Introduction

This document describes what is needed to do Qt development with Qt Creator and the Yocto-generated SDK/toolchain for the RZ/G.

Target Device

RZ/G1

Contents

1. Overview

This document describes how to integrate the cross-compilation SDK/Toolchain for the RZ/G into the Qt Creator IDE and use it to build applications. Both the SDK and Qt Creator are installed and run on a Linux Host machine running a popular Linux distribution line Ubuntu.

2. Quick Steps

This is just a summary of the steps needed to setup a development environment with Qt Creator and the RZ/G cross-compilation SDK/Toolchain. Please see the following sections for detailed instructions.

1. Install the Qt framework and Qt Creator:

```
$ sudo apt install qt-sdk qtcreator
```

2. Install the RZ/G SDK/Toolchain:

```
$ sudo ./poky-glibc-x86_64-core-image-weston-sdk-cortexa7hf-vfp-neon-
toolchain-2.0.1.sh
```

3. Create a blank file named ‘oe-device-extra.pri’ in the SDK install folder:

```
$ sudo touch /opt/poky/2.0.1/sysroots/cortexa7hf-vfp-neon-poky-linux-
gnueabi/usr/lib/qt5/mkspecs/oe-device-extra.pri
```

4. Setup the environment and launch Qt Creator:

```
$ source /opt/poky/2.0.1/environment-setup-cortexa7hf-vfp-neon-poky-linuxgnueabi
$ qtcreator
```

5. Integrate the RZ/G SDK into Qt Creator:
    a. From the menu, select Tools -> Options to open the Options dialog
    b. Select ‘Devices’ from the list and add a new Generic Linux Device for the RZ/G
    c. Select ‘Build & Run’ from the list
    d. Select ‘Compilers’ tab, add a new GCC compiler and specify the path to the C++ cross compiler in the SDK install folder, e.g. /opt/poky/2.0.1/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-cpp
    e. Select ‘Qt Versions’ tab and add the Qt version from the RZ/G SDK by providing the path to qmake, e.g. /opt/poky/2.0.1/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake
    f. Select ‘Kits’ and add a new kit specifying the created Device, Qt version and compiler. Provide the path to the target sysroot in the SDK, e.g. /opt/poky/2.0.1/ sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi

3. Installing Qt and Qt Creator

The Qt framework and the Qt Creator IDE are cross-platform and can be installed on a machine running Windows, Linux or OS X. The RZ/G SDK/Toolchain however runs on Linux, so we will use a Linux Host machine. It doesn’t have to be a physical machine (a desktop or a laptop), you can also use a virtual machine. Please see other app notes on setting up Linux VMs.

3.1 Installing from distribution repositories

The easiest way to install Qt and Qt Creator is to use your distribution’s package manager (e.g. ‘apt’ for Ubuntu). One downside to this method is that the versions of Qt and Qt Creator installed from your distribution’s repositories would typically be relatively old.

Note that the version of the Qt framework installed on your Development Host machine is not related to the version that runs on the RZ/G, so the two can be different.

To install Qt framework and Qt Creator on Ubuntu or other Debian-based distribution, run the following command:

```
$ sudo apt install qt-sdk qtcreator
```

3.2 Manual installation

Download the desired version of the Qt framework, e.g. qt-opensource-linux-x64-5.6.2.run. This file is an installer, so run it on your Linux Host machine and follow the steps. Everything is installed in a single folder, e.g. ‘Qt5.6.2’. The executable that we will be launching is ‘qtcreator’, located under ‘Qt5.6.2/Tools/QtCreator/bin’. You can add that folder to the environmental variable PATH, that way Qt Creator can be launched from any folder.

4. Installing the RZ/G SDK

The RZ/G SDK contains the tools and other components necessary to develop software for the RZ/G. This is a cross compilation SDK, i.e. it runs on one architecture (Intel x86) and produces executables that run on a different architecture (ARM).

