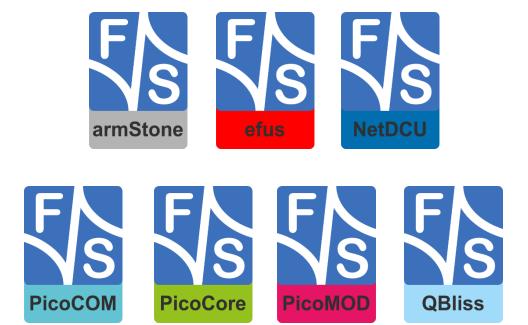
# **F&S Introduction to QT**

## Debugging an Application

Version 1.12 (02.05.2025)





© F&S Elektronik Systeme GmbH Untere Waldplätze 23 D-70569 Stuttgart Germany

> Phone: +49(0)711-123722-0 Fax: +49(0)711-123722-99

## **About This Document**

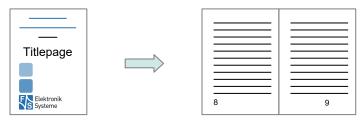
This document describes how to debugging remote device using QT Creator Automatic Remote Debugging Launcher under Linux. The software is configured for architectures fsimx6/fsimx6sx/fsimx6ul/fsimx7ulp/fsimx8mm from F&S under Linux/Buildroot.

#### Remark

The version number on the title page of this document is the version of the document. It is not related to the version number of any software release. The latest version of this document can always be found at <u>http://www.fs-net.de</u>.

#### How to Print This Document

This document is designed to be printed double-sided (front and back) on A4 paper. If you want to read it with a PDF reader program, you should use a two-page layout where the title page is an extra single page. The settings are correct if the page numbers are at the outside of the pages, even pages on the left and odd pages on the right side. If it is reversed, then the title page is handled wrongly and is part of the first double-page instead of a single page.



#### **Typographical Conventions**

We use different fonts and highlighting to emphasize the context of special terms:

File names

#### Menu entries



PC Input/ou

Listings

Generic input/output

Variables

# History

| Date       | V    | Platform | A,M,R       | Chapter              | Description   | Au |
|------------|------|----------|-------------|----------------------|---|----|
| 2018-04-27 | 1.0  | *        | A           | ALL                  | Create Documentation  | PJ |
| 2018-06-08 | 1.1  | *        | М           | 2.1, 5.2             | Improve setup   | PJ |
| 2018-06-14 | 1.2  | *        | М           | 3                    | Update figure 12, it is necessary to use cmake instead of qmake   | PJ |
| 2020-05-04 | 1.3  | *        | M<br>M<br>M | 2<br>3<br>All        | Improve some setences<br>Update figures for new QT Creator, improve some sentences<br>Improve layout  | PJ |
| 2021-03-11 | 1.4  | *        | М           | 3,<br>5.2<br>5.3     | Add GDB path for 64 bit CPUs<br>Add target path for Yocto<br>Add backend support for debugging  | PJ |
| 2022-07-19 | 1.5  | *        | M           | 2,<br>3,<br>5.1,5.2  | Add Yocto Paths<br>Updated some sentences   | TG |
| 2022-07-26 | 1.6  | *        | М           | 3                    | Changed Toolchain distro  | TG |
| 2022-09-19 | 1.7  | *        | A,M         | 1.2,<br>3,           | Add Paths for i.MX6   | TG |
| 2022-11-09 | 1.8  | *        | М           | 1.1                  | Added installation for QT Creator 4.15.2 for Fedora 30  | TG |
| 2023-01-10 | 1.9  | *        | М           | 1, 2.2, 3            | Updated figures for new QT Creator<br>Added PicoCORE in Introduction  | TG |
| 2023-06-09 | 1.10 | *        | М           | 1.1,2.1,3            | Add information about environment-setup<br>Add information about sftp<br>Fix missing <i>"/"</i> in command  | PG |
| 2023-09-28 | 1.11 | *        | A,M         | 1, 1.1,<br>1.2, 3, 4 | Add general support for QT6<br>Add colored separation of displayed steps for QT5 and QT6 +<br>Buildroot and Yocto separation<br>Add paths for QT6 | TG |
| 2025-05-02 | 1.12 | *        | A,M         | 3, 6                 | Build Improvements and CMAKE Build instructions   | SC |

A,M,R Au Author





# **Table of Contents**

| 1 | Intro  | oduction                           | 1  |
|---|--------|------------------------------------|----|
|   | 1.1    | Install Qt Creator on Fedora       | 2  |
|   | 1.2    | Tested Boards and Releases         | 4  |
| 2 | Rem    | note Connection                    | 5  |
|   | 2.1    | Setup SSH connection on SBC/SOM    | 5  |
|   | 2.2    | Setup SSH connection in Qt Creator | 7  |
| 3 | Setu   | up Application settings            | 10 |
| 4 | Crea   | ate an Application                 | 17 |
|   | 4.1    | Create hello world app             |    |
| 5 | Buil   | d and Debug Application            | 29 |
|   | 5.1    | Desktop Run                        |    |
|   | 5.2    | FS Cross Compile Kit for ARM Run   |    |
|   | 5.3    | Debug build                        |    |
| 6 | Buil   | d CMAKE Projects                   | 35 |
|   | 6.1    | Background                         |    |
|   | 6.2    | Prerequisites                      |    |
|   | 6.3    | Step-by-Step Instructions          |    |
| 7 | Арр    | endix                              | 36 |
|   | List o | f Figures                          |    |
|   | Impor  | rtant Notice                       |    |





# 1 Introduction

F&S offers a whole variety of Systems on Module (SOM) and Single Board Computers (SBC). There are different board families that are named NetDCU, PicoMOD, PicoCOM, PicoCore, armStone, QBliss, and efus.

Linux is available for all of these platforms. Also QT is available for all of these platforms. Linux offers a tool called Qt Creator which is a cross-platform C++, JavaScript and QML integrated development environment which is part of the SDK for the Qt GUI application development framework. It includes a visual debugger and an integrated GUI layout and forms designer.

This document describes how to debug an application on a remote device using the automatic remote debugging launcher plug-in.

This document describes both the steps for Qt5 and Qt6.

The Qt5 steps will be displayed like the following:

...example of paths or code.

And the Qt6 steps like this:

...example of paths or code.

If the steps are not displayed separately, it means that the steps work for both Qt5 and Qt6.



Introduction

### 1.1 Install Qt Creator on Fedora

Install Qt Creator on your development host with your Linux Distribution specific package manager system. In our development machine, which you can download from our home-page, QT Creator is already installed. On Fedora up to version 27:

sudo yum install qt-creator

On Fedora 30 we will need to install a new version of QT Creator:

wget https://download.qt.io/official\_releases/qtcreator/4.15/4.15.2/qtcreator-opensource-linux-x86 64-4.15.2.run

Next we will need to give the file execute permissions:

chmod +x qt-creator-opensource-linux-x86 64-4.15.2.run

Now execute the file and follow the Installers instructions:

./qt-creator-opensource-linux-x86 64-4.15.2.run

Finally you want to create a link to the new QT Creator. Go into the directory that has been created for the new QT Creator and execute the following command:

sudo ln -sf /bin/qtcreator /usr/bin/qtcreator

Then start Qt Creator:

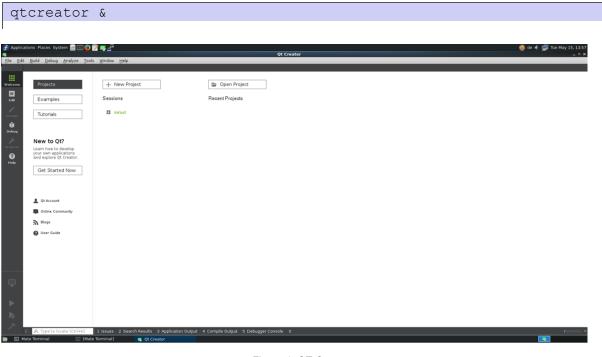


Figure 1: QT Creator



#### Note: If Yocto toolchain is used, qtcreator must be started from the same terminal where SDK environment are set by command: source <toolchain directory>/environment-setup-aarch64-poky-linux

See  ${\bf 3}$  Setup Application settings on how to build and install the QT5 SDK

#### Alternatively, create a shell script to start both commands sequentially.

E.g.: qt5-yocto.sh

```
#!/usr/bin/sh
source <toolchain directory>/environment-setupaarch64-poky-linux
qtcreator &
```



Introduction

## 1.2 Tested Boards and Releases

| Board:          | Revision: | Release:                      |
|-----------------|-----------|-------------------------------|
| PicoCoreMX8MM   | 1.20      | Buildroot (fsimx8mm-B2021.06) |
| PicoCoreMX8MM   | 1.20      | Yocto (fsimx8mm-Y2021.04)     |
| ArmStoneA9      | 1.10      | Yocto (fsimx6-Y2020.03)       |
| ArmStoneA9      | 1.10      | Buildroot (fsimx6-B2021.10.1) |
| PicoCoreMX8MPr2 | 1.10      | Yocto (fsimx8mp-Y2023.09)     |



# 2 Remote Connection

First of all the SSH connection on the remote system will be setup.

## 2.1 Setup SSH connection on SBC/SOM

You have to setup your F&S Board. Therefore you can have a look into F&S *LinuxOnFS-Boards\_eng.pdf*. After that boot your F&S Board.

To work with SSH the board should have a valid date. This is necessary to create certificates for SSH. To setup a date you can use the following command:

date "2020-05-04 10:13"

Afterwards we have to enable the network interface. You can also set the network on command in UBoot to enable network interface at each boot. For further information please take a look in *FSiMX6\_FirstSteps\_eng.pdf*.

Dynamically:

udhcpc -i eth0

Static (e.g. ip address is 10.0.0.84):

ifconfig eth0 10.0.0.84 up

The Root-Filesystem is read-only mounted, but we have to modify something in the filesystem so we need it read-writeable.

mount -o remount,rw /

Open sshd\_config file

vi /etc/ssh/sshd\_config

and edit the following lines:

```
…
PermitRootLogin yes
```

Optional you can also allow to login without password, but we recommend you to not do this because it's a security risk. If you want to do it anyway add the following line to *sshd\_config* file.

```
...
PermitEmptyPasswords yes
```



#### **Remote Connection**

Make sure to use the internal sftp if no external binary is installed.

```
...
Subsystem sftp internal-sftp
```

After that you have to start the SSH server (This step is not required when using Yocto).

/etc/init.d/S50sshd start

Now the SSH server is running on our SBC/SOM. Let's test it, therefore we are going back to our VM. To connect via ssh we open a Terminal and send the following command:

ssh root@10.0.84

[jakob@localhost ~]\$ ssh root@10.0.0.84 The authenticity of host '10.0.0.84 (10.0.0.84)' can't be established. ECDSA key fingerprint is b1:b1:aa:83:12:d1:f1:21:7b:e3:6a:61:89:6e:31:ea. Are you sure you want to continue connecting (yes/no)? yes Warning: Permanently added '10.0.0.84' (ECDSA) to the list of known hosts.

Figure 2: SSH connection terminal

Now the SSH connection is successfully established and tested.



## 2.2 Setup SSH connection in Qt Creator

|                 |  |  |  |   | Options   |   |  |   |   | ×  |
|-----------------|--|--|--|---|---|---|--|---|---|--|
| lter            |  | Enviror  | nment  |   |   |   |  |   |   |  |
| Environment     |  | Interface  | System   | Keyboard  | External Tools  | MIME  | Types  | Locator   |   |  |
| Text Editor     |  | User Int   | erface   |   |   |   |  |   |   |  |
| Fake∨im         |  | Color:   |  |   | Reset   |   |  |   |   |  |
| 🔊 Help          |  | Theme  | e:   | Flat  | <b>▲</b><br>▼   |   |  |   |   |  |
| ĵ <b>}</b> c++  |  | Langua   | age:   | <system< th=""><th>n Language&gt;</th><th><br/>▼</th><th></th><th></th><th></th><th></th></system<>   | n Language>   | <br>▼   |  |   |   |  |
| 🔍 Qt Quick      |  | Reset  | t Warnings   | i   |   |   |  |   |   |  |
| 🚯 Build & Run   |  |  |  |   |   |   |  |   |   |  |
| 🐊 Debugger      |  |  |  |   |   |   |  |   |   |  |
| 发 Designer      |  |  |  |   |   |   |  |   |   |  |
| 📕 Analyzer      |  |  |  |   |   |   |  |   |   |  |
| Version Control |  |  |  |   |   |   |  |   |   |  |
| 🏚 Android       | Ļ  |  |  |   |   |   |  |   |   |  |
|                 | •  |  |  |   |   |   | A  | pply  | <u>C</u> ancel  | <u>о</u> к   |
|                 | <ul> <li>Environment</li> <li>Text Editor</li> <li>FakeVim</li> <li>Help</li> <li>C++</li> <li>Qt Quick</li> <li>Build &amp; Run</li> <li>Debugger</li> <li>Designer</li> <li>Analyzer</li> <li>Version Control</li> </ul> | <ul> <li>Environment</li> <li>Text Editor</li> <li>FakeVim</li> <li>Help</li> <li>C++</li> <li>Qt Quick</li> <li>Build &amp; Run</li> <li>Debugger</li> <li>Designer</li> <li>Analyzer</li> <li>Version Control</li> </ul> | <ul> <li>Environment</li> <li>Text Editor</li> <li>FakeVim</li> <li>Calor:</li> <li>Help</li> <li>C++</li> <li>Qt Quick</li> <li>Build &amp; Run</li> <li>Debugger</li> <li>Designer</li> <li>Analyzer</li> <li>Android</li> </ul> | <ul> <li>Environment</li> <li>Text Editor</li> <li>FakeVim</li> <li>Cel++</li> <li>Qt Quick</li> <li>Build &amp; Run</li> <li>Debugger</li> <li>Analyzer</li> <li>Android</li> <li>Interface System</li> <li>User Interface</li> <li>Color:</li> <li>Theme:</li> <li>Language:</li> <li>Reset Warnings</li> </ul> | Iter Environment   Interface System   Text Editor User Interface   FakeVim Color:   Help Theme:   C++ Language:   Qt Quick Reset Warnings   Build & Run Reset Warnings   Debugger Analyzer   Analyzer Android | Environment   Text Editor   FakeVim   Help   C++   Qt Quick   Build & Run   Debugger   Onesigner   Analyzer   Version Control   Android   Interface System Keyboard External Tools User Interface Color: Color: Flat Flat Flat Flat Flat Flat Flat Flat | Iter   Environment   Interface   System   Keyboard   External Tools   MIME   User   Interface   Color:   Color:   Reset   Theme:   Flat   Color:   Reset   Theme:   Flat   Language:   System Language>   Reset Warnings | Iter     Environment     Text Editor   FakeVim   Help   C++   Qt Quick   Build & Run   Debugger   Analyzer   Version Control   Android    Environment   Interface   System Keyboard External Tools MIME Types    User Interface   Color:   Reset   System Language> | Iter     Environment     Interface     System Keyboard External Tools MIME Types Locator   User Interface      Color:   Reset   Color:   Theme:   Flat   Theme:   Flat   Language:   System Language>           Pebugger   Designer   Analyzer   Analyzer | <pre>ter Environment Text Editor Text Editor FakeVim Help C++ Qt Quick Build &amp; Run Debugger Analyzer Yersion Control Android</pre> |

Open Tools. In the toolbar and select Options....

