Tutorial

How to use h/p/cosmos coscom v3 .NET Controls
with Visual Studio 2008

Programmed for:
h/p/cosmos sports & medical gmbh
Am Sportplatz 8
DE 83365 Nussdorf-Traunstein
Germany
phone + 49 86 69 86 42 0
fax + 49 86 69 86 42 49
email@h-p-cosmos.com
www.h-p-cosmos.com

Author and programming:
M. Sc. Andreas Feil, Altotec GmbH, Altofing 7, 83367 Petting / Germany

Date: 2009-09-11

As a contribution to h/p/cosmos' efforts for development and updating the coscom protocol and the coscom.dll, all users of the coscom protocol and coscom features are obliged to list

the name and company logo h/p/cosmos and the Copyright of h/p/cosmos in their software menu and their user/operation manual on a well visible position.

Content

1. Introduction

The h/p/cosmos coscom interface protocol has it's origin in the year 1992 and has been developed for safe, reliable and advanced communication, control and links between different ergometers like running machines, treadmills, bicycle ergometers, ladder ergometers etc., as well as control and monitoring equipment like PC, EMG, ECG, EKG, ergospirometry, VO2max systems, metabolic carts, cardiopulmonary stress test systems, biomechanics and motion analysis systems, fitness and sports medical as well as lactate evaluation and analyzing software, etc.

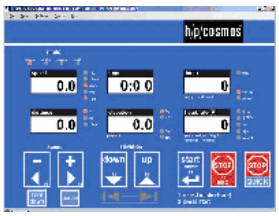
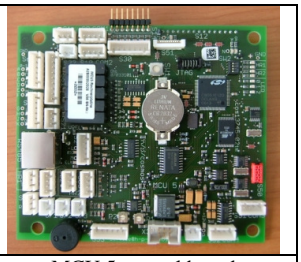
For easier implementation of h/p/cosmos coscom interface protocol functions, the coscom v3 .NET Objects and coscom v3 .NET Controls have been developed.

This document gives a short overview on how to use the coscom v3 .NET Objects within Microsoft's Visual Studio 2008.

It shows how to create a windows forms project where you easily can drag and drop the coscom v3 .NET Controls on the form.

2. Important Notes, Safety Precautions, Warnings

The coscom v3 .NET Controls can be used with h/p/cosmos running machines and OEM running machines which are equipped with MCU4 and MCU5 control board (UserTerminal).

			
h/p/cosmos mercury med	h/p/cosmos mercury lt med	h/p/cosmos para control® 4.0	MCU 5 control board

Do not use these coscom v3 .NET controls with a MCU4 device which has a firmware version between version MCU4 EPROM Firmware version 4.04.1 and 4.04.4. These versions contain implementation errors for coscom v3 features and must not be used with coscom v3. From MCU4 Firmware v 4.04.5.0025 the coscom v3 .NET controls can also be used, so please make sure the Firmware was updated this 4.04.5 or higher.

Pay attention to all safety instructions and warnings as well as the chapters "intended use" and "forbidden use" of the operation and service manual of the h/p/cosmos running machines and also the h/p/cosmos para control® 4.0 software!

Always activate "failsafe timeout mode", so the running machines stops automatically in case of termination of the interface communication.

Always communicate the "status" mode of the running machine with any host programs.

In case the running machine was stopped via STOP button on the running machine or via any other reason (for example power failure for the running machine supply), the host program must realize this “stop status” and must terminate automatically any control functions of speed any elevation control of the running machine. If not, a user may be caught by surprise and unexpected command from host program, although the user pressed STOP on the running machine before. This may lead to serious accident.

3. How To Create A Sample Application

3.1. Create The Project

First you have to create a windows forms application. Select *“File/New/Project…”* from the menu. Then you choose the kind of application you want to implement. Select a *c# Windows Forms Application* as shown in the following figure. Type in your application name and click the “OK” button.

