Instruction

Z-Wave DLL User Guide

Document No.:	INS10250
Version:	6
Description:	This document describes the architecture and functionality of the Z Wave DLL.
Written By:	DDA;JFR;JRM
Date:	2009-05-14
Reviewed By:	JFR;JRM
Restrictions:	Partners Only

Approved by:

Date	CET	Initials	Name	Justification
2009-05-14	10:13:21	NTJ	Niels Thybo Johansen	

This document is the property of Zensys A/S. The data contained herein, in whole or in part, may not be duplicated, used or disclosed outside the recipient for any purpose. This restriction does not limit the recipient's right to use information contained in the data if it is obtained from another source without restriction.

CONFIDENTIAL

REVISION RECORD

Doc. Rev	Date	By	Pages affected	Brief description of changes
1	20050406	JFR	All	Minor changes
2	20051212	TKR	All	Removed OpenNETCF Updated architecture
3	20060105	MVO	All	New 1st page/header/footer contents. New Doc No
4	20060615	CP JFR	All	Updated document to reflect implementation of PocketPC DLL
5	20060628	JRM	All	Updated information regarding setup on PC versus Pocket PC.
	20080204	IHM	All	Pocket PC platform has been removed
6	20080822	IHM	All	Completely revised according to the new architecture
7	20090415	DDA	All	Revised and minor updates added.

CONFIDENTIAL

Table of Contents

List of Figures

CONFIDENTIAL

1 ABBREVIATIONS

Abbreviation	Explanation
API	Application Programming Interface
CD	Compact Disc
COM	Serial port interface on IBM PC-compatible computers
DLL	Dynamic Link Library
GUI	Graphical User Interface
PC	Personal computer
USB	Universal Serial Bus, a serial bus standard to interface devices
XML	eXtensible Markup Language

2 INTRODUCTION

2.1 Purpose

This document describes the architecture and functionality of the Z-Wave DLL that supports Z-Wave enabled PC based applications.

This document includes conceptual overviews, step-by-step procedures, and information about samples.

More and up-to-date information on Z-Wave DLL API references and practical implementation can be found in the **Z_Wave DLL Documentation.chm** online help file, which is automatically generated when the Z-Wave DLL is installed (C:\DevKit_x_yy\PC\Bin\ZWaveDll\setup.exe). By default, it is located at

C:\Program Files\Zensys\Z-Wave Dll\Z_Wave DLL Documentation.chm.

2.2 Audience and prerequisites

The audience is Z-Wave partners and Zensys developers. It is assumed that partner developers already have and are familiar with the Z-Wave Developer's Kit.

CONFIDENTIAL

3 WHAT IS Z-WAVE DLL

Z-Wave DLL is a framework that simplifies the development of Z-Wave enabled Windows Forms or Console applications for Microsoft .NET Framework platform (Windows XP and Windows Vista applications).

The main features of Z-Wave DLL are:

- provides interfaces for calling Serial API functions
- handles request timeouts
- repeat requests if needed
- handles acknowledge signals from devices
- handles exceptions
- logs events

Attention! Please note that the current version contains basic set of the functions to operate with RS232 interface only.

Custom Z-Wave enabled .NET application, which is built on top of Z-Wave DLL, should be thin GUI-oriented Windows Forms application or console application with reference to this DLL in the solution.

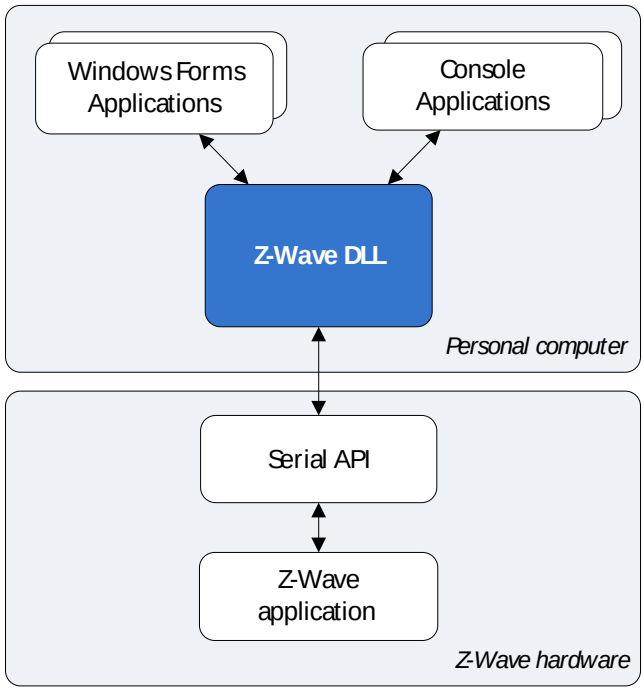

Figure 1. Positioning of Z-Wave DLL within custom solution

CONFIDENTIAL

4 ARCHITECTURE AND IMPLEMENTATION

Architecture of Z-Wave DLL includes the following layers:

1. **Transport Layer**

   Layer represents the communication between PC and Z-Wave device. It contains basic set of the functions to operate with RS232 (serial port) interface.

   For the future versions of Z-Wave DLL the USB is considered as additional interface to be implemented and supported.

2. **Session Layer**

   Layer represents the basic queue which operates with all the flows between layers (Transport, Application, etc.)

3. **Frame Layer**

   Layer represents the basic Frame types and structures, as well as the functions for parsing and conversion.

4. **Application Layer**

   Basic set of the functions (both application-level and protocol-level) which can be combined into workflows (usually as group of functions).

5. **High-level Application Layer**

   Layer represents the set of the end functions that involve some workflows from Application Layer. It also can contain the additional workflows.

Below please find the sequence diagram which shows the sample interaction between the Z-Wave DLL layers.

