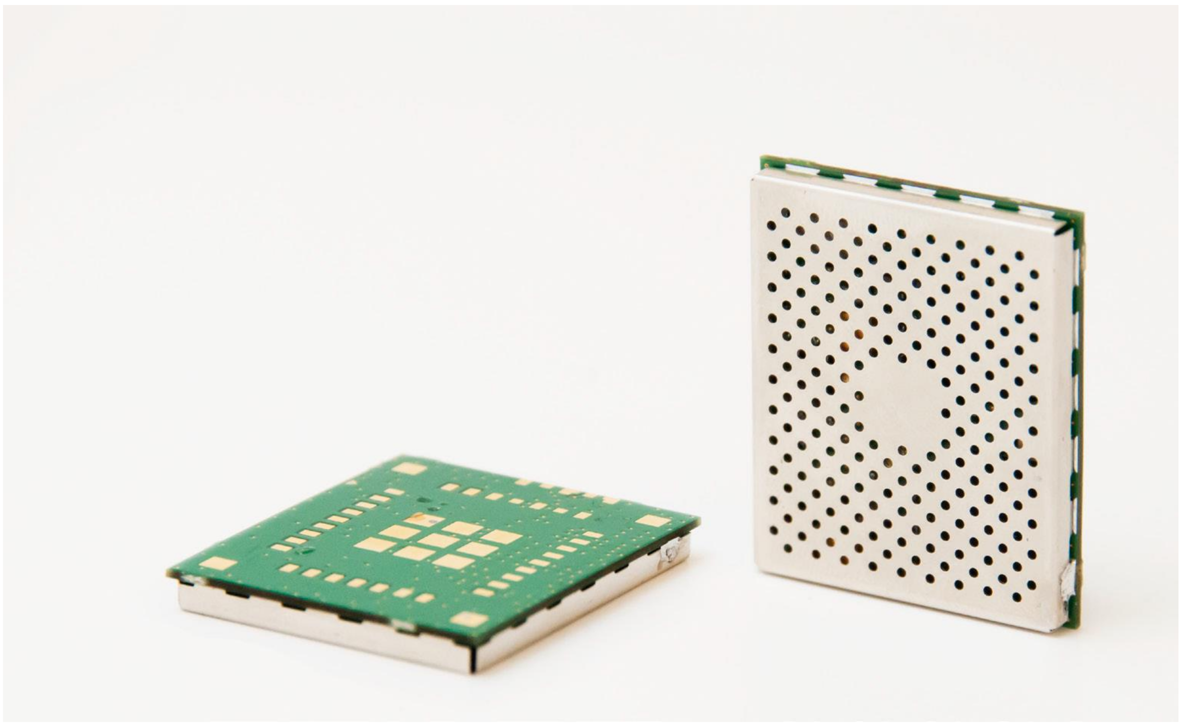
Programming host applications for NUR Module using .NET languages

Getting Started Guide

Contents

Overview

NUR-05W is a compact UHF RFID reader/writer module. Its small footprint, SMD, low power consumption and high RF output makes it an outstanding choice for everything from simple mobile readers to port readers with multiple antennas.



NUR-05W is compatible with the EPC C1G2 (ISO18000-6C) standard. The module meets the requirements of ETSI, FCC and IC radio regulations. NUR-05W is also compatible with the DRM (dense reader mode) requirements.

This document describes how to create .NET applications for NUR module. Code samples are collected to SDK packet and Visual Studio 2005 solution file is located in `../Samples/SimpleDotNetSamples` folder.

