Using R&S®CMU200 Drivers in Microsoft Visual Studio 2008 with Visual Basic .NET and C#
Application Note

Products:

| R&S®CMU200

This document describes the usage of the R&S®CMU200 Universal Radio Communication Tester VXIPlug&Play driver using Microsoft Visual Basic .NET and C#.

Application Note

Tom Bifiato, Juergen Straub

06.2009-1MA157_0e

ROHDE&SCHWARZ

Table of Contents

1 Preface

The aim of this application note is to provide information regarding Rohde & Schwarz (R&S) instrument drivers for R&S®CMU200. This document describes how to use the R&S®CMU200 VXIPlug&Play drivers in Microsoft Visual Basic.NET or C# using Visual Studio.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

National Instrument®, LabVIEW®, LabWindows/CVI® are U.S. registered trademarks of National Instrument.

Rohde & Schwarz® is a registered trademark of Rohde & Schwarz GmbH & Co. KG.

2 Driver installation

The VXIPlug&Play drivers may be downloaded from the Rohde & Schwarz web site for the R&S®CMU200.

Download the VXIplug&play driver and example from:
http://www.rohde-schwarz.com/driver/CMU200VXIplugplay.html

The default installation directory for the drivers is the ~VXIPnP\WinNT\ directory, where VXIPNPPATH is your VXIPnP environment variable pointing to your VXIPnP installation directory.

2.1 Installation directory contents

The Install program places the following files on default VXIPlug&Play installation directory defined in the environment variable VXIPNPPATH (~VXIPnP\WinNT\):

● ...\Include\rscmu200.h	Header file for use with C/C++ or Agilent VEE
● ...\rscmu200\rscmu200.c	Source code for use with C
● ...\rscmu200\rscmu200.def	Definition file for use with C++ when building the .dll library
● ...\rscmu200\rscmu200.fp	Function Panel file for use with Agilent VEE and LabWindows/CVI
● ...\rscmu200\rscmu200.bas	Module file for use with Visual Basic
● ...\rscmu200\rscmu200.vb	Module file for use with Visual Basic .NET
● ...\rscmu200\rscmu200.cs	Module file for use with C#
● ...\rscmu200\rscmu200_old.vb	Module file for use with Visual Basic .NET, no longer maintained
● ...\rscmu200\rscmu200_vxi.chm	Compressed HTML help
● ...\rscmu200\license.pdf	Instrument Driver License Agreement
● ...\rscmu200\readme.txt	This file contains general information
● ...\Lib\Msc\rscmu200.lib	Library file for use with MSVC++
● ...\Lib\Bc\rscmu200.lib	Library file for use with Borland
● ...\Bin\rscmu200_32.dll	Dynamic Link Library of instrument driver
● ...\rscmu200\rscmu200.llb	LabVIEW library containing driver VIs
● ...\rscmu200\rscmu200.chm	LabVIEW Context Help (LabVIEW 6.1 or higher)
● ...\rscmu200*.mnu	LabVIEW palette menu files of the driver

- Windows System(32) directory
- instrsup.dll Instrument support Dynamic Link Library file from LabWindows/CVI.

If a particular platform is not going to be used, the corresponding platform-specific files may be deleted.

All equipment directories contain the same type of files

3 Driver structure

The R&S$^{®}$CMU200 drivers are structured using the base driver rscmu200 with additional drivers that supplement the base. For example, there are additional drivers for GSM, TDMA, AMPS, CDMA2000, Bluetooth, WCDMA and 1xEVDO.

Each driver is defined by its own class. The class of the base driver is rscmu200, while the class for the GSM class is rscmuk2g.

The drivers use a wrapper which encapsulates the calls to the methods in the driver DLL's.

4 Creating a Visual Basic .NET example project in Microsoft Visual Studio 2008

The code for these examples can be found on the Rohde & Schwarz web site in the drivers section of the R&S®CMU200.

Download the VXIplug&play driver and example from:
http://www.rohde-schwarz.com/driver/CMU200VXIplugplay.html

How to create a project in Microsoft Visual Studio:

Create a Windows Form Application project in Visual Basic. NET. In this example, we will create a project called R&S®CMU200 Demo. Please use the project defaults.
Rename `Public Class Form1` to `Public Class CMU200_Demo`.
Now we will add the R&S®CMU200 Visual Basic wrapper files to the project, rscmu200.vb and *rscmuk2g.vb*. The *rscmu200.vb* is the base driver and the *rscmuk2g.vb* is the GSM driver.

