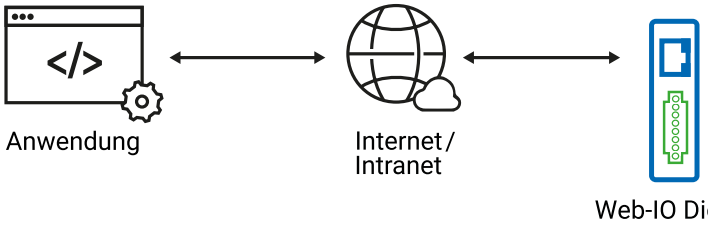
Applikation zum Web-IO Digital:

Web-IO Digital mit Delphi 2005 steuern und überwachen

Weitere Applikationen
Produktübersicht

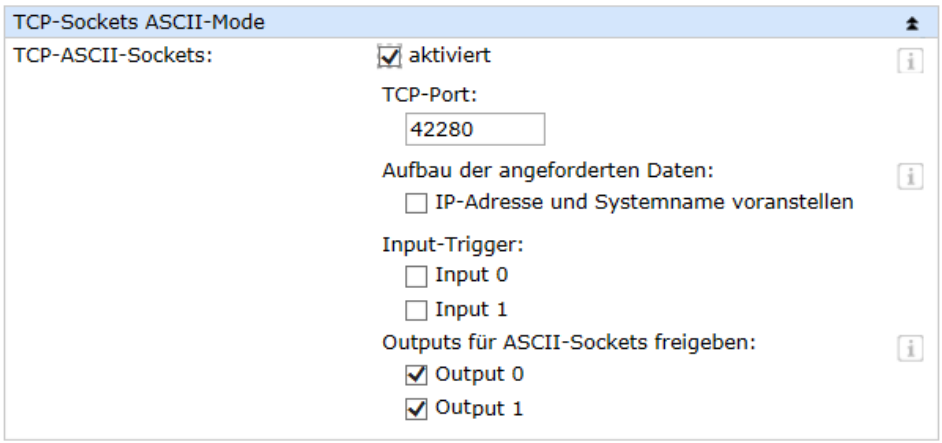
Als Nachfolger von Borland Delphi setzt Delphi 2005 auf das Microsoft.net Framework auf. Delphi 2005 (Delphi.net) bietet alles, was zum Programmieren von TCP/IP-Anwendungen nötig ist. Damit ist Delphi 2005 ein beliebtes Hilfsmittel, um Anwendungen zu erstellen, die mit dem Web-IO Digital kommunizieren. Zusätzliche Treiber oder DLLs werden nicht benötigt.



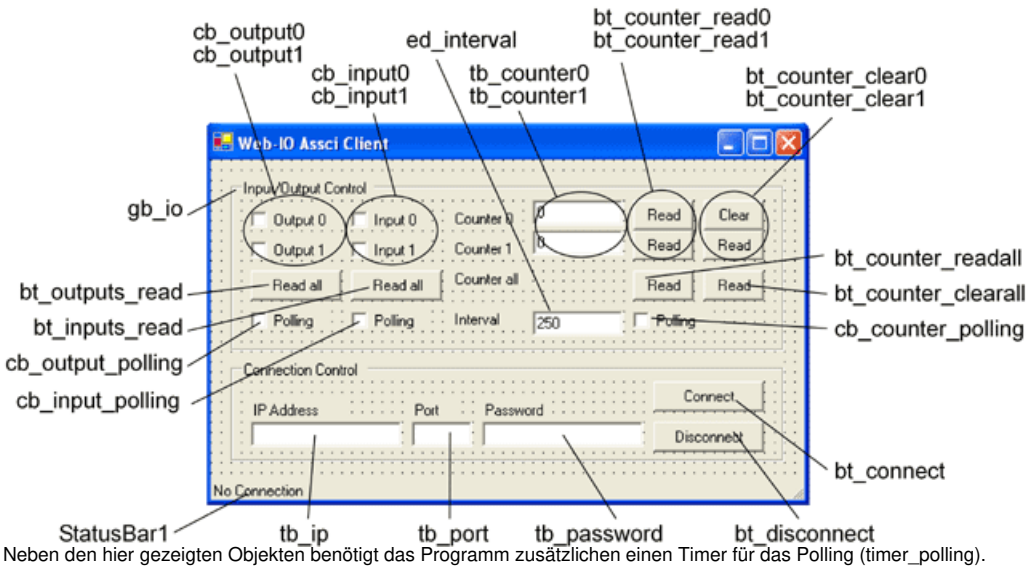
Mit dem folgenden Programmbeispiel können Sie Ihr Web-IO Digital mit seinen Inputs und Outputs in einer Windows-Anwendung abbilden. Darüber hinaus können Sie die Outputs des Web-IO schalten.

