VNA Programming with RsaToolbox For Qt 5

Nick Lalic

VNA Software Developer

Rohde & Schwarz America

May 2013

Contents

Introduction

“What’s this tutorial about?”

Overview

- This tutorial is an introduction to Vector Network Analyzer (VNA) automation applications using RsaToolbox for Qt 5
- By the end of this tutorial you will be able to:
  - Use one of the provided project wizards to get a simple Qt 5 VNA application up and running
  - Connect to the instrument via Ethernet or GPIB
  - Connect using either the Rohde & Schwarz Instrument Bus (RSIB) or the National Instruments Virtual Instrumentation Software Architecture (NI-VISA).
  - Identify the instrument make, model options and properties
  - Control the instrument
  - Log instrument communication to a text file for troubleshooting
  - Save your applications long-term settings
  - Use RSA Create Installer to create a Windows installer

Audience

- It is assumed that the reader has basic familiarity with:
  - Rohde & Schwarz Vector Network Analyzers (ZVA and/or ZNB)
  - C++
  - Object Oriented Programming
  - Developing applications with Graphical User Interfaces (GUIs)
  - Integrated Development Environments (IDEs) for GUI applications, such as Visual Studio
  - A basic awareness of the “Standard Commands for Programmable Instruments” (SCPI) protocol would be helpful, but is not required.

- This tutorial is meant to be a gentle introduction. If at any point you need additional information, please consult the Additional References listed at the end. These are resources that I have personally found helpful, and are a good place to start for more information.

Requirements

- The software used in this tutorial is either completely free or offered as a free version.
- The following tools will be used:
  - Visual C++ 2010 (Express or better)
  - Qt 5.0 Framework, including the Qt Creator 2.6 Integrated Development Environment (IDE)
  - Windows Installer XML (WIX)
  - Optional: NI-VISA runtime library (for GPIB connections only)

Software

“How do I get started?”

Software Tools

A note on the selection of Qt:

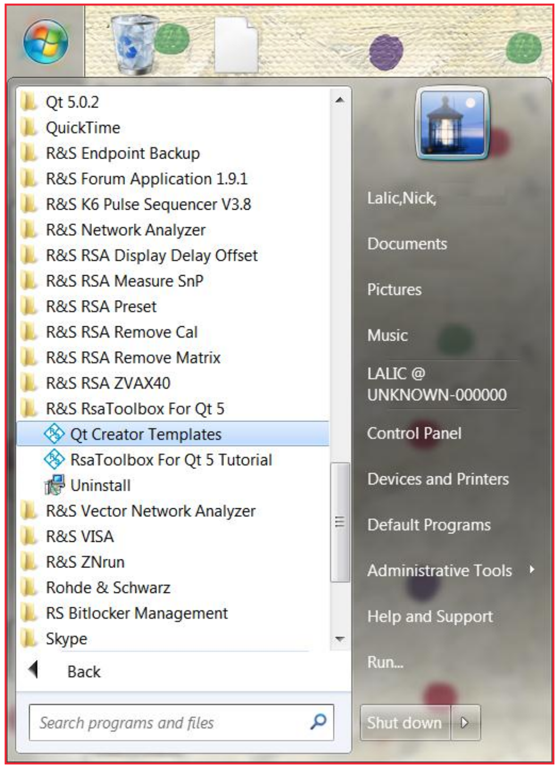
I started creating applications for VNA automation from scratch. As such I had a wide array of software platforms to choose from. My initial inclination was to use Visual Studio and either C# or C++. I wrote a few applications in both languages using Visual Studio 2010 and the .NET Framework. A large percentage of the applications I write are VNA "macros" installed on the instrument itself, to be accessed through the External Tools menu. This means that I have to accommodate both the ZVA and ZNB with my software. The ZNB comes with Windows 7, whereas the ZVA is still running Windows XP SP2. I soon ran into difficulties accommodating both platforms with one version of .NET and without updating the ZVA Windows XP operating system to Service Pack 3. For this reason, for the ease of targeting non-Windows platforms in the future, and for the fact that the ZNB firmware itself is written in Qt, I settled on Qt 5.0 for my applications. This being said, there seems to be no clear consensus as to which platform to use, and the decision is largely based on the application requirements and personal preference.

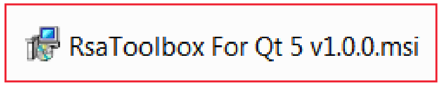
Installation

- Please install the following software tools. The order of installation is important since, for example, Qt depends on a 3rd party compiler and linker (we will use Visual C++ 2010).
- Please install:
  - **Microsoft Visual C++ 2010** (Express or otherwise). You can install everything, but at the very least install the Windows SDK and the Microsoft C++ compiler and linker. Note that Visual C++ is a part of Visual Studio.
  - **Windows Installer XML (WiX) toolset**. WiX is a tool that allows you to create Windows Installers from XML descriptions of your application. The current version as of this tutorial is 3.7, and the installer can be found here.
  - The **Qt 5 Framework**. Qt is a framework of libraries for creating platform-independent applications. The current version, 5.0.2, can be found here. Make sure that you download the installer marked “for Windows 32-bit (VS 2010…)”. Qt supports a wide range of platforms and compilers. We will be writing 32-bit Windows applications using Visual C++ 2010.

Installation

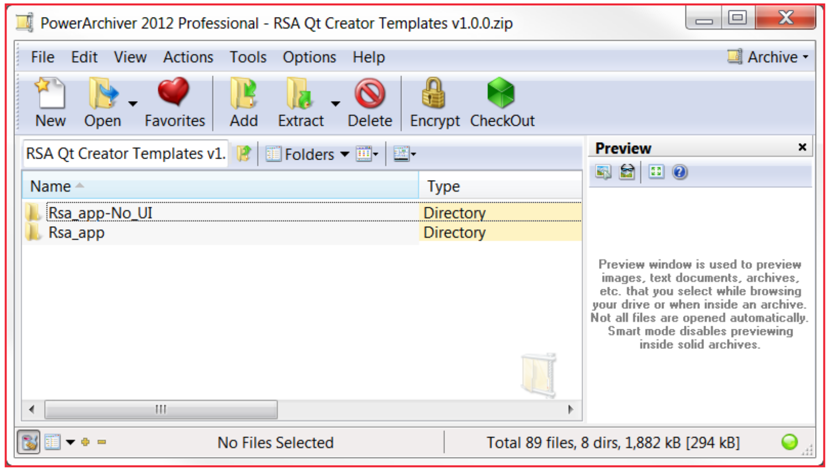
- If you have not already done so, please install RsaToolbox For Qt 5.
- The following tools are provided with RsaToolbox For Qt 5:
  - This tutorial ☺
  - Qt Creator templates (.zip file)
  - RSA Create Installer
    (not available via the Start Menu)





RSA Tools Setup

- The first step to setting up RsaToolbox is to copy the application templates into Qt Creator
  - Find the Qt Creator template wizard directory on your machine. This should be very similar if not identical to:
    C:\Qt\Qt5.0.2\Tools\QtCreator\share\qtcreator\templates\wizards
  - Find the Qt Creator Templates .zip file (linked to from the Start Menu) and extract the contents (*Rsa_app* and *Rsa_app-No_UI*) to the Qt Creator template wizard directory shown above. It is important to copy the folders.


PowerArchiver 2012 Professional - RSA Qt Creator Templates v1.0.0.zip
File Edit View Actions Tools Options Help
Archive
New
Open
Favorites
Add
Extract
Delete
Encrypt
CheckOut
RSA Qt Creator Templates v1.
Folders
Preview

Name	Type
Rsa_app-No_UI	Directory
Rsa_app	Directory

Preview window is used to preview images, text documents, archives, etc. that you select while browsing your drive or when inside an archive. Not all files are opened automatically. Smart mode disables previewing inside solid archives.

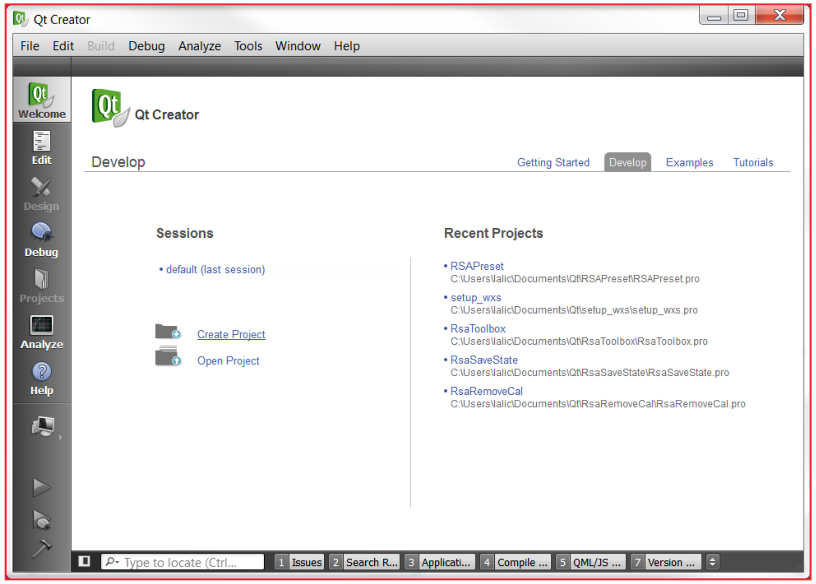
No Files Selected

Total 89 files, 8 dirs, 1,882 kB [294 kB]

ROHDE&SCHWARZ

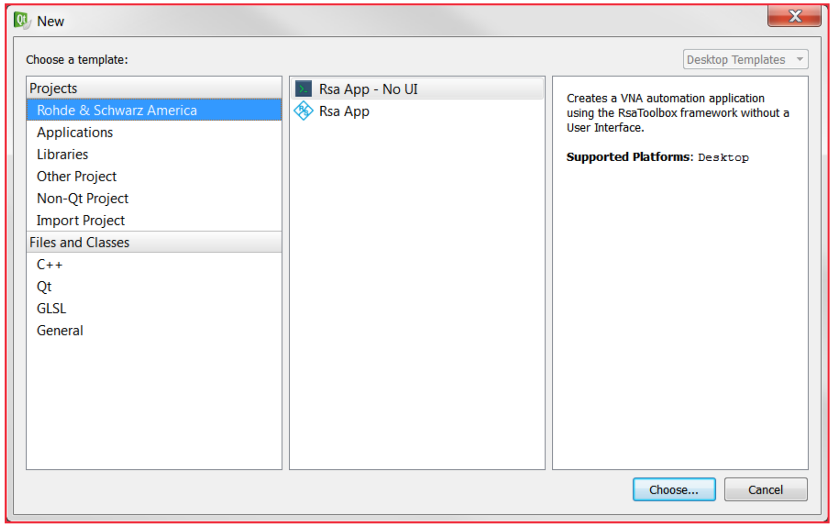
RSA Tools Setup

- Confirm that the templates are installed to Qt Creator
  - Open Qt Creator – Found in the Start Menu under *Qt 5.0.2* as of this tutorial
  - Click the *Create Project* link from the opening screen
    Alternatively you can click *File → New File or Project*
  - Confirm that the templates are available as a new project.



- default (last session)

- RSAPreset
  C:\Users\alic\Documents\Qt\RSAPreset\RSAPreset.pro
- setup_wxs
  C:\Users\alic\Documents\Qt\setup_wxs\setup_wxs.pro
- RsaToolbox
  C:\Users\alic\Documents\Qt\RsaToolbox\RsaToolbox.pro
- RsaSaveState
  C:\Users\alic\Documents\Qt\RsaSaveState\RsaSaveState.pro
- RsaRemoveCal
  C:\Users\alic\Documents\Qt\RsaRemoveCal\RsaRemoveCal.pro



Creates a VNA automation application using the RsaToolbox framework without a User Interface.

Supported Platforms: `Desktop`

Licensing

- Qt is available under two licenses: A commercial (paid) license from Nokia and the open source GNU LGPL v2.1 license. We will be using the LGPL license, which stipulates the following:
  - There are no restrictions on internal use
  - For applications deployed to a customer:
    - Qt libraries must be linked dynamically as external dlls
      (such that the executable is not considered a derivative work under LGPL)
    - The source code for Qt 5 must be available to the recipient
    - Any modifications to Qt must be released to the public as source code along with your application.
    - The source code for your application may remain closed if you wish
- Your source code and executable, so long as they dynamically link to the Qt library, can be released under different licensing terms such as the “Rohde & Schwarz Inc. Terms and Conditions for Royalty-Free Products”

Licensing

- The exact wording of the licensing terms for the Microsoft Visual C++ Runtime depends on the Microsoft product that they came with
- In general:
  - Files listed in the *redist.txt* file in the root installation folder can be redistributed along with your application so long as the license for your application is at least as restricting as the Microsoft license you received them under.
  - The C++ runtime files *msvcp100.dll* and *msvcr100.dll* for Visual C++ 2010 are covered by *redist.txt*
  - We will be providing a Rohde & Schwarz license, as described below, that should provide sufficient coverage.
- The RSA Create Installer tool automatically includes a copy of the “Rohde & Schwarz Inc. Terms and Conditions for Royalty-Free Products” with your application (see *License.txt* in the RsaToolbox installation folder).

Licensing

- RSIB can also be distributed to customers under the “Rohde & Schwarz Inc. Terms and Conditions for Royalty-Free Products” license.
- RSIB is also included automatically by RSA Create Installer
- NI-VISA, properties of National Instruments, cannot be distributed with an application to a customer, as Rohde & Schwarz does not hold the proper deployment license. However, RsaToolbox will work with NI-VISA when provided by the end-user.

Our First App

“How do I get started?”

