# Tutorial

How to use h/p/cosmos coscom v3 .NET Objects
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.

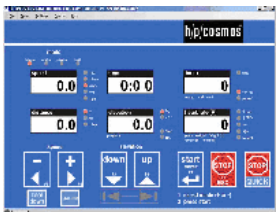
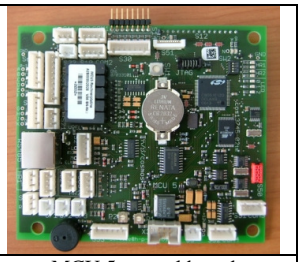
For easier implementation of h/p/cosmos coscom interface protocol functions, the coscom v3 .NET Objects and coscom v3 .NET Controls have been developed.

This document gives a short overview on how to use the coscom v3 .NET Objects within Microsoft's Visual Studio 2005.

It shows how to create a windows forms project where you easily can use the coscom v3 .NET Objects to control the running machine.

# 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.

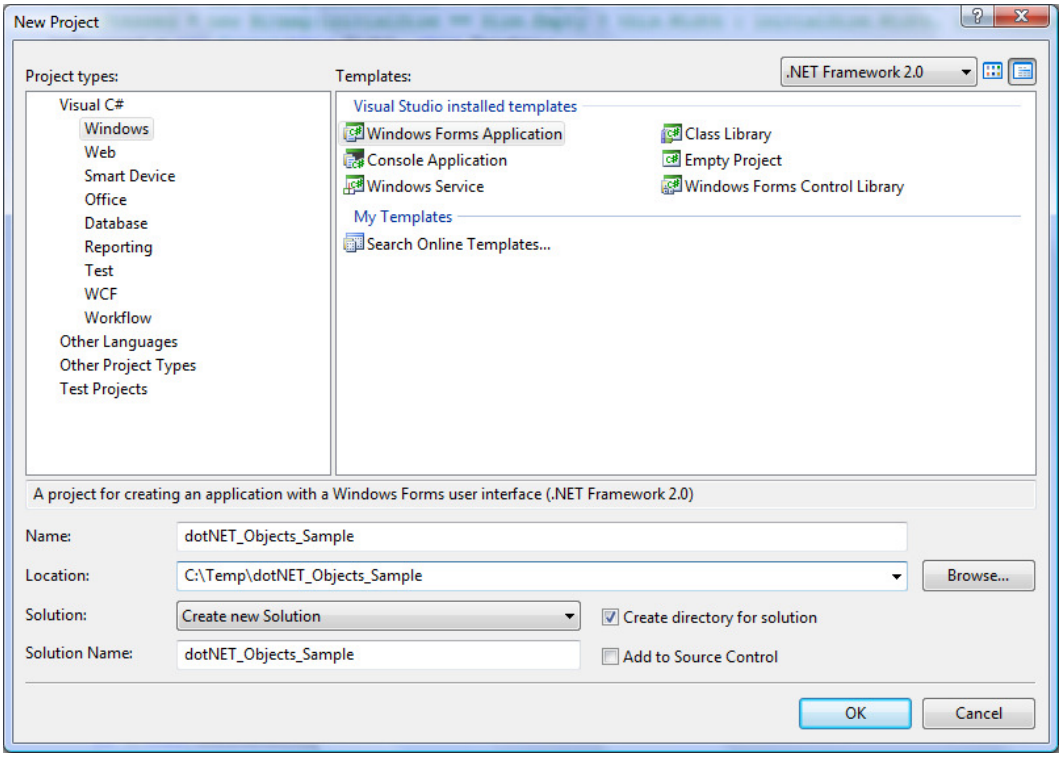



Figure 1: New Project Dialog

## *3.2. 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.

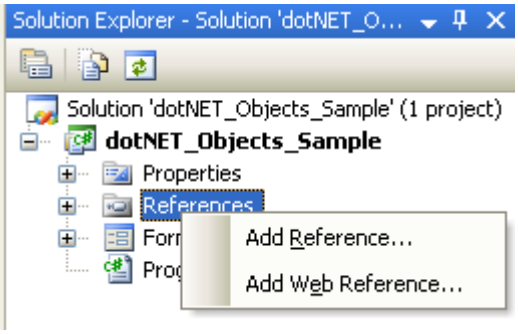



Figure 2: Add a reference to your project

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

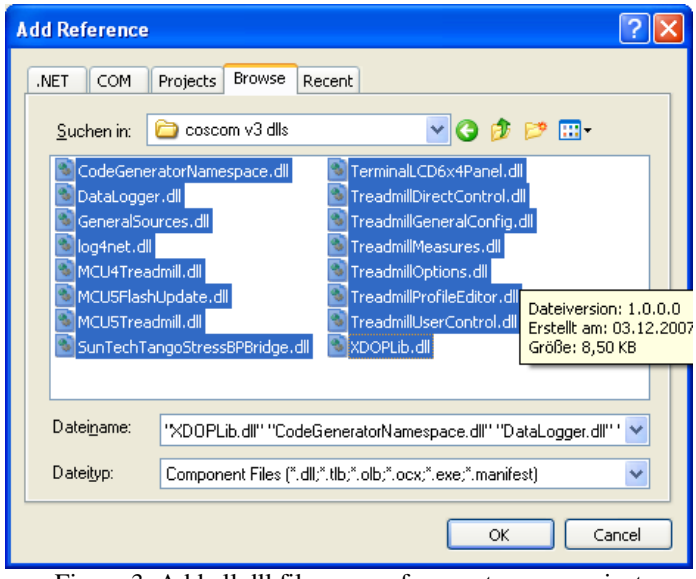



Figure 3: Add all dll files as a reference to your project

## 3.3. Drag And Drop Controls

Now you can drag and drop some .NET Controls onto your form. In this example you can set a new drive speed and grade through the controls on the form. Also the actual speed and grade are shown on the form.

