Development Hints and Best Practices for Using Instrument Drivers

Application Note

Products:

| All Instrument Drivers

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

Juergen Straub

12.2009-1MA153_1e

Application Note

ROHDE&SCHWARZ

Table of Contents

1 Preface

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

It does not describe any programming languages nor, 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, which simplify the remote control of your measurement instrument. Through the input and output (I/O) abstraction via hiding the SCPI[1] commands a much more easier way of programming is possible. As buzzword *command completion*[2] of integrated development environments (IDE) can be mentioned. This feature is not available for SCPI commands.

R&S instrument drivers are based on Virtual Instrument Software Architecture (VISA[3]) which enables an physical independent connection between the host PC and the instrument.

Generally instrument drivers allow greater system flexibility and accelerate development time for instance by the possibility of reusing software. Major advantage is the reduction of the development time for any kind T&M application due to the fact of using a high level programming language in contrast to implement an application with SCPI commands. For instance all string formatting which is necessary when using SCPI commands is already done by the instrument driver.

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 great extent. It is no longer necessary to search through manuals looking for commands; this has already be 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] "Command completion" of current integrated development environments offers the possibility to safe time and avoid typing errors during programming (e.g. in C++ or C#)

[3] Virtual Instrument Software Architecture (VISA) as Host PC software a well defined hardware abstraction layer and I/O programming interface for various interfaces (e.g. General Purpose Interface Bus (GPIB) or Ethernet)

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

Example: Simple spectrum analysis
Even for simple spectrum analysis it starts to be quite complicated and error prone using SCPI commands. 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 it is possible to easily process the read out data within the T&M application.

As one can imagine, for more complex T&M application, like mobile standard generation (e.g. for Wideband Code Division Multiple Access (WCDMA) Signals) or its measurement, using SCPI commands is quite time consuming. Therefore setting up your instrumentation using instrument drivers 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 due to the fact that debugging and error diagnostics is well supported by the instrument driver. So it is possible to easily get rid of wrong configurations as well as not understood commands. This is done by extensively using the instruments error reporting functionality. For further details please have a look in chapter 4.

During the lifetime of T&M applications requirements can change, also for this reason instrument drivers can help you modifying a application. The logical grouping of your instruments functionality allows one easily to extend or modify your application on a abstract and logical level.

The described pros to simplify your T&M task is also a reason why reuse of already written software can be realised. Due to the concept of instrument family drivers, software written for research and development reasons on a high end instrument like the Vector Signal Generator R&S SMU200A can easily be reused for production lines which are equipped with e.g. the mid-range Vector Signal Generator R&S SMBV100A. The instrument driver abstracts the instrument, so this software can often be reused without any changes. Of cause the functionality needs to be supported on both instruments.

2.3 VISA and physical interfaces

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

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

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

All Rohde & Schwarz instrument drivers requires at least National Instruments VISA (NI-VISA) revision 3.0 or higher. For Linux it is strongly recommended to use the latest version of NI-VISA. Also the usage of Agilent I/O Library Suite[4] is possible.

2.4 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 enables a very sophisticated way of T&M application programming.

Attribute based drivers can easily be indentified: All attribute based instrument drivers has an additional *Driver Attribute Help* file (*rsXXX_attr.chm*) within the drivers installation directory.

For using attributes to write more flexible application, please see chapter 4.5 and chapter 4.6 for further explanation.

It is important to realize that not every R&S instrument drivers is an attribute based driver.

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

2.5 What is the RsSpecAn instrument driver?

The new generation of R&S attribute based spectrum analyzer instrument drivers (RsSpecAn) gives great flexibility and ease of use by combining low-level access to nearly every SCPI command via attribute and well balanced high level functions and VIs, combining several attributes. For those who don't want to learn new attribute concept, complete functionality of instrument is covered by high level commands. In other words, you may use attributes, but you don't have to. Of course it is possible to combine both low level attributes and high level functions together exactly according to needs of every specific application.
Comparing to older family instrument drivers (so called legacy drivers), the source code of both LabWindows/CVI and LabVIEW RsSpecAn instrument drivers are divided into modules/folders according to personalities. This gives opportunity to easily get rid of non-used functions and VIs.

2.6 Are R&S instrument drivers compatible with the VXIPnP driver standard?

Yes. For further information please have a look into the *readme.txt* in side your instrument driver's installation directory.

2.7 Where to find the VXIPnP installation directory?

This path is defined via an environment variable on your operating system, which can be read out in the environment variable "*VXIPNPPATH*".

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

Note: With NI-VISA Version 4.2 the default installation path changed from *C:\VXIPNP* to *C:\Program Files\IVI Foundation\VISA.*

2.8 How to install R&S instrument drivers on Windows?

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

http://www2.rohde-schwarz.com/en/service_and_support/Downloads/Drivers

Further specific steps and hints for installing instrument driver are available on the driver download site available at the product internet site.

3 Best practice for software development

3.1 Getting started

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

http://www2.rohde-schwarz.com/en/service_and_support/Downloads/Drivers/

to get an idea of how 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 getting started it is applicable to use the *Instruments Driver Help* (rsXX*X[_vxi].ch*m) file inside your instruments driver's installation directory in combination with our instruments operating manual remote control command chapter.

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

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

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

Which information can be found inside the *Instrument Driver Help*, also *Attribute Driver Help* (*rsXXX[_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 certain 3GPP standard (Instrument Driver Tree Structure)
- Possibility to include instrument help to search for instrument and driver functionality at one time (Please note: this is not in available in every driver)
- For attribute based drivers (with rsXXX_attr.chm): Mapping of attributes and driver calls (see figure below)

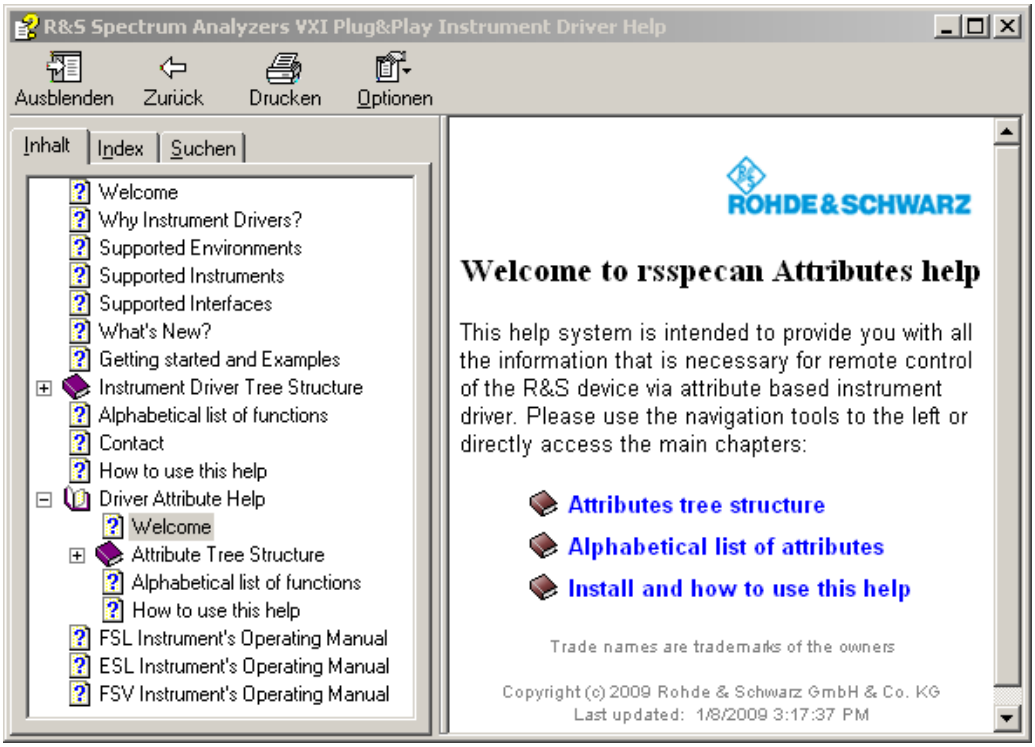

Figure 1: Mapping of attributes and driver calls within the Driver Attribute Help.

