MYD-YT113X_Qt Application Notes Qt environment installation and MEasy-HMI compilation

File Status:		
	FILE ID:	MYIR-MYD-YT113X-SW-AN-ZH-L5.4.61
[ ] Draft	**VERSION:**	V1.0[doc]
[√] Release	**VERSION:**	Nico
	CREATED:	2023-09-28
	UPDATED:	2023-09-28

Revision History

VERSION	AUTHOR	PARTICIPANT	DATE	DESCRIPTION
V1.0	Nico	--	20230928	Initial Version

CONTENT

1. Overview

QT is a cross-platform graphical application development framework that is used in different sizes of devices and platforms, and QT also provides different copyrighted versions for users to choose from. For Qt application development, it is recommended to use the QtCreator IDE, which allows you to develop Qt applications under Linux PC and automate the cross-compilation to the ARM architecture of the development board.

This chapter uses the SDK tool built by MYIR as a cross-compilation tool with QtCreator to quickly develop graphical applications.

2. Hardware resources

None

3. Software Resources

- Ubuntu18.04 desktop 或 Ubuntu20.04 desktop（推荐）
- Qtcreator 4.12
- gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz
- MYIR-MEasy-HMI 2.X

4. Environmental preparations

Need to install ubuntu desktop system, due to my original development environment for ubuntu18.04desktop, the subsequent are in ubuntu18.04 desktop version of the operation. Please install the desktop system by yourself

You need to install the compilation cross-toolchain provided by MYIR, path: "03-Tools\Complie Toolchain\gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz"

Get the mxapp2.X source code from the development SDK package provided by MYIR at "04-Sources/mxapp2.tar.gz".

Making a QT compilation chain

Before creating a QT compilation chain, be sure to complete the "MYD-YT113X_Linux Software Development Guide V1.1" document in section 3.4 Compiling QT steps.

First of all, the "03-Tools\Complie Toolchain\gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz" compilation chain unpacked, this article in the following path to unpack, the user please unpack according to their own practical situation

```
sur@myir:~/opt$ tar -xf gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi.tar.xz
```

After compiling QT, the "Qt_5.12.5" directory will be created in the following directory

```
sur@myir:~/auto-t113x-linux/platform/framework/qt/qt-everywhere-src-5.12.5/
Qt_5.12.5
```

Copy the directory "Qt_5.12.5" to the directory of the compilation chain extracted above.

```
sur@myir:~/auto-t113x-linux/platform/framework/qt/qt-everywhere-src-5.12.5$
cp -r Qt_5.12.5/ ~/opt/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/
```

Then create a "qt.conf" file in the "gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/ Qt_5.12.5/" directory and add the following contents

```
sur@myir:~/opt/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/Qt_5.12.5$
vim qt.conf

[Paths]
Documentation=../../Docs/Qt-5.12.2
Examples=../../Examples/Qt-5.12.2
```

```
Sysroot=../arm-buildroot-linux-gnueabi/sysroot
Prefix=..
```

last step，Since only the T113-S3 full image and the T113-I full image can install the QT environment

If it is a T113-S3 full image, copy the directory in the following path to the directory of the compilation chain you extracted earlier

```
sur@myir:~/auto-t113x-linux/out/t113/myir-image-yt113s3-emmc-full/longan/bu
ildroot/host$
cp -r arm-buildroot-linux-gnueabi/ ~/opt/gcc-linaro-5.3.1-2016.05-x86_64_arm-li
nux-gnueabi/Qt_5.12.5/
```

If it is a T113-I full image, copy the directory in the following path to the compilation chain directory that you extracted earlier

```
sur@myir:~/auto-t113x-linux/out/t113_i/myir-image-yt113i-full/longan/buildroot
/host$
cp -r arm-buildroot-linux-gnueabi/ ~/opt/gcc-linaro-5.3.1-2016.05-x86_64_arm-li
nux-gnueabi/Qt_5.12.5/
```

At this point the compilation chain is complete, follow the steps below to install the environment.

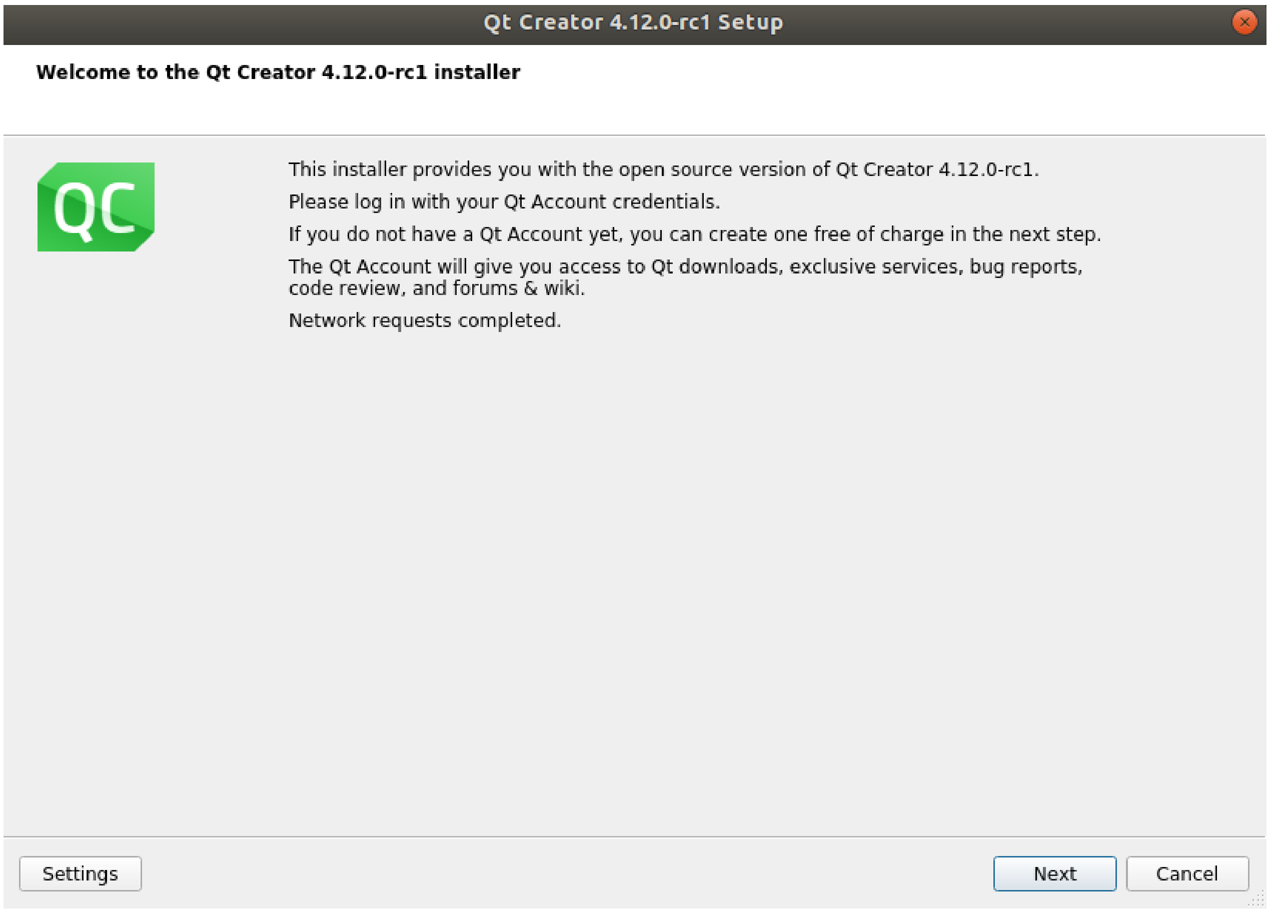
5. Operation steps

5.1. Installing Qt Creator

QT Creator official website download (take version 4.12-rc1 as an example)

Official Download Location： http://download.qt.io/development_releases/qtcreator/

The QtCreator installer is a binary program for ubuntu, which can be installed by executing it directly from a terminal in ubuntu . /qt-creator-opensource-linux-x86_64-4.12.0-rc1.run


Qt Creator 4.12.0-rc1 Setup

Welcome to the Qt Creator 4.12.0-rc1 installer

This installer provides you with the open source version of Qt Creator 4.12.0-rc1.

Please log in with your Qt Account credentials.

