Hermes™ IoT Development Platform Qt Cross-Compiler Setup Instructions

1 Platform Overview

Hermes is built around a BeagleBone Black (BBB) (www.beagleboard.org) single-board computer controlling a wireless sensor reader board for communicating with sensor tags.

2 Objective

The main objective is to provide a Qt cross-compiler development environment for the BBB. It is possible to develop Qt applications for the BBB in a standalone mode. But this is not convenient for large applications. Moreover, developing in a standalone mode precludes the use of advanced tools, like Qt Creator. Though it is possible to enable some of these tools, the drawback is that they become very slow. A cross-compiler environment not only enables the use of advanced tools, but it also speeds-up the development process by orders of magnitude.

These are the specific objectives:

1. Provide a Microsoft Windows host-based cross-compiler Qt 5.5.1 development environment for the BBB.
2. Keep the BBB standalone Qt 5.5.1 development environment capability.
3. Enable the use of Qt Creator 3.5.1 IDE for Windows.
4. Provide debugger capabilities from Qt Creator.
5. Provide support for Qwt 6.1.2 and integration with the Qt Creator Designer.

3 BBB Setup

Follow the process described in the IN009F10 Hermes Debian Distribution Build Guide.pdf document to setup the BBB. The simplest path is to download the image, program an SD card and then flash the BBB. Users can follow the rest of that document in case there is need to add specific user customizations. The BBB image provides standalone and cross-compiler support for Qt 5.5.1 and Qwt 6.1.2. It does not provide support for local use of Qt Creator.

4 Windows Host Setup Process

This process has been greatly simplified by providing pre-compiled components. User can unzip and deploy these precompiled components to the specified directories. But in addition to this simplified process, the last sections of this document describe the process to build these precompiled components. These sections are fairly more complex and are marked as **Optional**.

The process requires the installation of several components. It is highly recommended to install these components in their default location. The instructions in this document assume that all installations are made to their default locations.

These are the main steps to setup the Windows host:

1. Install Python 2.7
2. Install Windows GNU Toolchain for BeagleBone
3. Install Debugger GDB
4. Install Qt 5.5.1
5. Install Qwt 6.1.2
6. Configure Qt for BBB Cross-Compiling
7. Qt Project Setup
8. SmarTTY

The optional sections are:

1. How to Build Qt 5.5.1 for BBB Cross-Compiling
2. How to Build Qwt 6.1.2 for BBB Cross-Compiling
3. How to Build the GDB Debugger for Qt

The required components are located in two files:

1. `Qt-CrossCompiler-Components.zip`
2. `qt-opensource-windows-x86-msvc2013-5.5.1.exe`

The first file contains several individual components. The second one is the Qt 5.5.1 installer. These two files can be downloaded from the following location:

https://github.com/RFMicron/Hermes/blob/master/SoftwareDocs/Qt-Cross-Compiler-Components.md

5 Install Python 2.7

Python is a requirement for the GDB debugger. The users that do not wish to use the debugger can skip this section.

For Windows 64-Bit use: python-2.7.11.amd64.msi

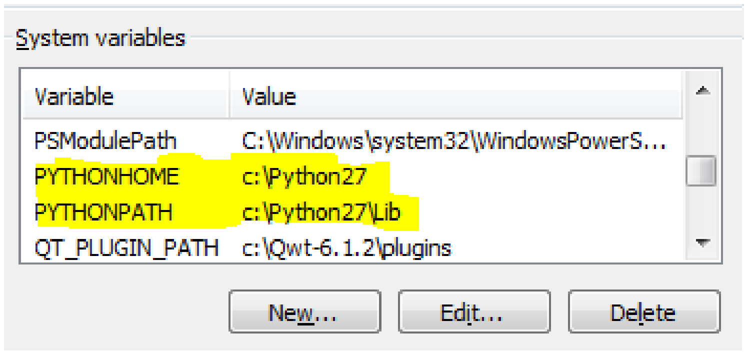
For Windows 32-Bit use: python-2.7.11.msi

Install Python in its default location: **c:\Python27**. Then add the following two system variables:

- `PYTHONHOME=c:\Python27`
- `PYTHONPATH=c:\Python27\Lib`

The variable definitions should look like in Figure 1. This dialog can be reached by navigating to:

Control Panel\System and Security\System



Variable	Value
PSModulePath	C:\Windows\system32\WindowsPowerS...
PYTHONHOME	c:\Python27
PYTHONPATH	c:\Python27\Lib
QT_PLUGIN_PATH	c:\Qwt-6.1.2\plugins

Figure 1 Python System Variables

Then select **Advanced system settings** in the left pane. Next, in the **Advanced** tab click on the **Environment Variables…** button.

