Use Native Libraries

Contents

Introduction

Most hardware manufacturers provide SDK for their hardware in native libraries, written in C language, Assembly language, or other unmanaged languages, that is, libraries not for Microsoft .Net Framework. Some manufacturers do not include libraries for .Net Framework in their SDK. In such cases, we need to create a .Net Framework wrapper so that such native libraries can be used in Limnor Studio.

We use K8055 Interface Board as an example to show how it is done.

Native DLL Location

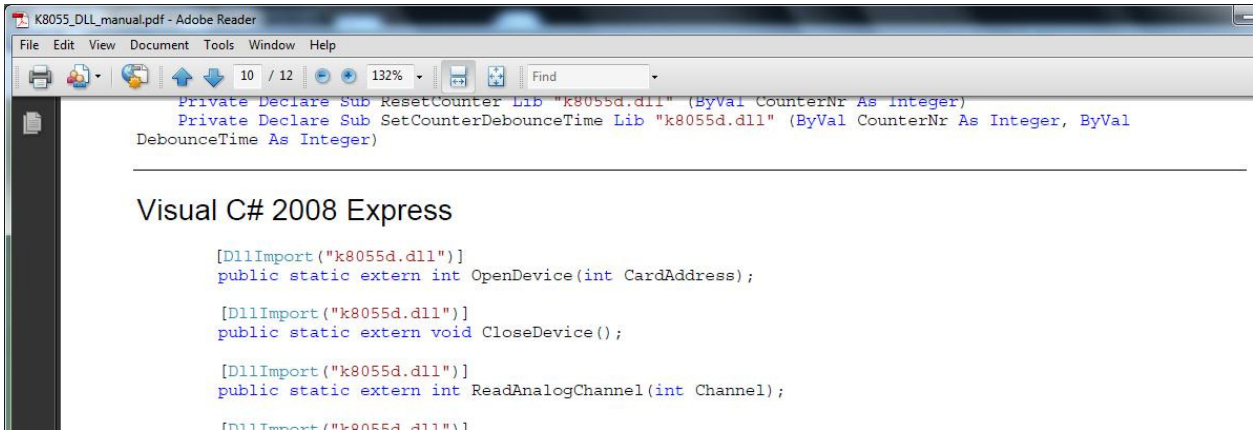
The native DLL must be copied to {System32} folder or {SysWOW64} folder, as the manufacturers stated in their SDK. Typically it is c:\Windows\System32, or c:\Windows\SysWOW64.

Get Library Function Declarations

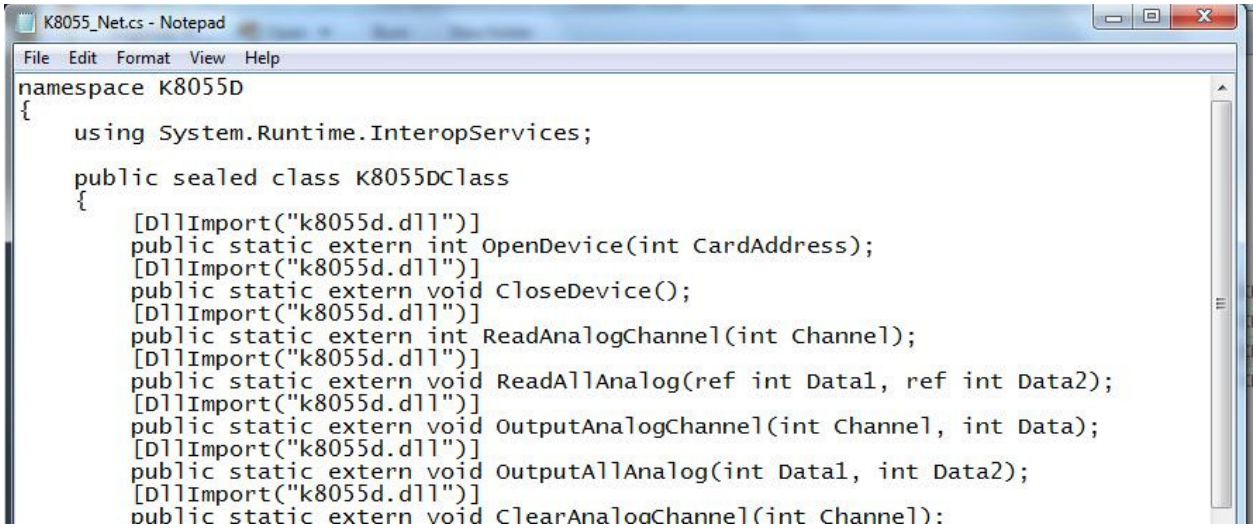
We need to get function declarations expressed in C# language. Some manufacturers provide C# function declarations. If a manufacturer does not provide C# function declarations then we need to convert from other languages provided by the manufacturer. Such conversion usually is straight forward.

Suppose we download the K8055 SDK from
http://www.velleman.eu/distributor/support/downloads/?code=K8055.

The SDK contains a file K8055_DLL_manual.pdf. This file lists C# function declarations for the SDK:



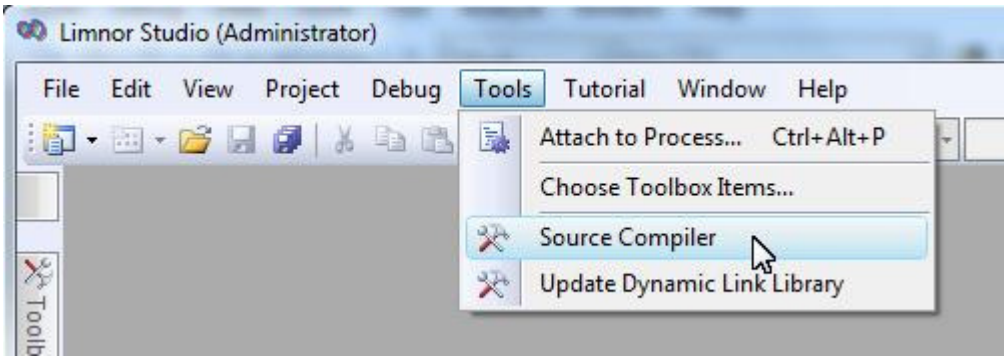
Copy all the C# declarations from this file into a text file and change the file extension from ".txt" to ".cs". Add a class name for it.