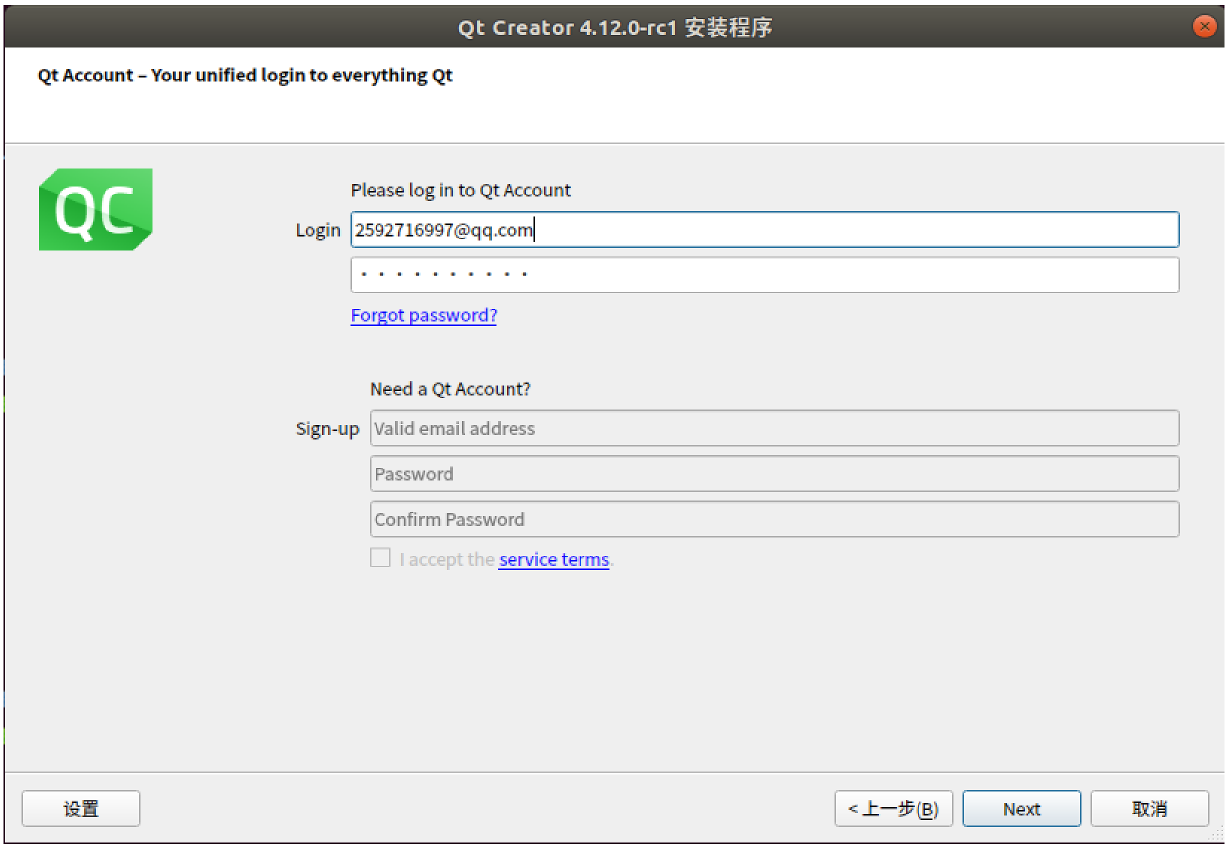
If you do not have a Qt Account yet, you can create one free of charge in the next step.

The Qt Account will give you access to Qt downloads, exclusive services, bug reports, code review, and forums & wiki.

Network requests completed.

Figure 5-1. QTCreator install

Click "next", enter your account password, you need to register your account on the official website https://login.qt.io/register


Qt Creator 4.12.0-rc1 安装程序

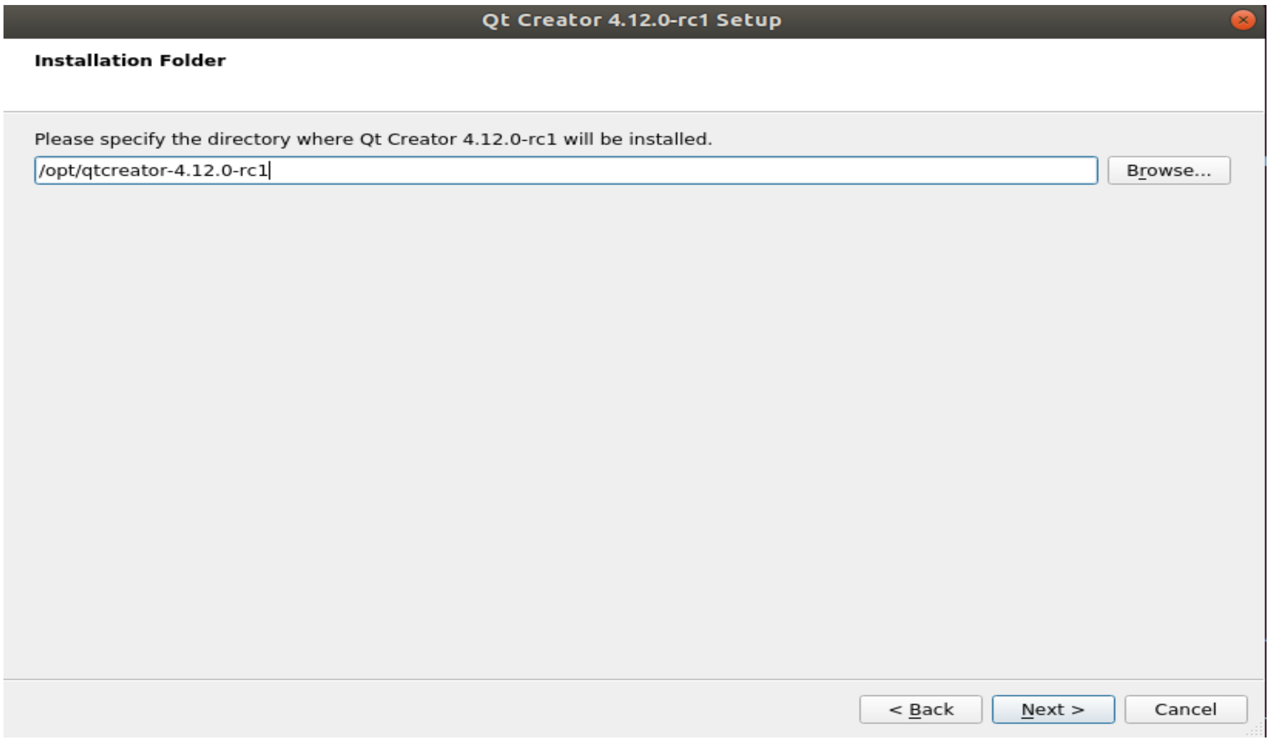
Qt Account – Your unified login to everything Qt

Please log in to Qt Account
Login 2592716997@qq.com
• • • • • • • • • •
Forgot password?

Need a Qt Account?
Sign-up Valid email address
Password
Confirm Password
I accept the service terms.

Figure 5-2. Login account password

Other configuration items please choose your own, configure the installation path

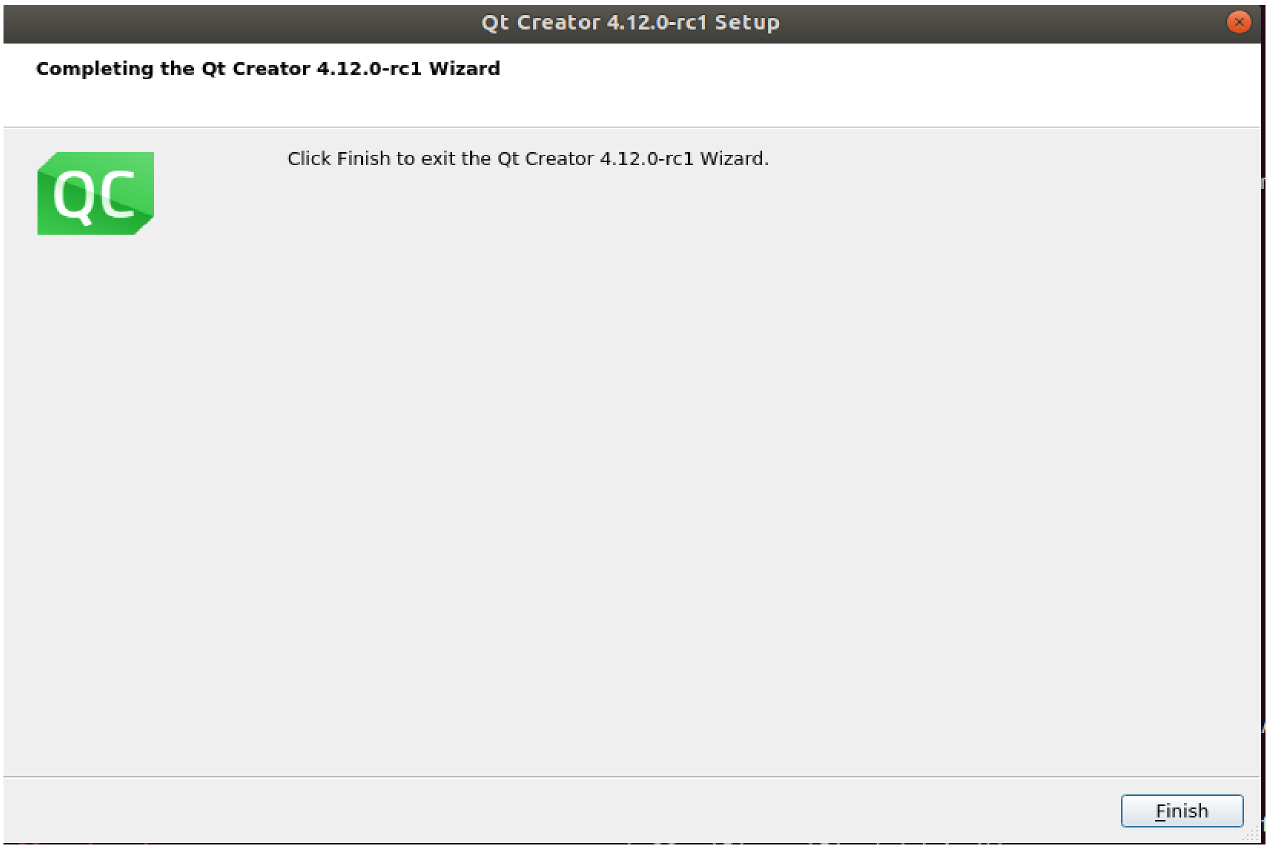

Qt Creator 4.12.0-rc1 Setup

Installation Folder

Please specify the directory where Qt Creator 4.12.0-rc1 will be installed.
/opt/qtcreator-4.12.0-rc1

Figure 5-3. Select installation path

Follow the instructions to complete the installation


Qt Creator 4.12.0-rc1 Setup

Completing the Qt Creator 4.12.0-rc1 Wizard

Click Finish to exit the Qt Creator 4.12.0-rc1 Wizard.

