WRxx SDK

Quick Start Guide

Raytech GmbH

Oberebenestrasse 11

CH 5620 Bremgarten

Switzerland

Version 1.00/ZF

Document Review			
Version	**Amendment**	**By**	**Date**
1.00	Initial	Z. Filipovic	17.06.2011

Contents

1 About this Guide

This quick reference guide provides a short introduction to WRxx SDK library. This manual clarifies the various possible applications of the WRxx SDK package, and provides a systematic approach for WRxx customization projects.

2 What is WRxx SDK

WRxx SDK is device management software. It is essentially a software layer (or driver) that resides between the Operating System (OS) IO system, custom Windows Application and WRxx device. WRxx SDK with native Raytech USB Driver for Windows provides the OS with full device functionality, appearing to OS as Raytech USB device. WRxx SDK can also be used without native Raytech USB drivers to control the device over standard RS 232 Serial port.

3 Prerequisites

- WRxx minimum acceptable firmware version 2.78
- PC (Windows XP SP 3, Vista, 7, 2003, 2008 operating systems)
- .NET compatible development kit (e.g. Visual Studio 2010, SharpDevelop 4.x)
- .NET 4.0
- WRxx SDK
- In order to use USB functionality, native WRxx USB Driver must be installed on the system.

4 Installing WRxx native USB Driver

USB Driver installation is not necessary in order to use RS 232 serial port for communication with device.

4.1 Windows XP

Connect the WRxx device with your PC using USB cable. In the Driver Installation Dialog choose a folder containing *WR_xx.INF* file and click “Next” Button. Ignore unsigned driver installation warning.

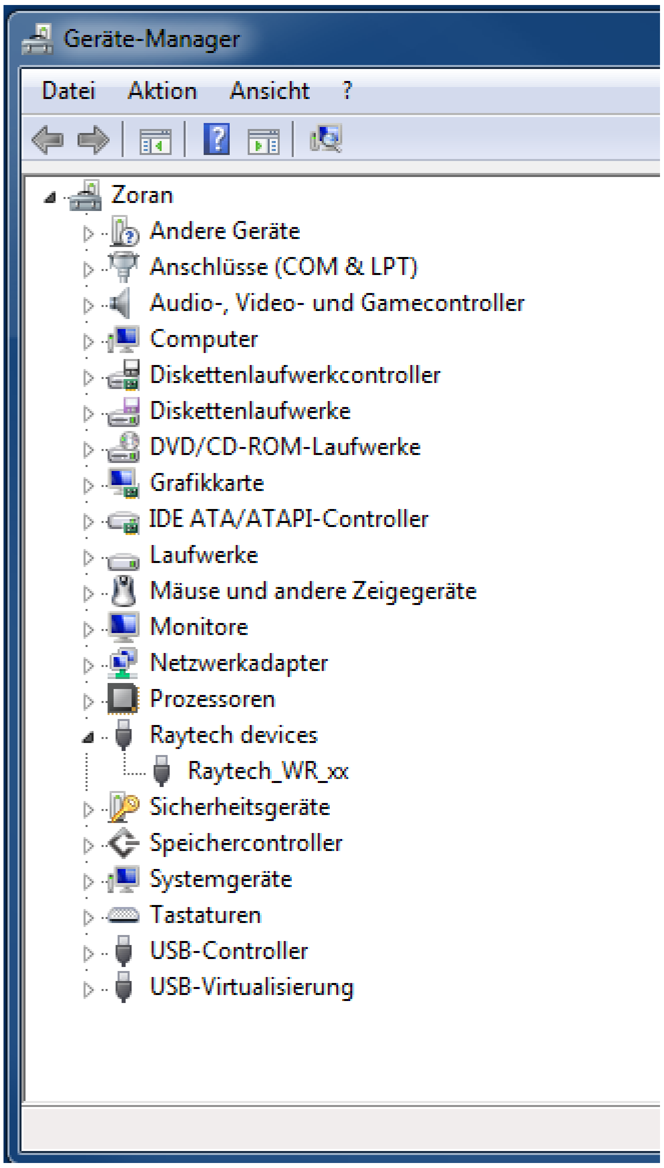
4.2 Windows Vista, 7, 2003 and 2008

Open the folder containing WRxx USB drivers. Double click on:

- *install_x86.exe*, automatic driver installer for 32 bit OS
- *install_x64.exe*, automatic driver installer for 64 bit OS
- *install_ia64.exe*, automatic driver installer for Itanium based systems

Ignore unsigned driver installation warning in order to complete installation.