Figure 3: Options

Figure 4: Options - Devices

Choose *Devices* on the left side.

| <b>8</b>        | Options  | ×                        |
|-----------------|--|--------------------------|
| Filter          | Devices  |                          |
| () c++          | Devices  |                          |
| 📣 Qt Quick      | Device: Local PC (default for Desktop)                       | <u>A</u> dd              |
| 🕕 Build & Run   | General  | <u>R</u> emove           |
| 🔍 Debugger      | Name: Local PC   | Set As Default           |
| 💥 Designer      | Type: Desktop<br>Auto-detected: Yes (id is "Desktop Device") | Show Running Processes   |
| Analyzer        | Current state: Unknown                                       |                          |
| Version Control | Type Specific  |                          |
| 🧔 Android       |  |                          |
| ®awx QNX        |  |                          |
| Devices         |  |                          |
| Code Pasting    |  |                          |
| 🅕 Qbs           |  |                          |
|                 | Apply  | <u>Cancel</u> <u>O</u> K |
|                 |  |                          |

#### **Remote Connection**

| 9               | Options X   |
|-----------------|---|
| Filter          | Devices   |
| () c++          | Devices           Device Configuration Wizard Selection         X |
| 🚄 Qt Quick      | Dev Available device types: <u>A</u> dd                           |
| 🕕 Build & Run   | G Generic Linux Device Remove                                     |
| 🔍 Debugger      | Set As Default  |
| 💥 Designer      | how Running Processes   |
| 🛄 Analyzer      |   |
| Version Control | ту  |
| 🤯 Android       |   |
| ≑anx QNX        |   |
| Devices         |   |
| Code Pasting    | <u>C</u> ancel Start Wizard                                       |
| 🗡 Qbs           |   |
|                 | Apply <u>C</u> ancel <u>O</u> K                                   |

Now add a new device. Press Add..., choose Generic Linux Device and click Start Wizard.

Figure 5: Add device configuration

Now set a name for the configuration, enter the IP address and the login data.

|                | New Generic Linux Device Configurati     | on Setup — Qt Creato | r                     | × |
|----------------|--|----------------------|-----------------------|---|
| Connection     |  |                      |                       |   |
|                |  |                      |                       |   |
| ₀≫ Connection  | The name to identify this configuration: | F&S Linux Device     |                       |   |
| Key Deployment | The device's host name or IP address:    | 10.0.0.11            |                       |   |
| Summary        | The username to log into the device:     | root                 |                       |   |
|                |  |                      |                       |   |
|                |  |                      |                       |   |
|                |  |                      |                       |   |
|                |  |                      |                       |   |
|                |  |                      |                       |   |
|                |  |                      |                       | _ |
|                |  | (                    | <u>N</u> ext > Cancel |   |
|                | Figure 6: Setup device co                | nnection             |                       |   |

QT5 Application Development



Afterwards press *Next* > *Deploy Public Key* > *Next* and *Finish*. Then the ssh connection will be automatically tested.

| 0 | Device Test  | × |
|---|--|---|
|   | Connecting to host<br>Checking kernel version<br>Linux 4.1.15-F+S armv7l<br>Checking if specified ports are available<br>All specified ports are available.<br><b>Device test finished successfully.</b> |   |
|   | Figure 7: Device test  |   |



# **3 Setup Application settings**

Before you can setup the application settings be sure that you have built your Root-Filesystem (**Buildroot/Yocto**) with the correct configuration (fsimx6/ul/sx\_qt5\_defconfig) once. This is necessary because we need the compiled GDB debugger and other packages from this directory. If you are using Yocto you will need to create a populated SDK as well. Building the toolchain will take at least 12 GB of RAM.

#### Note:

If the release version which you are using is less then fsimx6-V3.1, fsimx6sx-V2.1 or fsimx6ul-V2.1, then you have to add the following packages to your root filesystem and rebuild it, e.g. through "*make menuconfig*":

- ctest, located in Target packages/Development Tools/ctest
- *Python support*, located in Toolchain/Build Cross gdb for the host/Python support

If you have already built your root filesystem, then you have to run the following command in your terminal:

```
...buildroot-2019.05.3-fsimx6-B2020.04]$ make host-gdb-reconfigure
...buildroot-2019.05.3-fsimx6-B2020.04]$ make
```

#### Populate Yocto SDK

To build and install the Yocto-SDK head over to your build directory and run the following command:

```
...<yocto-build-directory>]$ source source_env
...<yocto-build-directory>]$ bitbake -c populate sdk <yocto-image>
```

Next you need to head to ...<Build-Dir>/tmp/deploy/sdk and run the .sh script that now exist. After that you can find your toolchain in your desired directory.



After that you have to install the new root filesystem on your remote target system. Now select *Tools* and *Options…* in your QT Creator. Select Kits on the left side bar.

|         |             | Options — Qt Creator                       | ×                        |
|---------|-------------|--|--------------------------|
| Filter  |             | Kits                                       |                          |
| 🖼 Kits  | <b>^</b>    | Kits Qt Versions Compilers Debuggers CMake |                          |
| 🖵 Envir | ronment     | Name<br>Auto-detected                      | Add                      |
| E Text  | Editor      | - Manual                                   | Clone                    |
| K. Fake | Vim         | Desktop (default)<br>Main                  | Remove                   |
| Help    |             |  | Make Default             |
| {} C++  |             |  | Settings Filter          |
| 🚽 Qt Q  |             |  | Default Settings Filter  |
| ➤ Build |             |  |                          |
| ·       |             |  |                          |
| ∰ Qbs   |             |  |                          |
| 🏦 Debu  |             |  |                          |
| 🖍 Desig |             |  |                          |
| n Pytho | on          |  |                          |
| 🔄 Analy | yzer        |  |                          |
| 🕒 Versi | ion Control |  |                          |
| Devic   | ces         |  |                          |
|         | T           |  | ● <u>C</u> ancel <b></b> |

Figure 8: Options - Kits



|                              | Options — Qt Creator  |               |
|------------------------------|---|---------------|
| Filter                       | Kits  |               |
| TH Kits                      | Kits Qt Versions Compilers Debuggers CMake  |               |
| Environment                  | Name r gmake Location   | Add           |
| Text Editor                  | ✓ Manual<br>Qt 5.12.5 in PATH (System) /usr/bin/qmake-qt5<br>Qt 5.15.0 (System) /opt/fus-imx-wayland/5.4-zeus/sysroots/x86_64-pokysdk-linux/usr/bin/qmake   | Remove        |
| 🖌 FakeVim<br>🕜 Help          |   | Link with Qt  |
| (} C++                       |   | Clean Up      |
| At Quick                     |   |               |
| Build & Run                  |   |               |
| ∰ Qbs<br>Mit Debugger        |   |               |
| <ul> <li>Designer</li> </ul> |   |               |
| ခ္စ Python                   | Version name:         Qt %(Qt:Version) (System)           gmake location:         /opt/fus-imx-wayland/5.4-zeus/sysroots/x86_64-pokysdk-linux/usr/bin/qmake |               |
| Analyzer                     |   |               |
| Version Control              | Qt version 5.15.0 for Desktop Details *   |               |
| Devices                      | Register documentation: Highest Version Only -  |               |
|                              | ✓ Apply   | I <u>₩0</u> K |

Choose QT Versions and select Add..., setup the path to the qmake location.