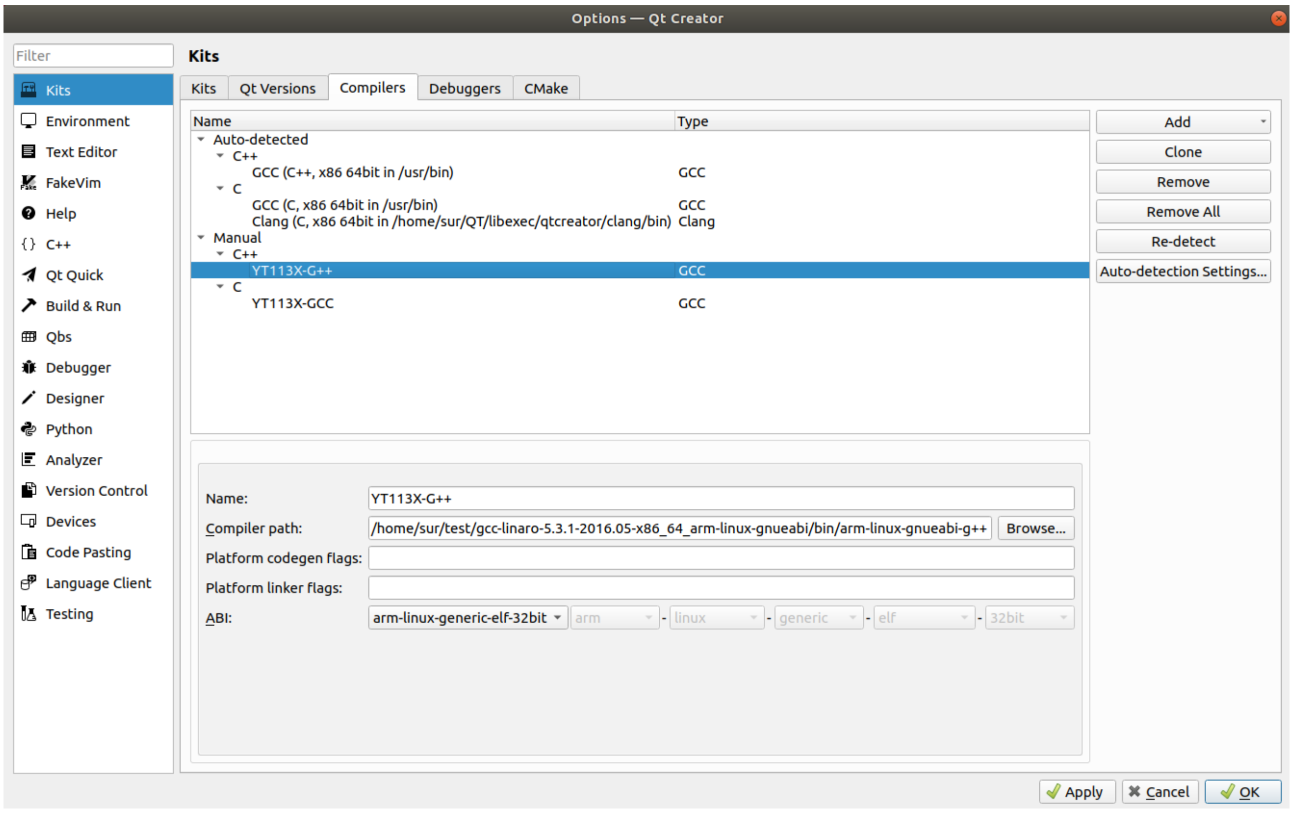
Figure 5-4. Installation completed

Qtcreator installation is complete and the development environment can be configured

5.2. Configuring the cross-compilation environment

To open QTCreator, please execute "qtcreator.sh" from the terminal to start QtCreator, refer to the following: /opt/qtcreator-4.12.0-rc1/bin/qtcreator.sh &

After running QtCreator, click Tools -> Options, the options dialog box appears, click Kits on the left, select the Compilers tab on the right.



Name	Type
Auto-detected	
C++	
GCC (C++, x86 64bit in /usr/bin)	GCC
C	
GCC (C, x86 64bit in /usr/bin)	GCC
Clang (C, x86 64bit in /home/sur/QT/libexec/qtcreator/clang/bin)	Clang
Manual	
C++	
YT113X-G++	GCC
C	
YT113X-GCC	GCC

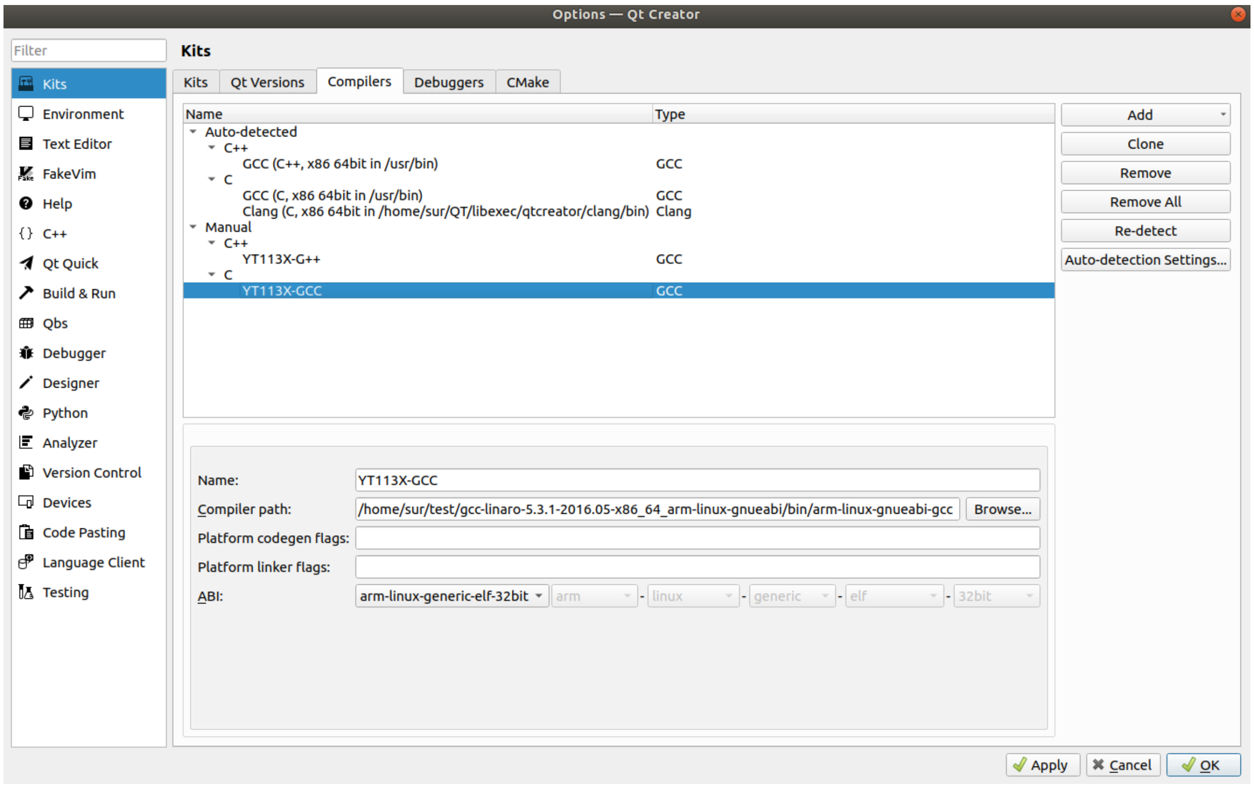
Name:	YT113X-G++
Compiler path:	/home/sur/test/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin/arm-linux-gnueabi-g++
Platform codegen flags:	
Platform linker flags:	
ABI:	arm-linux-generic-elf-32bit; arm - linux - generic - elf - 32bit

Figure 5-5. Select compiler

Click Add on the right side, after the drop-down list pops up, select G++, fill in the "Name" as "YT113X-G++", "Compiler path" and click "Browse..." next to it. button to select the path to arm-linux-gnueabi-g++, the path in the example is "/home/sur/test/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin"

Also select GCC and choose arm-linux-gnueabi-gcc compiler in the same way as G++.

After filling in the fields, click "Apply"