Delete Correction Data

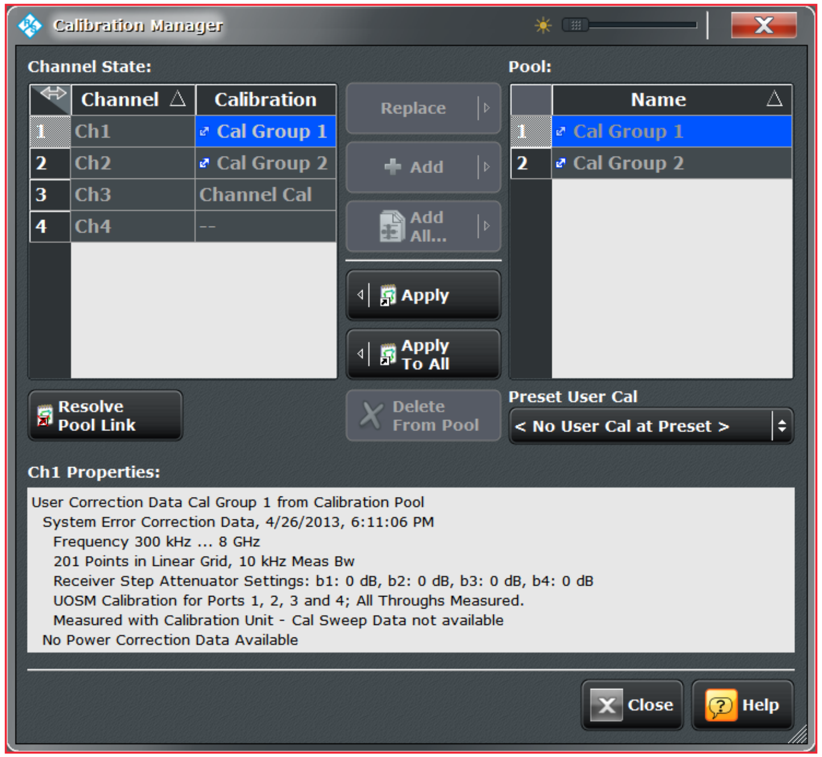
Problem:

Recently a customer needed to be able to save VNA settings to a Recall Set file. These Recall Set files would then be distributed to a pool of VNAs in a production environment with several technicians. As it turns out, the correction data for each channel is saved along with the instrument settings in the Recall Set. When the Recall Set is loaded on another instrument the instrument then appears to be calibrated even though calibration data is not valid. A technician, therefore, may assume that the instrument is calibrated and perform inaccurate measurements.

We can meet the customers needs by deleting correction data from each channel BEFORE saving the Recall Set. Unfortunately the ZVA and ZNB both do not provide this functionality. Let's write an application using RsaToolbox to get around this.

Delete Correction Data

- The following is a screenshot of the ZNB Calibration Manager
- As you can see there is no way to delete the correction data present in a channel. At best, we can "Resolve Pool Link", which disconnects the channel from the Cal Group it is associated with, but the result is calibration data local to the channel itself, as seen in channel 3 (Ch3).



	Channel	Calibration
1	Ch1	Cal Group 1
2	Ch2	Cal Group 2
3	Ch3	Channel Cal
4	Ch4	--

	Name
1	Cal Group 1
2	Cal Group 2

Ch1 Properties:

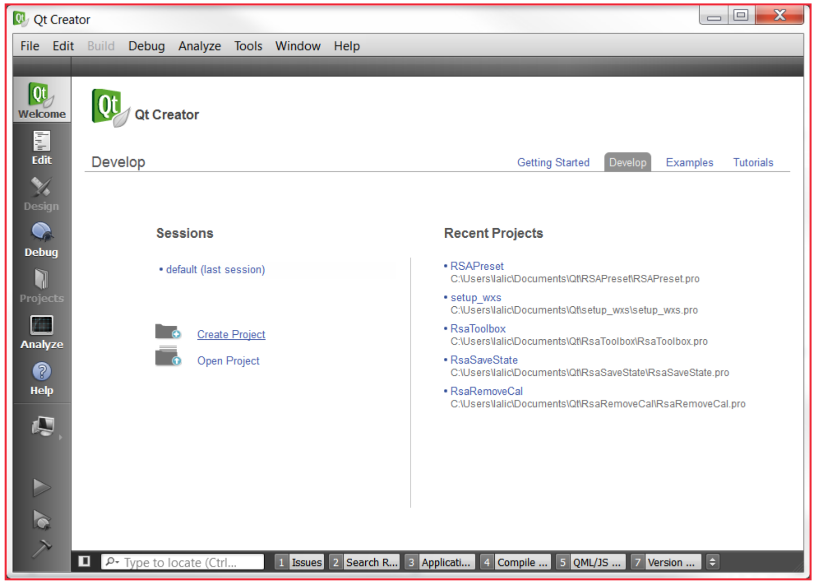
```
User Correction Data Cal Group 1 from Calibration Pool
  System Error Correction Data, 4/26/2013, 6:11:06 PM
    Frequency 300 kHz ... 8 GHz
    201 Points in Linear Grid, 10 kHz Meas Bw
    Receiver Step Attenuator Settings: b1: 0 dB, b2: 0 dB, b3: 0 dB, b4: 0 dB
    UOSM Calibration for Ports 1, 2, 3 and 4; All Throughs Measured.
    Measured with Calibration Unit - Cal Sweep Data not available
  No Power Correction Data Available
```

Delete Correction Data

- Often customer problems such as these can be solved by creating a “macro”, or a small application installed on the VNA itself
- These applications can be integrated into the “External Tools” menu of the instrument.
- The functionality provided in the manual interface of the VNA does not exactly match that provided through the software interface. That is, Functions that do not exist manually will exist in software and vice versa.
- We can use this fact to our advantage in this case.

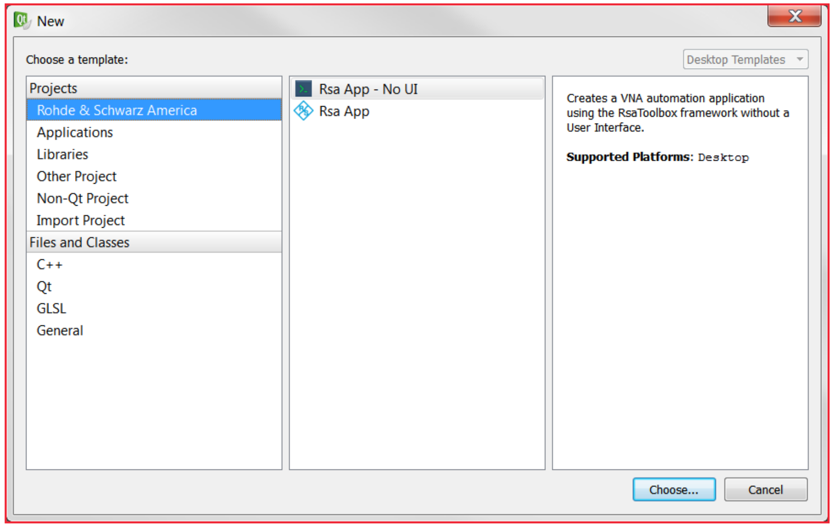
Create a New Project

- Let's get back to creating a new RsaToolbox project:
  - Open Qt Creator
  - Click the *<u>Create Project</u>* link from the opening screen
    or you can also click *File* → *New File or Project*
  - Select the "Rsa App – No UI" as we will not need a user interface (UI)



- default (last session)

- RSAPreset
  C:\Users\lalic\Documents\Qt\RSAPreset\RSAPreset.pro
- setup_wxs
  C:\Users\lalic\Documents\Qt\setup_wxs\setup_wxs.pro
- RsaToolbox
  C:\Users\lalic\Documents\Qt\RsaToolbox\RsaToolbox.pro
- RsaSaveState
  C:\Users\lalic\Documents\Qt\RsaSaveState\RsaSaveState.pro
- RsaRemoveCal
  C:\Users\lalic\Documents\Qt\RsaRemoveCal\RsaRemoveCal.pro



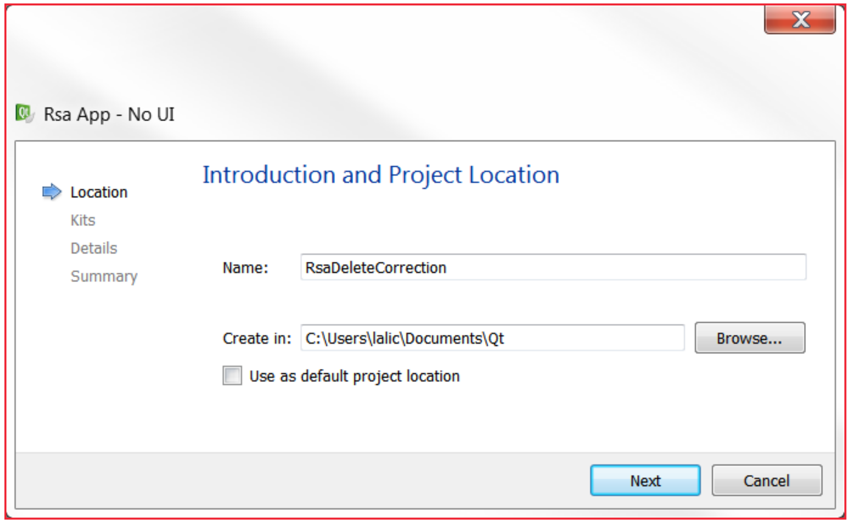
Creates a VNA automation application using the RsaToolbox framework without a User Interface.

Supported Platforms: `Desktop`

ROHDE&SCHWARZ

Create a New Project

- Name your project *RsaDeleteCorrection*
- Accept the default project location
- Click *Next*



Rsa App - No UI

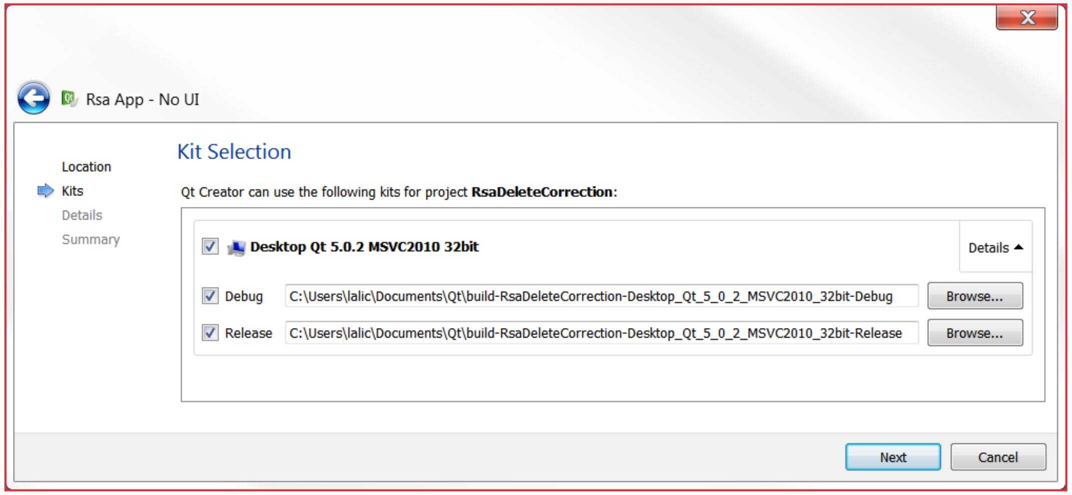
Introduction and Project Location

Name: RsaDeleteCorrection

Create in: C:\Users\lalic\Documents\Qt

☐ Use as default project location

- Accept the default “Kit”:
  - Desktop Application
  - Qt 5.0.2
  - Microsoft Visual C++ 2010
  - 32-bit
- Accept the default target directories
- Click *Next*



Rsa App - No UI

Kit Selection

Qt Creator can use the following kits for project **RsaDeleteCorrection**:

✓ **Desktop Qt 5.0.2 MSVC2010 32bit**

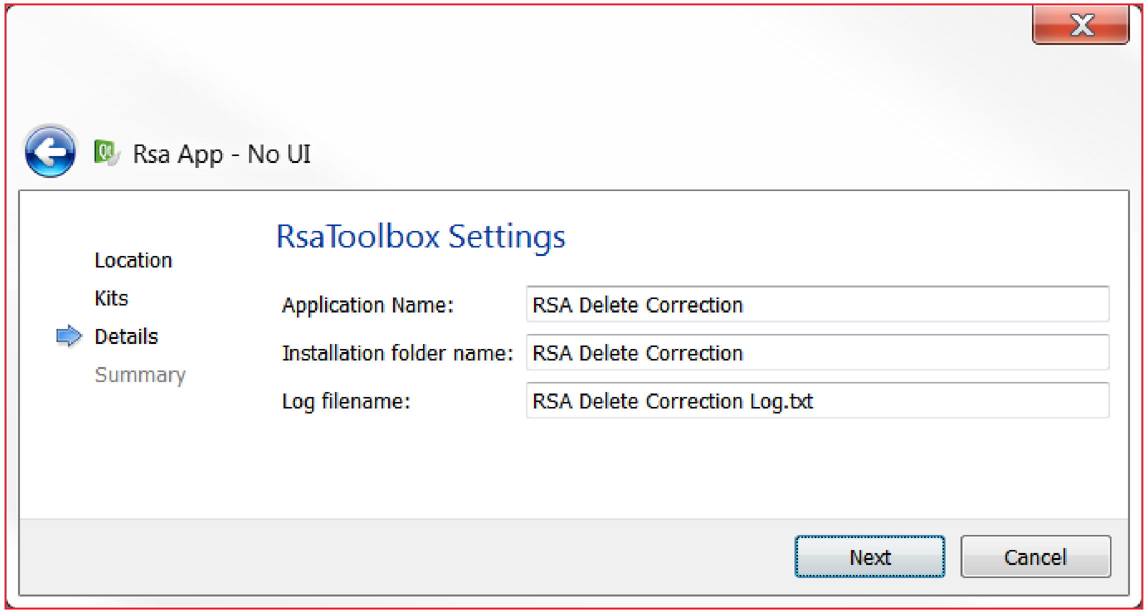
✓ Debug C:\Users\lalic\Documents\Qt\build-RsaDeleteCorrection-Desktop_Qt_5_0_2_MSVC2010_32bit-Debug