6 Install Windows GNU Toolchain for BeagleBone

6.1 Toolchain Installation

The installer for the Windows GNU Toolchain for BeagleBone (Debian + Qt 5.5) is:

Beaglebone-gcc4.6.3-Debian-r2.exe

For reference, its original location is at: http://gnutoolchains.com/download/

Install the toolchain in its default path: **c:\SysGCC**. Then make sure the following path is part of the system variable **path**: `C:\SysGCC\Beaglebone\bin`

6.2 Add Support for Qt 5.5.1

There is a precompiled component for this step: **qt5-arm-linux-gnueabihf-bbb.zip**. First delete the following directory:

```
C:\SysGCC\Beaglebone\arm-linux-gnueabihf\sysroot\usr\local\qt5
```

Then unzip the precompiled component and place it in the following directory:

```
C:\SysGCC\Beaglebone\arm-linux-gnueabihf\sysroot\usr\local\qt5
```

6.3 Add Support for Qwt 6.1.2

There is a precompiled component for this step: **qwt-6.1.2-arm-linux-gnueabihf-bbb.zip**. Unzip it and place it in the following directory:

```
C:\SysGCC\Beaglebone\arm-linux-gnueabihf\sysroot\usr\local\qwt-6.1.2
```

7 Install Debugger GDB

This is a pre-built component that was compiled with Python support. This is not standard but it is required by the latest versions of the Qt Creator IDE.

The precompiled component is **gdb-7.8.2-with-python.zip**. Extract the file to the following directory: **c:\gnu\out**. Under the **out** directory there should be four folders: **bin**, **include**, **lib** and **share**.

8 Install Qt 5.5.1

The Qt 5.5.1 installer for Windows is: **qt-opensource-windows-x86-msvc2013-5.5.1.exe**.

The only reason why the Microsoft Visual Studio 2013 version is recommended is because this is the version that it is required for building Qwt. The other versions for Visual Studio and for MinGW should work fine for BBB cross-compiling as well.

Install Qt 5.5.1 to its default location: **c:\Qt\Qt5.5.1**. Visual Studio and MinGW are not requirements for BBB cross-compiling. Visual Studio and/or MinGW can be installed if Qt will also be used to develop applications for Windows.

9 Install Qwt 6.1.2

The precompiled component for this step is: **qwt-6.1.2-compiled-with-VS2013.zip**. To install Qwt perform the following steps:

1. Unzip the precompiled component file to the default Qwt location: **c:\qwt-6.1.2**.
2. Add the following system variable:
   a. `QT_PLUGIN_PATH=c:\qwt-6.1.2\plugins`
3. Open the **Qt 5.5 32-bit for Desktop** terminal located in the **Start** menu as shown in Figure 2 and type the following command:
   a. `qmake -set QMAKEFEATURES c:\qwt-6.1.2\features`