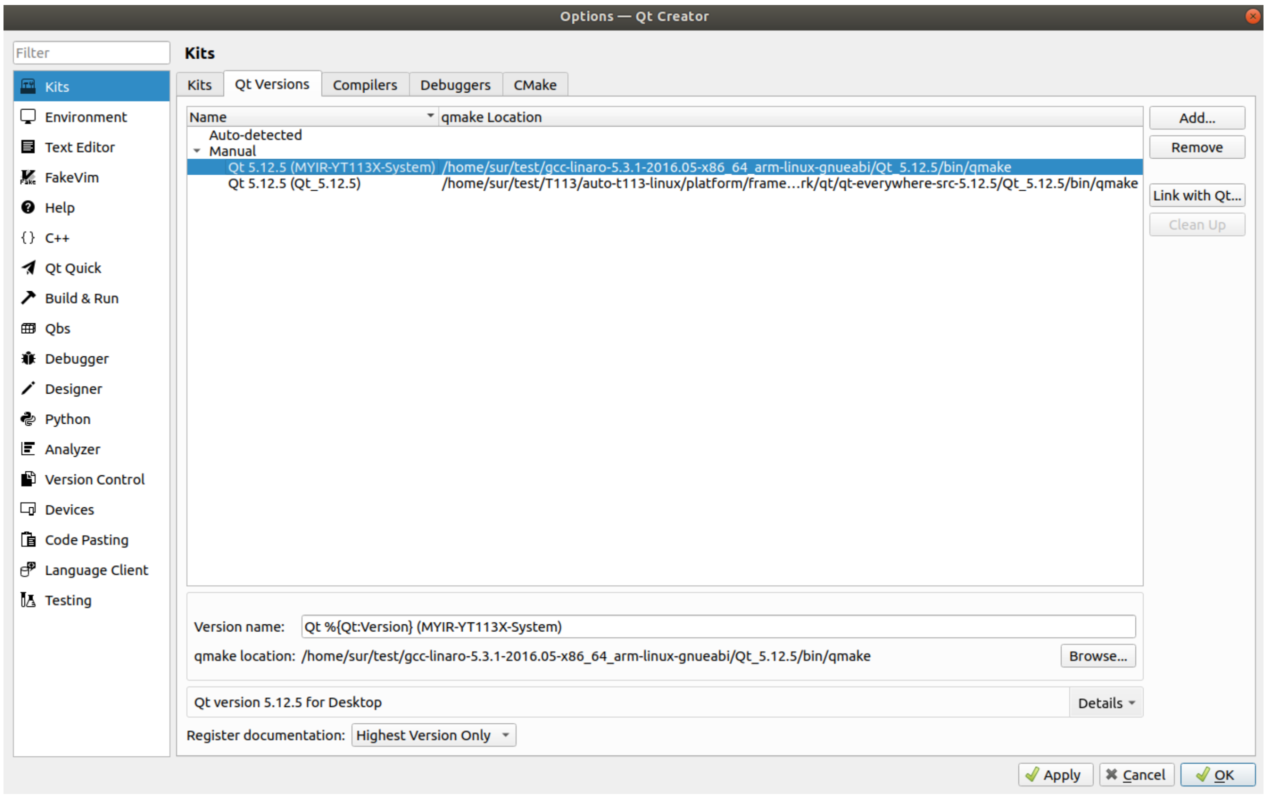


Name	Type
Auto-detected	
C++	
GCC (C++, x86 64bit in /usr/bin)	GCC
C	
GCC (C, x86 64bit in /usr/bin)	GCC
Clang (C, x86 64bit in /home/sur/QT/libexec/qtcreator/clang/bin)	Clang
Manual	
C++	
YT113X-G++	GCC
C	
YT113X-GCC	GCC

Name: YT113X-GCC
Compiler path: /home/sur/test/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/bin/arm-linux-gnueabi-gcc Browse...
Platform codegen flags:
Platform linker flags:
ABI: arm-linux-generic-elf-32bit arm - linux - generic - elf - 32bit

Figure 5-6. Select compiler path

Select the "Qt Version" tab, and click "Add..." on the right side. The qmake path selection dialog box will pop up, here is "/home/sur/test/gcc-linaro-5.3.1-2016.05- x86_64_arm-linux-gnueabi/Qt_5.12.5/bin/qmake" as an example. After selecting the "qmake" file, click the "Open" button. Version name" is changed to Qt %{Qt:Version} (MYIR-YT113X-System). Then click on the "Apply" button


Options — Qt Creator

Kits — Qt Versions

```
Name                                    qmake Location
  Auto-detected
▾ Manual
    Qt 5.12.5 (MYIR-YT113X-System)  /home/sur/test/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/Qt_5.12.5/bin/qmake
    Qt 5.12.5 (Qt_5.12.5)           /home/sur/test/T113/auto-t113-linux/platform/frame...rk/qt/qt-everywhere-src-5.12.5/Qt_5.12.5/bin/qmake
```

Version name: Qt %{Qt:Version} (MYIR-YT113X-System)

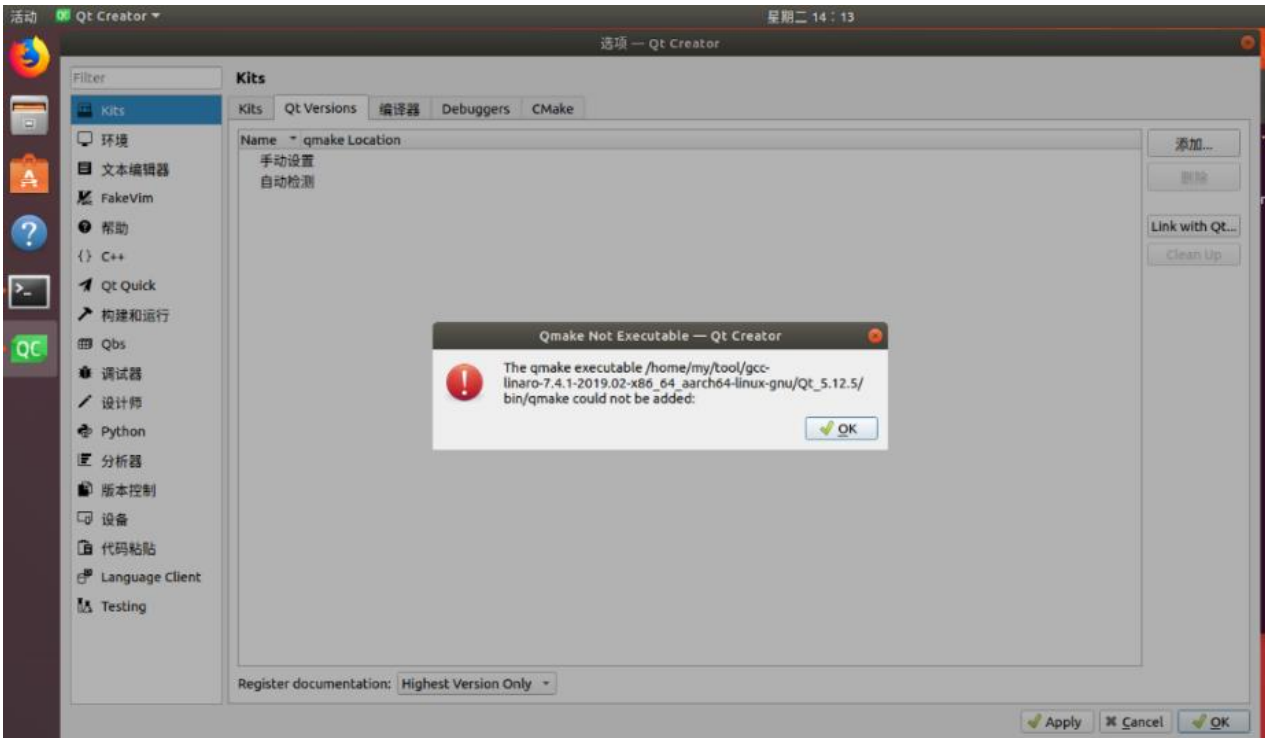
qmake location: /home/sur/test/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/Qt_5.12.5/bin/qmake

Qt version 5.12.5 for Desktop

Register documentation: Highest Version Only

Figure 5-7. Configuration version information

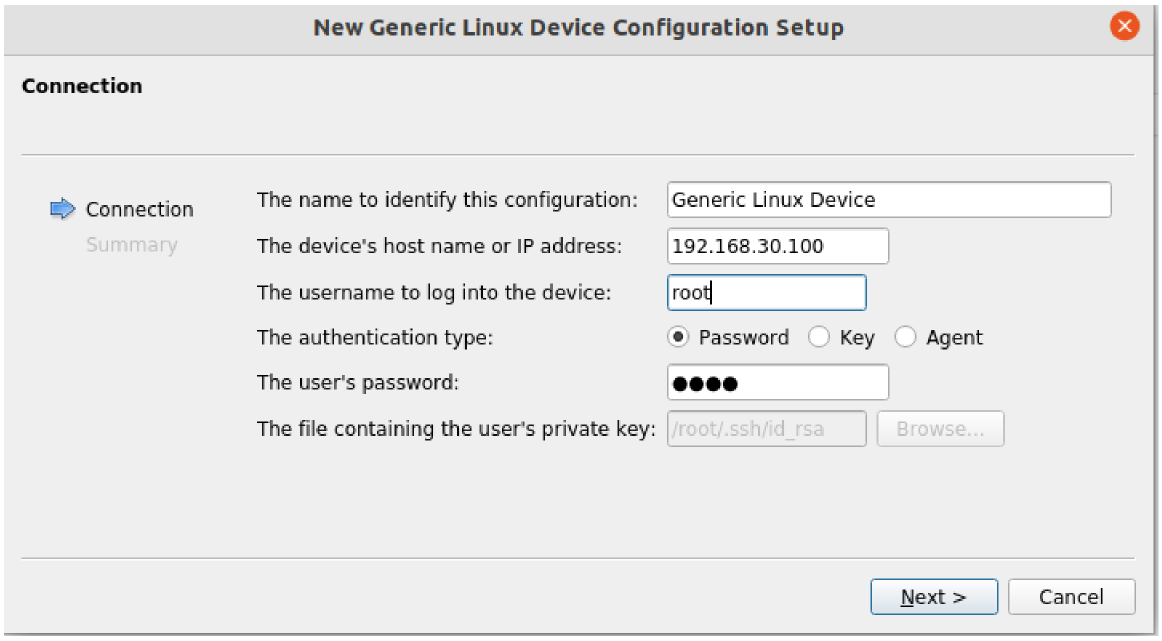
Note: If the following error occurs, the reason is that the GLIBC version needs to be greater than 2.28. Please update the glibc version or develop directly with Ubuntu 20 or above.


Qmake Not Executable — Qt Creator

The qmake executable /home/my/tool/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu/Qt_5.12.5/bin/qmake could not be added.