✓ Release C:\Users\lalic\Documents\Qt\build-RsaDeleteCorrection-Desktop_Qt_5_0_2_MSVC2010_32bit-Release

Create a New Project

- Set *Application Name* to "RSA Delete Correction"
- Set *Installation folder Name* to "RSA Delete Correction"
- Set *Log filename* to "RSA Delete Correction Log.txt"

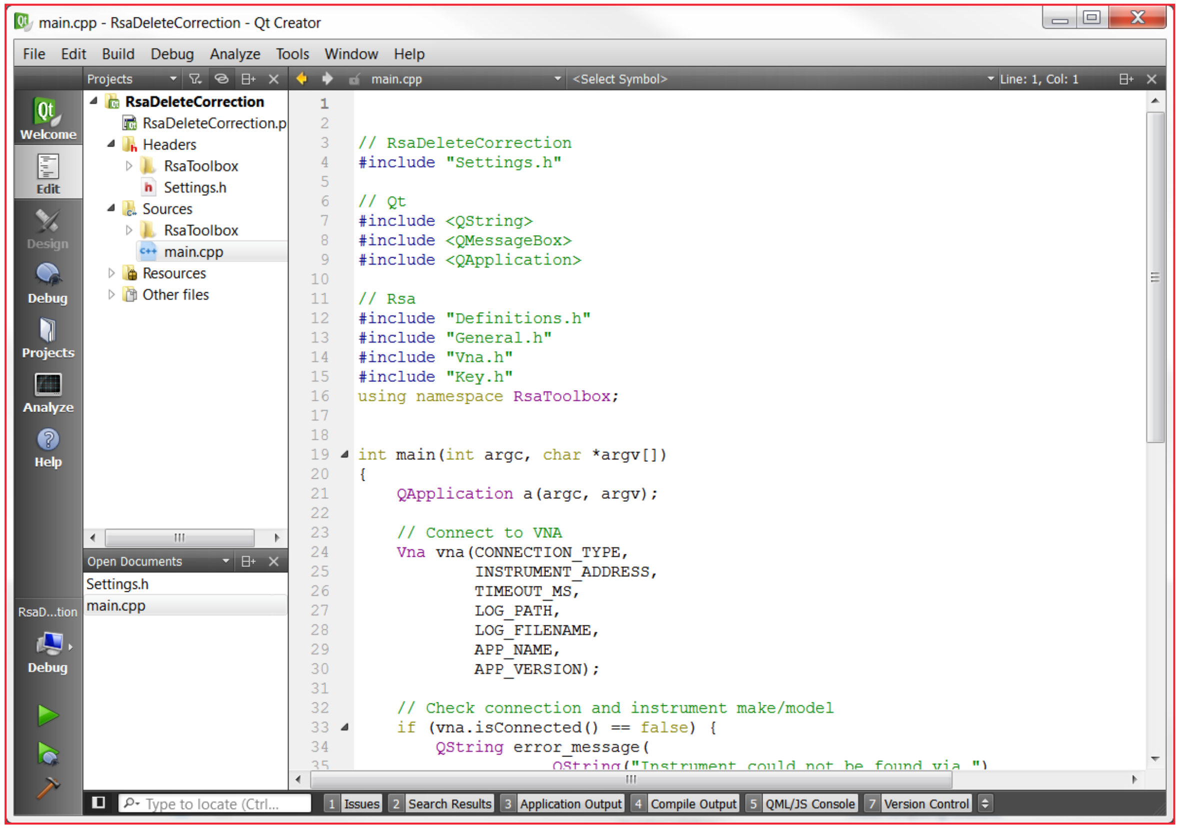
- Click *Next*
- *Click Finish*



Application Name: RSA Delete Correction
Installation folder name: RSA Delete Correction
Log filename: RSA Delete Correction Log.txt

Create a New Project

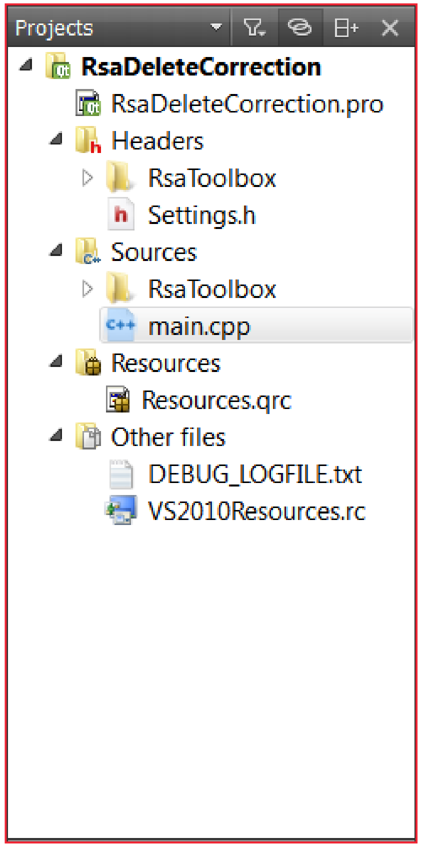
- At this point your project is created. You should see something similar to the following in Qt Creator



```


// RsaDeleteCorrection
#include "Settings.h"

// Qt
#include <QString>
#include <QMessageBox>
#include <QApplication>

// Rsa
#include "Definitions.h"
#include "General.h"
#include "Vna.h"
#include "Key.h"
using namespace RsaToolbox;


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    // Connect to VNA
    Vna vna(CONNECTION_TYPE,
            INSTRUMENT_ADDRESS,
            TIMEOUT_MS,
            LOG_PATH,
            LOG_FILENAME,
            APP_NAME,
            APP_VERSION);

    // Check connection and instrument make/model
    if (vna.isConnected() == false) {
        QString error_message(
                QString("Instrument could not be found via ")
```

Project Components

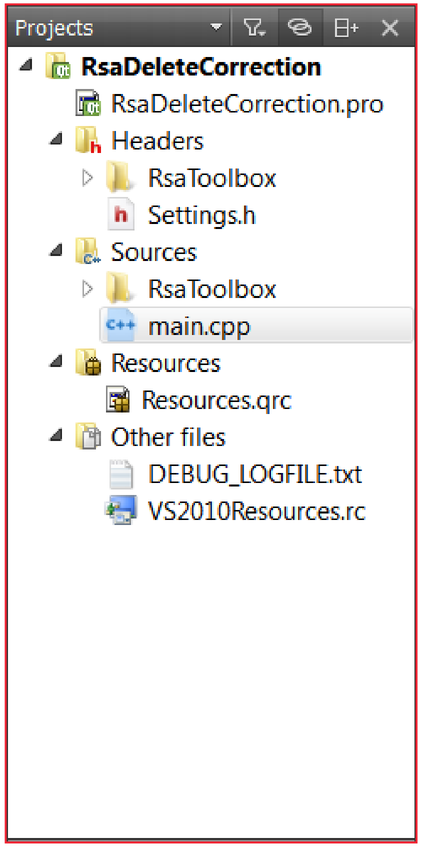
- Lets take a moment to discuss the various parts of this new project
- In the *Projects* pane you will notice that several files have been added to your project
  - **.pro** files are Qt Creator Project files. They contain project information such as what type of project this is, what files to include and compiler settings.
  - The **RsaToolbox** folder contains the source code for RsaToolbox.
  - **Settings.h** is a C++ header file with settings for RsaToolbox applications
  - **main.cpp** contains the main function. Main is a function common to all C\C++ applications that contains the actual code for your application.



ROHDE&SCHWARZ

Project Components

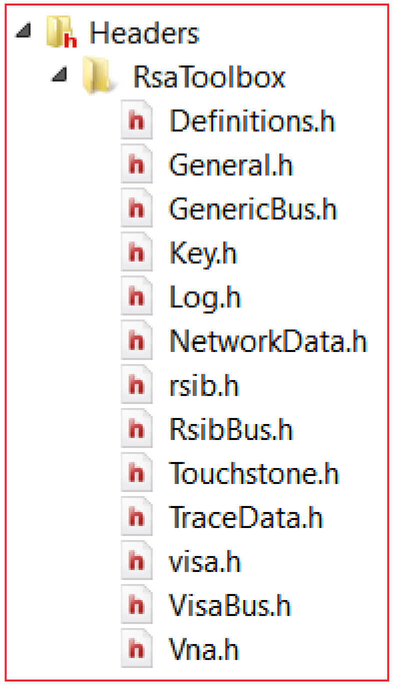
- **Resources.qrc** is a Qt Resource file that we are using to include a Rohde & Schwarz icon to be used with any Qt windows, such as error messages
- **VS2010Resources.rc** is a similar resource file in a Microsoft Visual C++ format that specifies the icon to apply to the executable file after compiling.
- DEBUG_LOGFILE.txt is just that; it is a log of all communication with the instrument as your commands run. This can be useful for debugging purposes. This file is written to when your project is in debug mode. In release mode, the log file is written to the installation directory instead.



ROHDE&SCHWARZ

Project Components

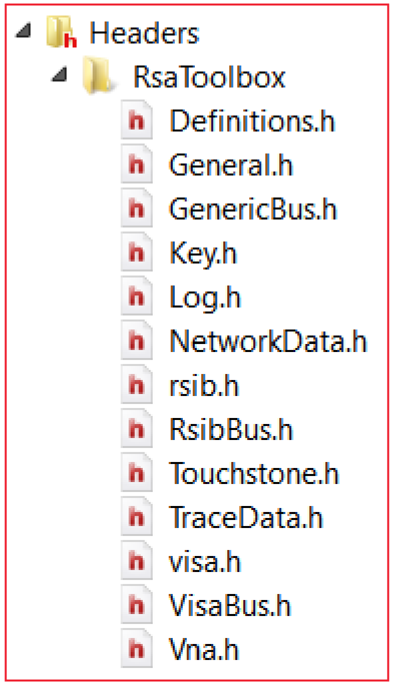
- **RsaToolbox** consists of the following:
  - **Definitions.h** contains a number of enumerations for describing the VNA settings and data format
  - **General.h** contains general-purpose functions, for example the ToString() function converts an enumerator to a human readable QString format.
  - **NetworkData.h** and **TraceData.h** are simple classes that hold measurement data and properties for S-Parameter networks and traces, respectively
  - **Touchstone.h** contains Read/Write functions for Touchstone (.sNp) Version 1.0 files.
  - **Key.h** contains a Key template class for reading and writing long-term application settings to binary files
  - **Vna.h** contains the Vna class which allows you to connect to and control an instrument.



```
Headers
  RsaToolbox
    Definitions.h
    General.h
    GenericBus.h
    Key.h
    Log.h
    NetworkData.h
    rsib.h
    RsibBus.h
    Touchstone.h
    TraceData.h
    visa.h
    VisaBus.h
    Vna.h
```

Project Components

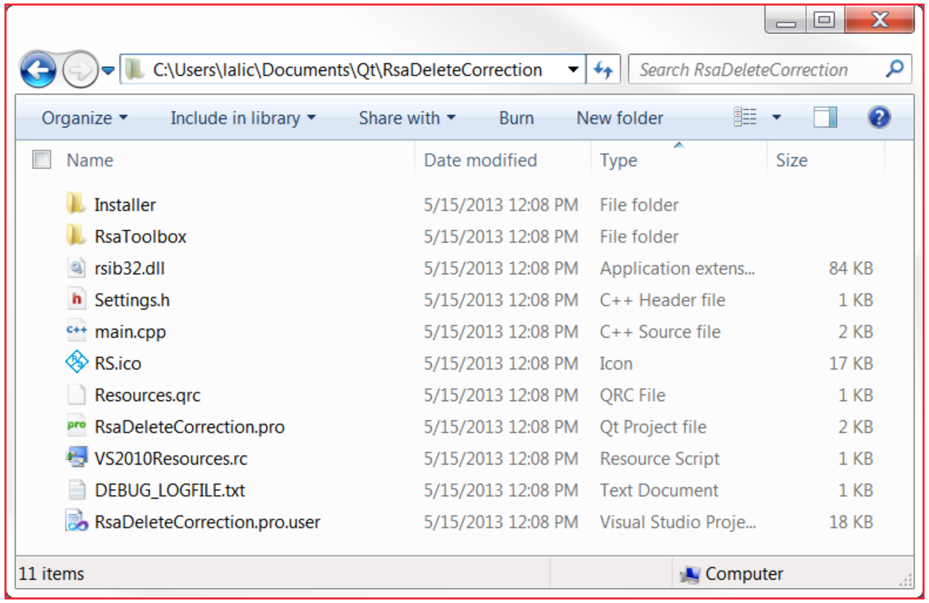
- The rest of the header and source files of RsaToolbox work behind the scenes as far as application development is concerned. You do not need to understand them unless you want to modify or contribute to RsaToolbox.



```
Headers
  RsaToolbox
    Definitions.h
    General.h
    GenericBus.h
    Key.h
    Log.h
    NetworkData.h
    rsib.h
    RsibBus.h
    Touchstone.h
    TraceData.h
    visa.h
    VisaBus.h
    Vna.h
```

Project Components

- If you navigate to the RsaDeleteCorrection project folder you will find all of the files previously mentioned plus a few more.
  - **.pro.user** files contain additional Qt Project settings specific to your computer.
  - **Rsib32.dll** contains the RSIB library, to be deployed with our application
  - The Installer folder contains additional files we will use later on to create a Windows installer for our application.
  - **RsaToolbox** contains the actual RsaToolbox code, including C++ headers and source files.



