Development Hints and Best Practices for Using Instrument Drivers

Application Note

Products:

| Instrument Drivers

This document answers frequently asked questions regarding Rohde & Schwarz instrument drivers and their programming environments.

Juergen Engelbrecht
12-Jan-15-1MA153_14e

Application Note

ROHDE&SCHWARZ

Table of Contents

1 Preface

The aim of this application note is to provide information regarding Rohde & Schwarz instrument drivers. This paper shall help application engineers and software developers to easily get an understanding of advanced techniques to develop test and measurement (T&M) applications by utilizing ROHDE & SCHWARZ instrument drivers. Furthermore the nomenclature used for ROHDE & SCHWARZ instrument drivers will be explained.

It does not describe any programming languages or how to develop software in any programming language.

2 Frequently Asked Questions (FAQ)

2.1 What Is an Instrument Driver?

An instrument driver, in the context of T&M application development, is a set of software routines that simplify the remote control of a measuring instrument. Input and output (I/O) abstraction by means of hiding the SCPI[1] commands makes programming much easier.

ROHDE & SCHWARZ instrument drivers are based on Virtual Instrument Software Architecture (VISA[2]), which enables a physically independent connection between the host PC and the instrument.

Instrument drivers generally allow greater system flexibility and accelerate development time, for example through the possibility of reusing software. A major advantage is the reduction of the development time for any kind of T&M application by using a high-level programming language as opposed to implementing an application with SCPI commands. For instance, all string formatting which is necessary when using SCPI commands is already done by the instrument driver. Also a commonly used term is *command completion*[3] of integrated development environments (IDE). This feature is not available for SCPI commands.

2.2 Why Should I Use an Instrument Driver Instead of SCPI Commands?

Writing an application can take a lot of time. But writing the application is not the end of the story, since the T&M instrument must also be driven. That is not so simple with complex equipment, since just the description of the command set may comprise several hundred pages. Applying yourself to this task can be time-consuming. Which is why ROHDE & SCHWARZ provides ready-to-use software instrument drivers, relieving designers of these efforts to a great extent. You no longer have to search through manuals looking for commands; this has already been done when developing the instrument driver.

---

[1] Standard Commands for Programmable Instruments (SCPI) is a command set for remote-controlling T&M instruments. Further information is available on http://www.scpiconsortium.org/.

[2] Virtual Instrument Software Architecture (VISA) as host PC software is a well-defined hardware abstraction layer and I/O programming interface for various interfaces, e.g. General Purpose Interface Bus (GPIB, IEEE 488.2) or Ethernet (VXI-11).

[3] "Command completion" of current integrated development environments offers the possibility to save time and avoid typing errors during programming (e.g. in C++ or C#).

Drawbacks of programming with SCPI commands:
For simple applications such as setting frequency/RF level and reading out simple measurement values (e.g. for digital multimeters), SCPI programming can be sufficient. However, for more sophisticated applications this approach reaches its limits quite fast.

Example: simple spectrum analysis
Even for simple spectrum analysis, the use of SCPI commands becomes quite complicated and error-prone. For instance, all data formatting of a read-out data stream has to be done manually when using SCPI commands. This formatting is already done by the instrument driver, so the read-out data can be easily processed within the T&M application.

As one can imagine, for more complex T&M applications – such as generating or measuring a mobile standard signal, e.g. for Wideband Code Division Multiple Access (WCDMA) – using SCPI commands is quite time-consuming. Therefore, using instrument drivers to set up your instrumentation avoids a lot of string formatting and other data parsing. Moreover, the local grouping of functions within the provided compressed HTML help file (chm file) offers you all configuration possibilities at a glance.

Development, maintenance and software reuse
Instrument drivers speed up your application development process. This can be achieved because debugging and error diagnostics are well supported by the instrument driver. As a result, you can easily get rid of wrong configurations as well as commands that are not understood by the instrument. This is done by extensively using the instrument's error-reporting functionality. For further details, please refer to section 4.

Requirements can change during the lifetime of T&M applications. This is another reason why instrument drivers can help you to modify an application. The logical grouping of your instrument's functionality makes it easy to extend or modify your application on an abstract and logical level.

The described arguments in favor of simplifying your T&M task are a further reason for reusing already written software. Due to the concept of instrument family drivers, software written for research and development reasons on a high-end instrument such as the R&S®SMW or SMU200A vector signal generator can easily be reused for production lines equipped with e.g. the mid-range R&S®SMBV100A vector signal generator. The instrument driver abstracts the instrument, so this software can often be reused without any changes. Of course, the functionality needs to be supported on both instruments.

2.3 VISA and Physical Interfaces

All ROHDE & SCHWARZ instrument drivers use VISA to avoid an individual driver for every different device interface. The VISA I/O software is generally an application programming interface (API) providing an input and output low-level API to communicate with measuring instruments. This standard was defined by the VXIplug&play Systems Alliance, with National Instruments, ROHDE & SCHWARZ and other companies represented in the consortium.

These I/O functions are independent of the used device interface.

An initialization string (so-called VISA resource identifier) in the relevant application program defines which device interface (IEEE 488.2 (GPIB), RS-232, LAN (VXI-11, raw socket), USB (USB-TMC)) to use. The subsequent program routine is completely independent of the interface used.

All ROHDE & SCHWARZ instrument drivers require at least National Instruments VISA[4] (NI-VISA) revision 4.0 or later. For Linux it is strongly recommended to use the latest version of NI-VISA. It is also possible to use the Agilent I/O Library Suite[5].