Figure 5-8. QmakeConfiguration error

Select "Device" on the left, click "Add..." on the right button, fill in the content "Name" as "MYD-YT113X-Board", "Host name" as the IP address of the development board (you can temporarily fill in any address), "Username" as "root", and then click "Next" to configure the next step. (This is optional)

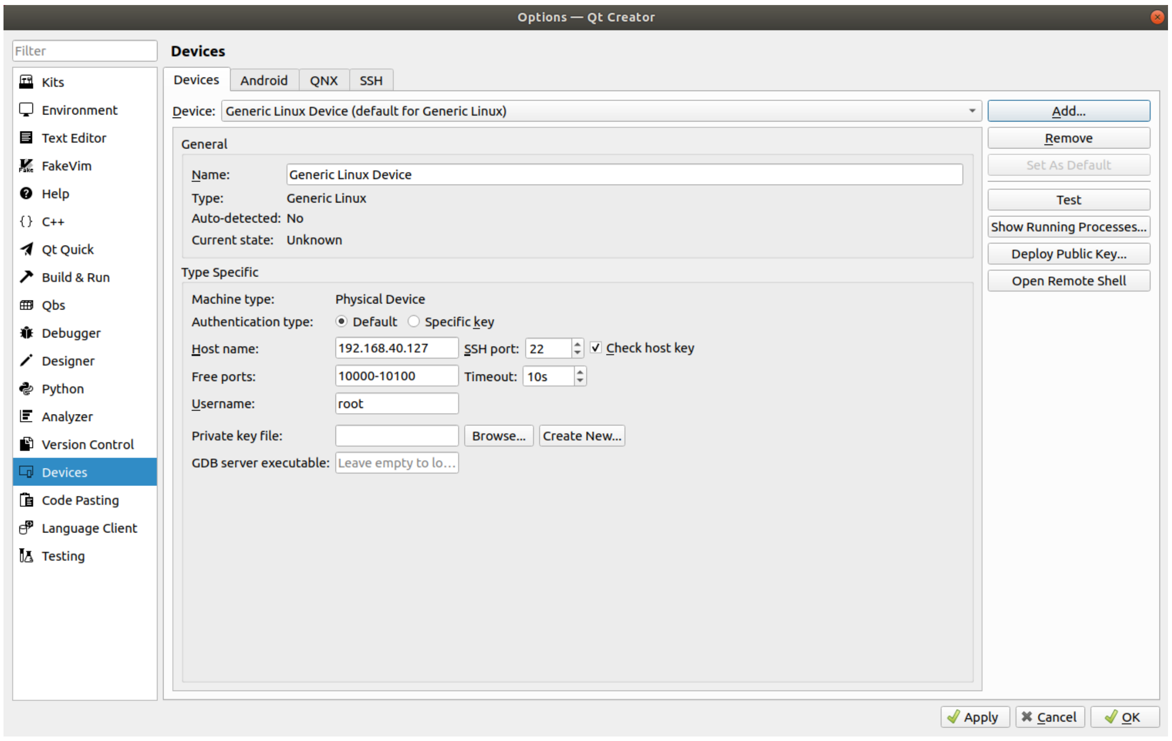


New Generic Linux Device Configuration Setup

Connection

The name to identify this configuration: Generic Linux Device
The device's host name or IP address: 192.168.30.100
The username to log into the device: root
The authentication type: [✓ Password] Key Agent
The user's password: ●●●●
The file containing the user's private key: /root/.ssh/id_rsa

Figure 5-9. Configure device information



Device: Generic Linux Device (default for Generic Linux)

General

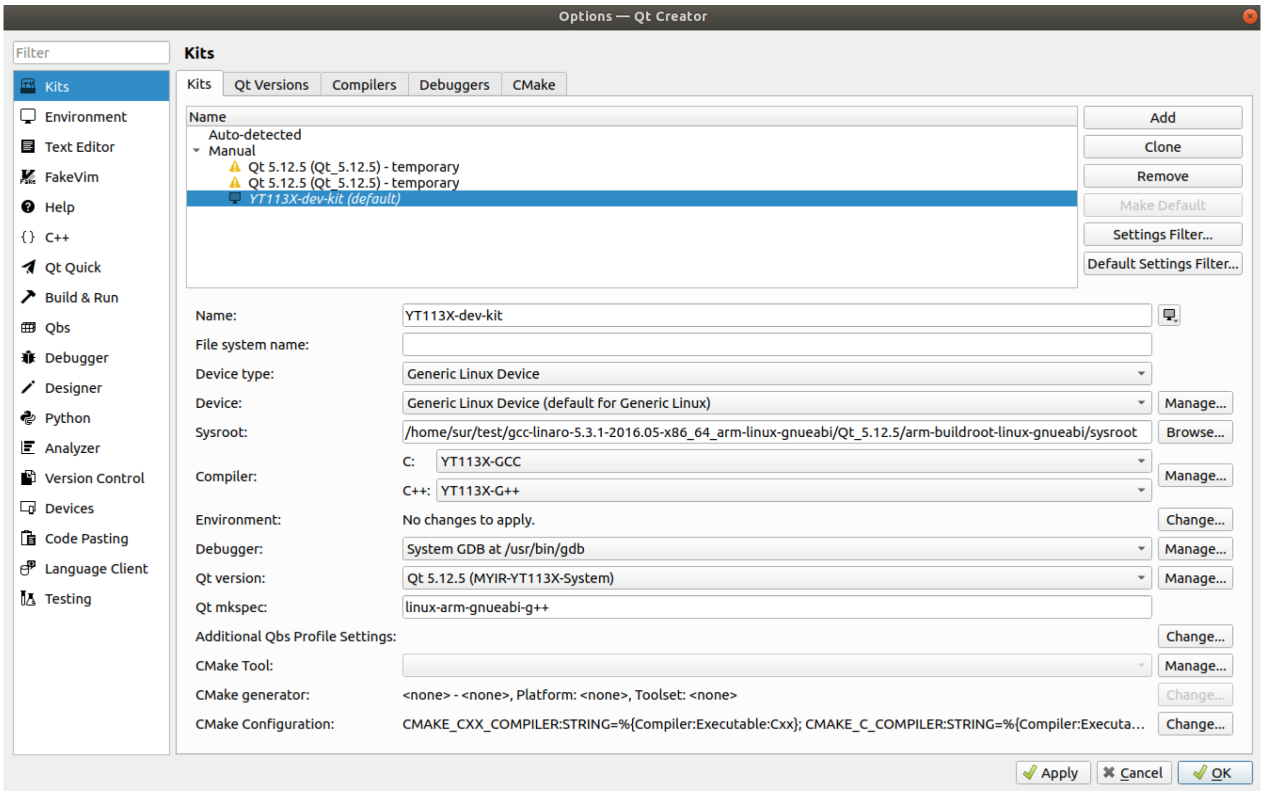
Name: Generic Linux Device
Type: Generic Linux
Auto-detected: No
Current state: Unknown

Type Specific

Machine type: Physical Device
Authentication type: [✓ Default] Specific key
Host name: 192.168.40.127 SSH port: 22 [✓ Check host key]
Free ports: 10000-10100 Timeout: 10s
Username: root
Private key file: Browse... Create New...
GDB server executable: Leave empty to lo...

Figure 5-10. Complete configuration of device information

Click "Build & Run" on the left to go back to the "Kits" tab, set "Name" to "YT113X- dev-kit" and "Device" to "Generic Linux Device". Sysroot" selects the system directory of the target device, here is "/home/sur/test/gcc-linaro-5.3.1-2016.05- x86_64_arm-linux-gnueabi/Qt_5.12.5/arm-buildroot-linux-gnueabi/ sysroot" for example." Compiler" select the previously configured name "YT113X-G++", "Qt version" select the previously configured name "Qt 5.12.5 (MYIR-YT113X-System)", "Qt mkspec" fill in "linux-arm-gnueabi-g++" (optional). Other default can be, and finally click "Apply" and "OK" button.



Name
Auto-detected
Manual
Qt 5.12.5 (Qt_5.12.5) - temporary
Qt 5.12.5 (Qt_5.12.5) - temporary
YT113X-dev-kit (default)

Name:	YT113X-dev-kit
File system name:	
Device type:	Generic Linux Device
Device:	Generic Linux Device (default for Generic Linux)
Sysroot:	/home/sur/test/gcc-linaro-5.3.1-2016.05-x86_64_arm-linux-gnueabi/Qt_5.12.5/arm-buildroot-linux-gnueabi/sysroot
Compiler:	C: YT113X-GCC C++: YT113X-G++
Environment:	No changes to apply.
Debugger:	System GDB at /usr/bin/gdb
Qt version:	Qt 5.12.5 (MYIR-YT113X-System)
Qt mkspec:	linux-arm-gnueabi-g++
Additional Qbs Profile Settings:	
CMake Tool:	
CMake generator:	<none> - <none>, Platform: <none>, Toolset: <none>
CMake Configuration:	CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable:Cxx}; CMAKE_C_COMPILER:STRING=%{Compiler:Executa...