C:\Users\lalic\Documents\Qt\RsaDeleteCorrection
Search RsaDeleteCorrection
Organize
Include in library
Share with
Burn
New folder

Name	Date modified	Type	Size
Installer	5/15/2013 12:08 PM	File folder	
RsaToolbox	5/15/2013 12:08 PM	File folder	
rsib32.dll	5/15/2013 12:08 PM	Application extens...	84 KB
Settings.h	5/15/2013 12:08 PM	C++ Header file	1 KB
main.cpp	5/15/2013 12:08 PM	C++ Source file	2 KB
RS.ico	5/15/2013 12:08 PM	Icon	17 KB
Resources.qrc	5/15/2013 12:08 PM	QRC File	1 KB
RsaDeleteCorrection.pro	5/15/2013 12:08 PM	Qt Project file	2 KB
VS2010Resources.rc	5/15/2013 12:08 PM	Resource Script	1 KB
DEBUG_LOGFILE.txt	5/15/2013 12:08 PM	Text Document	1 KB
RsaDeleteCorrection.pro.user	5/15/2013 12:08 PM	Visual Studio Proje...	18 KB

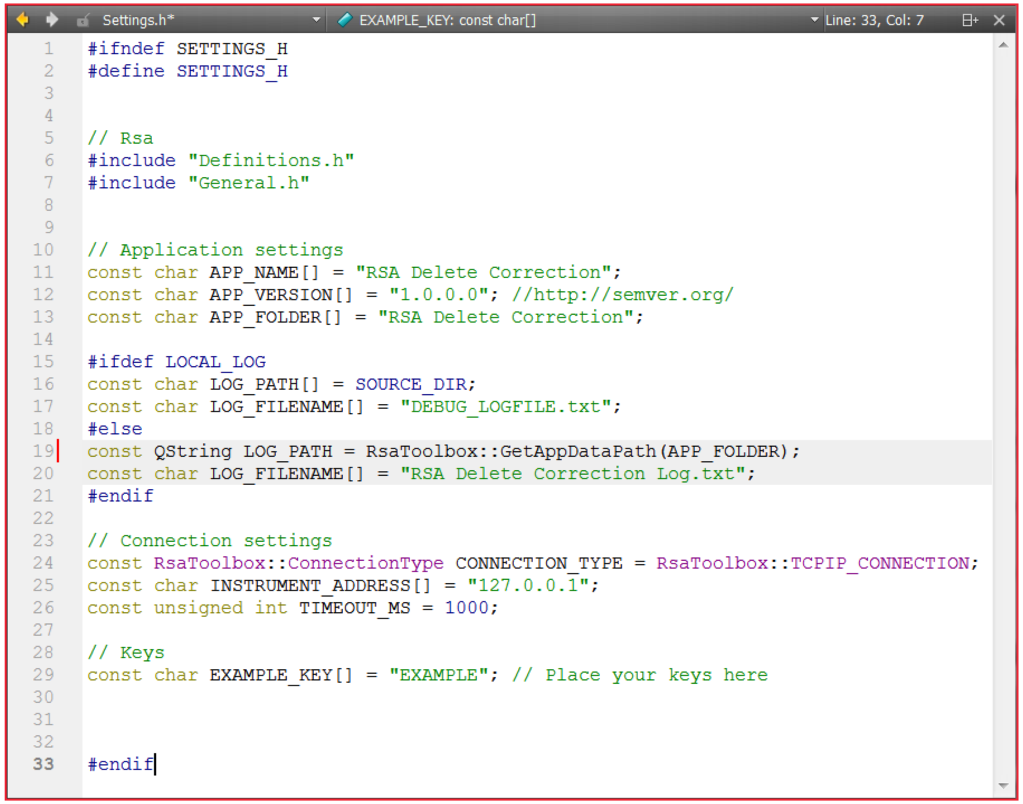
11 items

Computer

ROHDE&SCHWARZ

Project Components

- Let’s move back to Qt Creator
- Double-click **Settings.h** in the Project pane to open it.
- In addition to the information we provided to create this project, Settings.h contains a few additional settings such as the application version and instrument connection information.
- These settings are summarized in a header file for convenience, but can be defined elsewhere.
- We will leave them as-is for now.

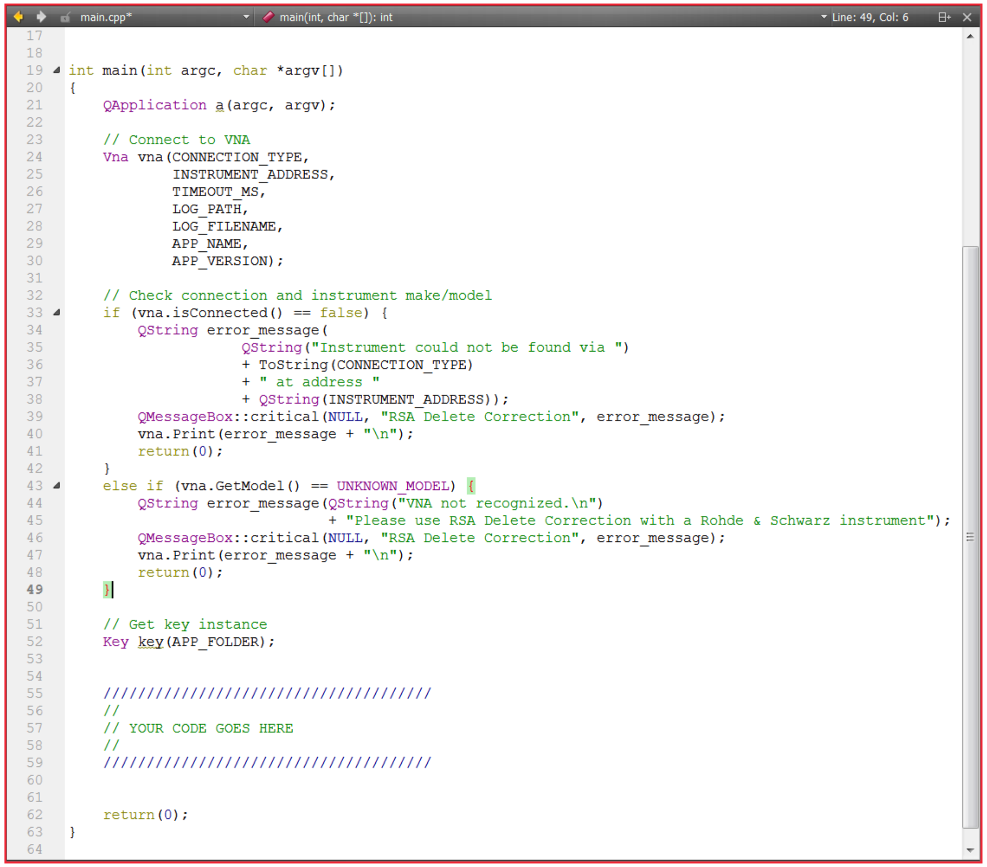


```
#ifndef SETTINGS_H
#define SETTINGS_H


// Rsa
#include "Definitions.h"
#include "General.h"


// Application settings
const char APP_NAME[] = "RSA Delete Correction";
const char APP_VERSION[] = "1.0.0.0"; //http://semver.org/
const char APP_FOLDER[] = "RSA Delete Correction";

#ifdef LOCAL_LOG
const char LOG_PATH[] = SOURCE_DIR;
const char LOG_FILENAME[] = "DEBUG_LOGFILE.txt";
#else
const QString LOG_PATH = RsaToolbox::GetAppDataPath(APP_FOLDER);
const char LOG_FILENAME[] = "RSA Delete Correction Log.txt";
#endif

// Connection settings
const RsaToolbox::ConnectionType CONNECTION_TYPE = RsaToolbox::TCPIP_CONNECTION;
const char INSTRUMENT_ADDRESS[] = "127.0.0.1";
const unsigned int TIMEOUT_MS = 1000;

// Keys
const char EXAMPLE_KEY[] = "EXAMPLE"; // Place your keys here



#endif
```

ROHDE&SCHWARZ

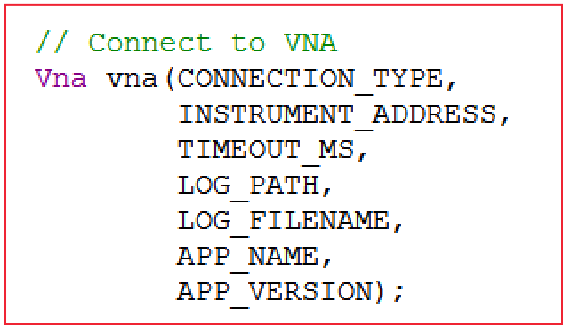
Project Components

- Now open **main.cpp**.
- Let’s focus on the contents of the main function for now
- This code can be broken down into parts:
  - Connect to the VNA
  - Check connection status
  - Check VNA make/model
  - Create Key instance
  - // YOUR CODE GOES HERE



```cpp


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    // Connect to VNA
    Vna vna(CONNECTION_TYPE,
            INSTRUMENT_ADDRESS,
            TIMEOUT_MS,
            LOG_PATH,
            LOG_FILENAME,
            APP_NAME,
            APP_VERSION);

    // Check connection and instrument make/model
    if (vna.isConnected() == false) {
        QString error_message(
                    QString("Instrument could not be found via ")
                    + ToString(CONNECTION_TYPE)
                    + " at address "
                    + QString(INSTRUMENT_ADDRESS));
        QMessageBox::critical(NULL, "RSA Delete Correction", error_message);
        vna.Print(error_message + "\n");
        return(0);
    }
    else if (vna.GetModel() == UNKNOWN_MODEL) {
        QString error_message(QString("VNA not recognized.\n")
                              + "Please use RSA Delete Correction with a Rohde & Schwarz instrument");
        QMessageBox::critical(NULL, "RSA Delete Correction", error_message);
        vna.Print(error_message + "\n");
        return(0);
    }

    // Get key instance
    Key key(APP_FOLDER);


    //////////////////////////////////////
    //
    // YOUR CODE GOES HERE
    //
    //////////////////////////////////////


    return(0);
}

```

Vna Class



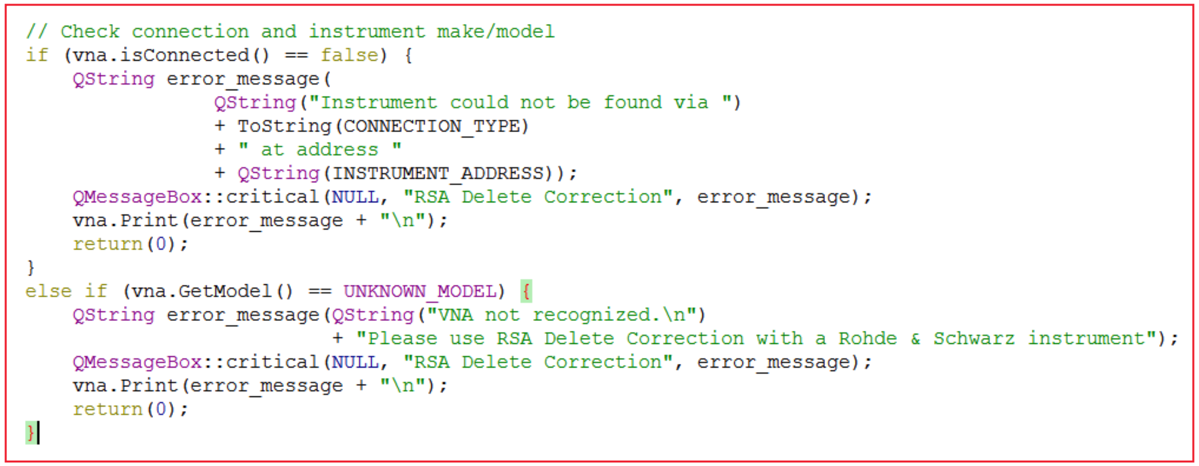
```
// Connect to VNA
Vna vna(CONNECTION_TYPE,
        INSTRUMENT_ADDRESS,
        TIMEOUT_MS,
        LOG_PATH,
        LOG_FILENAME,
        APP_NAME,
        APP_VERSION);
```

- The Vna class connects to and controls an R&S VNA
- The Vna constructor requires several arguments:
  - enum ConnectionType (TCPIP_CONNECTION or GPIB_CONNECTION)
  - QString instrument_address (such as “127.0.0.1” for a local TCPIP connection)
  - unsigned int timeout_ms, the time in milliseconds before instrument communication times out
  - QString log_path, the application’s path to the log file when released
  - QString Log file name
  - QString Application name
  - QString Application version
- The parameters used here, as defined in *Settings.h*, default to a local TCPIP connection (for apps running on-instrument).
- These parameters can be changed to meet your requirements.

ROHDE&SCHWARZ

Vna Class

- The Vna function isConnected() returns true or false, indicating whether or not an instrument was found.
- The Vna function GetModel() returns an enumerator as defined in *Definitions.h*, corresponding to the make/model of the instrument.
- Enum VnaModel has the following possible values:
  - ZVA_MODEL
  - ZVB_MODEL
  - ZVH_MODEL
  - ZVL_MODEL
  - ZVT_MODEL
  - ZNB_MODEL
  - ZNC_MODEL
  - UNKNOWN_MODEL



```
// Check connection and instrument make/model
if (vna.isConnected() == false) {
    QString error_message(
                QString("Instrument could not be found via ")
                + ToString(CONNECTION_TYPE)
                + " at address "
                + QString(INSTRUMENT_ADDRESS));
    QMessageBox::critical(NULL, "RSA Delete Correction", error_message);
    vna.Print(error_message + "\n");
    return(0);
}
else if (vna.GetModel() == UNKNOWN_MODEL) {
    QString error_message(QString("VNA not recognized.\n")
                          + "Please use RSA Delete Correction with a Rohde & Schwarz instrument");
    QMessageBox::critical(NULL, "RSA Delete Correction", error_message);
    vna.Print(error_message + "\n");
    return(0);
}
```

- These if/else statements display an error dialog if an instrument is not found or that instrument is not an R&S VNA model. The error is also printed to the log file

Deleting Corrections

- Let’s skip the Key declaration for now. We will use it in Part 2 of this tutorial.
- Despite the fact that you cannot create an empty Cal Group (no correction data) manually via the instrument interface, you CAN do it in software
- If you then apply this empty Cal Group to a channel you effectively wipe out any existing correction data, deleting corrections from the channel
- If we do this to each channel we can delete all the applied correction data from the instrument settings.
- Saving the Recall Set will create a calibration-free file.