2.4 Which 64-bit operating systems are supported by Instrument Drivers?

Yes, the most recent ROHDE & SCHWARZ instrument drivers are build to support 64-bit operating systems. Only 64-bit operating systems with a support for an VISA installation are supported by instrument drivers.

Are 32-bit Application supported on 64-bit Operating Systems?
Yes, there is the possibility to use the instrument drivers together with 64-bit applications, was well as an 32-bit application on 64-bit operating systems.

2.5 Where Do I Find the VXIplug&play Installation Directory?

This path is defined by the environment variable *VXIPNPPATH* on your operating system.

For Windows-based operating systems, this variable can be easily read out via Windows command prompt using the command "*set*".

[4] For further information please visit http://www.ni.com/visa.
[5] For further information please visit http://www.agilent.com.

Note: With NI-VISA Version 4.2, the default installation path changed from *C:\VXIPNP* to *C:\Program Files\IVI Foundation\VISA*, respectively *C:\Program Files (x86)\IVI Foundation\VISA* for WoW64[6].

64-bit support of National Instruments VISA
On 64-bit Windows operating systems, the 64-bit environment variable is called V*XIPNPPATH64*.

2.6 How Do I Install an Instrument Drivers?

For installation instructions, please use the driver-specific "How to use…", which can be downloaded from the instrument's driver download page:

http://www.rohde-schwarz.com/drivers.

Further specific steps and instructions for installing instrument drivers are available on the driver download site available at the product Internet site.

2.7 How Do I Determine the VISA Resource Identifier?

As VISA supports a wide range of interfaces to the instrument (GPIB, LAN with different protocols, USB, etc.), there must be a way of telling VISA which interface to use.
This is done by the resource descriptor/name/identifier. This is a string which is passed when calling the init function of an instrument driver (or *viOpen()*) and configures the interface to use.
The table below shows all relevant VISA resource identifiers if operating with Rohde & Schwarz instrument. Optional string segments are shown in square brackets ([ ]).

VISA resource identifier		
Interface	**Descriptor**	**Descriptor parameters (in blue)**

GPIB instrument	`GPIB[board]::`
	`primary address`
	`[::secondary address]`
	`[::INSTR]`
		GPIB board number (optional – typically skipped)
		Instrument primary GPIB address (0 to 31)
		Instrument secondary address (optional; R&S®CMU200 only)

[6] Windows 32-bit On Windows 64-bit; A subsystem of the Windows operating system to execute 32-bit applications on 64-bit Windows operating systems.

VISA resource identifier		
Interface	**Descriptor**	**Descriptor parameters (in blue)**

LAN
HiSLIP protocol[7]
	`TCPIP[board]::`
	`host address`
	`[::LAN device`
	`name/address]`
	`[::INSTR]`
		TCP/IP board number (optional – typically skipped)
		Host address as TC/IP address or computer name
		Device name, mandatory for HiSLIP, default: hislip0

LAN
VXI-11 protocol
	`TCPIP[board]::`
	`host address`
	`[::LAN device`
	`name/address]`
	`[::INSTR]`
		TCP/IP board number (optional – typically skipped)
		Host address as TC/IP address or computer name
		Device name (optional, default: inst0)

LAN
raw socket
	`TCPIP[board]::`
	`host address::`
	`port::`
	`SOCKET`
		TCP/IP board number (optional – typically skipped)
		Host address as TCP/IP address or computer name
		Port to use (typically: 5052)
		Indicate connection is socket connection

USB
TMC protocol
	`USB[board]::`
	`manufacturer ID::`
	`model code::`
	`serial number`
	`[::USB interface`
	`number]`
	`[::INSTR]`
		USB board number (optional – typically skipped)
		Manufacturer ID (0xAAD for ROHDE & SCHWARZ)
		Model ID (refer to instrument manual for details)
		Serial number (e.g. 100117)
		Interface number (optional – typically skipped)

Table 1: List of VISA resource identifiers.

[7] This feature requires support by the VISA library as well as support of the instrument firmware. Please consult the operation manual to get further information on the HiSLIP support of your ROHDE & SCHWARZ instrument.

3 Best Practices for Software Development

3.1 Getting Started

To get started please download one of the provided application examples from the driver download site

http://www.rohde-schwarz.com/drivers

to get an idea of what a remote-control application can look like.

It is recommended to use the provided instrument driver compressed HTML help files (chm files) for getting started.

3.2 How to Start to Develop my Own Application

To get started, it is helpful to use the *Instrument Driver Help* file (rsXYZ*[_vxi].chm*) in your instrument driver's installation directory in combination with the remote-control command section of your instrument's operating manual.

A good approach is to use the provided application examples as a skeleton for new applications.

The easiest way to find the proper driver function call is to find the SCPI command (in the long version) in the instrument operating manual and to look for the command in the *Instrument Driver Help* file using the index or search functionality.

For developing more complex applications, it is helpful to identify the manual operation steps and utilize the "Instrument Driver Tree Structure" to find your driver function calls for the manual configuration steps in this logical structure.