Pre-built RZ/G SDK images are available for download from renesas.com and on the Renesas Marketplace. There are different versions of the SDK based on the specific target architecture (Cortex A7 or A15) and the Linux Host machine's architecture (64-bit or 32-bit Intel x86).

You can also build an SDK yourself using the same Yocto build environment that is used for building OS images for the RZ/G. Please see other app notes or the online instructions on elinux.org on setting up and using the Yocto build environment and generating SDKs.

When building the SDK image yourself make sure to include the Qt libraries in it. They are needed for the cross compilation of Qt applications using the SDK. One easy way to include the Qt libraries is to use the configuration files (bblayer.conf and local.conf) from the 'QT-HMI' demo image available in the Yocto layer 'meta-rzg-demos'.

The SDK is distributed as a single large shell script. Run the script on your Linux Host machine to install it:

```
$ sudo ./poky-glibc-x86_64-core-image-weston-sdk-cortexa7hf-vfp-neon-toolchain-2.0.1.sh
```

By default, the SDK is installed under '/opt' but you can be installed in any location. If you install the above SDK in the default location, it will be under '/opt/poky/2.0.1/'.

The SDK installation folder contains the following components –

- Environment setup script – this script prepares the SDK for use by setting various environment variables. It needs to be run at the beginning of each new session
- Host sysroot – located in a folder named after the Host architecture, e.g. x86_64-pokysdk-linux. This folder contains the tools that run on the Host – the GCC cross compiler suite and others
- Target sysroot – located in a folder named after the Target architecture, e.g. cortexa7hf-vfp-neon-poky-linux-gnueabi. The contents of this older is based on the target OS image. It contains libraries, headers and other files that are needed for building images for the target architecture

4.1 Setting up the SDK for Qt development

Create an empty file named oe-device-extra.pri in the target sysroot – this is needed to resolve a warning in Qt Creator. If the SDK is installed in its default location the file should be located under <SDK install dir>/sysroots/<Target sysroot>/usr/lib/qt5/mkspecs. For example:

```
$ sudo touch /opt/poky/2.0.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi/usr/lib/qt5/mkspecs/oe-device-extra.pri
```

Inspect the SDK's environment setup script (e.g. /opt/poky/2.0.1/environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi). There should be several lines towards the end of the script starting with 'export OE_MAKE_...'. If you don't see such lines, add them manually:

```
export OE_QMAKE_COMPILER="${CC}"
export OE_QMAKE_CC="${CC}"
export OE_QMAKE_CFLAGS="${CFLAGS}"
export OE_QMAKE_CXX="${CXX}"
export OE_QMAKE_CXXFLAGS="${CXXFLAGS}"
export OE_QMAKE_LINK="${CXX}"
export OE_QMAKE_LDFLAGS="${LDFLAGS}"
export OE_QMAKE_AR="${AR}"
export OE_QMAKE_STRIP="echo"
export QT_CONF_PATH="${WORKDIR}/qt.conf"
```

5. Launching Qt Creator

Always launch Qt Creator from a terminal window.

Prepare the environment by sourcing the SDK's environment setup script:

```
$ source opt/poky/2.0.1/anvironment-setup-cortexa7hf-vfp-neon-poky-linuxgnueabi
```

