EcgSoft®

Software Developer's Guide to RestEcg

Innovative ECG Software

www.ecg-soft.com - e-mail: info@ecg-soft.com

This page is intentionally left blank

Contents

1. Introduction 4
2. About the Guide 4
3. RestEcg Open Interface Block Diagram 5
4. Creating an "OemDeviceLib.dll" for your ECG device 6
5. Sample Projects.....8
6. AcqDevice Abstract Class 9

.

1. Introduction

RestEcg is a real-time 12 channel resting ECG software application.

It enables you to monitor, record, review, edit, filter and print ECG signals acquired from any ECG device that implements the "Open Interface Access" of the RestEcg software.

"Open Interface Access" is the key to the universal usage of the RestEcg with various proprietary ECG devices.

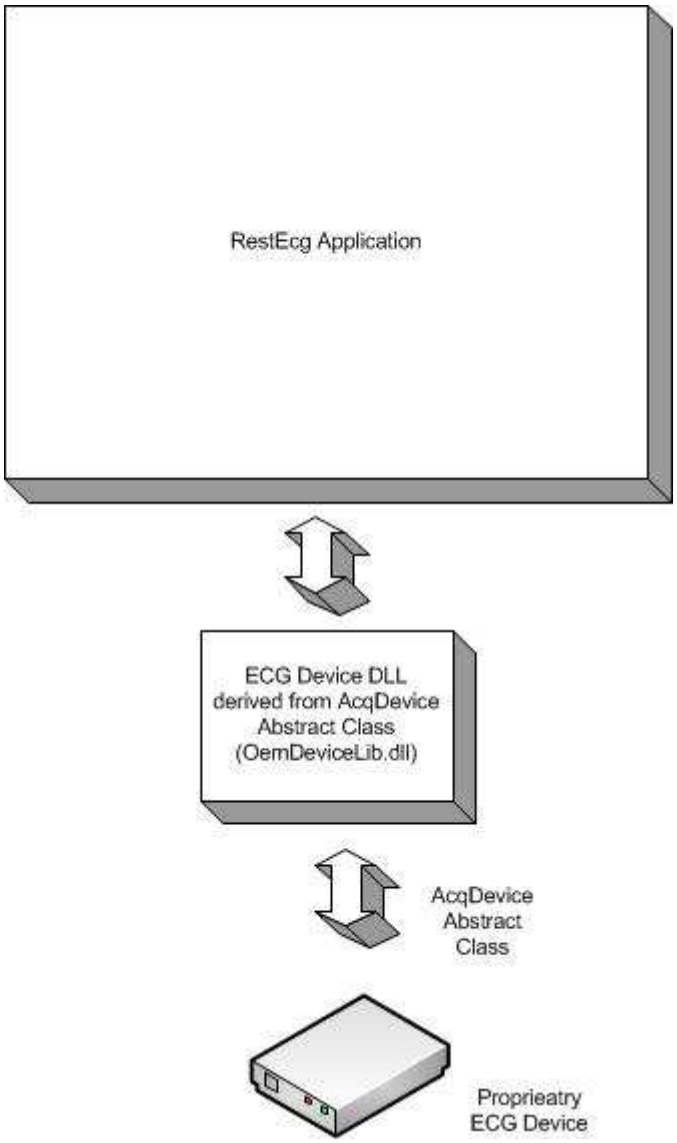
Real-time ECG signals from proprietary ECG devices can be acquired if a suitable plug in DLL (Dynamically Linked Library) is created conforming to "Open Interface Access".

2. About the Guide

This guide will help you create a suitable DLL software for your proprietary ECG device with minimal effort and time on your side.

3. RestEcg Open Interface Block Diagram

As illustrated in the diagram below, all data transfers to and from RestEcg is realized over the "OemDeviceLib.dll". This "dll" is inherited from the "AcqDevice" abstract class and implements the methods of the abstract class.



4. Creating an "OemDeviceLib.dll" for your ECG device

Each ECG lead data sample is a two bytes signed integer value.

RestEcg software acquires the data in a packet of 4 samples at a time.

Only 8 independent leads are needed (I,II,V1 to V6). Remaining dependent leads (III, aVR, aVL, aVF) are derived by calculation in RestEcg software.

