Manual

Scemtec STX/ETX .NET Library v0.3

project name **Scemtec STX/ETX .NET Library**		version **0.3**	
build date **05.02.14**	author	last change **24.11.15**	author
last print:		no. of pages **11**	
Path: E:\scemtec.net\Doc\Manual\scemtec .NET Library.odt			
project manager	development manager	Managing director scemtec	customer
<project manager>	Burkhard Bröhl	Jürgen Kalbitzer	<customer>

Important notice:

As with all electronic systems, the electronic circuitry on its own is not approved for safety-critical applications.

Contents

1 General....3
1.1 Package content....3
1.2 Prerequisites....3
2 Getting started....4
2.1 Including the library in your project....4
2.2 Reader discovery....5
2.2.1 Discover readers connected to a RS232 port....5
2.2.2 Discover readers connected to an USB port....5
2.2.3 Discover readers connected via TCP/IP....5
2.2.4 Discover Bluetooth devices....5
2.2.5 Discover all readers....5
2.3 Opening a connection to the reader....6
2.3.1 Connecting to a device found by the search methods....6
2.3.2 Connecting to a serial port....6
2.3.3 Connecting to USB devices....6
2.3.4 Connecting via TCP/IP....6
2.4 Reading Transponder data....6
2.4.1 Read a single block from every ISO15693 transponder in the field....7
2.4.2 Read a single word from a EM4205 transponder....7
2.4.3 Retrieve transponder IDs using the RealTimeInventory....7
3 Frequently Asked Questions....9
3.1 Operating Environment....9
3.1.1 What are the minimum system requirements?....9
3.1.2 Does this library run on different operating systems?....9
3.1.3 Which version of Microsoft .NET Framework must be installed?....9
3.2 Supported Devices and Functions....9
3.2.1 Does this library support the reader device X?....9
3.2.2 Why does this library don't support function X?....9
4 Related Documents....10
5 Document History....11

1 General

This document describes the scemtec .NET library and its general usage. This library simplifies the access of scemtecs RFID devices by handling the communication between the host and the device. This includes building the low level commands as well as the handling of the underlying connection (like RS232, USB or TCP/IP).

The developer using the library is presented an object oriented API to include in own C# or VB.NET projects.

This Document relates to the development version of the scemtec .NET library. In this early stage of the software development, the library may contain severe bugs and lack of functionality.
Furthermore, the information given in this document may change without notice.

1.1 Package content

Together with this document, you should have received:

- The library itself (`scemtec.dll`)
- The XML Documentation file for Visual Studio (`scemtec.xml`)
- The compiled Help file containing the API declaration (`scemtec.chm`)

Future releases of the library may include:

- An Visual Studio example project using the library

1.2 Prerequisites

This document assumes that the reader is familiar with the C# language and the .NET Framework. Despite the examples given in this document are written in C#, a developer using VisualBasic.NET will easily adopt the examples to VB source.

The Host which is used for development should have the following software installed to successfully use the library:

- Microsoft® Windows Vista® or newer[1]
- Microsoft® Visual Studio® 2012 or newer

1 Microsoft, Windows, Windows Vista, and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

2 Getting started

You can include this library in your existing project, or you can start by using the example project delivered with the library and expand it to your needs.

2.1 Including the library in your project

To include the library in your project, the files `scemtec.dll` and `scemtec.xml` should be copied in your project folder. In Visual Studio, add a reference to the library.

