Devkit Yocto and Qt5 for emcon_rzg1e

Documentation

Rev002 / 07.11.2016

emtrion GmbH

Revision: **002 / 07.11.2016**

Rev	Date/Signature	Changes
001	Rr/02.08.2016	First release
002	Rr/07.11.2016	Minor corrections

1 Introduction

This developer kit is based on the module emcon_rzg1e and the base board avari from emtrion. The BSP is a linux kernel version 3.10.31, provided by Renesas and optimized and adapted by emtrion. The RootFs has been created with Yocto Openembedded, daisy 1.6.3. The base recipes are coming from Renesas, created for a special Renesas board, but they have been modified and extended by emtrion so that Qt5 with OpenGL is also supported now.

This manual describes the contents of the developer kit DVD, how to set it up and gives a short overview on how to debug Qt5 applications with QtCreator.

It is assumed that users of emtrion Linux developer kits are already familiar with u-boot, with Linux, with Yocto and with creating and debugging applications with Qt5.2.1 and QtCreator. General Linux and programming knowledge are out of the scope of this document. Emtrion is happy to assist you in acquiring this knowledge. If you are interested in training courses or getting support, please contact the emtrion sales department.

2 Terms and Definitions

Term	Definition
Target	Module emcon_rzg1e with baseboard avari
Host	Workstation, Developer PC
Toolchain	Compiler, Linker, etc.
RootFS	Root file system, contains the basic operating system
Console	Text terminal interface for Linux
NFS	Network File System, can share directories over network
NFS_SHARE	Directory that is exported by the NFS for the purpose of updating and booting via network
U-Boot	Bootloader, hardware initialization, updating images, starting OS
YP	Yocto Project
INST_DIR	Directory where Yocto and the meta-layers are installed
MACHINE	Specifies the target device for which the image is built. The variable is set to emcon-rzg1e here
BUILD_DIR	Machine dependent build directory
BSP	Board Support Package
SDK	Software Development Kit

3 Preconfigured Linux virtual machine

emtrion delivers a DVD accompanying the Yocto Linux developer kits. This DVD contains a VMware virtual machine running Ubuntu 14.04 (Trusty Tahr). VMware player or VMware Workstation is used to start the virtual machine. For general information about VMware Player please read following guide:

Getting started with VMware Player 5:
http://www.vmware.com/pdf/desktop/vmware_player50.pdf

The virtual machine on the DVD is a compressed ZIP archive. Uncompress it on your system where it suits you. Please consider that the size of the virtual disk file will increase while you are working with it.

After uncompressing, basically two things have to be setup correctly to be able to use the virtual machine with our developer Kit:

- **Network Connection**: this should work directly, but depending on your PC you have to make adjustments. If so please read the guide, mentioned above (p.99)
- **Serial Connection**: See p.85 in the guide above on how to use the serial port of your PC.

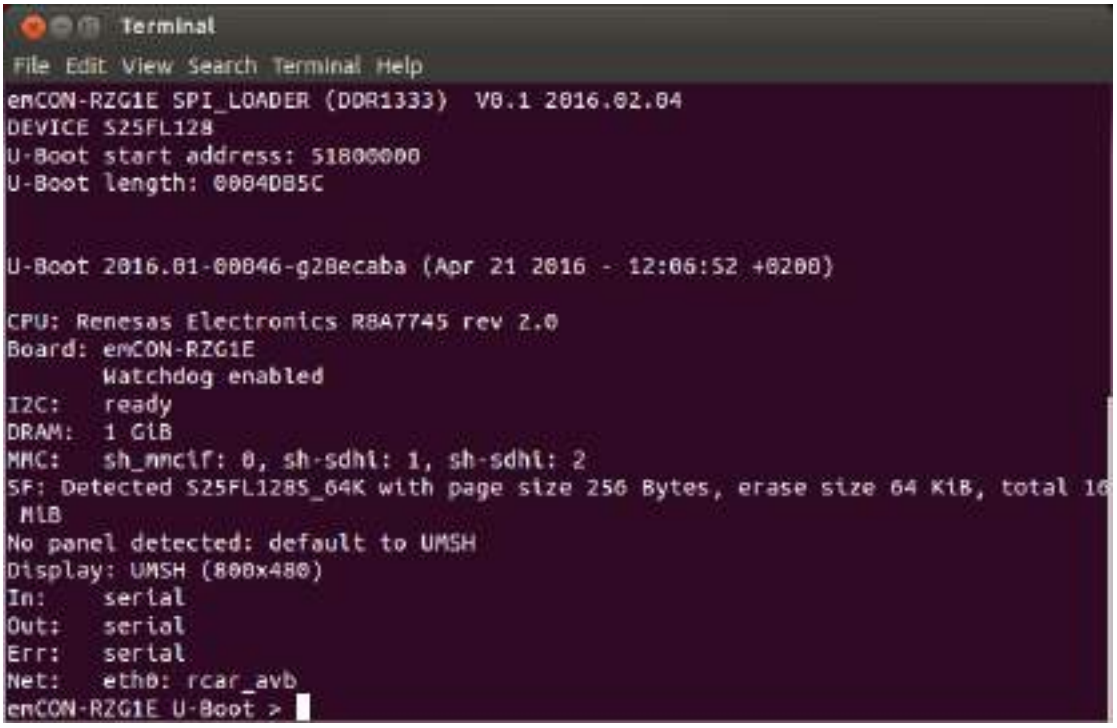
Some useful facts about the guest system inside the virtual machine:

- For login use **username: hico, password: hico** – it is recommended to change the password
- A TFTP Server is preinstalled and running by default. Its root directory is /tftpboot
- Serial ports added to the virtual machine appear at /dev/ttySn, USB serial converters at /dev/ttyUSBn, baudrates can be configured using the stty tool, picocom can be used as a terminal
- QtCreator is preinstalled as an IDE – have a look at the corresponding chapter in this manual on how to use it with our developer kit
- An NFS server exporting the /home/hico/nfs
- The serial terminal program picocom for connecting to the target
- Additional packages required by the build system Yocto
- The Yocto Layer **meta-emtrion** and **meta-emtrion-bsp** adapted to the devkit
- The version control **git**

Due to the settings of the VM and getting a system with good performance, the host has to be equipped with sufficient RAM (8 GBytes), free disk space (64 GBytes) and at least 4 cores for compiling Yocto.

3.1 Device Start Up

Connect the developer kit to the serial port attached to the virtual machine and to your network. Now you can open a serial terminal by using the link “Open Serial Terminal” on the Desktop.



1: Serial terminal showing U-Boot prompt

You may now power on the developer kit. You should see it booting U-Boot and Linux from Flash.

After the developer kit booted you are prompted for login:

- **user: root**
- **password: hico**

3.2 Device Network Setup

Per default the developer kit is setup to use a dhcp server. This is configurable by a bootloader environment variable “ip-method”. This variable can have the values “dhcp” or “static”.

You can check if there is a valid ip address with the command ifconfig.

```
Terminal
File Edit View Search Terminal Help
root@emcon-rzg1e:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1C:1E:00:15:CE
          inet addr:172.26.1.18  Bcast:172.26.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:30217 errors:0 dropped:233 overruns:0 frame:0
          TX packets:24757 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:36747621 (35.0 MiB)  TX bytes:3833902 (3.6 MiB)
          Interrupt:195 DMA chan:ff

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@emcon-rzg1e:~#
```

2: ifconfig output

If the setup is not correct you have to do it manually. Please check the description of the bootloader configuration on how to set up the variable “ip-method”.

Write down the IP address of the device. You need it to setup the connection in QtCreator.

3.3 Preconfigured Directories

Some directories have been created and filled with precompiled images. Below are their descriptions.

Placeholder	Path	remark
NFS_SHARE	/home/hico/nfs	Directory exported by the NFS
NFS_ROOTFS	/home/hico/nfs/root/rootfs	Rootfile system for mounting via nfs
NFS_IMAGE	/home/hico/nfs/images	Location of the images for updating the onboard flashes
INST_DIR	/home/hico/work/openembedded	Directory where Yocto and all the meta-layers will be stored to
BUILD_DIR	/home/hico/work/openembedded/builddir /machines/emcon-rzg1e/native	Build directory of the build system.
DOWNLOAD_ DIR	/home/hico/Downloads	Pre-built images, SDK
SDK_DIR	/opt	SDK with tools, sources and libraries
TFTP_DIR	/tftpboot	Used by the tftp server for exporting files

These variables are used as a place holder in this manual. They are set as soon as the build or setup script for yocto has been called.

3.4 Pre-built images and installations

In the directory **$DOWNLOAD_DIR** are two images of the root file system located, one with and one without SDK extensions. In addition you can find the SDK as a self extracting script in this directory and tar archives with recipes for building your own root file system. The images have been created and tested by emtrion and you can use them for designing your application with QtCreator.

To keep the virtual machine as small as possible for transport no further installations have been done so far. It's up to you to perform some steps before you can start to work with the developer kit.

The very first step is to unpack and copy the archives in **$DOWNLOAD_DIR** to the correct places. For this purpose there is a script file located in the home directory with the name "unpack_devkit.sh".

Please navigate to the home directory /home/hico and call

source ./unpack_devkit.sh

Afterwards the directory **$SDK_DIR** is populated with the unpacked SDK and **$NFS_SHARE** is prepared with the unpacked images of the root file system. The archives with the recipes for Yocto have been copied and unpacked to **$INST_DIR**.

4 The meta-layer for emcon_rzg1e

If you plan to work with QtCreator and you do not want to create new root file systems, you can skip this chapter.