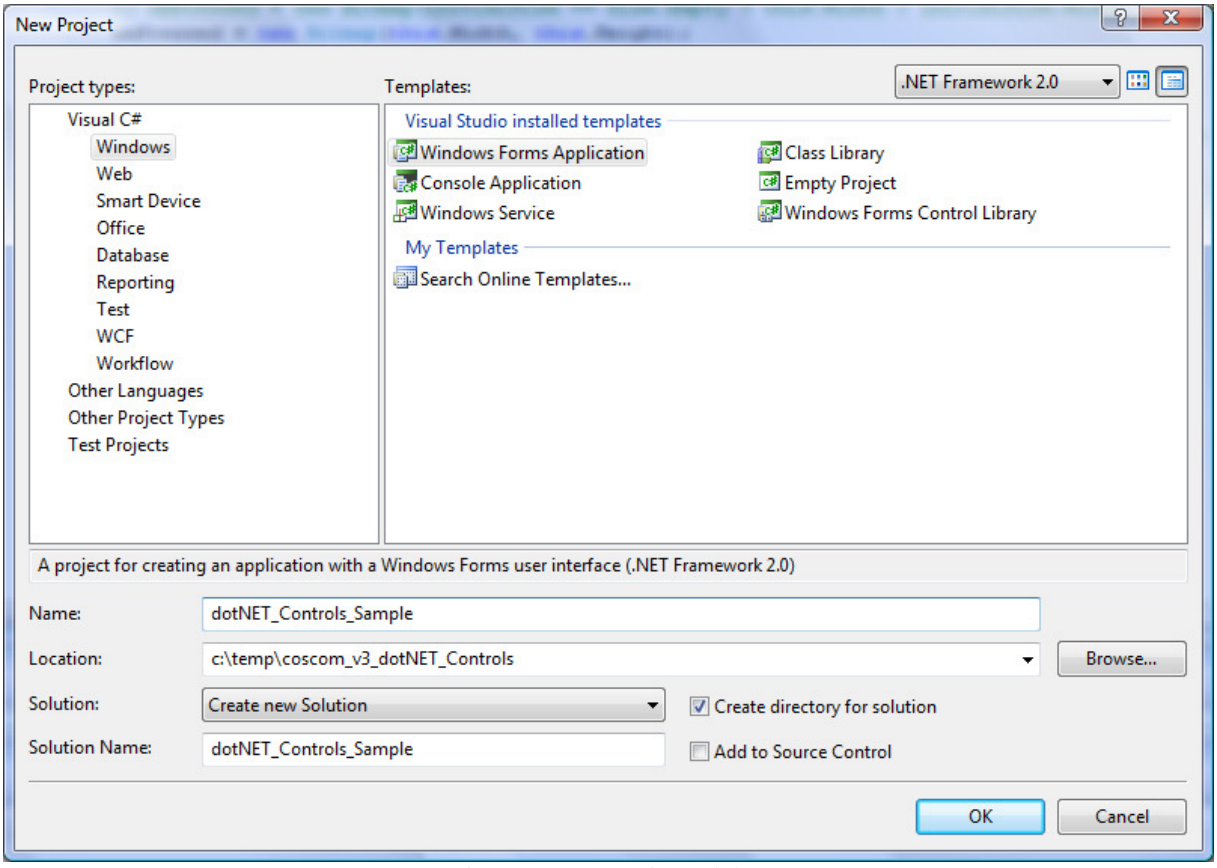


Figure 1: New Project Dialog

3.2. Choose Toolbox Items

In order to use the coscom v3 .NET Controls you have to add a reference to the toolbox. Right click on the toolbox and select “Choose Items” from the context menu as shown in the following figure.