Information contained in the *Instrument Driver Help* file, also *Driver Attribute Help* (*rsXYZ[_vxi].chm*)):

- Mapping of SCPI command and driver calls (and vice versa) via the search and index functionality
- Grouped driver functions to easily identify all functions for a specific 3GPP standard (Instrument Driver Tree Structure)
- For attribute-based drivers (with rsXYZ_attr.chm): mapping of attributes and driver calls (see Fig. 1)

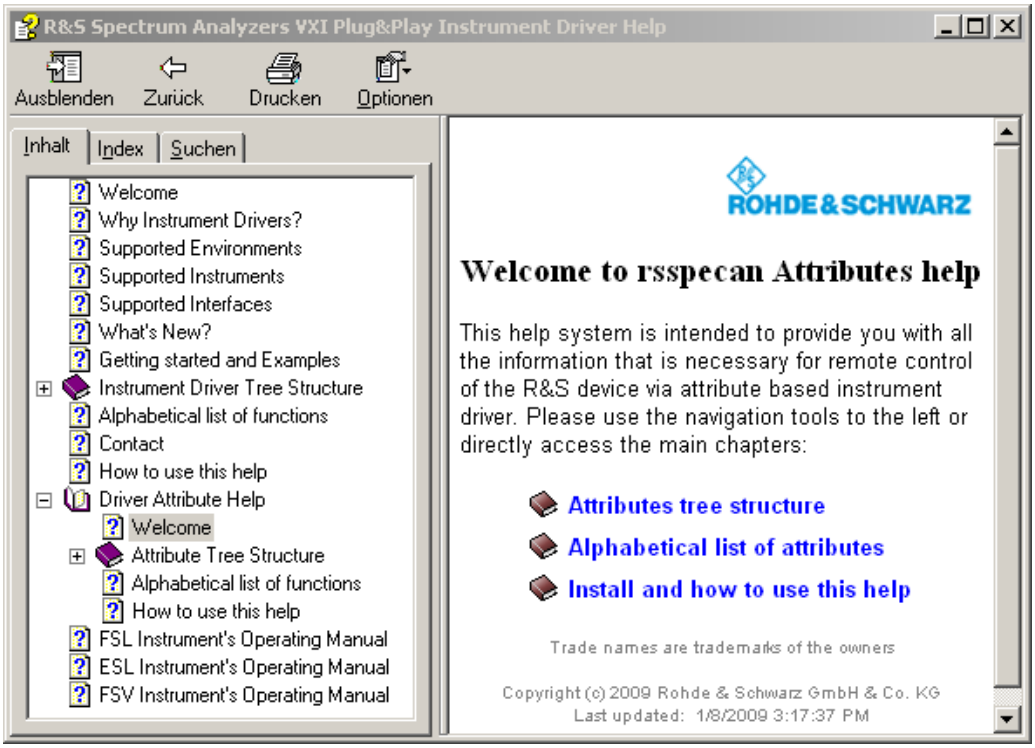

Figure 1: Mapping of attributes and driver calls within the Driver Attribute Help section.

This feature is not available in all ROHDE & SCHWARZ instrument drivers.

3.3 How to Integrate an Instrument Drivers into Microsoft Visual Studio

The following examples are referring to the *rsspecan* instrument driver. The described procedures are all adaptable to other ROHDE & SCHWARZ instrument drivers, only the naming of the files can be different. The naming convention is *rsXYZ*, where *XYZ* is the abbreviation of the instrument's (-family) driver.

3.3.1 C# and Instrument Drivers: Getting Started

A wrapper is necessary to enable a direct access to the driver dynamic linked library (*dll*). This wrapper is automatically installed in the *~VXIplug&play\WinNT\include* (or *~VXIplug&play64\Win64\include*) directory, where *VXIPNPPATH* (or *VXIPNPPATH64) is* the VXIplug&play environment variable pointing to your VXIplug&play installation directory. Please note, that for the instruments that in the meantime have a IVI.NET – a native .NET instrument driver available, the *.cs C# wrapper file is no more available in the VXIplug&play driver installation. We encourage the user to switch to the native .NET instrument driver.

Creating a new application project in Microsoft Visual Studio:

Create a new project, e.g. File -> New -> Project.

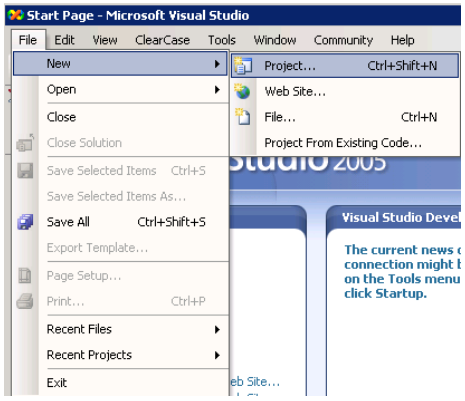

Figure 2: Creating a new project in Visual Studio.

Select your programming language and the template type, e.g. *Visual C# Windows Application* or *Console Application*.

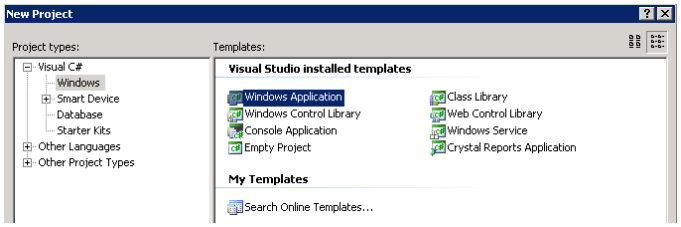

Figure 3: Selecting your desired template.

To access the instrument driver, you have to include the provided VXIplug&play wrapper for *C#,* e.g. *Project->Add Existing Item…* *~VXIplug&play\WinNT\include\rsspecan.cs* or *~VXIplug&play64\Win64\include\rsspecan64.cs* for 64-bit applications.

Note: It is very important to explicitly specify the target platform for the solution using the Configuration Manager. Either *x86* for 32-bit application or *x64* for 64-bit applications.

