salida en formato hexadecimal>
- Para un Input especial: inputx;<ON u OFF>
- Después hay también el string de respuesta para un contador que tiene el siguiente aspecto.
- 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.

En nuestra aplicación se llama el método receiveCallback() para recibir una noticia tal. En este método se lee y procesa el string de respuesta. Lo importante de esta función es la llamada dirigida por el evento, que se da tan pronto como el Web-IO envíe datos a la aplicación.

```
private void receiveCallback(IAsyncResult ar)
    {
     int bytesRead;
    try
    {
     bytesRead = client.EndReceive(ar);
     rcv = string.Empty;
     rcv = Encoding.ASCII.GetString(buffer);

     client.BeginReceive(buffer, 0, 255, SocketFlags.None, new AsyncCallback(receiveCallback), clie
    }
    catch(Exception)
    {
     bytesRead = 0;
    }
    if(bytesRead == 0 && client.Connected)
     disconnect();
    else if(rcv != null)
    {
     if(rcv[0] == 'o')
     {
     int i = Int32.Parse(rcv.Substring(7), System.Globalization.NumberStyles.HexNumber);
     if((i & 1) == 1)
      cb_Output0.Checked = true;
     else
      cb_Output0.Checked = false;
     if((i & 2) == 2)
      cb_Output1.Checked = true;
     else
      cb_Output1.Checked = false;
     }
     if(rcv[0] == 'i')
     {
     int i = Int32.Parse(rcv.Substring(6), System.Globalization.NumberStyles.HexNumber);
     if((i & 1) == 1)
      cb_Input0.Checked = true;
     else
      cb_Input0.Checked = false;
     if((i & 2) == 2)
      cb_Input1.Checked = true;
     else
      cb_Input1.Checked = false;
     }
     if(rcv[0] == 'c')
     {
     if(rcv[7] == '0')
      tb_Counter0.Text = rcv.Substring(9);
     if(rcv[7] == '1')
      tb_Counter1.Text = rcv.Substring(9);
     if(rcv[7] == ';')
      readAndClearCounter(rcv.Substring(8));
     }
     }
    }
```

7. Polling

Solicitud cíclica de determinados valores
Es de desear que el estado de un sólo componente se actualice por mí mismo y así la aplicación siempre presente el estado actual. Para ello se utiliza un temporizador en este programa, que envía consultas al Web-IO cíclicamente en un intervalo de tiempo fijado por el usuario.

El intervalo de tiempo puede fijarse en el campo IDC_POLLINGTEXT.

Naturalmente también se captura en el caso de que el usuario realice una entrada absurda, como p. ej. un valor negativo de tiempo.

```
private void tb_Intervall_TextChanged(object sender, System.EventArgs e)
    {
     try
     {
     if(Convert.ToInt32(tb_Intervall.Text) > 0)
     {
     intervall = Convert.ToInt32(tb_Intervall.Text);
     statusBar.Text = "New range: " + intervall.ToString() + " ms!";
     }
     else
     statusBar.Text = "Only positive Integer allowed!";
     }
     catch(Exception)
     {
     statusBar.Text = "Only positive Integer allowed!";
     }
    }
```

Para realizar ahora también la solicitud cíclica de los estados del Web-IO, lo que se llama también Polling, existe la elección entre el Polling de los Outputs, de los Inputs o de los contadores.

Se inicializa un temporizador propio por variante de Polling.

Si se activa la caja de chequeo cb_Polling_Outputs se aplica el Polling a los Outputs. Para ello se inicializa el temporizador correspondiente.

```
private void cb_Polling_Outputs_CheckedChanged(object sender, System.EventArgs e)
    {
     if(cb_Polling_Outputs.Checked)
     {
     outputs = new System.Windows.Forms.Timer();
     outputs.Interval = intervall;
     outputs.Start();
     outputs.Tick += new EventHandler(timer_handler);
     }
     else
     outputs.Stop();
    }
```

Lo mismo ocurre con las entradas y los contadores.

```
private void cb_Polling_Inputs_CheckedChanged(object sender, System.EventArgs e)
    {
     if(cb_Polling_Inputs.Checked)
     {
     inputs = new System.Windows.Forms.Timer();
     inputs.Interval = intervall;
     inputs.Start();
     inputs.Tick += new EventHandler(timer_handler);
     }
     else
     inputs.Stop();
    }
private void cb_Polling_Counter_CheckedChanged(object sender, System.EventArgs e)
    {
     if(cb_Polling_Counter.Checked)
     {
     counter = new System.Windows.Forms.Timer();
     counter.Interval = intervall;
     counter.Start();
     counter.Tick += new EventHandler(timer_handler);
     }
     else
     counter.Stop();
    }
```

En este método se captura el evento respectivo del temporizador, que se está anunciando, y se le asigna una acción determinada.

```
 private void timer_handler(object sender, System.EventArgs e)
    {
     if(sender == counter) bt_Readall_Counter_Click(sender, e);
     if(sender == outputs) bt_Readall_Outputs_Click(sender, e);
     if(sender == inputs)  bt_Readall_Inputs_Click (sender, e);
    }
```

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.

Descargar el programa ejemplo

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

Protección de datos