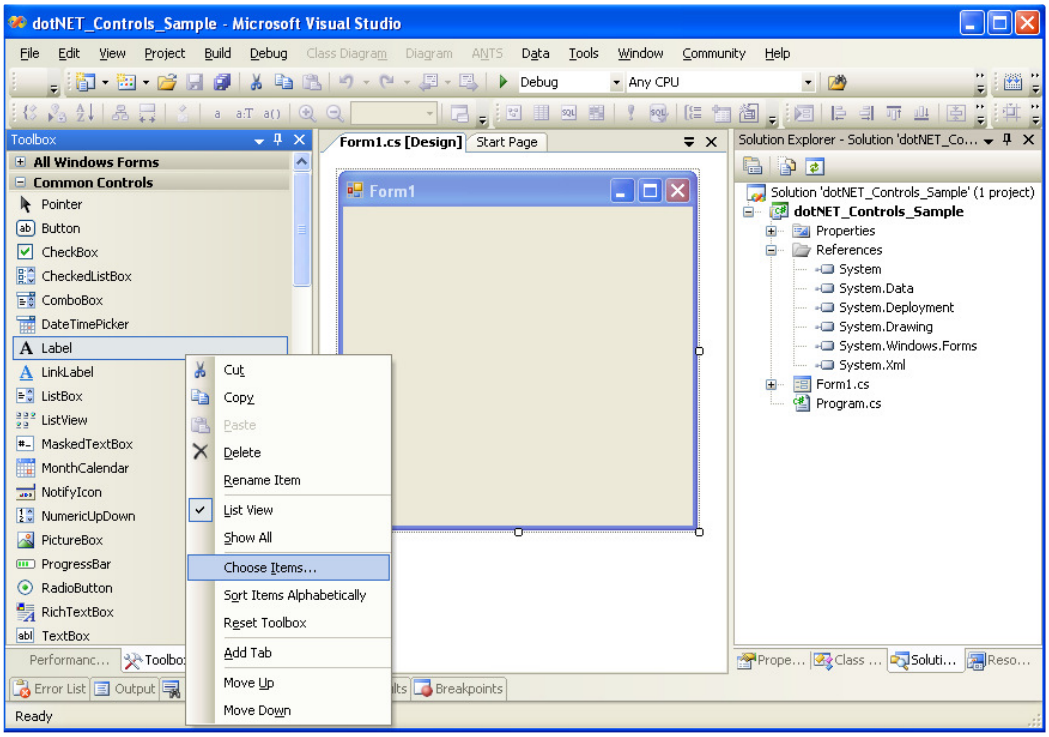


Figure 2: Choose Toolbox Items

In the “Choose Toolbox Items” dialog click on browse and select the “coscom NET Controls.dll” in the coscom dll directory and press “Open”. Confirm the dialog by clicking on the “OK” button.

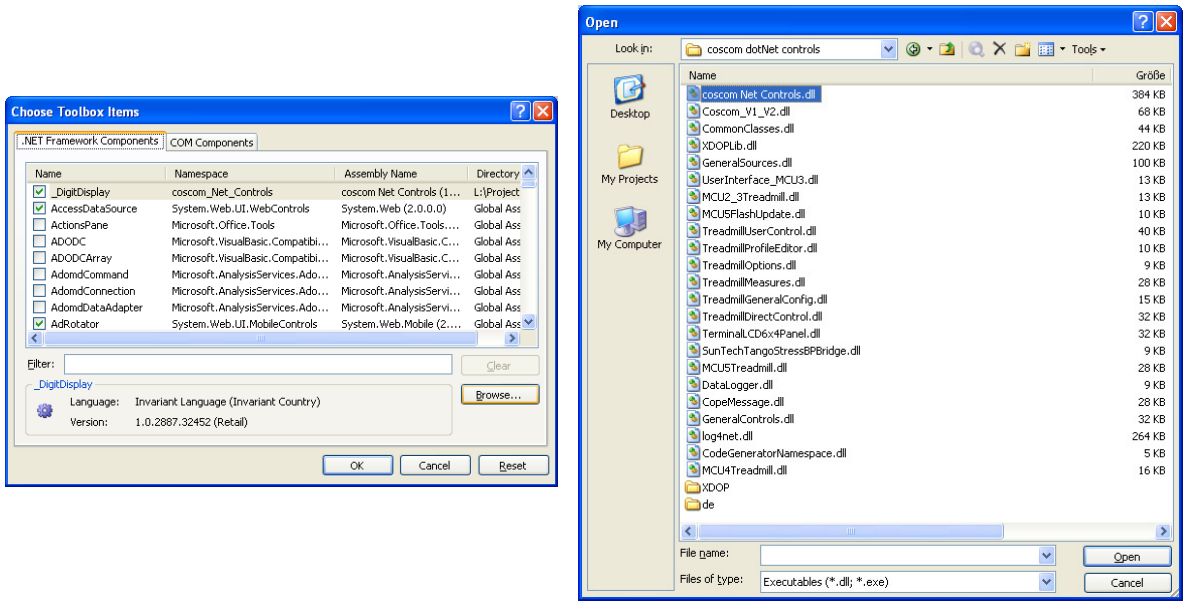


Figure 3: The “Choose Toolbox Items” dialogs

Now all coscom v3 .NET Controls appear in the toolbox as shown in figure 4.