For more information about the "Driver Attribute Help" please see chapter 3.4.

Note: This feature is not available in all R&S instrument drivers.

3.3 What is the purpose of the "Driver Attribute Help"?

For sophisticated programming the "Driver Attribute Help" as a chapter of the driver help file can help to identify instruments attributes. These low level configuration can be used to set or get instrument properties, which are normally encapsulated in high level driver functions. For some reasons this attributes need to be set or read out independently via get or set attribute functions.

How to use attributes is described in chapter 4.5 and chapter 4.6.

3.4 How to combine instrument manuals and the driver's compressed help files?

There is the possibility to include the "*Instrument Operating Manual*" inside the "*Instrument Driver Help*".
Please follow the instructions in the last chapter of your "*Instrument Driver Help*" file.
See for instance the RsSpecAn help in the figure below:

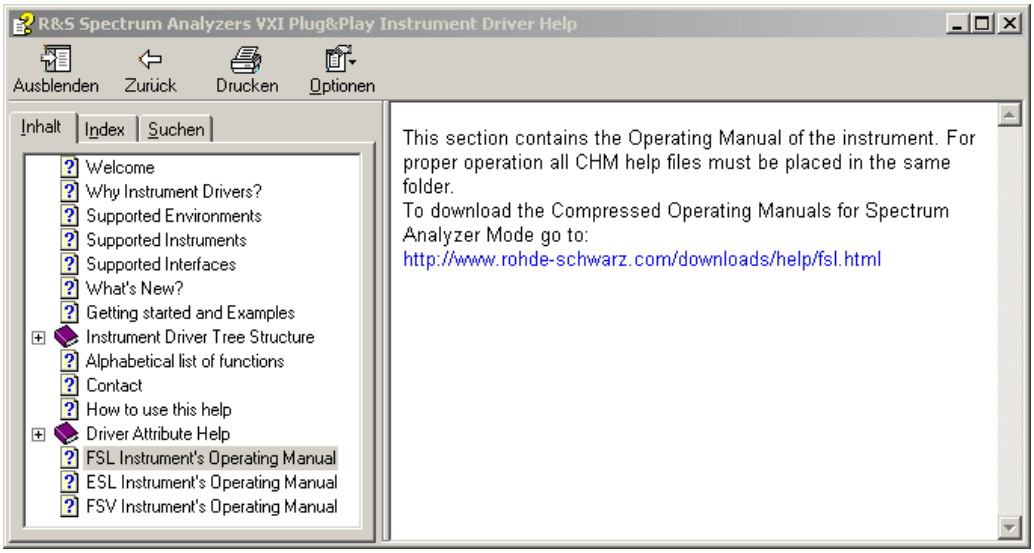

Figure 2: Where to download and include the Instruments Operating Manual.

Depending on your Windows operating system, it can be necessary to open the instruments operating manual once, to mark the file as trustworthy for the operating system.

Note: This feature is not available in all R&S instrument drivers.

3.5 How to integrate R&S instrument drivers into Microsoft Visual Studio 200x

The following examples are referring to the RsSpecAn instrument driver. The described procedures are all adaptable to other R&S instrument drivers, only the naming of the files can be different. The naming convention is *rsXXX*, where *XXX* is the abbreviation of the instrument's (-family) driver.

3.5.1 C# and instrument drivers: How to get started?

A wrapper is necessary to enable a direct access to the driver dynamic linked library (*dll*). This wrapper is automatically installed in the *~VXIPnP\WinNT\include* directory, where *VXIPNPPATH* is your VXIPnP environment variable pointing to your VXIPnP installation directory.

How to create a new T&M application project in Microsoft Visual Studio 200x:

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

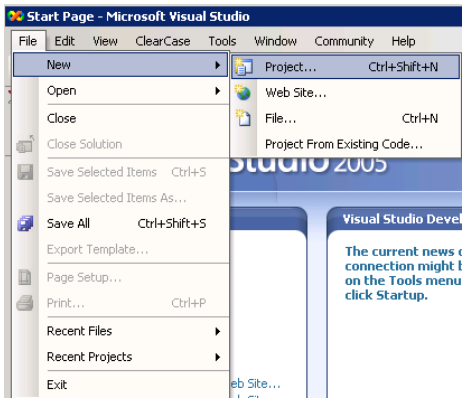

Figure 3: Create a new project in Visual Studio

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

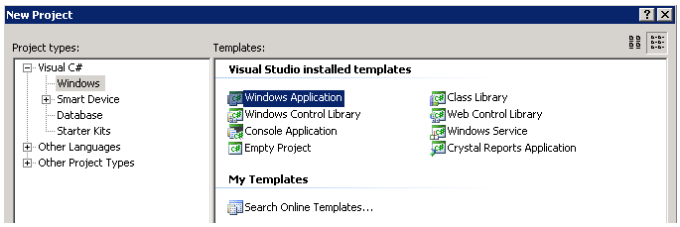

Figure 4: Select your desired template

To access the instrument driver, it is necessary to include the provided VXIPnP wrapper for *C#,* e.g. *Project->Add Existing Item… ~VXIPnP\WinNT\include\rsspecan.cs*

Also the drivers namespace has to be included to the current source code file for example via the directive *"using InstrumentDrivers"*. The result is shown in the picture below:

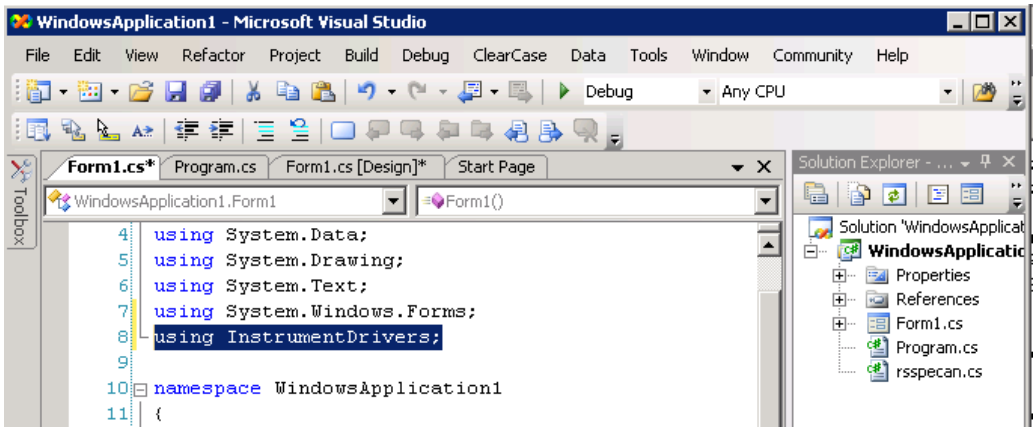

Figure 5: Include the RsSpecAn driver to your current project.