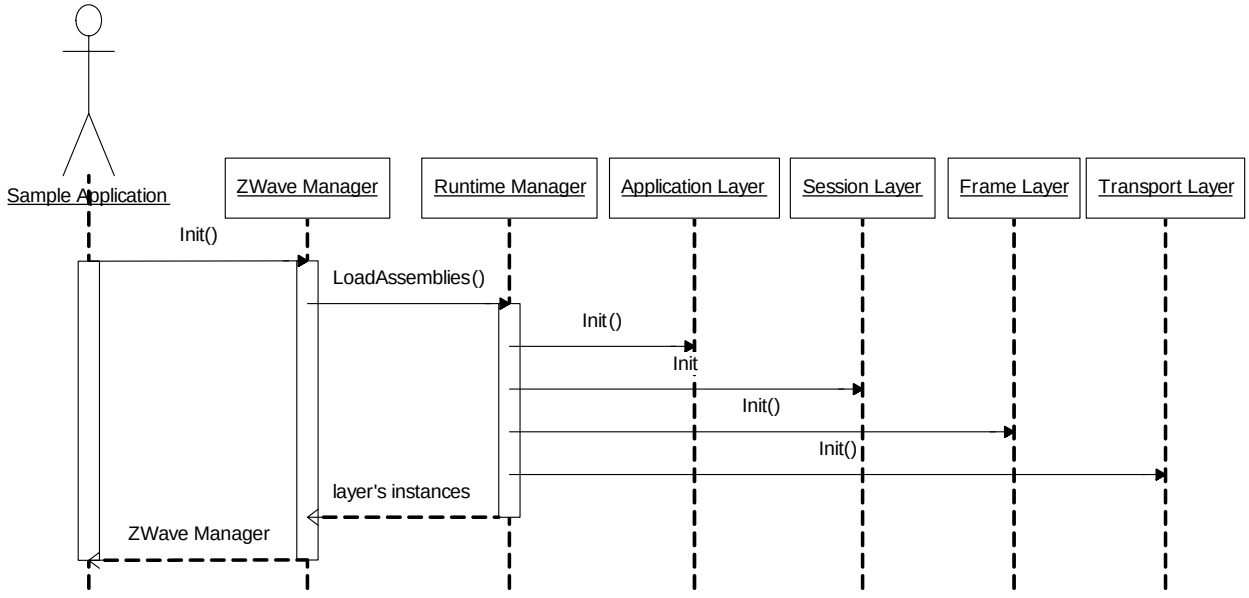

Figure 2. Sequence diagram of layers interaction

Please note ZWaveManager at the diagram above, which provides the methods for initialization and maintenance of the Z-Wave DLL library.

CONFIDENTIAL

Below please find another sequence diagram with sample of layer interaction during execution of Add Node feature.

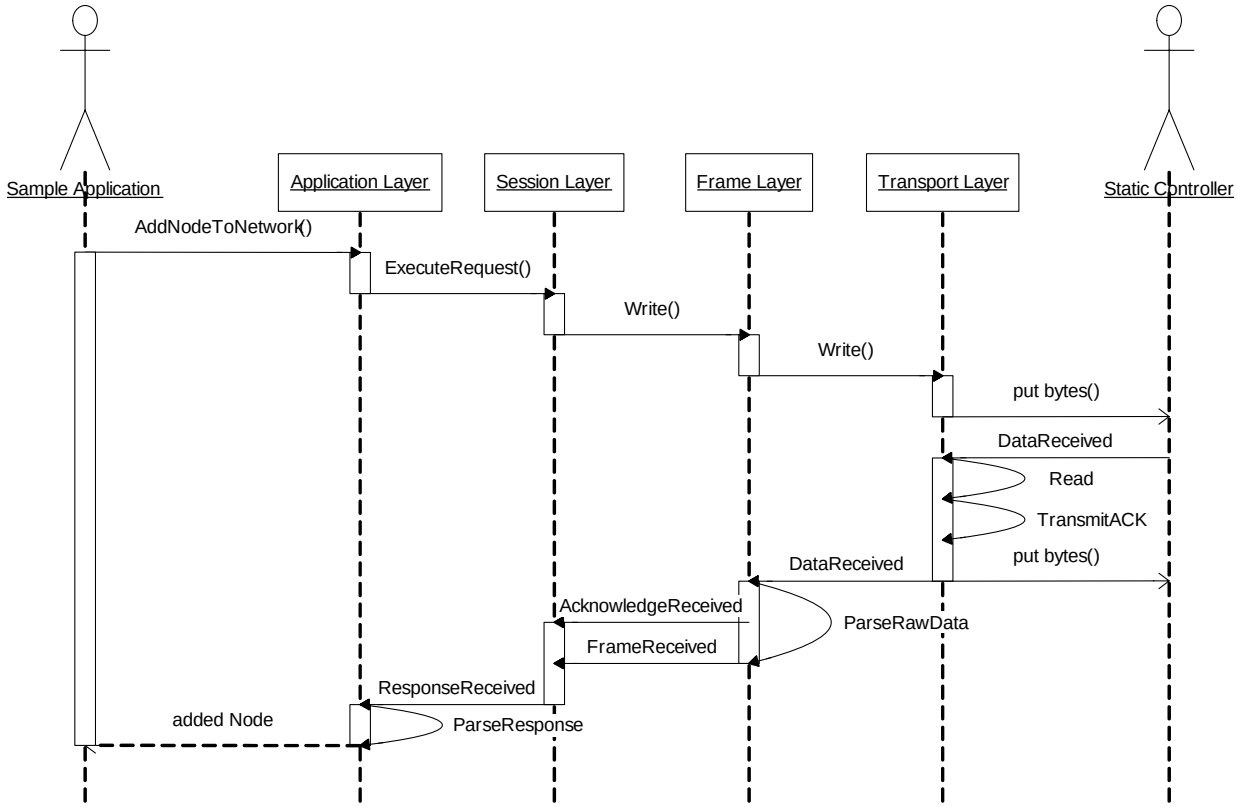

Figure 3. Add Node sequence diagram

The layers of Z-Wave DLL are implemented through interfaces.

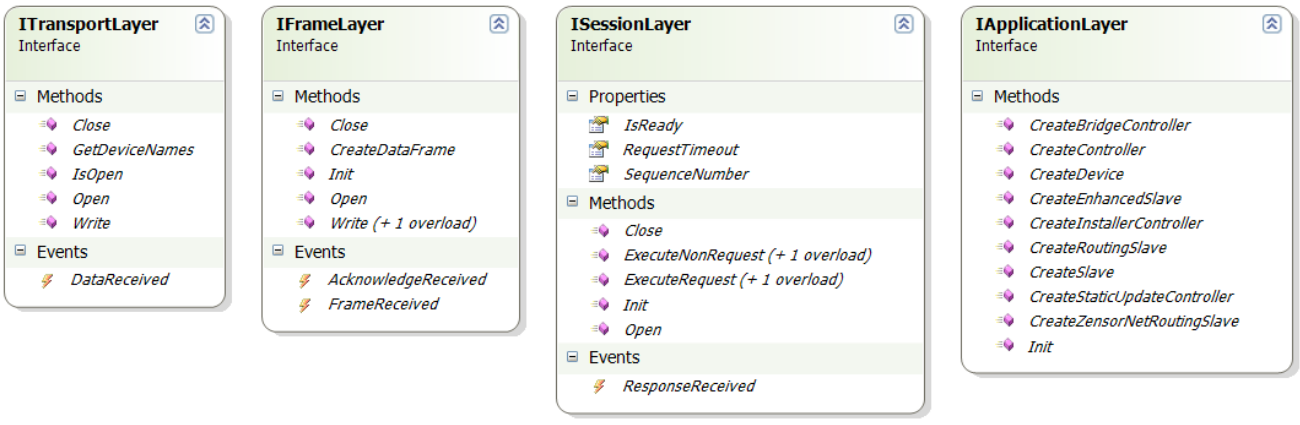

Figure 4. Layer interfaces (reduced set)

Below please find the default layer implementation.