Windows Device Manager after successfully driver installation. Notice new device-group entry “Raytech devices” with “Raytech_WR_xx” as child.



- Zoran
  - Andere Geräte
  - Anschlüsse (COM & LPT)
  - Audio-, Video- und Gamecontroller
  - Computer
  - Diskettenlaufwerkcontroller
  - Diskettenlaufwerke
  - DVD/CD-ROM-Laufwerke
  - Grafikkarte
  - IDE ATA/ATAPI-Controller
  - Laufwerke
  - Mäuse und andere Zeigegeräte
  - Monitore
  - Netzwerkadapter
  - Prozessoren
  - Raytech devices
    - Raytech_WR_xx
  - Sicherheitsgeräte
  - Speichercontroller
  - Systemgeräte
  - Tastaturen
  - USB-Controller
  - USB-Virtualisierung

NOTE

⇒ In order to use native Raytech USB drivers, WRxx firmware must be reconfigured. This will be achieved by typing service code 2001 into the device setup.

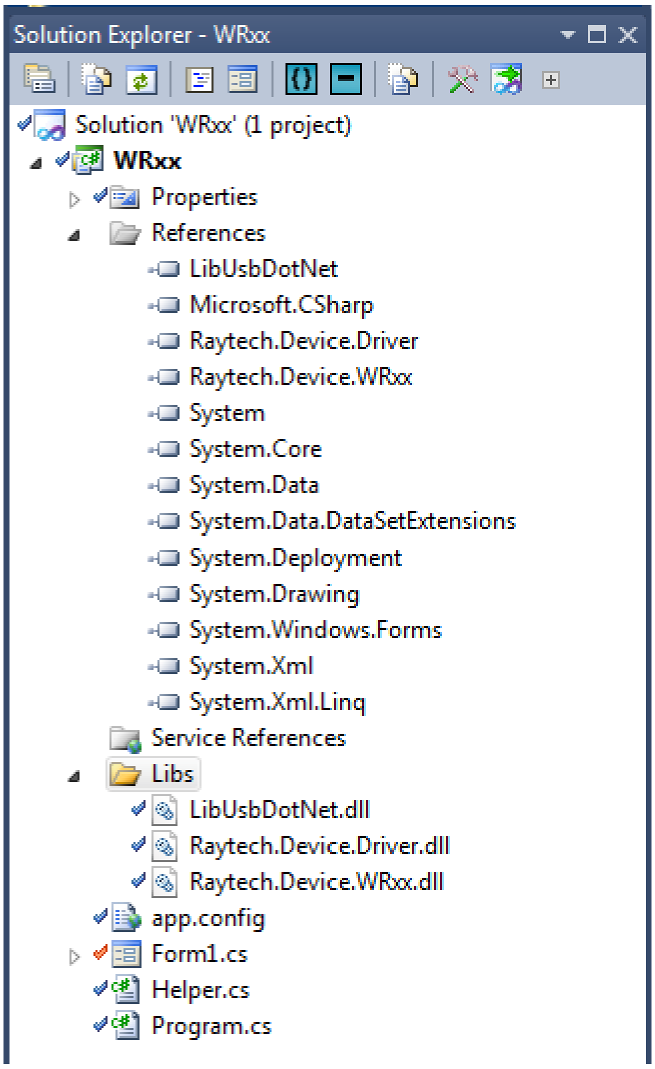
5 Getting started

5.1 Creating project

First of all, create new project in integrated development environment of your choice and add references to the following libraries:

- **LibUsbDotNet.dll** (.NET C# USB wrapper library for WinUSB, LibUsb-Win32 and libusb-1.0)
- **Raytech.Device.Driver.dll** (Base library for WRxx and other Raytech device software)
- **Raytech.Device.WRxx.dll** (Software library for interaction with WRxx)

It is recommended to copy the library files in some folder of the project.



```
Solution Explorer - WRxx
Solution 'WRxx' (1 project)
  WRxx
    Properties
    References
      LibUsbDotNet
      Microsoft.CSharp
      Raytech.Device.Driver
      Raytech.Device.WRxx
      System
      System.Core
      System.Data
      System.Data.DataSetExtensions
      System.Deployment
      System.Drawing
      System.Windows.Forms
      System.Xml
      System.Xml.Linq
    Service References
    Libs
      LibUsbDotNet.dll
      Raytech.Device.Driver.dll
      Raytech.Device.WRxx.dll
    app.config
    Form1.cs
    Helper.cs
    Program.cs
```

