Emtrion Developer Kits for emtrion-MX6

Yocto based BSP manual

Rev001 / 07.04.2017

emtrion GmbH

Revision: **001 / 07.04.2017**

Rev	Date/Signature	Changes
001	MI/07.04.2017	First release

1 Introduction

Emtrion produces and offers various base boards and modules with the imx6 and provides developer kits of them.

There are two base boards and two modules with a total of 4 imx6 variants available. They are listed below.

Base	Module	available with imx6 variant	common Device tree
Avari	emCON-mx6	Quad Dual DualLite Solo	Quad/Dual DualLite/Solo
Tarion	dimm-mx6	Quad Dual DualLite Solo	Quad/Dual DualLite/Solo

The BSP is a linux kernel OpenSource mainline version 4.9.7 adapted by emtrion. The BSP is commonly used for all of the developer kits. The differences between the various variants are considered by a corresponding device tree. Graphic support is OpenSource etnaviv 12.0.

In addition, using OpenSource is attended by some advantages.

- more independently
- maintaining
- changing to a newer release is easier
- more stability

The RootFS has been created with Yocto Openembedded, **morty** 2.2.1.

The recipes used by the meta-layer for emtrion-mx6 are mostly based on recipes of several other meta-layers. These recipes have been modified or extended by emtrion, so Qt5.7.1 with OpenGL is supported.

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

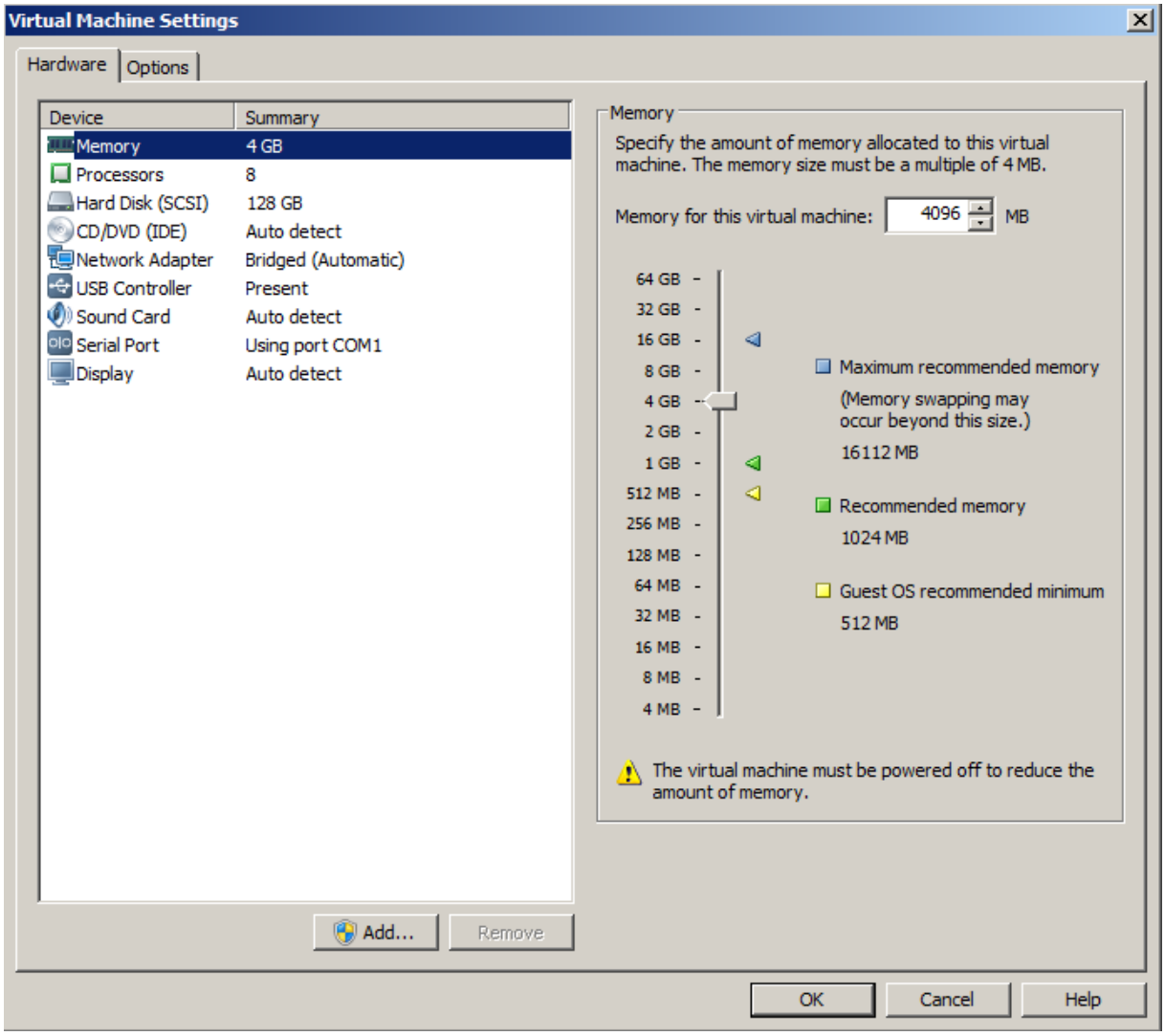
It is assumed that users of emtrion Linux developer kits are already familiar with U-boot, Linux, Yocto and creating and debugging applications with Qt5.7.1 and QtCreator. General Linux and programming knowledge are out of the scope of this document. emtrion is happy to assist you in acquiring this knowledge. 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-mx6 with baseboard Avari or dimm-mx6 with baseboard Tarion
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	Location that is exported by the NFS for the purpose of updating and booting by using NFS
U-Boot	Bootloader, hardware initialization, updating images, starting OS
YP	Yocto Project
INST_DIR	Location where Yocto and the meta-layers are installed
MACHINE	Specifies the target device for which the image is built. For the current target the name is set to **emtrion-mx6**
BUILD_DIR	Machine dependent build directory
BSP	Board Support Package
SDK	Software Development Kit

3 The Linux virtual machine VM

To support the development with emtrion's Yocto-Layer a VMware virtual machine was configured. To obtain a good performance the VM is configured with following properties.



However, the settings are strongly dependent on the PC and have to be adjusted later on your PC.

- Used memory
- Number of processors
- Network Adapter
- Serial Port

3.1 Content

As Linux Distribution **Debian 9** (stretch) is used. The Distribution was setting up by general Yocto Project system requirements described in the chapter "1.3. System Requirements" of the Reference Manual Yocto Project 2.2.1 Release.

http://www.yoctoproject.org/docs/2.2/ref-manual/ref-manual.html

Further components are included

- A preconfigured Qtcreator 4.2.0 for Qt-Development
- gdb-multiarch for debugging

- Serial ports added to the virtual machine appear at /dev/ttySn, USB serial converters at /dev/ttyUSBn
- A NFS server exporting the nfs share **/home/hico/nfs**
- The serial terminal program **picocom** for connecting to the target
- The Yocto-Layer **meta-emtrion-mx6**
- SDK and pre-built images of the Layer **meta-emtrion-mx6** in **~/Downloads**

3.2 Starting the VM

The VM is a compressed ZIP archive. Changing settings and starting the VM is used by the VMware Player or VMware Workstation. Here the link for downloading the VMware Player

https://www.vmware.com/go/downloadplayer

The corresponding manual is behind this link

VMware® Player™ Manual

After decompressing of the VM and importing it by the VMware Player, first check if the settings above are fit to your PC. If not, adjust the settings by reading the corresponding chapters of the specified manual.