CONFIDENTIAL

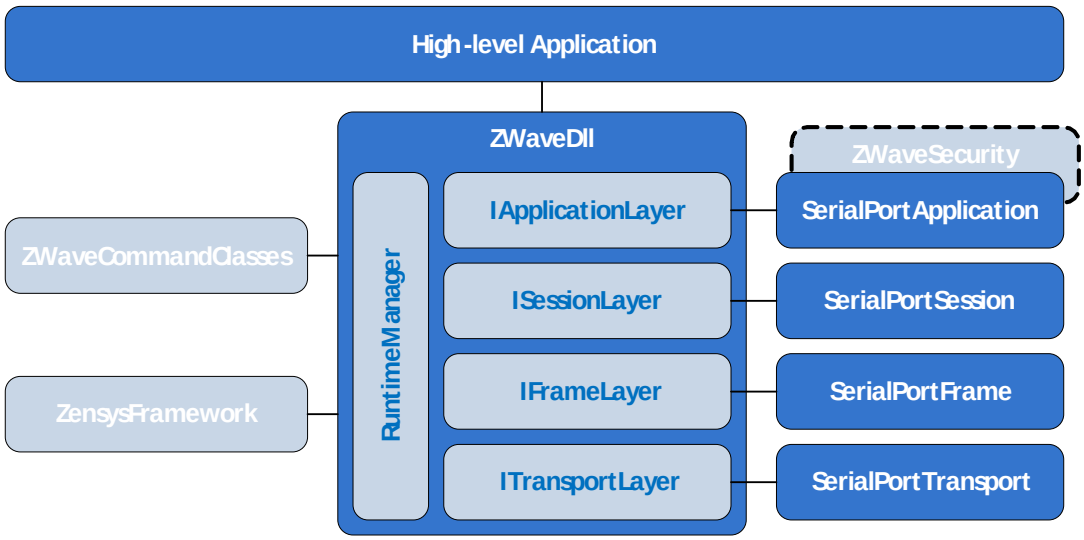

Figure 5. Default implementation

Each layer implementation is located in the dedicated project:

- **High-level Application** is the custom .NET application built on top of Z-Wave DLL
- **SerialPortApplication** is an application layer implementation
- **SerialPortFrame** is a frame layer implementation
- **SerialPortSession** is a session layer implementation
- **SerialPortTransport** is a transport layer implementation
- **ZWaveCommandClasses** is an external library (compiled into Z-Wave DLL) which contains the automatically generated C# classes and structures for all the Basic Devices, Generic Devices, Specific devices and Command Classes as they are defined in XML file.
- **ZensysFramework** is an external library (compiled into Z-Wave DLL) which contains the common components such as conversion functions, helpers, and wrappers that could be used in any .NET Z-Wave application.
- **ZWaveSecurity** is an optional component which provides encryption/decryption features to support the secure Z-Wave communications.

These layers could be custom-implemented. For example, the Z-Wave Programmer application uses a different API and Frame Layer implementation (see ProgrammerFrame project).

If you want to use a custom layer implementation, then you should call initialization Init() method of the ZWaveManager with appropriate assembly name and layer class name.

The Z-Wave DLL provides communication between high-level Z-Wave enabled application and Z-Wave hardware so that it implements Z-Wave Serial API functions as shown in the figure below.

CONFIDENTIAL

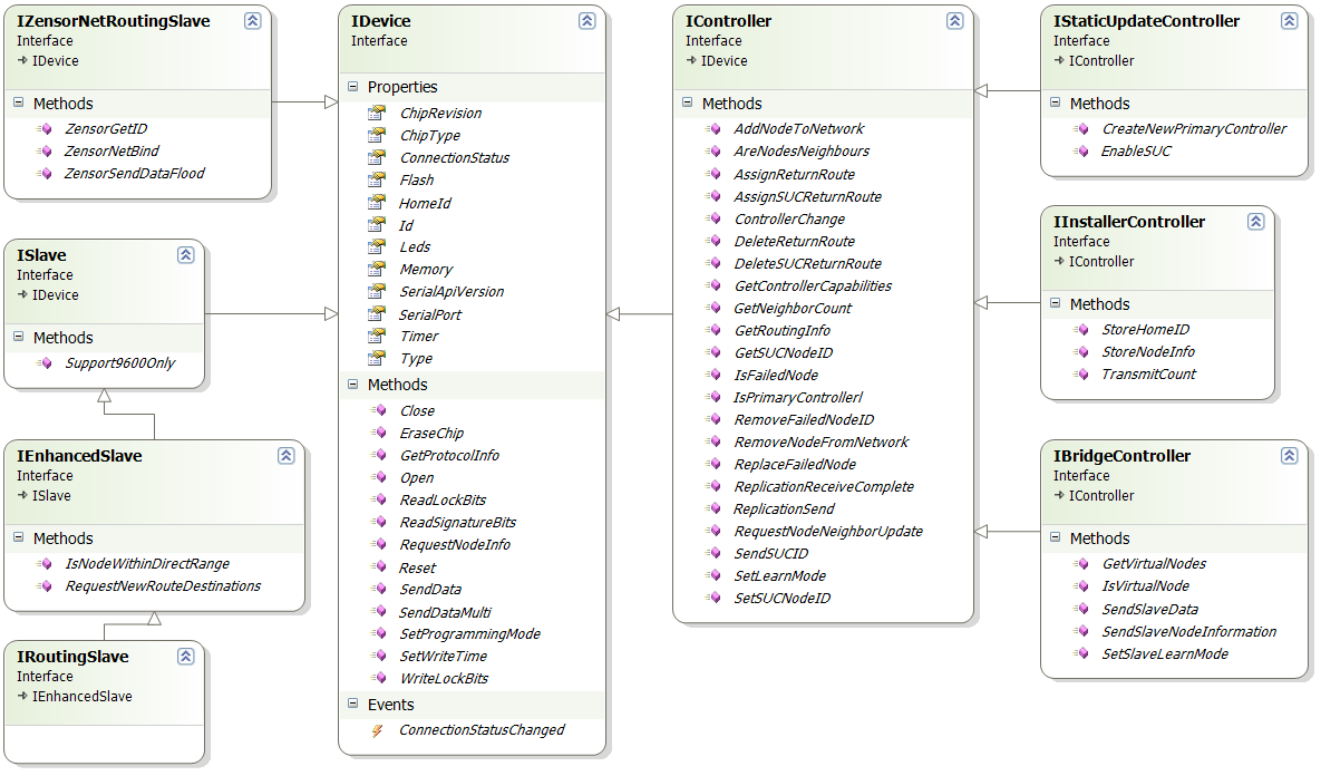

Figure 6. List of Serial API functions grouped by device types (reduced set)

CONFIDENTIAL

5 GETTING STARTED

5.1 Check the prerequisites

The following components should be installed on the computer that you need to develop, build, test, and deploy Z-Wave DLL based .NET application:

- Microsoft .NET Framework, version 2.0 or later
- The Microsoft Visual Studio 2005 development environment
- Windows Installer 3.0 (Windows Installer 3.1 or later is recommended).

Important: Make sure you have the latest service pack and critical updates for the version of Windows that you are running. To find the recent security updates, visit Windows Update.

5.2 Limitations

The setup has only been tested on Windows XP with Service Pack 2 and 3 (x86 platform), but may work on other versions as well.