Figure 5-11. Configuring Development Board Information

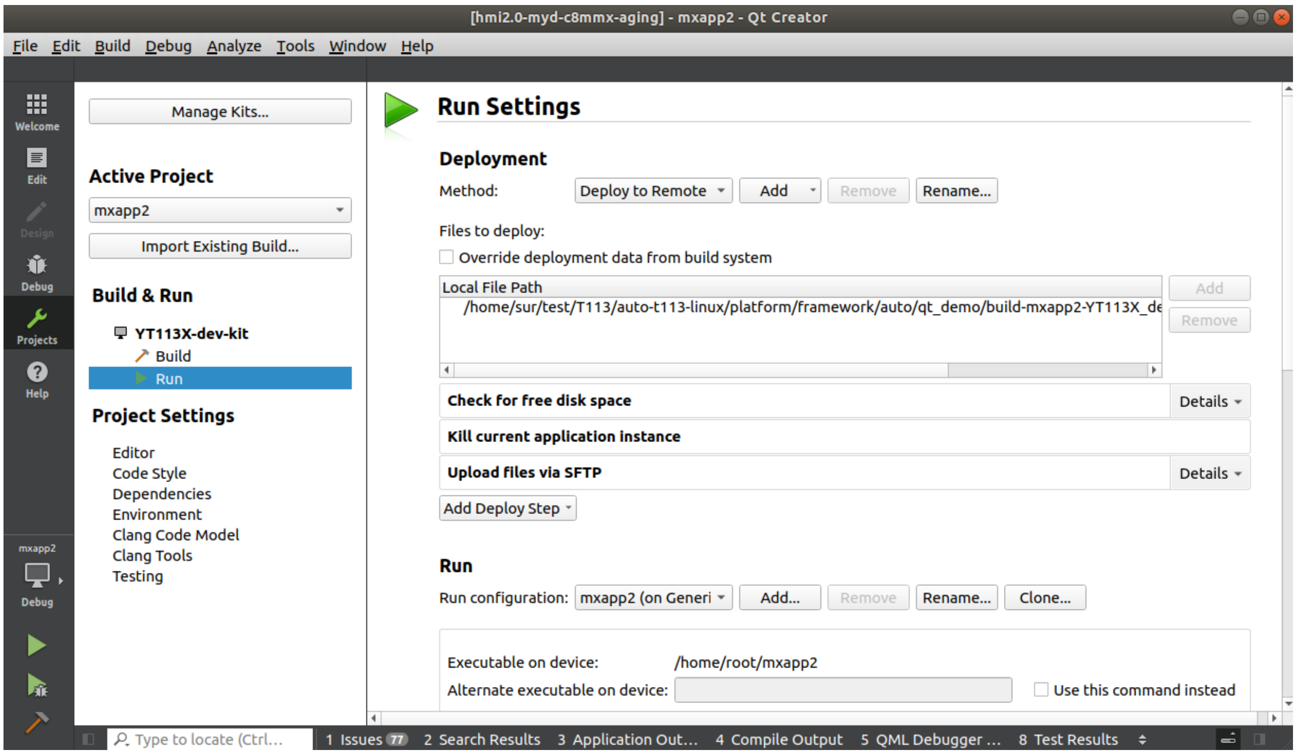
The QT development environment is now complete, and the subsequent development of QT applications can be carried out.

5.3. Measy HMI2.X Compilation

Copy mxapp2.tar.gz to a working directory under Ubuntu and extract the source code. Configure it with the appropriate compilation toolkit to compile the routine

Select "File"->"Open File or Project" in the menu bar, in the opened dialog box, browse to the directory of "MXAPP" routine, select the "mxapp2.pro" file, and click the "Open" button

After the project is opened, select the "Projects" icon in the left menu bar, the right interface will switch to the manage kits management interface, and under the "Build & Run" tab, select the "YT113X-dev-kit" option for the kit, so that the project will use the "YT113X -dev-kit" related configuration kit to build the application.


[hmi2.0-myd-c8mmx-aging] - mxapp2 - Qt Creator

Active Project: mxapp2

Build & Run: YT113X-dev-kit — Build, Run

Run Settings

Deployment
Method: Deploy to Remote

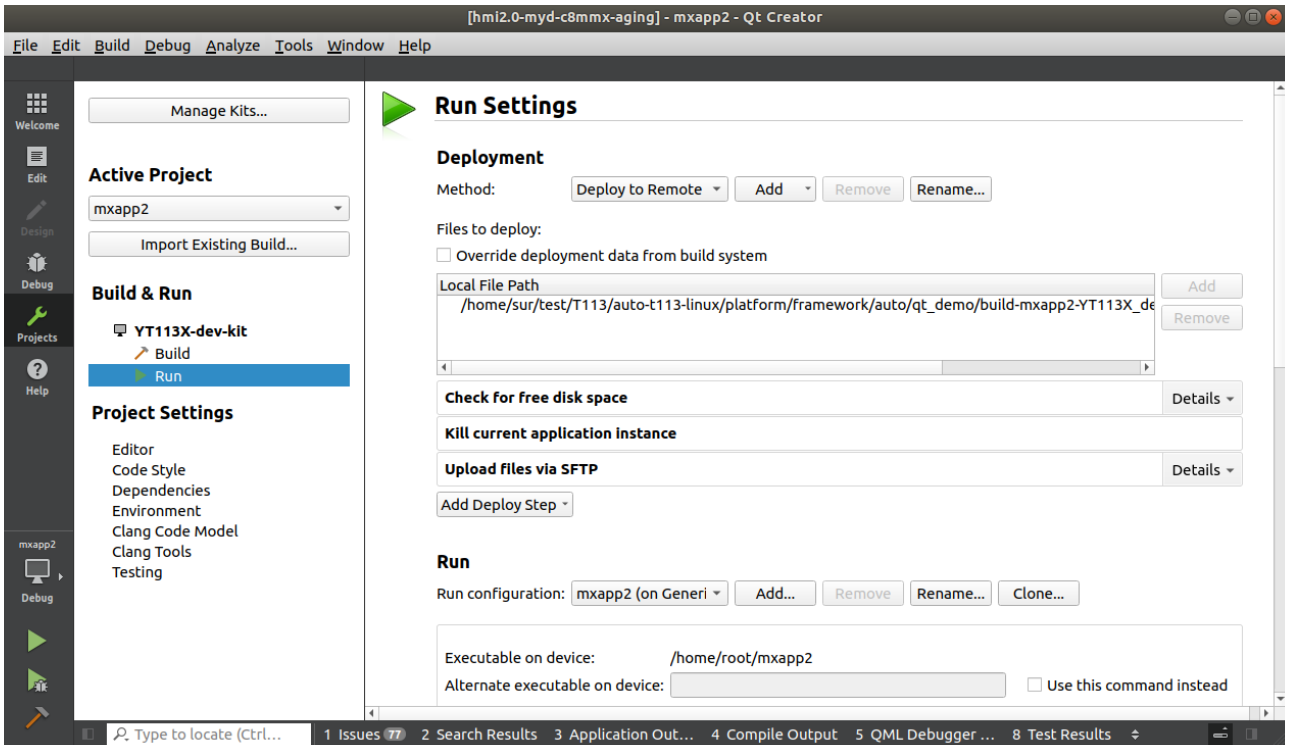
Files to deploy:
Override deployment data from build system
Local File Path: /home/sur/test/T113/auto-t113-linux/platform/framework/auto/qt_demo/build-mxapp2-YT113X_de

Check for free disk space
Kill current application instance
Upload files via SFTP

Run
Run configuration: mxapp2 (on Generi

Executable on device: /home/root/mxapp2
Alternate executable on device:
Use this command instead

Figure 5-12. Open project



Active Project: mxapp2

Method: Deploy to Remote

Override deployment data from build system

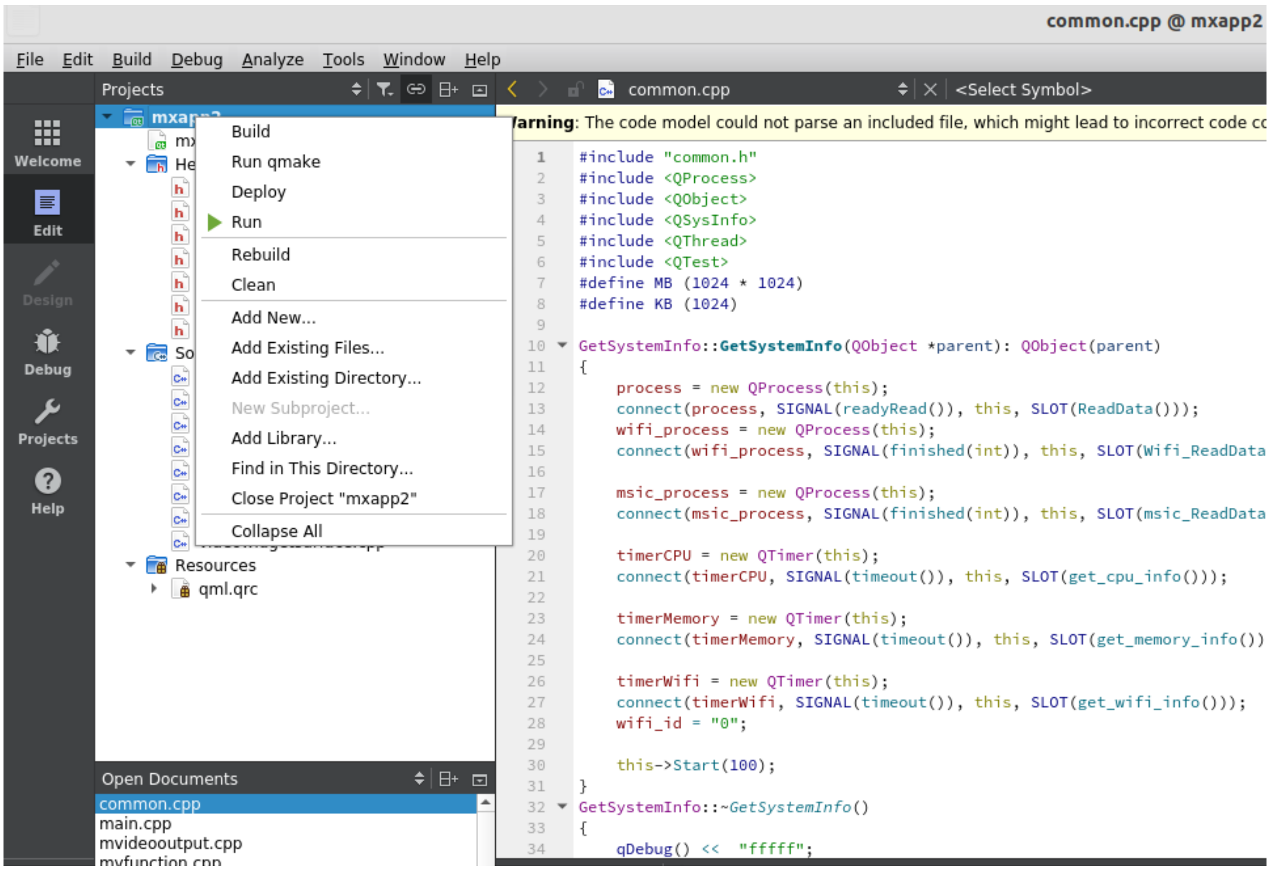
Local File Path
/home/sur/test/T113/auto-t113-linux/platform/framework/auto/qt_demo/build-mxapp2-YT113X_de

