MYD-YT507H Application Development Notes

The installation of Qt Creator is cross-compiled with Measy-HMI2.0

File state: draft [] Release [√]	**File identification:**	MYIR-MYD-YT507H-SW-AN-EN-L4.9.170
	Current version:	V1.0(DOC)
	As a person:	licy.li
	Creation date:	2020-08-04
	Last updated:	2022-07-10

Copyright © 2010-2022 Shenzhen MYIR Electronics Co., LTD. All Rights reserved

本文档仅限内部使用

Version history

version	The author	participants	The date of	note
V1.0	licy	--	20220710	Create the document

本文档仅限内部使用

contents

本文档仅限内部使用

1. An overview of the

Qt is a cross-platform graphical application development framework that can be used on different sizes of devices and platforms, while offering different copyright versions for users to choose from. MYD-YT507H uses Qt version 5.12.5 for application development.In the development of Qt application, it is recommended to use QtCreator integrated development environment, which can develop Qt application under Linux PC and automatically cross-compile into ARM architecture of development board.

In this chapter, we will use the SDK tools built by MYIR as a cross-compilation system to work with Qt Creator to quickly develop graphical applications.

2. Hardware resources

There is no

Software resources

- Ubuntu20.04 desktop
- Qtcreator 4.12
- gcc-linaro-7.4.1-2019.02-x86_64_aarch64-Linux-gnu-qt5.12.5-myir.tar.bz2
- mxapp2.0

4. Environment preparation

The Ubuntu desktop system is required. Install the Ubuntu 20.04 desktop system by yourself.

Need to install the compiler cross tool chain provided by MYIR electronics, path:

03 - Tools\Complie Toolchain\gcc-linaro-7.4.1-2019.02-x86_64_aarch64-Linux-gnu-qt5.12.5-myir.tar.bz2, For installation methods, see Section 2.3 of *MYD-YT507H Linux Development Guide*.

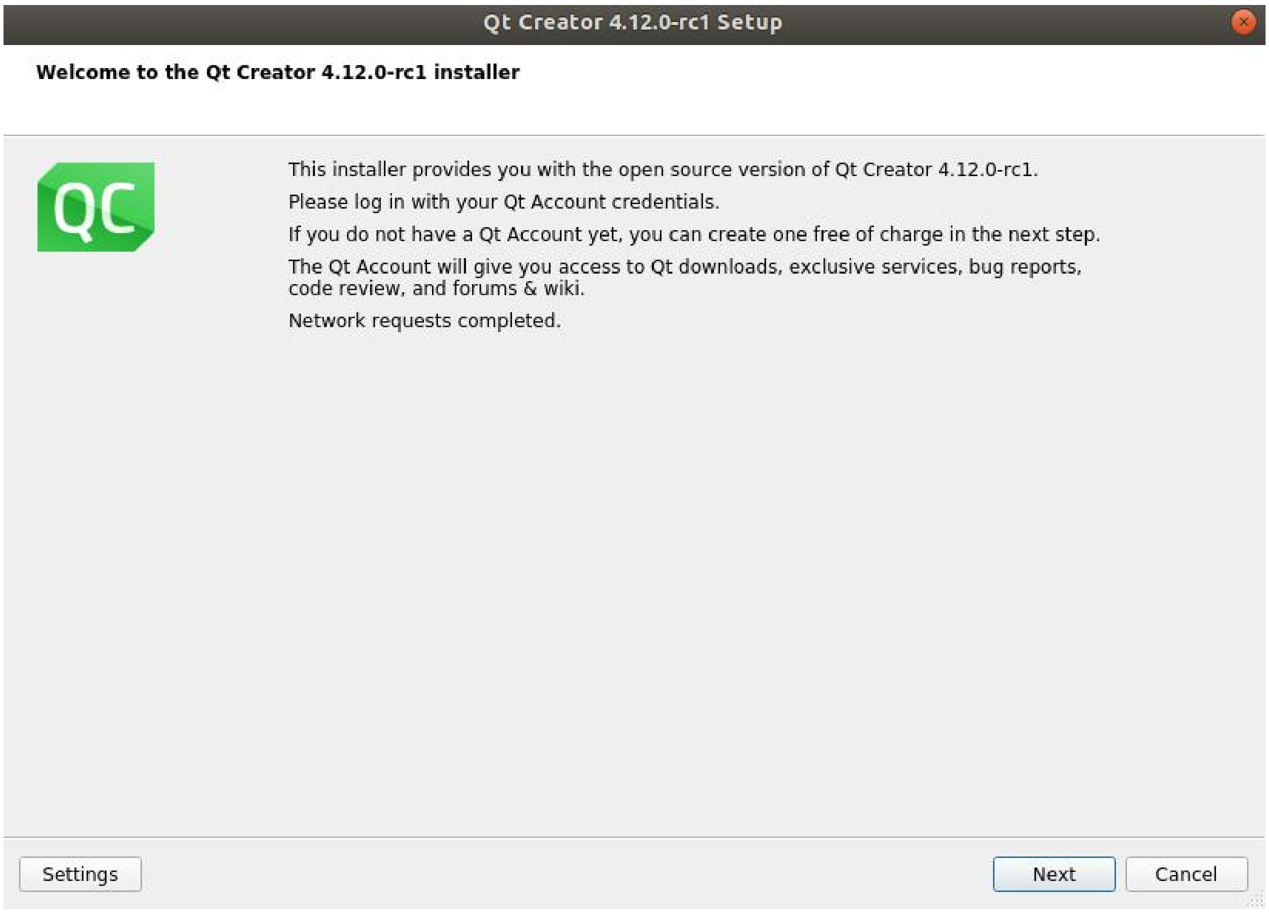
本文档仅限内部使用

Get the mxapp2.0 source code from the development SDK package provided by Mill at 04-sources/mxapp2.tar.gz or from platform/framework/auto/qt_demo/MXAPP.

5. Procedure

5.1. Install Qt Creator

Get the Qt Creator installation package from the QT download package or the official MYIR package, and download it from the QT website (4.12 as an example).
The official download address: http://download.qt.io/development_releases/qtcreator/
QtCreator installation package is an ubuntu system binaries, under ubuntu terminal direct execution can complete the installation./qt creator-opensource-linux-x86_64-4.12.0.run