Figure 9: Kits - Qt Versions

#### Examples:

#### QT5:

#### (Buildroot)

```
/home/developer/fsimx6-B2020.04/build/buildroot-2019.05.3-fsimx6-
B2020.04/output/host/usr/bin/qmake
```

#### (Yocto/i.MX8)

```
/opt/fus-imx-wayland/5.4-zeus/sysroots/x86_64-pokysdk-
linux/usr/bin/qmake
```

#### (Yocto/i.MX6)

/opt/poky/2.4.2/sysroots/x86 64-pokysdk-linux/usr/bin/qt5/qmake

#### QT6:

#### (Yocto/i.MX8)

```
/opt/fus-imx-wayland/5.15-kirkstone/sysroots/x86_64-pokysdk-
linux/usr/bin/qmake6
```



|  | Options — Qt Creator  |   | (   |
|--|---|---|---|
| Filter   | Kits     Qt Versions     Compilers     Debuggers     CMake  |   |   |
| <ul> <li>Environment</li> <li>Text Editor</li> <li>FakeVim</li> <li>Help</li> <li>C++</li> <li>Qt Quick</li> <li>Build &amp; Run</li> <li>Qbs</li> </ul> | Name         Clang (C, x86 64bit in /usr//bin)         GCC (C, arm 32bit in /usr/local/arm/fs-toolchain-9.3-armv7ahf/bin)         Clang (C, x86 64bit in /home/developer/qtoreator-4.15.2/libexec/qtoreator/clang/bin)         GCC (C, x86 64bit in /usr/lib64/ccache)         GCC (C++, x86 64bit in /usr/lib64/ccache)         GCT (C++, x86 64bi | Type ▲<br>Clang<br>GCC<br>Clang<br>GCC<br>GCC<br>Clang<br>GCC<br>Clang<br>GCC<br>Clang<br>GCC<br>Clang<br>CC<br>Clang<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC<br>CC | Add   Clone  Remove  Remove All  Re-detect  Auto-detection Settings |
| <ul> <li>✤ Debugger</li> <li>✓ Designer</li> <li>e Python</li> <li>E Analyzer</li> </ul>   | Name:     Compiler <u>C</u> ompiler path:     sroots/x86_64-pokysdk-linux/usr/bin/aarch64-poky-linux/aarch64-poky-linux       Platform codegen flags:   |   |   |
| Version Control Devices  | ABI:  | - 64bit     Apply   | Cancel  |

Figure 10:Kits - Compilers

Now select *Compilers* and Add *GCC then C++*. Setup the *Name* and the *Compiler path*. After that *Apply* the settings.

#### Examples:

#### QT5:

#### (Buildroot)

```
/home/developer/fsimx6-B2020.04/build/buildroot-2019.05.3-fsimx6-
B2020.04/output/host/usr/bin/arm-linux-g++
```

#### (Yocto/i.MX8)

```
/opt/fus-imx-wayland/5.4-zeus/sysroots/x86_64-pokysdk-
linux/usr/bin/aarch64-poky-linux/aarch64-poky-linux-g++
```

#### (Yocto/i.MX6)

/opt/poky/2.4.2/sysroots/x86\_64-pokysdk-linux/usr/bin/i586-pokylinux/i586-poky-linux-g++

#### QT6:

#### (Yocto/i.MX8)

```
/opt/fus-imx-wayland/5.15-kirkstone/sysroots/x86_64-pokysdk-
linux/usr/bin/aarch64-poky-linux/aarch64-poky-linux-g++
```



|          |                 |                        |  | Op               | otions — Q    | Creator                            |                               |          | ×       |
|----------|-----------------|------------------------|--|------------------|---------------|------------------------------------|-------------------------------|----------|---------|
| Filter   | r               | Kits                   |  |                  |               |                                    |                               |          |         |
| ΠŶ       | Kits 🗖          | Kits Qt Version        | s Compilers                              | Debuggers        | CMake         |                                    |                               |          |         |
| Q        | Environment     | Name<br>Auto-detected  |  |                  |               |                                    | Location                      |          | Add     |
| E        | Text Editor     | System GE<br>System LL | B at /usr/bin/gdb<br>DB at /usr/bin/lldt | )                |               |                                    | /usr/bin/gdb<br>/usr/bin/lldb |          | Clone   |
| <b>K</b> | FakeVim         | System LL              | B at /bin/gdb<br>DB at /bin/lldb         | 1 1/5 4          |               |                                    | /bin/gdb<br>/bin/lldb         |          | Remove  |
| 0        | Help            | → Manual<br>GDB        | JB at /opt/fus-im)                       | (-wayland/5.4-ze | eus/sysroots  | /x86_64-pokysdk-linux/usr/bin/lldb | /opt/fus-imx-wayia            |          |         |
| {}       | C++             |                        |  |                  |               |                                    | ,                             |          |         |
| 1        | Qt Quick        |                        |  |                  |               |                                    |                               |          |         |
| 7        | Build & Run     | 4                      |  |                  |               |                                    |                               | Þ        |         |
| œ₽       | Qbs             |                        |  |                  |               |                                    |                               |          |         |
| ŵ        | Debugger        | Name:                  | GDB                                      |                  |               |                                    |                               |          |         |
| 1        | Designer        | Path:                  | /opt/fus-imx-                            | wayland/5.4-zeu  | is/sysroots/x | 86_64-pokysdk-linux/usr/bin/aarcl  | n64-poky-linux/aa             | Browse   |         |
| ÷        | Python          | Туре:                  | GDB                                      |                  |               |                                    |                               |          |         |
| E        | Analyzer        | ABIs:                  | arm-linux-ge                             | neric-elf-64bit  |               |                                    |                               |          |         |
| B        | Version Control | Version:               | 8.3.1                                    |                  |               |                                    |                               |          |         |
| Q        | Devices         | Working directo        | y:                                       |                  |               |                                    |                               | Browse   |         |
| -        | Ŧ               |                        |  |                  |               |                                    | 🗸 Apply                       | / Oancel | <u></u> |

Figure 11: Kits - Debuggers

Afterwards select *Debuggers* and *Add*. Setup *Name* and *Path* and *apply* the settings.

Examples:

#### QT5:

# (Buildroot) 32-bit CPUs: /home/developer/fsimx6-B2020.04/build/buildroot-2019.05.3-fsimx6B2020.04/output/host/usr/bin/arm-linux-gdb 64-bit CPUs: /home/developer/fsimx8mm-B2020.08/build/buildroot-2019.05.3fsimx8mm-B2020.08/output/host//bin/aarch64-linux-gdb (Yocto) 32-bit CPUs: /opt/poky/2.4.2/sysroots/x86\_64-pokysdk-linux/usr/bin/i586-pokylinux/i586-poky-linux-gdb 64-bit CPUs:

/opt/fus-imx-wayland/5.4-zeus/sysroots/x86\_64-pokysdklinux/usr/bin/aarch64-poky-linux/aarch64-poky-linux-gdb