Structure of a single packet declared in C# language syntax is as follows:

```
// first sample of  the current data packet
short int     I;          //  2 bytes signed integer
short int     II;         //  2 bytes signed integer
short int     V1;         //  2 bytes signed integer
short int     V2;         //  2 bytes signed integer
short int     V3;         //  2 bytes signed integer
short int     V4;         //  2 bytes signed integer
short int     V5;         //  2 bytes signed integer
short int     V6;         //  2 bytes signed integer

// second sample of  the current data packet
short int     I;          //  2 bytes signed integer
short int     II;         //  2 bytes signed integer
short int     V1;         //  2 bytes signed integer
short int     V2;         //  2 bytes signed integer
short int     V3;         //  2 bytes signed integer
short int     V4;         //  2 bytes signed integer
short int     V5;         //  2 bytes signed integer
short int     V6;         //  2 bytes signed integer

// third sample of  the current data packet
short int     I;          //  2 bytes signed integer
short int     II;         //  2 bytes signed integer
short int     V1;         //  2 bytes signed integer
short int     V2;         //  2 bytes signed integer
short int     V3;         //  2 bytes signed integer
short int     V4;         //  2 bytes signed integer
short int     V5;         //  2 bytes signed integer
short int     V6;         //  2 bytes signed integer
```



```
// fourth sample of  the current data packet
short int     I;          //  2 bytes signed integer
short int     II;         //  2 bytes signed integer
short int     V1;         //  2 bytes signed integer
short int     V2;         //  2 bytes signed integer
short int     V3;         //  2 bytes signed integer
short int     V4;         //  2 bytes signed integer
short int     V5;         //  2 bytes signed integer
short int     V6;         //  2 bytes signed integer
```

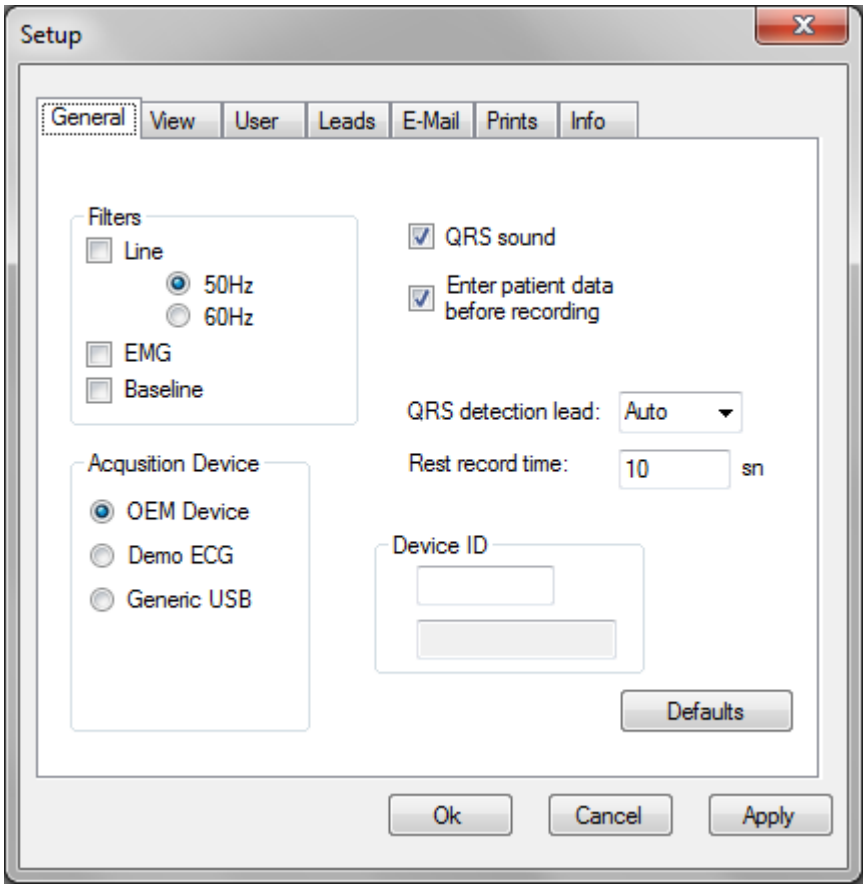
RestEcg assumes that a 1 mv ECG signal at the input of the patient cable is represented by the number 5464. (ADU value)

If your device represents 1 mv by X value, then for scaling purposes, you need to multiply your data by the scaling factor (5464/X) for proper amplitude calibration.

Default sampling rate of the RestEcg is 1000 samples per second. If your device has a different sampling rate, you need to downsize or upsize your sampling rate accordingly.