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

http://www2.rohde-schwarz.com/en/service_and_support/Downloads/Drivers/.

3.5.2 VB.NET and instrument drivers: How to get started?

A wrapper is necessary to enable a direct access to the driver dynamic linked library (*dll*). This wrapper is automatically installed in the *~VXIPnP\WinNT\include* directory, where *VXIPNPPATH* is your VXIPnP environment variable pointing to your VXIPnP installation directory.

How to create a new T&M application project in Microsoft Visual Studio 200x:

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

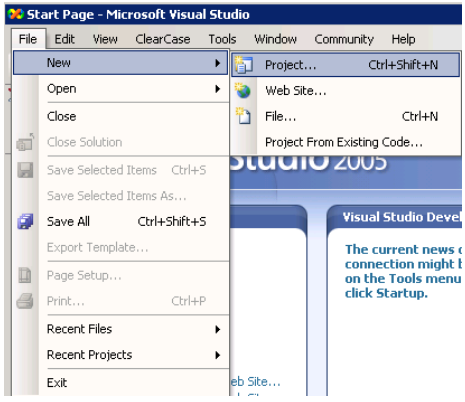

Figure 6: Create a new project in Visual Studio

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

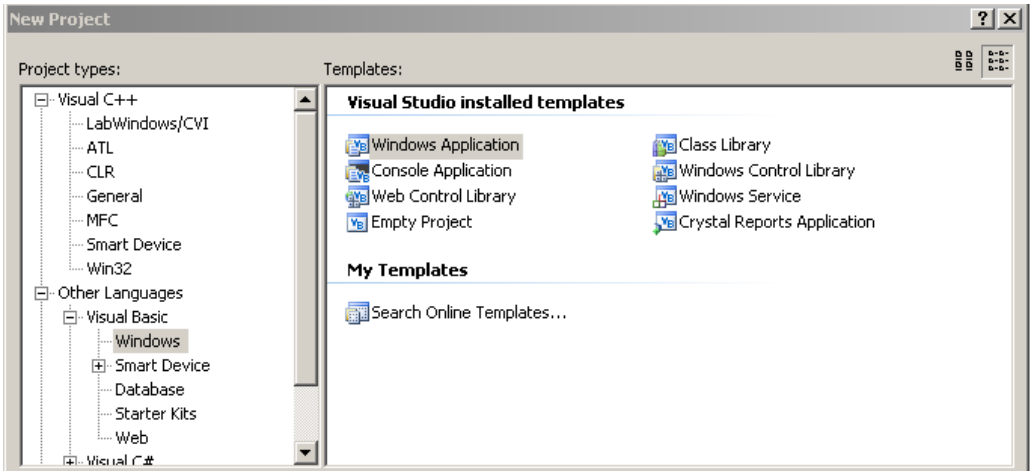

Figure 7: Select your desired template

Also the drivers namespace has to be included to the current source code file via the directive "*Imports WindowsApplication1.rsspecan*" (where *rsspecan* can be seen as example driver). The result is shown in the picture below:

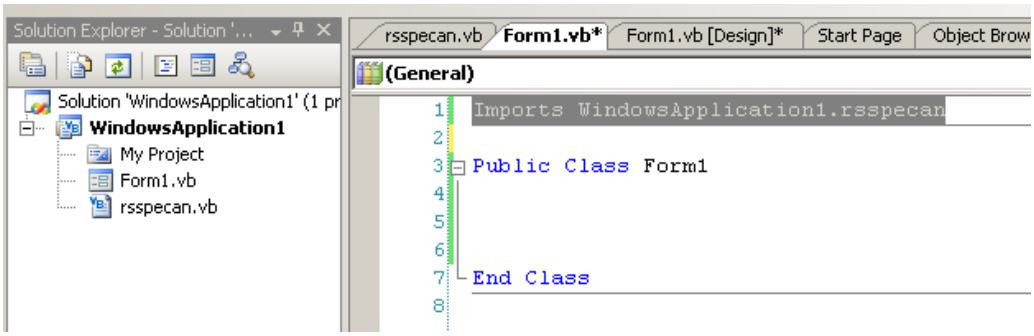

Figure 8: Include the RsSpecAn driver to your current project.

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

http://www2.rohde-schwarz.com/en/service_and_support/Downloads/Drivers/.

3.5.3 C++ and instrument drivers: General hints

Problem: The header file *rsXXX.h* was not found while compiling
Solution: Please set your VXIPnP include directory in your Visual Studio Project properties (see picture below). Your include directory is the *$VXIPNPPATH$\WinNT\include*, where *VXIPNPPATH* is your VXIPnP environment variable pointing to your VXIPnP installation directory.

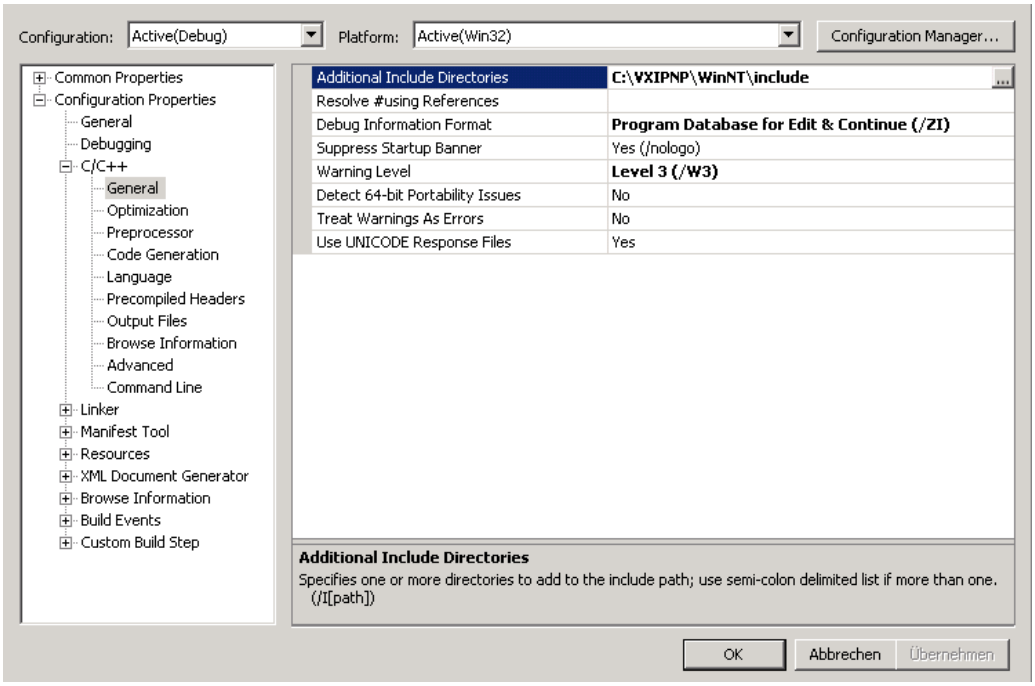

Figure 9: C++ Compiler settings for instrument driver header files.