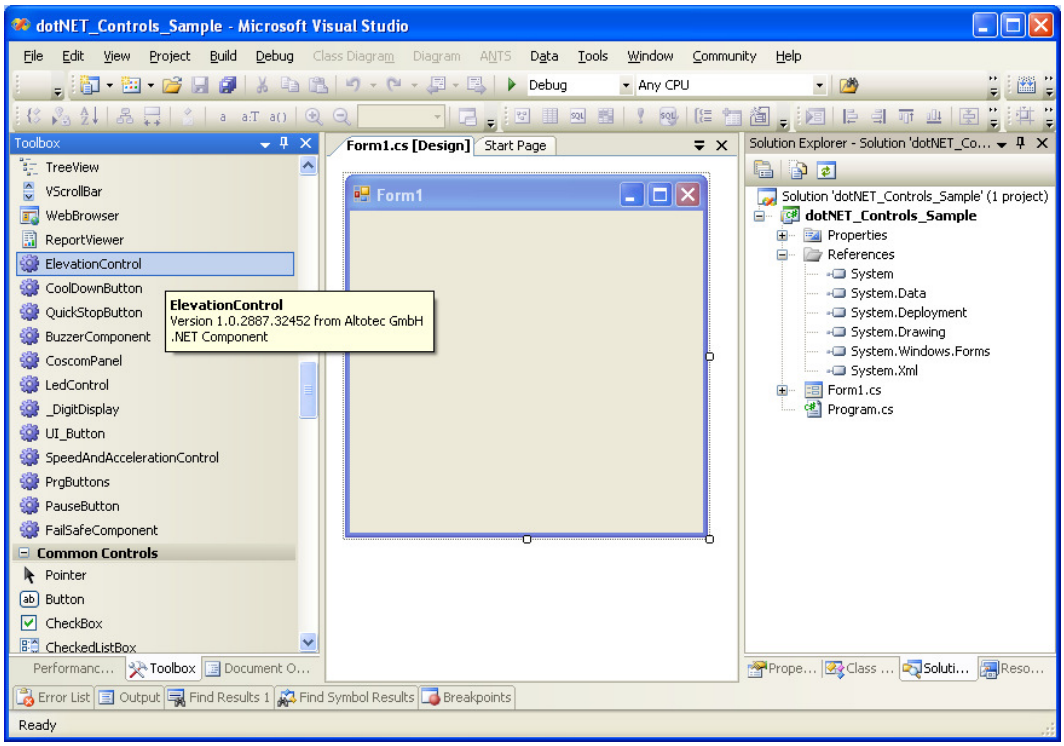


Figure 4: The coscom v3 .NET Controls in the toolbox

3.3. Add References

In order to use the controls you must add some references to your project. Right click on the references in the project explorer and select “Add Reference…” from the context menu.

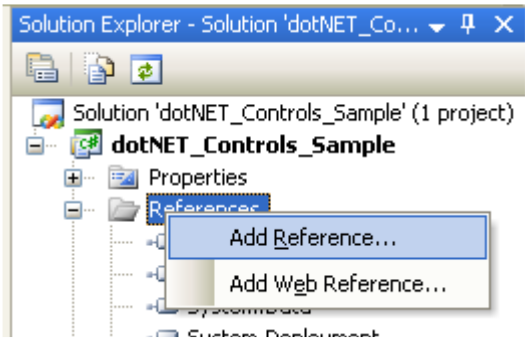


Figure 5: Add a reference to your project

Add the references to all dll files in the coscom dll directory. As shown in figure 6.