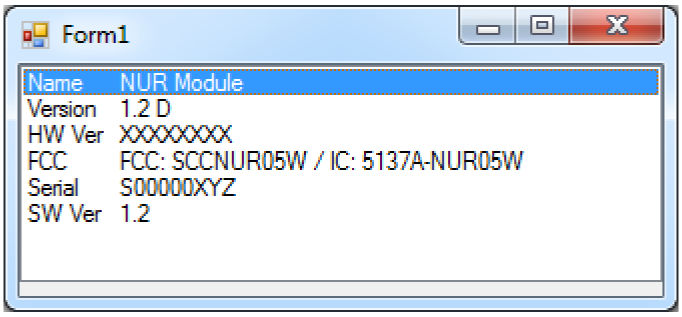
If you are using newer version of Visual Studio than 2005, you can convert it to newer one.
Sample code offers examples of most used functionalities like "Inventory" in Visual Basic and C#.

1.1 NurApi .NET

NurApiDotNet.dll is intended to Windows .NET based RFID applications.
NurApiDotNetWCE.dll is intended to mobile devices provided with Windows CE 6.0 operating system.
NURAPI.dll includes low-level functionalities.

NUR API .NET consist easy-to-use functionalities for creating sophisticate RFID applications for NUR module.

1.1.1 Hello NUR Module



```
Form1
Name     NUR Module
Version  1.2 D
HW Ver   XXXXXXXX
FCC      FCC: SCCNUR05W / IC: 5137A-NUR05W
Serial   S00000XYZ
SW Ver   1.2
```

This is simple VB.NET application just reading NUR module info.

Code: (VB)

```
Imports NurApiDotNet

'Hello NUR is simple sample which connects to Sampo S1 reader via USB and acquire reader
information and shows it in the ListBox.
'------------------------------------------
'Guidelines creating NurApiDotNet projects:
'-Add reference to NurApiDotNet.dll
'-Set target platform to x86
'-NURAPI.dll must be in same folder than executable. Add it in your project and set
property "Copy to Output directory" to "Copy always"
'-Use NurApi commands inside Try...Catch structure. See NurApi documentation which
methods throws exceptions.

Public Class Form1
    'This is handle to NUR RFID Reader. Use it always inside try..catch structure in
case of exceptions.
    Dim hNur As NurApi

    Public Sub New()

        ' This call is required by the Windows Form Designer.
        InitializeComponent()
        hNur = New NurApi()               'Nur Api handle

        'When NUR module connected via USB, this function finds it and connects
automagically...
        hNur.SetUsbAutoConnect(True)

        Dim readerInfo As NurApi.ReaderInfo
        Try 'GetReaderInfo from module
            readerInfo = hNur.GetReaderInfo()
            'Show results in Listbox
            ListBox1.Items.Add("Name" & vbTab & readerInfo.name)
            ListBox1.Items.Add("Version" & vbTab & readerInfo.GetVersionString)
            ListBox1.Items.Add("HW Ver" & vbTab & readerInfo.hwVersion)
            ListBox1.Items.Add("FCC    " & vbTab & readerInfo.fccId)
            ListBox1.Items.Add("Serial" & vbTab & readerInfo.serial)
            ListBox1.Items.Add("SW Ver" & vbTab & readerInfo.swVerMajor & "." &
readerInfo.swVerMinor)

        Catch ex As Exception 'Handle error by show error message in Listbox
            ListBox1.Items.Add("Error: GetReaderInfo:" & ex.Message)
        End Try
    End Sub
End Class
```

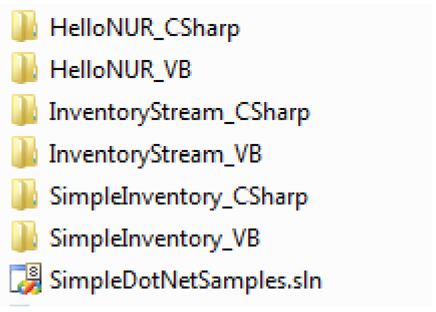
2 Setting up programming environment

Nordic ID provide “NUR API SDK” software packet including all software components, documents, samples etc. for starting programming for NUR RFID reader module.