5.3 Required Z-Wave hardware

Any Z-Wave enabled application requires the appropriate Z-Wave hardware to be connected:

- Program the Z-Wave device with appropriate HEX file, usually Serial API Controller, Installer, Bridge or Slave depending on the Windows application.
  Z-Wave 4.10, 4.11 or newer must be used.
- Establish serial communication to the Z-Wave module.

5.4 Install the Z-Wave DLL

1. Exit all programs.
2. In Microsoft Windows, click the **Start** button, and then click **Control Panel**.
3. In Classic view, double-click **Add or Remove Programs.**
4. Click **Add New Programs**, and then click **CD or Floppy**.
5. Click **Next** and then click **Browse** to locate the "**setup.exe**" in the "<optical_drive_name>\data\DevKit\Libraries\ZWaveDll" folder on Z-Wave Developer's Kit CD or "<drive name>\DevKit_x_xx\Libraries\ZWaveDll" folder on your hard drive in case you installed Z-Wave Developer's Kit already.
6. Click **Finish** to start the installation. The welcome page appears as shown at the figure below. Please note the copyright notification and click **Next** button.

CONFIDENTIAL

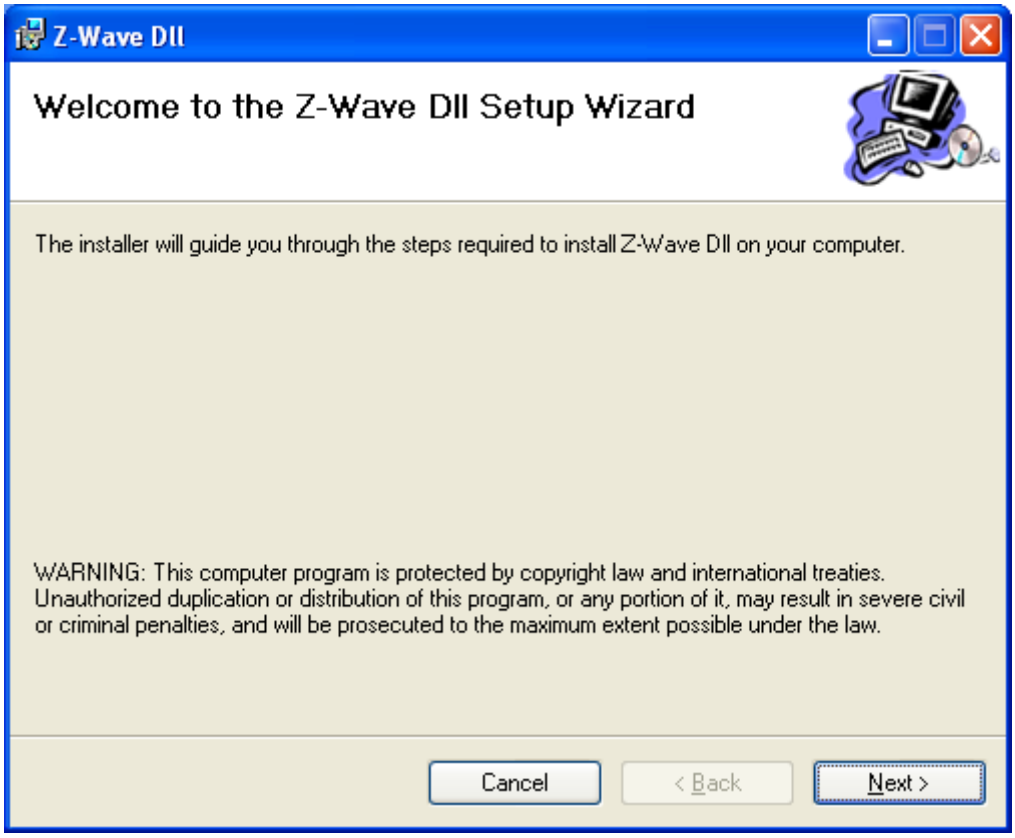

Figure 7. Welcome page of Z-Wave DLL installation

7. Select the installation folder and who should be able to use the Z-Wave DLL.
   Please note, that it is extremely important to do not move the Z-Wave DLL manually after it has been installed into the above specified folder.
   When done, click **Next**.