Please note, the size of the VM will increase up to 128 GB while you are working with it.

3.2.1 Login account

The login data are specified as follow

username: **hico**

password: **hico**

3.3 Preconfigured variables

Within the VM there are used some predefined locations. In the scope of this document the locations are assigned to specified placeholders. They are listed in the table below.

Placeholder	Assignment	remark
MACHINE	emtrion-mx6	machine name
HOME_DIR	/home/hico	home directory of user hico
NFS_SHARE	<**HOME_DIR**>/nfs	Location exported by the NFS
NFS_ROOTFS	<**NFS_SHARE**>/<**MACHINE**>/root/rootfs	nfs share for booting the root file system by using NFS
NFS_UPDATE	<**NFS_SHARE**/<**MACHINE**>/update	nfs share for updating the root file system by using NFS
INST_DIR	<**HOME_DIR**/openembedded	Location where Yocto and all the meta-layers will be stored to
BUILD_DIR	<**INST_DIR**>/builddir/emtrion/machines/<**MACHINE**>	Location of the build system.
BUILD_DWNL	<**INST_DIR**>/builddir/downloads	Location of the fetched downloads while the build process

BUILD_SSTATE	<**INST_DIR**>/builddir/sstate-cache	Location of the sstate-cache while the build process
HOME_DWNL	<**HOME_DIR**>/Downloads	Location of the pre-built images, SDK
SDK_DIR	/opt/poky/2.2.1	SDK with tools, sources and libraries

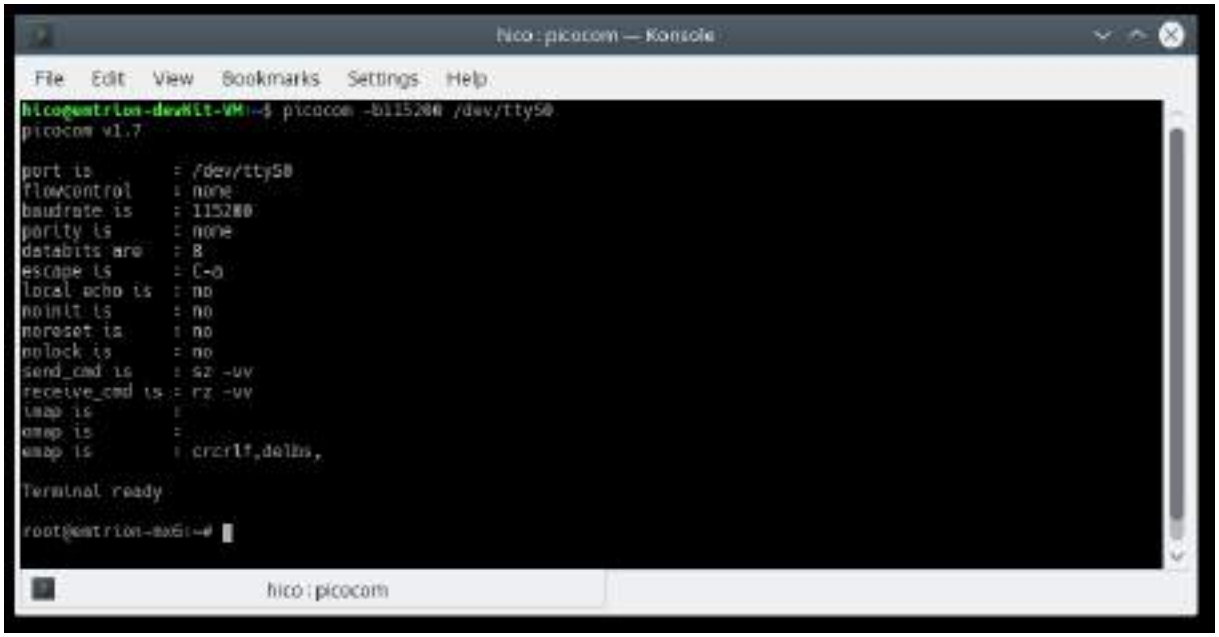
4 Device Start Up

Connect the developer kit to the serial port attached to the virtual machine and your network. Open a console in the VM and open a serial terminal by enter

picocom –b115200 /dev/ttySx.

ttySx has to be replaced with the device assigned to the connected serial port.

In the case of using an USB serial adapter replace it by the corresponding ttyUSBn.



1: Serial terminal showing U-Boot prompt

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

After the developer kit is booted you are prompted for login:

- emtrion-mx6 login: **root**

4.1 Demo

While the system is booting the QtDemo will be performed. To terminate the QtDemo you can enter in the terminal either

root@emtrion-mx6:~# **killall QtDemo**

or

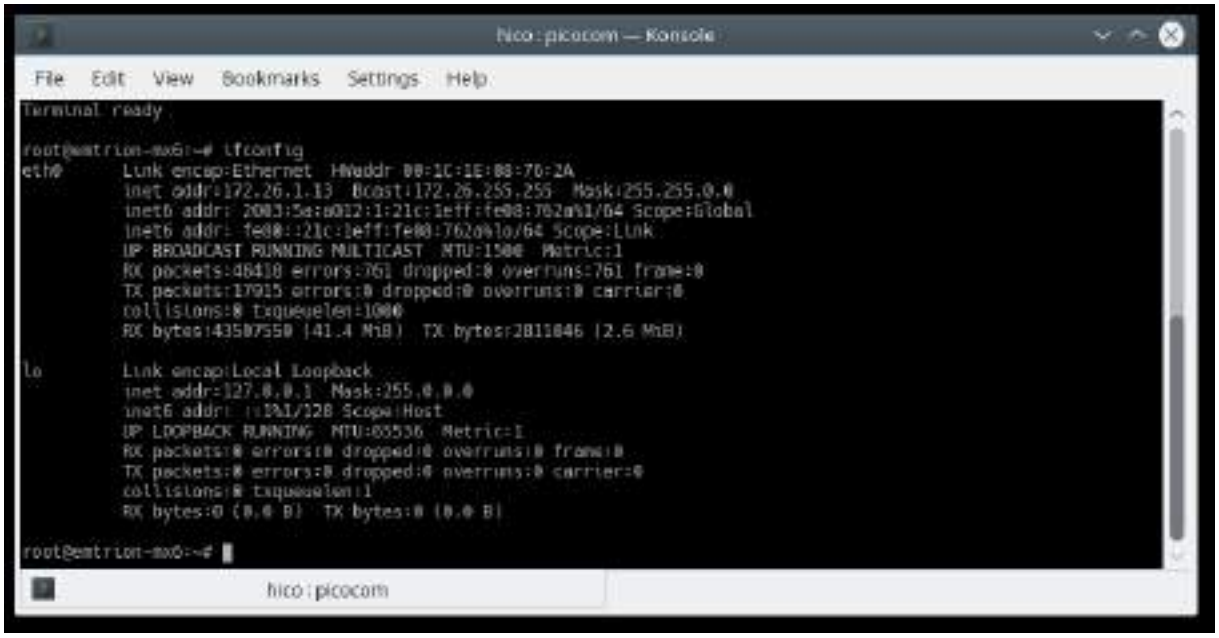
root@emtrion-mx6:~# **/etc/init.d/demo stop**

If you want to prevent performing the QtDemo at boot time, you can comment out it in the corresponding init script **/etc/init.d/demo**.