#### QT6:

| (Yocto)  |
|--|
| /opt/fus-imx-wayland/5.15-kirkstone/sysroots/x86 64-pokysdk- |
| linux/usr/bin/aarch64-poky-linux/aarch64-poky-linux-gdb      |

|                  |                 |              |                                   |  | Op                  | ptions — Q                       | t Creator  |                        | ×          |
|------------------|-----------------|--------------|-----------------------------------|--|---------------------|----------------------------------|--|------------------------|------------|
| Filte            | r               | Kits         |                                   |  |                     |                                  |  |                        |            |
| <b>T</b> 9       | Kits 📤          | Kits         | Qt Versions                       | Compilers                                  | Debuggers           | CMake                            |  |                        |            |
| Q                | Environment     | Name<br>- Au | uto-detected                      |  |                     | Location                         |  |                        | Add        |
| E                | Text Editor     |              | System CMake<br>System CMake      | <i>e at /usr/bin/cn</i><br>e at /bin/cmake | nake (Default)<br>e | <i>/usr/bin/cm</i><br>/bin/cmake | ake  |                        | Clone      |
| <b>K</b><br>Fake | FakeVim         | ⊸ Ma         | anual<br>CMake                    |  |                     | /opt/fus-im                      | x-wayland/5.4-zeus/sysroots/x86_64- <sub>t</sub> | ookysdk-linux/usr/bin/ | c Remove   |
| 0                | Help            |              |                                   |  |                     |                                  |  |                        | mano Donan |
| {}               | C++             |              |                                   |  |                     |                                  |  |                        |            |
| 1                | Qt Quick        |              |                                   |  |                     |                                  |  |                        |            |
| 7                | Build & Run     |              |                                   |  |                     |                                  |  |                        |            |
| œ                | Qbs             |              |                                   |  |                     |                                  |  |                        |            |
| ŵ                | Debugger        |              |                                   |  |                     |                                  |  |                        |            |
| 1                | Designer        | Nar          | me: CMake                         |  |                     |                                  |  |                        |            |
| Ş                | Python          | Pat          | h: /opt/fus-i                     | mx-wayland/5.                              | 4-zeus/sysroot      | s/x86_64-po                      | kysdk-linux/usr/bin/cmake                        | Browse                 |            |
| E                | Analyzer        |              | p file:                           |  |                     |                                  |  | Browse                 |            |
| P                | Version Control |              | Autorun CMake<br>Auto-create buil |  |                     |                                  |  |                        |            |
| D                | Devices         |              | nuto-oreate Dull                  | a anectones                                |                     |                                  |  |                        |            |
|                  |                 | ] [          |                                   |  |                     |                                  |  | Apply O                | ancel      |

Figure 12: Kits – Cmake

After that select *CMake* and *Add*. Setup *Name* and Path and *Apply* the settings.

#### Examples:

#### QT5:

# (Buildroot) /home/developer/fsimx6-B2020.04/build/buildroot-2019.05.3-fsimx6-B2020.04/output/host/usr/bin/cmake (Yocto/.MX8) /opt/fus-imx-wayland/5.4-zeus/sysroots/x86\_64-pokysdklinux/usr/bin/cmake (Yocto/i.MX6)



#### /opt/poky/2.4.2/sysroots/x86 64-pokysdk-linux/usr/bin/cmake

#### QT6:

# (Yocto/i.MX8) /opt/fus-imx-wayland/5.15-kirkstone/sysroots/x86\_64-pokysdklinux/usr/bin/cmake

Now everything is setup and the Kit can be created. Select *Kits* and *Add*. Setup the Kit (refer to figure 13).

#### Note:

If you are unable to find the file *cmake* in buildroot directory output/host/usr/bin, then you have to add the following package to your buildroot configuration and rebuild it, e.g. through *"make menuconfig"*:

host cmake, located in Host utilities/host cmake

|                   |                               |   | Q                            |  |  |
|-------------------|-------------------------------|---|------------------------------|--|--|
| Filter            | Kits                          |   |                              |  |  |
| FT Kits           | Kits Qt Versions Con          | npilers Debuggers CMake   |                              |  |  |
| Environment       | Name<br>Auto-detected         |   |                              |  |  |
| Text Editor       | ✓ Manual                      |   | Clone                        |  |  |
| FakeVim           | FS Cross Comp                 | ile Kit   | Remove                       |  |  |
|                   |                               |   | Make Default                 |  |  |
| •                 |                               |   | Settings Filter              |  |  |
| {} C++            |                               |   | Default Settings Filter      |  |  |
| A Quick           | Name:                         | FS Cross Compile Kit  |                              |  |  |
| ➤ Build & Run     | File system name:             |   |                              |  |  |
| 🖽 Qbs             | Device type:                  | Generic Linux Device  | •                            |  |  |
| 🏦 Debugger        | Device:                       | FS Embedded System  | ✓ Manage                     |  |  |
| 🖍 Designer        | Sysroot:                      |   | Browse                       |  |  |
| New Python        | Compiler:                     | C: BT   | ▼ Manage                     |  |  |
| Analyzer          | complier.                     | C++: Compiler   | ▼                            |  |  |
| Version Control   | Environment:                  | No changes to apply.  | Change                       |  |  |
| -                 | Debugger:                     | GDB   | ✓ Manage                     |  |  |
| -                 | Qt version:                   | Qt 5.15.0 (System)  | ✓ Manage                     |  |  |
| 🚡 Code Pasting    | Qt mkspec:                    |   |                              |  |  |
| 🗗 Language Client | Additional Qbs Profile Set in | gs:   | Change                       |  |  |
| A Testing         | CMake Tool:                   | CMake   | ✓ Manage                     |  |  |
|                   | CMake generator:              | <none> - Onix Makenies, Plation. <none>, rooiset. <none></none></none></none> | Change                       |  |  |
|                   | CMake Configuration:          | CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable:Cxx}; CMAKE_C_COMPILER:STRING | a=%{Co Change                |  |  |
|                   |                               | <b>√</b> Apply  | ● <u>C</u> ancel <b>J</b> OK |  |  |
|                   |                               | ( , thu   |                              |  |  |

Figure 13: Build & Run - Kits

Select *Apply* and *OK* to finish the setup.



#### QT5

After everything is setup, you can create a new project. From File select New File or Project... Select Application and QT Widgets Application and click Choose...

#### QT6

After everything is setup, you can create a new project. From File select New Project... Select Application (Qt) and Qt Widgets Application and click Choose...

| Choose a template:   |                           |
|--|---------------------------|
| shoose a template.   | All Templates             |
| Projects       Qt Widgets Application         Application       Q t Console Application         Library       Qt Quick Application         Other Project       Qt Quick Controls 2 Application         Non-Qt Project       Qt Quick Controls Application         Import Project       Qt Quick Controls Application         Files and Classes       Qt Labs Controls Application         Qt       GLSL         General       Java         Python       Python | application if available. |

Figure 14: New File or Project



Choose a name of the project and the location of the project. Press *Next* >.

| ■                          | Qt Widgets Application X  |
|----------------------------|---|
| ∘≫ Location                | Introduction and Project Location   |
| Kits<br>Details<br>Summary | This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and<br>includes an empty widget. |
|                            | Name:       QT_Test_App          Create in:       /home/developer         Browse       Use as default project location                        |
|                            | <u>N</u> ext > Cancel   |

Figure 15: Introduction and Project Location



# QT5:

Select the Kits which you want to use and press Next >.

| ■                          | Qt Widgets Application  | ×         |
|----------------------------|---|-----------|
| Location                   | Kit Selection   |           |
| ∘ <b>≫ Kits</b><br>Details | Qt Creator can use the following kits for project <b>QT_Test_App</b> :<br>Select all kits |           |
| Summary                    | 🕑 🖵 Desktop   | Details 🔻 |
|                            | Scross Compile Kit for ARM  | Details 🔻 |
|                            |   |           |
|                            | < <u>B</u> ack <u>N</u> ext   | > Cancel  |
|                            |   |           |

Figure 16: Kit Selection

Choose class information's and press Next >.

| =                             |                        | Qt Widgets Application X  |
|-------------------------------|------------------------|---|
| Location                      | Class Info             | rmation   |
| Kits                          | Specify basic inf      | ormation about the classes for which you want to generate skeleton source code files. |
| ∘ <b>∂ Details</b><br>Summary | <u>C</u> lass name:    | MainWindow  |
|                               | <u>B</u> ase class:    | QMainWindow   |
|                               | <u>H</u> eader file:   | mainwindow.h  |
|                               | <u>S</u> ource file:   | mainwindow.cpp  |
|                               | <u>G</u> enerate form: |   |
|                               | <u>F</u> orm file:     | mainwindow.ui   |
|                               |                        |   |
|                               |                        |   |
|                               |                        |   |
|                               |                        |   |
|                               |                        | < <u>B</u> ack <u>N</u> ext > Cancel  |