For working with Yocto a special directory setup is needed. In the table below some of the most important directories of the layer's directory structure are listed.

name	remarks
meta-emtrion/	main layer emtrion
./meta-emtrion-bsp/	sub layer bsp
./meta-emtrion-bsp/meta-emtrion-rzg_1x	sub layer for renesas rzg1x SOC
./meta-emtrion-bsp/meta-emtrion-rzg_1x /build-config/emcon-rzg1e	configuration files for emcon-rzg1e for setting up the build environment
./meta-emtrion-bsp/meta-emtrion-rzg_1x /conf/	configuration files of the sub layer
./meta-emtrion-bsp/meta-emtrion-rzg_1x /recipes-bsp/u-boot/	U-boot recipe
./meta-emtrion-bsp/meta-emtrion-rzg_1x /recipes-kernel/linux/	kernel recipe
. meta-emtrion-bsp/meta-emtrion-rzg_1x /recipes-graphics/images/	image recipes
./meta-emtrion-bsp/meta-emtrion-rzg_1x /scripts/	installation script

4.1 Preparing Yocto

Before starting of any bitbaking process, the Yocto build system has to know about some details about the machine and all of the layers required for. This information is stored in the configuration files bblayers-x.conf and local-x.conf. Both are located in the directory meta-emtrion-bsp. X is a place holder for special variants of the root file system. This developer kit supports a system where Qt resides directly on OpenGL without any window manager like X11 or Wayland. For this "native" approach x has to be replaced by "native":

bblayers-native.conf and local-native.conf

Both are renamed and copied to **$BUILD_DIR**/conf .The file bblayers.conf includes the different required meta-layers. Local.conf defines the machine and some other settings like the download and sstate-cache directory and more.

4.1.1 Script for setting up Yocto

The very first preparation process is done by sourcing the script **yocto_setup_rzg1e-native**. Please navigate to the directory **$INST_DIR** and call

source ./meta-emtrion-bsp/meta-emtrion-rzg_1x/scripts/ yocto_setup_rzg1e-native

The script is setting up the complete build system as below.

- The Yocto version is based on the release **daisy**.
- Installation of all the required layers by accessing the corresponding repository. If a layer is already installed before it is updated if necessary.
- Modifies the configuration files with values specified inside the installation script.

- Creating of the specified directory structure
- Copying of the maybe modified configuration files to the directory of **$BUILD_DIR/conf**.
- Setting up the build environment.

During the installation process, the script creates three main directories. These directories have a special meaning in the scope of the build process. The directories are listed below.

directory	remarks
$BUILD_DIR/	machine dependent build directory, where all the outputs will be stored
$INST_DIR/builddir/downloads/	directory, where all the downloads are copied to
$INST_DIR/builddir/sstate-cache/	directory, where the shared-state files and the compilation cache are placed

At the end of the installation process the prompt is automatically set to the directory **$BUILD_DIR**

/home/hico/work/openembedded/emcon-rzg1e/native/.

From now you are able to bitbake for the emcon-rzg1e module.

4.1.2 Script for restarting a build process

This script is usefull in the case you have closed the terminal window and reopened a new one. When the installation has already been done, a new build session can only be started by sourcing the build script **yocto_build_rzg1e-native**. This process is similar like sourcing oe-init-build-env. The difference is that some additional settings are done. First navigate to the directory **$INST_DIR** and then call:

source ./meta-emtrion-bsp/meta-emtrion-rzg_1x/scripts/ yocto_build_rzg1e-native

Some environment variables and paths are configured, the build and layer configuration files are copied from ./meta-emtrion-bsp/meta-emtrion-rzg_1x /build-config/emcon-rzg1e and the prompt is set to the directory **/home/hico/work/openembedded/emcon-rzg1e/native/**.

During the setup process you will also be asked if you want to remove the results of the last session (i.e. the contents of **$BUILD_DIR**/tmp). Type “yes” if this is your plan otherwise type “no”.

Note: The content of the directory **$BUILD_DIR** is removed completely but the directories **$INST_DIR**/builddir/downloads/ and **$INST_DIR**/builddir/sstate-cache/ are left unchanged. Additionally, removing everything takes a while since this can be several gigabytes of data.

After this process everything is ready for bitbaking.

Please also note that the configuration files bblayers.conf and local.conf are overwritten by the default files located in /home/hico/work/openembedded/meta-emtrion-bsp/meta-emtrion-rzg_1x /build-config/emcon-rzg1e. So be careful if you have done some modifications there.

4.2 Creating an image

After a successful installation you can start building recipes and images for the emcon_rzg1e module.

The build process is done in two steps. The first one creates some basic libraries, the tool chain, the kernel and prepares and populates a small rootfs for updating the software. To start the first build process call

bitbake core-image-purs

The second step creates the rootfs with OpenGL support, Qt5.2.1 support and adds a Qt demo application. To start this build process call

bitbake emtrion-devkit-image-native

These steps together take several hours for the first time and are strongly dependent on the performance of the host.