4.2 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**.



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.

5 Pre-built images and installations

To reduce the size of the VM for delivering, the VM is shipped with pre-build images without including all the outputs of the build process.

The pre-built images and others are located in <**HOME_DOWNL**>

- devkit-emtrion-mx6.tar.gz
- devkit-emtrion-mx6-sdk.tar.gz
- poky-glibc-x86_64-devkit-emtrion-mx6-sdk-cortexa9hf-neon-toolchain-2.2.1.sh
- update-emtrion-mx6.tar.gz

- meta-emtrion-mx6.tar.gz

The images have been tested and can be used for designing applications with QtCreator.

5.1 Installation

Dependent on the focus of the user the appropriate archives have to be installed before.

5.1.1 devkit-emtrion-mx6.tar.gz

This archive is the image without SDK extensions. It contains the root file system as well as the kernel.

Installing of the image you can boot the system by using NFS. From the home directory enter

sudo tar xf <**HOME_DOWNL**>/devkit-emtrion-mx6.tar.gz –C <**NFS_ROOTFS**>

How you can boot it by using NFS is described in the chapter 7 of the Bootloader.

5.1.2 devkit-emtrion-mx6-sdk.tar.gz

This archive is similar to the previous image with SDK extensions and without the QtDemo application.

You can install it as the previous image. However, this is not suitable for normal use.

5.1.3 poky-glibc-x86_64-devkit-emtrion-mx6-sdk-cortexa9hf-neon-toolchain-2.2.1.sh

This file presents the SDK as a self extracted script. The SDK is required for developing applications outside the <**BUILD_DIR**>.

Install the SDK from the home directory by prompting

<**HOME_DOWNL**>/ poky-glibc-x86_64-devkit-emtrion-mx6-sdk-cortexa9hf-neon-toolchain-2.2.1.sh

While performing the script, you will be asked for the installation directory. Let the default **/opt/poky/…** and confirm it.

5.1.4 update-emtrion-mx6.tar.gz

This archive includes files which support the update process of the root file system and kernel at the developer kit.

Install the files of the archive to the nfs share <**NFS_UPDATE**>/boot by prompting

tar xf <**HOME_DOWNL**>/update-emtrion-mx6.tar.gz –C <**NFS_UPDATE**>/boot

How you can update the system on the developer kit is described in the chapter 7 of the Bootloader.

5.1.5 meta-emtrion-mx6.tar.gz

This archive contains the meta-layer for emtrion-mx6. You have to install it if you want to create your own images.

Install the layer by prompting

tar xf <**HOME_DOWNL**>/meta-emtrion-mx6.tar.gz –C <**INST_DIR**>

6 The layer meta-emtrion-mx6

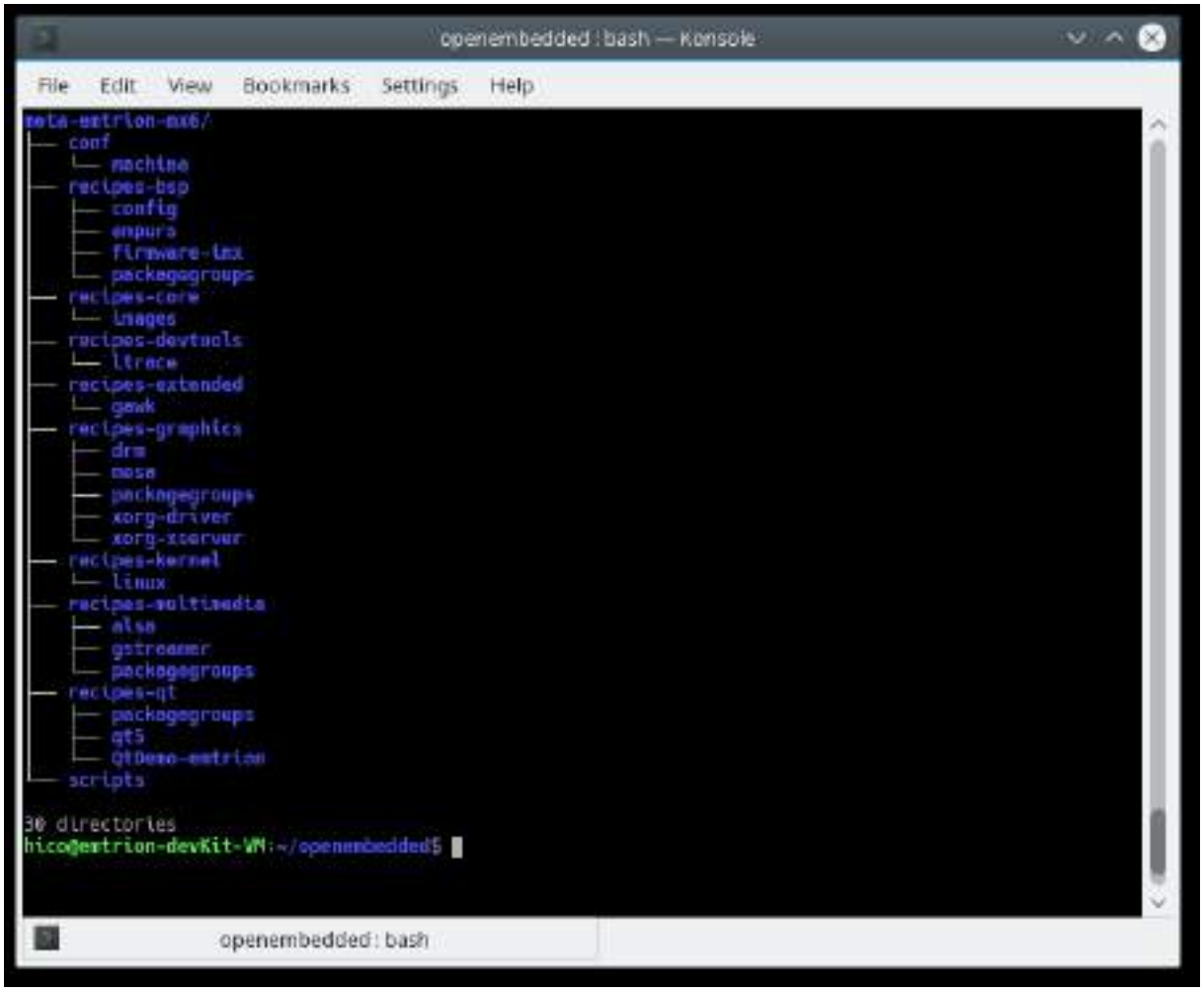
If you plan to work with QtCreator and you do not want to create new root file systems, you can skip this chapter.