Figure 17: Class Information



Select Project Management and click Finish.

| =               |  | Qt Widgets Application |               | ×         |  |
|-----------------|--|------------------------|---------------|-----------|--|
| Location        | Project Management   |                        |               |           |  |
| Kits<br>Details | Add as a subproject to project:  | <none></none>          |               | A<br>V    |  |
| >> Summary      | Add to <u>v</u> ersion control:  | <none></none>          | *<br>*        | Configure |  |
|                 |  |                        |               |           |  |
|                 |  |                        |               |           |  |
|                 |  |                        |               |           |  |
|                 | Files to be added in   |                        |               |           |  |
|                 | /home/developer/QT_Test_   | App:                   |               |           |  |
|                 | QT_Test_App.pro<br>main.cpp<br>mainwindow.cpp<br>mainwindow.h<br>mainwindow.ui |                        |               |           |  |
|                 |  |                        | < Back Finish | Cancel    |  |

Figure 18: Project Management



# **QT6**:

#### Choose *qmake* and press Next >.

| 0   |               | Qt Widgets Application — Qt Creator | ×  |
|---|---------------|-------------------------------------|----|
| Location  | Define Bu     | uild System                         |    |
| <ul> <li>Build System</li> <li>Details</li> <li>Translation</li> <li>Kits</li> <li>Summary</li> </ul> | Build system: | qmake                               | •  |
|   |               | < <u>B</u> ack <u>N</u> ext > Cance | el |

Figure 19: Define Build System



Choose class information's and press *Next* >.

| 0  | Qt Widgets Application — Qt Creator  | × |
|--|--|---|
| Location<br>Build System<br>P Details<br>Translation | Class Information<br>Specify basic information about the classes for which you want to generate skeleton<br>source code files. |   |
| Kits   | Class name: MainWindowS<br>Base class: QMainWindow   | ] |
|  | Header file: mainwindows.h<br>Source file: mainwindows.cpp<br>Generate form  | ] |
|  | Form file: mainwindows.ui  |   |
|  | < Back Next > Cancel   | _ |
|  | < <u>B</u> ack <u>N</u> ext > Cancel   |   |

Figure 20: Class Information QT6



| 0  |                   | Qt Widgets Application — Qt Creator  | $(\mathbf{x})$ |  |
|--|-------------------|--------------------------------------|----------------|--|
| Location   | Translatio        | n File                               |                |  |
| Build SystemIf you plan to provide translations for your project's user interface via the Qt Linguist tool, pleaseDetailsselect a language here. A corresponding translation (.ts) file will be generated for you. |                   |                                      |                |  |
| 🔊 Translation  | Language:         | <none></none>                        | •              |  |
| Kits<br>Summary  | Translation file: | <none></none>                        |                |  |
|  |                   | < <u>B</u> ack <u>N</u> ext > Cancel |                |  |
|  |                   | Figure 21: Translation File          |                |  |

Choose your Translation File or leave it blank. Press Next >.



| 0                       | Qt Widgets Application — Qt Creator                                       | ×         |  |  |  |
|-------------------------|---|-----------|--|--|--|
| Location                | Kit Selection The following kits can be used for project QT_Test_App_Qt6: |           |  |  |  |
| Build System<br>Details |   |           |  |  |  |
| Translation<br>• Kits   | Type to filter kits by name<br>✓ Select all kits                          |           |  |  |  |
| Summary                 | ✓   | Details 👻 |  |  |  |
|                         | ✓   | Details 👻 |  |  |  |
|                         | < <u>B</u> ack  | > Cancel  |  |  |  |
|                         |   |           |  |  |  |

Select the Kits which you want to use and press Next >.

Figure 22: Kit Selection QT6



#### Click Finish.

|                        | Qt Widgets   | Application — Qt Creator |                               | ×         |  |  |  |
|------------------------|--|--------------------------|-------------------------------|-----------|--|--|--|
| Location               | Project Management   |                          |                               |           |  |  |  |
| Build System           | Add as a subproject to project:  | <none> **</none>         |                               |           |  |  |  |
| Details<br>Translation | Add to <u>v</u> ersion control:  | <none></none>            | •                             | Configure |  |  |  |
| Kits<br>∞≫ Summary     |  |                          |                               |           |  |  |  |
|                        | Files to be added in<br>/home/developer/QT_Test                                  | _App_Qt6:                |                               |           |  |  |  |
|                        | CMakeLists.txt<br>main.cpp<br>mainwindows.cpp<br>mainwindows.h<br>mainwindows.ui |                          |                               |           |  |  |  |
|                        |  |                          | < <u>B</u> ack <u>F</u> inish | Cancel    |  |  |  |
|                        |  | Droject Management OTC   |                               |           |  |  |  |

Figure 23: Project Management QT6

Now the project is successfully created.



## 4.1 Create hello world app

Open the folder Forms on the left side in project view. Double click the mainwindow.ui.

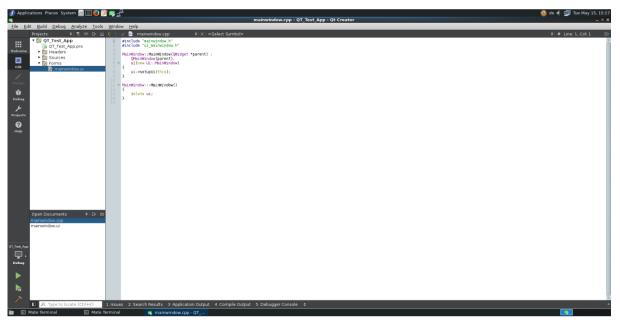


Figure 24: Project file - mainwindow.cpp

Now move 2 Push Buttons to the Main Window and move one Line Edit (refer to figure 20). After that change the name of the buttons with a double click on the button.



Figure 25: Project file - mainwindow.ui

QT5 Application Development



After that right click the "Set" button and select *Go to slot*.... Then select the signal *clicked()* and press *OK*.

| Type Here  |   |
|--|---|
| Clear  | Set   |
|  |   |
| 👦 Go t   | o slot X  |
| Select signal  |   |
| clicked()<br>clicked(bool)<br>pressed()<br>released()<br>toggled(bool) | QAbstractButton<br>QAbstractButton<br>QAbstractButton<br>QAbstractButton<br>QAbstractButton |
|  | <u>C</u> ancel <u>O</u> K   |

Figure 26: Select signal

The view changes to the cpp file. It automatically creates a function called *void MainWindow::on\_pushButton\_clicked()*. Now add the following line in the created function.

```
...
void MainWindows::on_pushButton_clicked()
{
    ui->lineEdit->setText("Hello World!");
}
```



Now double click mainwindow.ui again. After that right click the "Clear" button and select *Go* to slot.... Then select the signal clicked() and press *OK*. After that the view changes to the cpp file again. It automatically creates a function called *void MainWindow::on\_pushButton\_2\_clicked()*. Now add the following line in the created function.

```
...
void MainWindows::on_pushButton_2_clicked()
{
    ui->lineEdit->setText(``");
}
```



Select Build in the Toolbar and press Build Project "QT\_Test\_App". Otherwise you can click

the build icon in the left toolbar.