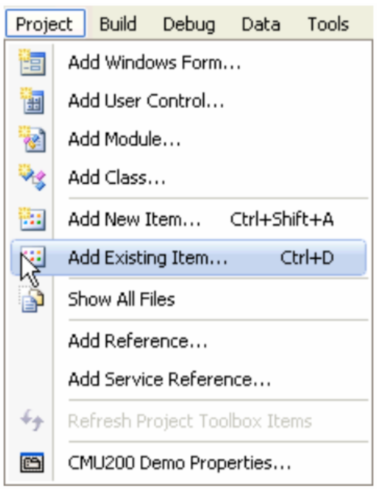
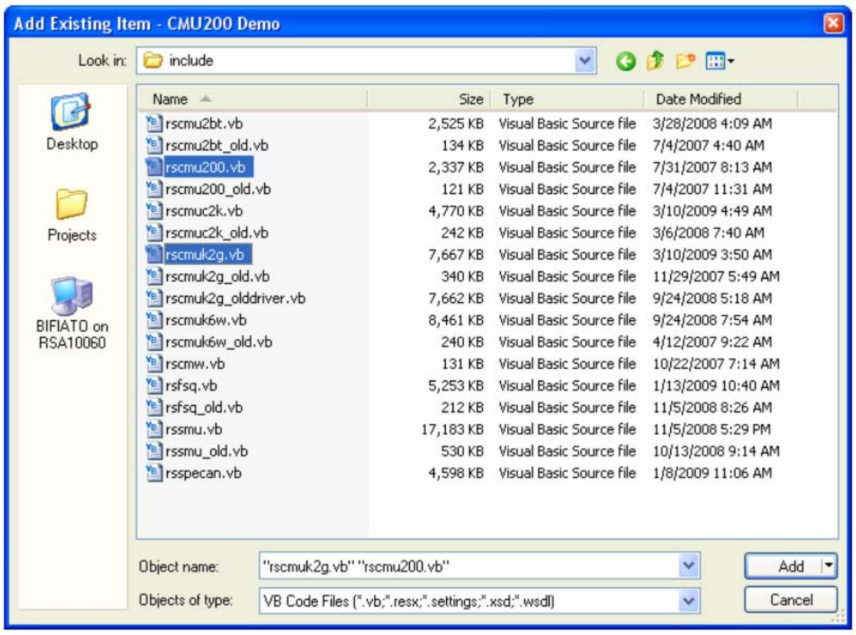