2.1.1 Needed equipment

1. Visual Studio 2005 or higher version.
   You can also use Express version of Visual Studio tools provided by Microsoft.
   http://www.microsoft.com/visualstudio/en-us/products/2010-editions/express
2. Evaluation Board of NUR module or Sampo S1
3. NURAPI SDK including sample programs
4. NurApiDotNet documentation (located in docs folder)

2.1.2 Running sample program


HelloNUR_CSharp
HelloNUR_VB
InventoryStream_CSharp
InventoryStream_VB
SimpleInventory_CSharp
SimpleInventory_VB
SimpleDotNetSamples.sln

1. Open Visual Studio 2005 solution `SimpleDotNetSamples.sln` from `/Samples/SimpleDotNetSamples`
2. Connect NUR module / Sampo to free USB port on PC.
   If USB driver not found, install USB driver from: “**USB driver/NUR USB Setup.exe**”
3. Run **HelloNUR_VB**

2.1.3 Creating Visual Studio project

Follow steps below to create Visual Studio project for programming NUR module applications.

1. Create Visual Studio project Visual Basic or C#
2. Add reference for NurApiDotNet.dll. (x86 or x64 version)
   NurApiDotNet.dll is located in NUR API SDK file path structure:
   x86 version: `...\NurApi\dotnet\dotnet_windows\x86`
   x64 version: `...\NurApi\dotnet\dotnet_windows\x64`
3. Select **Target platform** to x86 or x64 from project property.
4. Add NURAPI.dll in your project using menu items "Project→Add Existing Item..."
5. Change from NURAPI.dll property: [Copy to output directory] to "Copy always".
   (Need to be in same folder than executable)
6. Add `NurApiDotNet` namespace in your code.
   C#: `using NurApiDotNet;`
   VB: `Imports NurApiDotNet`

Sample C# code for needed namespace and defining NurApi handle:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using NurApiDotNet; //Need to use namespace of NurApiDotNet

namespace Inventory
{
    public partial class Form1 : Form
    {
        NurApi hNur = null;

        public Form1()
        {
            InitializeComponent();

            hNur = new NurApi(this); //Handle of NurApi
            //Just print NurApi Dll version in the MessageBox at  start
            MessageBox.Show("NurAPI Dll Version: " + hNur.GetFileVersion());
        }
    }
}
```

3 Programming guidelines

3.1 Exception handling

Most of NurApi .NET methods throw exception for error handling. It is important to use NurApi methods always inside `Try Catch` structure and handle exceptions properly.

Refer NurApiDotNet documentation for methods throwing exceptions.

Example:

```
      Try
            tags = hNur.GetTagStorage()
            For Each tag In tags
                ListBox1.Items.Add(tag.GetEpcString())
            Next
            If e.data.stopped Then
                'Start again if stopped
                hNur.StartInventoryStream()
            End If
            hNur.ClearTags() 'Clear NUR memory as well
      Catch ex As Exception
            'Error. Handle exception
            ListBox1.Items.Add(ex.Message)
      End Try
```

3.2 Events

NurAPI .NET provides notifications when Nur module has something to say for application. One of these kinds of event is `ConnectedEvent` indicating that NUR module is connected successfully to application via selected transport.

Example:

Initializing Connected event. (VB)

```
AddHandler hNur.ConnectedEvent, AddressOf NurConnected
```

Event handler (VB)

```
Sub NurConnected(ByVal sender As System.Object, ByVal e As NurApi.NurEventArgs)
        ‘Module connected. Check NurEventArgs for details
End Sub
```

Initializing Connected event. (C#)

```
hNur.ConnectedEvent += new EventHandler<NurApi.NurEventArgs>(hNur_ConnectedEvent);
```

Event handler (C#)

```
void hNur_ConnectedEvent(object sender, NurApi.NurEventArgs e)
{
       //Module connected. Check NurEventArgs for details
}
```

In Windows Form application, it is forbidden to call windows control methods inside event handler because control is accessed from a thread other than the thread it was created on.

To make thread-safe calls to Windows Forms controls in NurApi event handlers, add following line in to program initialization:

(VB)