Run configuration: mxapp2 (on Generi

Executable on device: /home/root/mxapp2
Alternate executable on device:
Use this command instead

Figure 5-13. Select the configured Kit

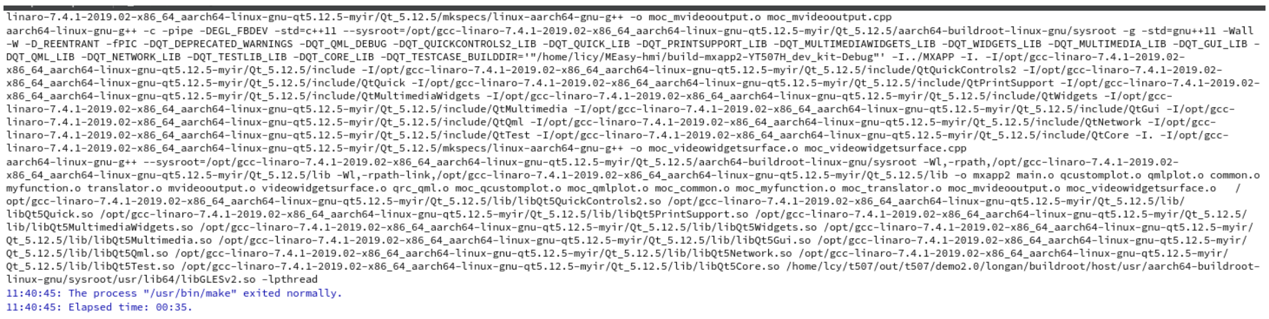
Click on the menu bar "Build"->"Build" button to complete the compilation of the project, and the compilation process will be output on the bottom side.



Warning: The code model could not parse an included file, which might lead to incorrect code co

```
#include "common.h"
#include <QProcess>
#include <QObject>
#include <QSysInfo>
#include <QThread>
#include <QTest>
#define MB (1024 * 1024)
#define KB (1024)

GetSystemInfo::GetSystemInfo(QObject *parent): QObject(parent)
{
    process = new QProcess(this);
    connect(process, SIGNAL(readyRead()), this, SLOT(ReadData()));
    wifi_process = new QProcess(this);
    connect(wifi_process, SIGNAL(finished(int)), this, SLOT(Wifi_ReadData

    msic_process = new QProcess(this);
    connect(msic_process, SIGNAL(finished(int)), this, SLOT(msic_ReadData

    timerCPU = new QTimer(this);
    connect(timerCPU, SIGNAL(timeout()), this, SLOT(get_cpu_info()));

    timerMemory = new QTimer(this);
    connect(timerMemory, SIGNAL(timeout()), this, SLOT(get_memory_info())

    timerWifi = new QTimer(this);
    connect(timerWifi, SIGNAL(timeout()), this, SLOT(get_wifi_info()));
    wifi_id = "0";

    this->Start(100);
}
GetSystemInfo::~GetSystemInfo()
{
    qDebug() <<  "fffff";
```

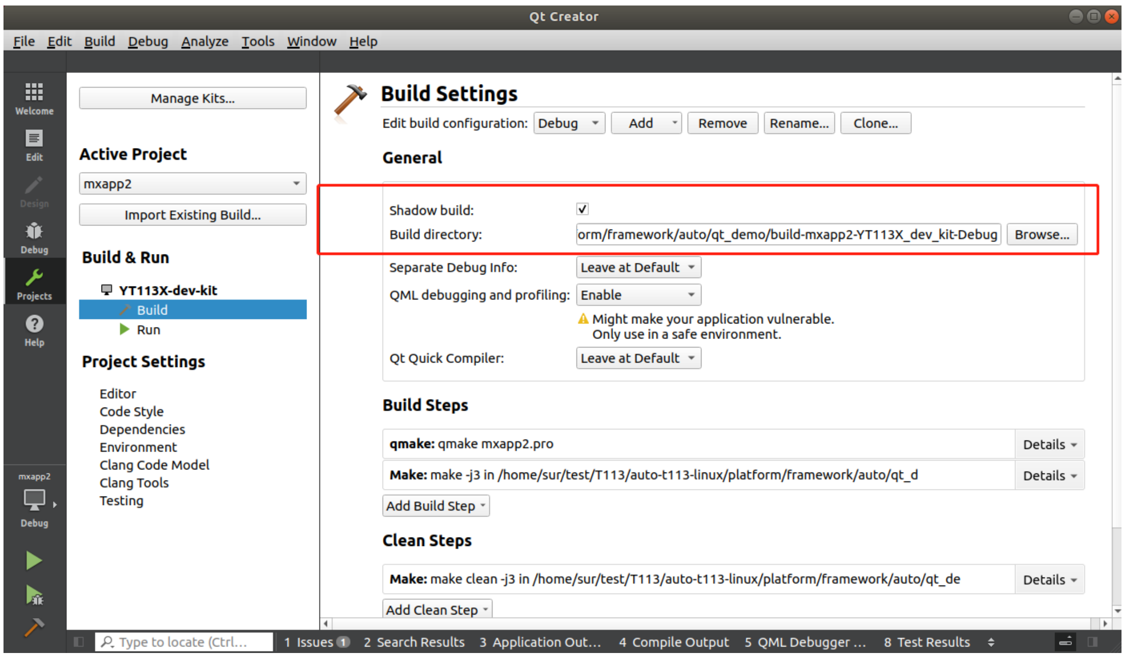
Figure 5-14. Compiling Projects



```
linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/mkspecs/linux-aarch64-gnu-g++ -o moc_mvideooutput.o moc_mvideooutput.cpp
aarch64-linux-gnu-g++ -c -pipe -DEGL_FBDEV -std=c++11 --sysroot=/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/aarch64-buildroot-linux-gnu/sysroot -g -std=gnu++11 -Wall
-W -D_REENTRANT -fPIC -DQT_DEPRECATED_WARNINGS -DQT_QML_DEBUG -DQT_QUICKCONTROLS2_LIB -DQT_QUICK_LIB -DQT_PRINTSUPPORT_LIB -DQT_MULTIMEDIAWIDGETS_LIB -DQT_WIDGETS_LIB -DQT_MULTIMEDIA_LIB -DQT_GUI_LIB -
DQT_QML_LIB -DQT_NETWORK_LIB -DQT_TESTLIB_LIB -DQT_CORE_LIB -DQT_TESTCASE_BUILDDIR='"/home/licy/MEasy-hmi/build-mxapp2-YT507H_dev_kit-Debug"' -I../MXAPP -I. -I/opt/gcc-linaro-7.4.1-2019.02-
x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include -I/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtQuickControls2 -I/opt/gcc-linaro-7.4.1-2019.02-
x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtQuick -I/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtPrintSupport -I/opt/gcc-linaro-7.4.1-2019.02-
x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtMultimediaWidgets -I/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtWidgets -I/opt/gcc-
linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtMultimedia -I/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtGui -I/opt/gcc-
linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtQml -I/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtNetwork -I/opt/gcc-
linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtTest -I/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/include/QtCore -I. -I/opt/gcc-
linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/mkspecs/linux-aarch64-gnu-g++ -o moc_videowidgetsurface.o moc_videowidgetsurface.cpp
aarch64-linux-gnu-g++ --sysroot=/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/aarch64-buildroot-linux-gnu/sysroot -Wl,-rpath,/opt/gcc-linaro-7.4.1-2019.02-
x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib -Wl,-rpath-link,/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib -o mxapp2 main.o qcustomplot.o qmlplot.o common.o
myfunction.o translator.o mvideooutput.o videowidgetsurface.o qrc_qml.o moc_qcustomplot.o moc_qmlplot.o moc_common.o moc_myfunction.o moc_translator.o moc_mvideooutput.o moc_videowidgetsurface.o   /
opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib/libQt5QuickControls2.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib/
libQt5Quick.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib/libQt5PrintSupport.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/
lib/libQt5MultimediaWidgets.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib/libQt5Widgets.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/
Qt_5.12.5/lib/libQt5Multimedia.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib/libQt5Gui.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/
Qt_5.12.5/lib/libQt5Qml.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib/libQt5Network.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/
Qt_5.12.5/lib/libQt5Test.so /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/lib/libQt5Core.so /home/lcy/t507/out/t507/demo2.0/longan/buildroot/host/usr/aarch64-buildroot-
linux-gnu/sysroot/usr/lib64/libGLESv2.so -lpthread
11:40:45: The process "/usr/bin/make" exited normally.
11:40:45: Elapsed time: 00:35.
```

Figure 5-15. Compilation completed

After building the mxapp2 project with QtCreator, the compiled binaries are stored in the specified directory. Then just copy the mxapp2 file to the development board and run it



Build Settings

Edit build configuration: Debug

General

Shadow build: ✓
Build directory: orm/framework/auto/qt_demo/build-mxapp2-YT113X_dev_kit-Debug
Separate Debug Info: Leave at Default
QML debugging and profiling: Enable
Might make your application vulnerable. Only use in a safe environment.
Qt Quick Compiler: Leave at Default