Figure 1: Adding Existing Item

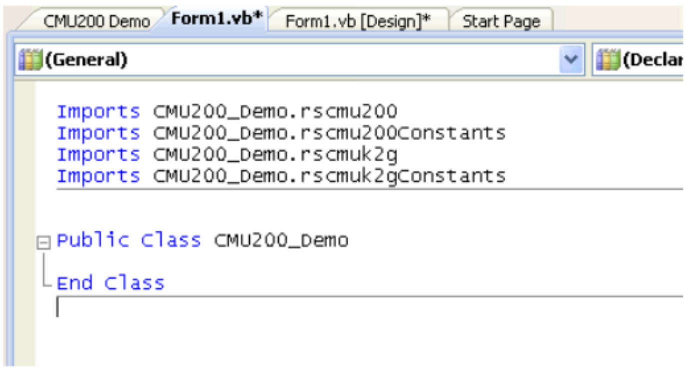


Name	Size	Type	Date Modified
rscmu2bt.vb	2,525 KB	Visual Basic Source file	3/28/2008 4:09 AM
rscmu2bt_old.vb	134 KB	Visual Basic Source file	7/4/2007 4:40 AM
rscmu200.vb	2,337 KB	Visual Basic Source file	7/31/2007 8:13 AM
rscmu200_old.vb	121 KB	Visual Basic Source file	7/4/2007 11:31 AM
rscmuc2k.vb	4,770 KB	Visual Basic Source file	3/10/2009 4:49 AM
rscmuc2k_old.vb	242 KB	Visual Basic Source file	3/6/2008 7:40 AM
rscmuk2g.vb	7,667 KB	Visual Basic Source file	3/10/2009 3:50 AM
rscmuk2g_old.vb	340 KB	Visual Basic Source file	11/29/2007 5:49 AM
rscmuk2g_olddriver.vb	7,662 KB	Visual Basic Source file	9/24/2008 5:18 AM
rscmuk6w.vb	8,461 KB	Visual Basic Source file	9/24/2008 7:54 AM
rscmuk6w_old.vb	240 KB	Visual Basic Source file	4/12/2007 9:22 AM
rscmw.vb	131 KB	Visual Basic Source file	10/22/2007 7:14 AM
rsfsq.vb	5,253 KB	Visual Basic Source file	1/13/2009 10:40 AM
rsfsq_old.vb	212 KB	Visual Basic Source file	11/5/2008 8:26 AM
rssmu.vb	17,183 KB	Visual Basic Source file	11/5/2008 5:29 PM
rssmu_old.vb	530 KB	Visual Basic Source file	10/13/2008 9:14 AM
rsspecan.vb	4,598 KB	Visual Basic Source file	1/8/2009 11:06 AM

Object name: "rscmuk2g.vb" "rscmu200.vb"
Objects of type: VB Code Files (*.vb;*.resx;*.settings;*.xsd;*.wsdl)

Figure 2: Existing Items to Add to project

Add Imports to class. The following imports need to be added to the class:

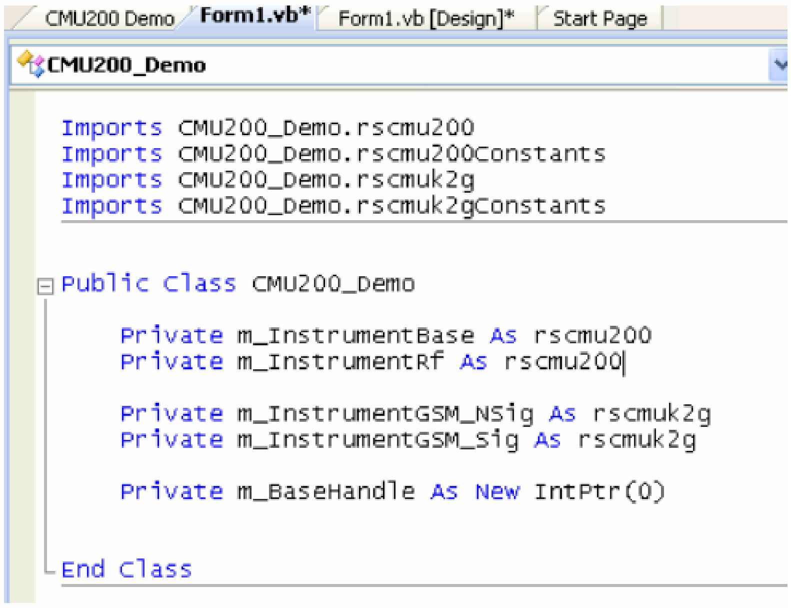


```
Imports CMU200_Demo.rscmu200
Imports CMU200_Demo.rscmu200Constants
Imports CMU200_Demo.rscmuk2g
Imports CMU200_Demo.rscmuk2gConstants

Public Class CMU200_Demo

End Class
```

Figure 3: Imports Added to project

The IntelliSense® of the IDE (integrated development environment) should be active to assist with the Imports statements.

Add properties to hold references to the classes.



```
Imports CMU200_Demo.rscmu200
Imports CMU200_Demo.rscmu200Constants
Imports CMU200_Demo.rscmuk2g
Imports CMU200_Demo.rscmuk2gConstants

Public Class CMU200_Demo

    Private m_InstrumentBase As rscmu200
    Private m_InstrumentRf As rscmu200

    Private m_InstrumentGSM_NSig As rscmuk2g
    Private m_InstrumentGSM_Sig As rscmuk2g

    Private m_BaseHandle As New IntPtr(0)

End Class
```

Figure 4: Add Private Properties

Develop the form1 interface. The form1 interface is shown in the figure below:

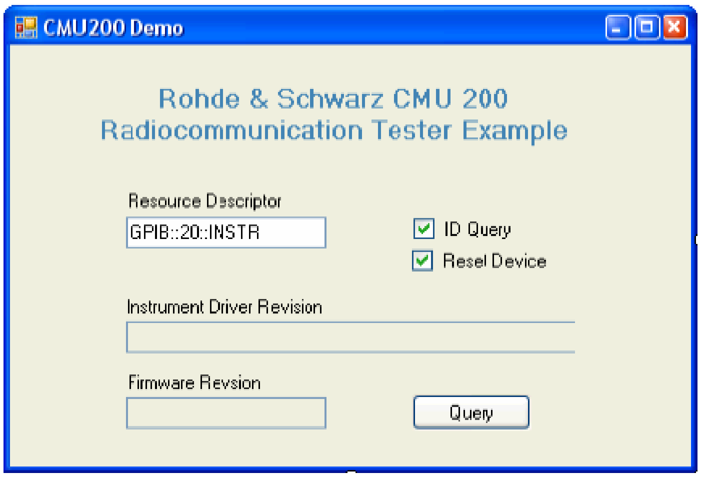


Figure 5: Form1 Interface

Add the following code to the project:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
        Try
            Dim err_message As New System.Text.StringBuilder(256)
            Dim driver_revision As New System.Text.StringBuilder(256)
            Dim fw_revision As New System.Text.StringBuilder(256)
            UseWaitCursor = True

            System.Windows.Forms.Cursor.Current = Cursors.WaitCursor

m_InstrumentBase = New rscmu200(ResourceDescriptor.Text.Insert(ResourceDescriptor.Text.LastIndexOf("::"),
"::0"), IDQuery.Checked, ResetDevice.Checked)
            m_InstrumentBase.Base_RevisionQuery(driver_revision, fw_revision, err_message)
            driverRevision.Text = driver_revision.ToString()
            fwRevision.Text = fw_revision.ToString()

            'get base handle
            m_InstrumentBase.GetInstrumentHandle(m_BaseHandle)
            m_InstrumentGSM_NSig = rscmuk2g.Init_GSM_NSig(m_BaseHandle, rscmuk2gConstants.Gsm900,
err_message)
            m_InstrumentGSM_NSig.GSM_NSig_TrigSlop(1, err_message)
            m_InstrumentGSM_NSig.GSM_NSig_ConfPowCont(rscmuk2gConstants.ValueAll,
rscmuk2gConstants.StatCount, err_message)
            m_InstrumentGSM_NSig.Close_GSM_NSig(err_message)

            m_InstrumentGSM_Sig = rscmuk2g.Init_GSM_Sig(m_BaseHandle, rscmuk2gConstants.Gsm900,
err_message)
            m_InstrumentGSM_Sig.GSM_Sig_TrigSlop(0, err_message)
            m_InstrumentGSM_Sig.Close_GSM_Sig(err_message)

        Catch ex As Exception

            Dim message As String

            message = "m_Instrument Status Error: "
            message += ex.Message
            MessageBox.Show(message)

        Finally
            Try
                m_InstrumentBase.Dispose()
                m_InstrumentGSM_NSig.Dispose()
                m_InstrumentGSM_Sig.Dispose()
            Catch nex As Exception
            End Try