Launch Qt Creator:

```
$ qtcreator
```

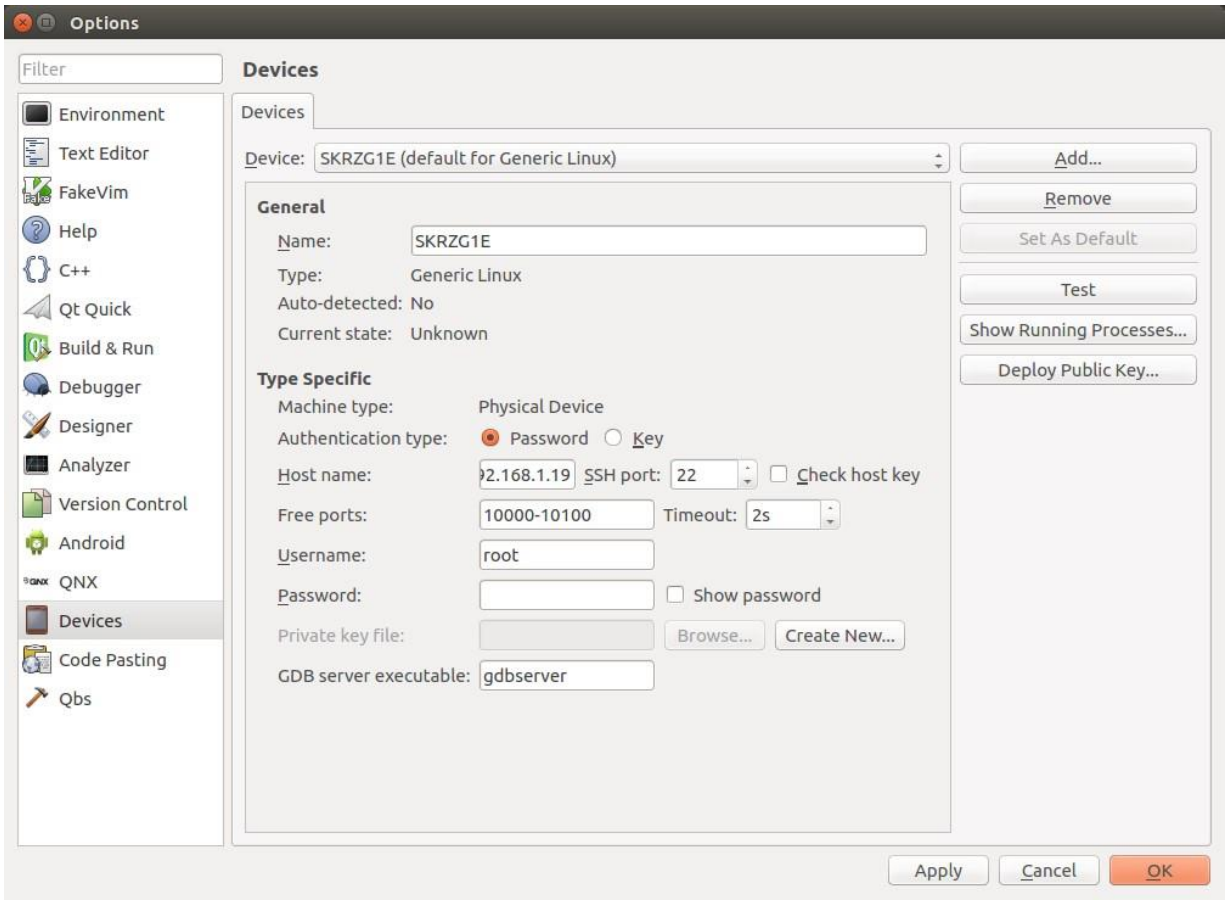
If you installed Qt Creator manually and did not add the path to its 'bin' directory to your PATH variable, you will need to specify the path to the qtcreator executable in the above command (or just 'cd' to the folder where the executable is)

6. Integrating the RZ/G SDK into Qt Creator

6.1 Create a new Device

A 'Device' in Qt Creator represents the target platform – Desktop, Android, Embedded Linux, etc. You need to create at least one device of type 'Generic Linux Device' to use with the RZ/G.

From the Qt Creator's menu select Tools -> Options to open the Options dialog. Select 'Devices' from the list on the left and add a new Generic Linux Device. The device creation wizard asks for device's host name or IP address and a user name to log into the device. You can enter any values for now so that the wizard can finish the device creation.



