

Debian-based Linux BSP

Quick Start Guide

Rev08 / 09.05.2016

© Copyright 2014-2015 **emtrion GmbH**

All rights reserved. This documentation may not be photocopied or recorded on any electronic media without written approval. The information contained in this documentation is subject to change without prior notice. We assume no liability for erroneous information or its consequences. Trademarks used from other companies refer exclusively to the products of those companies.

Revision: **08 / 09.05.2016**

Rev	Date/Signature	Changes
1	18.03.14/bi	--
2	30.07.14/bi	Added support for DIMM-AM335x
3	21.11.14/bi	Added support for SBC-SAMA5D36
4	21.11.14/bi	Update chapter "Debian build system"
5	02.12.14/bi	Update Appendix A "QtCreator setup"
6	16.04.15/bi	Add section about configuring capacitive touch screens for Qt
7	16.06.15/ft	Updated to include emCON-MX6
8	09.05.16/tue	Remove outdated sections Add Chapter 9 for BSP and Kernel Detail-Information

1 Table of contents

Revision: o8 / 09.05.2016	2
2 Introduction	4
3 Preconfigured Linux virtual machine	5
4 Using Debian	7
5 Device Start Up	8
5.1 Device Network Setup	9
6 How to use QtCreator with the developer kit	10
6.1 Device setup	10
6.2 Build & Run an Example	12
6.3 Input device support	13
6.3.1 Resistive touch (tslib based – dimm-based developer kit standard)	13
6.3.2 Capacitive touch (Standard on emcon based Developer-Kits)	13
6.3.3 Further documentation about input device configuration	14
7 Debian build system	15
7.1 Build the root file system	15
7.2 Further support	15
8 Special features of the Debian for our developer kit	15
9 Additional Information about the Root File System and the Kernel-Image	16
10 Kernel-Update	16
10.1 Online Update	16
10.2 Local/Offline Update	16
11 How to use a Network File System (NFS) as Root File System	16
12 Bootloader U-Boot	18
12.1 Basic U-Boot operation	18
12.2 Using U-Boot to change boot device or update parts of the system	18
12.2.1 Update	18
12.2.2 Booting	20
A QtCreator setup	21
B Cross compilation vs. native compilation	21
C Kernel-Package generation	22

2 Introduction

Welcome to emtrion's Debian-based Linux board support package. This short manual will give you a startup with our BSP. It will describe how to use the Virtual Machine Image for your host PC together with the emtrion developer kit hardware. Further it will give you a quick introduction to Debian, also explaining our motivation behind the decision of releasing a Debian-based BSP.

It is assumed that users of emtrion Linux developer kits are already familiar with Linux. General Linux and programming knowledge are out of the scope of this document. emtrion will gladly assist you in acquiring this knowledge. If you are interested in training courses or getting support, please contact the emtrion sales department.

This guide will show you how to do the start up of the developer kit and our virtual machine. It shows you how to setup a network connection and deploy a Qt5 Example from QtCreator on the developer kit.

The examples in this manual are demonstrated on specific hardware but if not mentioned otherwise they all work on all supported emtrion devices.