In addition, if you want to create an SDK for the root file system and a root file system with source information, enter

bitbake emtrion-devkit-image-native-sdk

4.3 Output files

During the build process lots of objects and images are created. However, the most relevant images are installed into

$BUILD_DIR/tmp/deploy/images/emcon-rzg1e

respectively

$BUILD_DIR /tmp/deploy/sdk.

The exact names of the images are listed below. Note: Some of them are symbolic links.

Images	description
u-boot.bin (*)	U-Boot
zImageemcon-rzg1e (*)	kernel
zImage-emtrion-emcon_rzg1e.dtb (*)	Device tree
emtrion-devkit-image-native.tar.bz2 (*)	RootFS
emtrion-devkit-image-native-sdk.tar.bz2 (*)	Rootfs for sdk
core-image-x11-hideyoshi.tar.bz2 (*)	RootFS
poky-eglibc-i686-emtrion-devkit-image-native-sdk-cortexa7hf-vfp-neon-toolchain-1.6.3.sh	SDK installer

(*) means a symbolic link

4.3.1 Root File System

As shown in the list above, the output of the root file system is a bz2 archive. You can decompress it by the tar command. For test purpose we recommend to decompress the archive to the **$NFS_SHARE**. Navigate to the directory **$BUILD_DIR** and call

**sudo tar xf tmp/deploy/images/ emcon-rzg1e /name_of_rootfs_archive –C **
/home/hico/nfs/root/rootfs

Don't forget "sudo" otherwise the kernel won't be able to modify the files during start up of the system.

4.3.2 boot directory

The directory structure of the root file system includes a directory boot. In addition to the kernel image, device tree, restore script and restore root file system and U-Boot binary a file **uboot_script.txt** is located there.

This file implements some U-Boot command sequences. You can use it for the purpose of updating and booting via network, like kernel, U-Boot and RootFS.

However, the environment of the U-Boot has to be set up before. This is discussed in detail in the chapter 5 of the Bootloader.

4.4 Further readings on Yocto

YP documentations: https://www.yoctoproject.org/documentation/archived

OpenEmbedded: http://www.openembedded.org/wiki/OpenEmbedded-Core

YP-Repositories: https://git.yoctoproject.org/

5 U-Boot Bootloader

The basic task of U-Boot is to load the operating system from bulk memory into RAM and then start the kernel. You can also use it to initiate an update of the kernel, the RootFS and of U-Boot itself. Furthermore you can configure directly from the medium the operating system is to be booted from, for example eMMC, NFS or a serial terminal.

5.1 Basic U-Boot operation

To work with U-Boot, first use a terminal program like picocom to connect to the serial line of the board. As soon as the U-Boot prompt appears in the terminal, U-Boot is ready to receive commands. The general U-Boot documentation can be found here: http://www.denx.de/wiki/U-Boot/Documentation

U-Boot has a set of environment variables which are used to store information needed for booting the operating system. Variables can contain information such as IP addresses, but they can also contain a whole script of actions to perform sequentially. The following commands explain the basic handling of environment variables:

U-Boot command	Explanation
printenv [variable]	This shows the value of the specified variable. If no variable is specified, the whole environment is shown.
setenv [variable] [value]	Set a variable to a specific value. If no value is specified, the variable gets deleted.
saveenv	Make your changes permanent, so they remain after power off or reboot.

5.2 Using U-Boot to change boot device or update parts of the system

This chapter describes how U-Boot has to be setup for updating and booting.

The variable serverip has to be set to the IP-address of the VM. You can get the IP-address by entering **ifconfig** in the terminal of your VM.

Take the IP-address of the corresponding network adapter and assign it to the variable serverip in the U-Boot console. The format of [IP-address] is dot decimal notation.

```
U-Boot # setenv serverip [IP-address]
```

5.2.1 Boot setup and updating the root file system

5.2.1.1 Update

Updating the kernel or the root file system is done by using the nfs export of VM.

Updates of the boot loader are performed using a TFTP-server, which has to run in the VM. It is already set up to run, when you boot it for the first time. The root folder of the TFTP-Server is **/tftpboot**. Every file which is needed in the update process must be copied into this folder.

File names:

The files needed in the update process need to have specific file names. These file names are specified in the following environment variables of U-Boot:

Variable name	Description
image.linux	Filename of the linux kernel image
image.uboot	Filename of the U-Boot image

Updating linux kernel and root file system (uses NFS)

To update both the kernel and the root file system you must copy the image file to **$NFS_ROOTFS**/images/ in your VM. This file can be found in the directory **$DOWNLOAD_DIR** or has been created during the bitbaking process. The name of this image is "emtrion-devkit-image-native-emcon-rzg1e.tar.bz2". Now you can use following commands in U-Boot prompt.

```
U-Boot # setenv serverip        [ip-address of emVM]
U-Boot # setenv nfsroot         [path to the RootFS in emVM]
U-Boot # setenv ip-method       [dhcp or static]
U-Boor # setenv ipaddr [ip-address for device, only needed for static ip]
U-Boot # setenv netmask [netmask for device, only needed for static ip]
U-Boot # run restore_sys
```

This starts the update process. Please be patient as the process of fetching the root file system image via network and decompressing it to the flash storage can take a few minutes.