```
hNur.SetNotificationReceiver(Me)
```

(C#)

```
hNur.SetNotificationReceiver(this);
```

4 Connection setup

NUR Sampo or Evaluation kit can connected to PC via mini USB connector.

Easiest way to connect your application in to the NUR module is to use:

```
SetUsbAutoConnect(True)
```

NurApi looking for correct USB device continuously and launches `ConnectedEvent` when found.
If USB cable removed out, `DisconnectEvent` launches.

To stop USB Auto connecting, call:

```
SetUsbAutoConnect(False)
```

NURApi supports various commands for connecting module to application:

Call `IsConnected()` to determine if module is connected or not.

4.1 Serial port

NUR module is provided with TTL level serial port. Default serial communication parameters are 115200,8,n,1
No flow control.

Note: if USB cable is connected, serial port cannot be used.

Method for connecting to serial port using default baudrate (115200bps):

```
ConnectSerialPort(int comNumber);
```

Refer to NurApiDotNet documentation for all connection options.

5 Settings

NUR module can be configured for various usage purposes. One of most important settings is Tx-level for adjusting reading distance which may need to adjust many times during application running. Settings like Region, Tx-modulation, Link frequency and Rx Decoding are usually needed to set once.

Refer to NUR Implementation guide for more information about module settings.

Load settings from file directly to module:

```
LoadSetupFile(string filename);
```

Save module settings to file:

```
SaveSetupFile(string filename);
```

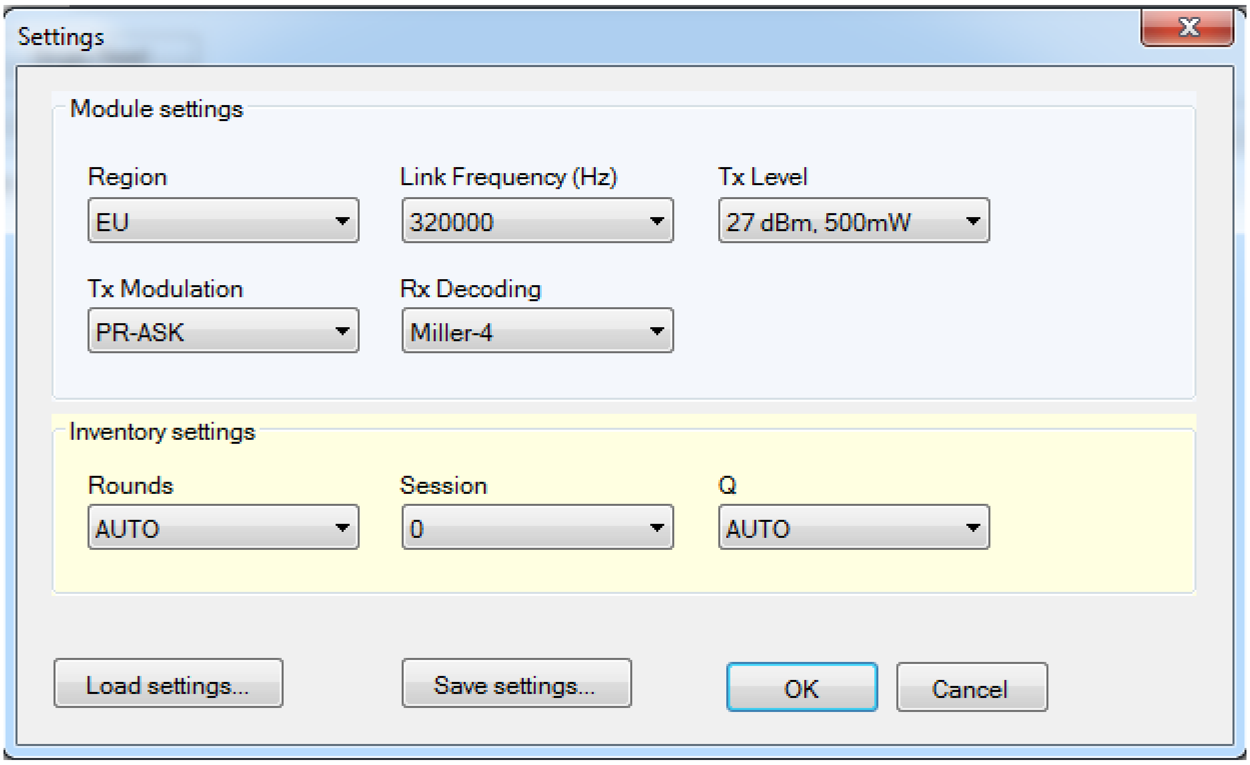
Store settings to module's non-volatile memory