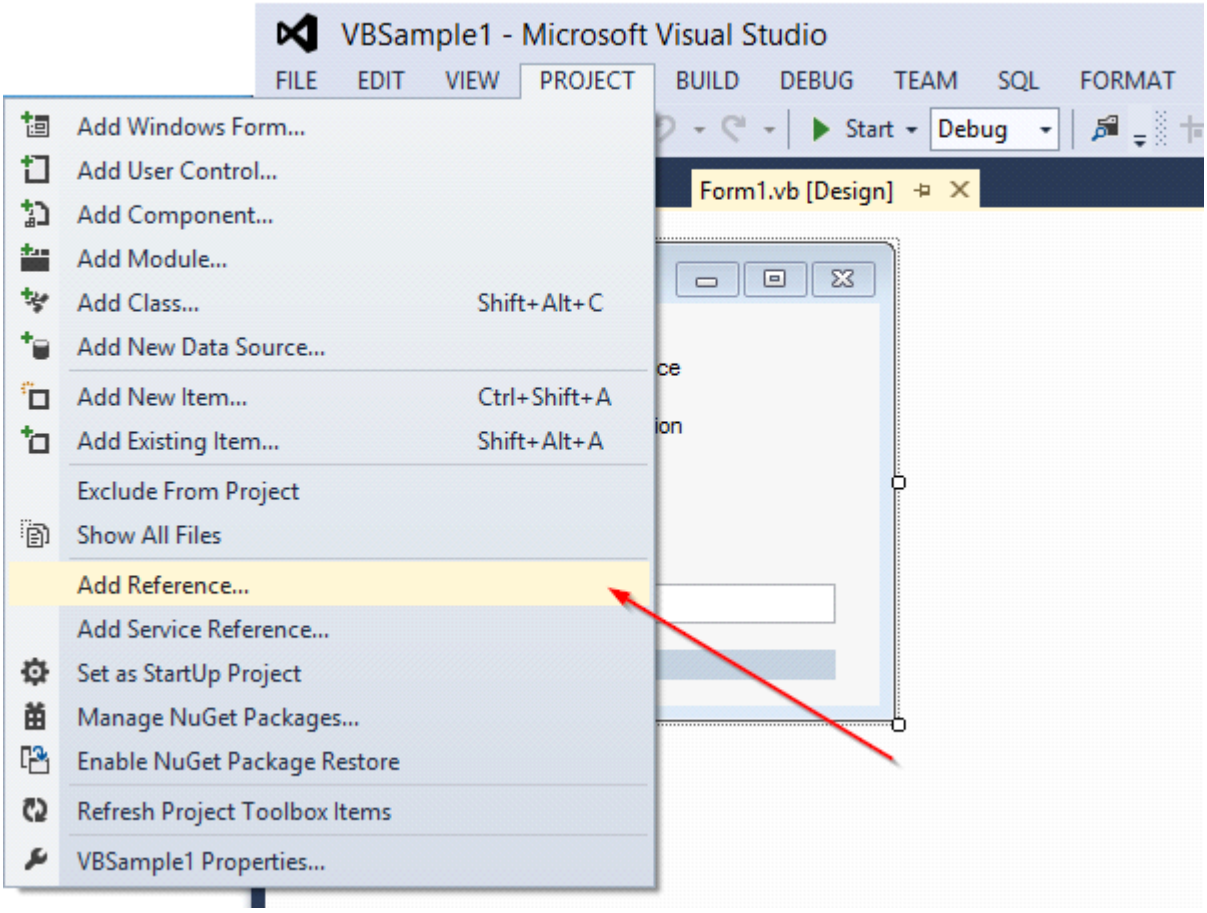

figure 1: Adding a reference to the project

Every source file which uses the library can include a `using` directive to allow the use of types in the `Scemtec` namespace without qualifying the use of a type in that namespace

```
using Scemtec;
```

2.2 Reader discovery

2.2.1 Discover readers connected to a RS232 port

The method `GetSerialPorts()` returns all available serial ports. Since this method does not verify if a device is connected, all virtual serial ports like USB-to-serial adapters or some Bluetooth devices are detected by this function as well.

```
Device[] devices = Reader.GetSerialPorts(); // Get all serial ports
```

2.2.2 Discover readers connected to an USB port

To find a device which is connected via USB, the library implements a static method which returns a list of all scemtec USB devices found.

```
Device[] devices = Reader.GetUsbPorts(); // Get all scemtec USB devices
```

2.2.3 Discover readers connected via TCP/IP

The method `GetTcpIpPorts()` tries to discover devices connected via a TCP/IP network. The discovery uses multicasts which may be blocked by a firewall on the host computer.

```
Device[] devices = Reader.GetTcpIpPorts(); // Get all scemtec TCP/IP devices
```

2.2.4 Discover Bluetooth devices

To discover Bluetooth devices, the method `GetBluetoothPorts()` can be used. By default, it will return all outgoing ports associated with paired scemtec Bluetooth devices. The discovery and pairing with the Bluetooth devices has to be done by the user using the Windows Bluetooth Environment.

```
Device[] devices = Reader.GetBluetoothPorts(); // Get all paired Bluetooth devices
```

2.2.5 Discover all readers

The `GetPorts()` method combines all previous method to get a list of all scemtec devices, regardless of the connection.

```
Device[] devices = Reader.GetPorts(); // Get all scemtec devices
```

2.3 Opening a connection to the reader

2.3.1 Connecting to a device found by the search methods

If one of the discovery method was used, the items of the device list can be passed to the constructor to initiate a connection to the device.

```
Reader reader = new Reader(devices[0]); // Connect to first device found by the GetPorts()
method
```

2.3.2 Connecting to a serial port

In case the reader device is connected to a serial port, a new instance if the reader class can be created by calling the constructor with the name of the port as argument. The port can be a physical serial RS232 port or a virtual serial port used by USB or Bluetooth devices.

```
Reader reader = new Reader("COM1"); // Connect to COM1
```

2.3.3 Connecting to USB devices

The constructor of the reader class without arguments will try to establish a connection to a reader connected via USB. If no device could be found, the constructor will throw an `IOException`.

```
Reader reader = new Reader(); // Try to connect to first USB-Reader
```

```
Reader reader = new Reader("COM3"); // Connect to COM3
```

2.3.4 Connecting via TCP/IP

To connect to a reader using the TCP/IP protocol, the readers IP address can be passed to the constructor of the reader class. If the device has a name which can be resolved to an IP address, the name can be used instead of the address.

```
Reader reader = new Reader("192.168.1.101"); // Connect to given IP address
```

```
Reader reader = new Reader("reader.localdomain"); // Connect to given hostname
```

2.4 Reading Transponder data

The methods to access transponders are organized in the scheme `<ReaderObject>.<TransponderType>.<method>`.