Updating the linux kernel (uses NFS)

To update the kernel you must copy the kernel image to **$NFS_ROOTFS**/boot/ in your VM. The image is called "zImage" (if the concrete file name differs, add a soft link with this name or rename it). Now you can use following commands in U-Boot prompt.

```
U-Boot # setenv serverip        [ip-address of emVM]
U-Boot # setenv nfsroot         [path to the RootFS in emVM]
U-Boot # setenv ip-method       [dhcp or static]
U-Boor # setenv ipaddr [ip-address for device, only needed for static ip]
U-Boot # setenv netmask [netmask for device, only needed for static ip]
U-Boot # run update_kernel
```

Be careful, updating the kernel without compiling the root file system against this kernel or kernel headers might lead to some problems when i.e. loading modules. The safest ways is to bitbake and update the whole root file system.

Updating of U-Boot bootloader and bootstrapper (uses TFTP)

Attention: If the board is turned off while updating the bootloader and bootstrapper or another error occurs, the board will be rendered unusable to you. Please only update the bootloader if you are explicitly instructed to do so by emtrion.

Copy the bootloader image to the TFTP folder in VM. Then you can start the update process with the following command in U-Boot prompt:

```
U-Boot # setenv serverip        [ip-address of emVM]
U-Boot # setenv ip-method       [dhcp or static]
U-Boor # setenv ipaddr [ip-address for device, only needed for static ip]
U-Boot # setenv netmask [netmask for device, only needed for static ip]
U-Boot # run update_uboot
```

5.2.1.2 Booting

The default boot device in U-Boot is determined by the variable “bootcmd”. If you want to set up one of the following boot options as a default you have to set “bootcmd” to the command mentioned below.

Boot from on-board flash

This is the default boot option configured when you receive the developer kit from emtrion. To start it manually simply use this command:

```
U-Boot # run flash_boot
```

Boot via network using a NFS share

First you have to export the RootFS in emVM using the command “em_nfs -e”. Then you have to perform the following commands in U-Boot:

```
U-Boot # setenv serverip        [ip-address of emVM]
U-Boot # setenv nfsroot         [path to the RootFS in emVM]
U-Boot # setenv ip-method       [dhcp or static]
U-Boor # setenv ipaddr [ip-address for device, only needed for static ip]
U-Boot # setenv netmask [netmask for device, only needed for static ip]
U-Boot # saveenv
U-Boot # run net_boot
```

Now the board should boot via network using the NFS share in VM.

6 SDK

In order to develop applications outside the directory of **$BUILD_DIR** you need to set up your host development system. For this purpose the YP offers several installation methods.

One of the methods creating a SDK is using the build system as described in the chapter 4.2.

The result is a SDK installer containing the toolchain and the sysroot which includes and matches the target root file system. The installer is stored below the directory of **$BUILD_DIR** in

$BUILD_DIR/tmp/deploy/sdk/

6.1 Installing the SDK

Performing the SDK installer, you are asked for the installation directory. You can either accept the suggested one or a created by yourself. From inside the directory of **$BUILD_DIR** start the installer as follows.

source ./tmp/deploy/sdk/ poky-eglibc-i686-emtrion-devkit-image-native-sdk-cortexa7hf-vfp-neon-toolchain-1.6.3.sh

6.1.1 Setting up the SDK environment

Before you can start developing apps you have to setup the environment. For that purpose a script is installed during the installation process of the SDK. The script is stored in the SDK's directory of **$SDK_DIR**.

Performing the setup procedure, the script has to be sourced as follows.

source $SDK_DIR /environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi

The environment is only valid in the context of the terminal where this script has been called.

7 How to use QtCreator with the developer kit

QtCreator is an integrated development environment created by the Qt Project. It is preinstalled in the virtual machine and also preconfigured to work with our target device.