6.1 Installation

The installation of the layer described in the previous chapter, will locate the layer in the location <**INST_DIR**>.

6.2 Overview of the recipe structure

The recipe structure provided by the layer is shown below. The various recipes implement and define what packages are included in the image provided by the layer.



Location	Remarks
meta-emtrion-mx6/	layer for emtrion-imx6
├── conf	configuration files
│ └── machine	machine configuration file

├── **recipes-bsp**	
│ ├── **config**	emtrion's update support
│ ├── **empurs**	emtrion's update functionality
│ ├── **firmware-imx**	add firmware for imx-vpu
│ └── **packagegroups**	summarises tools
├── **recipes-core**	images
│ └── **images**	provides the images
├── **recipes-devtools**	development
│ └── **ltrace**	append recipe ltrace
├── **recipes-extended**	
│ └── **gawk**	append recipe add file /bin/awk to package
├── **recipes-graphics**	graphic support by OpenSource, etnaviv
│ ├── **drm**	libdrm append recipe etnaviv support
│ ├── **mesa**	mesa append recipe
│ ├── **packagegroups**	summarises graphics packages
│ ├── **xorg-driver**	append recipe → including dev package
│ └── **xorg-xserver**	configuration file
├── **recipes-kernel**	kernel
│ └── **linux**	kernel recipe mainline 4.9.7
├── **recipes-multimedia**	multimedia support
│ ├── **alsa**	alsa-plugins append recipe
│ ├── **gstreamer**	gstreamer version 1.10.3
│ └── **packagegroups**	summarises multimedia packages
├── **recipes-qt**	qt support
│ ├── **packagegroups**	summarises qt packages
│ ├── **qt5**	qt append recipes
│ └── **QtDemo-emtrion**	emtrion's qt demo
└── **scripts**	empty

6.3 Provided images

The layer provides three images.

- **core-image-purs**
  initramfs used for emtrion's update mechanism
- **devkit-emtrion-mx6**
  Image based on the core-image-minimal provided by the layer meta of poky, however with extensive functionality
- **devkit-emtrion-mx6-sdk**
  The SDK variant of devkit-emtrion-mx6

6.4 Configuration

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 several configuration files.

- layer.conf
  → Specifies the layer's priority and makes it recipes visible while the build process.
- ${**MACHINE**}.conf

→ Defines the target specific properties like what kernel will be used, type of kernel image, type of the RFS image, features will be added to the RFS image and more. For the developer kit the variable MACHINE has assigned to **emtrion-mx6**.

- bblayers.conf
  → Lists all the meta-layers required by the build process. The various meta-layers provide the packages used by the build process and included in the final image.
- local.conf
  → Include settings of the build environment. The global variable MACHINE is assigned, the max number of processors are available, number of running threads at the same time, location of directories <**BUILD_DWNL**> and <**BUILD_SSTATE**> and more. The file is automatically created by the setup script of the layer.

This developer kit supports a system where Qt resides directly on OpenGL without any window. **eglfs** has configured as default platform.

6.5 Setting up the build system

Setting up the build system, a user friendly setup script "**setup-environment**" exists. This script is located in the root of the layer **meta-emtrion-mx6**. Performing the script, all the required meta-layers will be installed for the release **morty** and the build environment is completed. In addition, to save disk space, the central locations <**BUILD_DWNL**> and <**BUILD_SSTATE**> are created.

6.5.1 Behaviour of the setup script "setup-environment"

The setup script can not only used for setting up a clean build system. It is also useful in the case you have closed the console of a completed build environment and you want to reopen a new one.

However, restarting the script to reopen a new build session, the behavior of the setup script is different if it is performed the first time.

The differences are listed below.

behaviour items	**at first time**	**at restart**	**remarks**
meta-layers	fetched	updated	
<BUILD_DIR>	created	obtained or created*	
configuration file	created	updated or created*	local.conf
configuration file	copied	copied	bbayers.conf
directory downloads	created	obtained	
directory sstate-cache	created	obtained	
asking for deleting build system	no	yes	delete process can take several minutes
asking for NXP EULA	yes	yes	

(*) if deleting of the build system is performed

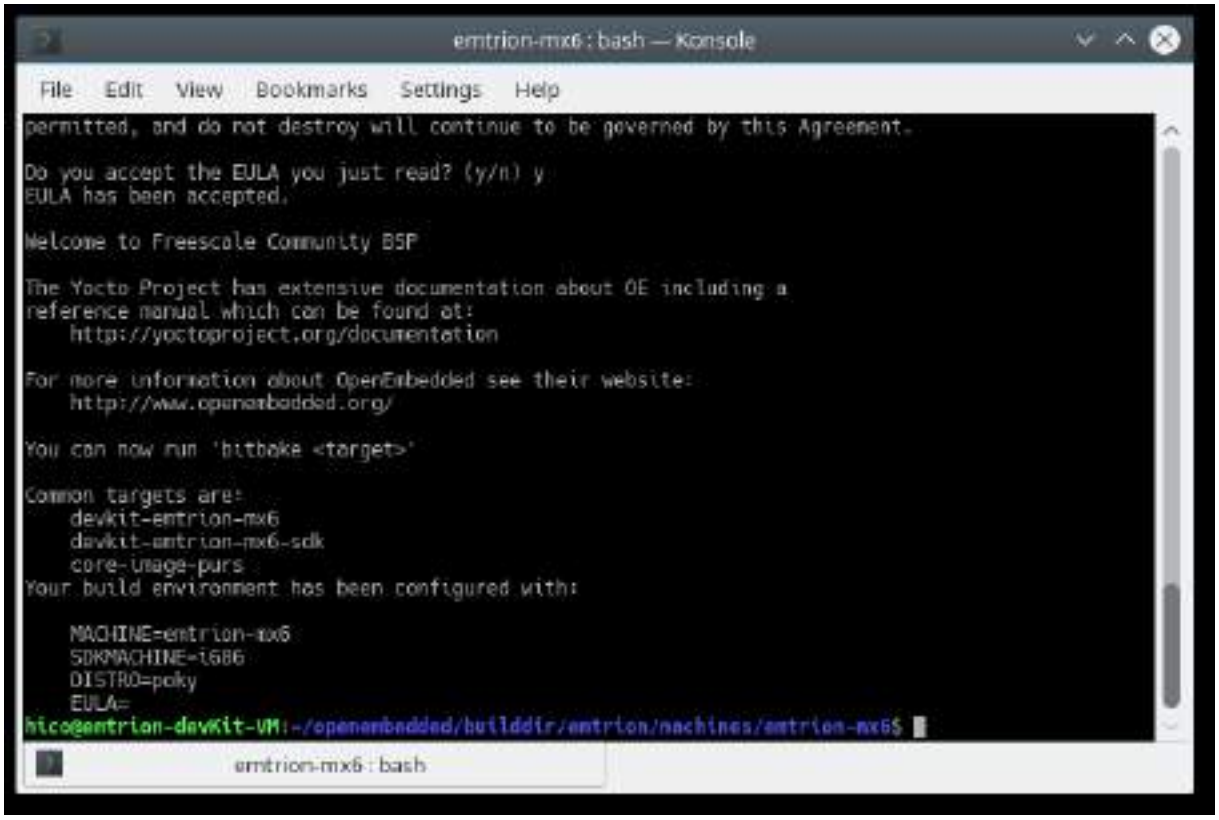
6.5.2 Performing the setup script

Starting the setup process, the script has to be sourced. Enter from the layer's location <**INST_DIR**>/meta-emtrion-mx6