Also the driver's namespace has to be added to the current source code file, for example via the directive *"using InstrumentDrivers"*. The result is shown in Fig. 4:

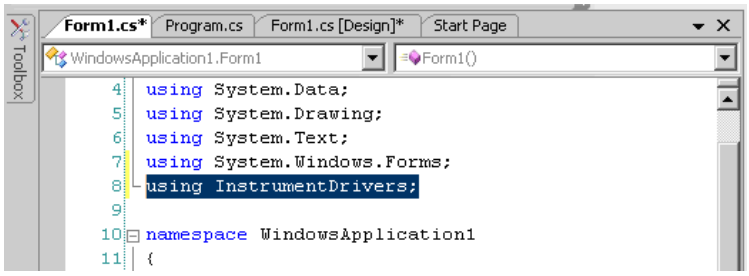

Figure 4: Adding the rsspecan driver to your current project.

Now the development environment is configured in a proper way. For getting your application development started, please refer to the examples provided on your instrument's driver download site:
http://www.rohde-schwarz.com/drivers.

3.3.2 VB.NET and Instrument Drivers: Getting Started

A wrapper is necessary to enable a direct access to the driver's dynamic linked library (*dll*). This wrapper is automatically installed in the *~VXIplug&play\WinNT\include* (or *~VXIplug&play64\Win64\include*\) directory, where *VXIPNPPATH* (or *VXIPNPPATH64)* is the VXIplug&play environment variable pointing to your VXIplug&play installation directory. Please note, that for the instruments that in the meantime have a IVI.NET – a native .NET instrument driver available, the *.vb Visual Basic.NET wrapper file is no more available in the VXIplug&play driver installation. We encourage the user to switch to the native .NET instrument driver.

How to create a new application project in Microsoft Visual Studio 20xy:

Create a new project, e.g. File -> New -> Project

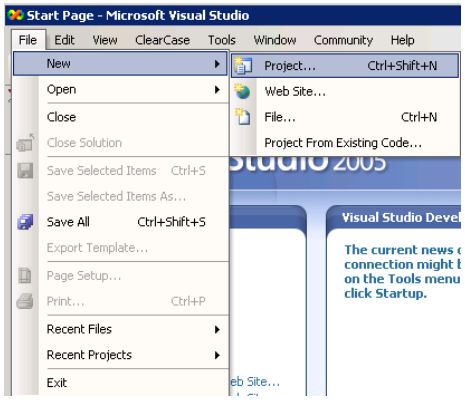

Figure 5: Creating a new project in Visual Studio.

Select your programming language and your template type, e.g. *Visual Basic Windows Application* or *Console Application*.

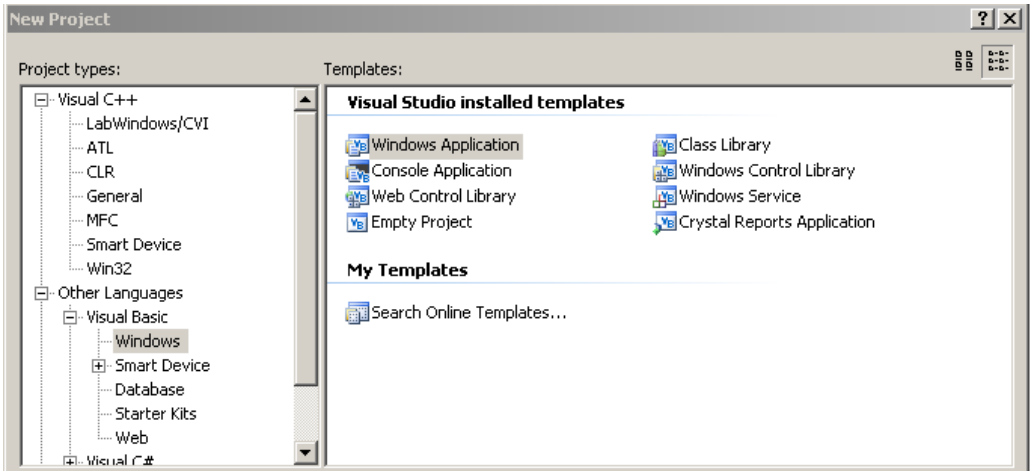

Figure 6: Selecting your desired template.

To access the instrument driver, you have to include the provided VXIplug&play wrapper for VB.NET, e.g. *Project->Add Existing Item… ~VXIplug&play\WinNT\include\rsspecan.vb* or (*~VXIplug&play64\Win64\include\rsspecan64.vb*)

Note: It is very important to explicitly specify the target platform for the solution using the Configuration Manager. Either *x86* for 32-bit application or *x64* for 64-bit applications.

Also the driver's namespace has to be added to the current source code file via the directive "*Imports WindowsApplication1.rsspecan*" (where *rsspecan* can be seen as an example driver). The result is shown in Fig. 7:

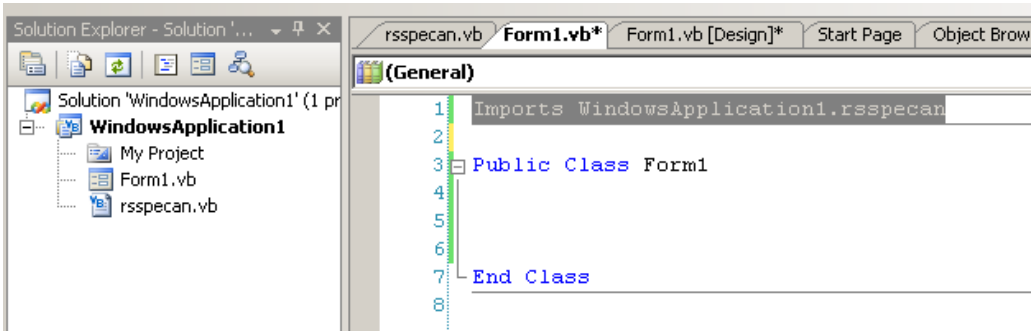

Figure 7: Adding the rsspecan driver to your current project.