Before starting QtCreator the environment has to be set up. Open a terminal window and call

source $SDK_DIR/environment-setup-cortexa7hf-vfp-neon-poky-linux-gnueabi

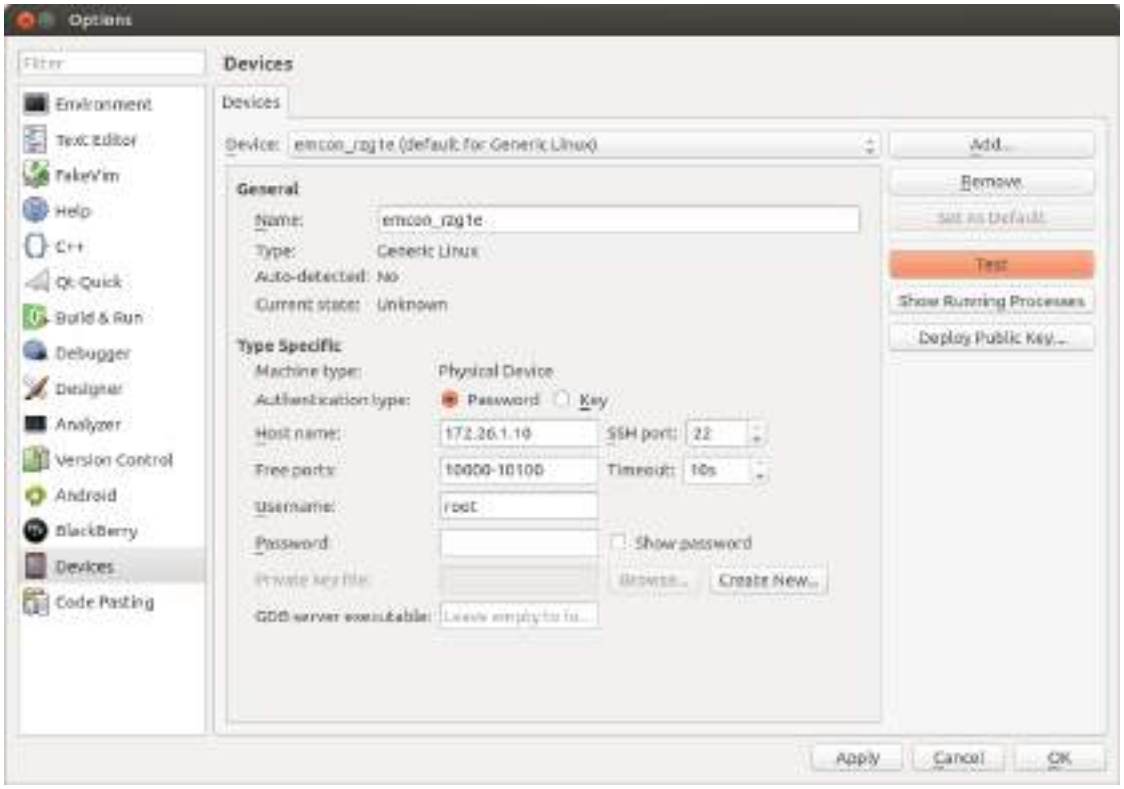
This configures some environment variables so that QtCreator is able to find the cross compiler, the header files and the library files. Afterwards call

qtcreator &

in the same terminal window.

7.1 Device setup

For general information about QtCreator you can check the link "**Get Started Now**". Before doing this, we **setup the connection to the device**. Please open "**Tools->Options**". In this options dialog, select "**Devices**" on the left. The device configuration for our developer kit should be selected. Set the correct IP address of the device.



Then you can use the button labeled "**test**". A successful test looks like this:

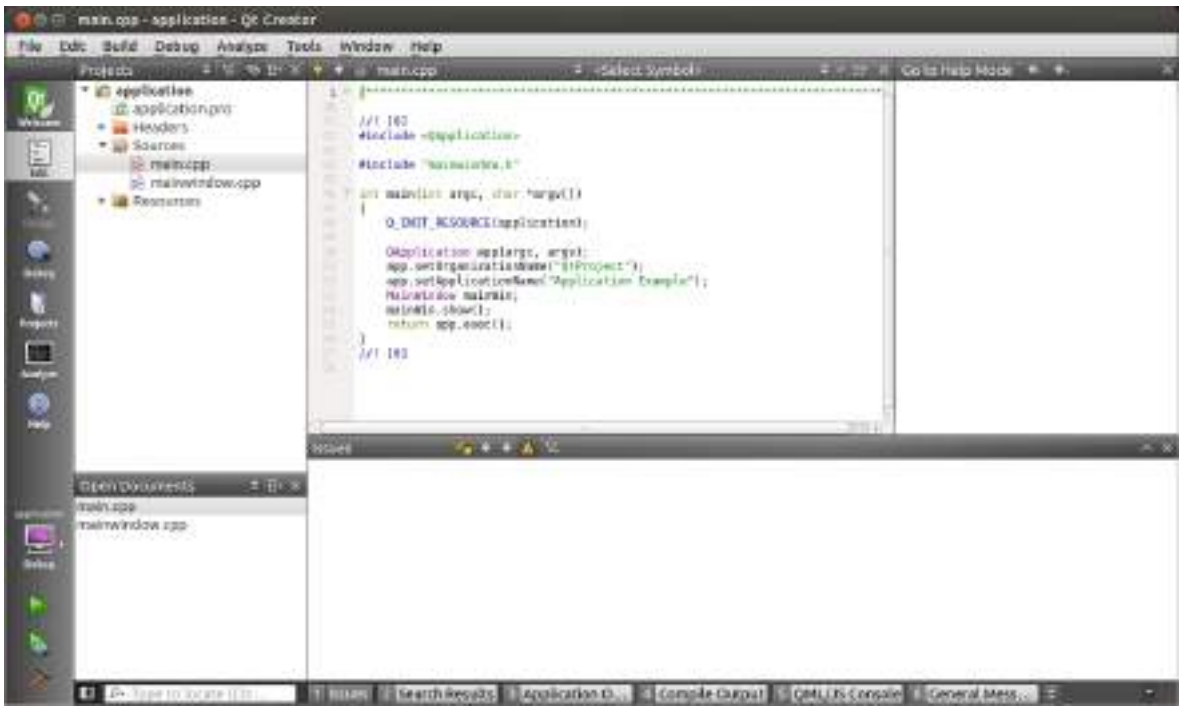
If this does not work, you have to check your network setup.

