

Work with Digi C# API

1. Start with Digi C# API

1.1. What is it?

The main function of this API is to make the application development easier to the user, this mean, hide the hardest part of develop application with the standard functions, hide handlers, configurations that are allways the same and so on.

1.2. Installation

Uncompress the zip file, and then you can find the API in the folder where you uncompress the zip file with the following contain:

- **Apps:** Demo application source code
- **Bin:** folder that contains the compiled dlls
- **Digi.Common:** source code of the dll that includes the common functions, constants... using in the drivers
- **Digi.Driver:** source code of the dll that includes the different Digi drivers (GPIO, I2C, SPI, ADC...)
- **Digi.Imported:** source code of the dll that includes the files imported from C (header files)
- **Digi.Zigbee:** soucre code of the dll that inludes ZigBee stuff.
- **Documentation:** Doxygen documentation of the API (html format), this document.

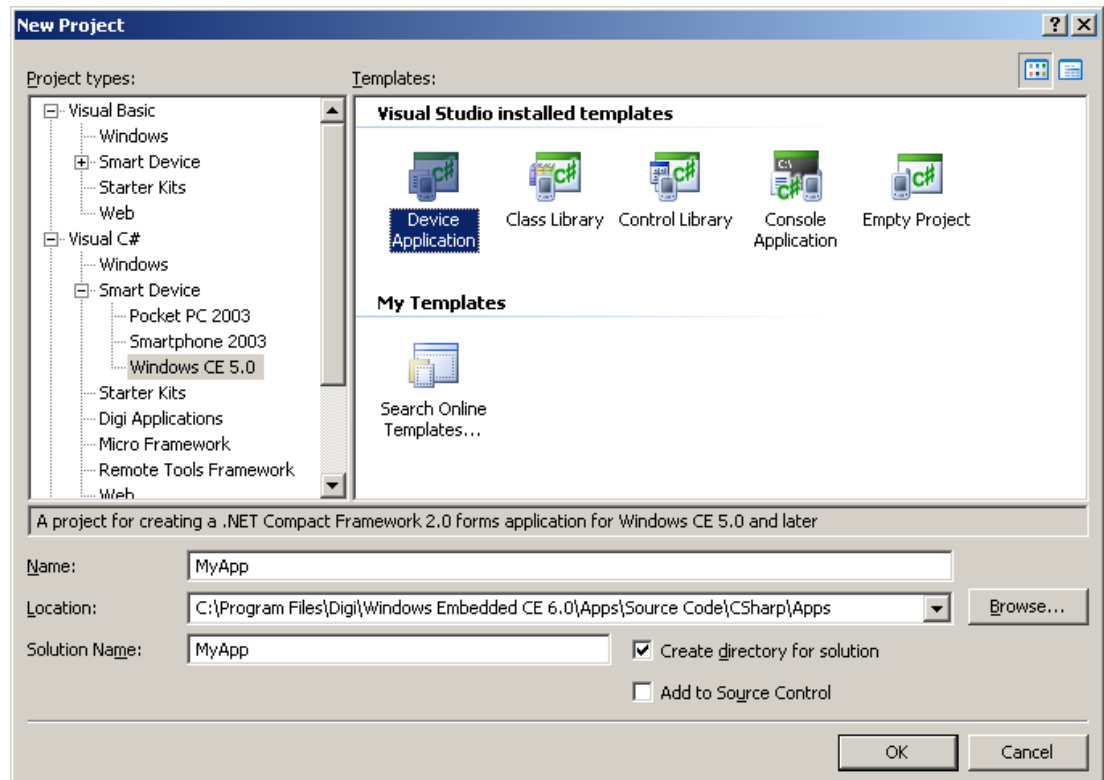
2. Develop an application with Visual Studio 2005 and Digi C#API

This task creates and builds a simple application using Visual C#® and the Digi C# API. In this example the user learns how to integrate the Digi C# API into his project with an application that manages the gpio driver.

2.1. Create the project

1. Select **File > New > Project**.

The **New Project** dialog box opens:



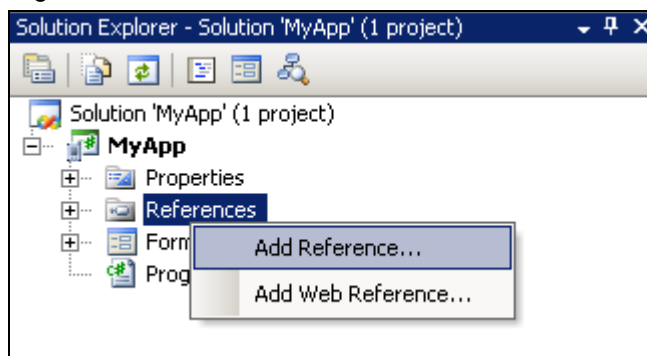
2. Do these steps:

- In the **Project Types** tree select **Visual C#** (if Visual C# is not your default language in Visual Studio, expand **Other Languages > Visual C#**) select **SmartDevice**, and **Windows CE 5.0** (This selection is also valid for Windows Embedded CE 6.0.)
- In the **Templates** section, select **Device Application**.
- In the bottom part of the dialog box, in the **Name** input box, enter name for project.
- In the **Location** input box, enter the location for the source files.