3 Preconfigured Linux virtual machine

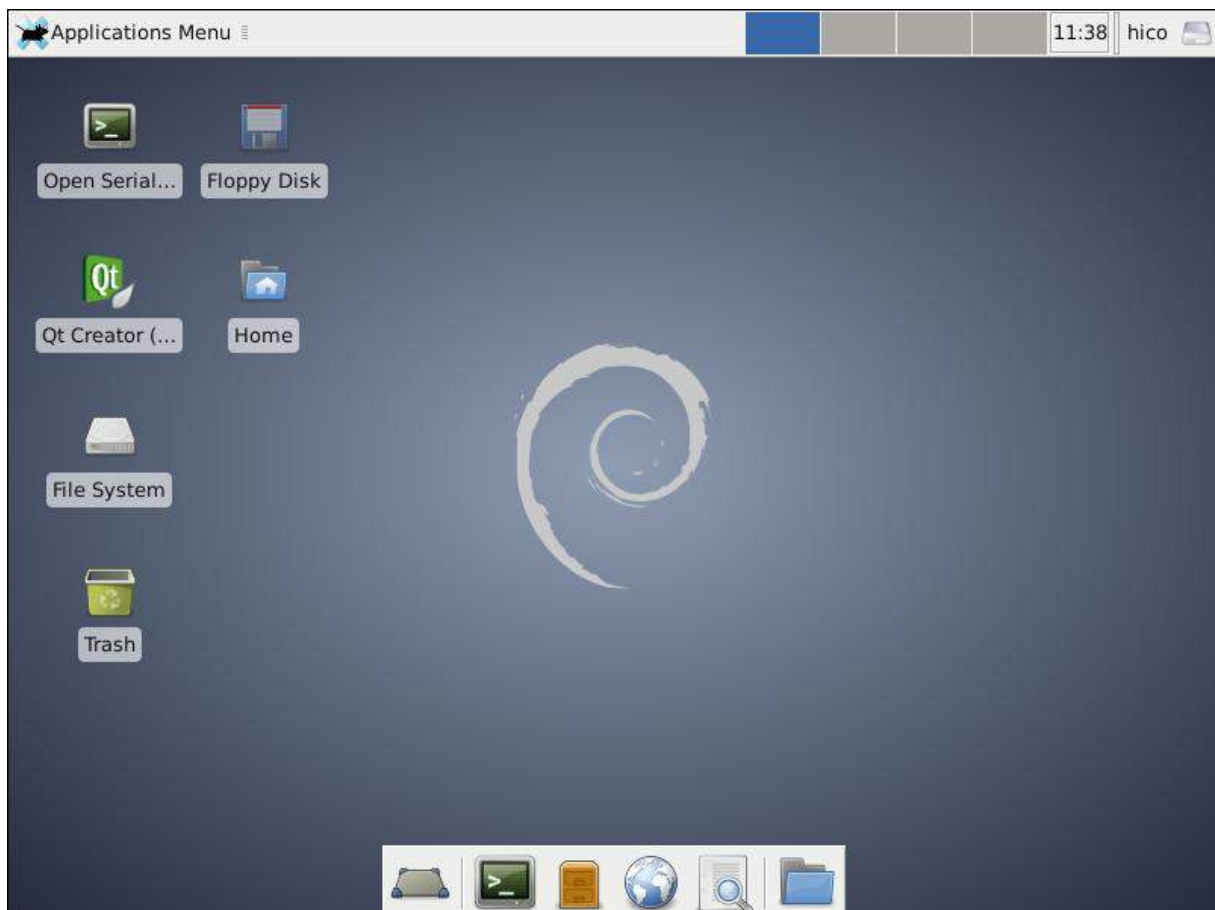
emtrion delivers a DVD accompanying the Debian Linux developer kits. This DVD contains a VMware virtual machine running x86 Debian Linux and the VMware player used to start the virtual machine. For general information about VMware Player please read following guide:

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



1: Desktop of the virtual machine

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

- The system is a Debian Jessie installation.
- 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, `microcom` 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

4 Using Debian

Since Debian is a long established Linux distribution you find good and detailed documentation on the internet. Debian is used both in the virtual machine for the Host PC as well as on the developer kit. This increases the ease of use because administrative tasks, tools etc. work in the exact same manner.

A good starting point to learn about Debian is the Debian wiki:

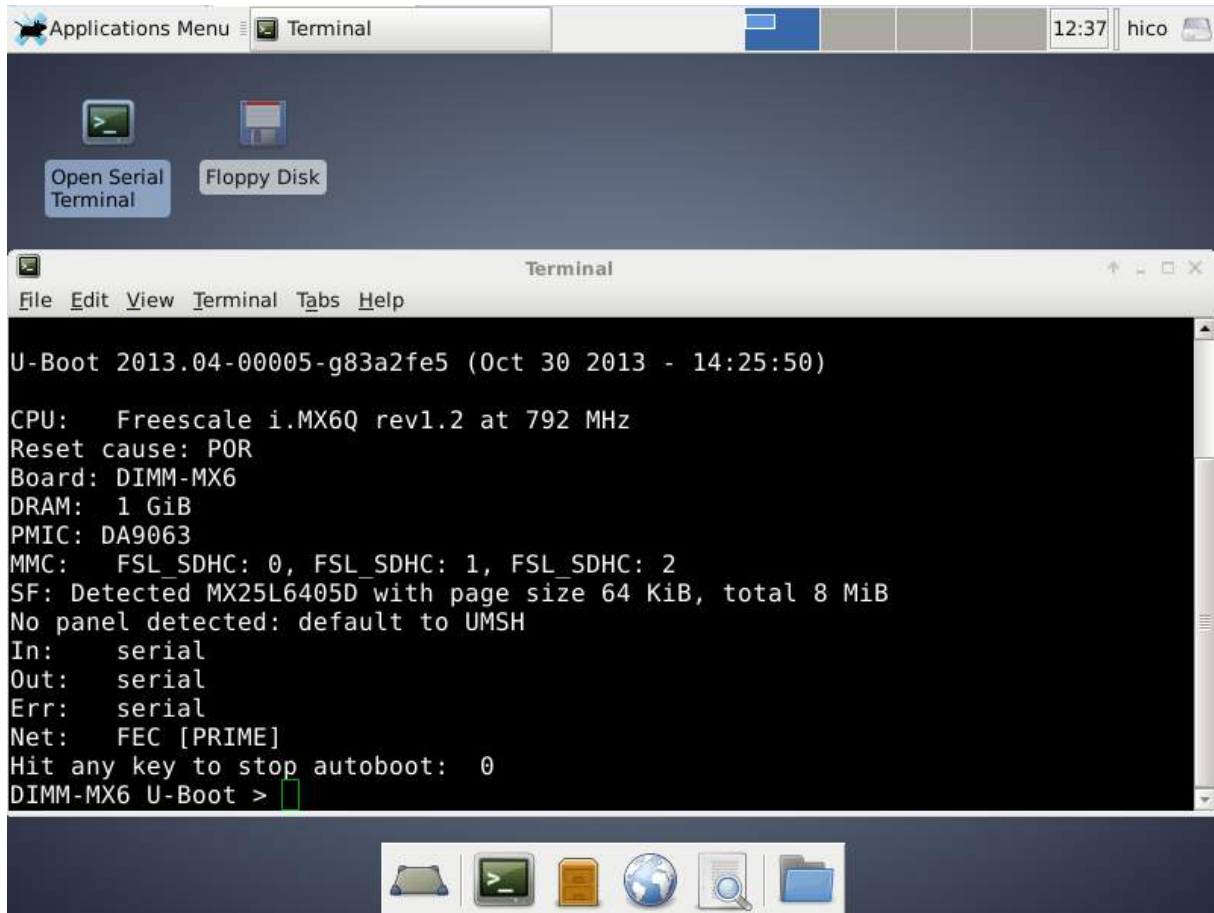
<https://wiki.debian.org>

There also exists an online book about Debian as extensive documentation:

<http://debian-handbook.info/>

5 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.



2: 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:

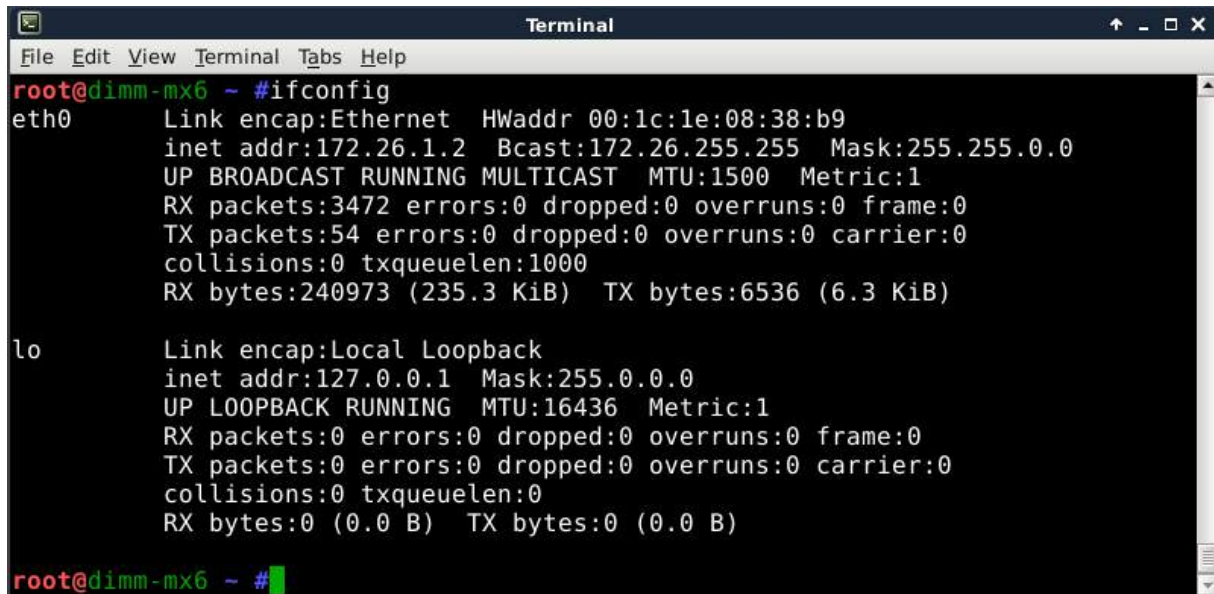
- **user: root**
- **password: hico**

5.1 Device Network Setup

If your network offers a DHCP server the developer kit can be triggered to use it by using following command in a terminal:

```
root@emtrion-mx6 ~ #dhclient eth0
```

You can check the result with the command **ifconfig**.



```
Terminal
File Edit View Terminal Tabs Help
root@dimm-mx6 ~ #ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1c:1e:08:38:b9
          inet addr:172.26.1.2  Bcast:172.26.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3472 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:240973 (235.3 KiB)  TX bytes:6536 (6.3 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  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@dimm-mx6 ~ #
```

3: ifconfig output

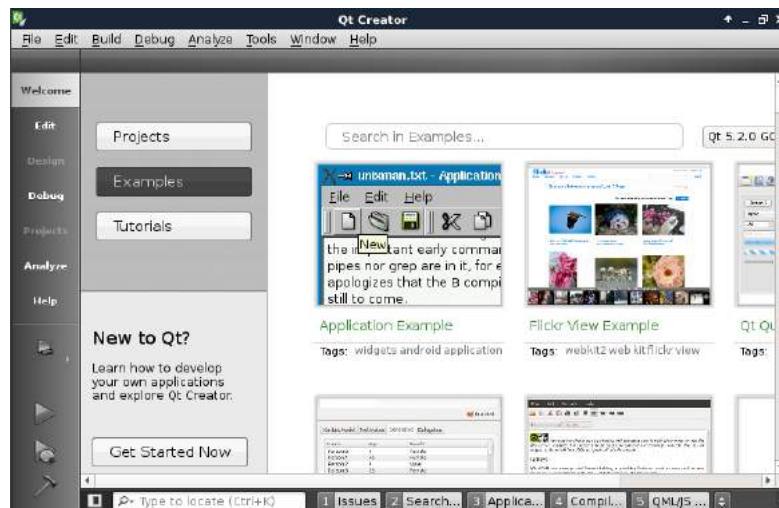
If the setup is not correct you have to do it manually. Please read the Debian documentation referenced in the chapter [Using Debian](#).

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

6 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.

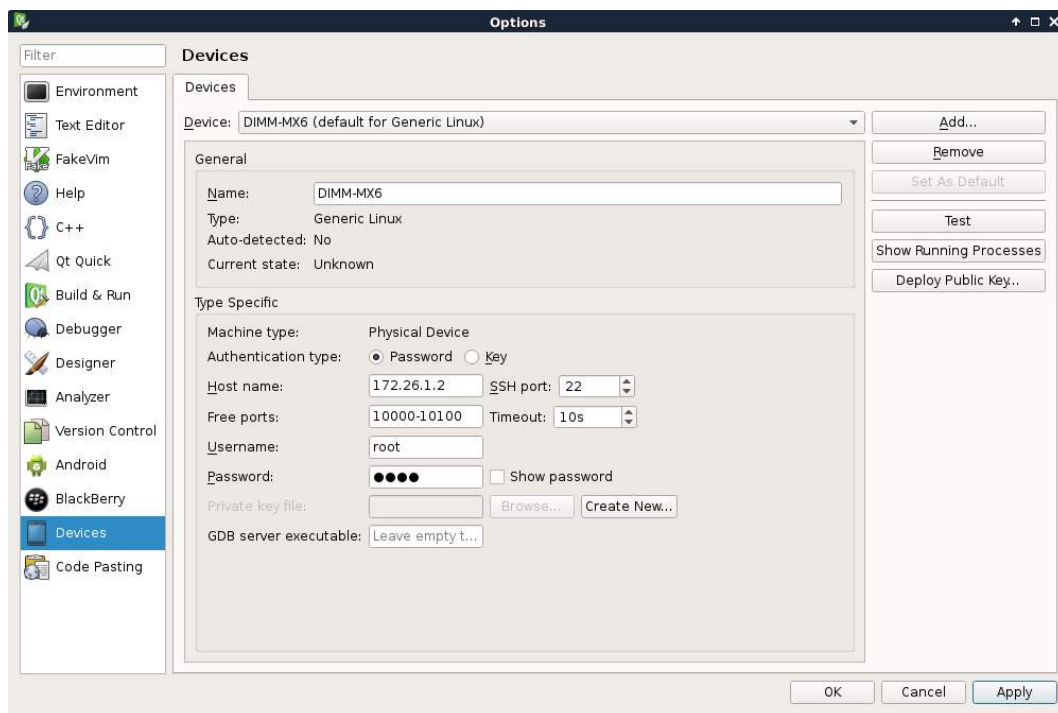
Start it via the shortcut labeled "Qt Creator" on the desktop.