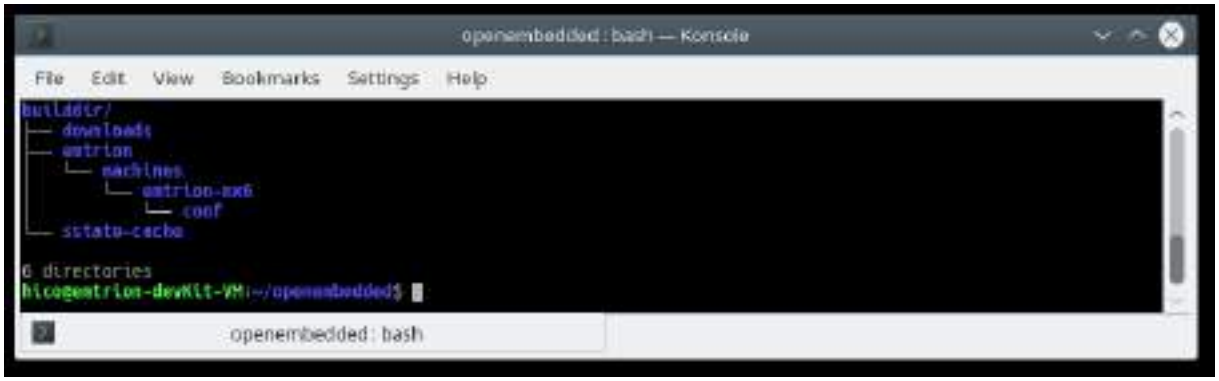
MACHINE=emtrion-mx6 source ./setup-environment

While performing, the setup script will ask you to confirm the NXP EULA using firmware for the imx-vpu. Confirming the EULA, the setup completes and finishes with prompting the build environment as shown below. At the output you can see the provided images.

From now you are able to build images for your developer kit with emcon-mx6 or dimm-mx6.



The created directory structure while performing the setup script is shown below



Location	Remarks
builddir/	
├── downloads	locates fetched data while the build process
├── emtrion	

└── **machines**	locates build directories of various machines
└── **emtrion-mx6**	build directory of emtrion-mx6
└── **conf**	locates local.conf, bblayers.conf
└── **sstate-cache**	locates the build states while the build process

6.6 Creating an image

After setting up the build system you can start building recipes and images for the emcon-mx6 and dimm-mx6 modules.

As mentioned before, the layer meta-emtrion-mx6 provides three images. You can start building an image by prompting bitbake following the name of the image recipe. Enter in the terminal of the build environment

bitbake <**name_of_ image_ recipe**>

- bitbake **core-image-purs**
  Builds the initramfs that is used for emtrion's update mechanism. Due to the image is included by the images devkit-emtrion-mx6 and devkit-emtrion-mx6-sdk, the image is automatically build by bitbaking of these images, but only if the image was still not yet built. For this reason the image has to build explicitly, if any changes were made before building one of the other images.
- bitbake **devkit-emtrion-mx6**
  Builds the image for emtrion imx6. It creates a root file system with OpenGL support, Qt5.7.1 and adds a Qt demo application. Additionally it includes the initramfs, the kernel and device tree.
- bitbake **devkit-emtrion-mx6-sdk**
  Builds the SDK variant of the emtrion imx6 image with source information, however without the Qt demo

6.7 Output files

During the build process a lot of objects and images are created. However, the most relevant images are installed in

<**BUILD_DIR**>**/tmp/deploy/images/emtrion-mx6**

respectively

<BUILD_DIR> /tmp/deploy/sdk.

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

Images	**Description**
zImage (*)	Kernel
zImage-imx6dl-dimm.dtb (*) **zImage-imx6dl-emcon.dtb(*)** **zImage-imx6q-dimm.dtb(*)** **zImage-imx6q-emcon.dtb(*)**	Device trees of the various mx6 modules
devkit-emtrion-mx6.tar.gz (*)	RootFS

devkit-emtrion-mx6-sdk.tar.gz (*)	RootFS for SDK
initramfs-emtrion-mx6.cpio.gz	Ramdisk for update mechanism
poky-glibc-x86_64-devkit-emtrion-mx6-sdk-cortexa9hf-neon-toolchain-2.2.1.sh	SDK installer

(*) means a symbolic link

6.7.1 Root File System

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

sudo tar xf tmp/deploy/images/ emtrion-mx6 /name_of_rootfs_archive –C <**NFS_ROOTFS**>

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

6.7.2 boot directory

The directory structure of the root file system includes a location boot. In addition to the kernel image, device tree and initramfs(restore root file system) a file **uboot_script** is located there.

This file implements some U-Boot command sequences. You can use it for the purpose of updating and booting the RootFS by using NFS.

However, the environment of the U-Boot has to be set up before. This is discussed in detail in the chapter 7 of the Bootloader.

6.8 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/

7 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 or NFS.

7.1 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.

7.2 Using U-Boot to change boot device or update 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 prompting

sudo ip a

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>
```

7.2.1 Boot setup and updating of the system

7.2.1.1 Updating of the system(root file system and kernel)

Due to an image archive contains the root file system as well as the kernel, updating of the system affects always both.

Before performing the update process, following steps have to be done

On VM

- Install the update archive **update-emtrion-mx6.tar.gz** as described in the chapter 5, if not yet done.
- Copy the image to the nfs share <**NFS_UPDATE**>**/boot**.

  For the image created by the Yocto build process enter the command

  cp <**BUILD_DIR**>/tmp/deploy/images/emtrion-mx6/devkit-emtrion-mx6.tar.gz <**NFS_UPDATE**>/boot/

On U-Boot

- Setting up the environment as follow.

```
U-Boot # setenv serverip        <ip-address of the VM>
U-Boot # setenv nfsroot         <NFS_UPDATE>
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.

7.2.1.2 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 NFS

Before perform booting, following steps have to be done.