Problem: Linker error while building your software project.
Solution: Please specify your VXIPnP libraries in your Visual Studio Project properties (see picture below). Your specific library *rsXXX.lib* can be found in *$VXIPNPPATH$\WinNT\library\msc*, where *VXIPNPPATH* is your VXIPnP environment variable pointing to your VXIPnP installation directory.

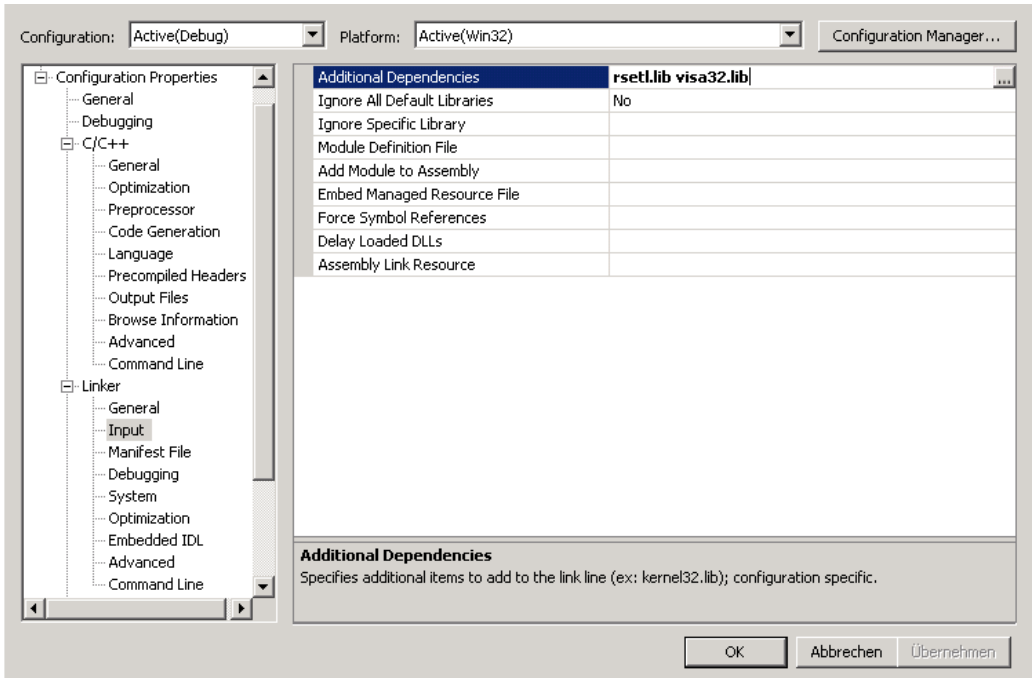

Figure 10: C++ Linker settings for instrument driver libraries.

Problem: The library *rsXXX.lib* was not found while building your software project.
Solution: Please set your VXIPnP library path in your Visual Studio Project properties (see picture below). Your library directory is the *$VXIPNPPATH$\WinNT\library\msc*, where *VXIPNPPATH* is your VXIPnP environment variable pointing to your VXIPnP installation directory.

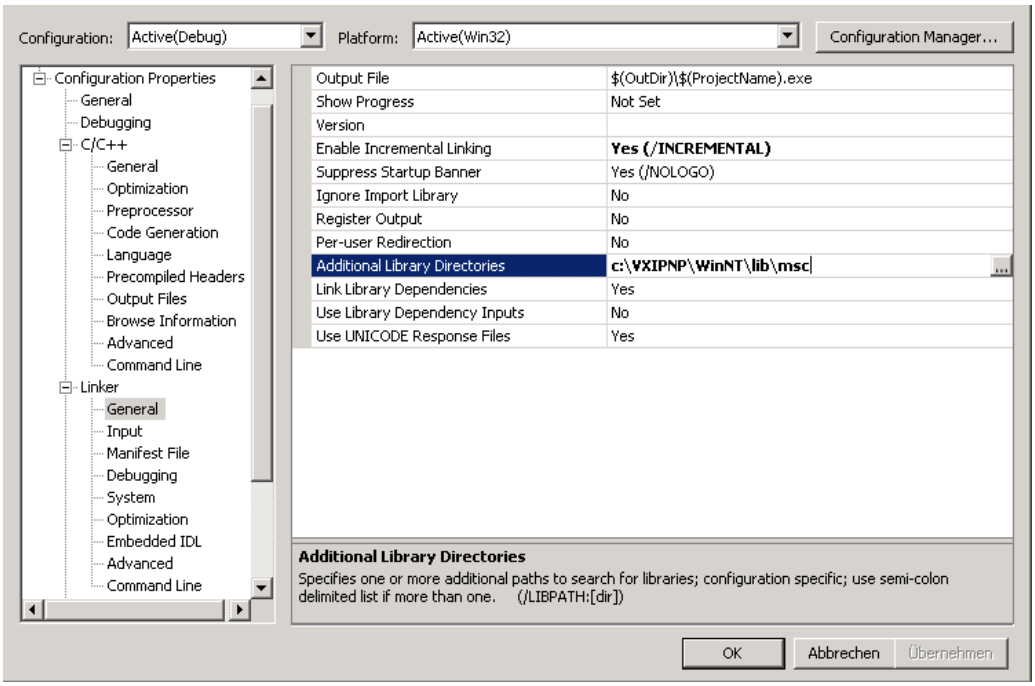

Figure 11: C++ Linker settings for instrument driver libraries.

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 picture below).

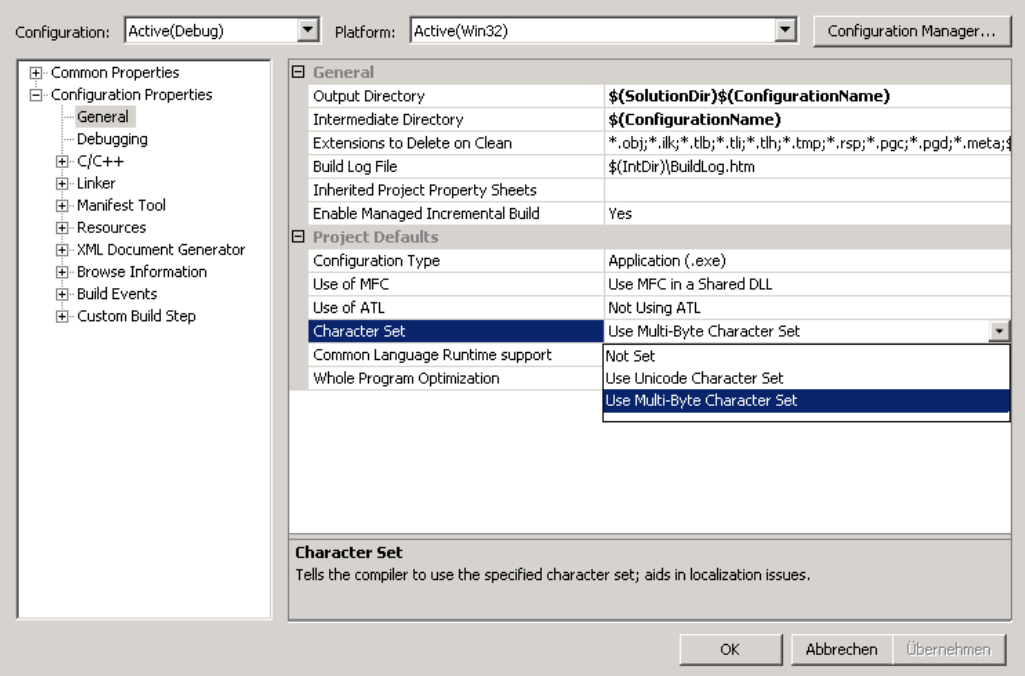

Figure 12: Correct character settings for Visual Studio 2009

3.5.4 C++ and SCPI commands: How to get started?

This chapter shall help developers to set up their development environment to 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 figures below are showing a possible configuration. Note that this depends on your host PC VXIPnP installation.