7.2 Build & Run an Example

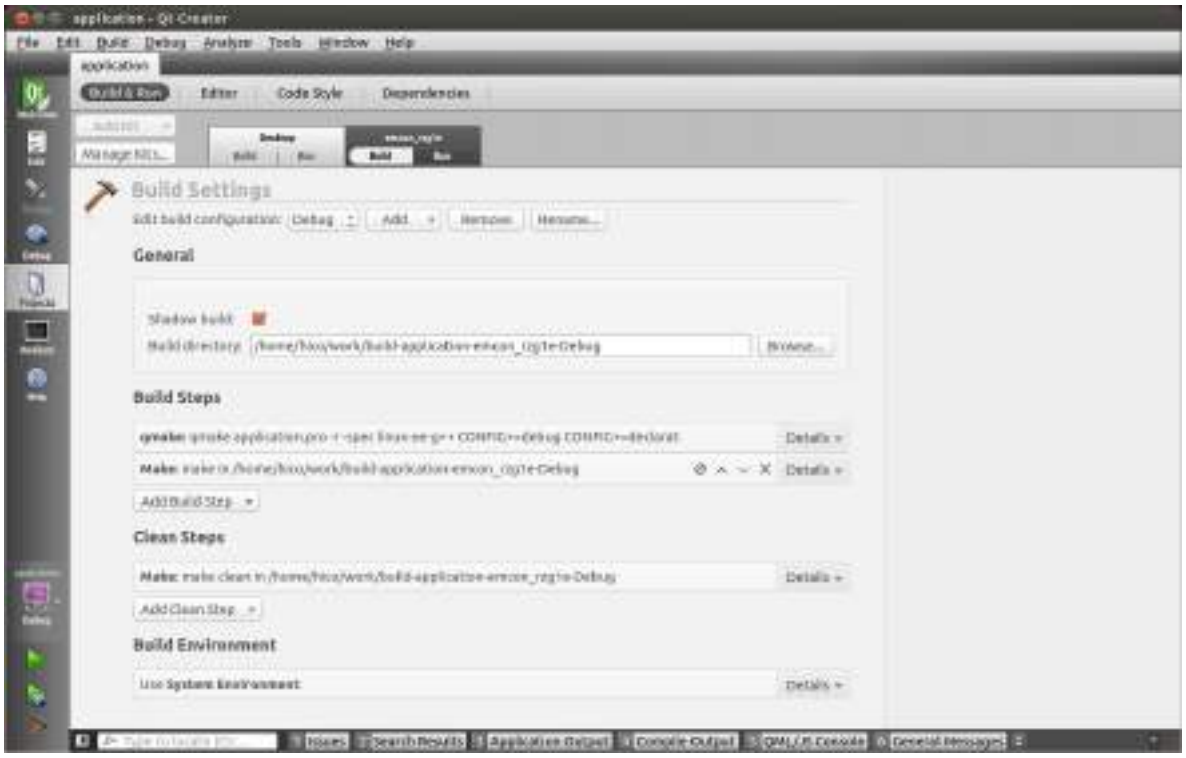
When the connection to the developer kit is successfully established, you can build and run one of the Qt examples on it. First stop the demo from running on the device. Go again to "**Tools->Options**" and select "**Device**" on the left. Now run "**Show running processes**" on the right. In this dialog select the process named "**QtDemo**" and click "**Kill Process**".

Close the windows to get back to the main view. Click on "**Welcome"** on the left bar and select "**Examples**" in the main view. Make sure that the search path is set to "Qt 5.2.1 in PATH (qt5)". Select the first example called "**Application Example**". Choose a working directory and hit the button "Copy Project and Open".

You can now build this example for the host machine or the device. The "**Kit**" and configuration you are building is selected in the left bar with the computer-shaped symbol. After you made your selection you can either "**Run**", "**Debug**" or only "**Build**" the application by pressing the corresponding buttons on the left bar of Qt Creator.