On VM

- Clean a possibly installed image in the nfs share
  sudo rm –r <**NFS_ROOTFS**>/*
- Install of either the pre-build image or the image created by the Yocto build process to the nfs share <**NFS_ROOTFS**>.
  In case of the created image enter the command
  sudo tar xf <**BUILD_DIR**>/tmp/deploy/images/emtrion-mx6/ \
  devkit-emtrion-mx6.tar.gz –C <**NFS_ROOTFS**>

On U-Boot

- Setting up the environment as follow.

```
U-Boot # setenv serverip        <ip-address of the VM>
U-Boot # setenv nfsroot         <NFS_ROOTFS>
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 by using NFS.

8 SDK

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

One of the methods to create a SDK is using the build system as described in the chapter 6.66.

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 in

<BUILD_DIR>/tmp/deploy/sdk/

8.1 Installing the SDK

Performing the SDK installer, you are asked for the installation directory. The default location is **/opt/poky/…**. Let the default and confirm it. From inside the location **<BUILD_DIR>** start the installer as follows.

./tmp/deploy/sdk/ poky-glibc-x86_64-devkit-emtrion-mx6-sdk-cortexa9hf-neon-toolchain-2.2.1.sh

8.1.1 Setting up the SDK environment

Before you can start developing applications 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-cortexa9hf-neon-poky-linux-gnueabi

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