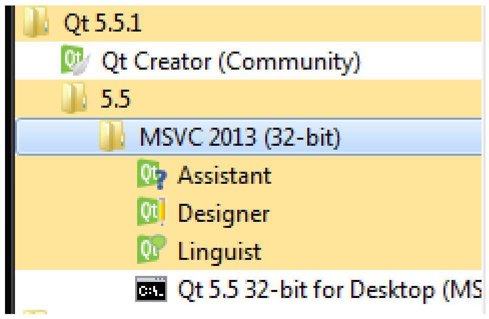


Figure 2 Qt 5.5 32-bit for Desktop Terminal

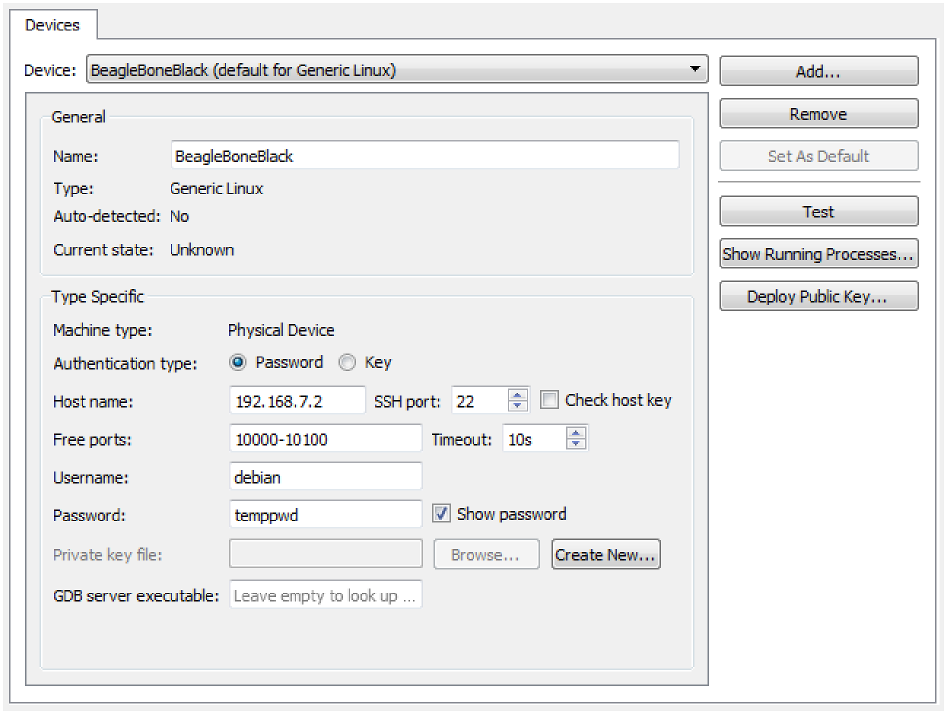
4. If Qwt is going to be used for Windows application development then the following path must be added to the system variable **path**:
   a. `c:\qwt-6.1.2\lib`
5. Download Qwt's help file **qwt-6.1.2.qch** located at:
   a. https://sourceforge.net/projects/qwt/files/qwt/6.1.2/
6. Install the help file with the following steps:
   a. Open Qt Creator.
   b. Open the settings dialog from the **Tools->Options** menu.
   c. Raise the tab **Help->Documentation**.
   d. Press the **Add** button and select the **qwt-6.1.2.qch** file.

10 Configure Qt Creator for BBB Cross-Compiling

10.1 Create a Device Specification

1. Open **Qt Creator** and select **Options** under the **Tools** menu.
2. In the left pane select the **Devices** icon.

3. Click the **Add** button and create the device configuration shown in Figure 3.



Devices

Device: BeagleBoneBlack (default for Generic Linux)

General

Name: BeagleBoneBlack
Type: Generic Linux
Auto-detected: No
Current state: Unknown

Type Specific

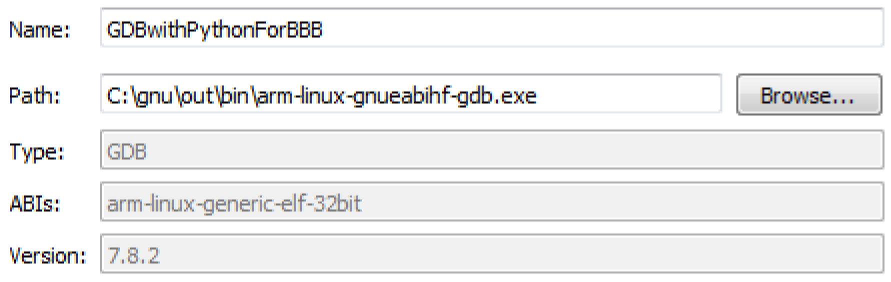
Machine type: Physical Device
Authentication type: [✓ Password] Key
Host name: 192.168.7.2 SSH port: 22 Check host key
Free ports: 10000-10100 Timeout: 10s
Username: debian
Password: temppwd [✓] Show password
Private key file:
GDB server executable: Leave empty to look up ...