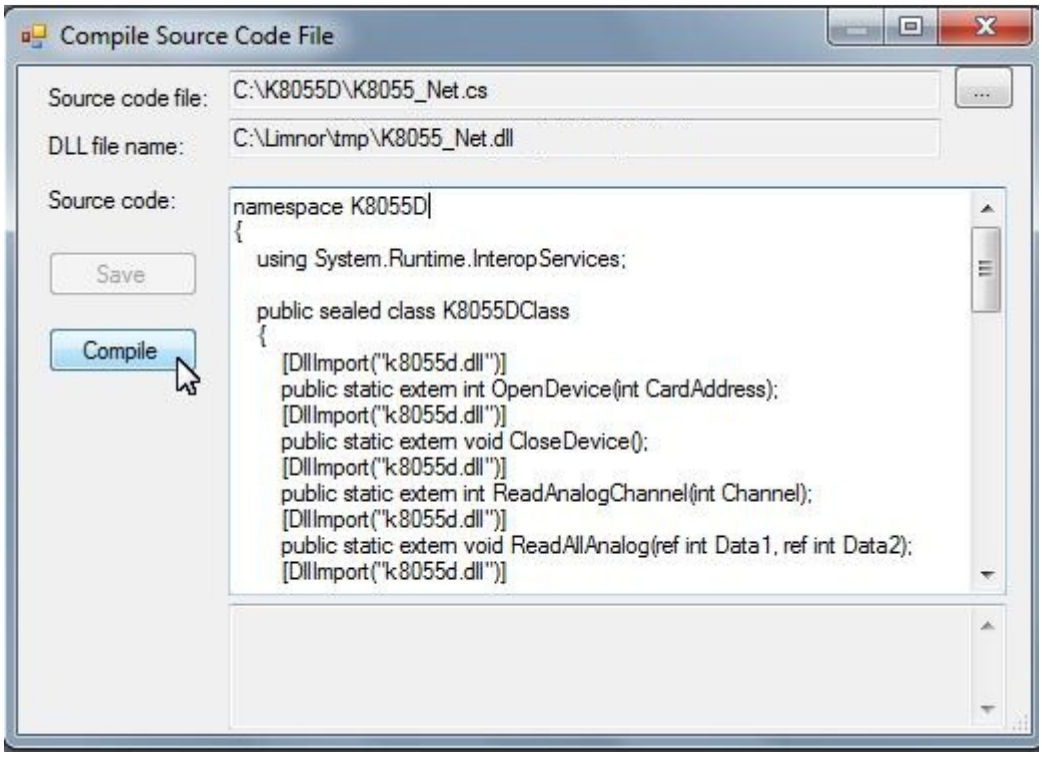
"K8055D" and "K8055DClass" are two names we arbitrarily chosen for this sample. The whole file will be included at the end of this document. We may also call K8055DClass a wrapper class.

Generate Wrapper DLL

In Limnor Studio, choose menu "Tools | Source Compiler".



Select the cs file we created previously:

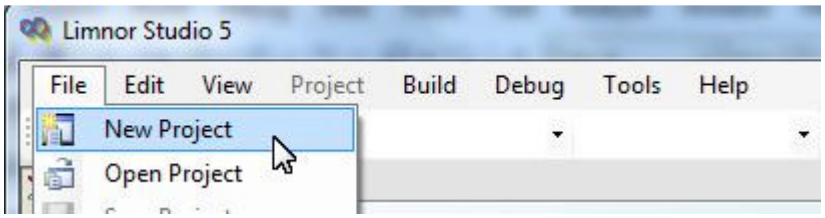


Remember where the new DLL file will be generated because later we will need to use the DLL file.

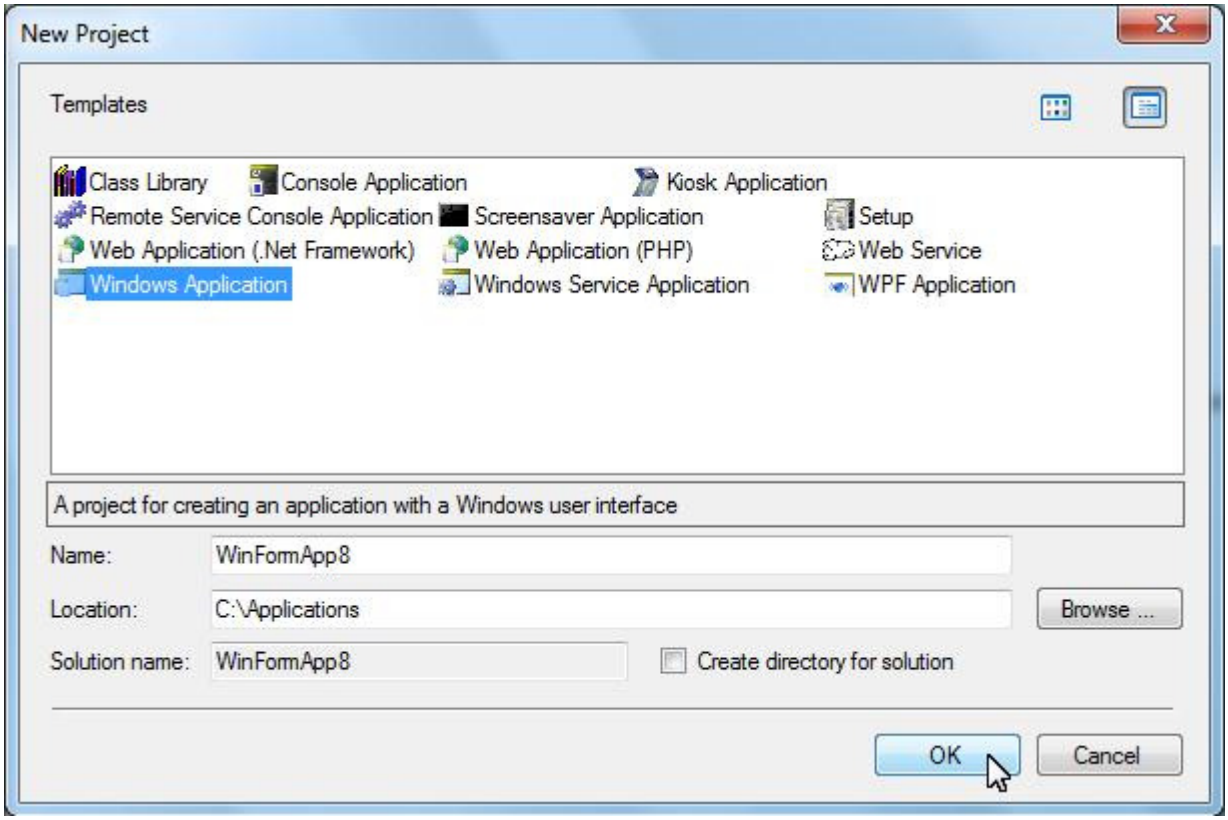
Click the “Compile” button. The DLL file is generated.