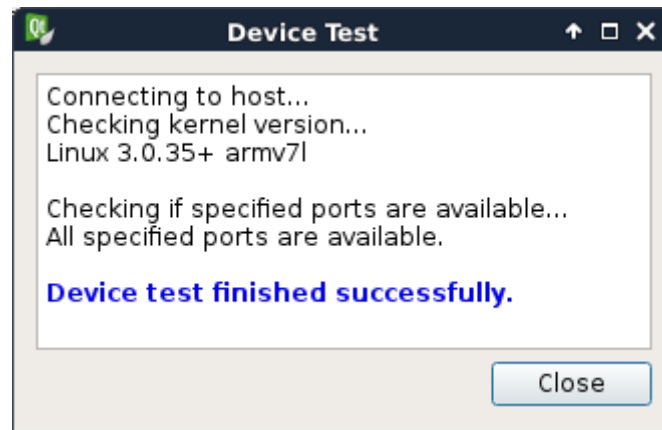
4: QtCreator start up screen

6.1 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. Check following steps:

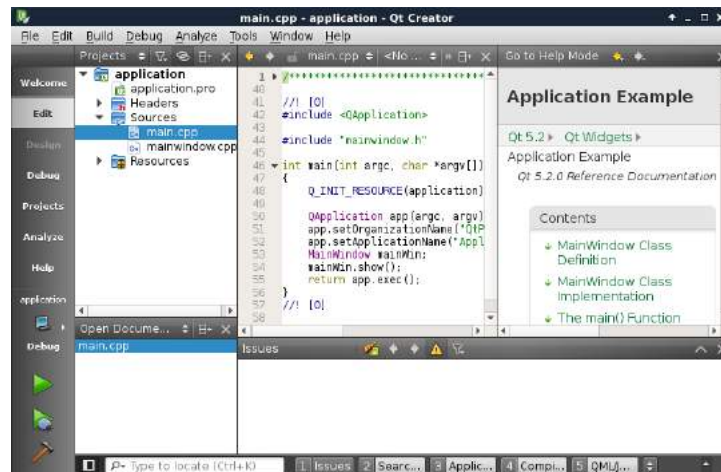
- VMware setup of the virtual network device. Please refer to the chapter about our [Preconfigured Linux virtual machine](#)
- Linux networking setup of the Debian Linux inside the virtual machine. Please refer to the chapter [Using Debian](#)
- The configuration of the device. Please refer to chapter [Device Network Setup](#)

6.2 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. Select the first example called **"Application Example"**.

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.