Vna Class

- We will use the following Vna class functions to do this:
  - Reset()
  - CreateSet(QString *set_name*)
  - CloseSet(QString *set_name*)
  - QVector<unsigned int> GetChannels()
  - Channel(unsigned int *channel*)::SaveCalGroup(QString *cal_file*)
  - Channel(unsigned int *channel*)::SetCalGroup(QString *cal_file*)

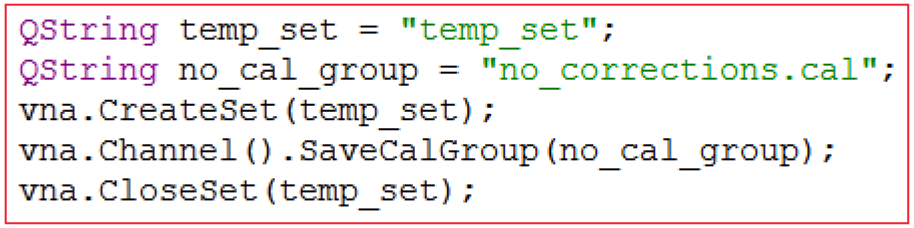
- The Vna::Channel(unsigned int channel) interface provides channel functionality
- Similarly, the following interfaces for traces and diagrams are provided:
  - Vna::Trace(QString trace_name)
  - Vna::Diagram(unsigned int diagram)
- Many other functions exist. They are defined in Vna.h
- Note: RsaToolbox is currently in progress. Documentation does not yet exist, other than this tutorial. Please check Vna.h for more information.

Deleting Corrections

- In order to simplify this example, let’s make some basic assumptions about the state of the instrument
  - No User Preset is defined
  - No Default Calibration on Preset is defined
  - Only one Recall Set is active
- Our first step is to create a so-called “empty” Cal Group, as follows:
  - Create a temporary Recall Set with factory default settings (no calibration)
  - Save the calibration data (there is none) to a Cal Group.
  - Close this Recall Set, returning the instrument to its previous state

Deleting Corrections

- Place the following code snippet, which does does just that, into main.cpp in place of the // YOUR CODE GOES HERE section.



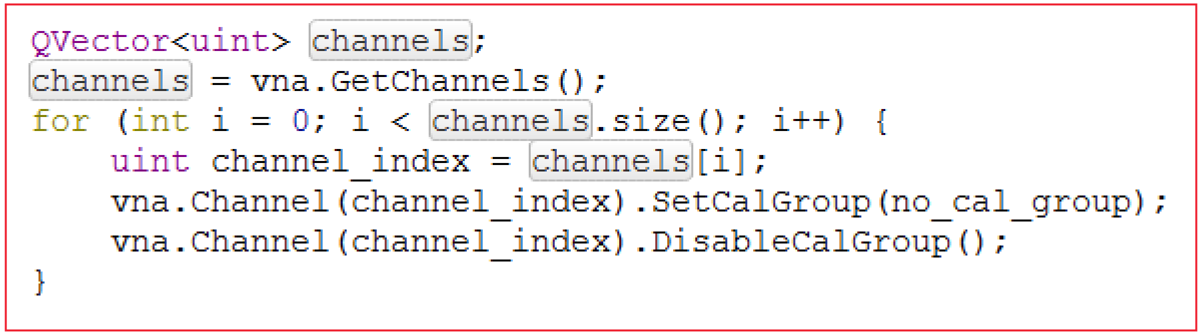
```
QString temp_set = "temp_set";
QString no_cal_group = "no_corrections.cal";
vna.CreateSet(temp_set);
vna.Channel().SaveCalGroup(no_cal_group);
vna.CloseSet(temp_set);
```

- Note that a call to Channel() with no argument defaults to channel 1

ROHDE&SCHWARZ

Deleting Corrections

- Now that we have created our “no_corrections.cal” Cal Group, we can apply it to each active channel.
  - Call GetChannels() to get a Qvector list of active channels
  - For each channel, call SetCalGroup(“no_corrections.cal”) to effectively delete any correction data
  - Disassociate the channel from the Cal Group by calling DisableCalGroup()
- Add the following code to main.cpp to do this:

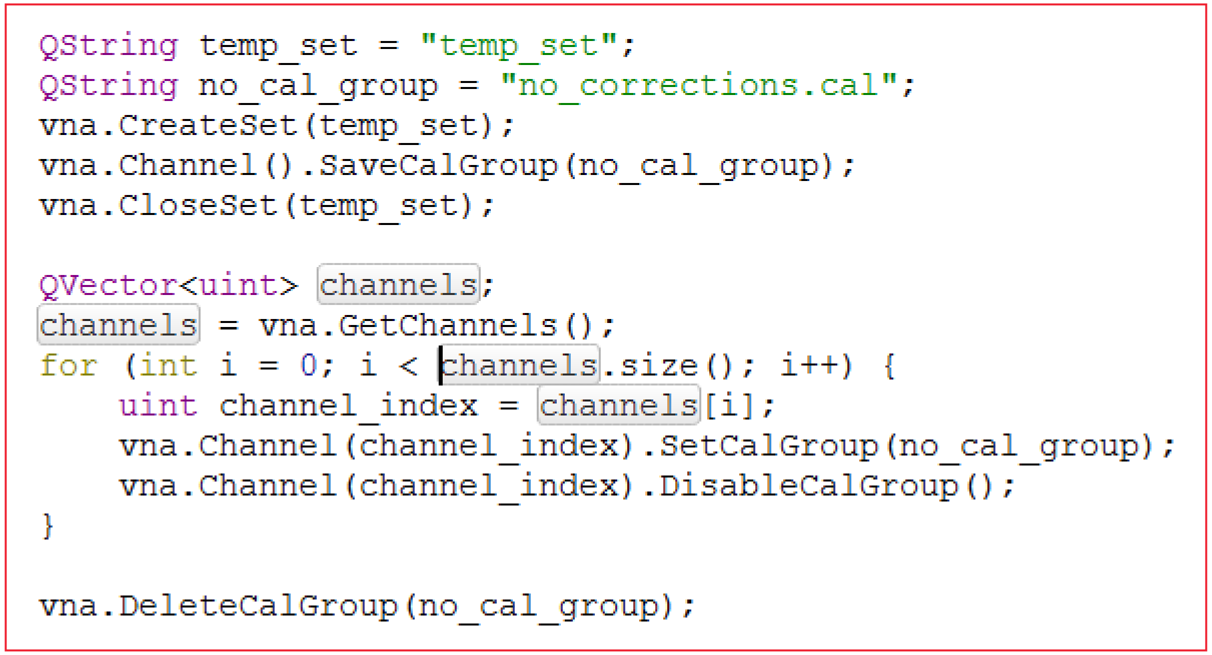


```
QVector<uint> channels;
channels = vna.GetChannels();
for (int i = 0; i < channels.size(); i++) {
    uint channel_index = channels[i];
    vna.Channel(channel_index).SetCalGroup(no_cal_group);
    vna.Channel(channel_index).DisableCalGroup();
}
```

Deleting Corrections

- To clean up, let's delete the Cal Group that we created with the DeleteCalGroup() command.

- After adding this, we are left with the following code for our simple application:

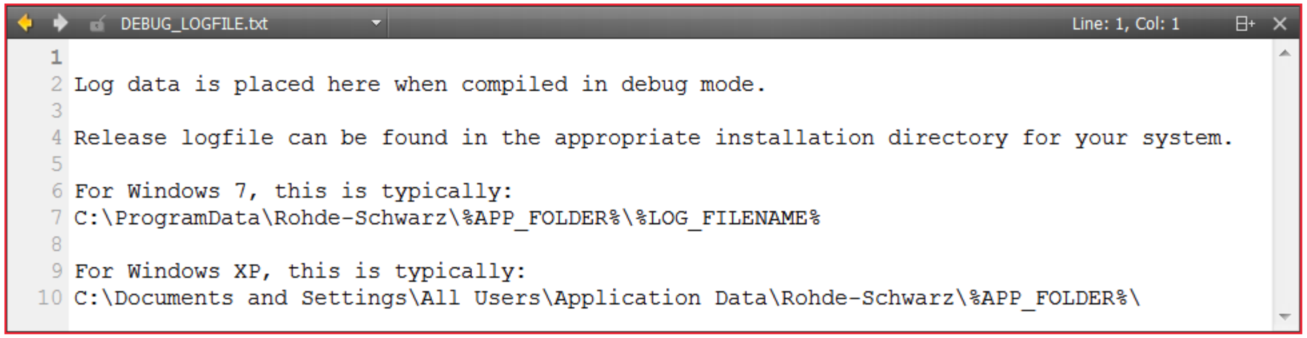


```
QString temp_set = "temp_set";
QString no_cal_group = "no_corrections.cal";
vna.CreateSet(temp_set);
vna.Channel().SaveCalGroup(no_cal_group);
vna.CloseSet(temp_set);

QVector<uint> channels;
channels = vna.GetChannels();
for (int i = 0; i < channels.size(); i++) {
    uint channel_index = channels[i];
    vna.Channel(channel_index).SetCalGroup(no_cal_group);
    vna.Channel(channel_index).DisableCalGroup();
}

vna.DeleteCalGroup(no_cal_group);
```

ROHDE&SCHWARZ

Compiling and Running

- Before we compile, let's take a look at the current contents of DEBUG_LOGFILE.txt:



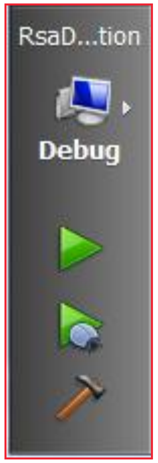
```

Log data is placed here when compiled in debug mode.

Release logfile can be found in the appropriate installation directory for your system.

For Windows 7, this is typically:
C:\ProgramData\Rohde-Schwarz\%APP_FOLDER%\%LOG_FILENAME%

For Windows XP, this is typically:
C:\Documents and Settings\All Users\Application Data\Rohde-Schwarz\%APP_FOLDER%\
```

- Right now it contains a reminder of the location of the log file once your application is compiled in Release mode. This is the location of your log file when your application is deployed.
- After we compile our application in debug mode and run it, this text will be replaced with information about our application, the instrument and any commands sent to it.

ROHDE&SCHWARZ

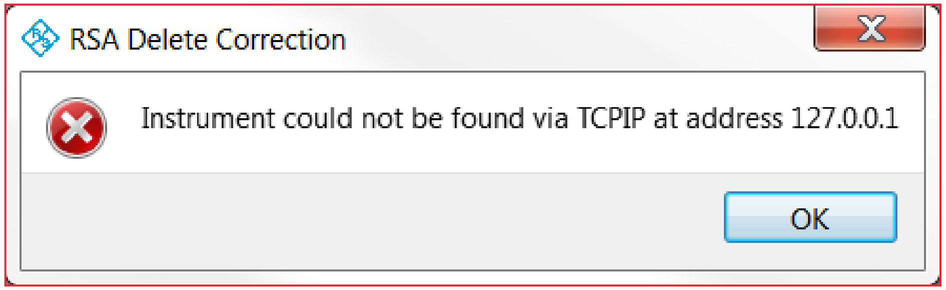
Compiling and Running

- Click File → Close "DEBUG_LOGFILE.txt" to close the log file.
- To compile and run your application in debug mode:
  - Confirm that Qt Creator is set to "Debug" by referencing the bottom-left panel of Qt Creator. If it instead says "Release" for some reason, click the computer icon to change it back to Debug
  - Click the big green play icon to compile and run.

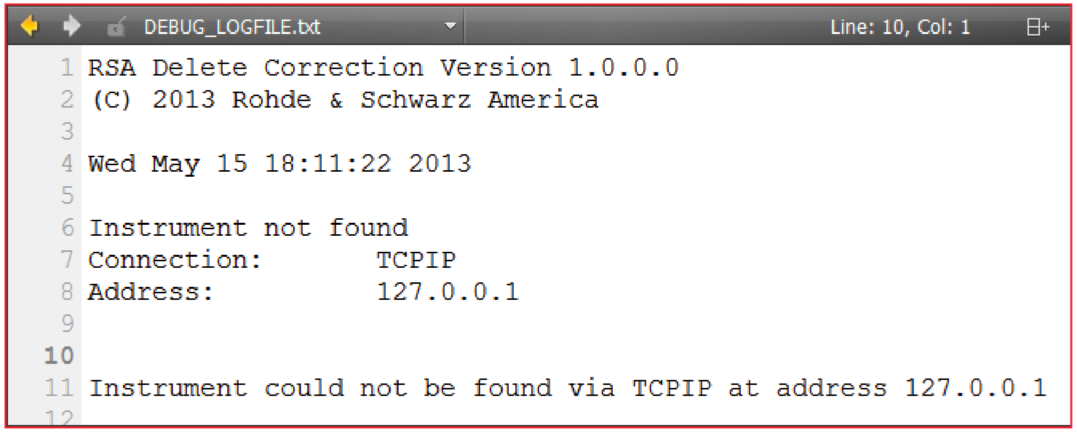


Compiling and Running

- If you do not have a VNA emulator running on your development machine, you will get the following dialog and a mostly empty debug log file.