Use the Wrapper DLL.

We create a Windows Application project to use the K8055 SDK.



Select “Windows Application”:

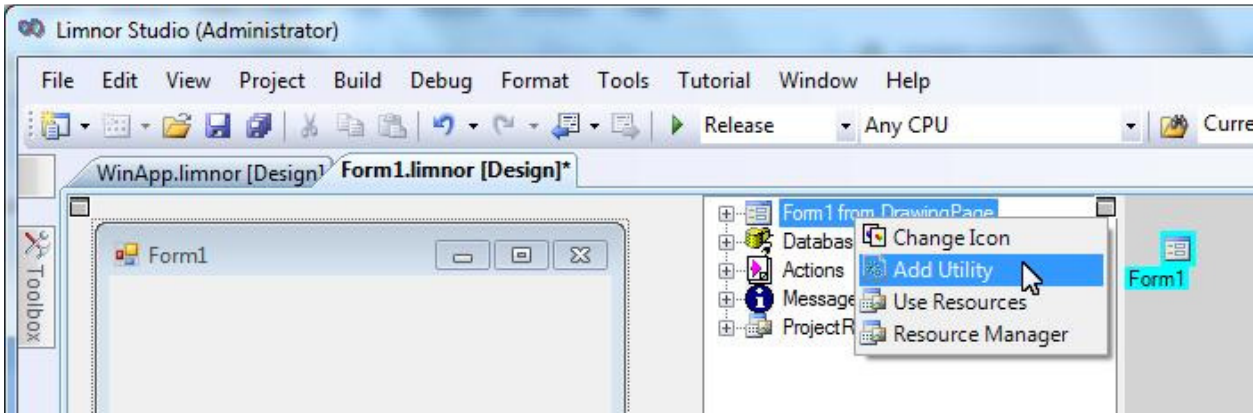


A new Windows Application project is created. It has one Form named Form1.

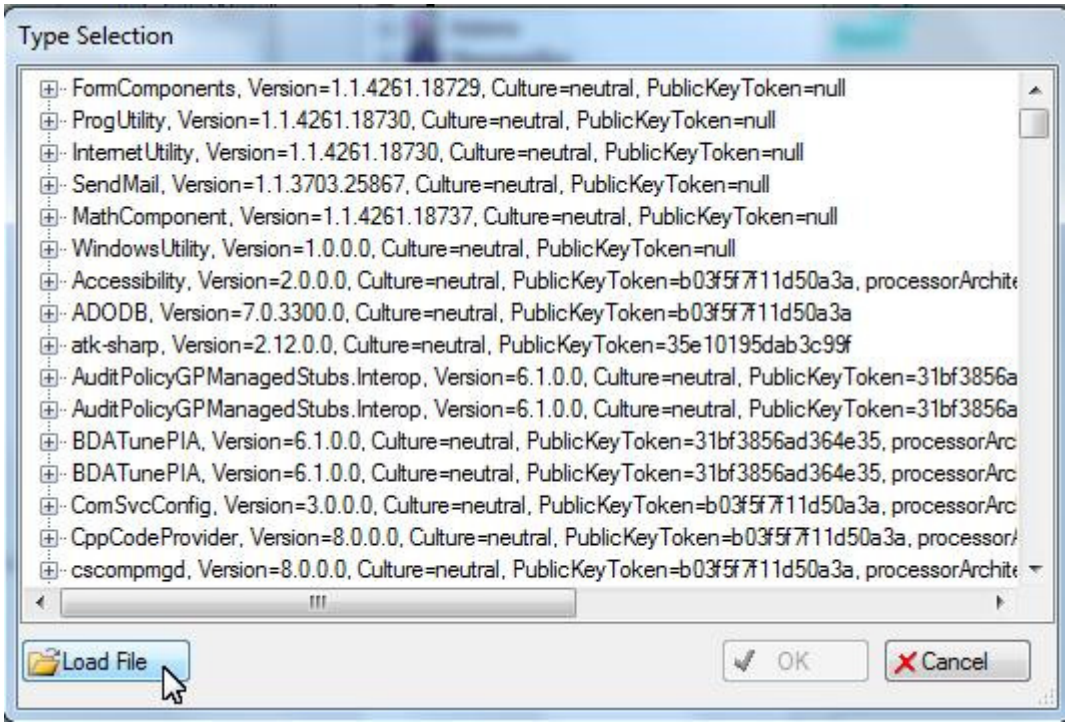
The way to use the wrapper DLL is to create actions using the functions contained in the DLL.

To make it easier, we may add the wrapper class to Form1 as shown below.

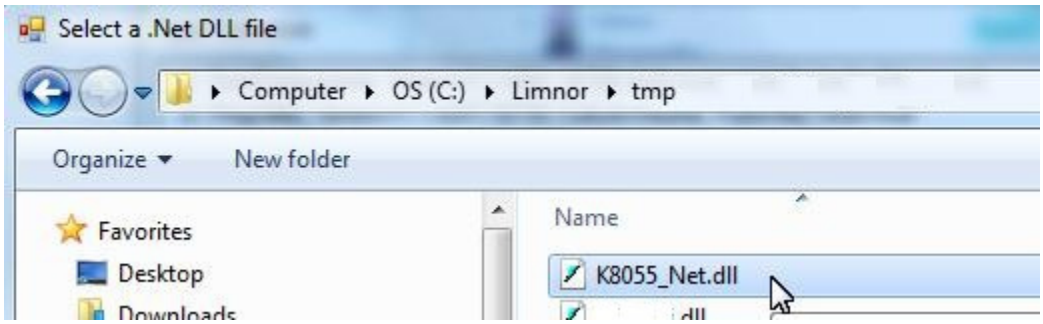
Right-click a root node, choose “Add Utility”:



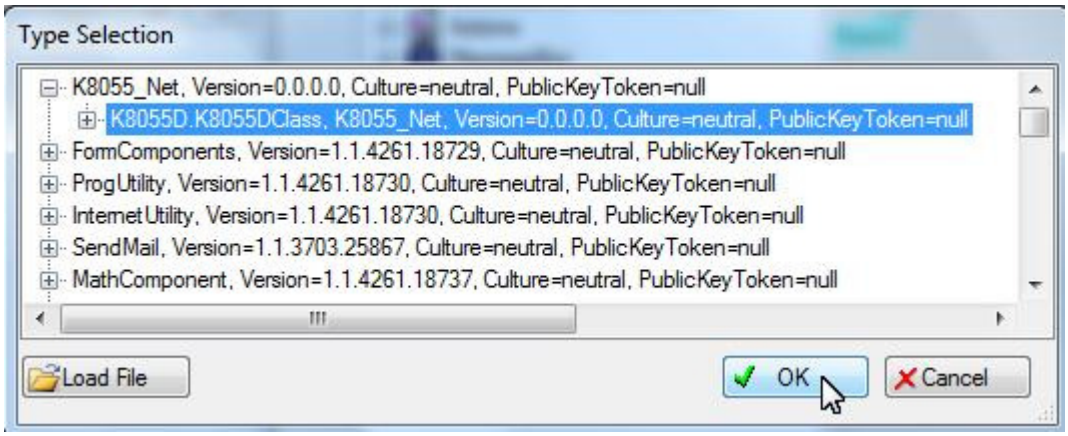
Click “Load File”:



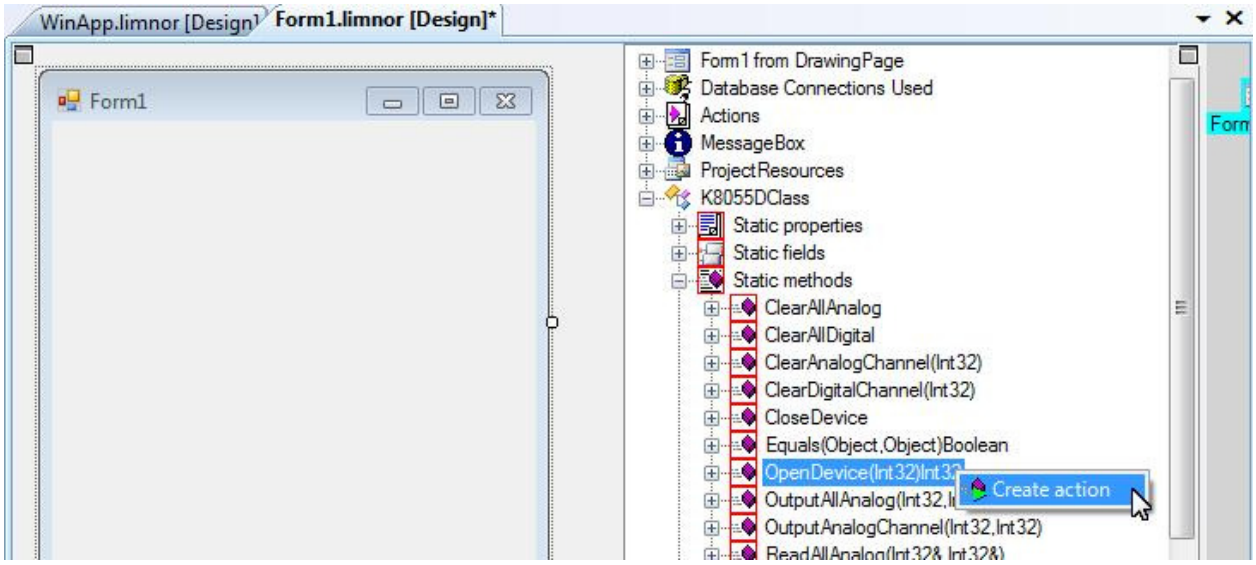
Select the wrapper DLL we generated:



Select the wrapper class in the wrapper DLL:



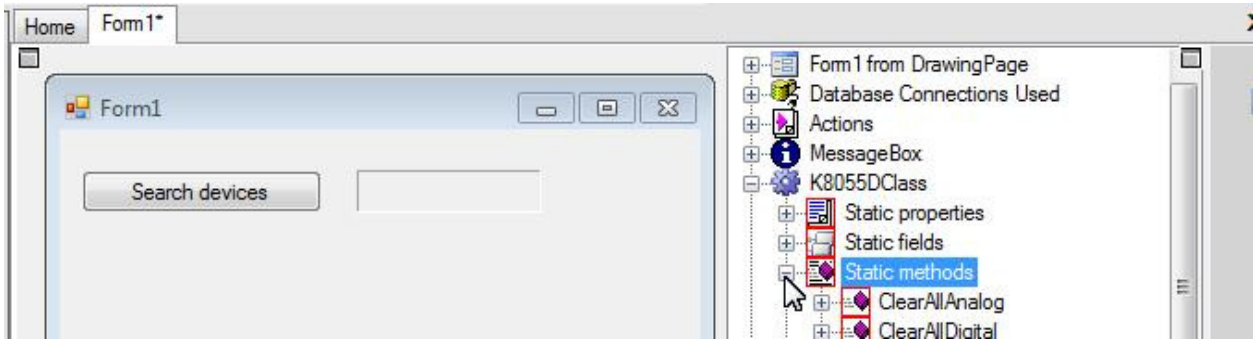
The wrapper class appears in the Object Explorer. All its functions are listed under it. To use a function, right-click it and choose “Create action”:



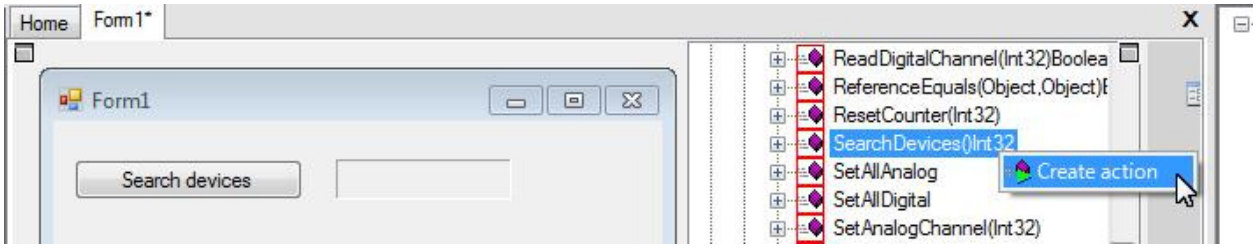
Let’s show an example of executing the interface board action.