Qt Creator 4.12.0-rc1 Setup

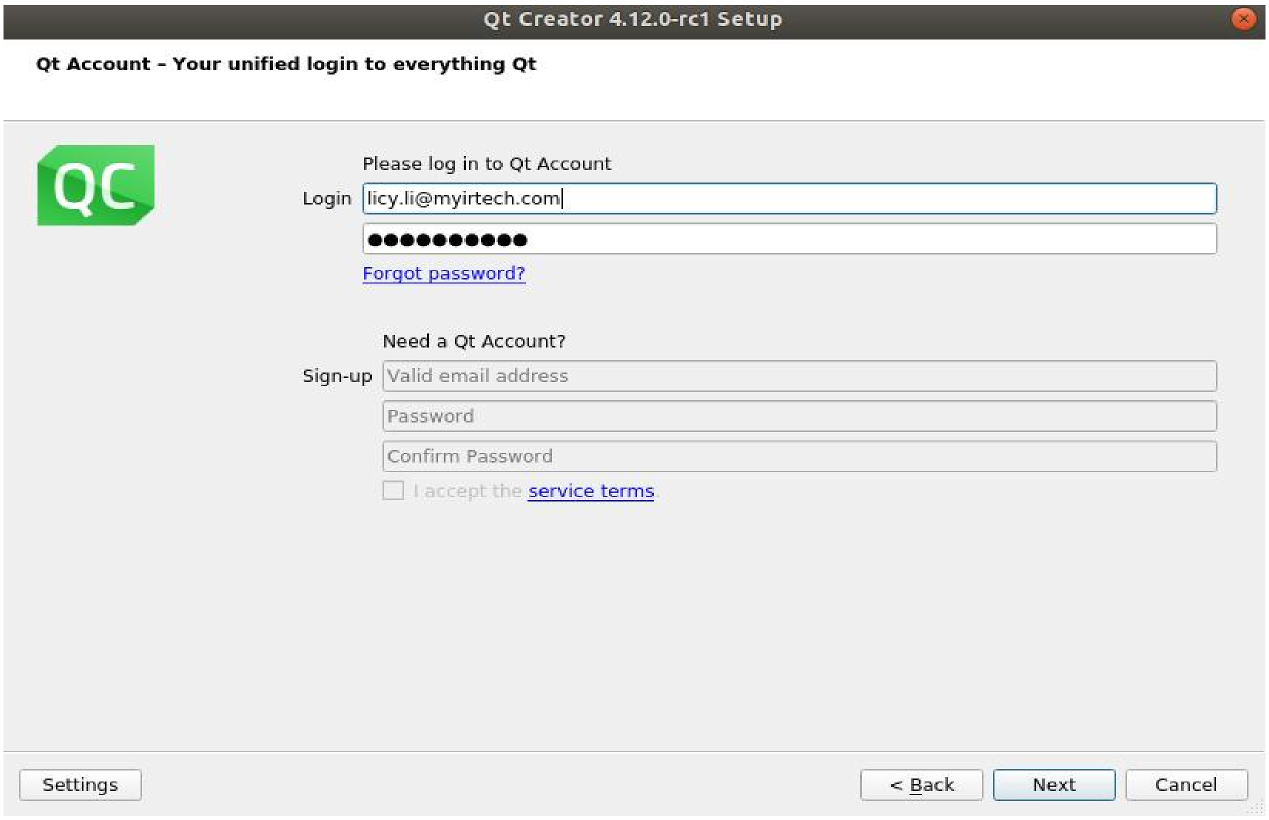
Welcome to the Qt Creator 4.12.0-rc1 installer

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 installation

Click next, enter account passwords, account need to registered in's official website https://login.qt.io/register

本文档仅限内部使用


Qt Creator 4.12.0-rc1 Setup

Qt Account – Your unified login to everything Qt

Please log in to Qt Account

Login: licy.li@myirtech.com

●●●●●●●●●●●

Forgot password?

Need a Qt Account?

Sign-up: Valid email address

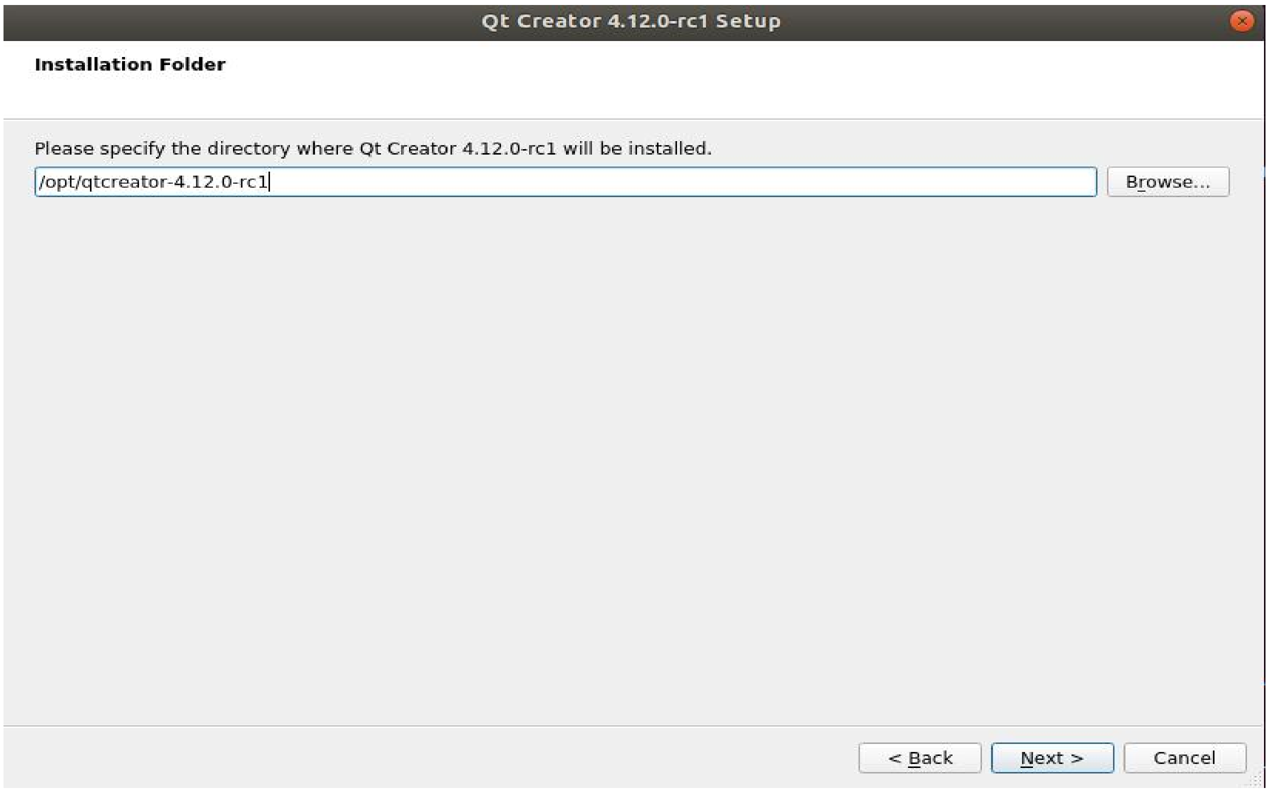
Password

Confirm Password

I accept the service terms

Figure 5-2. Login account and password

Set the installation path for other configuration items.