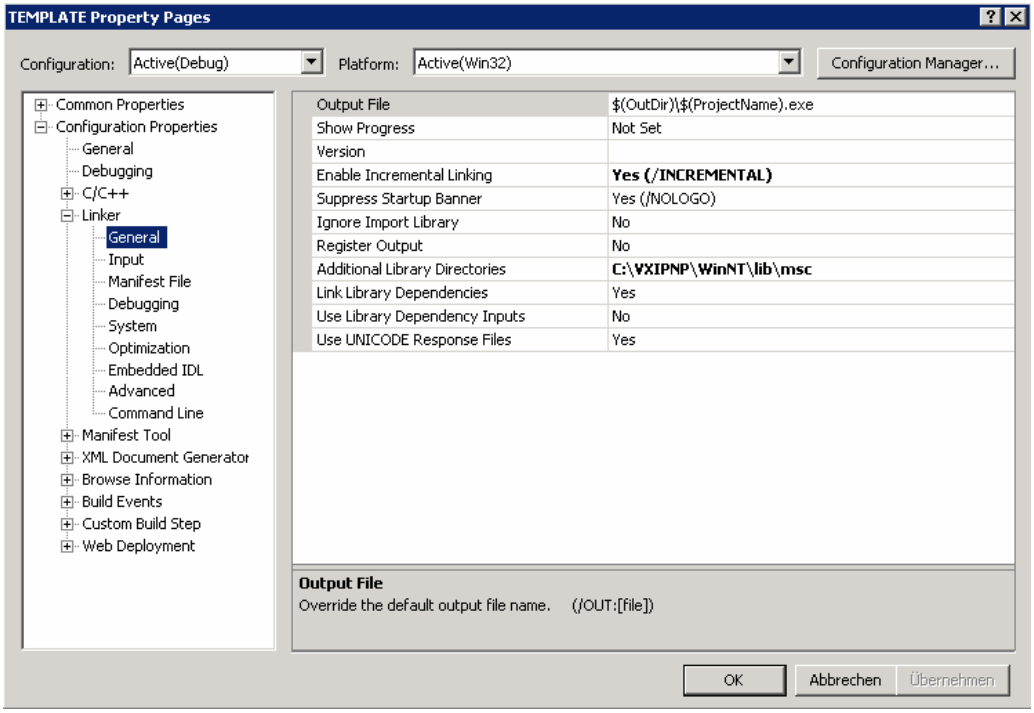

Figure 13: Setting up the "Additional Library Directory"

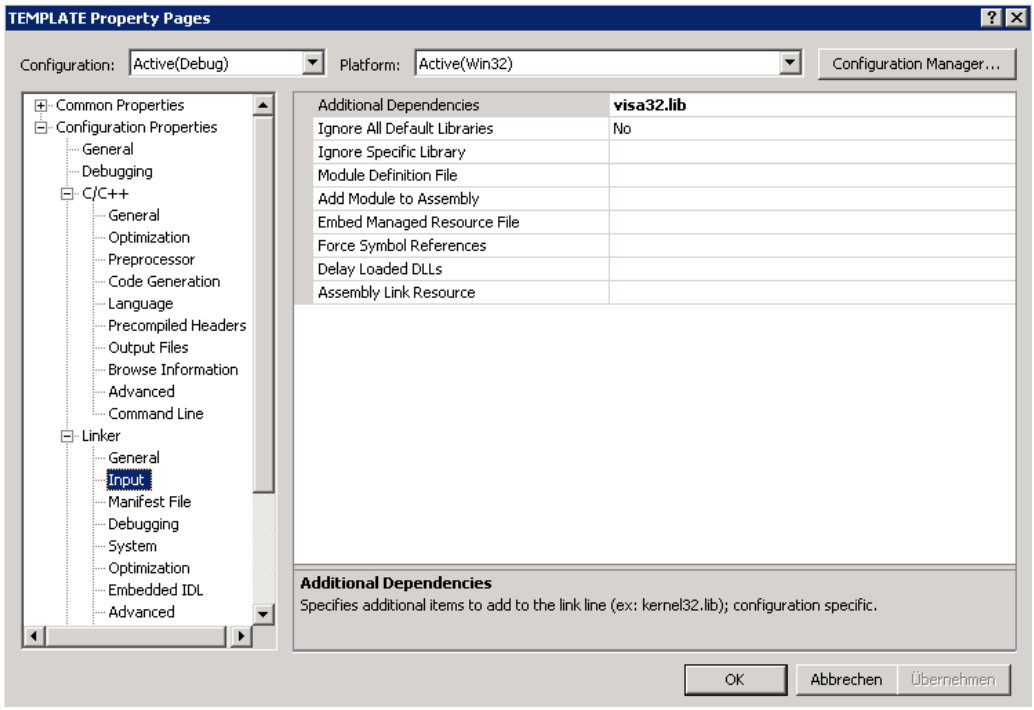

Figure 14: Setting up linker parameter "Additional Dependencies"

4 General programming hints

The following is referring to the RsSpecAn instrument driver. The described procedures are all adaptable to other R&S instrument drivers, only the naming of the files can be different. The naming convention is *rsXXX*, where *XXX* 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. `rsXXX_setCheckOption(ViSession instrumentHandle,`
`ViBoolean` **`optionChecking`**`);`

In the LabVIEW instrument driver the command is called:

e.g. `rsXXX Option Checking.vi`

Where *XXX* is instrument driver specific. Please note that this function is not available in all instrument drivers.

Note: Inside the folder "*Instrument Driver Tree Structure->Utilities*" 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. `rsXXX _setCheckStatus (ViSession instrumentHandle, ViBoolean state);`

In the LabVIEW instrument driver the command is called:

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

Where *XXX* is instrument driver specific. Please note that this function is not available in all instrument drivers.

Note: Inside the folder "*Instrument Driver Tree Structure->Utilities*" 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 instrument *chm* help file.

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

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

Where *XXX* 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 chapter 4.5.

Note: Inside the folder "*Instrument Driver Tree Structure->Utilities*" 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 operation mode a status check is preformed after every command. It is strongly recommend to utilize this functionality to identify errors fast and reliable.

For improving execution time in many drivers it is possible to disable error status checking. To do so, please disable error/status checking, see chapter 4.2 for details.

4.5 How to use attributes in LabVIEW?

The following example is referring to the RsSpecAn instrument driver. The described procedures are all adaptable to other attribute based R&S instrument drivers, only the naming of the files can be different. The naming convention is *rsXXX*, where *XXX* is the abbreviation of the instrument's (-family) driver.