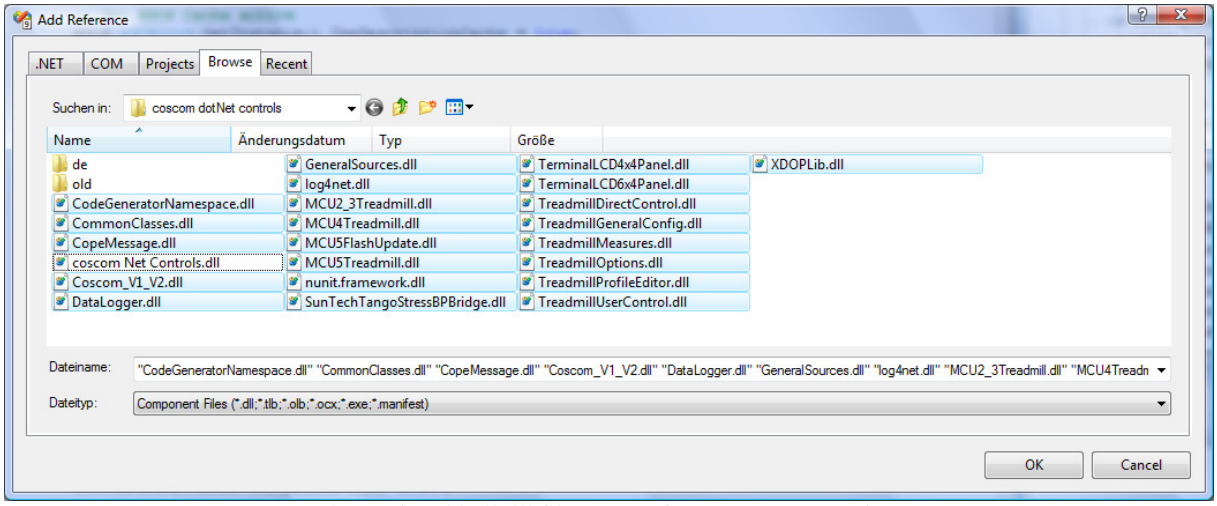


Figure 6: Add all dll files as a reference to your project

3.4. Drag And Drop Controls

Now you can drag and drop your coscom v3 .NET Controls onto your form.