Now the development environment is configured in a proper way. For getting your application development started, please refer to the examples provided on your instrument's driver download site:

http://www.rohde-schwarz.com/drivers.

3.3.3 C++ and Instrument Drivers: General Hints

Problem: The header file *rsXYZ.h* was not found while compiling
Solution: Set your VXIplug&play include directory in your Visual Studio Project properties (see Fig. 8). The include directory is the *~VXIplug&play\WinNT\include* (or *~VXIplug&play64\Win64\include*\), where *VXIPNPPATH* (or *VXIPNPPATH64)* is your VXIplug&play environment variable pointing to your VXIplug&play installation directory.

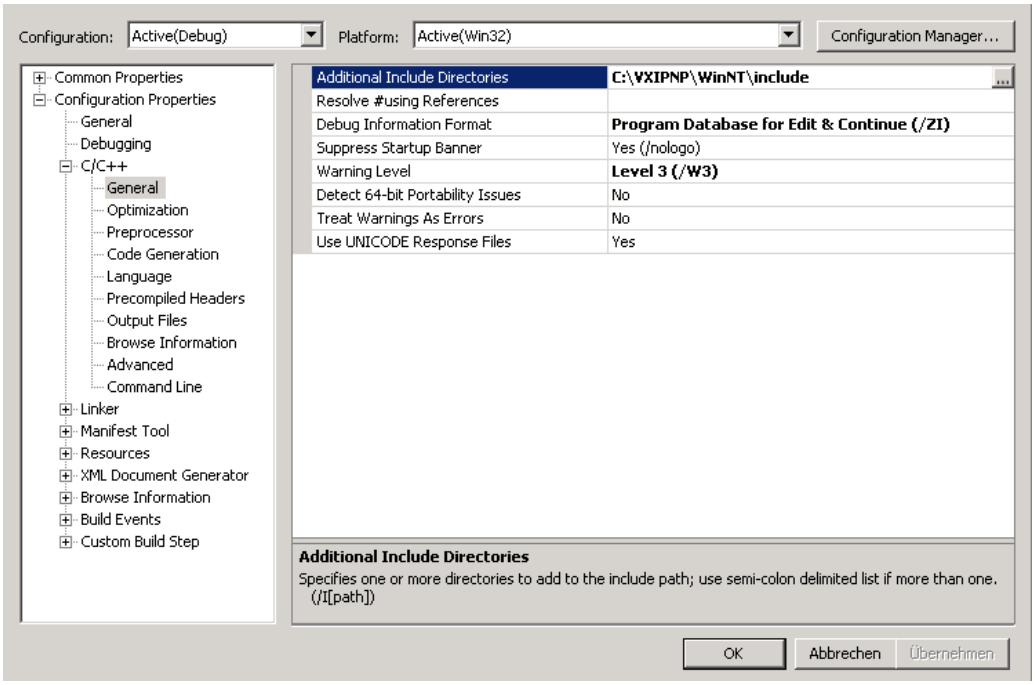

Figure 8: C++ Compiler settings for instrument driver header files.

Problem: Linker error while building your software project.
Solution: Specify your VXIplug&play libraries in your Visual Studio Project properties (see Fig. 9). Your specific library *rsXYZ.lib* can be found in *~VXIplug&play\WinNT\library\msc*, where *VXIPNPPATH* is your VXIplug&play environment variable pointing to your VXIplug&play installation directory.

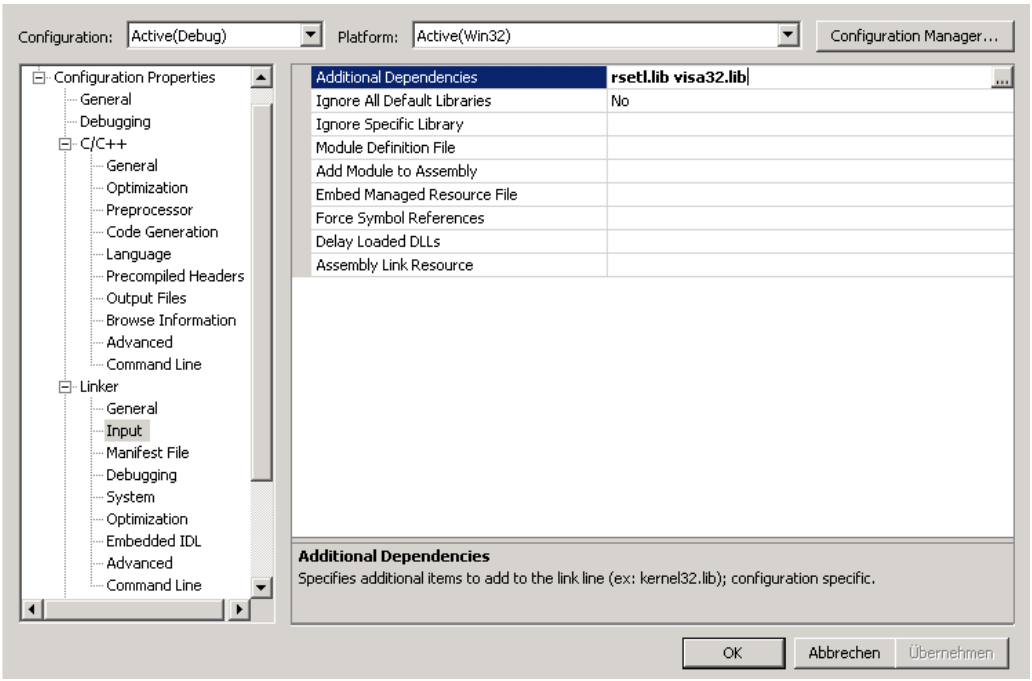

Figure 9: C++ Linker settings for instrument driver libraries.