This chapter shows how to use attributes in the RsSpecAn driver. This feature is useful for sophisticated instrument control.

Please also use the *Driver Attribute Help* chm file. This help file is accessible via the *Instruments Driver Help* (rsXX*X[_vxi].ch*m).

Example: How to set the *Frequency Start* value by attributes

To select an attribute use the provided Express VI, included in instrument driver package. This instrument driver Express VI can be found in:

User Libraries => Express User Libraries => rsspecan

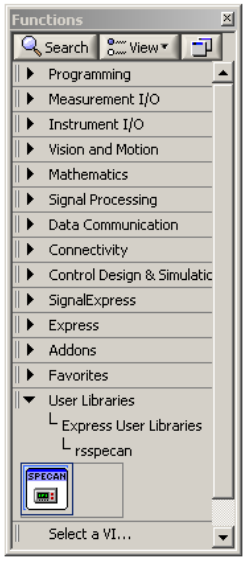

Figure 15: Palette Menu of RsSpecAn driver

Drag and drop the *rsspecan Attribute* in your block diagram. After placing the express VI the front panel of the Express VI will be opened (see Figure below).

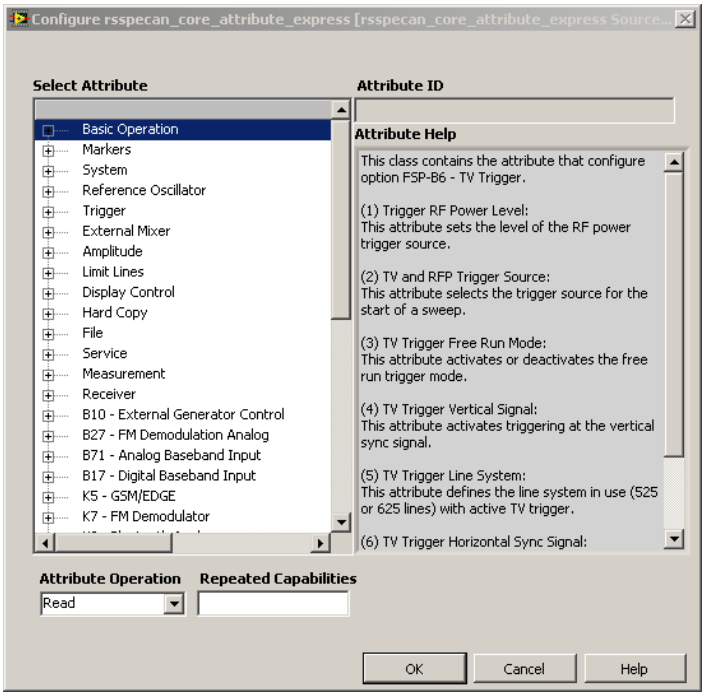

Figure 16: Front panel of rsspecan_core_attribute_express Source.vi

Now you can select the desired attribute from tree structured list. Each attribute can has a different access type (*read only, write only or read/write)*. Proper *read/write* operation can be selected by "*Attribute Operation*" in the bottom left corner. Repeated capabilities are set by using a string control at the bottom of the panel called "*Repeated Capabilities*". The attribute value is entered using standard LabVIEW controls for each data type (numeric, boolean, string or ring).

For example use in Basic Operation "*Set Frequency Start*" value:

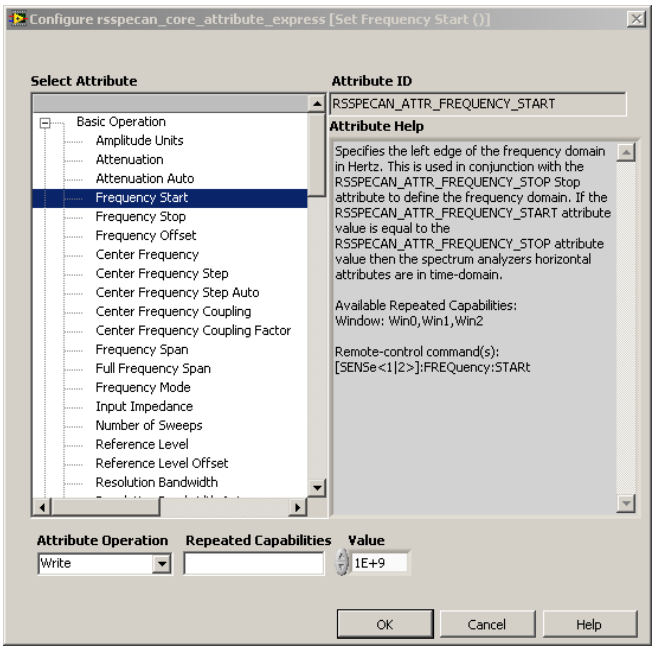

Figure 17: Configure "Set Frequency Start" example

1. Select the desired attribute. To find the designated attribute easily the *rsspecan.chm* help file can be used to look for it. Be aware of the fact that the tree structures in the help file and the Express VI are the same
2. Leave "*Repeated Capabilities*" empty
3. Type the desired frequency value in the text box "*Value*"
4. After clicking the "*OK*" button, a new instance attributed based VI is generated. The name of the VI and its predefined values and also the help will be adapted.

The result is shown in following figure.

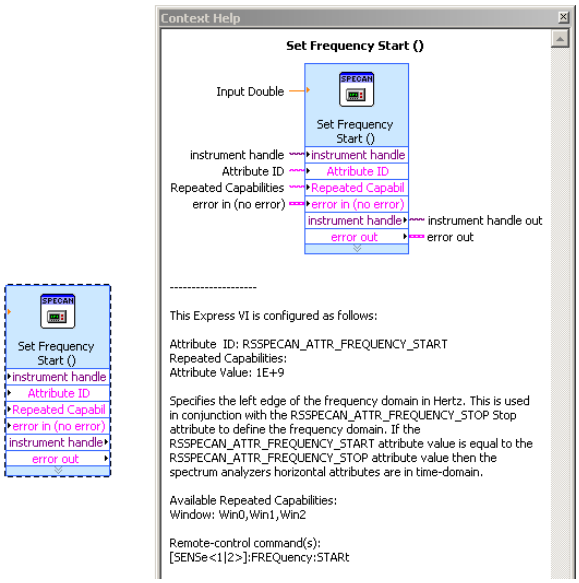

Figure 18: Instanced attribute based VI "Set Frequency Start"

4.5.1 Repeated capabilities

Many measurement instruments have capabilities which are duplicated. For example, an oscilloscope might have several channels with identical functionality or a power supply may have several outputs. For this reason repeated capabilities are introduced.

Repeated capability instances are specified by a string parameter for each property. It also can be specified by a function which selects the active instance.

To define the usage of particular capability fill the text box "*Repeated Capability*" in the front panel of the Express VI with the proper value. To use more than one repeated capability in one attribute separate them with a comma.