Figure 3 Device Specification

4. Adjust the IP address as needed. In this case 192.168.7.2 is the default address when the BBB is connected to the host computer using the USB connection.
5. Click the **Test** button to verify the connection.
6. At this point, clicking the **Show Running Processes…** button must display a dialog with the BBB processes.

10.2 Create Build & Run Kit Configuration

1. Open **Qt Creator** and select **Options** under the **Tools** menu.
2. In the left pane select the **Build & Run** icon.
3. Select the **Debuggers** tab.
4. Click the **Add** button and create the configuration shown in Figure 4.

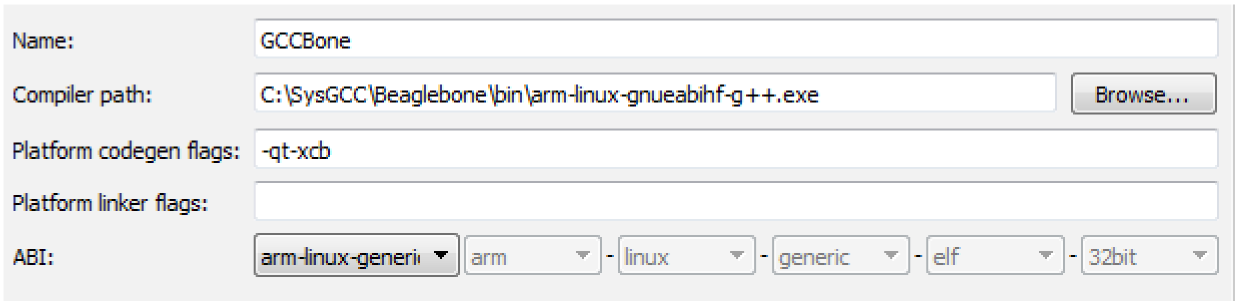


Name: GDBwithPythonForBBB
Path: C:\gnu\out\bin\arm-linux-gnueabihf-gdb.exe
Type: GDB
ABIs: arm-linux-generic-elf-32bit
Version: 7.8.2

Figure 4 Debugger Configuration

5. Select the **Compilers** tab.

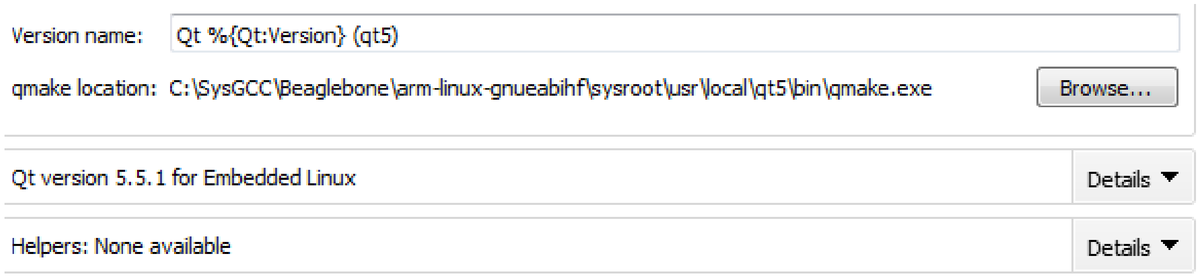
6. Click the **Add** button and create the compiler configuration shown in Figure 5.



Name:	GCCBone
Compiler path:	C:\SysGCC\Beaglebone\bin\arm-linux-gnueabihf-g++.exe
Platform codegen flags:	-qt-xcb
Platform linker flags:	
ABI:	arm-linux-generi - arm - linux - generic - elf - 32bit

Figure 5 Compiler Configuration

7. Select the **Qt Versions** tab.
8. Click the **Add** button and create the Qt version configuration shown in Figure 6.