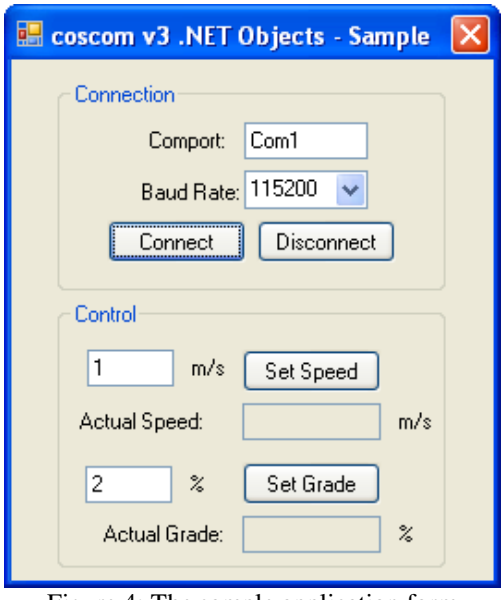



Figure 4: The sample application form

## 3.4. Get Device Connection

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

```
23          private void buttonConnect_Click(object sender, EventArgs e)
24          {
25              // Search the device in a new thread
26              Thread t = new Thread(new ParameterizedThreadStart(Connect));
27              t.Start(new object[] { textBoxPort.Text, int.Parse(comboBoxBaudrate.Text) });
28
29          }
30
31          IDMCU5Treadmill mcu5device = new MCU5Treadmill();
32          XDOP.schemas_coscom_org.service.ISTreadmillDirectControl IServiceDirectcontrol = null;
33          private void Connect(object args)
34          {
35              try
36              {
37                  // Set XDOP Cache active
38                  XDOP.XDOPCore.GetInstance().UseDescriptionCache = true;
39                  // Set XDOP Cache to another location than default path
40                  XDOP.XDOPCore.GetInstance().DescriptionCachePath = Path.Combine(Application.StartupPath, "XDOP");
41                  // Create a new protocol parameter instance
42                  XDOP.ProtocolParameter ppara = new XDOP.ProtocolParameter(true, true, 4000, 3000, 18000, 0, 0);
43                  // Create a new serial connection parameter for the specified com port and baud rate
44                  XDOP.ConnectionParameterCOM comparameter =
45                      new XDOP.ConnectionParameterCOM((string)(((object[])args)[0]), (int)(((object[])args)[1]));
46                  // Get a transport object by using the connection parameter
47                  XDOP.ITransportObject itransport = XDOP.TransportObjectCOM.getIXDOPComs(comparameter)[0];
48
49                  // connect to the device
50                  mcu5device.Connect(itransport, ppara);
51
52                  // iterate through all services and save a reference to the direct control service
53                  foreach (CodeGenerator.IServiceObject service in mcu5device.Services)
54                  {
55                      if (service.ServiceType.Equals(TreadmillDirectControl.CodeGeneratorServiceType))
56                          IServiceDirectcontrol = (ISTreadmillDirectControl)service;
57                  }
58
59                  // subscribe to changes of the actual speed and grade by adding the following event handlers
60                  IServiceDirectcontrol.ActualGradeChanged +=
61                      new OnChangeTreadmillDirectControlActualGrade(IServiceDirectcontrol_ActualGradeChanged);
62                  IServiceDirectcontrol.ActualSpeedChanged +=
63                      new OnChangeTreadmillDirectControlActualSpeed(IServiceDirectcontrol_ActualSpeedChanged);
64              }
65              catch (Exception ex)
66              {
67                  MessageBox.Show("Error: " + ex.Message);
68              }
69          }
```

Figure 5: 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.

## 3.5. Control The Running Machine

Now that you got a connection two event handlers were added to the set speed and set grade buttons. The necessary code for setting a new drive speed is shown in figure 6.

```
98          private void buttonSetSpeed_Click(object sender, EventArgs e)
99          {
100             if (IServiceDirectcontrol != null)
101             {
102                 try
103                 {
104                     // set a new drive speed
105                     IServiceDirectcontrol.SetDriveSpeed(float.Parse(textBoxSpeed.Text), 3, 0);
106                 }
107                 catch (Exception er)
108                 {
109                     MessageBox.Show("Error: " + er.Message);
110                 }
111             }
112         }
```

Figure 6: Set speed code snippet

The running machine is able to send you the new speed and grade values it they change. Therefore you need to register for this so called *evented variable* by adding an event handler to the "ActualSpeedChanged" event of the direct control service.

```
73          // Event handler for speed changes
74          void IServiceDirectcontrol_ActualSpeedChanged(float newValue)
75          {
76              // Because of control changes have to be an the gui thread a invoke must be called to set the control text
77              this.Invoke(new SetControlText(DelegateImplementation), new object[] { textBoxActualSpeed, newValue.ToString() });
78          }
```

Figure 7: Event handler code of the actual speed changed event

## 3.6. Close Device Connection

At the end of the remote controlling of the running device you have to close the communication by disposing the MCU device object. Simply call the "DisposeDevice" method like shown in figure 8.

```
135         private void Form1_FormClosing(object sender, FormClosingEventArgs e)
136         {
137             // Dispose the device
138             if (mcu5device != null)
139                 mcu5device.DisposeDevice();
140
141         }
```

Figure 8: Close device connection