See a marker example using repeated capabilities pictured in the following figure:

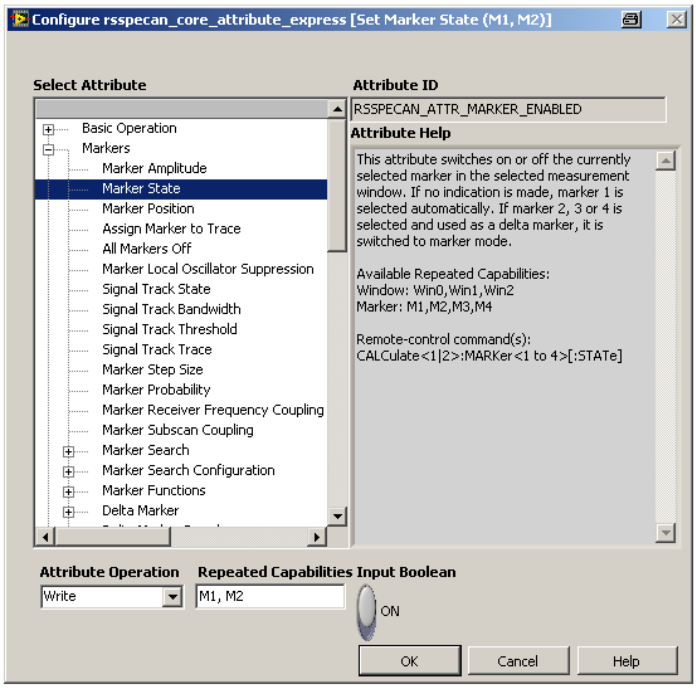

Figure 19: Example for "Repeated Capabilities"

The attribute and its features can be changed any time later by double clicking the VI, which recalls the shown Express VIs front panel.

4.6 How to use attributes in LabWindows/CVI?

The following example is referring to the RsSpecAn instrument driver. The described procedures are all adaptable to other attribute based R&S instrument drivers, only the naming of the files can be different. The naming convention is *rsXXX*, where *XXX* is the abbreviation of the instrument's (-family) driver.

The following chapter shows how to use attributes in the RsSpecAn driver. This is only necessary in rare cases, for instance if the driver is not supporting the functionality via high level API.

Please use the *Driver Attribute Help* chm file. This help file is accessible via the *Instruments Driver Help* (rsXX*X[_vxi].ch*m).

Note: The *rsspecan.sub* file has to be added to the project. This is mandatory for browsing attributes.

Example: How to set the *Frequency Start* value by attributes

To select an attribute use the provided function panels, included in instrument driver package. This instrument driver function panels can be found in:

Instruments=>R&S SpecAn=>Configuration=>Set/Get/Check Attribute/Repeated Capabilities

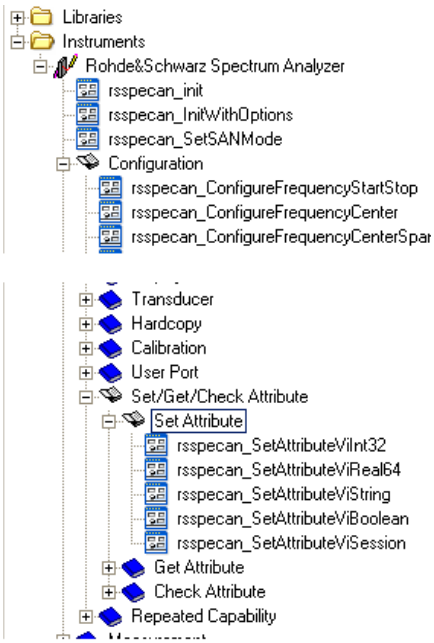

Figure 20: Function panel for RsSpecAn driver attributes

To open the function panel double click on the desired function. In the figure below the function panel of "*rsspecan_SetAttributeViReal64*" is pictured. By clicking on text box "'*Attribute ID*" the "*Select Attribute Constant*" window will be opened.

Now it is possible to select the desired attribute from tree structured list. Each attribute can have different access type (*read only, write only or read/write)*. Proper *read/write* operation can be selected by the proper function panel, either *Set Attribute*, *Get Attribute*, or *Check Attribute*.
Repeated capabilities are set by using identifier or identifier names within the function panel called *rsspecan_GetAttributeRepeatedCapabilityId[s|Names]*.

For example use in Basic Operation "*Set Frequency Start*" value:

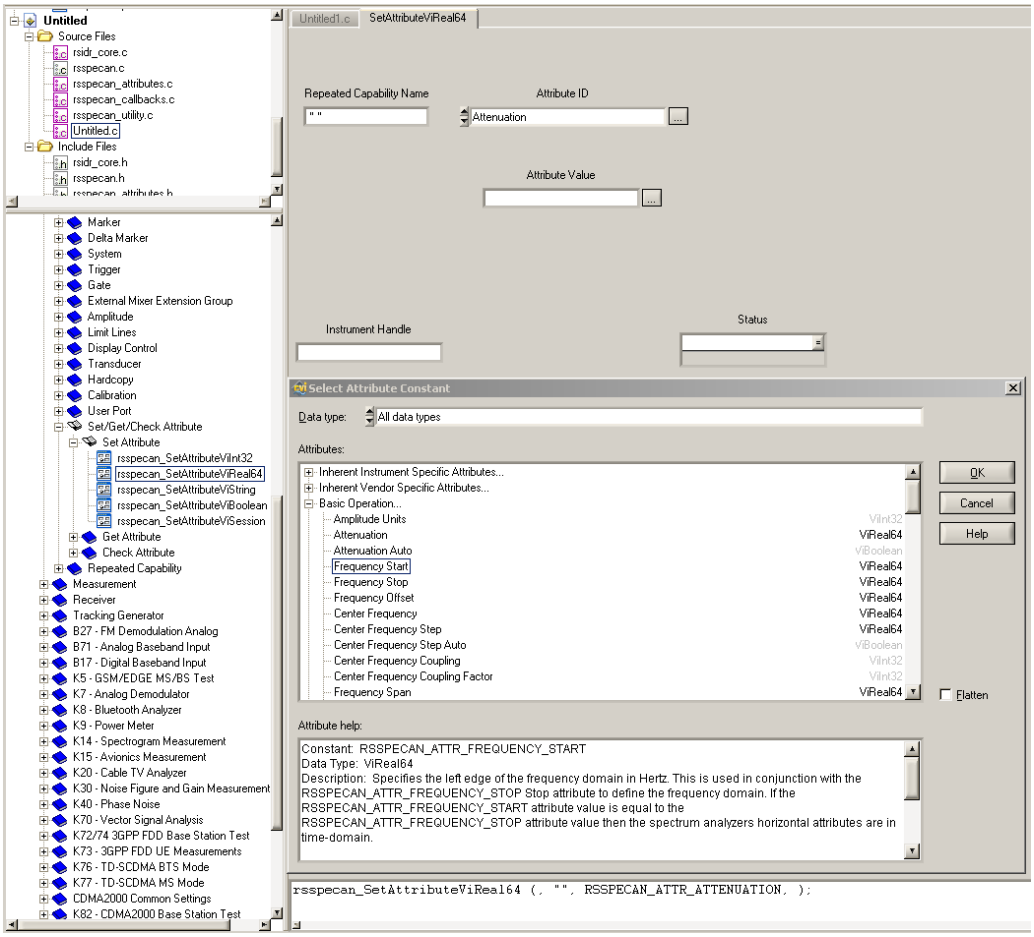

Figure 21: How to select a attribute in the "SetAttributeViReal64" function panel

1. Please select the desired attribute. To find the designated attribute easily the *rsspecan.chm* help file can be used to look for it. Be aware of the fact that the tree structures in the help file and the Express VI are the same
2. Leave "*Repeated Capabilities*" string empty
3. Type the desired frequency value in the text box "*Attribute Value*", e.g. "*1e+9*"
4. Enter a valid instrument handle in the text box "*Instrument Handle*"
5. It is strongly recommended to fetch the return value of your function, if you type the name of a declared integer variable in the text box "*Status*"
6. To include the function into your source code, you can copy the created function string out of the window on the button of the function panel and insert it into the source code.