If you have unsaved files then the system ask you what to do with these unsaved changes. We press Save All, because we want to save these modifications.

| <b>Q</b>                       | Save Changes                             | × |  |  |
|--------------------------------|--|---|--|--|
| The f                          | ollowing files have unsaved changes:     |   |  |  |
| Б                              | mainwindow.h /home/developer/QT_Test_App |   |  |  |
|                                |  |   |  |  |
|                                |  |   |  |  |
|                                |  |   |  |  |
|                                |  |   |  |  |
| Always save files before build |  |   |  |  |
|                                | Do not Save <u>C</u> ancel Save All      |   |  |  |

Figure 27: Save changes

To be sure everything is successfully built select at the bottom bar *4 Compile Output* and have a look at the output message.



## 5.1 Desktop Run

After everything is successfully built, select *Projects* on the left sidebar. After that select *Run* in the Desktop Kit.

If you don't see the kit in the tab, select Add Kit and add your corresponding Kit.

| Manage Kits                | Run Settings  |   |
|----------------------------|---|---|
| Import Existing Build      | Deployment Method: Deploy Configuration  Add  Remove Rename                             |   |
| Active Project             |   |   |
| yoctoaa                    | No Deploy Steps<br>Add Deploy Step *  |   |
|                            |   |   |
| Build & Run                | Run   |   |
| 🖵 Desktop 🥕 Build          | Run configuration: yoctoaa 🔹 Add 🔹 Remove Rename  |   |
| Run                        |   |   |
| FS QT5 FS QT5 Yocto-Fin    | Executable: /home/developer/build-yoctoaa-Desktop-Debug/yoctoaa Command line arguments: |   |
| ild 🎢                      |   |   |
| 🕨 Run                      | Working directory: /home/developer/build-yoctoaa-Desktop-Debug Browse 5 Run in terminal | , |
| Project Settings           |   |   |
| Editor                     | ✓ Add build library search path to LD_LIBRARY_PATH                                      |   |
| Code Style<br>Dependencies |   |   |
| Clang Static Analyzer      | Debugger Settings   |   |
|                            | ✓ Enable C++  |   |
|                            | Enable QML What are the prerequisites?  |   |
|                            | Valgrind Settings   |   |

Figure 28: Project - Desktop Run

Then select *Build* on the toolbar and click *Run*. Otherwise you can click the Run icon on the left toolbar.



| MainWindow   | _ = × |
|--------------|-------|
| E.           |       |
| Hello World! |       |
| Clear Set    |       |
|              |       |
|              |       |
|              |       |
|              |       |
|              |       |

Figure 29: Application

Now the application will be executed on your development machine. Test the app by clicking the *Set* and the *Clear* buttons.

You can click the x in the right corner of the application to stop it.



## 5.2 FS Cross Compile Kit for ARM Run

First of all we have to setup something in our project. Select .pro file in my case QT\_Test\_App.pro and add at the end following lines:

```
...
linux-* {
# Buildroot
    target.path = /root
# Yocto
    target.path = /home/root
    INSTALLS += target
}
```



Now save the changes and select *Projects* on the left toolbar.

To run the compiled example for the ARM system, select Build in the FS Cross Compile Kit for ARM. Then select *Browse…* and create a folder and set the Build directory.

| Welcome        | Manage Kits<br>Import Existing Build  | $\sim$ | Build Settings<br>Edit build configuration: Debug - Add - Remove Rename            |           |
|----------------|---------------------------------------|--------|--|-----------|
| Edit           |                                       |        | General  |           |
| <b>D</b> esign | Active Project                        |        | Shadow build: 🗸  |           |
|                |                                       |        | Build directory: /home/developer/build-yoctoaa-FS_QT5_Yocto_Fin-Debug/debugproj    | Browse    |
| Debug          | Build & Run                           |        | Build Steps  |           |
| Switch t       | o Projects Build                      |        | <b>qmake:</b> qmake yoctoaa.pro -spec linux-oe-g++ CONFIG+=debug CONFIG+=qml_debug | Details 🔻 |
| Help           | Run                                   |        | Make: make in /home/developer/build-yoctoaa-FS_QT5_Yocto_Fin-Debug/debugproj       | Details 🔻 |
|                | FS QT5 Yocto-Fin Build                |        | Add Build Step *   |           |
|                | Run                                   |        | Clean Steps  |           |
|                | Project Settings                      |        | Make: make clean in /home/developer/build-yoctoaa-FS_QT5_Yocto_Fin-Debug/debugproj | Details 🔻 |
|                | Editor<br>Code Style                  |        | Add Clean Step ~   |           |
|                | Dependencies<br>Clang Static Analyzer |        | Build Environment  |           |
|                |                                       |        | Use System Environment   | Details 🔻 |
|                |                                       |        |  |           |
|                |                                       |        |  |           |

Figure 30: Project - Build FS Cross Compile Kit for ARM

Then build the system by clicking *Build* in the toolbar and select *Build Project* "QT\_Test\_App". Otherwise you can click the build icon in the left toolbar.



Then select *Run* in the FS Cross Compile Kit for ARM. Normally it should automatically add the files to deploy with the correct local path and the remote directory.

|                       | Manage Kits   | ≽ Run Settings  |  |
|-----------------------|---|---|--|
| Welcome<br>E<br>Edit  | Import Existing Build                               | Deployment Method: Deploy to Remote Linux Host      Add      Remove Rename  |  |
| Design                | Active Project                                      | Files to deploy:  |  |
| û<br>Debug            | Build & Run   | Local File Path Remote Directory<br>/home/developer/build-yoctoaa-FS_QT5_Yocto_Fin-Debug/debugproj/yoctoaa /home/root |  |
| <b>پر</b><br>Projects |   |   |  |
| •                     | <ul> <li>Build</li> <li>Run</li> </ul>              | Check for free disk space Deta  |  |
| Help                  | FS QT5  | Kill current application instance   |  |
|                       | FS QT5 Yocto-Fin Build                              | Upload files via SFTP Deta  |  |
|                       | ▶ Run   | Add Deploy Step -   |  |
|                       | Project Settings                                    |   |  |
|                       | Editor  | Run   |  |
|                       | Code Style<br>Dependencies<br>Clang Static Analyzer | Run configuration: yoctoaa (on Remote Device) 🔹 Add 👻 Remove Rename   |  |
|                       | ciong state Analyzer                                | Executable on host: /home/developer/build-yoctoaa-F5_QT5_Yocto_Fin-Debug/debugproj/yoctoaa                            |  |
|                       |   | Executable on device: /home/root/yoctoaa  |  |
|                       |   | Alternate executable on device: Use this command inste  |  |
|                       |   |   |  |
| yoctoaa               |   |   |  |
| Debug                 |   |   |  |
|                       |   | Debugger Settings   |  |

Figure 31: Project - Run FS Cross Compile Kit for ARM

Now everything is setup and you can run your application on your embedded device. Click *Build* in the Toolbar and select *Run*. Otherwise you can click the run icon on the left toolbar.

Now the application is running on your system.

## 5.3 Debug build

To debug the application you have to choose *Debug* in the toolbar, *Start Debugging* and *Start Debugging* again instead of Build/Run. Otherwise you can click the debug icon on the

left toolbar. 🌆

Now you can debug your application on your system.

#### Note:

If you are using a backend like wayland / X11 you have to fetch the environment variables from the embedded device in QT-Creator. Select Projects on the left sidebar. After that select Run in the FS Cross Compile Kit for ARM. Now scroll down to Run Environment and press Fetch Device Environment. Now you should see the environment of the embedded device. It is necessary that the SSH connection works.