Vorbereitungen

- Web-IO mit Spannung versorgen und IOs verdrahten
- Web-IO mit dem Netzwerk verbinden
- IP-Adressen vergeben
- Beim Web-IO im Bereich *Kommunikationswege >> Socket-API* die *TCP-ASCII-Sockets* aktivieren und die *Outputs zum Schalten freigeben*



Zusammenstellen der verschiedenen Bedienelemente und Anzeigeobjekte in VB.net-Form



Neben den hier gezeigten Objekten benötigt das Programm zusätzlichen einen Timer für das Polling (timer_polling).

Bei der Benennung der einzelnen Objekte ist es hilfreich, sinngebende Namen zu verwenden. In diesem Beipiel beschreibt der erste Teil des Namens die Art des Objektes und der zweite Teil die Funktion.

Programmstart

Während der Aufbau grafischer Benutzeroberfächen in Delphi 2005 genauso einfach zu bewerkstelligen ist, wie mit den älteren Delphi Versionen, gestaltet sich die Netzwerkkommunikation unter Delphi.net etwas schwieriger. Das liegt in erster Linie daran, dass für Framework-Anwendungen das ClientSocket-Steuerelement nicht mehr zur Verfügung steht. Stattdessen muss für den Netzwerkzugriff im uses-Bereich der Namespace für die verwendeten Socket-Klassen angegeben werden.

```
uses
  System.Drawing, System.Collections, System.ComponentModel,
  System.Windows.Forms, System.Data, System.Net,
  System.Net.Sockets, System.IO, System.Text, System.Configuration;
```

Eigene Prozeduren, die nicht von Delphi selbst angelegt werden, müssen im Kopf des Quelltextes deklariert werden (Zum Inhalt dieser Prozeduren später mehr).

```
private
  { Private-Deklarationen }
public
  constructor Create;
  procedure callback_connect(ari : IAsyncResult);
  procedure callback_readdata(ari : IAsyncResult);
  procedure sendcommand(sendstring : String);
  procedure close();
end;
```

Darüber hinaus muss der Socket, auf dem die Kommunikation abgewickelt werden soll, angelegt werden und es muss ein Buffer für die Eingangsdaten sowie eine AsyncResult-Variable definiert werden.

```
var
  TCP_Client : Socket;
  ReceiveBuffer : array [0..512] of byte;
  ar : IAsyncResult;
```

Die Verbindungskontrolle

Einleiten der Verbindung
Durch Eingabe der IP-Adresse des Web-IO in das Textfeld *tb_ip* und Klick auf den Button *bt_connect* wird der Verbindungsaufbau gestartet.

```
procedure TWinForm.bt_connect_Click(sender: System.Object; e: System.EventArgs);
var
  WebioEP : IPEndPoint;
begin
  if (tb_ip.Text <> '') and (tb_port.Text <> '') then
  begin
    bt_connect.Enabled := false;
    TCP_Client := Socket.Create (AddressFamily.InterNetwork, SocketType.Stream,ProtocolType.TCP);
    StatusBar1.Text := 'Try to connect to ' + tb_ip.Text;
    timer_polling.Enabled := true;
    try
      TCP_Client.BeginConnect(WebioEP,callback_connect, TCP_Client);
    except
      on ex : Exception
      do
      begin
        StatusBar1.Text := 'ERROR on connecting';
        close();
      end;
    end;
  end;
end;
```