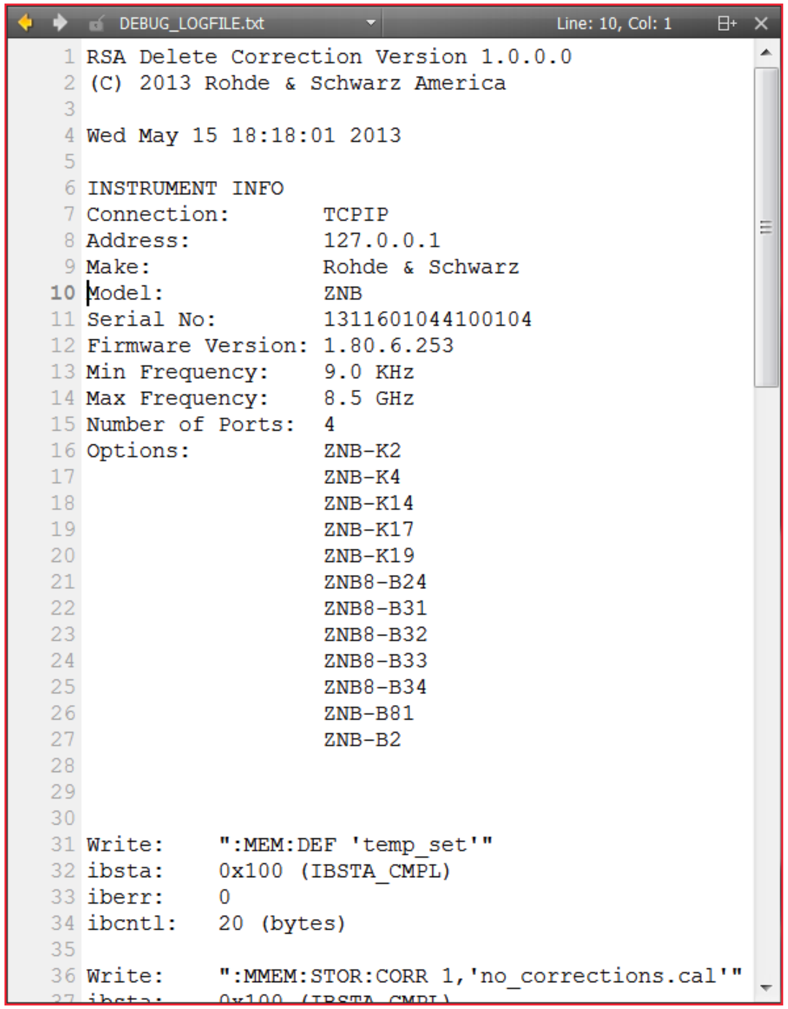
Instrument could not be found via TCPIP at address 127.0.0.1



```
RSA Delete Correction Version 1.0.0.0
(C) 2013 Rohde & Schwarz America

Wed May 15 18:11:22 2013

Instrument not found
Connection:       TCPIP
Address:          127.0.0.1


Instrument could not be found via TCPIP at address 127.0.0.1
```

Compiling and Running

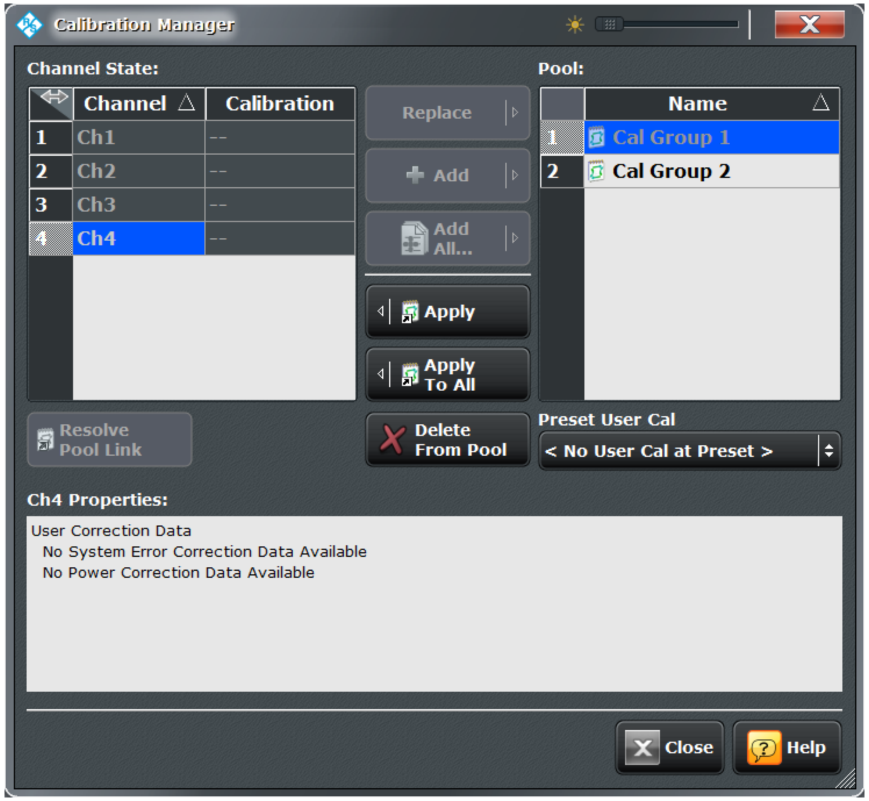
- I happen to have a ZNB emulator running on my development machine. Here is an example of a filled out log file.
- The application name and version is listed along with a time stamp and information about the instrument
- If the instrument is not from R&S the log file will simply contain the identification string of the instrument
- Following the log file header, you will see a list of every SCPI command or data transfer and the status codes of the instrument bus.



```
RSA Delete Correction Version 1.0.0.0
(C) 2013 Rohde & Schwarz America

Wed May 15 18:18:01 2013

INSTRUMENT INFO
Connection:       TCPIP
Address:          127.0.0.1
Make:             Rohde & Schwarz
Model:            ZNB
Serial No:        1311601044100104
Firmware Version: 1.80.6.253
Min Frequency:    9.0 KHz
Max Frequency:    8.5 GHz
Number of Ports:  4
Options:          ZNB-K2
                  ZNB-K4
                  ZNB-K14
                  ZNB-K17
                  ZNB-K19
                  ZNB8-B24
                  ZNB8-B31
                  ZNB8-B32
                  ZNB8-B33
                  ZNB8-B34
                  ZNB-B81
                  ZNB-B2



Write:     ":MEM:DEF 'temp_set'"
ibsta:     0x100 (IBSTA_CMPL)
iberr:     0
ibcntl:    20 (bytes)

Write:     ":MMEM:STOR:CORR 1,'no_corrections.cal'"
ibsta:     0x100 (IBSTA_CMPL)
```

Compiling and Running

- You can also see in Calibration Manager that the channel correction data has been deleted after running our application:



	Channel	Calibration
1	Ch1	--
2	Ch2	--
3	Ch3	--
4	Ch4	--

	Name
1	Cal Group 1
2	Cal Group 2

```
User Correction Data
  No System Error Correction Data Available
  No Power Correction Data Available
```

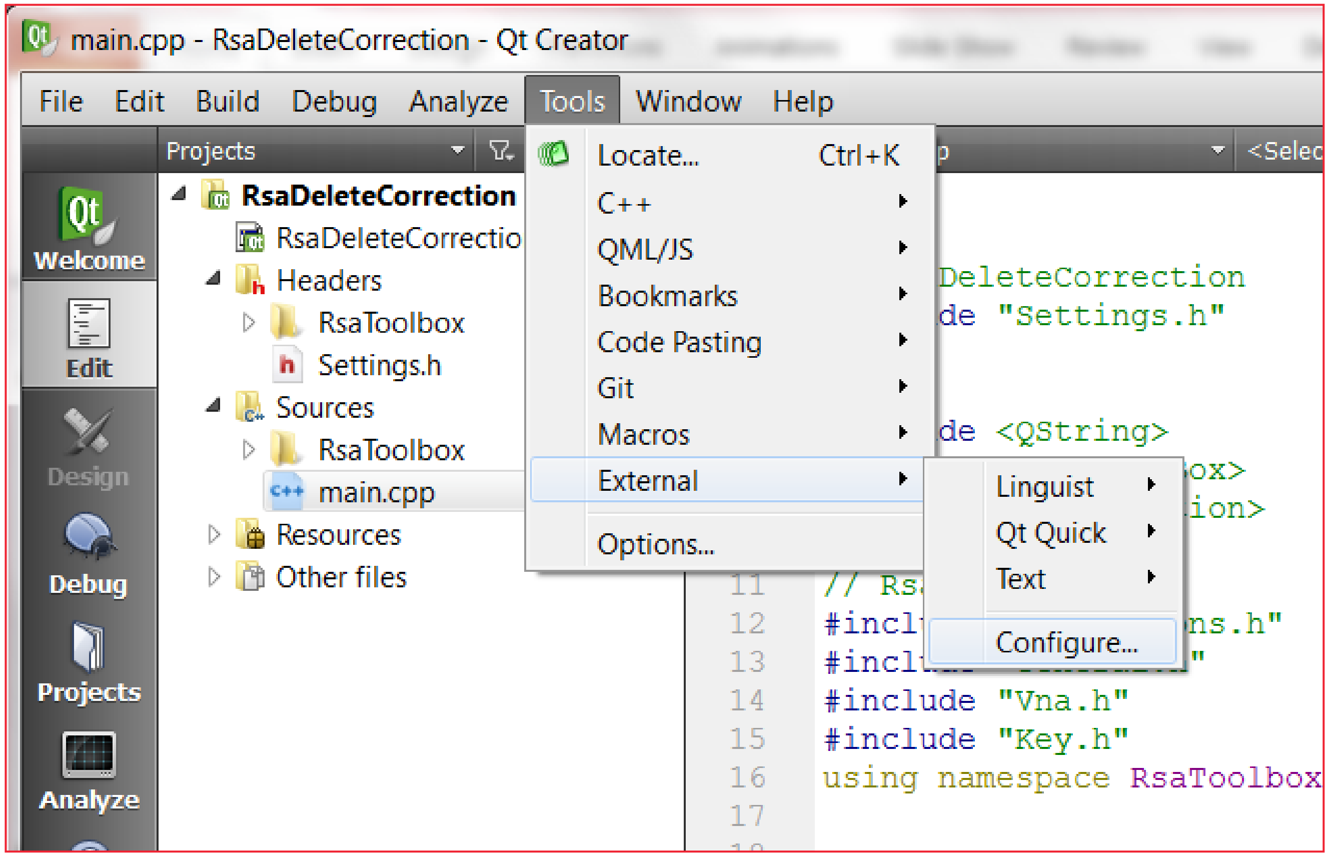
ROHDE&SCHWARZ

Creating an Installer

- At this point, with our application compiling successfully, we can create an installer
- To help you navigate the convoluted world of Windows Installer XML (WiX), I have created a tool, RSA Create Installer, to compliment projects created with these templates
- RSA Create Installer processes a WiX template named Product.wxs found in the *Installer* folder of your project.
- RSA Create Installer is run from Qt Creator itself as an external tool.
- We will now set up Qt Creator to use it.

RSA Create Installer Setup

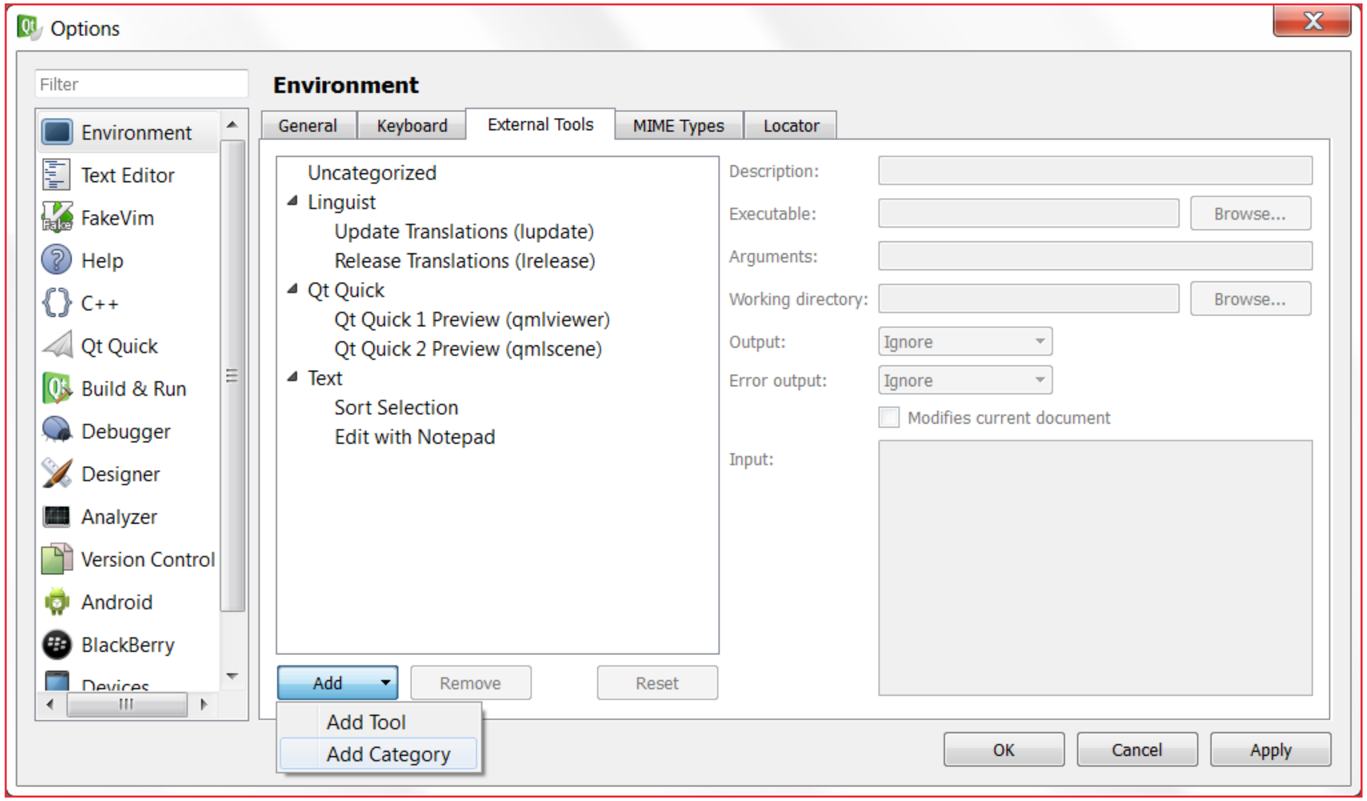
- Click Tools → External → Configure to access the tool configuration menu



```
DeleteCorrection
de "Settings.h"

de <QString>
ox>
ion>
// Rs
#incl            ns.h"
#incl              "
#include "Vna.h"
#include "Key.h"
using namespace RsaToolbox
```