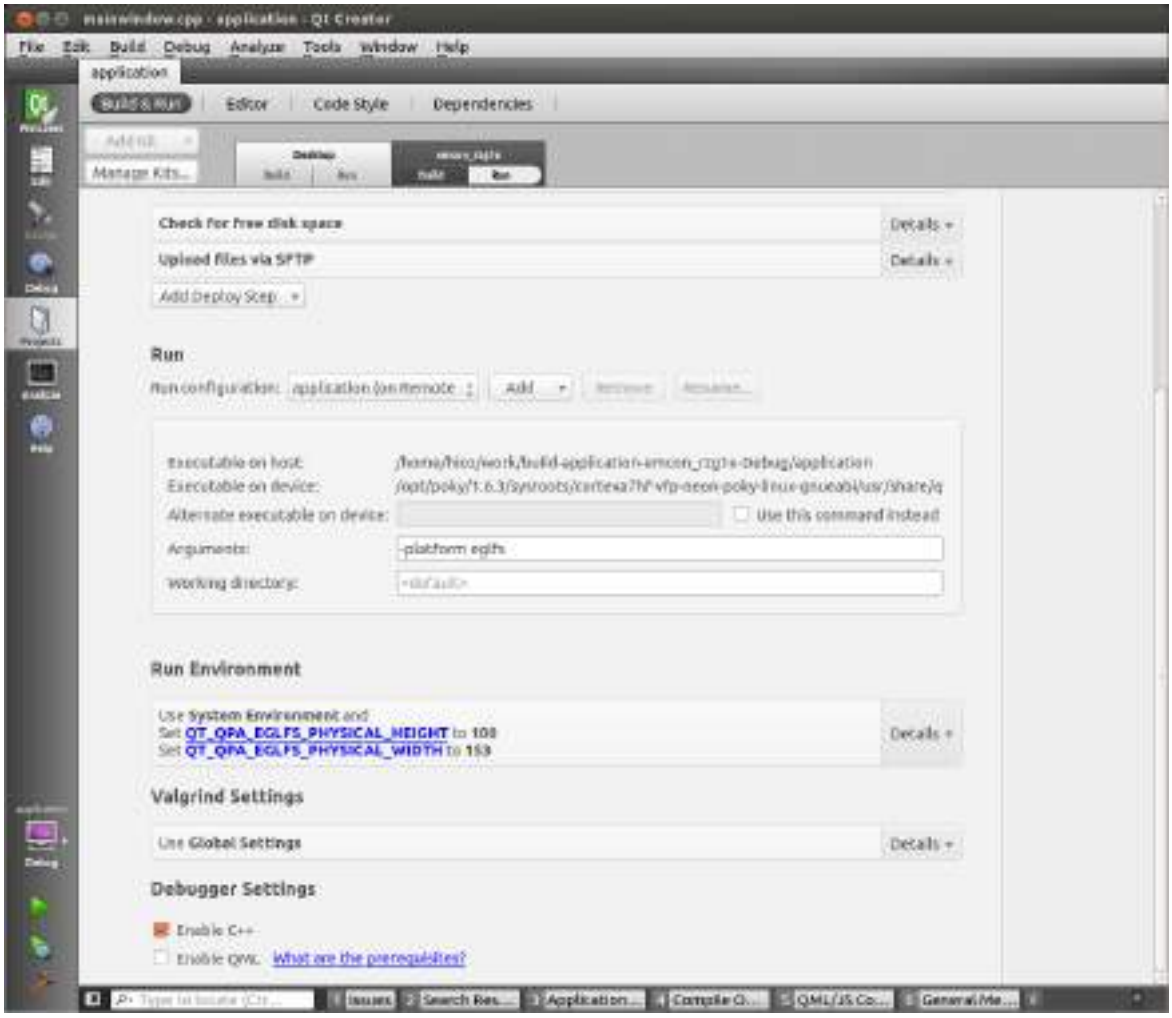


It is possible to compile the application fort he host and to let it run on the host. It is also possible to compile the application for the target and to let it run or debug in the target. To configure the target select the “Projects” window on the side bar of QtCreator. Then select the “application” button for selecting the device and the run mode. Choose the kit and the build settings.



For running an application a platform has to be set. Otherwise a graphical output won’t be possible. The run arguments has to be set to “-platform eglfs”.

The “Application Output” tab will show up some harmless warnings. All of them can be ignored here.



7.2.1 Further documentation about input device configuration

For more detailed information's about input device configuration for Qt5 please look at the official Qt5 documentation:

http://qt-project.org/doc/qt-5/embedded-linux.html

Here you find detailed information about how to configure mouse, keyboard and touch screen together with the respective Qt QPA plugins.

8 Further Information

8.1 Online resources

Further information can be found on the emtrion support pages.

www.support.emtrion.de

8.2 We support you

emtrion offers different kinds of services, among them Support, Training and Engineering. Contact us at sales@emtrion.com if you need information or technical support.