        End Try

        System.Windows.Forms.Cursor.Current = Cursors.Arrow
        UseWaitCursor = False
    End Sub
```

Code details:

- The *m_InstrumentBase* is instantiated with the *New* statement on the *rscmu200* base using as parameters the *ResourceDescriptor* from form1, the *IDQuery* and the *ResetDevice* check boxes.
- The *driver_revision* and *fw_revision* are queried from the instrument and converted to strings for display on the form.
- The base instrument handle is used in other *rscmu200* base drivers, such as the GSM driver in this example. The method *m_InstrumentBase.GetInstrumentHandle(m_BaseHandle)* returns the handle of the base instrument.
- The next code area instantiates a GSM 900 MHz Nonsignalling class using the method *rscmuk2g.Init_GSM_NSig*. The *m_BaseHandle* is passed in as well as the function group name using the class *rscmuk2gConstants* and an *err_message* variable. If there is an error, the error message will be contained in this variable.
- Once the *m_InstrumentGSM_NSig* is instantiated in the above step, you will have access to all of the *rscmuk2g* class methods. Two of those methods are shown in the example, *GSM_NSig_TrigSlop* and *GSM_NSig_ConfPowCont*.
- Once you are finished using the function group, the function group should be closed using the *Close_** method.
- The same steps are used to instantiate the signaling function group using the method *rscmuk2g.Init_GSM_Sig*.