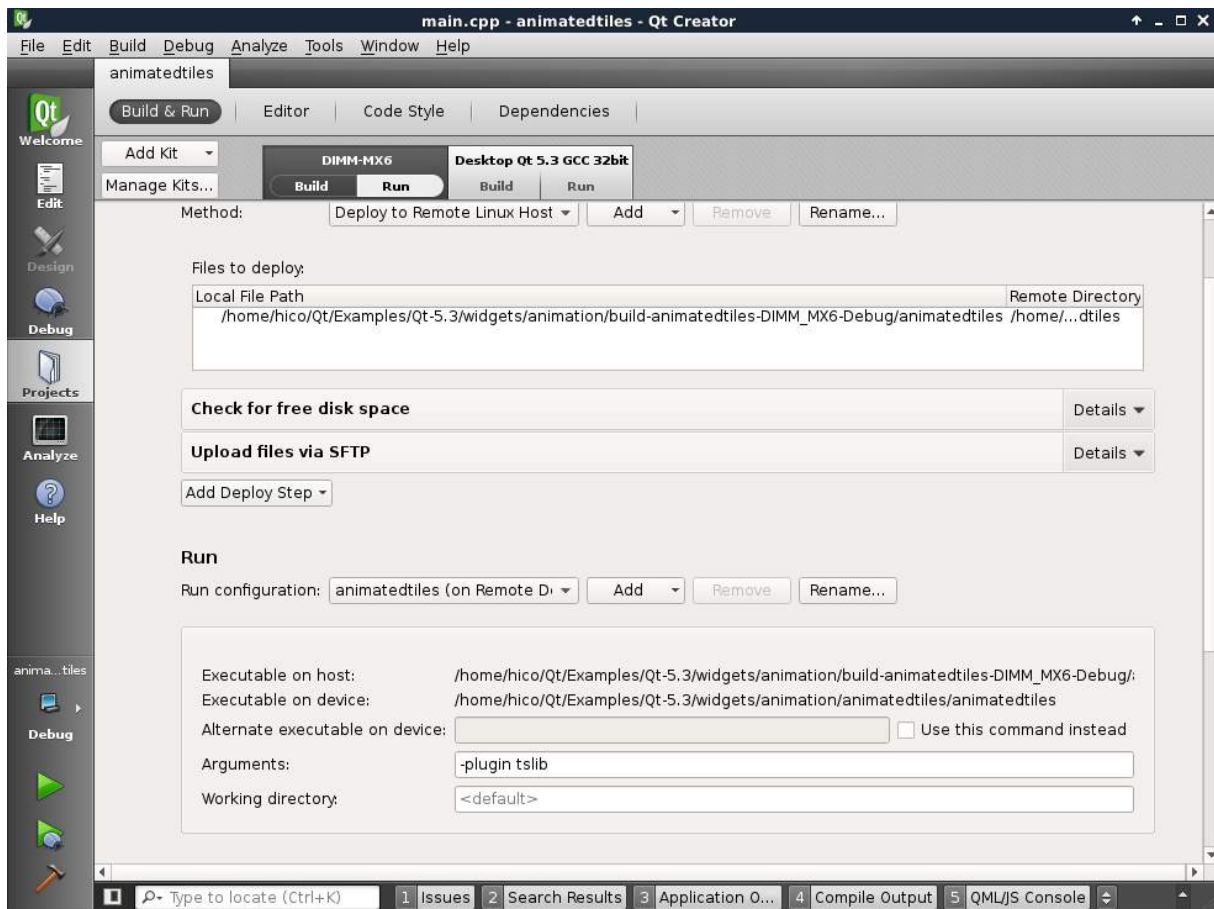
SBC-SAMA5D36: Please note that there is only limited support for the examples using this board because it does not have a GPU. Basically QtQuick 2.0 based examples are not supported but also some others are configured with OpenGL support. Those may work after reconfiguration. In general QtWidget-based GUIs should work.

6.3 Input device support

6.3.1 Resistive touch (tslib based – dimm-based developer kit standard)

If the touch screen is not working when executing the example, it may be necessary to pass the parameter **"-plugin tslib"** when starting the demo on the device.

You can add run time arguments by selecting the **"Projects"** windows on the side bar of QtCreator. Then select the **"Run"** tab for the device as shown on the screenshot below.



6.3.2 Capacitive touch (Standard on emcon based Developer-Kits)

DIMM-MX6/emCON-MX6:

Depending on the kernel version the device enumeration differs: When using kernel version **3.0.35** the device file of the capacitive touch screen is **/dev/input/event1**. But with kernel version **3.10.53** it is **/dev/input/event0**.

To pass this information to Qt you have to set the value of the environment variable **QT_QPA_EVDEV_TOUCHSCREEN_PARAMETERS** to the device file. You can do this in the **"Run"** tab for the device in Qt Creator under the Section **"Run Environment"**.