CONFIDENTIAL

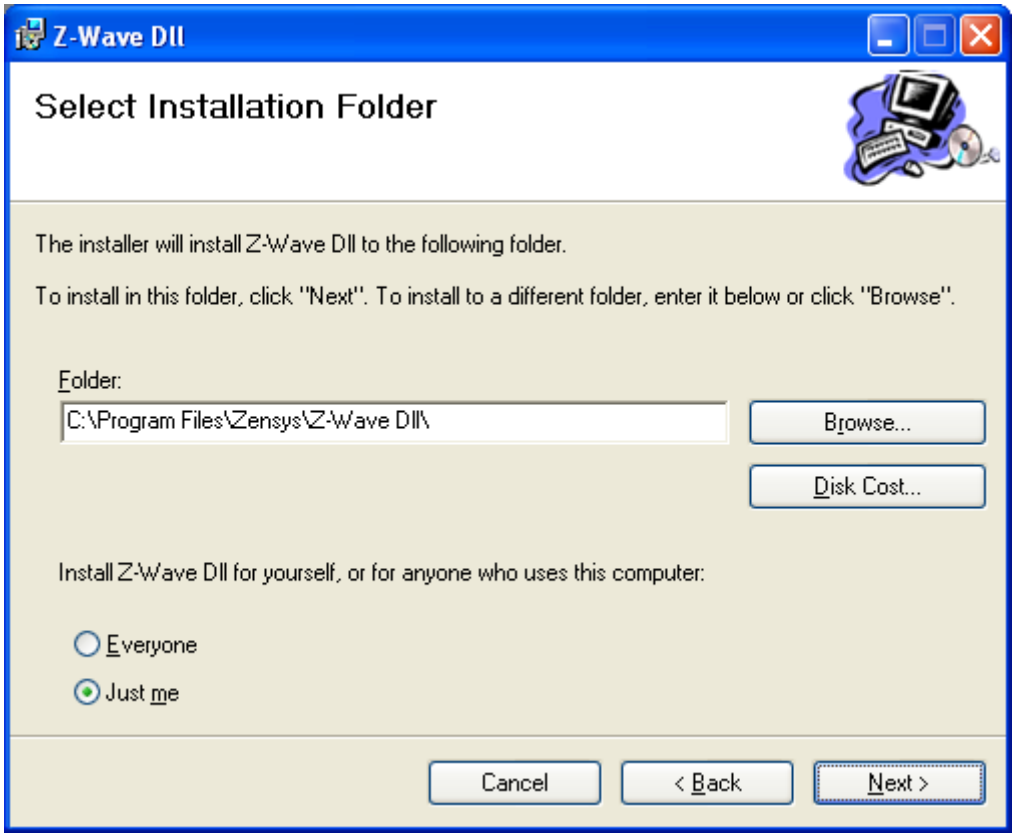

Figure 8. Installation Folder

8. Installation confirmation appears. Click **Next** again to confirm and start the installation.

CONFIDENTIAL

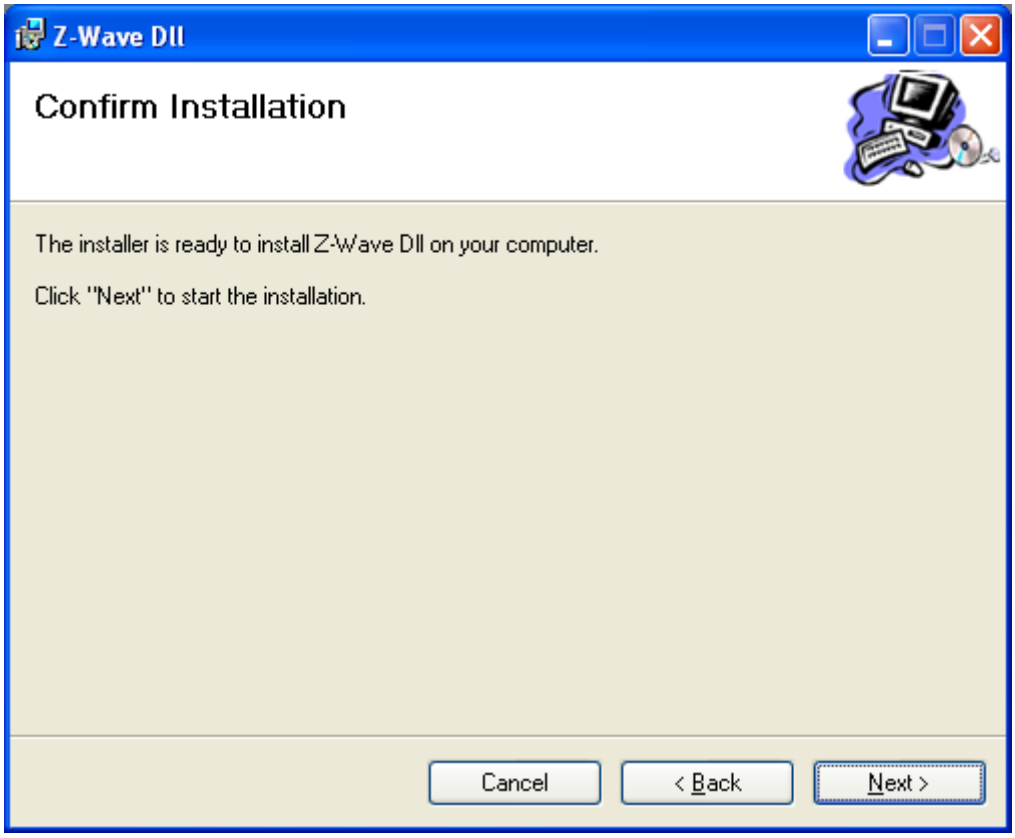

Figure 9. Confirmation page of Z-Wave UPnP Bridge installation

9. The actual installation procedure will pass with progress indicator and final confirmation appears.

CONFIDENTIAL

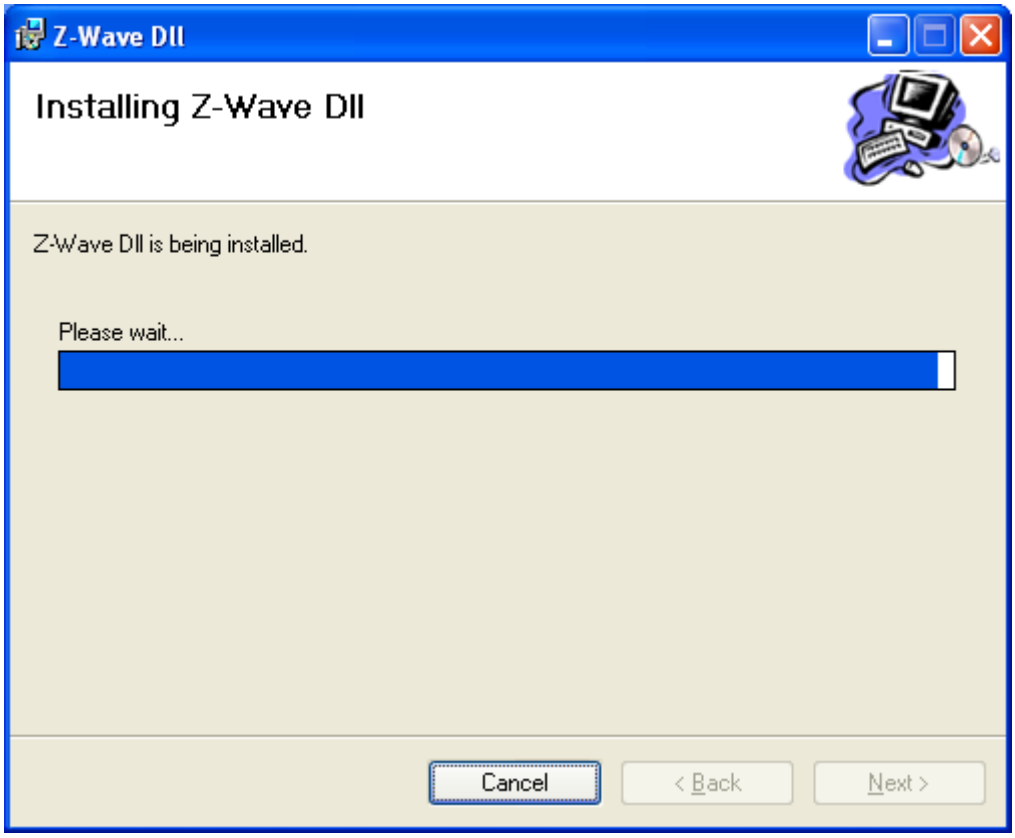

Figure 10. Installation progress

10. Click **Close** to complete the installation.

CONFIDENTIAL

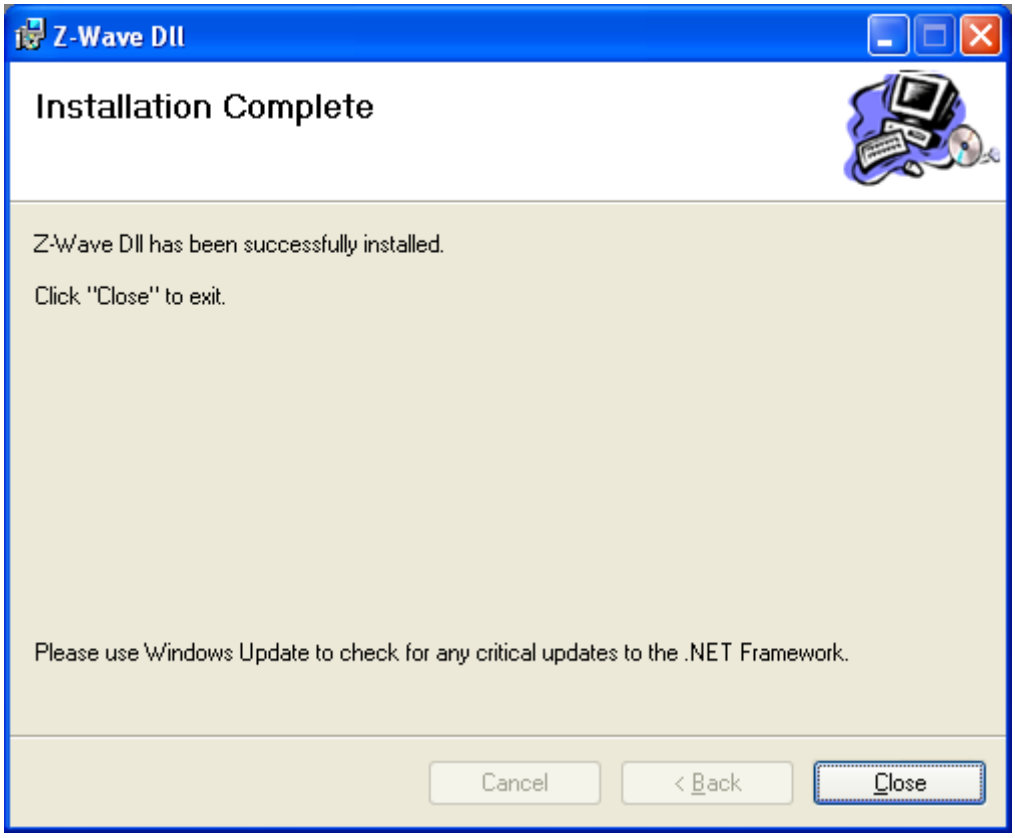

Figure 11. Installation complete

5.5 Remove Z-Wave DLL

You can uninstall Z-Wave DLL from your computer if you no longer use it.

1. Open **Add or Remove Programs** in Control Panel.

   To do it, click **Start**, then click **Control Panel** (in Classical View – click **Start**, then point to **Settings**, and click **Control Panel)**, and then double-click **Add or Remove Programs**.

2. Click the program in the list and then click the **Remove** button. You can sort programs by selecting different options in **Sort by**.

3. Standard confirmation dialog appears. Click **Yes** to continue the removal of the Z-Wave DLL.

4. Z-Wave DLL will be removed without prompting you further.

CONFIDENTIAL

6 CREATION OF A Z-WAVE DLL BASED APPLICATION