5 Creating a C# example project in Microsoft Visual Studio 2008

The code for these examples can be found on the Rohde & Schwarz web site in the drivers section of the R&S®CMU200.

Download the VXIplug&play driver and example from:
http://www.rohde-schwarz.com/driver/CMU200VXIplugplay.html

How to create a project in Microsoft Visual Studio:

Create a Windows Form Application project in Visual C#.NET. In this example, we will create a project called R&S®CMU200 Example. Please use the project defaults.
Rename `Public Partial Class Form1` to `Public Partial Class CMU200_Example`.
Now we will add the rscmu200 Visual C# wrapper files to the project, *rscmu200.cs* and *rscmuk2g.cs*. The *rscmu200.cs* is the base driver and the *rscmuk2g.cs* is the GSM driver.

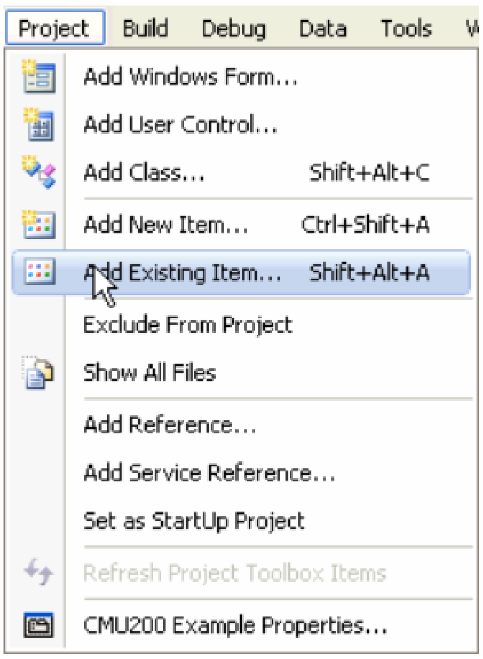
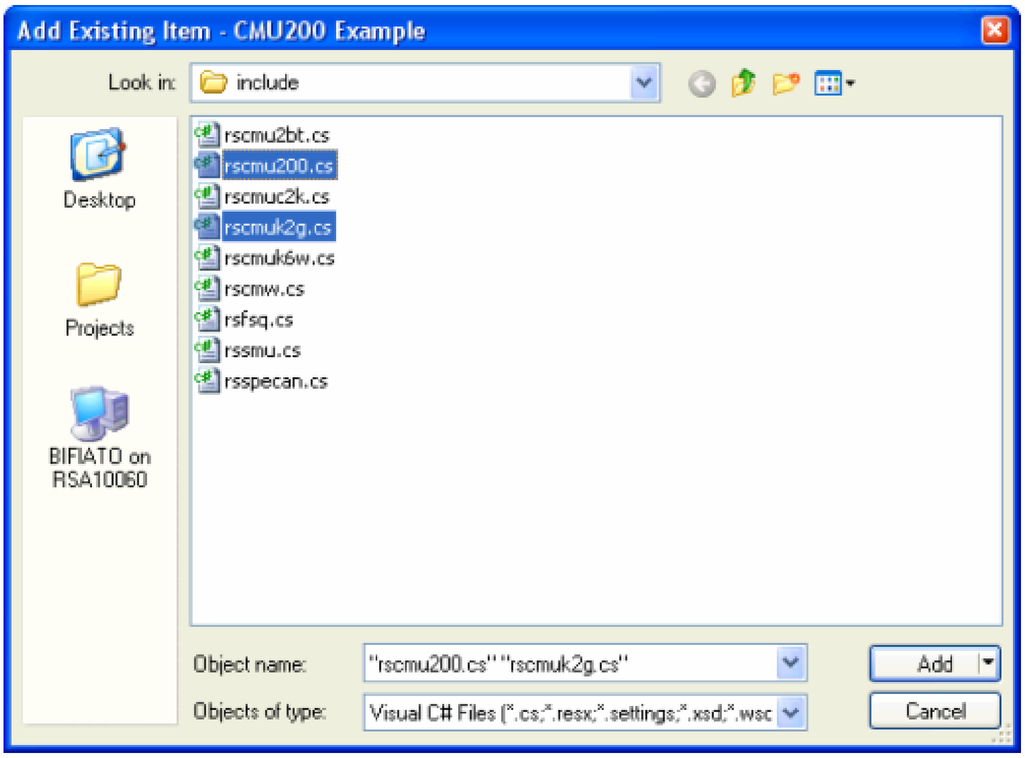


Figure 6: Add Existing Item

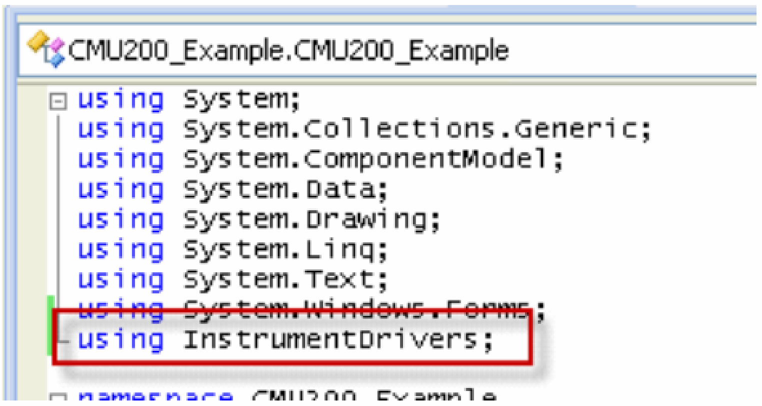


rscmu2bt.cs
rscmu200.cs
rscmuc2k.cs
rscmuk2g.cs
rscmuk6w.cs
rscmw.cs
rsfsq.cs
rssmu.cs
rsspecan.cs

Object name: "rscmu200.cs" "rscmuk2g.cs"
Objects of type: Visual C# Files (*.cs;*.resx;*.settings;*.xsd;*.wsc

Figure 7: Existing Items to Add to project

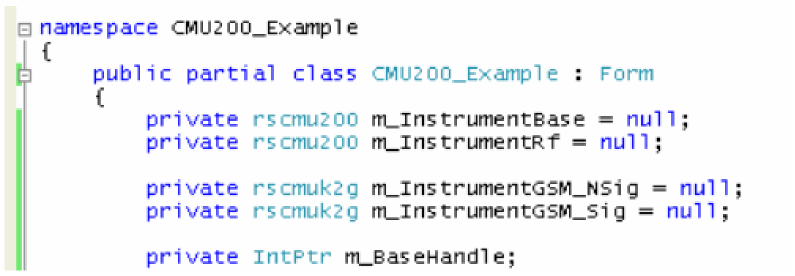
The C# driver Import/Using differs from the Visual Basic .NET in that only the namespace *InstrumentDrivers* need to be imported.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using InstrumentDrivers;
```