6.3.3 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.

7 Debian build system

7.1 Build the root file system

The Debian system running on our device was created using the build system ELBE (<http://elbe-rfs.org/index.html>)

If you want to use ELBE to recreate our image you can follow the ELBE documentation which provided online on the website mentioned above.

emtrion is using the current development beta version ELBE where we added some patches our won. The tweaked ELBE binary is already installed in the virtual machine and is automatically updated through our package repository. The folder **~/Downloads** contains the **xml** file which was used to create the delivered root file system. If you want to rebuild it yourself use the following commands in the virtual machine:

```
hico@debian:~$ elbe init --directory [build-dir]
                        ~/Downloads/[device-specific].xml
hico@debian:~$ cd [build-dir]
hico@debian:~$ make
```

7.2 Further support

We also offer to support you if you want to create custom Debian packages and a custom image for your device with the build system. Additionally we offer to do maintenance work by checking the software in your image for security/stability updates and notifying you when important updates should be applied to your image.

8 Special features of the Debian for our developer kit

To be able to use Debian on our boards we setup our own package repository, which contains Debian packages specific to our boards. This includes:

- Modified Linux kernel package
- Special configurations
- Packages for board specific libraries, e.g. HW accelerators
- Modified packages for better performance (modified Qt5 to make best use of the hardware accelerators)

When we do an update of the software contained in these packages, we publish the updated packages via our repository. This means that you can update the software on our developer kit quite easily by executing "**apt-get update && apt-get upgrade**" from within a Linux shell. Of course the developer kit must be connected to the internet.

Attention: The use of our server is only allowed together with our developer kits. The servers are not meant to be used from final devices in the field. Emtrion reserves its right to change the access permissions to the server as well as the support for certain platforms at any time.

9 Additional Information about the Root File System and the Kernel-Image

On ancient BSP-Versions from emtrion, mostly based on emDist (out-of-date) or Yocto, the U-Boot Kernel Image format was used, formerly known as "ulmage". This has changed for the most Debian BSP-Releases and is now a position-independent "zImage".

With the change to the new zImage Kernel, the default boot-behavior changed. In replacement of the old "boardfile" architecture, we moved to the devicetree. This is a technique which should (in future) enable you to run a single Kernel-Image on different boards, as it is standard on PC's.

Basically the devicetree is an additional external file with the ".dtb" extension. This file has to be loaded in place additionally by the bootloader.

On our official debian releases this is handled automatically by uboot_script which is contained in the emtrion Board support package.

10 Kernel-Update

10.1 Online Update

To install a new emtrion Kernel release it is just necessary to run the following commands on the device connected to the internet:

```
root@emtrion-mx6 ~ # apt-get update
root@emtrion-mx6 ~ # apt-get upgrade
```

10.2 Local/Offline Update

If you have downloaded or created a custom Kernel-Package (.deb extension) you can update it by running the following command:

```
dpkg -i linux-image-3.10.53-emtrion 3.10.53-emtrion-6 armhf.deb
```

11 How to use a Network File System (NFS) as Root File System

For development purposes it is convenient to boot via network using NFS. The root file system of the device is then a subdirectory in your development machine. For detailed information about NFS setup in Debian you can read this wiki page <https://wiki.debian.org/NFSserverSetup> .

In our virtual machine there already is an NFS-exported folder: **"/home/hico/devkit-rootfs"**

This folder contains the root file system which our QtCreator configuration uses to cross compile. If you want to know which packages are needed on top on the tarball of the root file system that we release, please read chapter [QtCreator setup](#). The tarball of the root file system is placed under **~/Downloads**.

You can download a current Debian root file system as a tar ball from our support pages (www.support.emtrion.de). This tar ball can be extracted to the exported directory and then you can boot from it. The exact boot loader commands can be found in the chapter [Booting](#).

12 Bootloader U-Boot

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 root file system and of U-Boot itself. Furthermore you can configure directly from the medium the operating system is to be booted from, for example MMC, NFS or a serial terminal.

12.1 Basic U-Boot operation

To work with U-Boot, first use a terminal program 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. To interrupt the automatic boot mechanism press a keyboard key while using the terminal. 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.

12.2 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 exact variable assignment is board specific. DIMM-MX6, emCON-MX6 and DIMM-AM335X can be used in the same way.