This section describes the creation of the new Z-Wave DLL based .NET Windows Application and its customization.

The following steps are examined:

- Create a new Z-Wave DLL based .NET Windows Application project.
- Obtain available Serial Port Interfaces using ZensysFramework library that used Windows Management Instrumentation components (WMI).
- Obtain available Serial Port Interfaces using SerialPortTransport library that implement ITransportLayer interface.
- Implement "Detect Device" functionality.

Attention! The steps below are based on a C# project. However you can use the same steps for a Visual Basic project, except for specifics such as file name extensions and code.

6.1 Create a .NET Windows Application project

In this section, you create a solution with .NET Windows Application project in Visual Studio.

1. Open Visual Studio.
2. In the **File** menu, point to **New**, and then click **Project**.
3. In the New Project dialog box, expand **Visual C#**, and then click **Windows**.
   Under Visual Studio installed templates, select **Windows Application**.
   Select **Create Directory for Solution.**
   In the Name field, type *MyApplication*.
   And finally enter the location for the solution (C:\MyZWave for example).

CONFIDENTIAL

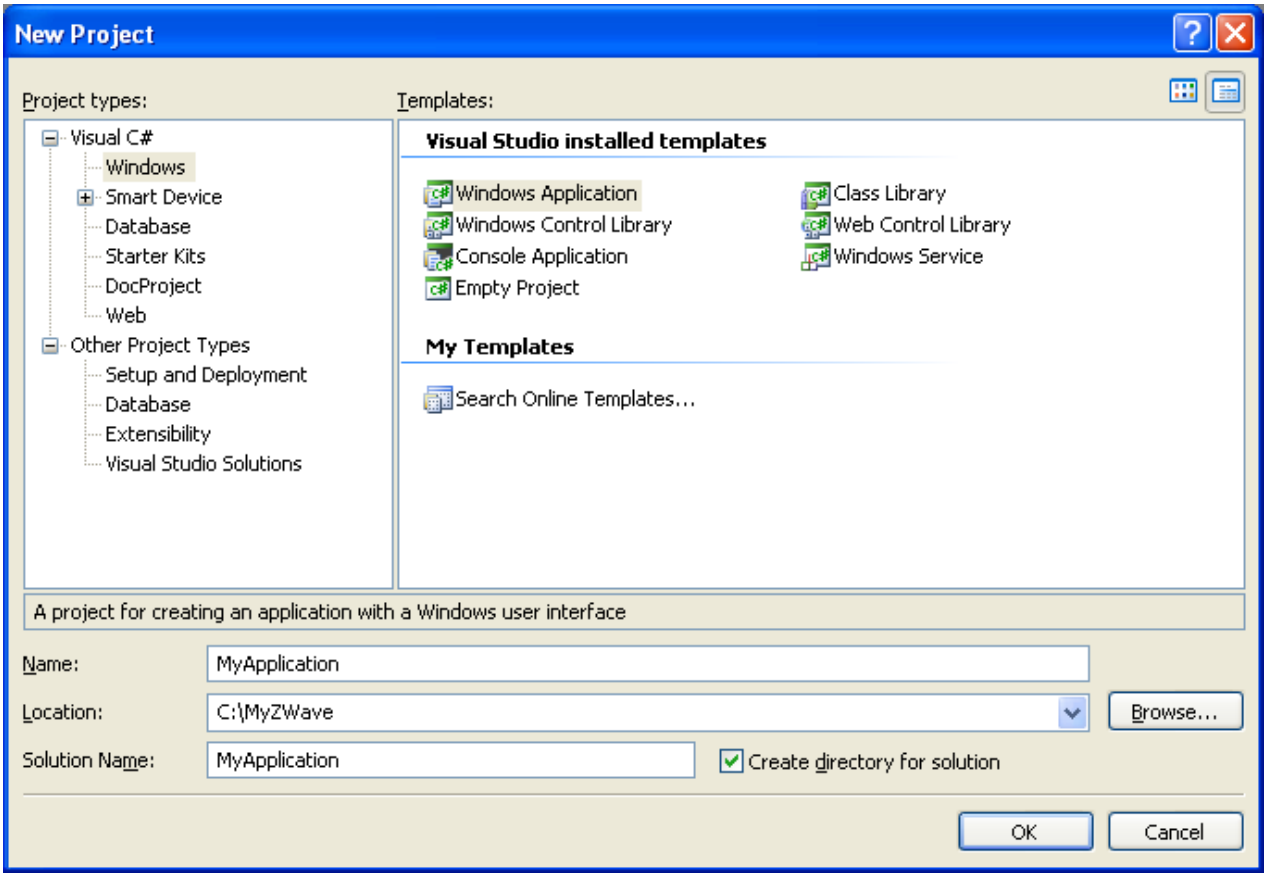

Figure 12. Create a new Visual C# application.