Problem: The library *rsXYZ.lib* was not found while building your software project.

Solution: Set your VXIplug&play library path in your Visual Studio Project properties (see Fig. 10). Your library directory is the *~VXIplug&play\WinNT\library\msc*, where *VXIPNPPATH* is your VXIplug&play environment variable pointing to your VXIplug&play installation directory.

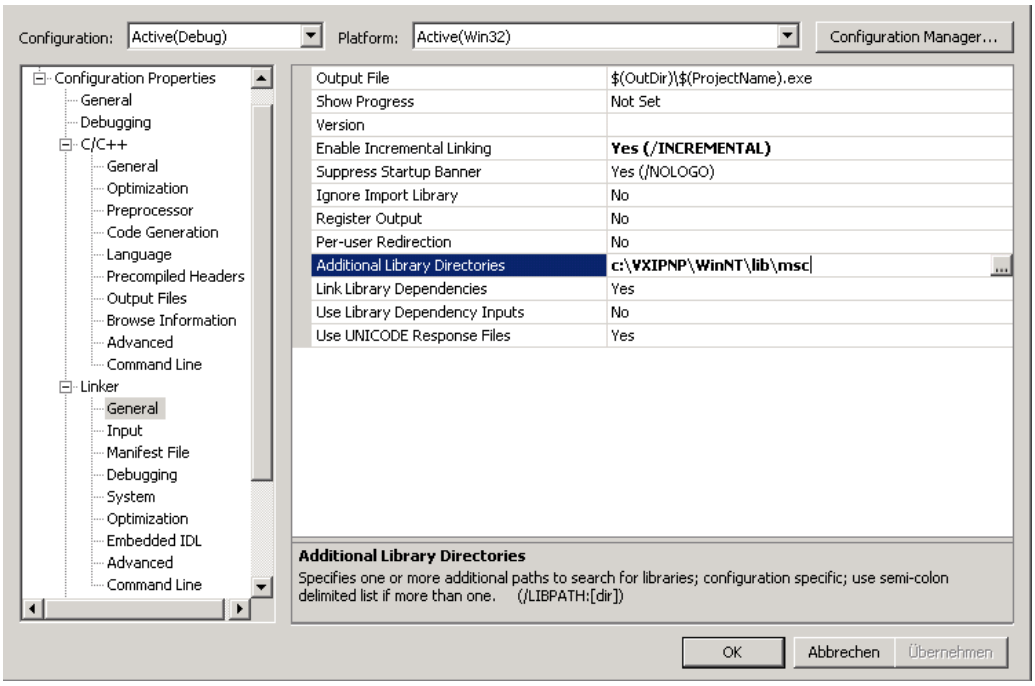

Figure 10: C++ Linker settings for instrument driver libraries.

Problem: The instrument was not found and VISA resource string in NI-Spy has only the size of one character
Solution: Correct your character settings in your Visual Studio Project properties (see Fig. 11).

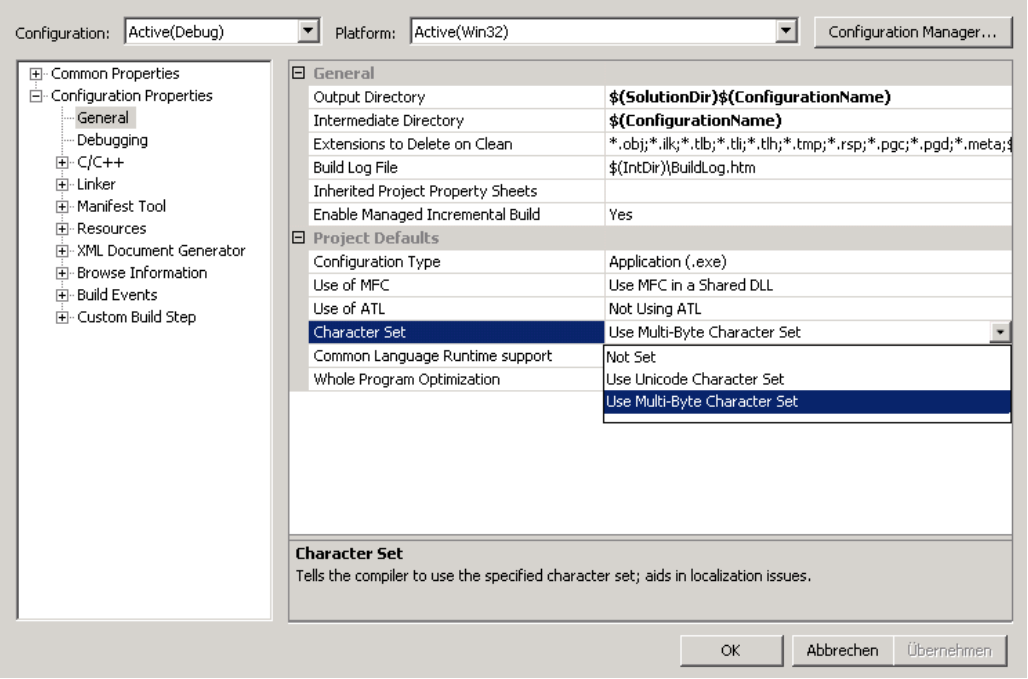

Figure 11: Correct character settings for Visual Studio 2008

3.3.4 C++ and SCPI Commands: Getting Started

This section will help developers set up their development environment for easily getting started without using instrument drivers.