6.2 Integrate the RZ/G SDK

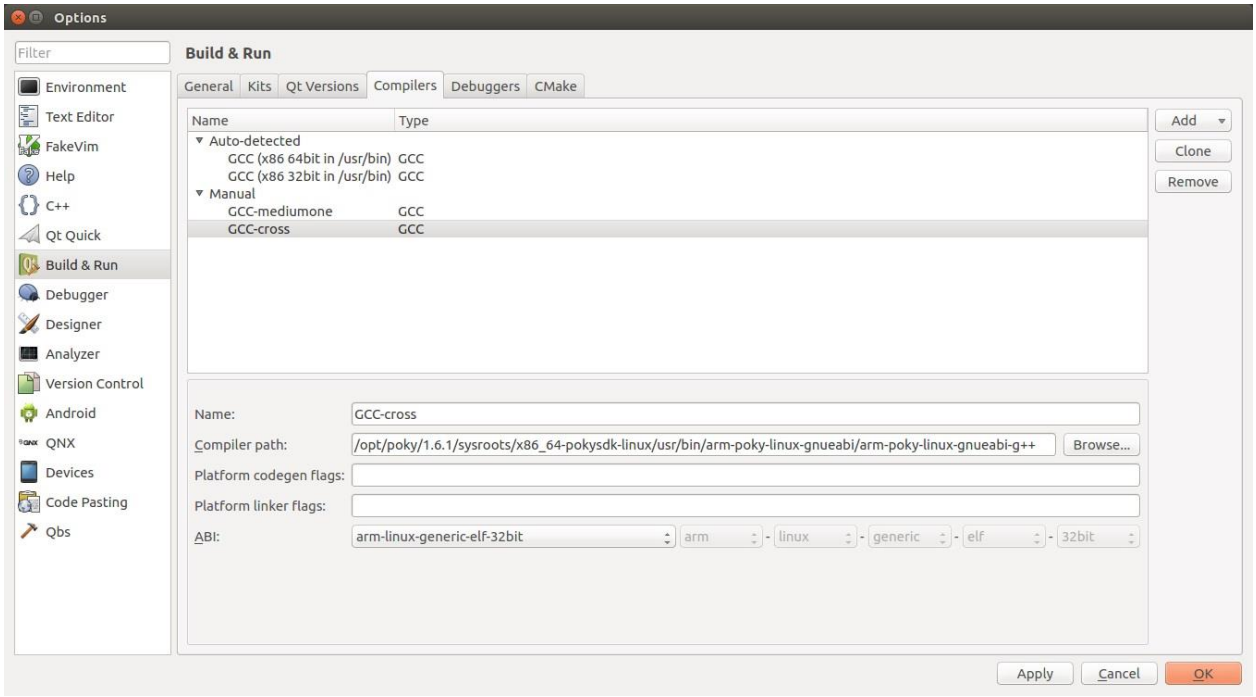
This is done using the same Options dialog, section 'Build & Run'. There are several tabs that need to be configured – Compilers, Debuggers, CMake, Qt Versions, Kits.

6.2.1 Compilers

Click 'Add' to add a new compiler, select GCC' and specify a name for the compiler, e.g. 'GCC-cross'. Specify the path to the C++ cross compiler in the RZ/G SDK's install location. For example –

/opt/poky/2.0.1/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-cpp

There is no need to configure any other compiler settings.



6.2.2 Debuggers

Add a new debugger, give it a name and specify the path to the cross debugger in the installed SDK. It is in the same folder as the C++ compiler, e.g.

/opt/poky/2.0.1/sysroots/x86_64-pokysdk-linux/usr/bin/arm-poky-linux-gnueabi/arm-poky-linux-gnueabi-gdb

6.2.3 CMake

You shouldn't need to do anything on this tab. Qt Creator should automatically detect your Host machine's cmake and add it here. If it doesn't detect it that may indicate that you don't have a Host development environment installed on your Linux machine. You can install that along with other useful tools by installing the recommended host package by Yocto, e.g. –

sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat

6.2.4 Qt Versions

The Qt version associated with the RZ/G SDK should be automatically detected during the compiler selection step. If it isn't - add it manually. Click Add and specify the path to the qmake executable in the SDK, e.g.

/opt/poky/2.0.1/sysroots/x86_64-pokysdk-linux/usr/bin/qt5/qmake

6.2.5 Kits

A 'kit' combines all previous components (device, compiler, debugger, Qt framework) into one configuration. You need to create a separate kit for each unique combination of those components.