The result is shown in following figure.

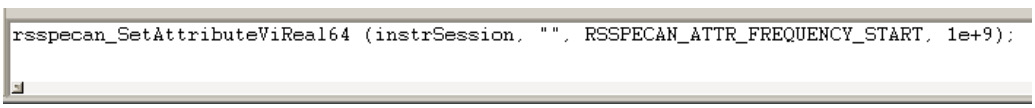

Figure 22: Build a function string on the button of the "rsspecan_SetAttributeViReal64" function panel

The attribute and its features can be changed any time later by right clicking the function and recall the function panel. See picture below:

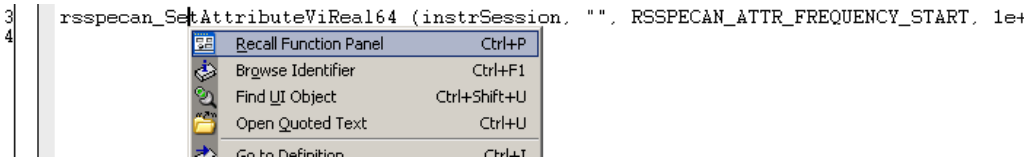

Figure 23: Recalling the function panel to modify the attributes

4.6.1 Repeated capabilities

Many instruments have capabilities which are duplicated. For example, an oscilloscope might have several channels with identical functionality or a power supply may have several outputs. For this reason repeated capabilities are introduced.

Repeated capability instances are specified by a string parameter for each property.

To define the usage of particular capability fill the text box "*Repeated Capability Id*" in the front panel with the proper value. To use more than one repeated capability in one attribute separate them with a comma.

See a marker example using repeated capabilities pictured in the following figure:

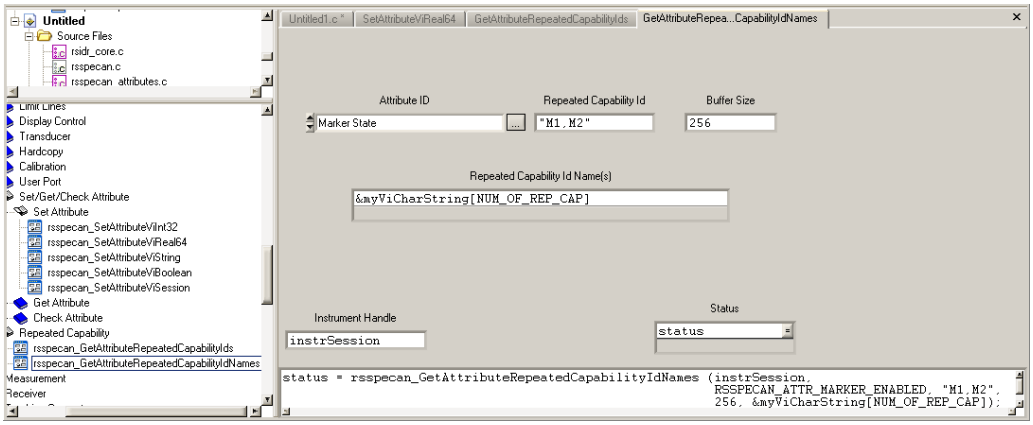

Figure 24: Example for "Repeated Capabilities" in LabWindows/CVI

4.7 How to use attributes in Agilent VEE?

Due to the fact that Agilent VEE[5] does not parse the *sub*-file corresponding to the VXIPnP standard, there is only the possibility to look for the corresponding attributes[6] in the *rsspecan_attr.chm* file and use the *Rohde&Schwarz Spectrum Analyzer-> Configuration -> Get/Set/CheckAttribute* function set.

[5] Tested with Agilent VEE Pro 9.0, see http://www.home.agilent.com/agilent/product.jspx?nid=-34095.806312.00&cc=US&lc=eng

[6] The following is referring to the RsSpecAn instrument driver. The described procedures are all adaptable to other attribute based R&S instrument drivers, only the naming of the files can be different. The naming convention is *rsXXX*, where *XXX* is the abbreviation of the instrument's (-family) driver.

5 RsSpecAn specific hints

5.1 How to run a scan measurement with the R&S®ESL EMI Test Receiver and the RsSpecAn instrument driver?

Please find necessary commands the *rsspecan_vxi.chm* help file. Crucial for a continuous reading of the scan trace is 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 have a look at the scan measurement example application on the instrument's driver download site.

6 Linux instrument drivers

Due to the fact that the Linux as desktop operating system is becoming more and more popular Rohde & Schwarz supports instrument drivers for Linux distributions.

The aim of the this chapters is to give a brief overview of how to get started with R&S instruments drivers on a Linux operating system.

6.1 Supported distributions

Rohde & Schwarz instrument drivers are build up on the VISA standard and thereafter they are compatible to the ANSI-C99 standard[7]. National Instruments VISA installation package, which is crucial for using Rohde & Schwarz instrument drivers, is currently supporting[8] following distributions:

- Mandriva Linux 2008 and Mandriva Linux 2009
- openSUSE 10.3 and openSUSE 11.0

Rohde & Schwarz supports these distributions as well.

6.2 How to view "Compressed HTLM Help Files" (chm) in Linux

On Linux operating systems many viewers for Compressed HTML Help files available, for instance:

KDE desktop environment:
In the KDE environment the *kchmviewer*[9] is available.

[7] For futher information please visit http://www.open-std.org/JTC1/SC22/WG14/www/docs/n1256.pdf
[8] For latest updates please visit http://www.ni.com/visa/
[9] Available under http://www.kchmviewer.net/

6.3 How to install R&S instruments driver?

Getting started with LabVIEW 7 or higher:
Please be aware of the fact that for various instruments *dll*-based instrument driver are available (a), otherwise native LabVIEW driver are provided (b).

a. For *dll*-based VXIPnP instrument driver please download the *rsXXX-linux-xxx.tar.gz* and the *rsXXX-linux-lvx-xxx.tar.gz* from the instrument's download site. Extract the *tar* files and follow the instructions of the *README* file.
b. For native LabVIEW drivers it is only necessary to download the *rsXXX-linux-lvx-xxx.tar.gz* file. Thereafter extract the *tar* file and follow the instructions of the *README* file.

Getting started with C/C++:
Visit your instrument's driver site to download the *rsXXX-linux-xxx.tar.gz* (Linux Instrument Driver) *tar* file. Thereafter extract the *tar* file and follow the instructions of the *README* file.

7 Resources

- Rohde & Schwarz Driver download site http://www2.rohde-schwarz.com/en/service_and_support/Downloads/Drivers
- 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/
- Agilent VEE: www.agilent.com/find/vee
- IVI Foundation: http://www.ivifoundation.org/

Accessed: 05/27/2009

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, radio monitoring 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