After creating a new project, it is important to enable your development environment's linker to access the VISA library. The following figures show a possible configuration. Note that this depends on your host PC VXIplug&play installation.

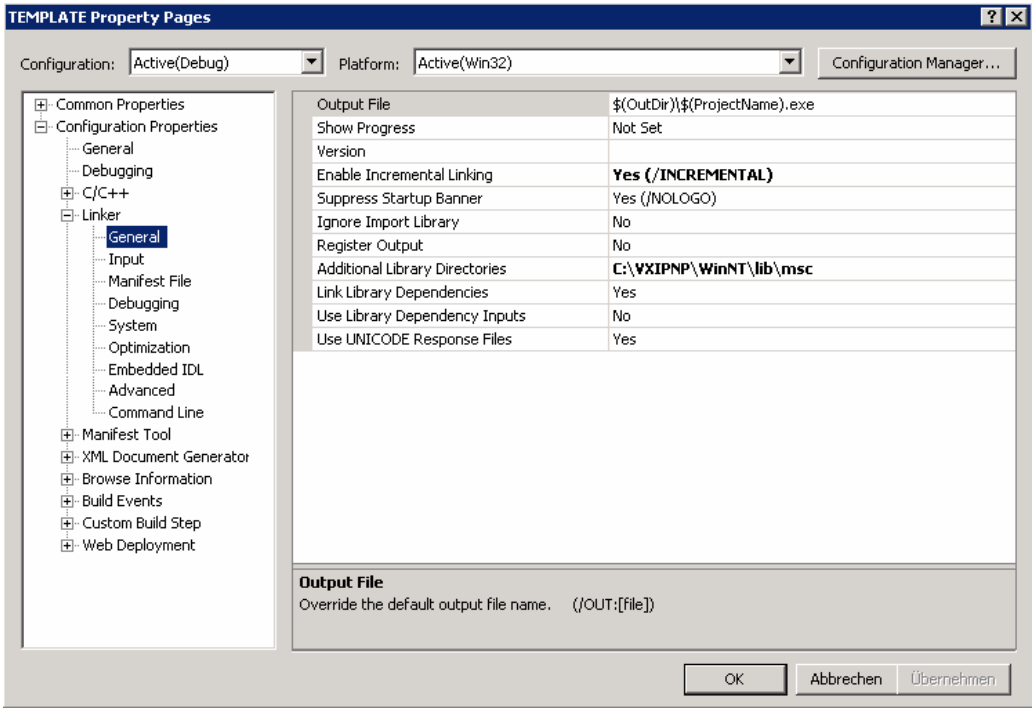

Figure 12: Setting up the "Additional Library Directory".

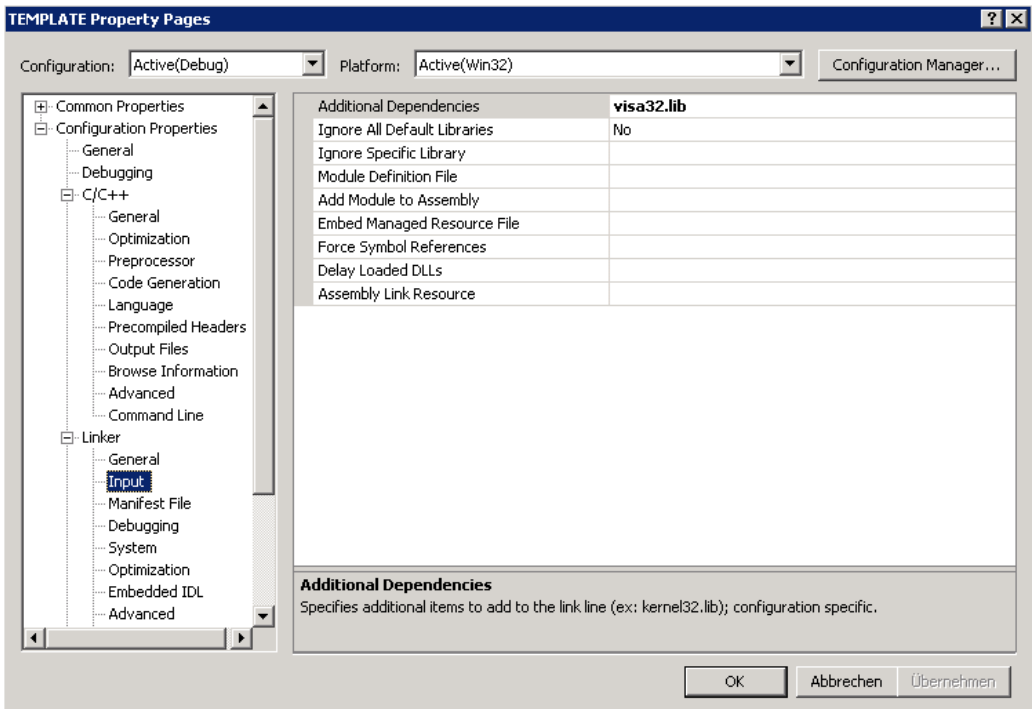

Figure 13: Setting up linker parameter "Additional Dependencies".