6

6.1 Namespaces

In order to use full potential of the SDK, object from following namespaces can be used:

```
using Raytech.Device.WRxx;
using Raytech.Device.Driver.Common;
using Raytech.Device.Driver.USB;
using Raytech.Device.Driver.Helpers;
using Raytech.Device.Driver.InformationHandling;
```

6.2 WRxx Driver instance

There are several ways to use instance driver in your application.

1. Direct(simple)

```
WRxxDevice driver = new WRxxDevice();
```

2. Using IDriverBase interface (recommended):

```
IDriverBase driver = new WRxxDevice();
```

3. As module (plugin) in MEF (Managed Extensibility Framework) based application (advanced):

```
[ImportMany(typeof(IDriver))]
public List<Lazy<IDriver, IDriverMetadata>> DeviceApplications {
  get; private set;
}


private void LoadDeviceControlApplications() {
   try {
      DirectoryCatalog directoryCatalog = new DirectoryCatalog(
                 Environment.CurrentDirectory + @"\Devices");

      this.compositionContainer = new CompositionContainer(directoryCatalog);

      this.compositionContainer.ComposeParts(this);
   } catch (Exception ex) {
        Messenger.Default.Send(new ErrorDialogMessage(ex.Message, null));
   }
}

public IDriverBase GetDriver() {
   return DeviceApplications.Value.DeviceControlDriver;
}
```

6.3 Connect and Disconnect

Serial connection mode:

```
this.driver.SerialPortName = (sender as ComboBox).Text;
```

```
driver.SetSerialCommunicationMode();
driver.Connect();
```

USB connection mode,

```
//USB device selection change event
this.cmbUsbDevice.SelectedIndexChanged += (sender, e) => {

   if (!string.IsNullOrEmpty(this.cmbUsbDevice.Text)) {
      //Find selected USB device
      var query = from d in this.usbDeviceHandler.GetConnectedDevices()
                  where this.cmbUsbDevice.Text == d.GetProductString()
                  select d;

      //Expose USB device to the driver
      if (query.Count() > 0)
         this.driver.UsbDevice = query.First();
   }
};
```

```
driver.SetUsbCommunicationMode();
driver.Connect();
```

Disconnect and reset:

```
driver.Disconnect();
```

6.4 Get Connected USB Devices

Class UsbDeviceHandler is USB information source:

```
UsbDeviceHandler usbDeviceHandler = new UsbDeviceHandler();


var devices =  this.usbDeviceHandler.GetConnectedDevices();
```

React on device plugged in and out:

```
this.usbDeviceHandler.OnUsbNotify += delegate {
   //Do something
};
```

6.5 SendAndReceive method

The *SendCommand* method sends a *command* to the *device* and retrieves the results synchronously.

```
int timeout = 1000;
ExecutionData answer = driver.SendAndReceive("WR", timeout, ReceiveMode.Once);
```

If more than one answer is expected, *ReceiveMode* must be set to *Multiple*.

6.6 Send Method

The *Send* method sends a *command* to the *device* and retrieves no results.

```
driver.Send(“GV“);
```

6.7 Start and Stop measuring

```
driver.StartMeasure<WRxxMeasurementMode>(
        WRxxMeasurementMode.NoLocalResultSave, IntermediateResultDisplay.Hide);

this.driver.StopMeasure();
```

6.8 Retrieving Measuring results, asynchronous events

```
this.driver.OnSendResult += (sender, e) => {
   this.DisplayResults(e.Result as WRxxMeasurement);
};
```

Measurement is specific WRxx object.

6.9 Receive all Data, asynchronous events

Organized as ExecutionData…

```
this.driver.OnSendExecutionData += (sender, e) => {
   // implementation
};
```

Or row data as strings…

```
this.driver.OnSendDataRow += (sender, e) => {
   //implementation
};
```

7 Appendix

For the full reference list of the WRxx commands, please refer to the "*Command Set 90104-2.09 Command Set WR-xx.pdf*" delivered with this package, or call the Raytech GmbH technical support.