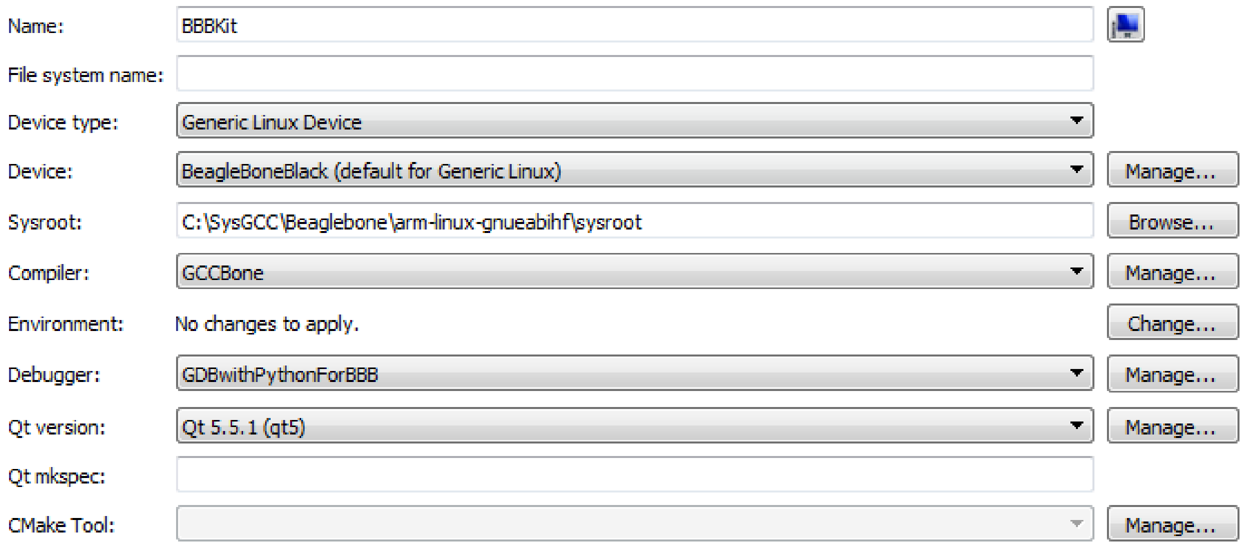
Version name:	Qt %{Qt:Version} (qt5)
qmake location:	C:\SysGCC\Beaglebone\arm-linux-gnueabihf\sysroot\usr\local\qt5\bin\qmake.exe

Qt version 5.5.1 for Embedded Linux

Helpers: None available

Figure 6 Qt Version Configuration

9. Select the **Kits** tab
10. Click the **Add** button and create the configuration shown in: Figure 7.



Name:	BBBKit
File system name:	
Device type:	Generic Linux Device
Device:	BeagleBoneBlack (default for Generic Linux)
Sysroot:	C:\SysGCC\Beaglebone\arm-linux-gnueabihf\sysroot
Compiler:	GCCBone
Environment:	No changes to apply.
Debugger:	GDBwithPythonForBBB
Qt version:	Qt 5.5.1 (qt5)
Qt mkspec:	
CMake Tool:	

Figure 7 Kit Configuration

11 Qt Project Setup

11.1 Qt Project Setup

11.1.1 Qwt Configuration

Applications that make use of the Qwt library must include the following settings in the project file (**.pro)**:

- `CONFIG += qwt`
- `INCLUDEPATH += C:/SysGCC/Beaglebone/arm-linux-gnueabihf/sysroot/usr/local/qwt-6.1.2/include`
- `LIBS += -LC:/SysGCC/Beaglebone/arm-linux-gnueabihf/sysroot/usr/local/qwt-6.1.2/lib`

If users want to compile the application locally in the BBB then the following line must be added at the top of the project file (**.pro**):

```
include (/usr/local/qwt-6.1.2/features/qwt.prf)
```

11.1.2 Target Settings

The following target settings must be included in the project file (**.pro**):

- `target.path = /home/debian`
- `INSTALLS += target`

The target path is where the executable is going to be deployed by the cross-compiler process. Users can adjust this path as needed.

12 SmarTTY

This application is located at:

```
C:\SysGCC\Beaglebone\TOOLS\PortableSmartty\SmarTTY.exe
```

This is a very useful SSH terminal emulator. The menu **SCP** includes very easy to use options for downloading and uploading files. SmarTTY also includes an X Server for Windows so the applications executed at the command line will be displayed as another window in the Windows host computer.