We add a button and a label to the form. When the user clicks the button, we want to execute SearchDevices and display the result of this action to the label.

Let’s create an action to execute SearchDevices. To find the SearchDevices methods, expand “Static methods” of the K8055DClass:

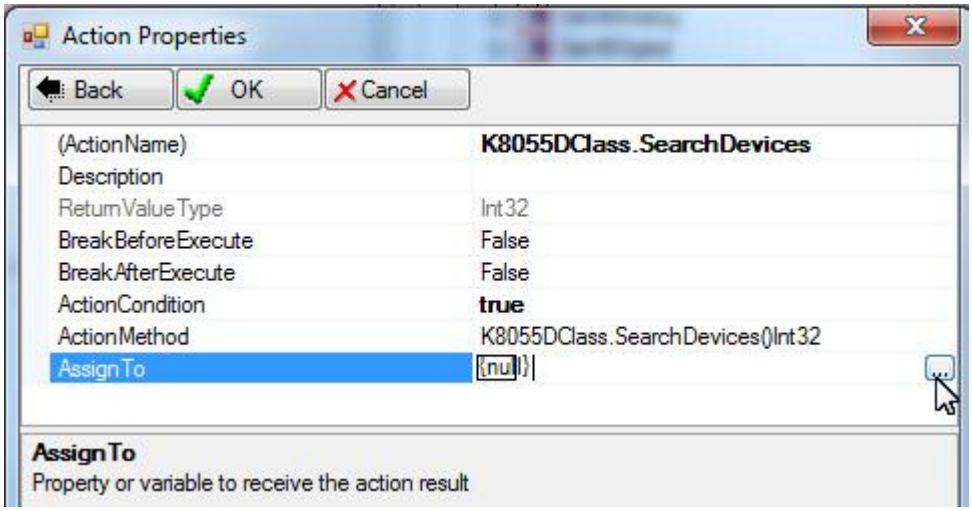


Scroll down and find SearchDevices. Right-click SearchDevices; choose “Create action”:

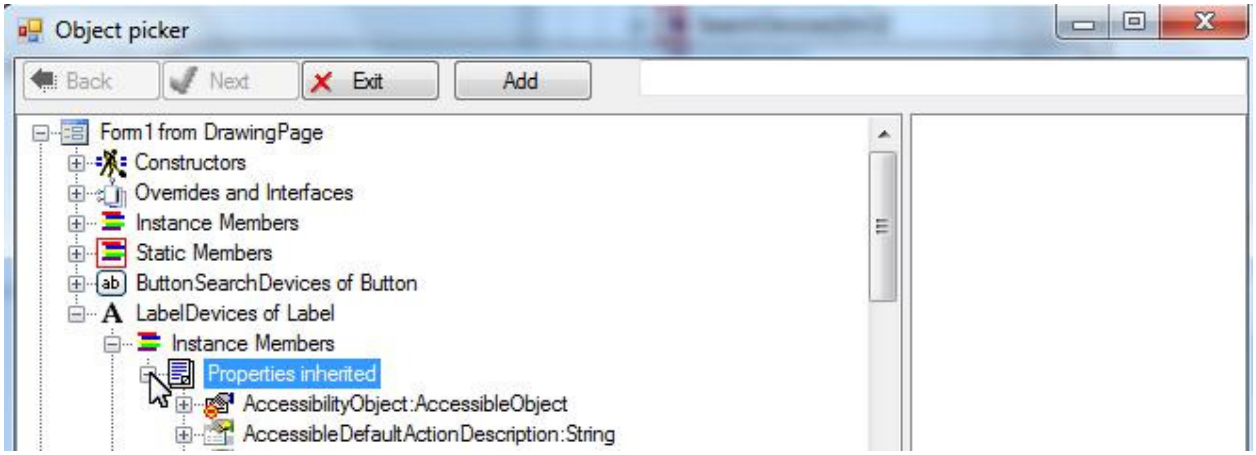


A dialogue box appears showing the properties of the action.

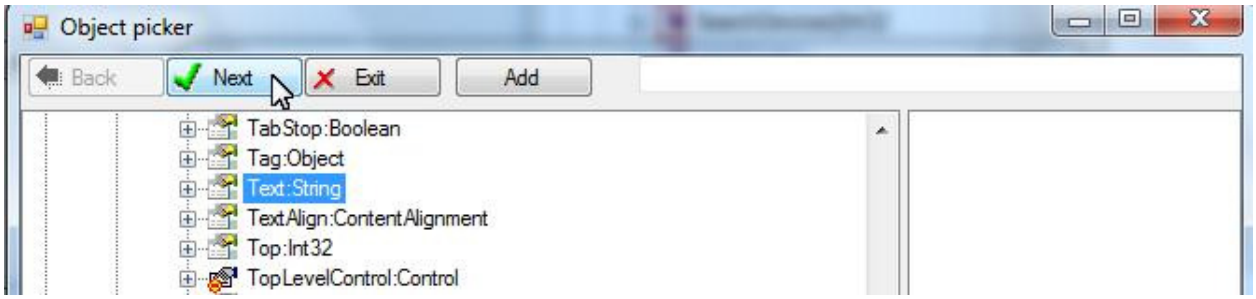
According to the SDK manual, SearchDevices will return an integer as the result. We may set the action’s AssignTo property to the label so that the action result will be displayed in the label:



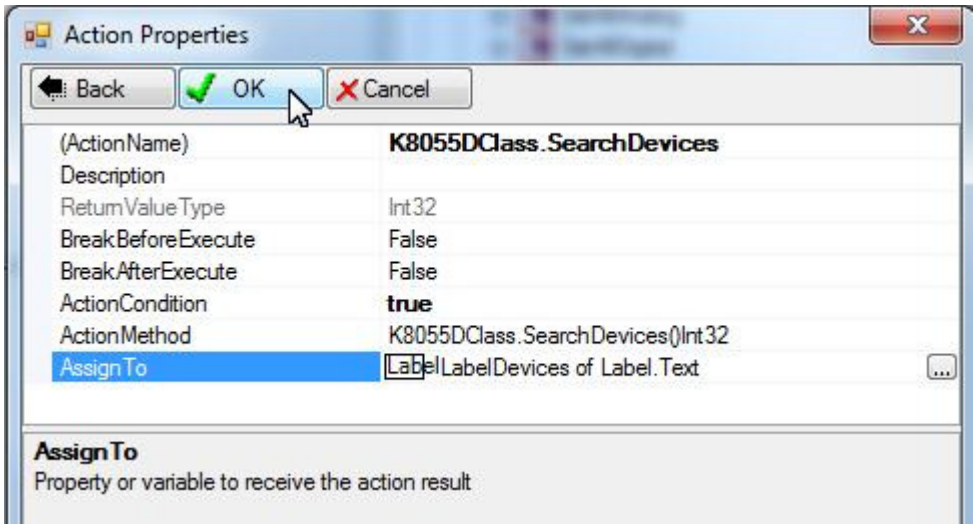
We need to assign the action result to the Text property of the label. So, expand the “Inherited properties” node of the label:



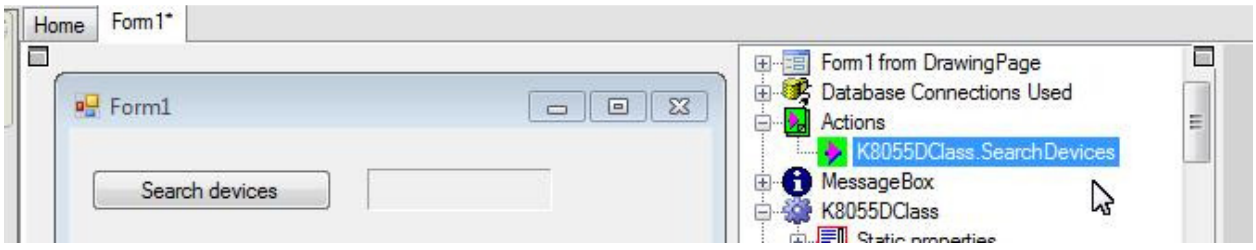
Scroll down to find and select the Text property:



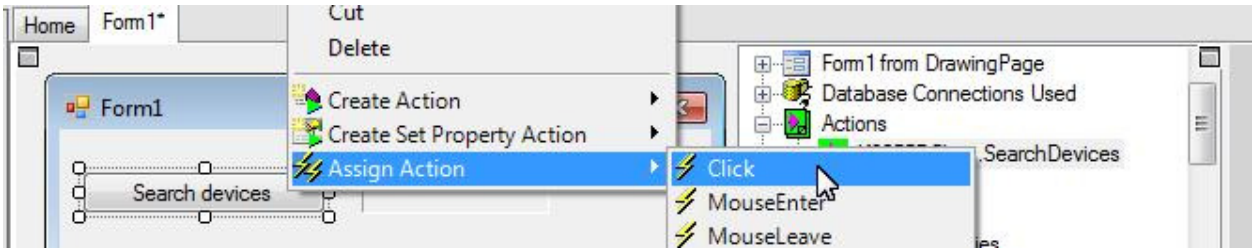
This action execute the SearchDevices action and displays the result in the label:



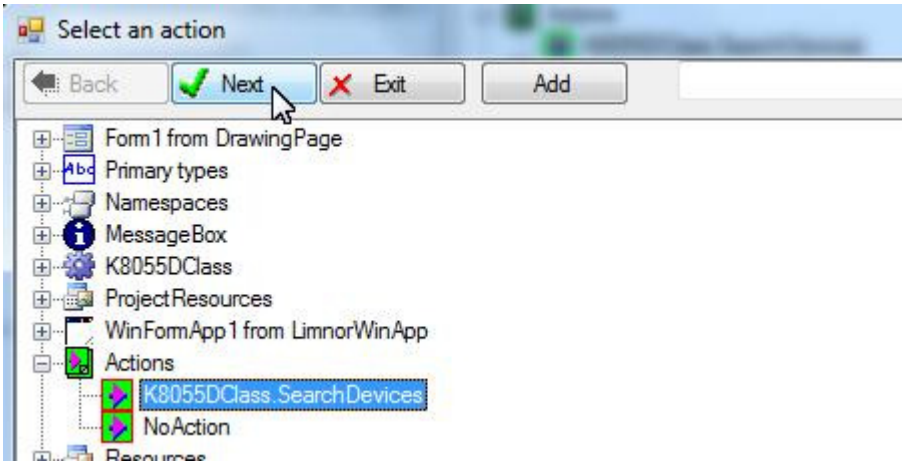
The action can be found under the SearchDevices node and Actions node:



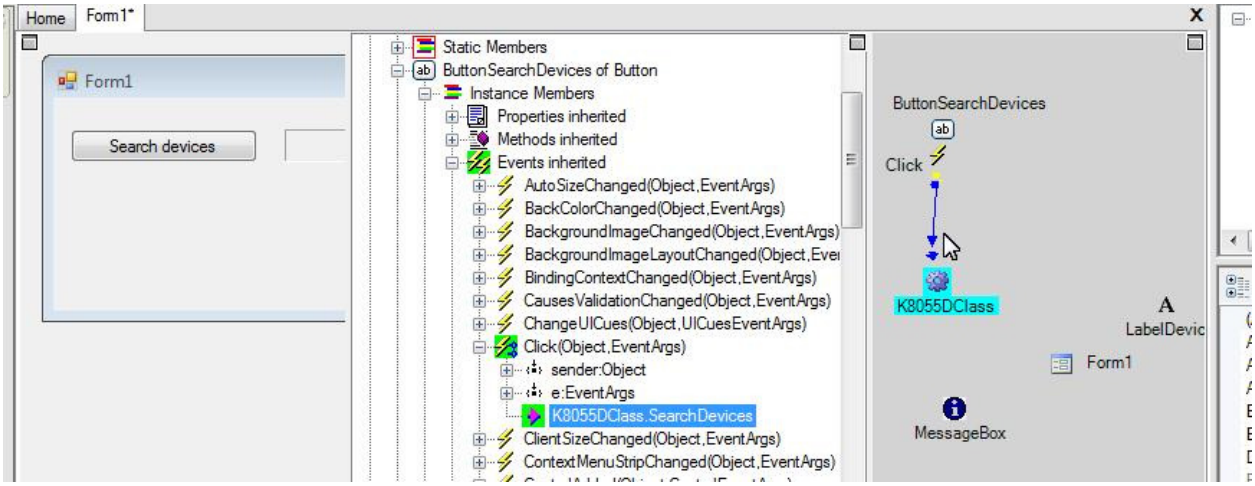
To execute this action when the user clicks the button, we need to assign this action to the Click event of the button. Right-click the button; choose “Assign Action”; choose “Click” event:



Choose the action we just created and click “Next”:



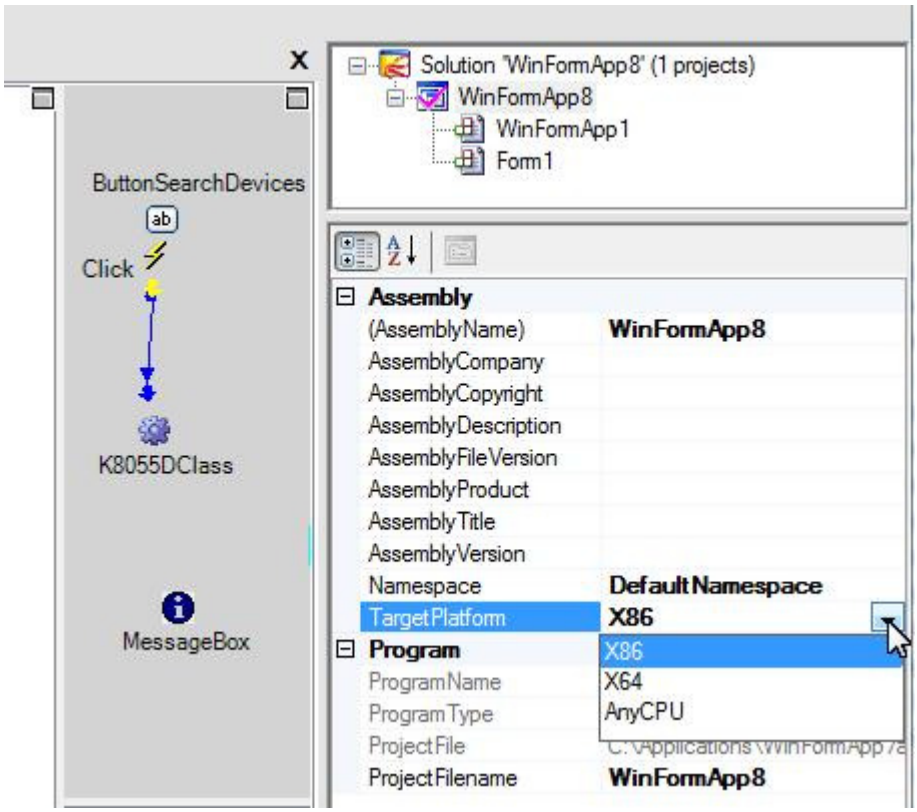
The action is assigned to the Click event of the button. We can see that the action appears under the Click event:



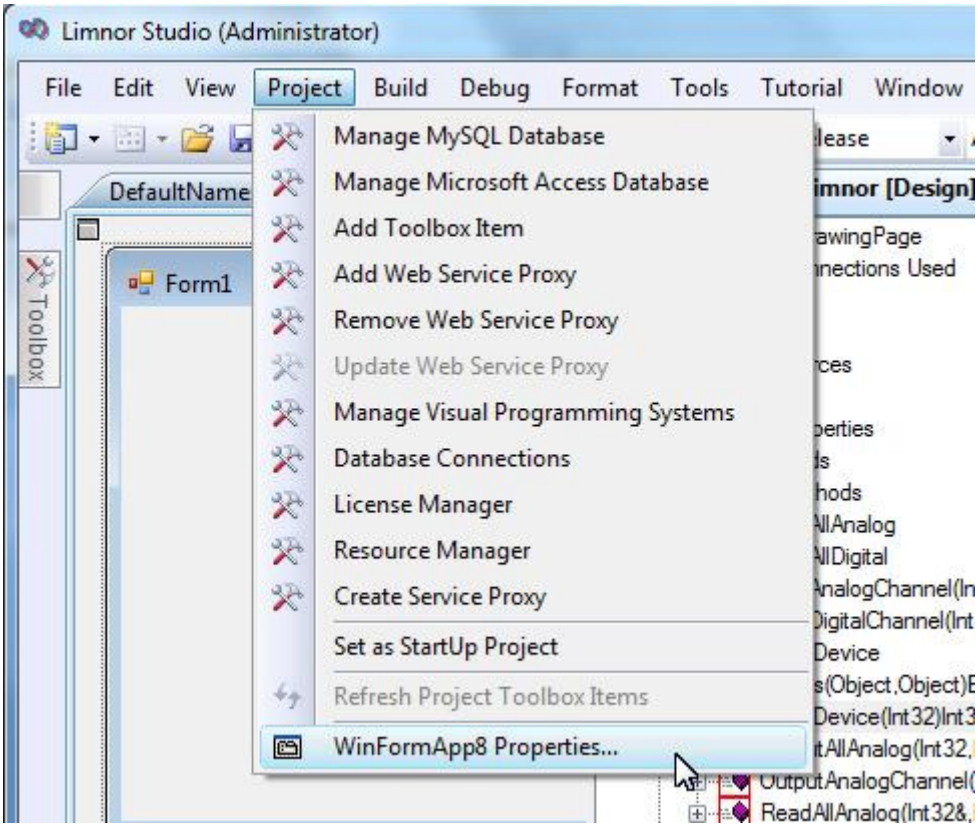
32-Bit Libraries

If the native DLL is of 32-bit then your project must be compiled for 32-bit.