Qt Creator 4.12.0-rc1 Setup

Installation Folder

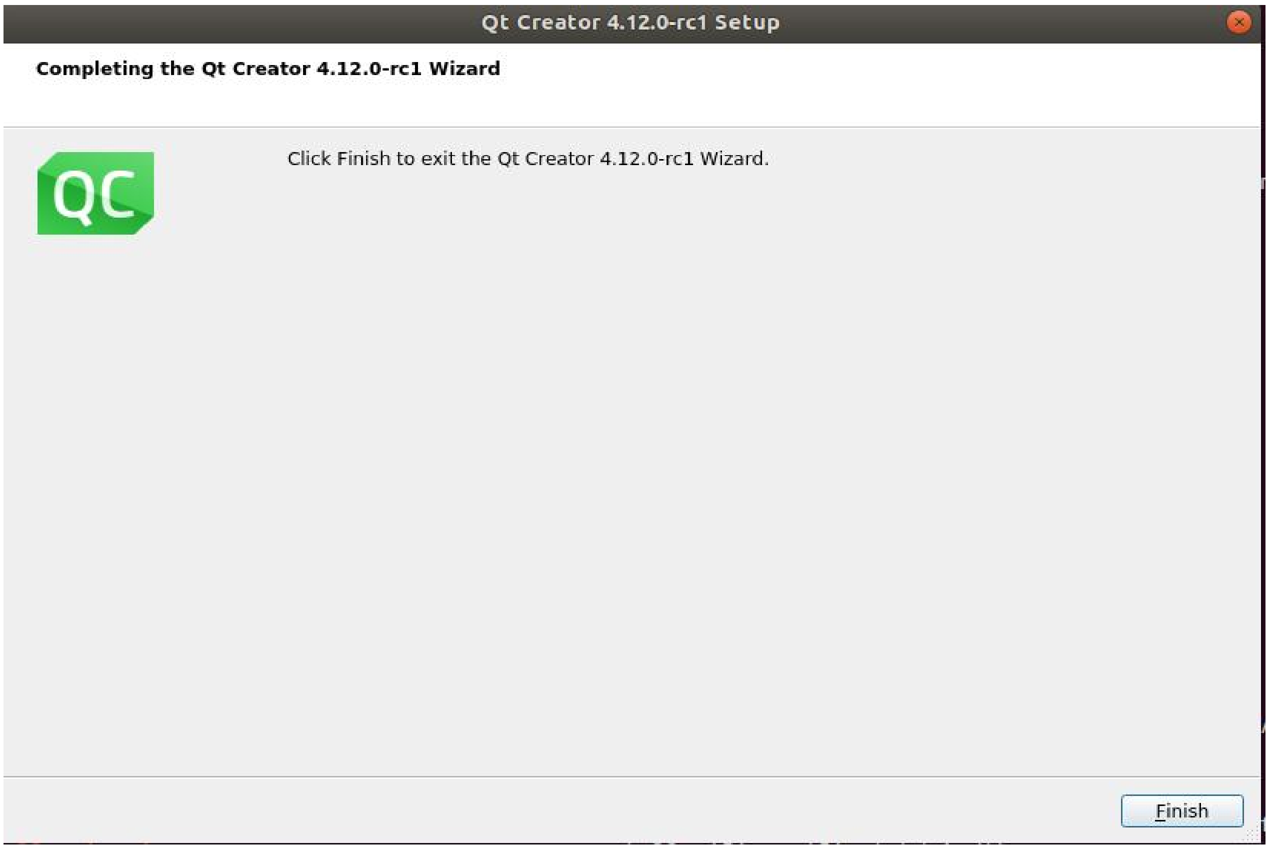
Please specify the directory where Qt Creator 4.12.0-rc1 will be installed.

/opt/qtcreator-4.12.0-rc1

Figure 5-3. Select the installation path

本文档仅限内部使用

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.

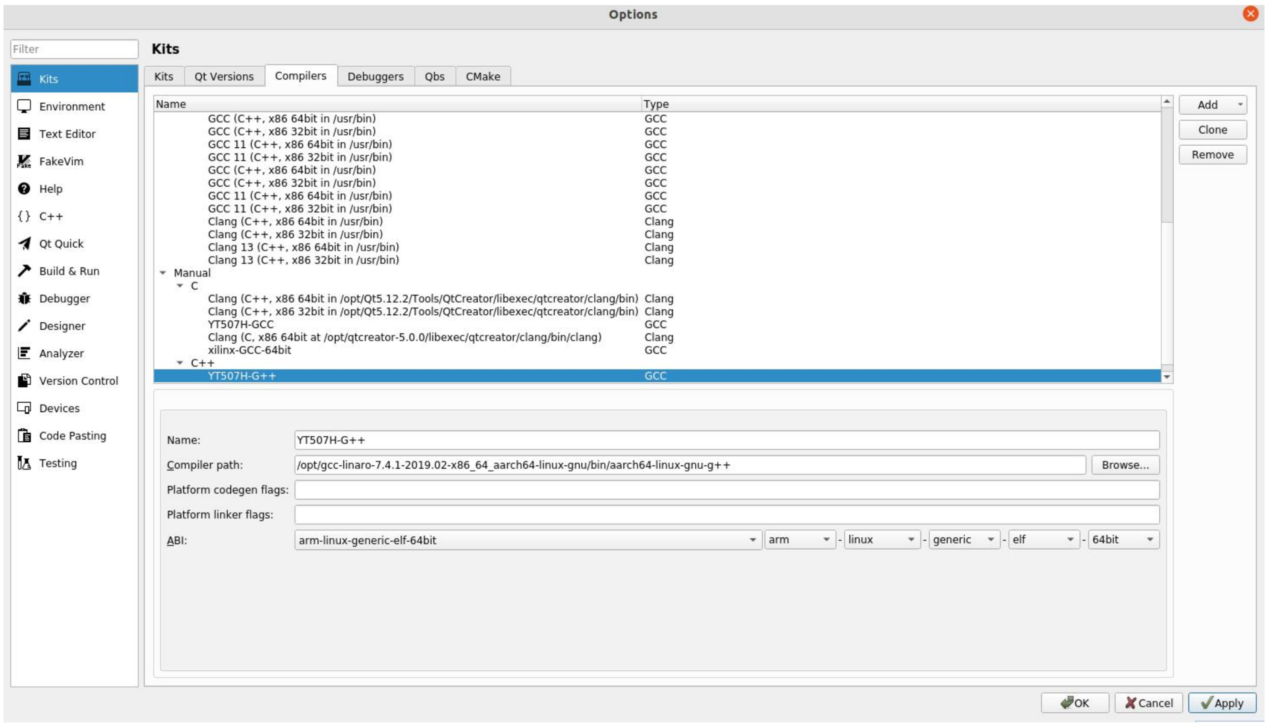
Figure 5-4. Installation completed

After installing Qtcreator, you can configure the development environment

本文档仅限内部使用

5.2. Configuring the cross-compilation environment

1) To start QTCreator, run "qtcreator.sh" from the terminal as follows: /opt/qtcreator-4.12.0-rc1/bin/qtcreator.sh &
2) After running QtCreator, click Tools -> Options in turn to bring up the Options dialog box. Click Kits on the left and select the Compilers TAB on the right.



```
Name                                                                              Type
        GCC (C++, x86 64bit in /usr/bin)                                          GCC
        GCC (C++, x86 32bit in /usr/bin)                                          GCC
        GCC 11 (C++, x86 64bit in /usr/bin)                                       GCC
        GCC 11 (C++, x86 32bit in /usr/bin)                                       GCC
        GCC (C++, x86 64bit in /usr/bin)                                          GCC
        GCC (C++, x86 32bit in /usr/bin)                                          GCC
        GCC 11 (C++, x86 64bit in /usr/bin)                                       GCC
        GCC 11 (C++, x86 32bit in /usr/bin)                                       GCC
        Clang (C++, x86 64bit in /usr/bin)                                        Clang
        Clang (C++, x86 32bit in /usr/bin)                                        Clang
        Clang 13 (C++, x86 64bit in /usr/bin)                                     Clang
        Clang 13 (C++, x86 32bit in /usr/bin)                                     Clang
  Manual
    C
        Clang (C++, x86 64bit in /opt/Qt5.12.2/Tools/QtCreator/libexec/qtcreator/clang/bin)  Clang
        Clang (C++, x86 32bit in /opt/Qt5.12.2/Tools/QtCreator/libexec/qtcreator/clang/bin)  Clang
        YT507H-GCC                                                                GCC
        Clang (C, x86 64bit at /opt/qtcreator-5.0.0/libexec/qtcreator/clang/bin/clang)       Clang
        xilinx-GCC-64bit                                                          GCC
    C++
        YT507H-G++                                                                GCC

Name:                   YT507H-G++
Compiler path:          /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++   Browse...
Platform codegen flags:
Platform linker flags:
ABI:                    arm-linux-generic-elf-64bit   arm - linux - generic - elf - 64bit
```