Verbindungsaufbau
Für die Abwicklung des TCP/IP-Handlings wird zunächst ein IPEndPoint aus IP-Adresse und TCP-Port definiert und damit der TCP_client Socket initialisiert. Im Zuge der Verbindungsanforderung wird ein Verweis auf eine Callback-Prozedur geschaffen.

Verbindung kommt zustande
Sobald das Web-IO die Verbindung annimmt, wird die Callback-Prozedur ausgeführt. In der Statuszeile wird das Zustandekommen der Verbindung angezeigt, die Bedienelemente werden zur Benutzung freigegeben und der Disconnect-Button wird bedienbar. Der Connect-Button wird für die Bedienung gesperrt. Darüber hinaus wird ein Verweis auf eine Callback-Routine für den Datenempfang erstellt.

```
procedure TWinForm.callback_connect(ari : IAsyncResult);
begin
  gb_io.enabled := true;
  bt_disconnect.Enabled := True;
  statusbar1.Text := 'Connected to' + tb_ip.Text;
  try
    TCP_Client.EndConnect(ari);
    TCP_Client.BeginReceive(ReceiveBuffer, 0, 512, SocketFlags.None, callback_readdata, ari.AsyncState);
  except
    on ex : Exception
    do
    begin
      StatusBar1.Text := 'ERROR on connect: ' + ex.ToString;
      close();
    end;
  end;
end;
```

Trennen der Verbindung

Die Verbindung bleibt solange bestehen, bis sie vom Benutzer durch Klick auf den Disconnect-Button beendet wird oder das Web-IO die Verbindung beendet.

In diesem Fall wird eine entsprechende Prozedur aufgerufen.

```
procedure TWinForm.close();
begin
  timer_polling.Enabled := False;
  Try
    TCP_client.EndReceive(ar);
  except
  end;
  Try
    TCP_client.Shutdown(SocketShutdown.Both);
  except
  end;
  Try
    TCP_client.Close()
  except
  end;
  bt_connect.Enabled := True;
  gb_io.Enabled := False;
  bt_disconnect.Enabled := False;
  StatusBar1.Text := 'no connection';
end;

procedure TWinForm.bt_disconnect_Click(sender: System.Object; e: System.EventArgs);
begin
  close();
end;
```

Verbindungsfehler
Alle, die TCP/IP-Kommunikation betreffenden, Aktionen werden innerhalb der Try-Anweisung ausgeführt. Treten Fehler auf, wird ebenfalls die CloseConnection-Prozedur aufgerufen.

Bedienung und Kommunikation von Client-Seite

Sobald eine Verbindung mit dem Web-IO zustande gekommen ist, kann der Anwender durch Bedienung der entsprechenden Programmelemente Kommandos an das Web-IO senden.

```
procedure TWinForm.sendcommand(sendstring: String);
var
  SendBuffer : array[0..512] of byte;
begin
  SendBuffer := System.Text.Encoding.ASCII.GetBytes(sendstring);
  try
    TCP_Client.Send(SendBuffer, 0, sendstring.Length, SocketFlags.None );
  except
    on ex : Exception
    do
    begin
      StatusBar1.Text := 'ERROR on send: ' + ex.ToString;
      close();
    end;
  end;
end;
```