Figure 8: Imports/using to project

Add properties to hold references to the classes.



```
namespace CMU200_Example
{
    public partial class CMU200_Example : Form
    {
        private rscmu200 m_InstrumentBase = null;
        private rscmu200 m_InstrumentRf = null;

        private rscmuk2g m_InstrumentGSM_NSig = null;
        private rscmuk2g m_InstrumentGSM_Sig = null;

        private IntPtr m_BaseHandle;
```

Figure 9: Add properties to the project

Develop the form1 interface. The form1 interface is shown in the figure blow:

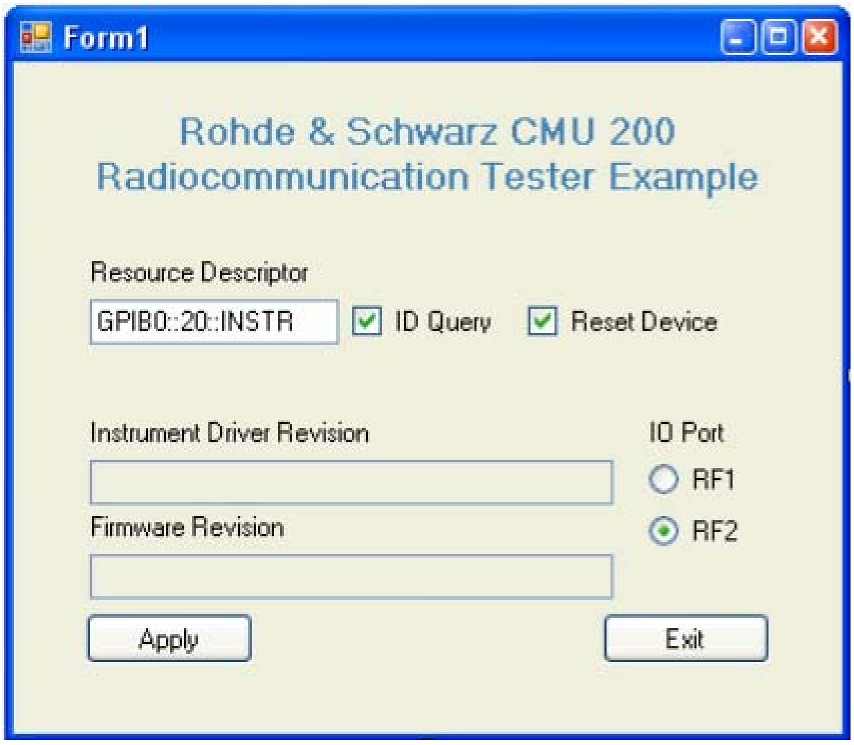


Figure 10: Form 1 Interface

Add the following code to the project:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using InstrumentDrivers;

namespace CMU200_Example
{
    public partial class CMU200_Example : Form
    {
        private rscmu200 m_InstrumentBase = null;
        private rscmu200 m_InstrumentRf = null;

        private rscmuk2g m_InstrumentGSM_NSig = null;
        private rscmuk2g m_InstrumentGSM_Sig = null;

        private IntPtr m_BaseHandle;

        public CMU200_Example()
```

```
{
    InitializeComponent();
}

private void ExitButton_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Are you sure you want to exit?", "Exit?", MessageBoxButtons.YesNo) ==
     DialogResult.Yes)
        Close();
}

private void Apply_Click(object sender, EventArgs e)
{
    try
    {
        StringBuilder err_message = new StringBuilder(256);
        StringBuilder driver_revision = new StringBuilder(256);
        StringBuilder fw_revision = new StringBuilder(256);

        UseWaitCursor = true;
        System.Windows.Forms.Cursor.Current = Cursors.WaitCursor;

        // Sets up Instrument Base
        if (m_InstrumentBase == null)
            m_InstrumentBase = new
            rscmu200(ResourceDescriptor.Text.Insert(ResourceDescriptor.Text.LastIndexOf("::"),
            "::0"), IDQuery.Checked, ResetDevice.Checked);

        m_InstrumentBase.Base_RevisionQuery(driver_revision, fw_revision, err_message);
        driverRevision.Text = driver_revision.ToString();
        fwRevision.Text = fw_revision.ToString();

        // Gets the base handle
        m_InstrumentBase.GetInstrumentHandle(out m_BaseHandle);

        //Sets up the Non-signalling GSM Instrument
        if (m_InstrumentGSM_NSig == null)
            m_InstrumentGSM_NSig = rscmuk2g.Init_GSM_NSig(m_BaseHandle, rscmuk2gConstants.Gsm900,
            err_message);

        m_InstrumentGSM_NSig.GSM_NSig_TrigSlop(0, err_message);
        m_InstrumentGSM_NSig.Close_GSM_NSig(err_message);

        //Sets up the Signalling GSM Instrument
        if (m_InstrumentGSM_Sig == null)
            m_InstrumentGSM_Sig = rscmuk2g.Init_GSM_Sig(m_BaseHandle, rscmuk2gConstants.Gsm900,
            err_message);

        m_InstrumentGSM_Sig.GSM_Sig_TrigSlop(0, err_message);
        m_InstrumentGSM_Sig.Close_GSM_Sig(err_message);

        //Sets up the RF Instrument
        if (m_InstrumentRf == null)
            m_InstrumentRf = new
            rscmu200(ResourceDescriptor.Text.Insert(ResourceDescriptor.Text.LastIndexOf("::"),
            "::1"), IDQuery.Checked, ResetDevice.Checked);

        m_InstrumentBase.GetInstrumentHandle(out m_BaseHandle);
        m_InstrumentRf.initRFNSig(m_BaseHandle);

        if (rf1.Checked)
        {
            m_InstrumentRf.RF_NSig_Inp(rscmu200Constants.InputRf1, err_message);
            m_InstrumentRf.RF_NSig_Outp(rscmu200Constants.OutputRf1, err_message);
        }
        else
        {
            m_InstrumentRf.RF_NSig_Inp(rscmu200Constants.InputRf2, err_message);
            m_InstrumentRf.RF_NSig_Outp(rscmu200Constants.OutputRf2, err_message);
        }

        m_InstrumentRf.RF_NSig_InitRfg(err_message);
        m_InstrumentRf.RF_NSig_InitRfan(err_message);