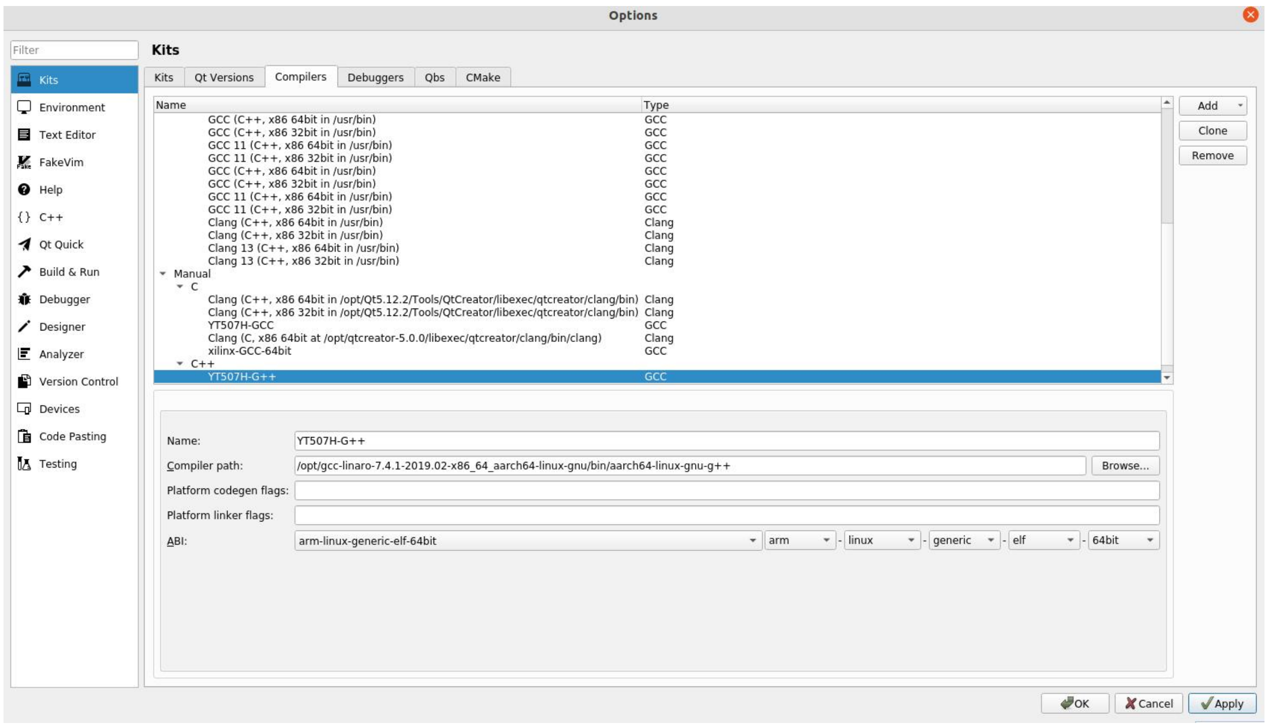
Figure 5-5.Selecting a compiler

3) Click Add on the right. When the drop down list appears, select G++ and fill in "Name" as "YT507H-g++", "Compiler path" next to "Browse.."Button to aarch64-linux-gnu-g++, Examples of path is/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/bin/aarch64-linux-gnu-g++".

   To do the same, choose the aarch64-linux-gnu-gcc compiler as G++

When you're done, click "Apply".

本文档仅限内部使用



```
Name                                                                            Type
      GCC (C++, x86 64bit in /usr/bin)                                          GCC
      GCC (C++, x86 32bit in /usr/bin)                                          GCC
      GCC 11 (C++, x86 64bit in /usr/bin)                                       GCC
      GCC 11 (C++, x86 32bit in /usr/bin)                                       GCC
      GCC (C++, x86 64bit in /usr/bin)                                          GCC
      GCC (C++, x86 32bit in /usr/bin)                                          GCC
      GCC 11 (C++, x86 64bit in /usr/bin)                                       GCC
      GCC 11 (C++, x86 32bit in /usr/bin)                                       GCC
      Clang (C++, x86 64bit in /usr/bin)                                        Clang
      Clang (C++, x86 32bit in /usr/bin)                                        Clang
      Clang 13 (C++, x86 64bit in /usr/bin)                                     Clang
      Clang 13 (C++, x86 32bit in /usr/bin)                                     Clang
Manual
  C
      Clang (C++, x86 64bit in /opt/Qt5.12.2/Tools/QtCreator/libexec/qtcreator/clang/bin)  Clang
      Clang (C++, x86 32bit in /opt/Qt5.12.2/Tools/QtCreator/libexec/qtcreator/clang/bin)  Clang
      YT507H-GCC                                                                GCC
      Clang (C, x86 64bit at /opt/qtcreator-5.0.0/libexec/qtcreator/clang/bin/clang)  Clang
      xilinx-GCC-64bit                                                          GCC
  C++
      YT507H-G++                                                                GCC
```

Name: YT507H-G++

Compiler path: /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-g++

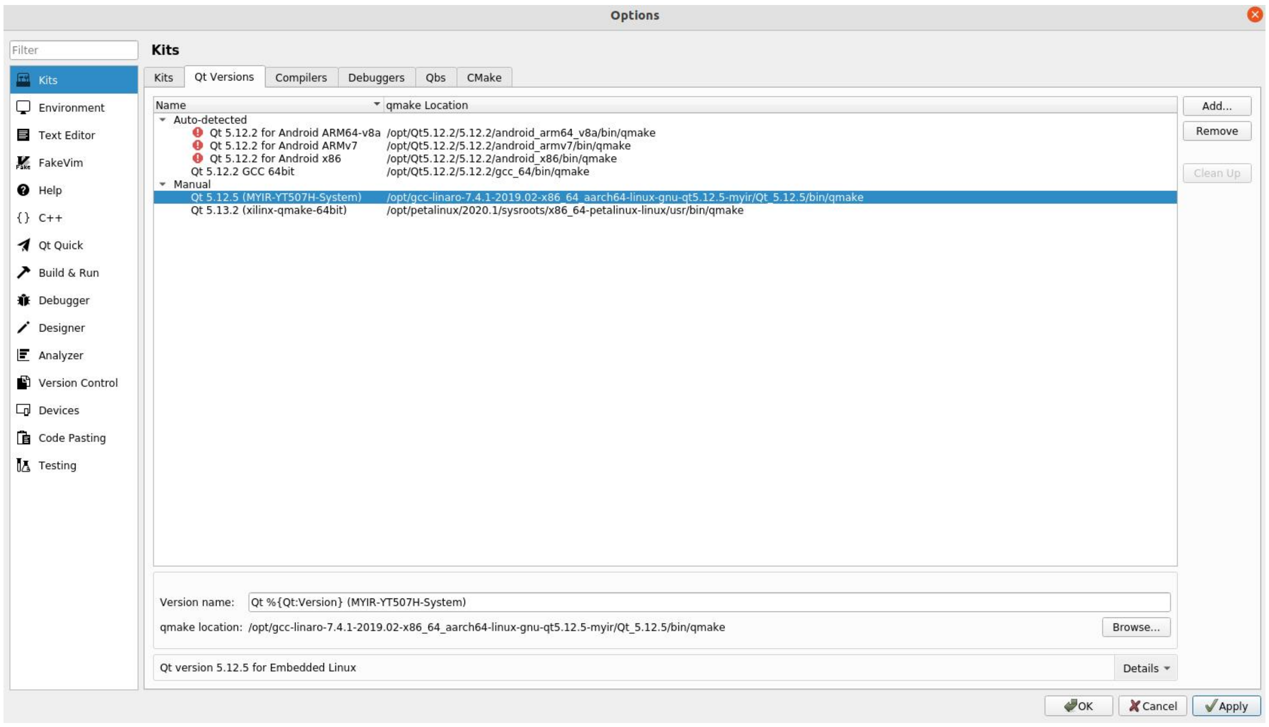
Platform codegen flags:

Platform linker flags:

ABI: arm-linux-generic-elf-64bit | arm - linux - generic - elf - 64bit

Figure 5-6.Selecting the compiler path

4) Select the "Qt Version" TAB and click "Add..." on the right., will pop up qmake path selection dialog, here with "/opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-Linux-gnu-qt5.12.5-myir/Qt_5.12.5/bin/qmake" as an example.After selecting the "qmake" file, click the "Open" button.Change Version name" to Qt %{Qt:Version} (MYIR-YT507H-system).Then click the "Apply" button.



```
Name                                  qmake Location
Auto-detected
    Qt 5.12.2 for Android ARM64-v8a   /opt/Qt5.12.2/5.12.2/android_arm64_v8a/bin/qmake
    Qt 5.12.2 for Android ARMv7       /opt/Qt5.12.2/5.12.2/android_armv7/bin/qmake
    Qt 5.12.2 for Android x86         /opt/Qt5.12.2/5.12.2/android_x86/bin/qmake
  Qt 5.12.2 GCC 64bit                 /opt/Qt5.12.2/5.12.2/gcc_64/bin/qmake
Manual
  Qt 5.12.5 (MYIR-YT507H-System)      /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/bin/qmake
  Qt 5.13.2 (xilinx-qmake-64bit)      /opt/petalinux/2020.1/sysroots/x86_64-petalinux-linux/usr/bin/qmake
```

Version name: Qt %{Qt:Version} (MYIR-YT507H-System)

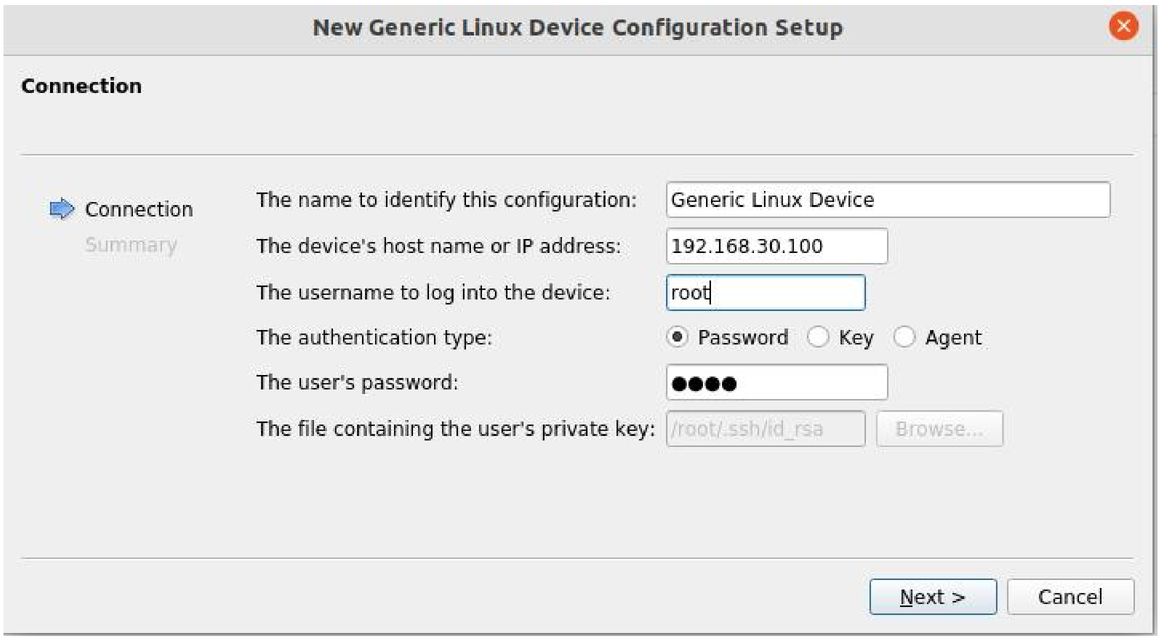
qmake location: /opt/gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu-qt5.12.5-myir/Qt_5.12.5/bin/qmake

Qt version 5.12.5 for Embedded Linux

本文档仅限内部使用

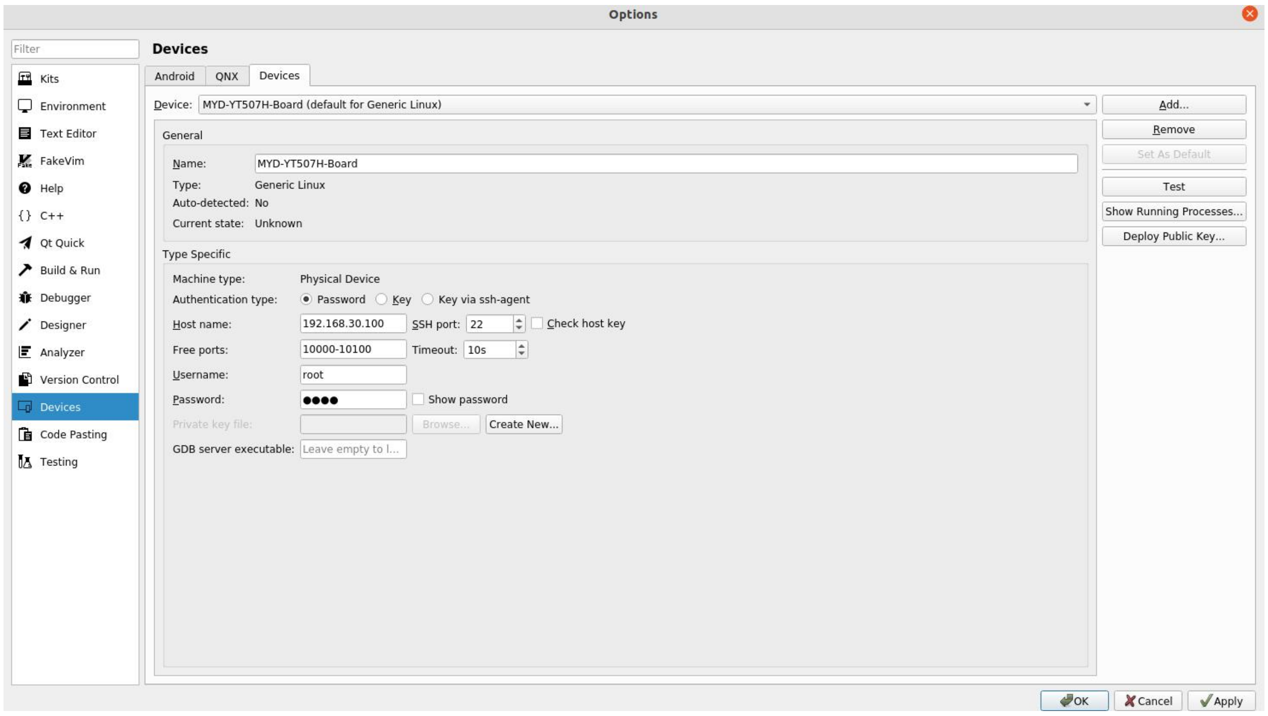
Figure 5-7. Configure the version information

5) Select "Device" on the left and click "Add..." on the right.Button, fill in the content "Name" as "MYD-YT507H-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 proceed with the Next configuration.(This item is optional)



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 5-8. Configure device information



Device: MYD-YT507H-Board (default for Generic Linux)