The examples given in this section use only a small subset of the methods provided by the library.

See the online tooltip documentation inside Visual Studio or the included help file `scemtec.chm` to explore more classes and methods.

2.4.1 Read a single block from every ISO15693 transponder in the field

To get every transponder in the field, the method `CreateAndGetInventory` should be used. It will return an instance of the `Inventory` class, which implements the `IEnumerable` interface. Using the methods exposed by this interface, it is possible to access all transponders in a `foreach` loop. Inside the loop, the program will read two blocks of the selected transponder and write its content to the console.

```
Inventory inv = reader.Iso15693.CreateAndGetInventory();
foreach (Iso15693Tag tag in inv) {
  Console.WriteLine(tag);
  ulong[] blocks = tag.ReadBlocks(0, 2); // Read Block 0 and 1 from transponder
  Console.WriteLine(blocks[0].ToString("X8")); // Write content of block 0
  Console.WriteLine(blocks[1].ToString("X8")); // Write content of block 1
}
```

2.4.2 Read a single word from a EM4205 transponder

This example will try to read a single word from a EM4205 transponder and write it to the console. Please keep in mind that if the operation fails, the `Read` function will throw an `StxEtxException` which should be caught by the application.

```
try {
  Console.Write("Read Word 1: ");
  ulong data = reader.Em4205.Read(1); // Read Word 1 (ID) from transponder
  Console.WriteLine(data.ToString("X8"));
} catch (StxEtxException e) {
  Console.WriteLine("Failed (" + e.Message + ")");
}
```

2.4.3 Retrieve transponder IDs using the RealTimeInventory

In contrast to the default Inventory functions, the RealTimeInventory function raises an event as soon as a transponder is detected. Additionally, a further event is raised after every complete inventory cycle.

```
void StartRealTimeInventory() {
  reader.RealTimeInventoryFinished += reader_RealTimeInventoryFinished;
  reader.RealTimeInventoryTag += reader_RealTimeInventoryTag;
  reader.Iso15693.StartRealTimeInventory();
}
```

The RealTimeInventory will try to detect transponders until it is stopped by the program. During

this time, it is not possible to use any other method to access the reader.

```
void StopRealTimeInventory() {
  reader.Iso15693.StopRealTimeInventory();
}
```

The `RealTimeInventoryTag` event is raised on every detected transponder. An instance of the `Tag` class is passed to the event handler. The `RealTimeInventoryFinished` event is raised after every cycle.

```
void reader_RealTimeInventoryTag(Tag tag) {
  // TODO...
}
```

```
void reader_RealTimeInventoryFinished(int count) {
  // TODO...
}
```

During a RealTimeInventory, it is not possible to access the reader using other functions like Read or Write functions. If your applications needs to access transponders, you should consider using the default Inventory functions like `CreateAndGetInventory`.

3 Frequently Asked Questions

3.1 Operating Environment

3.1.1 What are the minimum system requirements?

Windows XP, Windows Vista, Windows 7, Windows 8
Microsoft .NET Framework 4.0 or 4.5

3.1.2 Does this library run on different operating systems?

Though not officially supported by scemtec, the library is reported to work using the mono framework (http://www.mono-project.org) under Linux based operating systems. Ports can be specified by using the device node name (e.g. ”`/dev/ttyS0`”, “`/dev/ttyACM0`”, ...)

3.1.3 Which version of Microsoft .NET Framework must be installed?

This library can be used starting with Version 4.0 up to the current version.

3.2 Supported Devices and Functions

3.2.1 Does this library support the reader device X?

This library is developed to support the current devices offered by scemtec Transponder Technology GmbH. Older devices may be supported by this library and unsupported functions can be accessed using the generic `Reader.StxEtxRequest()` method.

3.2.2 Why does this library don't support function X?

The current version of this library supports only a small subset of the functionality provided by the scemtec Transponder Technology GmbH devices. Most functions, including most of the transponder access functions, will be implemented in a future version of this library. However, it is possible to use every STX/ETX-Protocol function using the generic `Reader.StxEtxRequest()` method.

4 Related Documents

[STXETX]

STX/ETX Protocol description

Scemtec's STX/ETX Protocol description is distributed with every Reader on the product CD

5 Document History

Version	Date	Changed by	Description
0.1	2014-02-05	C. Pauls	– Initial Version
0.2	2015-04-02	C. Pauls	– Update to library version 0.4
0.3	2015-11-24	C. Pauls	– Update to library version 0.6 – RealTimeInventory added. – Bluetooth Discovery added.
			–
			–