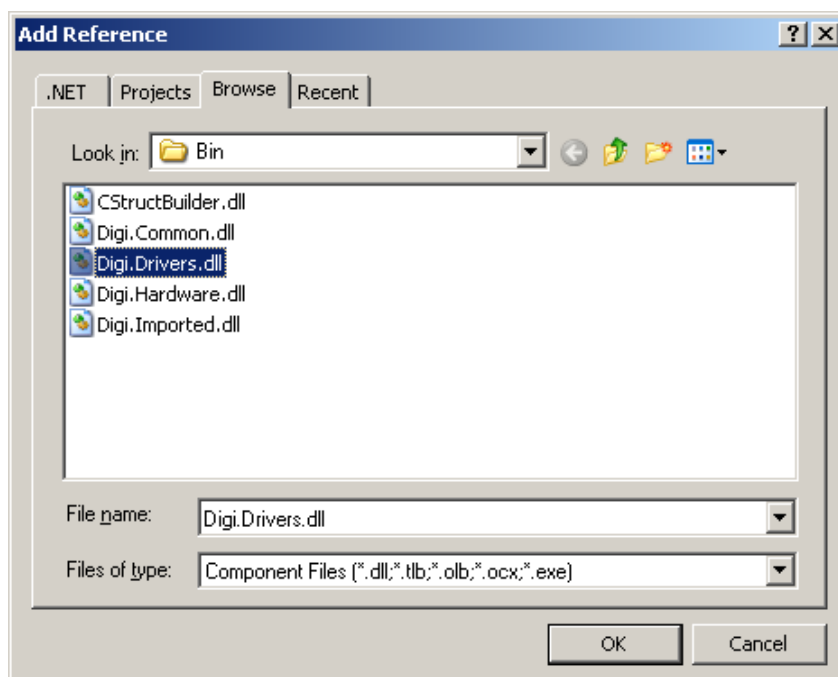
When everything is entered, click **OK**.

2.2. Add the C# API dlls

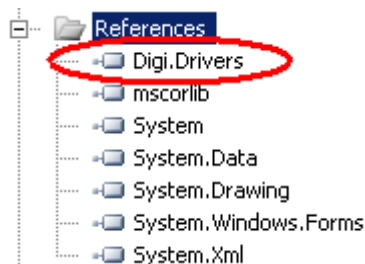
1. Select the Solution Explorer view, **View > Solution Explorer**.
2. Right-click on **References**, and select **Add Reference...**



The **Add Reference** dialog box opens:



3. In the Browse tab navigate through the browser to the Bin folder in the decompressed folder and select **Digi.Drivers.dll** and click **OK**.
4. See that the Digi.Drivers.dll is added in the Reference tree:



5. After add the reference of the Digi.Drivers.dll add the header to be able to use dll's classes, in the top of the main file (in the example case Form1.cs) add the following code:



```
using Digi.Drivers;
```

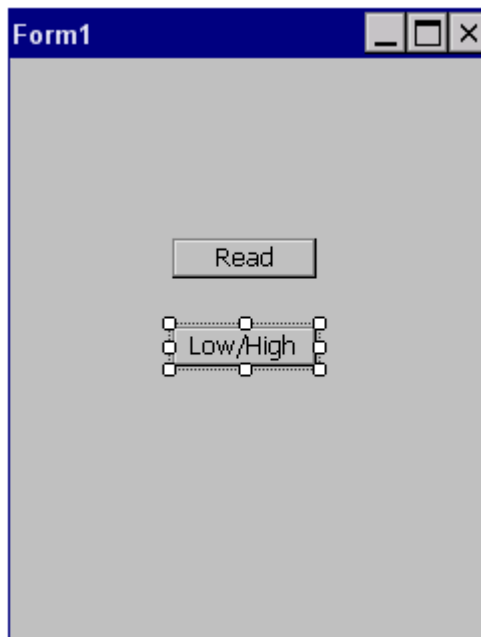
2.3. Develop the application

The application consists in a gpio configured as output (to change his value Low/High) and a button that shows through a message box the value of a gpio configured as input.