The variable serverip has to be set with the IP address of the virtual machine. You can get the IP address by entering `sudo ifconfig` in the terminal of your virtual machine.

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]
```

12.2.1 Update

Updating the kernel or the root file system is done by using a NFS exported in the virtual machine. For further information please look into the chapter [How to use a Network File System \(NFS\) as Root File System](#).

Updates of the boot loader are performed using a TFTP server, which is running in virtual machine. 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
Image.bootstrap	Filename of the Bootstrapper image

Update of the root file system (uses NFS)

Make sure you exported a directory via NFS in your virtual machine. Of course the directory must contain an extracted Root File System of the Debian BSP for the board. Updated Root File System tar balls can be downloaded from our support pages (<http://support.emtrion.de>):

```
U-Boot # setenv serverip      [ip-address of virtual machine]
U-Boot # setenv nfsroot      [path to root file system]
U-Boot # setenv ip-method    [dhcp or static]
U-Boot # 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 rootfs
```

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

Update 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 the virtual machine. Then you can start the update process with the following command in U-Boot prompt:

```
U-Boot # setenv serverip      [ip-address of virtual machine]
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
```

12.2.2 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

Make sure you exported a directory via NFS in your virtual machine. Of course the directory must contain an extracted Root File System of the Debian BSP for the board. Updated Root File System tar balls can be downloaded from our support pages (<http://support.emtrion.de>):

```
U-Boot # setenv serverip      [ip-address of virtual machine]
U-Boot # setenv nfsroot      [path to the root file system]
U-Boot # setenv ip-method    [dhcp or static]
U-Boot # 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 the virtual machine.

A QtCreator setup

QtCreator is configured to cross compile for our developer kit. Therefore we need an installation of a development root file system in our virtual machine. In the state of delivery this root file system is located in the folder `"/home/hico/devkit-rootfs"` which is also exported as NFS share. (see chapter [How to use a Network File System \(NFS\) as Root File System](#))

If you want to **extend** this root file system just boot via NFS and install the necessary packages.

If you deleted this root file system and want to **recreate** it just take our release root file system and extract it into the folder. Then boot from it via NFS. Now execute these commands on the developer kit to install the needed packages:

DIMM-MX6 / emCON-MX6

```
root@emtrion-mx6 ~ # apt-get install libqt.*5-dev qt.*5-dev libqt5webkit5
root@emtrion-mx6 ~ # apt-get install qml-module-qtwebkit
root@emtrion-mx6 ~ # apt-get install libstdc++-4.9-dev
```

DIMM-AM335X

```
root@dimmm-am335x ~ # apt-get install am335x-gpu-driver-dev
root@dimmm-am335x ~ # apt-get install libqt.*5-dev qt.*5-dev libqt5webkit5
root@dimmm-am335x ~ # apt-get install qml-module-qtwebkit libglib2.0-dev
root@dimmm-am335x ~ # apt-get install libstdc++-4.9-dev
```

SBC-SAMA5D36

```
root@dimmm-am335x ~ # apt-get install qtbase5-dev
root@dimmm-am335x ~ # apt-get install libstdc++-4.9-dev
```

B Cross compilation vs. native compilation

QtCreator is setup for cross development. This works very good with Qt because the project cares a lot about being compatible with many platforms and so cross development is a common setup.

Yet this is not the case for every software project. Many software projects are created without minding cross development/building. It can be a very painful task to enable those projects to be cross build. So **emtrion wants to remind you that with our Debian BSP it is possible and perfectly reasonable to compile software natively on our developer kit**. Even if the compilation itself takes a little longer than on the host, it is so much easier than enabling a project to be cross build.

C Kernel-Package generation

Please note: the information in this section is a quick How-To note, to get you a quick point on how to create a valid Kernel package for your system and intended for experienced users.

For the creation of a Debian-Package there is a special make-Target contained in the Linux-Kernel sources for years. This Target is named "deb-pkg". To get a fully compatible Kernel-Package it is necessary to build on a device running the matching system type (armhf for example). However it is also possible to cross-compile a Kernel -Package for quick testing.

To have a valid Package as output the following make call will help you:

```
make KBUILD_DEBARCH=armhf KBUILD_VERSION=3.10.53-emtrion deb-pkg
```