To setup debug points select your c file which you want to debug and left click on the line in your program, where you want to set a debug point. Click Debug in the toolbar to see the different step options.



|                  | cations PlacesSystem 📄 🔛 🥹 🗒          |   | ie 🚸 🚅 Wed May 16, 10:22 |
|------------------|---------------------------------------|---|--------------------------|
| <b>85</b>        |                                       | mainwindow.cpp - QT_Test_App - Qt Creator   | _ ¤ X                    |
| <u>File</u> Ec   | it <u>Build Debug</u> Analyze Tools   |   |                          |
|                  | Projects ≑ T. ⊕ ⊟+ 🗆                  |   | \$ # Line: 18, Col: 5 ⊟+ |
|                  | QT_Test_App<br>QT_Test_App.pro        | arino Lude "mainwindew.h"<br>a wino Lude "uia mainwindew.h"   | Ô                        |
| Welcome          | ► M Headers                           |   |                          |
| E                | Sources                               | HainWindow.WainHindow.UWiiget *parent) :<br>OWainWindow.Wainerent).   |                          |
| EdR              | 🔻 🔯 Forms                             | 0 v u::/Nainttindow)  |                          |
| COR              | 🃝 mainwindow.ui                       | 8 uit->setupli (this);  |                          |
| 1                |                                       |   |                          |
| Design           |                                       | <pre>ii * HainWindow:-\HainWindow()</pre>   |                          |
| ŵ                |                                       | 12 C delete Mi  |                          |
| Debug            |                                       | 3   |                          |
| بر               |                                       | v void HainWindow::on_pushButton_clicked()  |                          |
| Projects         |                                       | <pre>blb {</pre>  |                          |
|                  |                                       |   |                          |
| <b>O</b><br>Helo | -                                     | v void MainWindow::on pushButton 2 clicked()  |                          |
| neip             |                                       | ( ui->lineGit->setText(");  |                          |
|                  |                                       | and selfective selfective (   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  |                                       |   |                          |
|                  | Open Documents 💠 🗄 🗖                  |   |                          |
|                  | mainwindow.cpp                        |   |                          |
|                  | mainwindow.h                          |   |                          |
|                  | mainwindow.ui                         | Application Output   J+ < > h 🔳   h + -   |                          |
|                  | moc_mainwindow.cpp<br>OT Test App.oro |   | ^ 🗆                      |
| QT_Test_App      |                                       | QT_Test_App 🗋 QT_Test_App (on Remote Device) 📮 QT_Test_App (on Remote Device) 🖸   |                          |
| μ,               |                                       | Debugging starts<br>Listening on port 10000   | *                        |
| Debug            |                                       | Renote debugging from host 10.0.0.128   |                          |
| ,                |                                       | Process /roof/01 Test App created; pid = 444<br>File '/home/developeri/size-03.0/build/buildroot-2016.05-fsize-0-9.0/buildroot-linux-anueabih/sysreot/lib/libstdc++.so.6.0.21-adb.av" auto-loadina has been declined by your 'auto- | load safe-path' set to   |
|                  |                                       | "Satebugdir:Sdatadir/auto-load".<br>UGclFSVirIntegration vill set environment variable FB MLTI BUFFER-2 to enable double buffering and vaync.   |                          |
|                  |                                       | If this is not desired, you can override this via: export OT EGLFS INX6 NO FB MLITI BUFFER-1  |                          |
| 4                |                                       | Unable to query physical screen size, defaulting to 100 dps.<br>To overrade, est qT OPE ALERS PMYSICLA MOTH and qT OPE ALERS PMYSICLA HEIGHT (in millimeters).  |                          |
| ~                |                                       |   | *                        |
|                  | R. Type to locate (Ctrl+K) Mai        | Mate Terminal : Search Results 3 Application Output 4 Complie Output 5 Debugger Console 🗢   |                          |
| <b>a</b> 🗉       | Mate Terminal 🔲 (Mate '               | ste Terminal) 📪 mainwindow.cpp - QT   |                          |

Figure 32: Debug Application



# 6 Build CMAKE Projects

This guide explains how to build CMake-based Qt projects using the Yocto populated SDK within Qt Creator.

## 6.1 Background

Unlike qmake, where references to dependent libraries are automatically known through the toolchain, CMake does not include these references by default. However, the Yocto SDK provides a CMake toolchain file that contains all necessary paths and settings. This file can easily be integrated into your project's CMAKE configuration.

## 6.2 Prerequisites

- Installed and configured Yocto SDK (populated SDK)
- Qt Creator
- A CMake -based Qt project

## 6.3 Step-by-Step Instructions

Carry out the following steps:

- 1. Import the Qt Project: Launch Qt Creator and import your existing Qt project based on CMAKE.
- 2. Configure the Toolchain Settings Go to the following settings in Qt Creator: Menu: Projects Selection: CMake Tab: Initial Configuration
- **3.** Add the Following Entries Replace <yocto-version> with the appropriate version of your Yocto SDK.

```
CMAKE_FIND_ROOT_PATH = /opt/fus-imx-wayland/<yocto-
version>/sysroots/x86_64-pokysdk-linux/
CMAKE_TOOLCHAIN_FILE = /opt/fus-imx-wayland/<yocto-
version>/sysroots/x86_64-pokysdk-
linux/usr/share/cmake/OEToolchainConfig.cmake
```



# 7 Appendix

## List of Figures

| Figure 1: QT Creator                                    | 2  |
|---|----|
| Figure 2: SSH connection terminal                       | 6  |
| Figure 3: Options                                       | 7  |
| Figure 4: Options - Devices                             | 7  |
| Figure 5: Add device configuration                      | 8  |
| Figure 6: Setup device connection                       | 8  |
| Figure 7: Device test                                   | 9  |
| Figure 8: Options - Kits                                | 11 |
| Figure 9: Kits - Qt Versions                            | 12 |
| Figure 10:Kits - Compilers                              | 13 |
| Figure 11: Kits - Debuggers                             | 14 |
| Figure 12: Kits – Cmake                                 | 15 |
| Figure 13: Build & Run - Kits                           | 16 |
| Figure 14: New File or Project                          | 17 |
| Figure 15: Introduction and Project Location            | 18 |
| Figure 16: Kit Selection                                | 19 |
| Figure 17: Class Information                            | 19 |
| Figure 18: Project Management                           | 20 |
| Figure 19: Define Build System                          | 21 |
| Figure 20: Class Information QT6                        | 22 |
| Figure 21: Translation File                             | 23 |
| Figure 22: Kit Selection QT6                            | 24 |
| Figure 23: Project Management QT6                       | 25 |
| Figure 24: Project file - mainwindow.cpp                | 26 |
| Figure 25: Project file - mainwindow.ui                 | 26 |
| Figure 26: Select signal                                | 27 |
| Figure 27: Save changes                                 | 29 |
| Figure 28: Project - Desktop Run                        | 30 |
| Figure 29: Application                                  | 31 |
| Figure 30: Project - Build FS Cross Compile Kit for ARM | 32 |
| Figure 31: Project - Run FS Cross Compile Kit for ARM   | 33 |
| Figure 32: Debug Application                            | 34 |



## **Important Notice**

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. F&S Elektronik Systeme assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained in this documentation.

F&S Elektronik Systeme reserves the right to make changes in its products or product specifications or product documentation with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

F&S Elektronik Systeme makes no warranty or guarantee regarding the suitability of its products for any particular purpose, nor does F&S Elektronik Systeme assume any liability arising out of the documentation or use of any product and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Products are not designed, intended, or authorised for use as components in systems intended for applications intended to support or sustain life, or for any other application in which the failure of the product from F&S Elektronik Systeme could create a situation where personal injury or death may occur. Should the Buyer purchase or use a F&S Elektronik Systeme product for any such unintended or unauthorised application, the Buyer shall indemnify and hold F&S Elektronik Systeme and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorised use, even if such claim alleges that F&S Elektronik Systeme was negligent regarding the design or manufacture of said product.