2.3.1. Generate the interface

Now add some information in the form to generate the interface:

1. To open the toolbox, select **View > Toolbox**.
2. Drag and drop two buttons into the empty form so the interface looks like this:



4. Right-click the first button and select **Properties**. Then, in the **Appearance > Text** field, enter **Read** as the new text. Right-click the second button and select **Properties**. Then, in the **Appearance > Text** field, enter **Low/High** as the new text.

2.3.2. Generate the source code

Next, add some code in the source code and buttons's click methods.

1. To open the first button's click method source code, double-click the first button on the form. Code similar to this is displayed:



```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Digi.Drivers;

namespace MyApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private void button1_Click(object sender, EventArgs e)
            {

            }

        }
    }
}
```

2. The same with the second button, double-click the second button on the form. Code similar to this is displayed:



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Digi.Drivers;

namespace MyApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private void button1_Click(object sender, EventArgs e)
            {

            }
            private void button2_Click(object sender, EventArgs e)
            {

            }

        }
    }
}
```

3. Now define the two gpio variables:
Configure one gpio variable as input.
Configure the other gpio as output and assign it the value "Low".



```
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Digi.Drivers;

namespace MyApp
{
    public partial class Form1 : Form
    {
        GPIO Input1 = new GPIOiMX51(GPIOiMX51.Port.Port1, 2,
                                     GPIO.DirectionValue.Input);
        GPIO Output1 = new GPIOiMX51(GPIOiMX51.Port.Port1, 3,
                                     GPIO.DirectionValue.Output,
                                     GPIO.IoValue.Low);

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
        }

        private void button2_Click(object sender, EventArgs e)
        {
        }
    }
}
```

4. Fill the methods, button1_Click correspond to button **Read**, we add the code to Read the Input1 value and show it.



```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Button1 value: " + Input1.Value.ToString());
}
```

5. The button2_Click correspond to button Low/High, add the code to change the output value of the Output1:

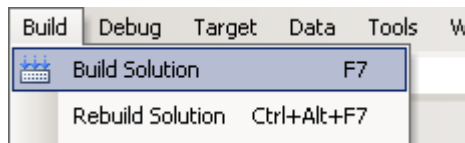


```
private void button2_Click(object sender, EventArgs e)
{
    if (Output1.Value == GPIO.IoValue.Low)
        Output1.Value = GPIO.IoValue.High;
    else
        Output1.Value = GPIO.IoValue.Low;
}
```

6. Save the file.

2.4. Build the sample application

To build the sample application, select **Build > Build Solution**:



When the build completes, messages in the output window (**View > Output**) indicate:

- That the build process was successful
- The location of the executable image

3. Transfer the application to the target

Copy the dlls and the exe file of the application to the target. There are located it in the folder bin/Release of the project.

To run the application correctly are necessary MyApp.exe, Digi.Common.dll, Digi.Drivers.dll and Digi.Imported.dll.



In the develop Digi.Drivers is added to the project, but this dll uses other dlls that are necessities to run the application correctly.

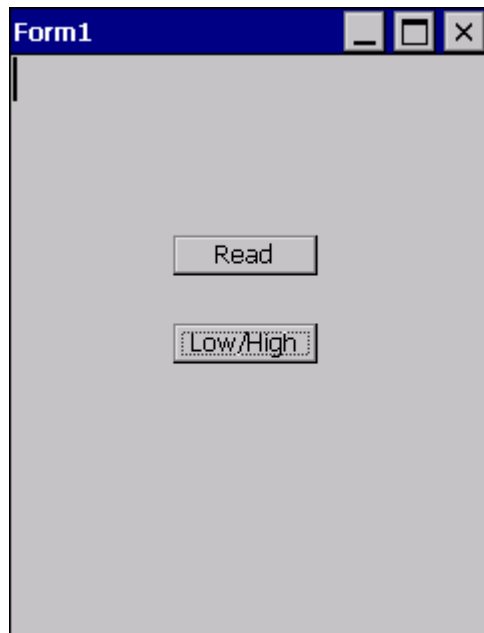


Most of the application will need a dll called CStructBuilder.dll, this dll help the API to communicate between C and C#, this dll is located at the Bin folder into the API folder.

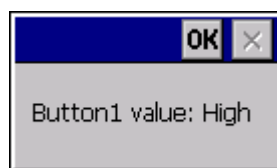
4. Run the application

Now go to the target device and run the application.

1. On the target's desktop, double-click **My Device**, and the double-click **MyApp**.



2. Click the **Read** button to read the value of the Button1.
A message box is displayed with the value of the Button1:



3. Click the **Low/High** button to change the value of the gpio configured as Led1, you will see that the gpio changes his value
4. Close the application by clicking the **X** on the application's title bar.