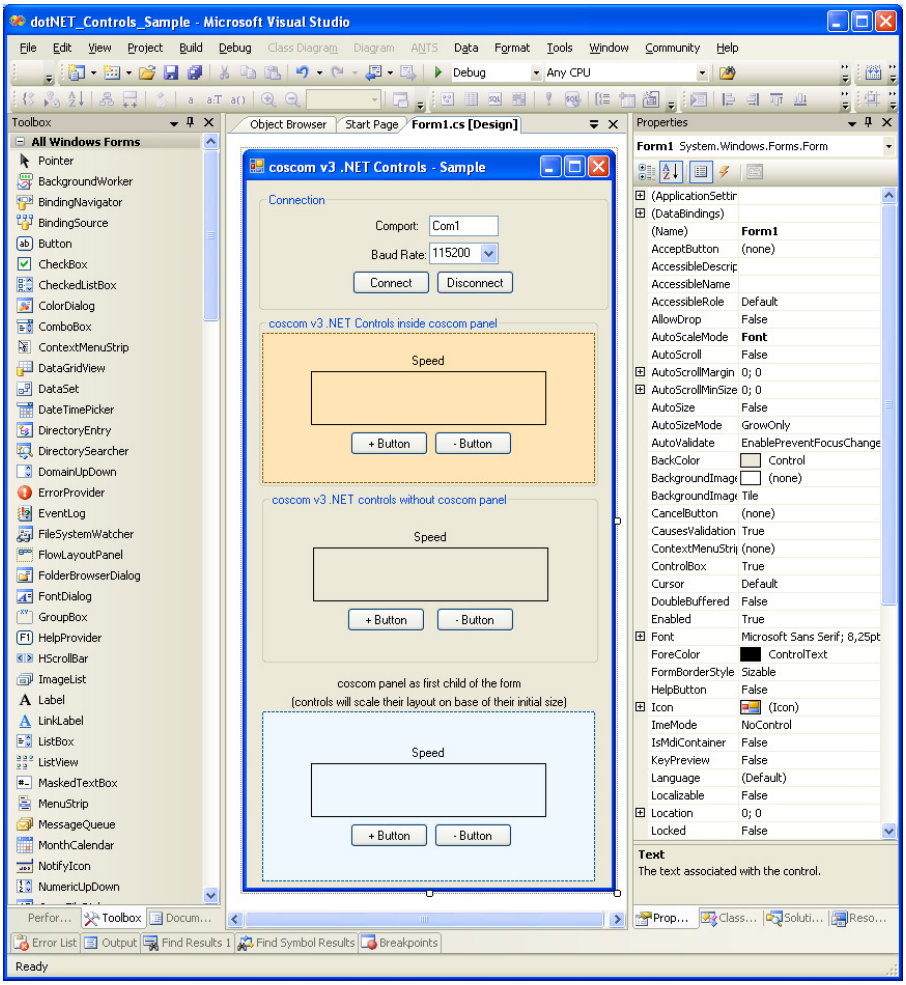


Figure 7: The sample application form

In this example one display and two buttons where added three times to the form.

First the controls where added within a coscom panel (orange) which itself was added to a group box control.

Second the coscom v3 .NET Controls where directly added to the form without a coscom panel.

And at last the controls where added to a coscom panel which is directly located on the form.

The buttons where configured to be the "+" and "–" buttons of the display terminal. Therefore the property "CommandChar" was set to "+" and "–" as shown in figure 8.

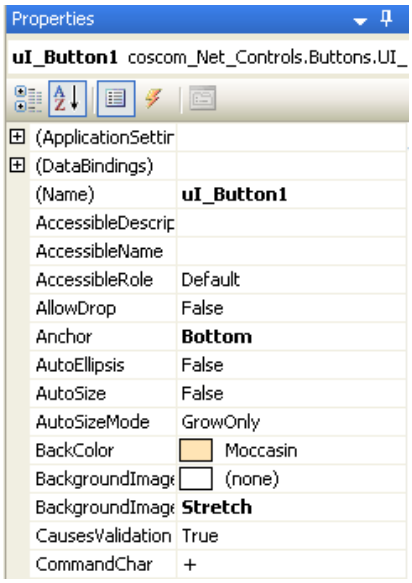


Figure 8: The properties of a ui_button

The display is configured to represent the speed display of the terminal. Therefore the property "DisplayString" was set to "UpperLeft".