To continue, click **OK**.

6.2 Add references to Z-Wave DLL components

1. Use the **Add Reference** dialog to add references to components at design time.
2. Browse to folder where Z-Wave DLL has been installed.
   By default it is installed into "C:\Program Files\Zensys\Z-Wave Dll\".

CONFIDENTIAL

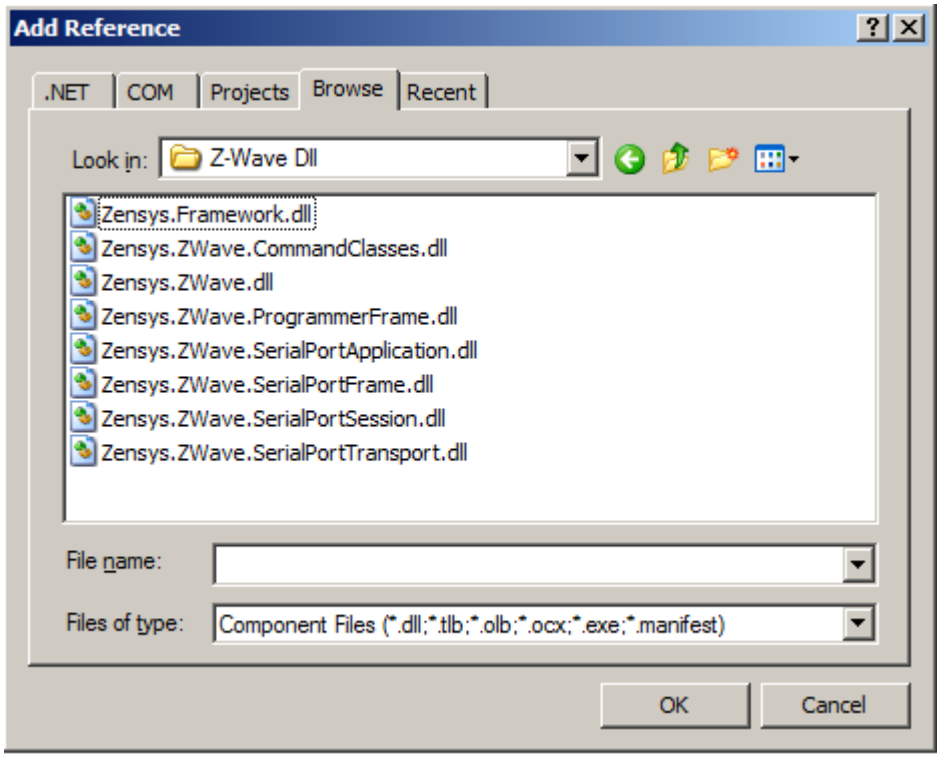

Figure 13. Select all *.dll files in this folder and click OK

3. Select all *.dll files in this folder and click **OK**.

6.3 Obtain available Serial Port Interfaces using ZensysFramework library

1. Add **ListBox** and **Button** controls to the **Form1**.

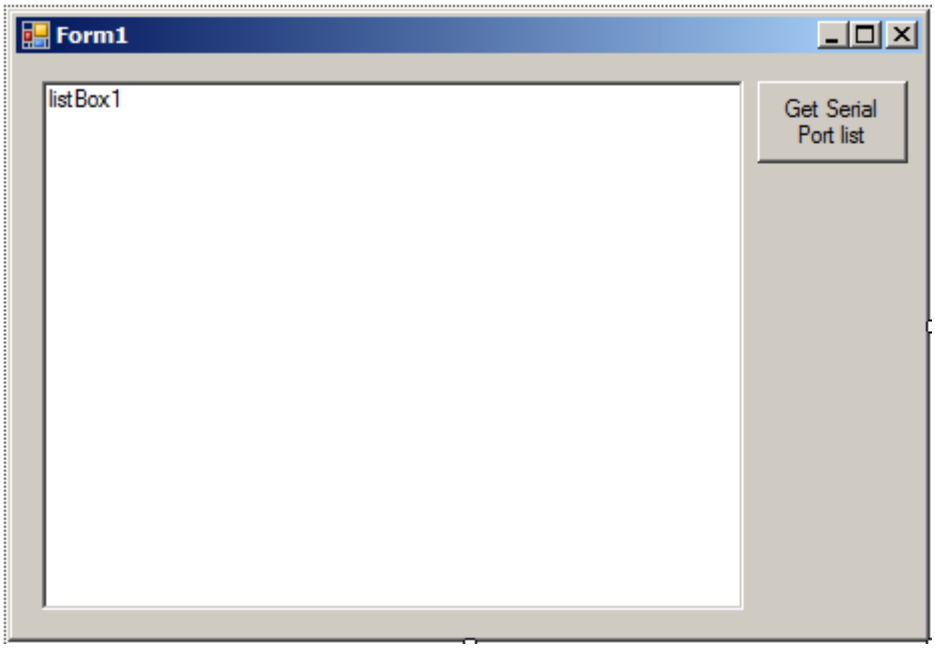

Figure 14. Implement Click **event for** button1 **control**

CONFIDENTIAL

2. Implement **Click** event for **button1** control.

```
private void button1_Click(object sender, EventArgs e)
{
    List<Zensys.Framework.Win32PnPEntityClass> interfaces =
Zensys.Framework.ComputerSystemHardwareHelper.GetWin32PnPEntityClassSerialPortDevices();
    foreach (Zensys.Framework.Win32PnPEntityClass serialPortInfo in interfaces)
    {
        listBox1.Items.Add(serialPortInfo.Caption);
    }
}
```