```
StoreCurrentSetup( );
```

Retrieve module setup

```
GetModuleSetup( );
```

Refer sample VB_Sample_VS2010 for example code of module settings. (Visual Basic)


Settings

Module settings

Region: EU
Link Frequency (Hz): 320000
Tx Level: 27 dBm, 500mW
Tx Modulation: PR-ASK
Rx Decoding: Miller-4

Inventory settings

Rounds: AUTO
Session: 0
Q: AUTO

6 Reading RFID tags

The operation of reading Tags. Inventory round begins by transmitting a Query command in one of four sessions. One or more Tags may reply. The NUR module detects a single Tag reply and requests the PC, EPC, and CRC-16 from the Tag.

6.1.1 Simple Inventory

Simple inventory performs one Inventory process and returns tags found.
Refer to “SimpleInventory_VB” and “SimpleInventory_CSharp” applications for details.

Basic Simple inventory (VB code)

```
'Performs simple inventory of tags using current rounds-, Q and session parameters
        Try
            'Clear existing tags from NurModule memory
            hNur.ClearTags()
            'Information about inventory store here
            Dim response As NurApi.InventoryResponse
            'Make Inventory..
            response = hNur.SimpleInventory()
            'Read result to TagStorage object
            Dim inv As NurApi.TagStorage = hNur.FetchTags(True)
            'Show results in the ListBox (EPC code)
            Dim tag As NurApi.Tag
            For Each tag In inv
                ListBox1.Items.Add(tag.GetEpcString())
            Next

        Catch ex As Exception
            'Something went wrong.(usually transport not connected) Show reason in
the MessageBox.
            MessageBox.Show(ex.ToString(), "Exception")
        End Try
```

6.1.2 Inventory Stream

In inventory stream, NUR module continuously reading tags and provides reading result to TagStorage. Refer to “InventoryStream_VB” and “InventoryStream_CSharp” sample applications for details.

Starting InventoryStream:

```
StartInventoryStream()
```

Nur module starts continuously reading tags. `InventoryStream` event notifies when data is available.

VB code of `InventoryStream` event handler:

```
Sub InventoryStreamReady(ByVal sender As System.Object, ByVal e As
NurApi.InventoryStreamEventArgs)
        'Copy tags from NurApi internal tag storage to application tag storage
        'Let's read them using GetTagStorage and show tag EPC in the list box
        Dim tag As NurApi.Tag

        Try
            For Each tag In hNur.GetTagStorage()
                If tags.AddTag(tag) Then
                    'New unique tag added
                    ListBox1.Items.Add(tag.GetEpcString())
                End If
            Next
            'Clear NurApi internal tag storage
            hNur.ClearTags()

            If e.data.stopped Then
                'Start again if stopped
                hNur.StartInventoryStream()
            End If

        Catch ex As Exception
            'Error. Handle exception
            ListBox1.Items.Add(ex.Message)
        End Try
    End Sub
```

Stopping inventory stream:

```
StopInventoryStream()
```

7 Support

support@nordicid.com