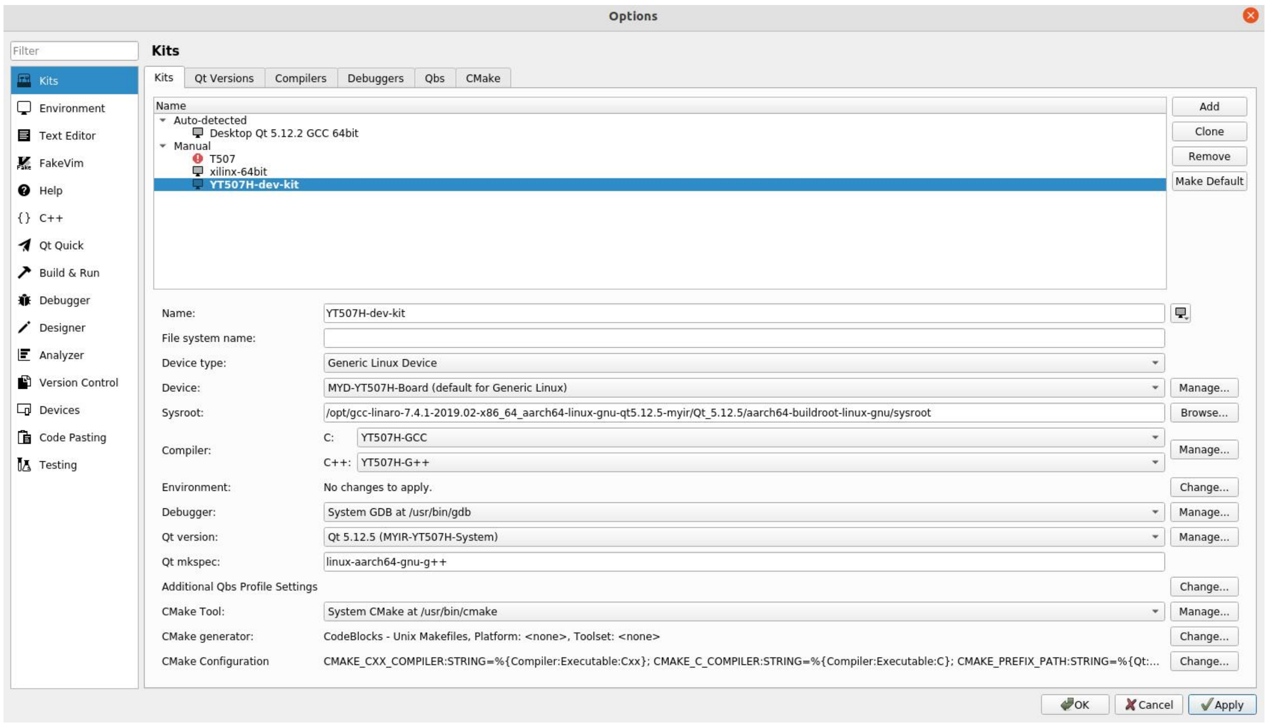
General
Name: MYD-YT507H-Board
Type: Generic Linux
Auto-detected: No
Current state: Unknown

Type Specific
Machine type: Physical Device
Authentication type: [✓ Password] Key Key via ssh-agent
Host name: 192.168.30.100 SSH port: 22 Check host key
Free ports: 10000-10100 Timeout: 10s
Username: root
Password: ●●●● Show password
Private key file:
GDB server executable: Leave empty to l...

Figure 5-9. Complete device configuration

本文档仅限内部使用

6) Click on the left side of the "Build & Run back to" Kits "tag", "Name" is set to "YT507H-dev-kit", "Device" selected "MYD-YT507H-Board" option."Sysroot "select a target device system directory, Here with "/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" for Case."Compiler" selects the previously configured name "YT507H-G++", "Qt version" selects the previously configured name "Qt 5.12.5 (MYIR-YT507H-System)", Set "Qt mkspec" to "linux-aarch64-gnu-g++" (optional).Then click the "Apply" and "OK" buttons.



Name
- Auto-detected
  - Desktop Qt 5.12.2 GCC 64bit
- Manual
  - T507
  - xilinx-64bit
  - YT507H-dev-kit

Name:	YT507H-dev-kit
File system name:	
Device type:	Generic Linux Device
Device:	MYD-YT507H-Board (default for Generic Linux)
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
Compiler:	C: YT507H-GCC C++: YT507H-G++
Environment:	No changes to apply.
Debugger:	System GDB at /usr/bin/gdb
Qt version:	Qt 5.12.5 (MYIR-YT507H-System)
Qt mkspec:	linux-aarch64-gnu-g++
Additional Qbs Profile Settings	
CMake Tool:	System CMake at /usr/bin/cmake
CMake generator:	CodeBlocks - Unix Makefiles, Platform: <none>, Toolset: <none>
CMake Configuration	CMAKE_CXX_COMPILER:STRING=%{Compiler:Executable:Cxx}; CMAKE_C_COMPILER:STRING=%{Compiler:Executable:C}; CMAKE_PREFIX_PATH:STRING=%{Qt:...

Figure 5-10. Configuring development board information

At this point, the development environment of QT is completed, and the subsequent development of QT applications can be carried out.

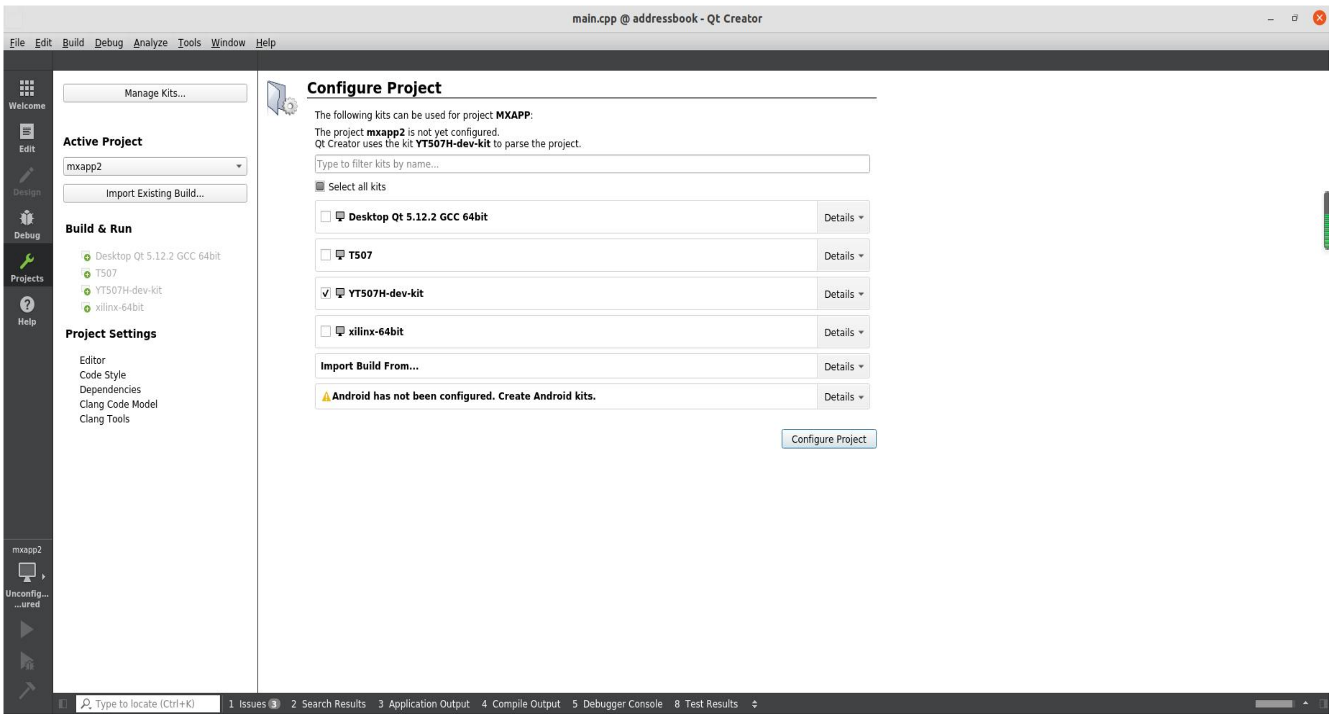
本文档仅限内部使用

5.3. Measy HMI2.0 compilation

Copy mxapp2.tar.gz to a working directory under Ubuntu and unzip the source code.This routine can be compiled by configuring it to the appropriate compiler suite.