9 How to use QtCreator with the developer kit

QtCreator is an integrated development environment created by the Qt Project. It is preinstalled in the VM. A suitable kit **emtrion-MX6-yocto** and device **emtrion-mx6-remote** is configured to work with our target device.

Before starting QtCreator be sure the SDK has been installed.

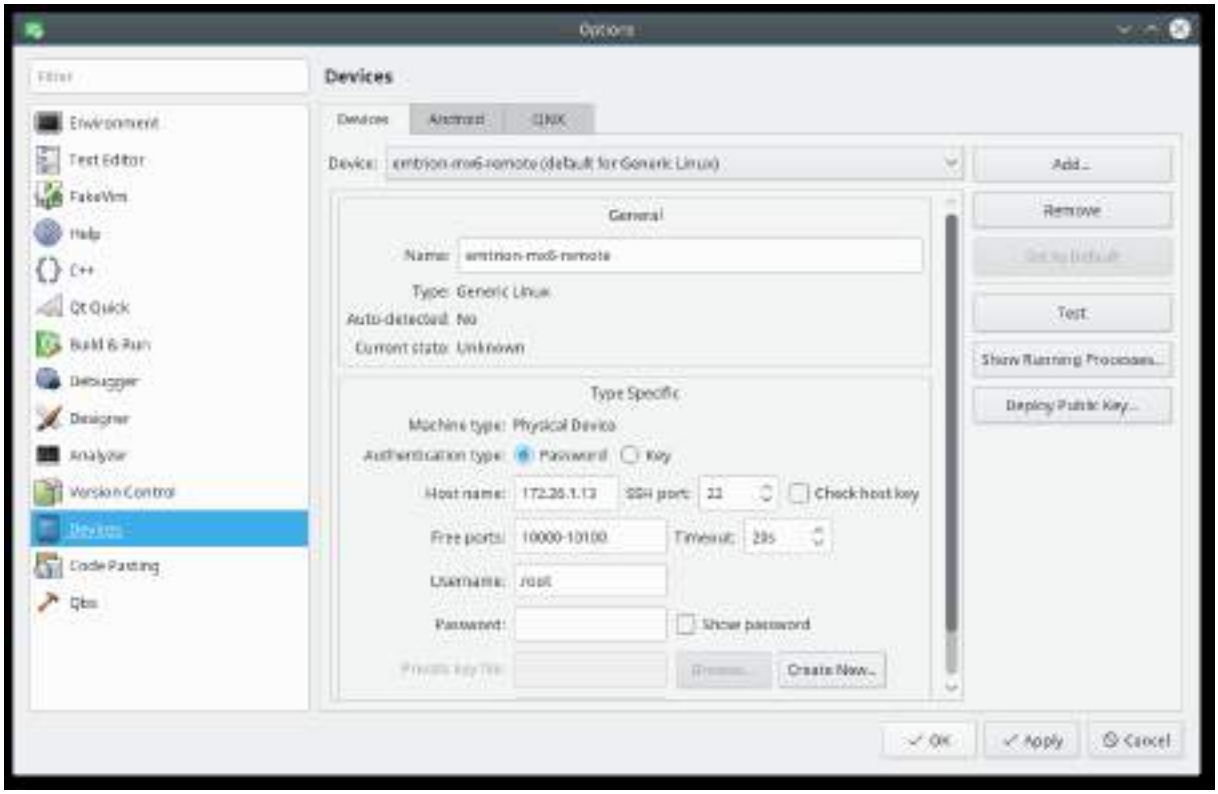
Starting of QtCreator, enter the following command in an open window terminal.

qtcreator &

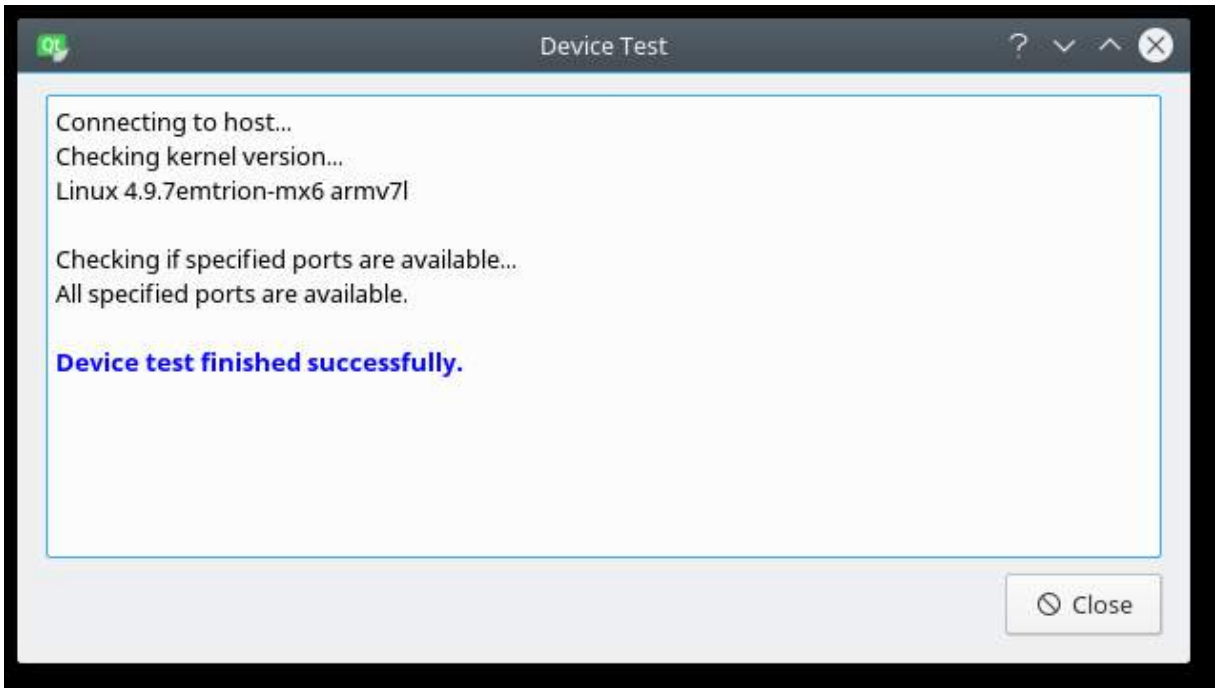
9.1 Device setup

For general information about QtCreator you can check the link "**Get Started Now**".

As a first step 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:



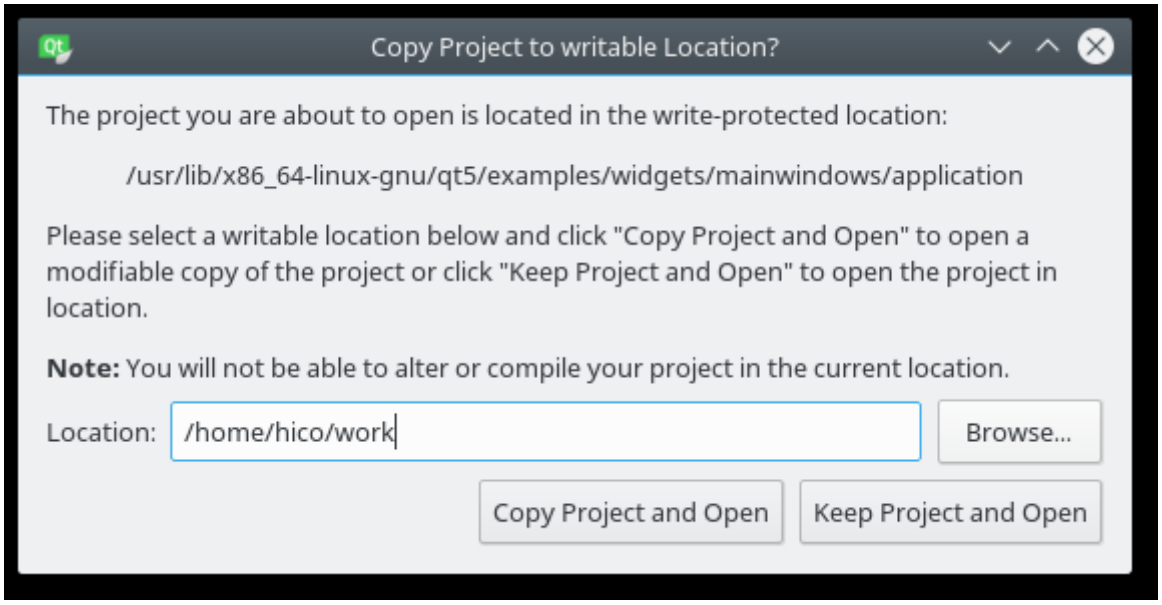
If this does not work, you have to check your network setup.

9.2 Build & Run an Example

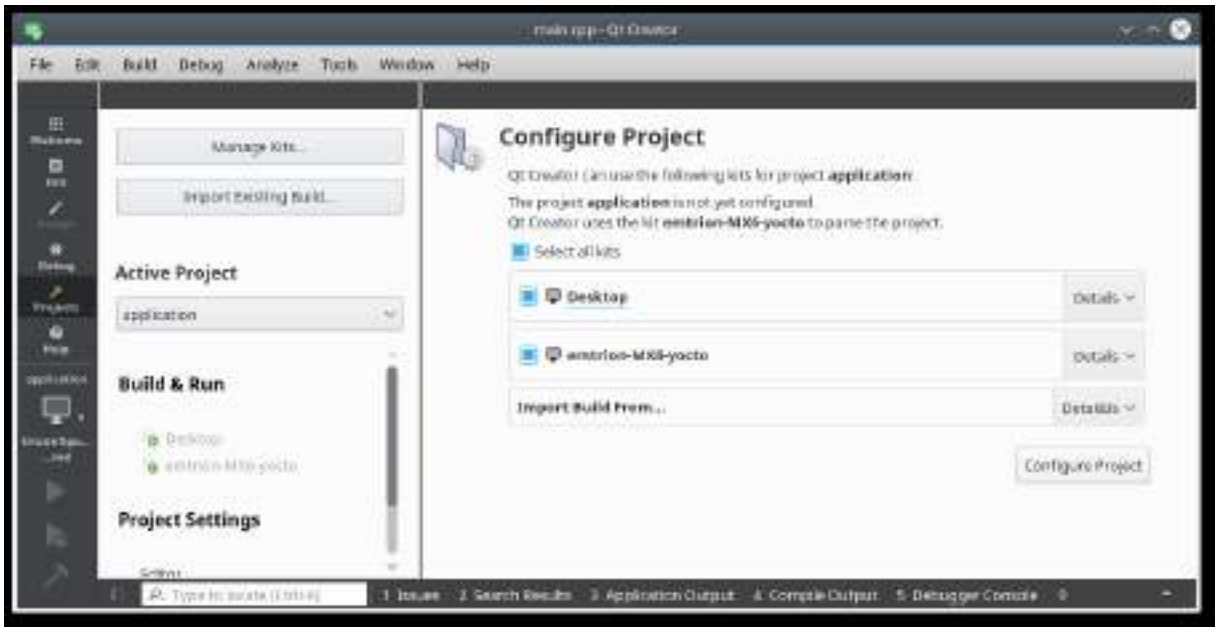
When the connection to the developer kit is successfully established, you can build and run one of the Qt examples on it. First stop the demo from running on the device. Go again to "**Tools->Options**" and select "**Device**" on the left. Now run "**Show running processes**" on the right. In this dialog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

Close the windows to get back to the main view. In the left bar of the main view select "**Examples**"