Build Steps

qmake: qmake mxapp2.pro
Make: make -j3 in /home/sur/test/T113/auto-t113-linux/platform/framework/auto/qt_d

Clean Steps

Make: make clean -j3 in /home/sur/test/T113/auto-t113-linux/platform/framework/auto/qt_de

Figure 5-16. Output Directory

5.4. SDK integration applications

When the application development is finished, it needs to be integrated with the system; boot up and self-start, etc

First, copy the QT application source code to the "platform/framework/auto/qt_demo" directory

and add the following configuration code to build.sh in that directory

```
if [ -d ./MXAPP ];then
        cd ./MXAPP
        make distclean
        ./makeMXAPP
        echo "=====build MXAPP success!!!======"
        cd ../
fi
```

Add the following code to the clean.sh directory

```
if [ -d ./MXAPP ];then
        cd ./MXAPP
        make distclean
        cd ../
fi
```

Add a compile control script file makeMXAPP (named the same as added in build.sh) to the MXAPP directory with the following code

```
#!/bin/sh
PATH=$LICHEE_BR_OUT/host/bin/:$PATH
$QT_INSTALL_DIR/bin/qmake  -o Makefile mxapp2.pro
make -j32
```

Add it to the boot self-start. The boot script path is: "platform/framework/auto/rootfs/etc/qtenv.sh"

```
export  QTDIR=/usr/local/Qt_5.12.5
if [ -d $QTDIR ];then


        export  QT_ROOT=$QTDIR
```

```
    export  PATH=$QTDIR/bin:$PATH
    export  LD_LIBRARY_PATH=$QTDIR/lib:/usr/lib/cedarx/:$LD_LIBRARY_PATH

    export QT_QPA_PLATFORM_PLUGIN_PATH=$QT_ROOT/plugins
    export QT_QPA_PLATFORM=linuxfb:tty=/dev/fb0
    export QT_QPA_FONTDIR=$QT_ROOT/fonts

    export QML_IMPORT_PATH=$QTDIR/qml
    export QML2_IMPORT_PATH=$QTDIR/qml

    TouchDevice="generic ft5x06 (79)"
    for InputDevices in /sys/class/input/input*
    do
        DeviceName=`cat $InputDevices/name`
        if [[ $DeviceName == $TouchDevice ]];then
          TouchDeviceNum=${InputDevices##*input}
          export QT_QPA_EVDEV_TOUCHSCREEN_PARAMETERS=/dev/input/eve
nt$TouchDeviceNum
          echo "add "/dev/input/event$TouchDeviceNum "to Qt Application."
          break
        fi
    done
    if [ ! -n "$TouchDeviceNum" ]; then
     echo "Error:Input device $TouchDevice can not be found,plz check it!"
    fi

    export QT_QPA_PLATFORM=linuxfb
    export set TSLIB_TSDEVICE=/dev/input/dev/input/event$TouchDeviceNum
    export set TSLIB_CONFFILE=/etc/ts.conf
    export set TSLIB_PLUGINDIR=/usr/lib/ts
    export set TSLIB_CALIBFILE=/etc/pointercal
    export set TSLIB_CONSOLEDEVICE=none
    export set TSLIB_FBDEVICE=/dev/fb0
```

```
        export QT_QPA_GENERIC_PLUGINS=evdevtouch,evdevmouse:/dev/input/eve
nt6
        #export QT_QPA_EGLFS_INTEGRATION=eglfs_mali
        #export QT_QPA_FB_HIDECURSOR=1
        #export QT_QPA_EGLFS_HIDECURSOR=1
        #export QT_QPA_EGLFS_ROTATION=90

        #export QWS_MOUSE_PROTO=Intellimouse:/dev/input/event6

        #export DBUS_SESSION_BUS_ADDRESS=`cat /tmp/dbusaddr`
        mkdir -p /dev/shm
        ulimit -c unlimited
        mxapp2 &
        echo "find qt5 installed done"
fi
```

- /dev/fb0： Indicates the interface of the first main display
- generic ft5x06 (79)： Indicates the interface of the first main display
- mxapp2 &： Application Launch Name

The QT application development and integration is now complete, and the production image can be packaged by executing the following command in the SDK directory

```
PC$ ./build.sh
PC$ ./build.sh qt
PC$ ./build.sh
PC$ ./build.sh pack
```

For details, please refer to the relevant chapter of MYD-YT113X Linux System Development Guide.

6. References

https://ubuntu.com/download/desktop

https://www.qt.io/

Appendix A

Warranty & Technical Support Services

MYIR Electronics Limited is a global provider of ARM hardware and software tools, design solutions for embedded applications. We support our customers in a wide range of services to accelerate your time to market.

MYIR is an ARM Connected Community Member and work closely with ARM and many semiconductor vendors. We sell products ranging from board level products such as development boards, single board computers and CPU modules to help with your evaluation, prototype, and system integration or creating your own applications. Our products are used widely in industrial control, medical devices, consumer electronic, telecommunication systems, Human Machine Interface (HMI) and more other embedded applications. MYIR has an experienced team and provides custom design services based on ARM processors to help customers make your idea a reality.

The contents below introduce to customers the warranty and technical support services provided by MYIR as well as the matters needing attention in using MYIR’ s products.

Service Guarantee

MYIR regards the product quality as the life of an enterprise. We strictly check and control the core board design, the procurement of components, production control, product testing, packaging, shipping and other aspects and strive to provide products with best quality to customers. We believe that only quality products and excellent services can ensure the long-term cooperation and mutual benefit.

Price

MYIR insists on providing customers with the most valuable products. We do not pursue excess profits which we think only for short-time cooperation. Instead, we hope to establish long-term cooperation and win-win business with customers. So we will offer reasonable prices in the hope of making the business greater with the customers together hand in hand.

Delivery Time

MYIR will always keep a certain stock for its regular products. If your order quantity is less than the amount of inventory, the delivery time would be within three days; if your order quantity is greater than the number of inventory, the delivery time would be always four to six weeks. If for any urgent delivery, we can negotiate with customer and try to supply the goods in advance.

Technical Support

MYIR has a professional technical support team. Customer can contact us by email (support@myirtech.com), we will try to reply you within 48 hours. For mass production and customized products, we will specify person to follow the case and ensure the smooth production.

After-sale Service

MYIR offers one year free technical support and after-sales maintenance service from the purchase date. The service covers:

Technical support service

MYIR offers technical support for the hardware and software materials which have provided to customers:

- To help customers compile and run the source code we offer;
- To help customers solve problems occurred during operations if users follow the user manual documents;
- To judge whether the failure exists;
- To provide free software upgrading service.

However, the following situations are not included in the scope of our free technical support service:

- Hardware or software problems occurred during customers’ own development;
- Problems occurred when customers compile or run the OS which is tailored by themselves;
- Problems occurred during customers’ own applications development;
- Problems occurred during the modification of MYIR’ s software source code.

After-sales maintenance service

The products except LCD, which are not used properly, will take the twelve months free maintenance service since the purchase date. But following situations are not included in the scope of our free maintenance service:

- The warranty period is expired;
- The customer cannot provide proof-of-purchase or the product has no serial number;
- The customer has not followed the instruction of the manual which has caused the damage the product;
- Due to the natural disasters (unexpected matters), or natural attrition of the components, or unexpected matters leads the defects of appearance/function;
- Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the boards, all those reasons which have caused the damage of the products or defects of appearance;
- Due to unauthorized weld or dismantle parts or repair the products which has caused the damage of the products or defects of appearance;
- Due to unauthorized installation of the software, system or incorrect configuration or computer virus which has caused the damage of products.

Warm tips

1. MYIR does not supply maintenance service to LCD. We suggest the customer first check the LCD when receiving the goods. In case the LCD cannot run or no display, customer should contact MYIR within 7 business days from the moment get the goods.

2. Please do not use finger nails or hard sharp object to touch the surface of the LCD.

3. MYIR suggests user purchasing a piece of special wiper to wipe the LCD after long time use, please avoid clean the surface with fingers or hands to leave fingerprint.

4. Do not clean the surface of the screen with chemicals.

5. Please read through the product user manual before you using MYIR’ s products.

6. For any maintenance service, customers should communicate with MYIR to confirm the issue first. MYIR’ s support team will judge the failure to see if the goods need to be returned for repair service, we will issue you RMA number for return maintenance service after confirmation.

Maintenance period and charges

- MYIR will test the products within three days after receipt of the returned goods and inform customer the testing result. Then we will arrange shipment within one week for the repaired goods to the customer. For any special failure, we will negotiate with customers to confirm the maintenance period.
- For products within warranty period and caused by quality problem, MYIR offers free maintenance service; for products within warranty period but out of free maintenance service scope, MYIR provides maintenance service but shall charge some basic material cost; for products out of warranty period, MYIR provides maintenance service but shall charge some basic material cost and handling fee.

Shipping cost

During the warranty period, the shipping cost which delivered to MYIR should be responsible by user; MYIR will pay for the return shipping cost to users when the product is repaired. If the warranty period is expired, all the shipping cost will be responsible by users.

Products Life Cycle

MYIR will always select mainstream chips for our design, thus to ensure at least ten years continuous supply; if meeting some main chip stopping production, we will inform customers in time and assist customers with products updating and upgrading.

Value-added Services

1. MYIR provides services of driver development base on MYIR’ s products, like serial port, USB, Ethernet, LCD, etc.

2. MYIR provides the services of OS porting, BSP drivers’ development, API software development, etc.

3. MYIR provides other products supporting services like power adapter, LCD panel, etc.

4. ODM/OEM services.

MYIR Electronics Limited

Room 04, 6th Floor, Building No.2, Fada Road,

Yunli Inteiligent Park, Bantian, Longgang District.

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com