In the menu bar, choose "File"->"Open File or Project". In the dialog box that opens, browse to the directory of "MXAPP" routines, select the "mxapp2.pro" File, and click the "Open" button.

After the project is opened, select the "Projects" icon in the left menu column, and switch to the Manage Kits interface in the right interface. Under the "Build & Run" TAB, select the Kit of the "YT507H-dev-Kit" option. The project will then build the application using the associated configuration kit for "YT507H-dev-Kit".



Configure Project

The following kits can be used for project **MXAPP**:

The project **mxapp2** is not yet configured.
Qt Creator uses the kit **YT507H-dev-kit** to parse the project.

Type to filter kits by name...

Select all kits

- [ ] **Desktop Qt 5.12.2 GCC 64bit** — Details
- [ ] **T507** — Details
- [✓] **YT507H-dev-kit** — Details
- [ ] **xilinx-64bit** — Details

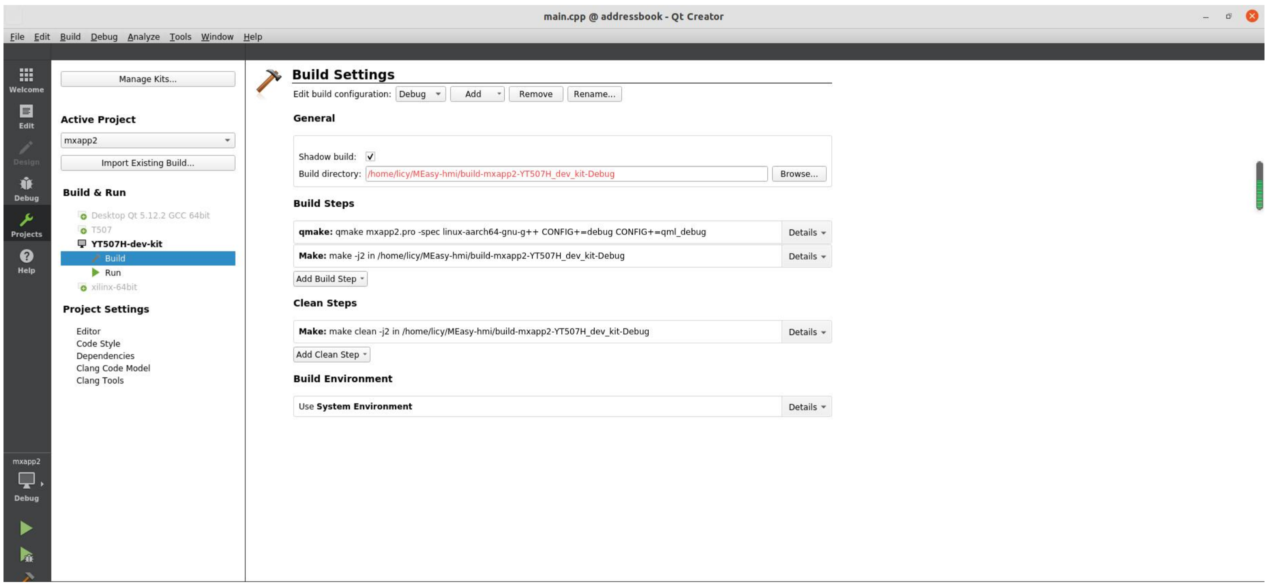
Import Build From... — Details

⚠ **Android has not been configured. Create Android kits.** — Details

Configure Project

Figure 5-10. Opening the project

本文档仅限内部使用



Active Project: mxapp2

Build & Run / **Project Settings**

Build Settings

Edit build configuration: Debug

General

Shadow build: ✓

Build directory: /home/licy/MEasy-hmi/build-mxapp2-YT507H_dev_kit-Debug

Build Steps

```
qmake: qmake mxapp2.pro -spec linux-aarch64-gnu-g++ CONFIG+=debug CONFIG+=qml_debug
Make: make -j2 in /home/licy/MEasy-hmi/build-mxapp2-YT507H_dev_kit-Debug
```

Clean Steps

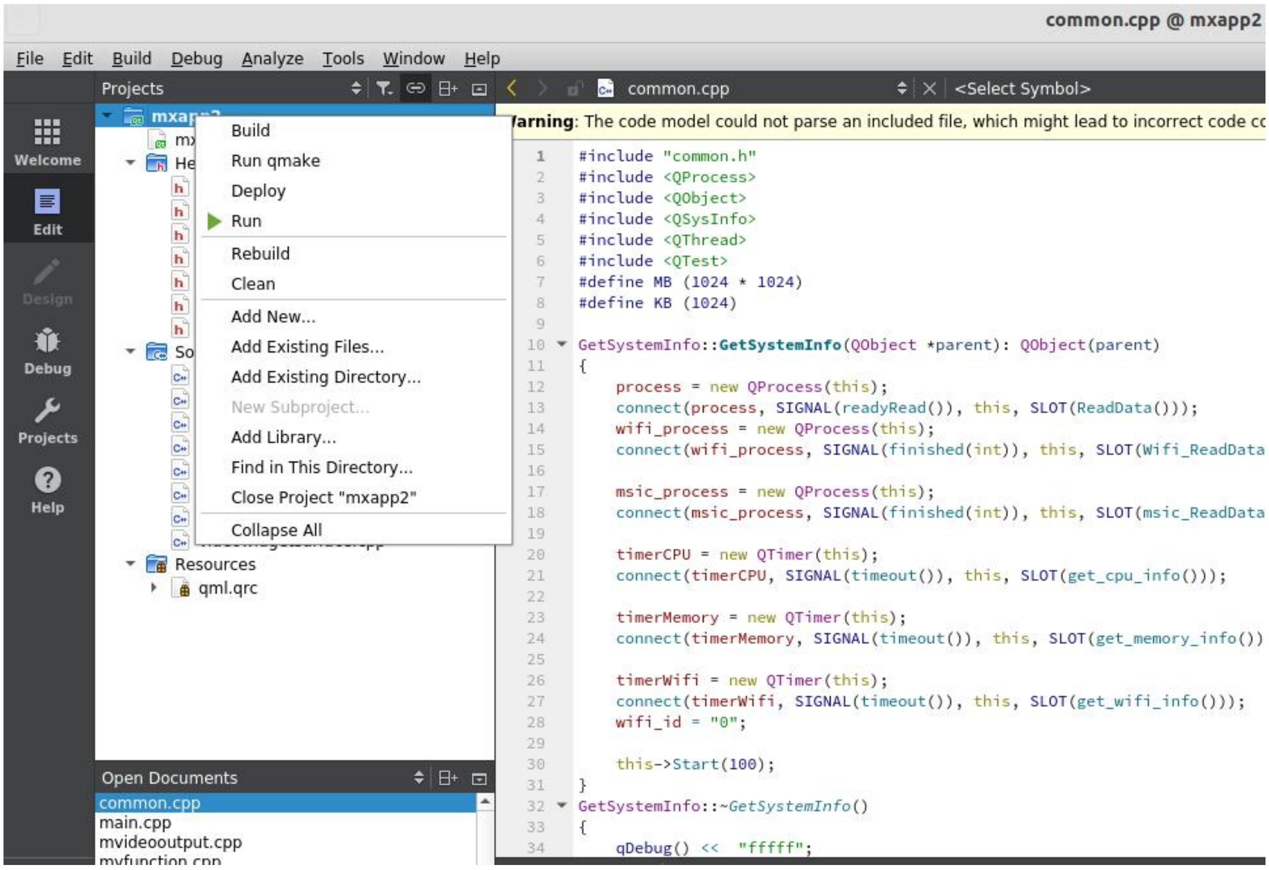
```
Make: make clean -j2 in /home/licy/MEasy-hmi/build-mxapp2-YT507H_dev_kit-Debug
```