Make sure that the search path is set to "Qt 5.7.1 in PATH (qt5)". Select the first example called "**Application Example**". As default working directory **/home/hico/work** is displayed. Confirm and hit the button "Copy Project and Open".

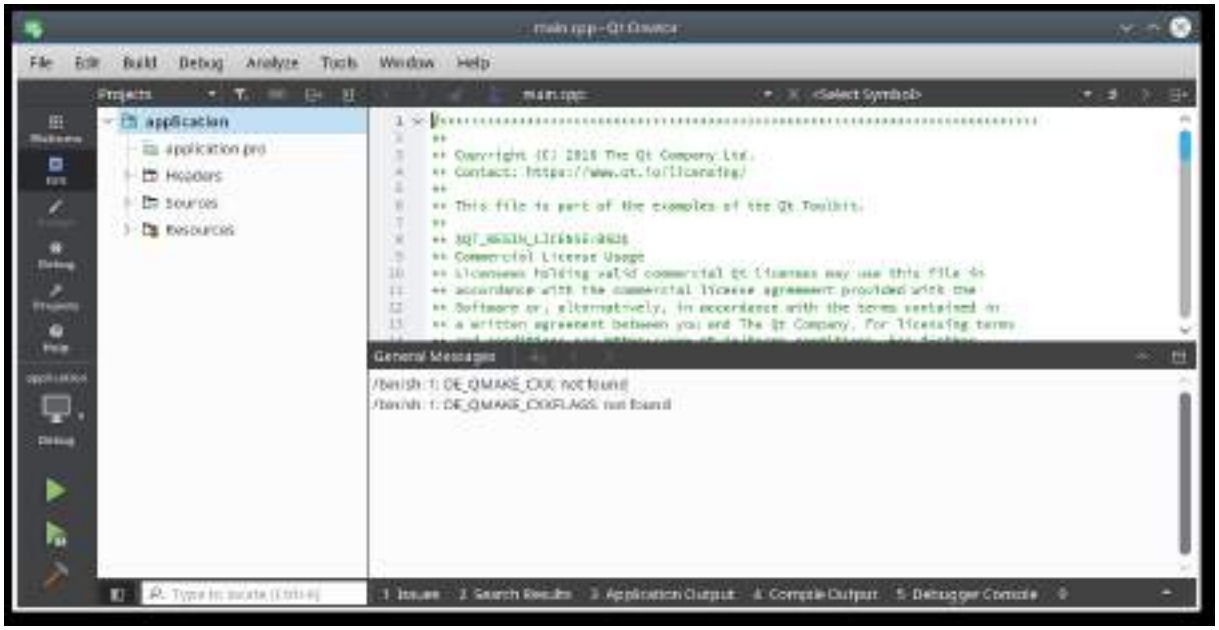


The just opened project is not configured. You can select the kits the project to configure for. At the symbol “application” of the left bar the configure state is shown.

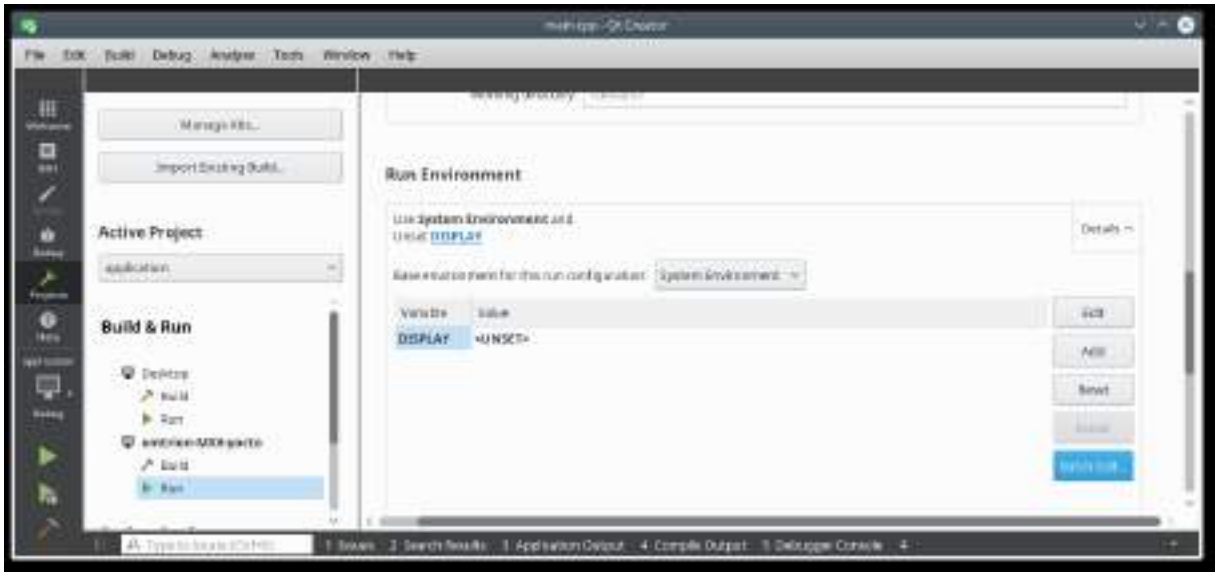


Selecting both kits is allowed, however emtrion-MX6-yocto is the default for building and running. Press the button “Configure Project” to configure the project.

Following window opens.

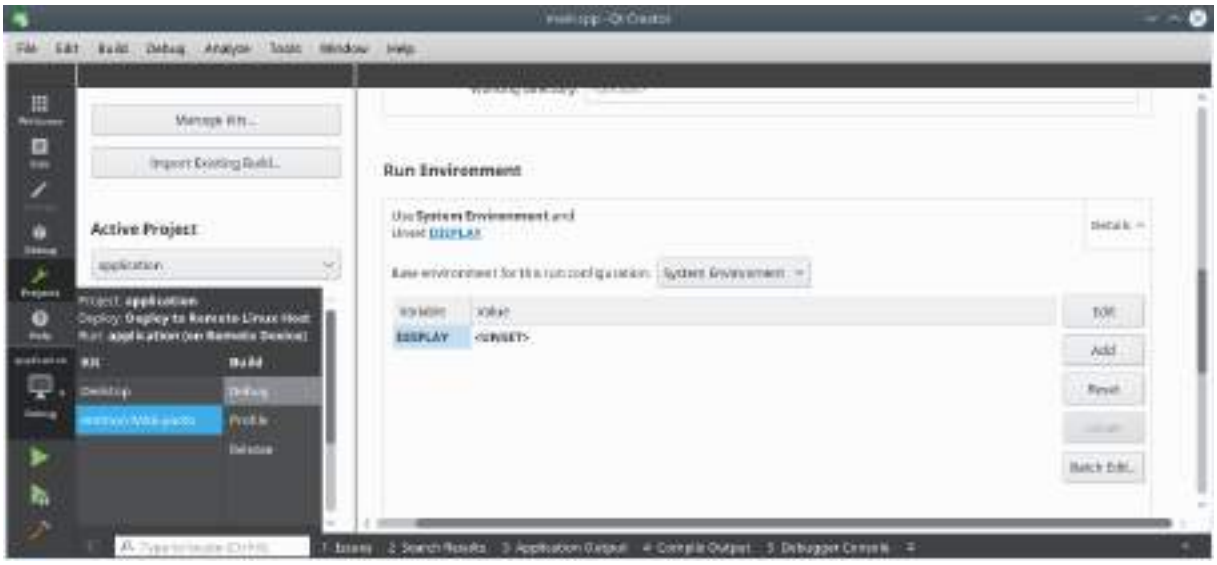


From here you can change build and run settings. In the case of the kit **emtrion-MX6-yocto**, the variable DISPLAY of the Run Environment has to be unset. Select the symbol “Projects” of the left bar and navigate to the Run Environment of the Run settings to unset the variable display.



Now the project is ready for building, debugging and running using the symbols Hammer, green triangle of the left bar. emtrion-MX6-yocto is the default.

Within the symbol “application” you can change the Kit, the project is building debugging and running for.



9.2.1 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.

10 Further Information

10.1 Online resources

Further information can be found on the emtrion support pages.

www.support.emtrion.de

10.2 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.