4 General Programming Hints

The following is referring to the *rsspecan* instrument driver. The described procedures are all adaptable to other ROHDE & SCHWARZ instrument drivers, only the naming of the files can be different. The naming convention is *rsXYZ*, where *XYZ* is the abbreviation of the instrument's (-family) driver.

4.1 How to Disable Option Checking

In the LabWindows/CVI instrument driver the command is called:

e.g.
```
rsXYZ_setCheckOption(ViSession instrumentHandle,
                     ViBoolean optionChecking);
```

In the LabVIEW instrument driver the command is called:

e.g. `rsXYZ Option Checking.vi`

where *XYZ* is instrument driver-specific. Please note that this function is not available in all instrument drivers.

Note: In the *Instrument Driver Tree Structure->Utilities* folder of the instrument driver's *chm* help file, you can check whether the command is available.

4.2 How to Disable Error/Status Checking

In the LabWindows/CVI instrument driver the command is called:

e.g.
```
rsXYZ _setCheckStatus (ViSession instrumentHandle, ViBoolean state);
```

In the LabVIEW instrument driver the command is called:

e.g. `rsXYZ Instrument Status Checking.vi`

where *XYZ* is instrument driver-specific. Please note that this function is not available in all instrument drivers.

Note: In the *Instrument Driver Tree Structure->Utilities* folder of the instrument driver's *chm* help file, you can check whether the command is available.

4.3 How to Disable Range Checking

While initializing a driver session, the range checking functionality can be disabled using an option string. Further details can be found inside the driver's *chm* help file.

Instead of using the `rsXYZ_Init (…)` function to initialize a driver session, a second function is available. In the LabWindows/CVI instrument driver the function is called:

e.g. `rsXYZ_InitWithOptions (ViRsrc resourceName, ViBoolean idQuery, ViBoolean resetDevice, ViString optionString, ViSession* instrumentHandle);`

where *XYZ* is instrument driver specific.

In the LabVIEW instrument driver the express VI *rsspecan_core_attribute_express Source.vi* can be used to set the attribute *RS_ATTR_RANGE_CHECK*. How to modify attributes in LabVIEW is described in http://www.rohde-schwarz.com/appnote/1MA170.html.

Note: In the *Instrument Driver Tree Structure->Utilities* folder of the instrument driver's *chm* help file, you can check whether the command is available. This function is not available in all instrument drivers.

4.4 How to Speed up the Driver Execution

In normal operating mode a status check is performed after every command. It is strongly recommended to utilize this functionality to identify errors fast and reliably.

To improve execution time in many drivers, error status checking can be disabled. To do so, disable error/status checking; see section 4.2 for details.

5 Hints on How to Get Started with Attribute-Based Instrument Drivers

5.1 What Are Attribute-Based Instrument Drivers?

The concept of attribute-based drivers is that nearly every SCPI command is accessible via an attribute. Its modularity and the possibility to access almost every instrument property via attributes provides a very sophisticated way of programming T&M applications.

Attribute-based drivers can easily be identified: All attribute-based instrument drivers have an additional *Driver Attribute Help* file (*rsXYZ_attr.chm*) within the driver's installation directory.

It is important to realize that not every ROHDE & SCHWARZ instrument driver is an attribute-based driver.

Further information about attribute-based instrument driver can be found here: 1MA170: Introduction to Attribute Based Instrument Drivers.

5.1.1 How to Run a Scan Measurement with the R&S®ESL EMI Test Receiver and the rsspecan Instrument Driver

Find the necessary commands in the *rsspecan_vxi.chm* help file. Crucial for a continuous reading of the scan trace is the following function:

```
rsspecan_ConfigureReceiverTraceFeedControl
                                (instrSession, RSSPECAN_VAL_TRAC_FEED_ALWAYS);
```

For LabVIEW applications the command is called:

```
rsspecan Configure Receiver Trace Feed Control.vi
```

For further information, please refer to the scan measurement example application on the instrument's driver download site.

6 Related Document

- Application Note 1MA170: Introduction to Attribute Based Instrument Drivers
  http://www.rohde-schwarz.com/appnote/1MA170.html

  - Application Card: Simplify software development for measurement applications
    http://cdn.rohde-schwarz.com/dl_downloads/dl_driver/remote_control/Remote_Control_Drivers_Simplify_software_development_en.pdf

7 Resources

- ROHDE & SCHWARZ Instrument Driver Download Site: http://www.rohde-schwarz.com/drivers
- ROHDE & SCHWARZ Remote Control Knowledge Base: http://www.rohde-schwarz.com/rckb
- National Instruments VISA: http://www.ni.com/visa/
- National Instruments LabVIEW: http://www.ni.com/labview/
- National Instruments LabWindows/CVI: http://www.ni.com/lwcvi/
- IVI Foundation: http://www.ivifoundation.org/

Accessed: August 18, 2010

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 more than 75 years ago, Rohde & Schwarz has a global presence and a dedicated service network in over 70 countries. Company headquarters are in Munich, Germany.

Environmental commitment
- Energy-efficient products
- Continuous improvement in environmental sustainability
- ISO 14001-certified environmental management system

Certified Quality System
ISO 9001

Regional contact

USA & Canada
USA: 1-888-TEST-RSA (1-888-837-8772)
from outside USA: +1 410 910 7800
CustomerSupport@rohde-schwarz.com

East Asia
+65 65 13 04 88
CustomerSupport@rohde-schwarz.com

Rest of the World
+49 89 4129 123 45
CustomerSupport@rohde-schwarz.com

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

www.rohde-schwarz.com