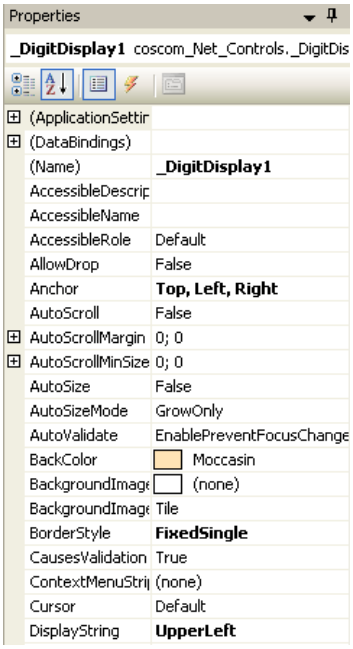


Figure 9: The properties of a _DigitDisplay

3.5. Get Device Connection

Now you need to get a device connection in order to use the controls. Therefore an event handler was added to the click event of the connect button. The code is shown in figure 10.

```
25          private void buttonConnect_Click(object sender, EventArgs e)
26          {
27              // Search the device in a new thread
28              Thread t = new Thread(new ParameterizedThreadStart(Connect));
29              t.Start(new object[] { textBoxPort.Text,int.Parse(comboBoxBaudrate.Text) });
30
31          }
32
33          IDMCU5Treadmill mcu5device = new MCU5Treadmill();
34          private void Connect(object args)
35          {
36              try
37              {
38                  // Set XDOP Cache active
39                  XDOP.XDOPCore.GetInstance().UseDescriptionCache = true;
40                  // Set XDOP Cache to another location than default path
41                  XDOP.XDOPCore.GetInstance().DescriptionCachePath = Path.Combine(Application.StartupPath, "XDOP");
42                  // Create a new protocol parameter instance
43                  XDOP.ProtocolParameter ppara = new XDOP.ProtocolParameter(true, true, 4000, 3000, 18000, 0, 0);
44                  // Create a new serial connection parameter for the specified com port and baud rate
45                  XDOP.ConnectionParameterCOM comparameter =
46                      new XDOP.ConnectionParameterCOM((string)(((object[])args)[0]), (int)(((object[])args)[1]));
47                  // Get a transport object by using the connection parameter
48                  XDOP.ITransportObject itransport = XDOP.TransportObjectCOM.getIXDOPComs(comparameter)[0];
49
50                  // connect to the device
51                  mcu5device.Connect(itransport, ppara);
52
53                  // for a save communication you should enable the messgae checksum on device side by invoking the
54                  // SetProtocolParams action in the TreadmillGeneralConfig service
55
56                  // You also should use the failsafe component in your application as done in this small sample
57
58                  // Inform all coscom controls within the coscom panels that a new device connection was made.
59                  // The second bool parameter defines that coscom components such as the buzzer and failsafe control
60                  ///on the form will be set to.
61                  coscomPanel1.SetDevice(mcu5device, true);
62                  coscomPanel2.SetDevice(mcu5device, true);
63
64                  // Manually set the device for the controls that are not located in a coscom panel
65                  _DigitDisplaySpeed.setDevice(mcu5device);
66                  uI_ButtonMinus.setDevice(mcu5device);
67                  uI_ButtonPlus.setDevice(mcu5device);
68              }
69              catch (Exception ex)
70              {
71                  MessageBox.Show("Error: "+ex.Message);
72              }
73          }
```

Figure 10: The connection code snippet

You should try to make the connection in a separate thread. Connecting will take some time therefore it shouldn't be done in the GUI-Thread.

You can speed up connecting to a device with baud rate 9600 by using the XDOP description cache. This feature saves the necessary device descriptions to a local path and doesn't query them each time a connection is made. Therefore only the first connecting will query the descriptions, subsequent calls will use the local stored descriptions. This feature only helps with slow baud rates of 9600, higher baud rates of 115200 are fast enough that the cache wouldn't speed up anything.

Another feature worth mentioning is the XDOP.ProtocolParameter class. This class defines some basic protocol parameters. You can set the following parameters:

- *Send default indices:* If set default indices of 0 will be added in the protocol message. These indices are optional therefore you can disable this feature.
- *Use XDOP checksum:* You should set the value to true in order to use the message checksum on application side.
- *General timeout:* The general message timeout of the communication.
- *Root description timeout:* The timeout for the device root description. This timeout should be between 2 and 3 seconds (2000 – 3000).
- *Service description timeout:* The timeout for service description responses from a device. For long service descriptions on a low baud rate this should be a value higher than 10000 (10 seconds).
- *Count retry send:* Currently not used. Should be value 0.
- *Retry send interval*: Currently not used. Should be value 0.

3.6. Control behavior

At this point some differences in the control behavior will be explained. The simplest way to use the controls is to put them in a coscom panel (orange panel in sample application). This panel will handle the distribution of the device object to all containing coscom v3 .NET objects. You only have to call the "SetDevice" method of the panel (code line 51 in the above figure).

There is a difference between the layout of the controls in a coscom panel depending if the panel is located directly on the form or in a container object such as a group box or a normal panel. If it is located in a container all controls will perform their layout as usual. Otherwise if the coscom panel is located directly on the form (the blue coscom panel in the sample application) the controls within the panel will perform their layout in order to scale their location and size properly in dependence of the background image of the panel. Only in this case you must initialize the coscom panel with the "AfterDesignerInit" method when it has its initial size (e. g. in the constructor of the form class as shown in code line 21 in figure 11).

```
16          public Form1()
17          {
18              InitializeComponent();
19              // In case the coscom panel is a direct child of the containing form
20              // you have to call this method in order to proper scaling of the controls.
21              coscomPanel2.AfterDesignerInit();
22          }
```

Figure 11: Init the coscom panel if it is directly located on the form

The other way to use the controls is to use them directly without a coscom panel. Therefore you must assign each control with a device object manually (code line 55 – 57 in figure 11).

3.7. Disconnect

At the end of the program you have to close the device connection by using the "DisposeDevice" method of the device object as shown in the following figure.

```
83          private void Form1_FormClosing(object sender, FormClosingEventArgs e)
84          {
85              if (mcu5device != null)
86                  mcu5device.DisposeDevice();
87          }
```

Figure 12: Disconnect from the device