RSA Create Installer Setup

- Click the Add button, then select Add Category
- Name this new category "Rohde-Schwarz"



Options

Filter

Environment

General | Keyboard | External Tools | MIME Types | Locator

Environment
Text Editor
FakeVim
Help
C++
Qt Quick
Build & Run
Debugger
Designer
Analyzer
Version Control
Android
BlackBerry
Devices

- Uncategorized
- Linguist
  - Update Translations (lupdate)
  - Release Translations (lrelease)
- Qt Quick
  - Qt Quick 1 Preview (qmlviewer)
  - Qt Quick 2 Preview (qmlscene)
- Text
  - Sort Selection
  - Edit with Notepad

Description:
Executable: Browse...
Arguments:
Working directory: Browse...
Output: Ignore
Error output: Ignore
Modifies current document
Input:

Add | Remove | Reset

Add Tool
Add Category

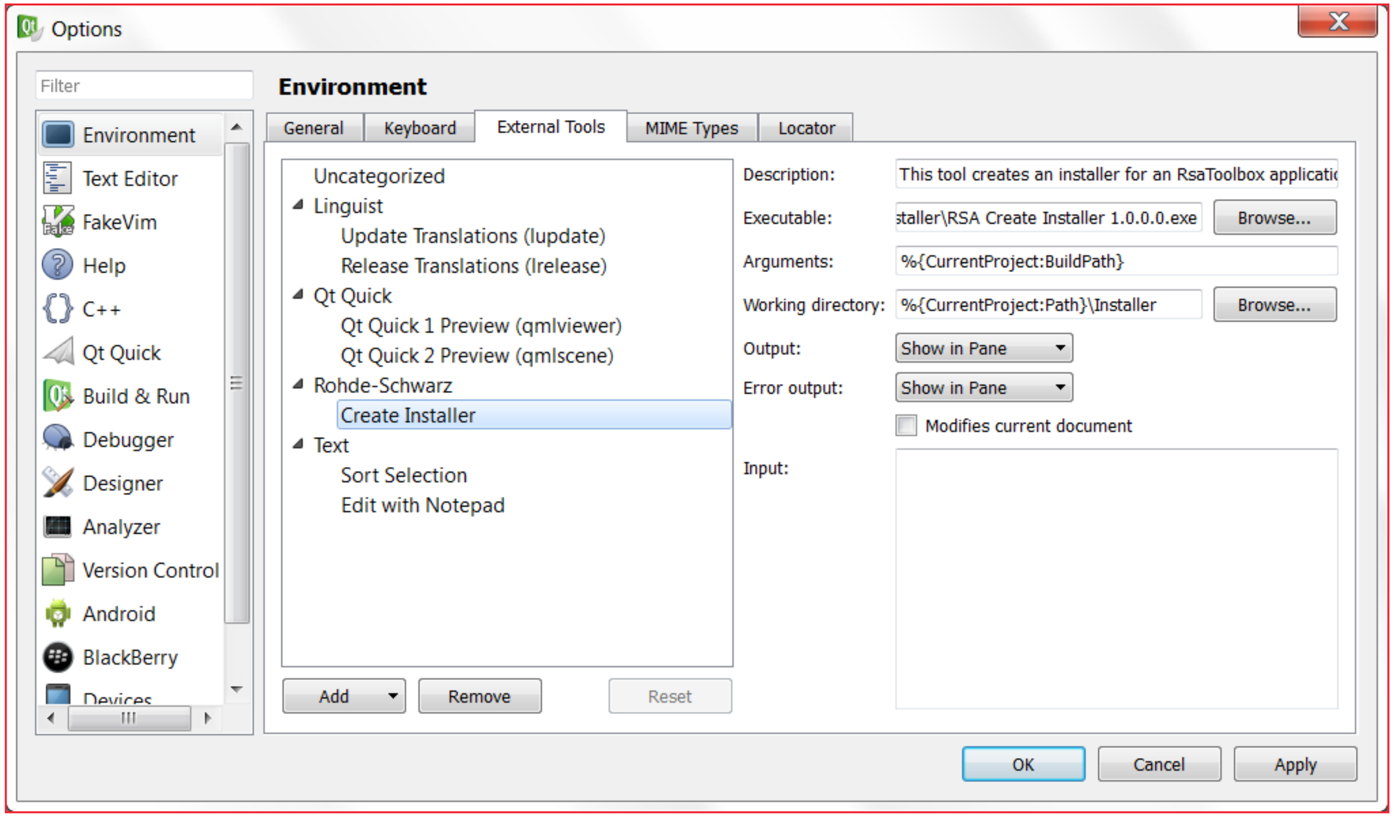
OK | Cancel | Apply

RSA Create Installer Setup

- With “Rohde-Schwarz” highlighted, once again click the Add button, then click Add Tool
- Name the tool “Create Installer” and give it the following settings:
  - **Description:** “This tool creates an installer for an RsaToolbox application”
  - **Executable Path:** This will depend on the installation location of RsaToolbox For Qt 5. On my Windows 7 64-bit development machine, it is located at: *C:\Program Files (x86)\Rohde-Schwarz\RsaToolbox For Qt 5\RSA Create Installer\ RSA Create Installer 1.0.0.0.exe*
  - **Arguments:** RSA Create Installer needs the current project’s build path to be able to find your executable. Click the Arguments text edit field, then click the Variables icon on the right. Double-click “CurrentProject:BuildPath” and exit. The Argument field should now contain “%{CurrentProject:BuildPath}”
  - **Working Directory:** Similarly, click the Variables icon and choose CurrentProject:FilePath, then append the text “/Installer” to the end. The Working directory field should now read “%{CurrentProject:Path}\Installer”

RSA Create Installer Setup

- The result should look something like this:



Options

Environment

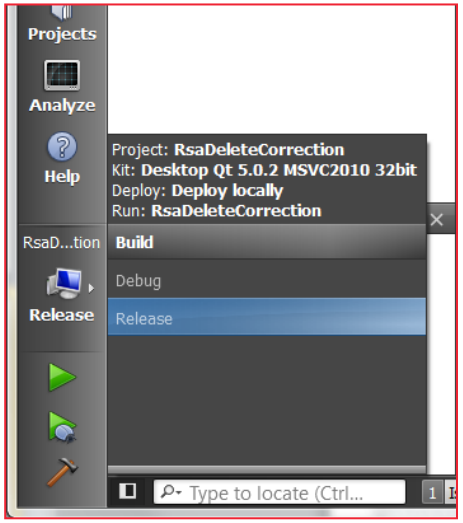
Tabs: General, Keyboard, External Tools, MIME Types, Locator

- Uncategorized
- Linguist
  - Update Translations (lupdate)
  - Release Translations (lrelease)
- Qt Quick
  - Qt Quick 1 Preview (qmlviewer)
  - Qt Quick 2 Preview (qmlscene)
- Rohde-Schwarz
  - Create Installer
- Text
  - Sort Selection
  - Edit with Notepad

Description:	This tool creates an installer for an RsaToolbox applicati
Executable:	staller\RSA Create Installer 1.0.0.0.exe
Arguments:	%{CurrentProject:BuildPath}
Working directory:	%{CurrentProject:Path}\Installer
Output:	Show in Pane
Error output:	Show in Pane
	☐ Modifies current document
Input:	

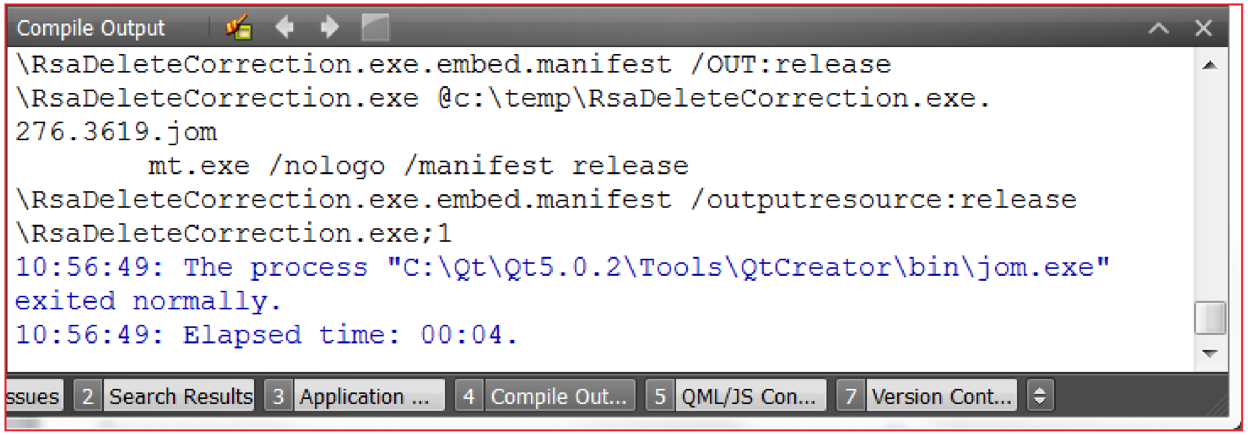
RSA Create Installer Setup

- Click Ok to accept these changes and return to your project.
- To use Create Installer, we must first put our project into Release mode and compile.
- To do this, Click the computer icon in the bottom-left to access the build options.
- Select Release mode to build with Release settings.
  - Compiling in debug mode and deploying your app with Release dlls will cause your program to crash. To prevent this, RSA Create Installer requires your project to be set to Release mode before you run it.
- From the menu select Build then click Rebuild All.


Project: RsaDeleteCorrection
Kit: Desktop Qt 5.0.2 MSVC2010 32bit
Deploy: Deploy locally
Run: RsaDeleteCorrection

RSA Create Installer Setup

- You can confirm that your application was built successfully by checking the Compile Output tab in the bottom-right.

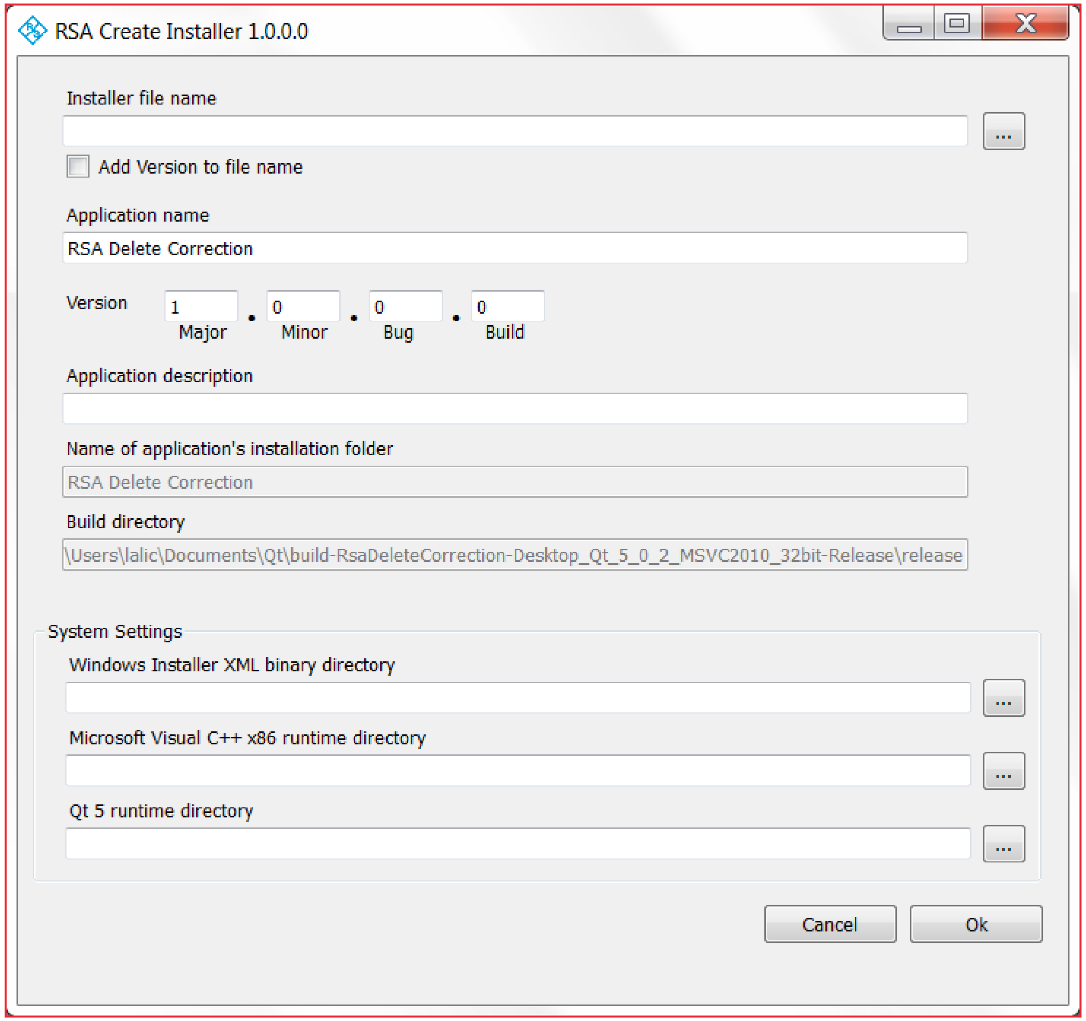


```
\RsaDeleteCorrection.exe.embed.manifest /OUT:release
\RsaDeleteCorrection.exe @c:\temp\RsaDeleteCorrection.exe.
276.3619.jom
        mt.exe /nologo /manifest release
\RsaDeleteCorrection.exe.embed.manifest /outputresource:release
\RsaDeleteCorrection.exe;1
10:56:49: The process "C:\Qt\Qt5.0.2\Tools\QtCreator\bin\jom.exe"
exited normally.
10:56:49: Elapsed time: 00:04.
```