Build Environment

Use **System Environment**

Figure 5-11. Select the configured Kit

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

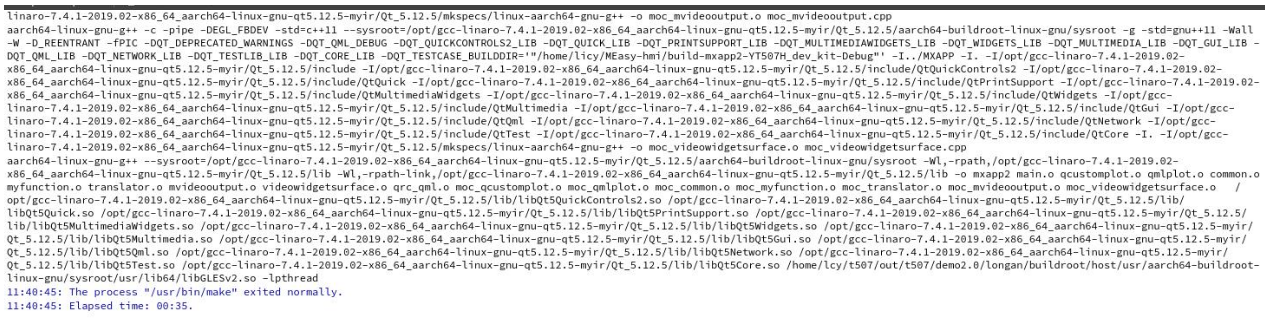


Warning: The code model could not parse an included file, which might lead to incorrect code c

```
#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 5-12.Compiling the project

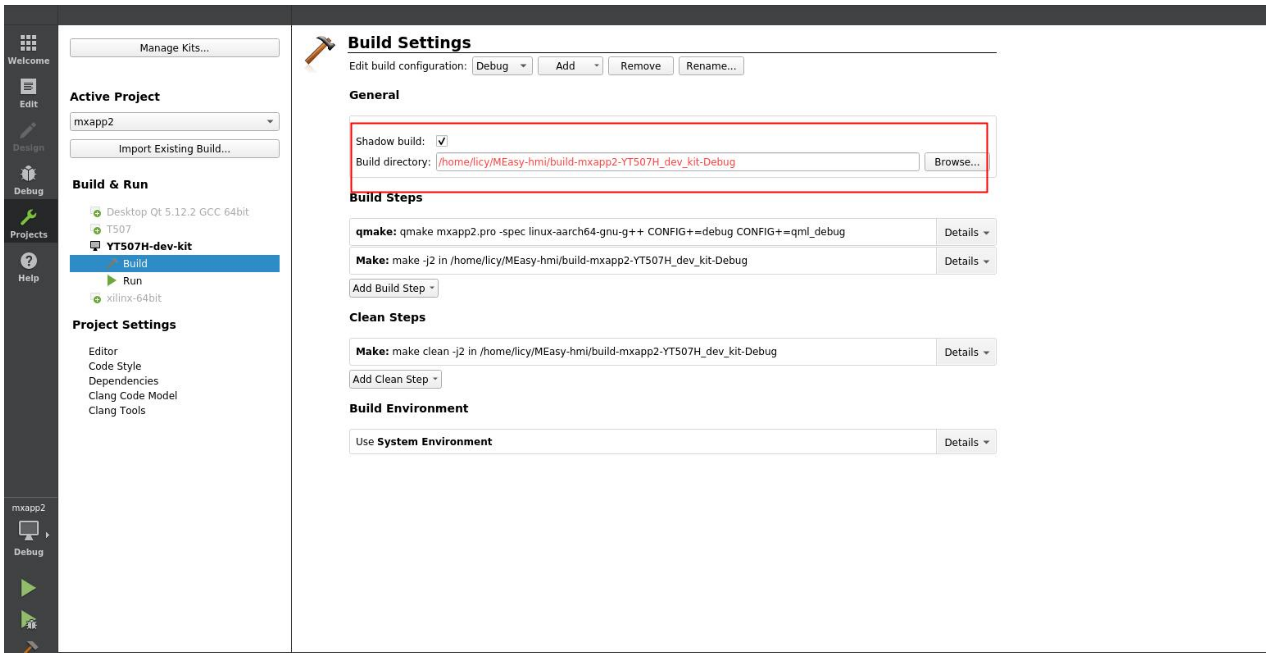
本文档仅限内部使用



```
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 5.13.Compilation completed

After QtCreator build mxapp2 program, compiled binary files stored in the specified directory.Then mxapp2 file copy operation under the development board.



Build Settings

Edit build configuration: Debug | Add | Remove | Rename...

General

Shadow build: ✓

Build directory: /home/licy/MEasy-hmi/build-mxapp2-YT507H_dev_kit-Debug Browse...

Build Steps

qmake: qmake mxapp2.pro -spec linux-aarch64-gnu-g++ CONFIG+=debug CONFIG+=qml_debug Details

Make: make -j2 in /home/licy/MEasy-hmi/build-mxapp2-YT507H_dev_kit-Debug Details

Add Build Step

Clean Steps

Make: make clean -j2 in /home/licy/MEasy-hmi/build-mxapp2-YT507H_dev_kit-Debug Details

Add Clean Step

Build Environment

Use **System Environment** Details

Figure 5-14. The output directory

本文档仅限内部使用

5.4.The SDK integrates applications

When, after the completion of the application development requires and system integration;Since the launch of turning operations.

The first copy the QT application source code to the platform/framework/auto/qt_demo directory

And this directory will be build. Sh increase the following configuration code

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

Clean. Sh directory add the following code

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

Increase in MXAPP directory compile a control script file makeMXAPP (named with the build.sh) is added, the code is as follows:

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

Added to the boot from the start.Boot script path is: the platform/framework/auto/rootfs/etc/qtenv.sh

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

本文档仅限内部使用

```
        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=ft5x_ts

        for InputDevices in /sys/class/input/input*
        do
             DeviceName=`cat $InputDevices/name`
             if [ $DeviceName == $TouchDevice ];then
               TouchDeviceNum=${InputDevices##*input}
               export TSLIB_CONSOLEDEVICE=none
               export QT_QPA_EVDEV_TOUCHSCREEN_PARAMETERS=/dev/input/ev
ent$TouchDeviceNum
               echo "add "/dev/input/event$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=eglfs
        export QT_QPA_GENERIC_PLUGINS=evdevtouch
        export QT_QPA_EGLFS_INTEGRATION=eglfs_mali
        #export QT_QPA_FB_HIDECURSOR=1
```

本文档仅限内部使用

```
        #export QT_QPA_EGLFS_HIDECURSOR=1
        #export QT_QPA_EGLFS_ROTATION=90

        export QWS_MOUSE_PROTO=IntelliMouse:/dev/input/mouse0
        export DBUS_SESSION_BUS_ADDRESS=`cat /tmp/dbusaddr`
        mkdir -p /dev/shm
        ulimit -c unlimited
        #debug Launcher &
        #hellogles3 &
         mxapp2 &
        echo "find qt5 installed done"
fi
```

- /dev/fb0: indicates the first interface to be displayed
- Ft5x_ts: indicates the name of the touch screen interface
- Mxapp2 : Application startup name

At this point, QT application development and integration are complete. Run the following command in the SDK directory to package the production image

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

For details, see the relevant chapters of MYD-YT507H Linux System Development Guide

本文档仅限内部使用

6. References

https://ubuntu.com/download/desktop

https://www.qt.io/

本文档仅限内部使用

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

本文档仅限内部使用

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:

本文档仅限内部使用

- 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.

本文档仅限内部使用

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

本文档仅限内部使用

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

本文档仅限内部使用