Setzen der Outputs
Das Setzen der Outputs wird dem Anwender über zwei Checkboxen *cb_outputx* ermöglicht. Das Programm nutzt dazu das MouseUP-Ereignis dieses Objektes. Wird ein MouseUp, also ein Loslassen der Output-Checkbox registriert, führt das Programm die entsprechende Prozedur aus und gibt - je nachdem, ob die Checkbox gesetzt ist oder nicht - das passende Kommando an das Web-IO weiter.

```
procedure TWinForm.cb_output0_MouseUp(sender: System.Object; e: System.Windows.Forms.MouseEventArgs);
begin
  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;

procedure TWinForm.cb_output1_MouseUp(sender: System.Object; e: System.Windows.Forms.MouseEventArgs);
begin
  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;
```

Output/Input-Status abfragen
Den Status der Outputs und Inputs kann der Anwender durch Anklicken des zugehörigen Buttons anfordern.

```
procedure TWinForm.bt_outputs_read_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /output?PW='+ tb_password.Text + '&');
end;

procedure TWinForm.bt_inputs_read_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /input?PW='+ tb_password.Text + '&');
end;
```

Counter abfragen/löschen
Auch die Zählerstände der Input-Counter lassen sich abfragen bzw. löschen.

```
procedure TWinForm.bt_counter_read0_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /counter0?PW='+ tb_password.Text + '&');
end;

procedure TWinForm.bt_counter_read1_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /counter0?PW='+ tb_password.Text + '&');
end;

procedure TWinForm.bt_counter_clear0_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /counterclear0?PW='+ tb_password.Text + '&');
end;

procedure TWinForm.bt_counter_clear1_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /counterclear1?PW='+ tb_password.Text + '&');
end;
```

Natürlich lassen sich auch alle Counter zeitgleich lesen bzw. löschen.

```
procedure TWinForm.bt_counter_readall_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /counter?PW='+ tb_password.Text + '&');
end;

procedure TWinForm.bt_counter_clearall_Click(sender: System.Object; e: System.EventArgs);
begin
  sendcommand('GET /counterclear?PW='+ tb_password.Text + '&');
end;
```

Datenempfang vom Web-IO

Auswerten und Anzeigen der empfangenen Daten
Alle Kommandos und Anfragen an das Web-IO werden mit einem Antwort-String quittiert. Dabei haben die Antworten je nach Type einen spezifischen Aufbau:

- Für die Outputs: output;<Binärwert des Outputstatus im hexadezimalen Format>
- Für die Inputs: input;<Binärwert des Outputstatus im hexadezimalen Format>
- Für die Counter: counterx;<dezimaler Zählerstand>
- oder counter;<dezimaler Zählerstand 0 >; <dezimaler Zählerstand 0 >; ..... wenn alle Counter auf einmal gelesen werden sollen.
- Alle Antwort-Strings sind mit einem 0-Byte abgeschlossen.

Bei Datenempfang wird die entsprechende Callback-Prozedur aufgerufen

```
procedure TWinForm.callback_readdata(ari : IAsyncResult);
var
  bytesread : Integer;
  receivestring : string;
begin
  try
    bytesread := TCP_Client.EndReceive(ari)
  except
  end;
    if Bytesread = 0 then
      close()
    else
    begin
      receivestring := System.Text.Encoding.ASCII.GetString(ReceiveBuffer,0,bytesread);
      try
        TCP_Client.BeginReceive(ReceiveBuffer,0, 512,SocketFlags.None, callback_readdata,ari.AsyncState);
      except
        on ex : Exception
        do
        begin
          statusbar1.Text :='ERROR on read: ' + ex.ToString;
          close();
        end;
    end;
    if receivestring[1] = 'o' then
    begin
      if(convert.ToInt16(receivestring[8]) and 1) = 1 then
        cb_output0.Checked := true
      else
        cb_output0.Checked := false;
      if(convert.ToInt16(receivestring[8]) and 2) = 2 then
        cb_output1.Checked := true
      else
        cb_output1.Checked := false;
    end;
    if receivestring[1] = 'i' then
    begin
      if(convert.ToInt16(receivestring[7]) and 1) = 1 then
        cb_input0.Checked := true
      else
        cb_input0.Checked := false;
      if(convert.ToInt16(receivestring[7]) and 2) = 2 then
        cb_input1.Checked := true
      else
        cb_input1.Checked := false;
    end;
    if receivestring[1] = 'c' then
    begin
      if copy(ReceiveString, 8, 1) = '0' then
        tb_counter0.Text := copy(ReceiveString, 10,length(ReceiveString)-10);
      if copy(ReceiveString, 8, 1) = '1' then
        tb_counter1.Text := copy(ReceiveString, 10,length(ReceiveString)-10);
      if copy(ReceiveString, 8, 1) = ';' then
      begin
        ReceiveString[8] := ' ';
        tb_counter0.Text := copy(ReceiveString, 9, pos(';',ReceiveString)-9);
        tb_counter1.Text := copy(ReceiveString,pos(';',ReceiveString)+1,length(ReceiveString)-pos(';',ReceiveString)-1);
      end;
    end;
  end;
end;
```