13 How to Build Qt 5.5.1 for BBB Cross-Compiling (Optional)

This procedure is based on the tutorial located at:

http://visualgdb.com/tutorials/beaglebone/qt5/cross-compile/

1. Install Python 2.7 (see Section 5).
2. Install the Windows GNU Toolchain for BeagleBone (see Section 6.1).
3. Download the GNU toolchain for MinGW32 installer **mingw32-gcc4.8.1.exe** located at: http://gnutoolchains.com/mingw32/.
4. Install the GNU toolchain for MinGW32 in its default directory: `C:\SysGCC\MinGW32`
5. Download the Qt 5.5.1 source package **qt-everywhere-opensource-src-5.5.1.tar.xz** located at: http://download.qt.io/archive/qt/5.5/5.5.1/single/
6. Launch the msys shell from the MinGW toolchain located at:
    a. `C:\SysGCC\MinGW32\msys\1.0\msys.bat`
7. Go to the directory containing the archive with the Qt source and extract with the following command: `tar xf qt-everywhere-opensource-src-5.5.1.tar.xz`
8. Make sure that the following path: `C:\SysGCC\MinGW32\bin` is part of the system variable **path**. **WARNING: only one MinGW installation can be part of the system path. Otherwise msys will not find the right compiler and the cross-compiler process will fail.**
9. Open the file located in the Qt source code under the path: `qtbase\mkspecs\devices\linux-beagleboard-g++\qmake.conf` and perform the following two changes:
    a. Change the value of the **QT_QPA_DEFAULT_PLATFORM** variable to **xcb**.
    b. Remove or comment the compiler flag **–mfloat-abi=softtp**
10. Open the file located in the Qt source code under the path: `qtbase\mkspecs\win32-g++\qmake.conf` and perform the following change:
    a. Add the option **–U__STRICT_ANSI__** to **QMAKE_CXXFLAGS**.
11. Create a directory for the build process and start the build with the following commands:
    a. `mkdir qt-build`
    b. `cd qt-build`
    c. `../qt-everywhere-opensource-src-5.5.1/configure -platform win32-g++ -xplatform linux-arm-gnueabi-g++ -release -device linux-beagleboard-g++ -sysroot C:/SysGCC/Beaglebone/arm-linux-gnueabihf/sysroot -prefix /usr/local/qt5 -no-largefile –opensource –confirm-license`
12. Within a few minutes the build process will fail. This is normal as long as **qmake.exe** got built. Check this by running **qtbase/binqmake –v**
13. Open the file, within the Qt source tree, **qtbase\configure** and change the line number 3913 (just after the line with the **#build qmake** text) from this text: **if true; then ###['!' -f "$outpath/bin/qmake"];** to this text: **if ['!' -f "$outpath/bin/qmake.exe"]; then**
14. Start the configure script again but this time with the following command:
    a. `../qt-everywhere-opensource-src-5.5.1/configure -platform win32-g++ -xplatform linux-arm-gnueabi-g++ -release -device linux-beagleboard-g++ -sysroot C:/SysGCC/Beaglebone/arm-linux-gnueabihf/sysroot -prefix /usr/local/qt5 -no-largefile –`

```
opensource -confirm-license -device-option
CROSS_COMPILE=C:/SysGCC/Beaglebone/bin/arm-linux-gnueabihf- -qt-
xcb
```

15. The configuration process will last a couple more minutes. Once the configure script reports that the configuration is complete, run the **make && make install** command to build the entire Qt framework and install it into the cross-compiler directory. This process will last several hours, even on a fast computer.
16. At the end of the build process the compiled Qt framework will be placed at:
    a. `C:\SysGCC\Beaglebone\arm-linux-gnueabihf\sysroot\usr\local\qt5`

14 How to Build Qwt 6.1.2 for BBB Cross-Compiling (Optional)

This process has to be done twice:

1. First Qwt is built for Windows so Qwt can be used from Qt Creator and Qt Designer.
2. Then Qwt is built for the BBB's Debian distribution so Qwt can be cross-compiled in the Windows host computer and executed on the BBB.