Click 'Add' and specify a name for the kit. Select Device type 'Generic Linux Device' and then select the device created before.

Next, specify the sysroot. This should be the 'target' sysroot in the SDK's install location, not the 'host' sysroot that was used on the previous steps. E.g. –

/opt/poky/2.0.1/sysroots/cortexa7hf-vfp-neon-poky-linux-gnueabi

Select the compiler, debugger and Qt version created on previous steps. Leave the field 'Qt mkspec' empty and make sure that the CMake Tool is specified.

Qt Creator is now ready for building Qt applications for the RZ/G.

7. Application development and debugging

7.1 Application development

To build Qt applications for the RZ/G make sure to execute ('source') the SDK's environment setup script before launching qtcreator from a new console window. When creating new projects make sure to select one of the RZ/G 'kits' that you have configured in Qt Creator.

Follow these steps to create a test Qt GUI application and build it -

- From the Qt Creator menu select File -> New File or Project
- Select 'Qt Quick Application'
- Specify a name and location for the project
- Specify the Qt version
- Select the RZ/G kit created on previous steps

Build the project – this should complete with no errors and generate an executable that can be run on the RZ/G

7.2 Debugging

Qt Creator can automatically download the generated executables to the target board and remotely debug them. To do that it needs the following tools to be present on the board – gdbserver, a ssh server and a sftp server.

7.2.1 Necessary tools on the target board

(a) **gdbserver**

This is an application that runs on the RZ/G board and handles the executable that is being debugged. The remote GDB running on the Development Host machine connects to the gdbserver on the board.

An easy way to add gdbserver and other debugging tools to the OS image generated by Yocto is to add "debug-tweaks" and "tools-debug" to the EXTRA_IMAGE_FEATURES variable in local.conf.

(b) **A ssh server**

Qt Creator opens a SSH session to the target board during debugging, so the target board needs to have support for SSH. In most cases that is enabled on the Yocto generated images by default, if it isn't the above modification to local.conf would do that.

(c) **A sftp server**

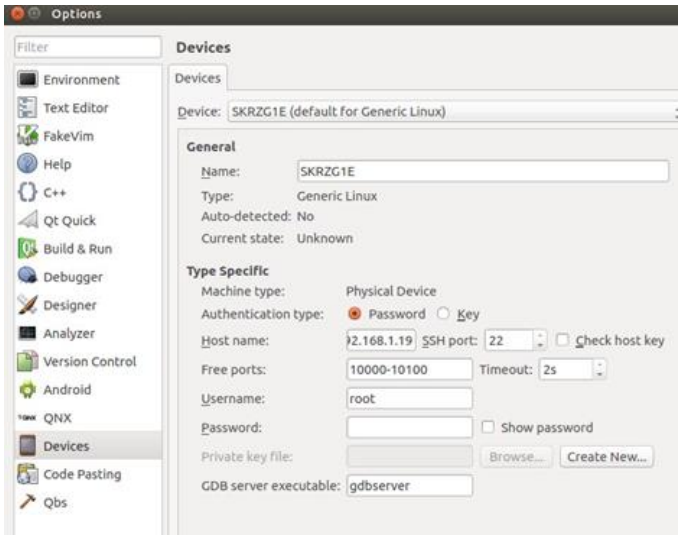
Qt Creator uses SFTP to download images to the target, so some SFPT server needs to be available on the target. The package openssl provides an SFTP server but it is not included in the images by default. To include it, add "openssh-sftp-server" to the IMAGE_INSTALL variable in local.conf.

7.2.2 Qt Creator configuration

Open the Devices section on the Options dialog and make sure that the RZ/G device created before is selected. Enter or modify the following fields –

- Host name – enter the IP address or Host name of the target board
- Timeout – make sure that the timeout is not 0, which would result in errors during debugger startup
- Username/Password for logging to the target board
- GDB server executable – type gdbserver

You should now be able to debug the application.



Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Apr 17, 2018	—	First edition issued