    }
    catch (Exception ex)
    {
        if (m_InstrumentBase != null)
        {
            m_InstrumentBase.Dispose();
            m_InstrumentBase = null;
        }
        if (m_InstrumentRf != null)
        {
            m_InstrumentRf.Dispose();
            m_InstrumentRf = null;
        }

        if (m_InstrumentGSM_NSig != null)
        {
            m_InstrumentGSM_NSig.Dispose();
            m_InstrumentGSM_NSig = null;
        }

        if (m_InstrumentGSM_Sig != null)
        {
            m_InstrumentGSM_Sig.Dispose();
            m_InstrumentGSM_Sig = null;
        }

        String message;

        message = "m_Instrument Status Error: ";
        message += ex.Message;
        MessageBox.Show(message);
    }
    finally
```

```
            {
                if (m_InstrumentBase != null)
                {
                    m_InstrumentBase.Dispose();
                    m_InstrumentBase = null;
                }

                if (m_InstrumentGSM_NSig != null)
                {
                    m_InstrumentGSM_NSig.Dispose();
                    m_InstrumentGSM_NSig = null;
                }

                if (m_InstrumentGSM_Sig != null)
                {
                    m_InstrumentGSM_Sig.Dispose();
                    m_InstrumentGSM_Sig = null;
                }

                if (m_InstrumentRf != null)
                {
                    m_InstrumentRf.Dispose();
                    m_InstrumentRf = null;
                }
            }

            System.Windows.Forms.Cursor.Current = Cursors.Arrow;
            UseWaitCursor = false;
        }
    }
}
```

Code details:

- The *m_InstrumentBase* is instantiated with the *new* statement on the *rscmu200* base using as parameters the *ResourceDescriptor* from *form1*, the *IDQuery* and the *ResetDevice* check boxes.
- The *driver_revision* and *fw_revision* are queried from the instrument and converted to strings for display on the form.
- The base instrument handle is used in other *rscmu200* base drivers, such as the GSM driver in this example. The method *m_InstrumentBase.GetInstrumentHandle(m_BaseHandle)* returns the handle of the base instrument.
- The next code area instantiates a GSM 900 MHz Nonsignalling class using the method *rscmuk2g.Init_GSM_NSig*. The *m_BaseHandle* is passed in as well as the function group name using the class *rscmuk2gConstants* and an *err_message* variable. If there is an error, the error message will be contained in this variable.
- Once the *m_InstrumentGSM_NSig* is instantiated in the above step, you will have access to all of the *rscmuk2g* class methods. Two of those methods are shown in the example, *GSM_NSig_TrigSlop* and *GSM_NSig_ConfPowCont*.
- Once you are finished using the function group, the function group should be closed using the *Close_** method.
- The same steps are used to instantiate the signalling function group using the method *rscmuk2g.Init_GSM_Sig*.

About Rohde & Schwarz
Rohde & Schwarz is an independent group of companies specializing in electronics. It is a leading supplier of solutions in the fields of test and measurement, broadcasting, radiomonitoring and radiolocation, as well as secure communications. Established 75 years ago, Rohde & Schwarz has a global presence and a dedicated service network in over 70 countries. Company headquarters are in Munich, Germany.

Regional contact
Europe, Africa, Middle East
+49 1805 12 42 42* or +49 89 4129 137 74
customersupport@rohde-schwarz.com

North America
1-888-TEST-RSA (1-888-837-8772)
customer.support@rsa.rohde-schwarz.com

Latin America
+1-410-910-7988
customersupport.la@rohde-schwarz.com

Asia/Pacific
+65 65 13 04 88
customersupport.asia@rohde-schwarz.com

Certified Quality System
ISO 9001
DQS REG. NO 1954 QM

Certified Environmental System
ISO 14001
DQS REG. NO 1954 UM

Rohde & Schwarz GmbH & Co. KG
Mühldorfstraße 15 | D - 81671 München
Phone + 49 89 4129 - 0 | Fax + 49 89 4129 – 13777

www.rohde-schwarz.com