3. Run application and click **Get Serial Port list** Button.

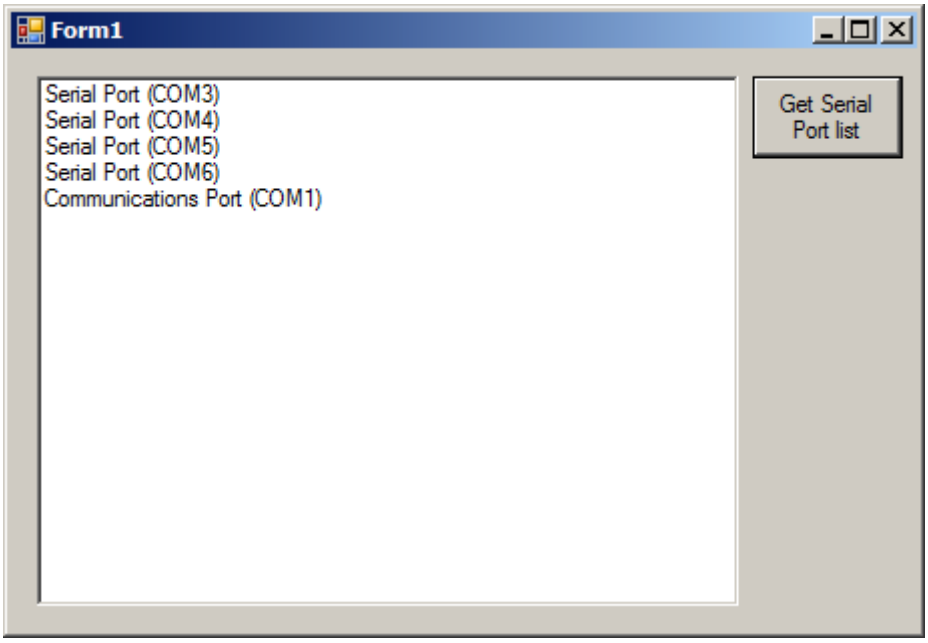

Figure 15. Run application and click Get Serial Port list **Button**

6.4 Obtain available Serial Port Interfaces using SerialPortTransport library.

1. Implement **Click** event for **button1** control.

```
private void button1_Click(object sender, EventArgs e)
{
    Zensys.ZWave.ZWaveManager zwaveManager = new Zensys.ZWave.ZWaveManager();
    zwaveManager.Init(Zensys.ZWave.Enums.LibraryModes.None);
    foreach (string serialPortName in zwaveManager.TransportLayer.GetDeviceNames())
    {
        listBox1.Items.Add(serialPortName);
    }
```


CONFIDENTIAL

}

2. Run application and click **Get Serial Port list** button.

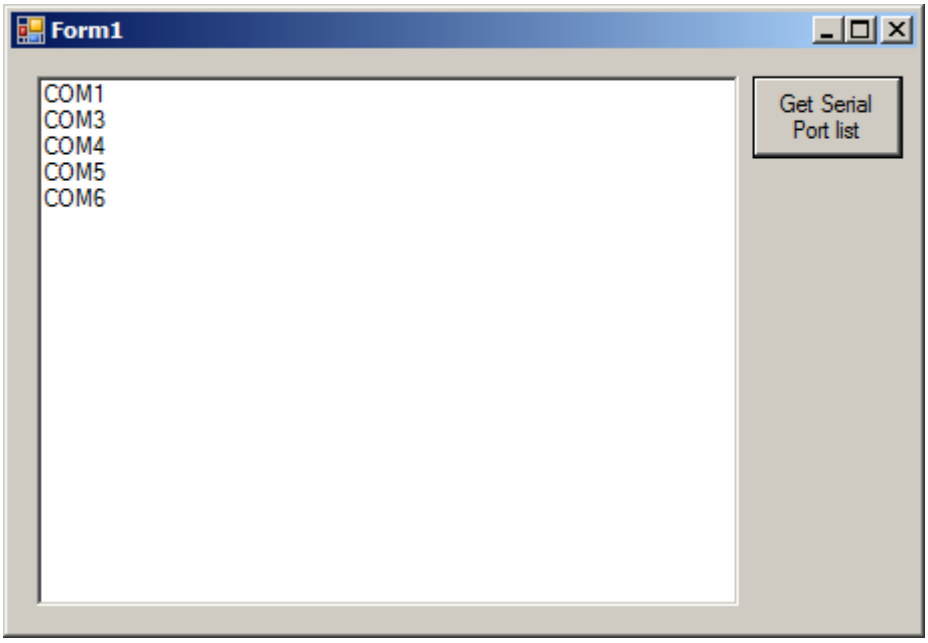

Figure 16. Run application and click Get Serial Port list **button**

6.5 Implement "Detect Device" functionality.

1. Add new **Button** control to the **Form1**.
2. Implement **Click** event for this new **button2** control.

```
private void button2_Click(object sender, EventArgs e)
{
   Zensys.ZWave.ZWaveManager zwaveManager = new Zensys.ZWave.ZWaveManager();
   zwaveManager.Init(Zensys.ZWave.Enums.LibraryModes.PcController);
   Zensys.ZWave.Devices.IController device = zwaveManager.ApplicationLayer.CreateController();
   List<Zensys.Framework.Win32PnPEntityClass>
   interfaces = Zensys.Framework.ComputerSystemHardwareHelper.GetWin32PnPEntityClassSerialPortDevices();

   foreach (Zensys.Framework.Win32PnPEntityClass serialPortInfo in interfaces)
   {
      Zensys.ZWave.Enums.Libraries deviceLibrary = Zensys.ZWave.Enums.Libraries.NoLib;
      try
      {
        device.Open(serialPortInfo.DeviceID);
       device.GetVersion();
       deviceLibrary = device.Version.Library;
```


CONFIDENTIAL

```
    }
    catch (Zensys.ZWave.Exceptions.RequestTimeoutException ex)
    {
       System.Diagnostics.Debug.WriteLine(ex.Message);
      //It means that there is no Device connected to Serial Port.
    }
    finally
    {
      device.Close();
    }
    listBox1.Items.Add(String.Format("{0}: {1}", serialPortInfo.Caption, deviceLibrary.ToString()));
  }
}
```

3. Run application and click on **Detect** button.

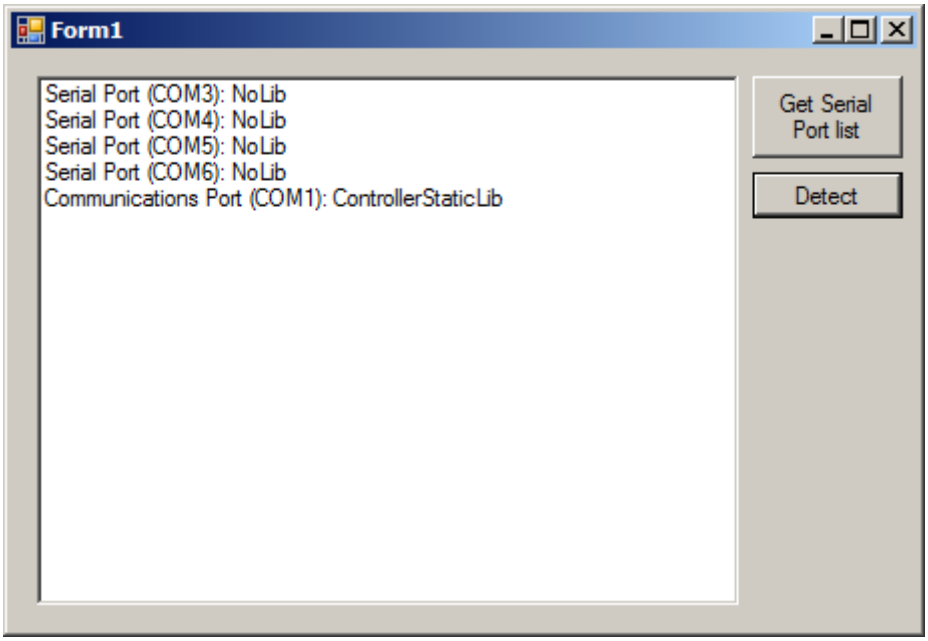

Figure 17. Run application and click on Detect **button**

CONFIDENTIAL

7 REFERENCES

[1] Zensys, INS10245, Instruction, Z-Wave Bridge User Guide.
[2] Zensys, INS10240, Instruction, PC based Controller User Guide.

CONFIDENTIAL