To create the "OemDeviceLib.dll", you need to implement these steps in Visual Studio:

1. Create a new class library solution and name both the solution and project as OemDeviceLib.
2. Create a new class which is named "OemDevice" and add "`using EcgSoftLib;`" statement to use EcgsoftLib.dll in the project.
3. In the Solution Explorer -> OemDeviceLib->references, add "EcgsoftLib.dll" reference to the project. This "dll" resides in the application folder of RestEcg at the "..\ProgramFiles\EcgSoft\RestEcg" path.
4. Let OemDevice class Inherit from the "AcqDevice" abstract class as "`public class OemDevice : AcqDevice`"
5. Implement the methods of AcqDevice abstract class in the OemDevice as related to your ECG device. Only two methods, **InitDevice** and **GetData** methods needs to be implemented. Rest is optional .Methods which must be implemented by the derived class are named abstract. Methods which already have default implementations in base class are named virtual.
6. Compile the solution and copy the "OemDeviceLib.dll" into the application folder of RestEcg at "..\ProgramFiles\EcgSoft\RestEcg"
7. Finally, as shown below, select from RestEcg -> Setup-> General-> Acqusition Device->OEM Device option so that RestEcg would use your proprietary ECG device implementation as the default acquisition device.



4. Sample Projects

There are two sample projects that can be downloaded from the www.ecg-soft.com\restecg web site which illustrates the creation of "OemDeviceLib.dll".

Project 1 Folder: OemDeviceLib-demo.bin-Net4.0-VS2010-Sample-Solution

In this project you will be able acquire real-time data from a demo.bin ECG file . This binary ECG file will act as an OEM ECG device sending data to RestEcg application. Demo.bin ECG file should be copied to application executable path.

Project 2 Folder: OemDeviceLib-generic-usb-Net4.0-VS2010-Sample-Solution

In this project you will be able acquire real-time data from an OEM ECG device via the USB port.

Uses LibUsbDotNet.dll library which is a wrapper for the Open Source general purpose "libusb-win32" USB driver . It is assumed that firmware side is implemented on the ECG device.

5. AcqDevice Abstract Class

Implementing the abstract class enables RestEcg application to acquire data from a proprietary ECG device in real-time. Below is the C# code for the abstract class:

Only two methods **InitDevice** and **GetData** methods needs to be implemented. Rest is optional.

```
using System;

namespace EcgSoftLib
{


	/// <summary>
	/// AcqDevice is an abstract base class which defines standart interface for
	/// ECG signal acqusition devices
	/// such as USB,demo file, etc.
	/// </summary>
	/// <remarks>
	/// Methods which must be implemented by the derived class are named
	/// abstract.
	/// Methods which already have default implementations in base class are
	/// named virtual.
	/// <para>
	///  use override in the derived class to implement these methods  </para>
	/// </remarks>
	///
	public abstract class AcqDevice
	{
		/// <summary>
		/// Constructor
		/// </summary>
		public AcqDevice()
		{
		}

		/// <summary> Find number of devices attached and initialize the first
		///  device
		/// </summary>
		/// <returns>number of devices attached</returns>
		///
		abstract public int InitDevice();


		/// <summary>Get signal data</summary>
		/// <param name="sampleSize">number of samples received </param>
		/// <param name="buffer">array to store data</param>
		/// <returns>true if succesfull</returns>
		///
		abstract public bool GetData(ref int sampleSize, short[] buffer);


		/// <summary> Set sampling rate in Hz </summary>
		/// <param name="freq">frequency in Hz</param>
		/// <returns>true if succesfull</returns>
		///
		public virtual bool SetSamplingRate(int freq)
		{
			return true;
		}
```

```
        /// <summary>sets number of channels to be converted </summary>
        /// <param name="chn">number of channels</param>
        /// <returns>true if succesfull</returns>
        ///
        public virtual bool SetNumberOfChannels(int chn)
        {
              return true;
        }

        /// <summary>sets the data buffer size at the device side </summary>
        /// <param name="size">buffer size in bytes</param>
        /// <returns>true if succesfull</returns>
        ///
        public virtual bool SetDeviceBufferSize(int size)
        {
              return true;
        }

        /// <summary>gets the 32 bit serial number of device</summary>
        /// <returns>32 bit serial number</returns>
        ///
        public virtual int GetSerialNo()
        {
              return -1;         // i.e no serial number by default
        }


        /// <summary>gets error code</summary>
        /// <param name="data">single byte error code</param>
        /// <returns>true if succesfull</returns>
        ///
        public virtual bool GetErrorCode(ref byte data)
        {
              return true;
        }

        /// <summary>turn on device</summary>
        /// <returns>true if succesfull</returns>
        ///
        public virtual bool TurnOn()
        {
              return true;
        }


        /// <summary>turn off device or part of device(i.e isolated
        ///section)
        /// </summary>
        /// <returns>true if succesfull</returns>
        ///
        public virtual bool TurnOff()
        {
              return true;
        }

    }

}
```