14.1 How to Build Qwt for Windows

1. Download the source code for Windows (**qwt-6.1.2.zip**) located at:
    a. https://sourceforge.net/projects/qwt/files/qwt/6.1.2/
2. Unzip the Qwt source code to its default location: **c:\qwt-6.1.2**
3. Qt 5.5.1 for Visual Studio 2013 must be installed (see 4). It has to be the version for Visual Studio because if it is not then the Qwt widgets will not show up in the components tab of Qt Designer.
4. Install Visual Studio 2013. The free version can be used which is: Visual Studio Express 2013 for Windows Desktop with Update 5. This version can be downloaded from:
    a. https://www.microsoft.com/en-us/download/details.aspx?id=48131
5. It's important to clarify that the default location of Visual Studio 2013 is in the following directory:
    a. `C:\Program Files (x86)\Microsoft Visual Studio 12.0`
6. Open the **Developer Command Prompt for VS2013** located at:
    a. `C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Tools\Shortcuts`
7. Execute the following commands:
    a. `cd c:\qwt-6.1.2`
    b. `c:\Qt\Qt5.5.1\5.5\msvc2013\bin\qmake qwt.pro`
    c. `nmake`
    d. `nmake install`
8. Qwt will be built and installed in its default directory: **c:\qwt-6.1.2**

14.2 How to Build Qwt for the BBB's Debian Distribution

The following steps must be performed in a BBB with Debian 7.9 and Qt 5.5.1:

1. Download the source code for Linux (**qwt-6.1.2.tar.bz2**) located at:
    a. https://sourceforge.net/projects/qwt/files/qwt/6.1.2/
2. Execute the following commands:
    a. `tar xvf qwt-6.1.2.tar.bz2`
    b. `cd qwt-6.1.2`
    c. `qmake qwt.pro`
    d. `make`
    e. `sudo make install`
3. Qwt is installed in the **/usr/local/qwt-6.1.2** directory. This is the folder that will end up being copied to the host Windows computer (see 6.3).

15 How to Build the GDB Debugger for Qt (Optional)

The latest versions of Qt Creator require a debugger with Python support. GDB, the GNU debugger, is usually built and distributed without Python support. And this is the case with the debuggers that are shipped with the Windows GNU Toolchain for BBB and with the GNU Toolchain for MinGW32. Therefore a GDB with Python support has to be built in order to provide debugger functionality to Qt for BBB.

This build process is based in the process described in the following page:

http://gnutoolchains.com/building/

1. Download MinGW and MSys. Use the installer **mingw-get-setup.exe** from the following location:
    a. https://sourceforge.net/projects/mingw/files/MinGW/Base/
2. Execute the **mingw-get-setup.exe** installer and make sure the following **Basic Setup** packages are selected:
    a. `mingw-developer-toolkit`
    b. `mingw32-base`
    c. `mingw32-gcc-g++`
    d. `msys-base`
3. Then click on **All Packages** and select the **mingd32-libexpat** class **dev**.
4. Next click on **Apply Changes** under the **Installation** menu. MinGW is installed in its default location **c:\MinGW**
5. Open MSys by double-clicking the file **msys.bat** located at: **c:\MinGW\msys\1.0**
6. Make sure there are no other MinGW compilers in the system path. This can be checked with the following command**: echo $PATH**
7. Create the following default directory for building GNU toolchain components: **c:\gnu**

8. Download the source code of GDB version 7.8.2. This is the version that it is compatible with the Qt cross-compiler toolchain. The source code is in the file **gdb-7.8.2.tar.gz** that can be downloaded from the following location:
   a. ftp://sourceware.org/pub/gdb/releases/
9. Place the **gdb-7.8.2.tar.gz** file in the **c:\gnu** directory and execute the following commands:
   a. `tar xvf gdb-7.8.2.tar.gz`
   b. `mkdir out`
   c. `mkdir gdb-7.8.2-build`
   d. `cd gdb-7.8.2-build`
   e. `../gdb-7.8.2/configure --host=i686-pc-mingw32 --target=arm-linux-gnueabihf --with-expat --prefix /c/gnu/out --with-python`
   f. `make`
   g. `make install`
10. The compiled GDB will be installed in the following directory: **c:\gnu\out**

16 Notices