Die Empfangsprozedur überprüft anhand des ersten Zeichens der Empfangsdaten, ob es um Input-, Output- oder Counter-Meldungen geht. Abhängig davon wird z.B. festgestellt, welcher Output welchen Status hat. Bei den Countern ist es sowohl möglich, einzelne Zählerwerte abzufragen, als auch alle Counter in einem Zug auszulesen. Die einzelnen Zählerstände werden dann dezimal mit Semikolon getrennt in einem String ausgegeben.

Polling

Zyklisches Abfragen bestimmter Werte
Um auch eine automatische Aktualisierung der Anzeige zu ermöglichen, wird ein Timer benutzt.

In Abhängigkeit der Checkboxen für Output-, Input- und Counter-Polling werden die entsprechenden Informationen im eingestellten Intervall vom Web-IO abgerufen.

```
procedure TWinForm.timer_polling_Tick1(sender: System.Object; e: System.EventArgs);
begin
  if TCP_Client.Connected then
  begin
    if cb_output_polling.Checked then
      sendcommand('GET/output?PW=' + tb_password.Text + '&');
    if cb_input_polling.Checked then
      sendcommand('GET/input?PW=' + tb_password.Text + '&');
    if cb_counter_polling.Checked then
      sendcommand('GET/counter?PW=' + tb_password.Text + '&');
  end;
end;
```

Das gewünschte Intervall kann in das entsprechende Textfeld eingegeben werden. Bei Änderung wird das Timer-Intervall dann automatisch angepasst.

```
procedure TWinForm.tb_interval_TextChanged(sender: System.Object; e: System.EventArgs);
begin
  timer_polling.Interval := convert.ToInt16(tb_interval.Text);
end;
```

Das Beispiel-Programm unterstützt alle gängigen Funktionen des Web-IO im Kommando-String-Modus, optimiert für das Web-IO 2x Digital Input, 2x Digital Output. Für die anderen Web-IO Modelle müssen ggf. Anpassungen am Programm vorgenommen werden. Weitere Programmbeispiele zur Socket-Programmierung finden Sie auf den Tool-Seiten zum Web-IO. Eine detaillierte Beschreibung zur Socket-Schnittstelle der Web-IO Digital Modelle finden Sie im Referenzhandbuch.

Programmbeispiel herunterladen

Produkte

Web-IO 4.0 Digital
2xIn, 2xOut

Bei Bedarf auch über PoE zu versorgen

Web-IO 4.0 Digital
12xIn, 12xOut

12x Eingänge,
12x Ausgänge

Weitere Web-IOs

Alle W&T Web-IO Digital 24V

Wir sind gerne persönlich für Sie da:

Wiesemann & Theis
GmbH
Porschestr. 12
42279 Wuppertal
Tel.: 0202/2680-110 (Mo-Fr. 8-17 Uhr)
Fax: 0202/2680-265
info@wut.de