- You are now ready to run the RSA Create Installer tool by clicking Tools → External → Rohde-Schwarz → Create Installer

RSA Create Installer Setup

- You should see a screen like this:



RSA Create Installer 1.0.0.0

Installer file name

☐ Add Version to file name

Application name: RSA Delete Correction

Version: 1 . 0 . 0 . 0 (Major . Minor . Bug . Build)

Application description

Name of application's installation folder: RSA Delete Correction

Build directory: \Users\lalic\Documents\Qt\build-RsaDeleteCorrection-Desktop_Qt_5_0_2_MSVC2010_32bit-Release\release

System Settings

Windows Installer XML binary directory

Microsoft Visual C++ x86 runtime directory

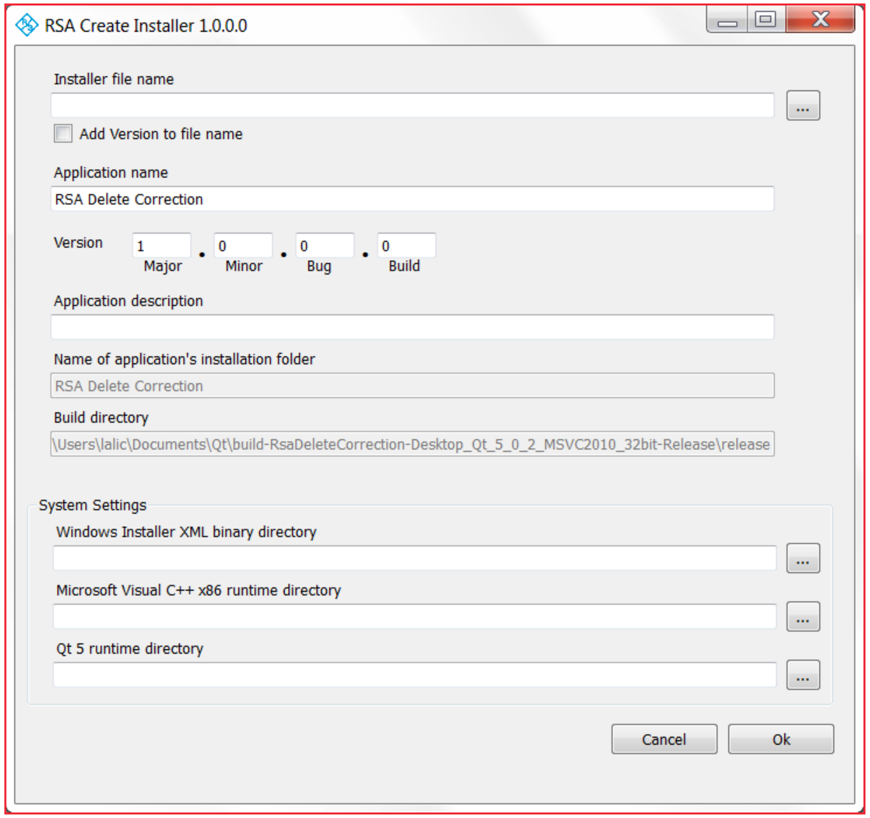
Qt 5 runtime directory

ROHDE&SCHWARZ

RSA Create Installer Setup

- Create Installer has the following settings:
  - **Installer File name**
  - **Application Name**
  - **Version**
  - **Application Description**
- Create Installer also redisplays:
  - The Application **Installation Folder**
  - The project's **Build Directory**
- Note that the Installation Folder is specified when the project is created. The generated project depends on it, and so it is not changeable.

- These settings are remembered on a per-project basis.


RSA Create Installer 1.0.0.0

Installer file name

☐ Add Version to file name

Application name: RSA Delete Correction

Version: 1 . 0 . 0 . 0 (Major, Minor, Bug, Build)

Application description

Name of application's installation folder: RSA Delete Correction

Build directory: \Users\lalic\Documents\Qt\build-RsaDeleteCorrection-Desktop_Qt_5_0_2_MSVC2010_32bit-Release\release

System Settings

Windows Installer XML binary directory

Microsoft Visual C++ x86 runtime directory

Qt 5 runtime directory

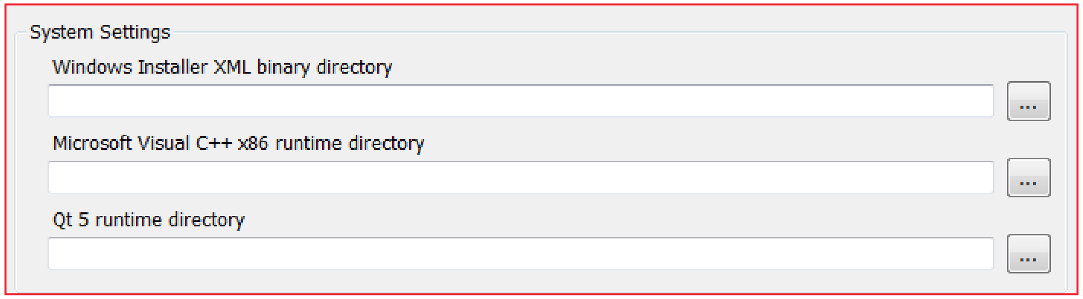
RSA Create Installer Setup

- Use the following settings:
  - **Installer File name: “***RSA Delete Correction*” in the *Installer* folder
  - Check the **Add version to file name** checkbox
  - **Application Name: “***RSA Delete Correction”*
  - **Version:** 1.0.0.0
  - **Application Description: “**Deletes all correction data from the active Recall Set”

- **Warning:**
  While these settings are in part the same as those found in Settings.h, it is important to remember that they are specified separately in the Create Installer tool. As such, you will want to make sure that you coordinate any changes you make with Settings.h. Failing to do so will create an application with different names and versions specified in the app itself and the log file, compared to the installer and Windows Registry.

RSA Create Installer Setup

- There are some additional system-level settings that point RSA Create Installer to the resources it needs.
- These directories are set across all RsaToolbox projects and only need to be specified once.
- They can be changed at any time to reflect any updates or system changes.
- System Settings include:
  - **The WiX Binary Directory**
  - **Visual C++ x86 Runtime Directory**
  - **Qt 5 Runtime Directory**



System Settings

Windows Installer XML binary directory

Microsoft Visual C++ x86 runtime directory

Qt 5 runtime directory

RSA Create Installer Setup

- **The WiX Binary Directory**
  - RSA Create Installer uses the WiX toolset to generate your installer. Specifically it is looking for the candle.exe compiler and light.exe linker. On a Windows 7 64-bit system, these are usually found in:
    C:\Program Files (x86)\WiX Toolset v3.7\bin

- **Visual C++ x86 Runtime Directory**
  - Your installer will the C\C++ Runtime redistributables needed to run your application. Specifically, we need the *msvcp100.dll* and *msvcr100.dll* files. Using Visual Studio 2010 on Windows 7 64-bit, they are usually found here:
    C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\redist\x86\Microsoft.VC100.CRT

RSA Create Installer Setup

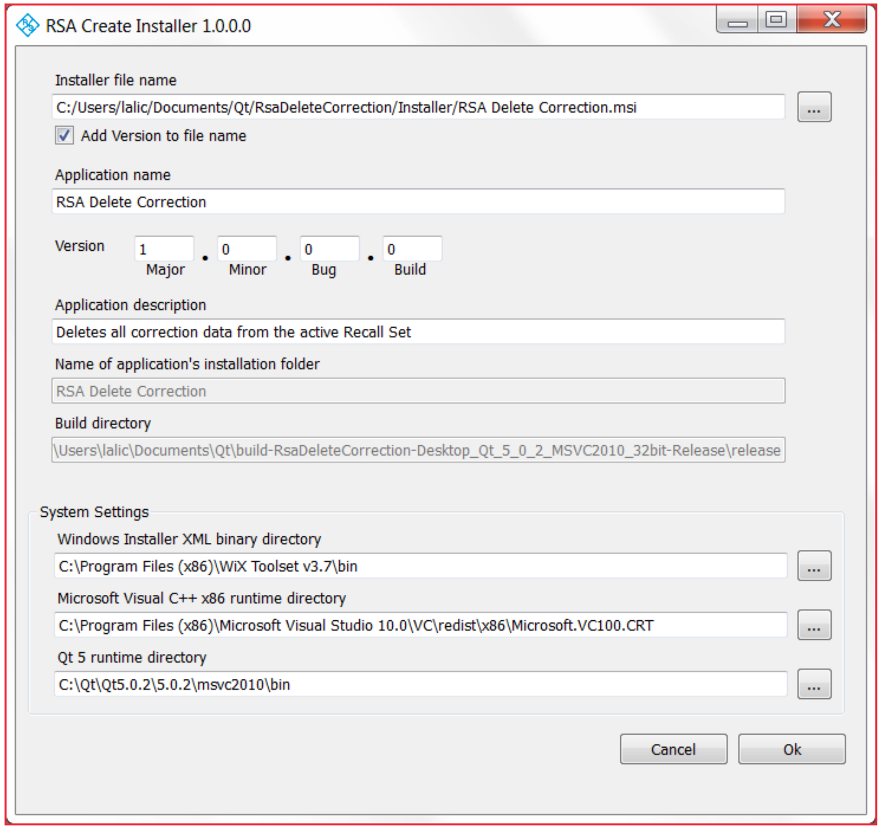
- **Qt 5 Runtime Directory**
  - We need to deploy a myriad of Runtime files associated with Qt 5:
    - Qt5Core.dll
    - Qt5Gui.dll
    - Qt5Widgets.dll
    - D3DCompiler_43.dll
    - Icudt49.dll
    - Icuin49.dll
    - Icuuc49.dll
    - libEGL.dll
    - libGLESv2.dll
  - These files are usually located at a directory similar to: C:\Qt\Qt5.0.2\5.0.2\msvc2010\bin
  - One additional file, qwindows.dll, is needed for Windows applications. This file can be found relative to the rest by RSA Create Installer.

RSA Create Installer Setup

- If you have not already done so, click the browse buttons to locate the WiX Toolset, Visual C++ Runtime and the Qt 5 Runtime.
- RSA Create Installer will attempt to display the default locations for you. If it cannot find the files you will have to navigate to them manually.
- As previously mentioned, you only have to do this once per system. RSA Create Installer will remember these settings for subsequent uses.

RSA Create Installer Setup

- At this point your installer settings should look similar to this:



RSA Create Installer 1.0.0.0

Installer file name
C:/Users/lalic/Documents/Qt/RsaDeleteCorrection/Installer/RSA Delete Correction.msi
[✓] Add Version to file name

Application name
RSA Delete Correction

Version 1 . 0 . 0 . 0 (Major . Minor . Bug . Build)

Application description
Deletes all correction data from the active Recall Set

Name of application's installation folder
RSA Delete Correction

Build directory
\Users\lalic\Documents\Qt\build-RsaDeleteCorrection-Desktop_Qt_5_0_2_MSVC2010_32bit-Release\release

System Settings

Windows Installer XML binary directory
C:\Program Files (x86)\WiX Toolset v3.7\bin

Microsoft Visual C++ x86 runtime directory
C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\redist\x86\Microsoft.VC100.CRT

Qt 5 runtime directory
C:\Qt\Qt5.0.2\5.0.2\msvc2010\bin

RSA Create Installer Setup

- Click Ok to accept these settings and create a Windows Installer
- If all goes well the status bar at the bottom of the window should display the message “Compiling…” and then “Linking… This may take some time”.
- The WiX linking process, even for simple applications like this, can take as long as 45 seconds. Please be patient.
- When WiX is finished RSA Create Installer will show you a dialog and then exit.
- Navigate to the project’s Installer folder and you should find your installer. It will be named “RSA Delete Correction 1.0.0.0.msi”, per our request. Notice that the version number “1.0.0.0” has been added for your convenience.

- Our application can now be deployed to any Rohde & Schwarz VNA!
- The log file for your deployed application can be found at:
  C:\Documents and Settings\All Users\Application Data\Rohde-Schwarz\RSA Delete Correction
  for a ZVX VNA running Windows XP, and on a ZNB:
  C:\ProgramData\Rohde-Schwarz\RSA Delete Correction\

Additional References

- Installation:
  - Qt 5
  - Visual Studio 2010 Express
  - Windows Installer XML (WiX) Toolset
- Licensing
  - Qt 5: GNU Lesser General Public License (LGPL) v2.1
  - Find End User License Terms for Microsoft Software