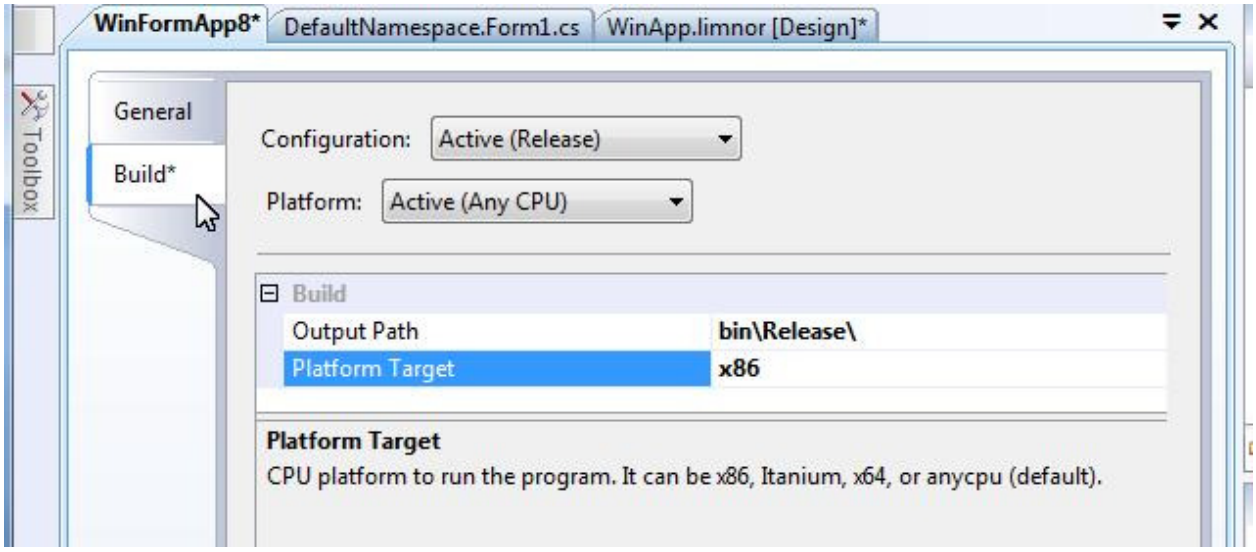
For Limnor Studio 5, select the project in the Solution Explorer; set the TargetPlatform to X86:



For Limnor Studio VS, choose menu “Project | {Project Name} Properties”:



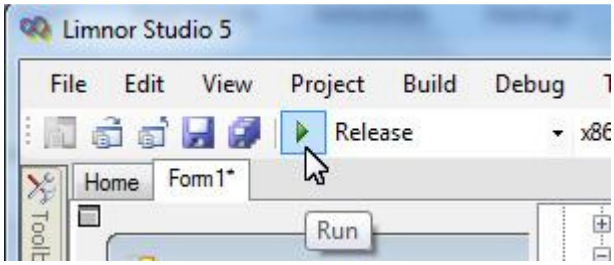
Set the Platform Target to x86:



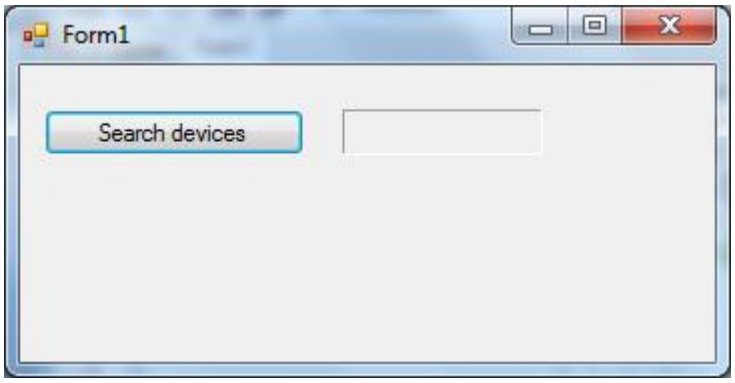
For more information, see http://www.limnor.com/studio_x86.html

Test

We may test the application:



The form appears:



Click the button. The SearchDevices function of the interface board executes. “0” appears in the label indicating that the computer does not have a K8055 interface board:

Sample C# file

Below is the C# file we used for generating the wrapper DLL. All function declarations are copied from the SDK manual K8055 interface Board.

```
namespace K8055D
{
    using System.Runtime.InteropServices;

    public sealed class K8055DClass
    {
        [DllImport("k8055d.dll")]
        public static extern int OpenDevice(int CardAddress);
        [DllImport("k8055d.dll")]
        public static extern void CloseDevice();
        [DllImport("k8055d.dll")]
        public static extern int ReadAnalogChannel(int Channel);
        [DllImport("k8055d.dll")]
        public static extern void ReadAllAnalog(ref int Data1, ref int Data2);
        [DllImport("k8055d.dll")]
        public static extern void OutputAnalogChannel(int Channel, int Data);
        [DllImport("k8055d.dll")]
        public static extern void OutputAllAnalog(int Data1, int Data2);
        [DllImport("k8055d.dll")]
        public static extern void ClearAnalogChannel(int Channel);
        [DllImport("k8055d.dll")]
        public static extern void SetAllAnalog();
        [DllImport("k8055d.dll")]
        public static extern void ClearAllAnalog();
        [DllImport("k8055d.dll")]
        public static extern void SetAnalogChannel(int Channel);
        [DllImport("k8055d.dll")]
        public static extern void WriteAllDigital(int Data);
        [DllImport("k8055d.dll")]
        public static extern void ClearDigitalChannel(int Channel);
        [DllImport("k8055d.dll")]
        public static extern void ClearAllDigital();
        [DllImport("k8055d.dll")]
        public static extern void SetDigitalChannel(int Channel);
        [DllImport("k8055d.dll")]
        public static extern void SetAllDigital();
        [DllImport("k8055d.dll")]
```

```
        public static extern bool ReadDigitalChannel(int Channel);
        [DllImport("k8055d.dll")]
        public static extern int ReadAllDigital();
        [DllImport("k8055d.dll")]
        public static extern int ReadCounter(int CounterNr);
        [DllImport("k8055d.dll")]
        public static extern void ResetCounter(int CounterNr);
        [DllImport("k8055d.dll")]
        public static extern void SetCounterDebounceTime(int CounterNr, int
DebounceTime);
        [DllImport("k8055d.dll")]
        public static extern int Version();
        [DllImport("k8055d.dll")]
        public static extern int SearchDevices();
        [DllImport("k8055d.dll")]
        public static extern int SetCurrentDevice(int lngCardAddress);
    }
}
```