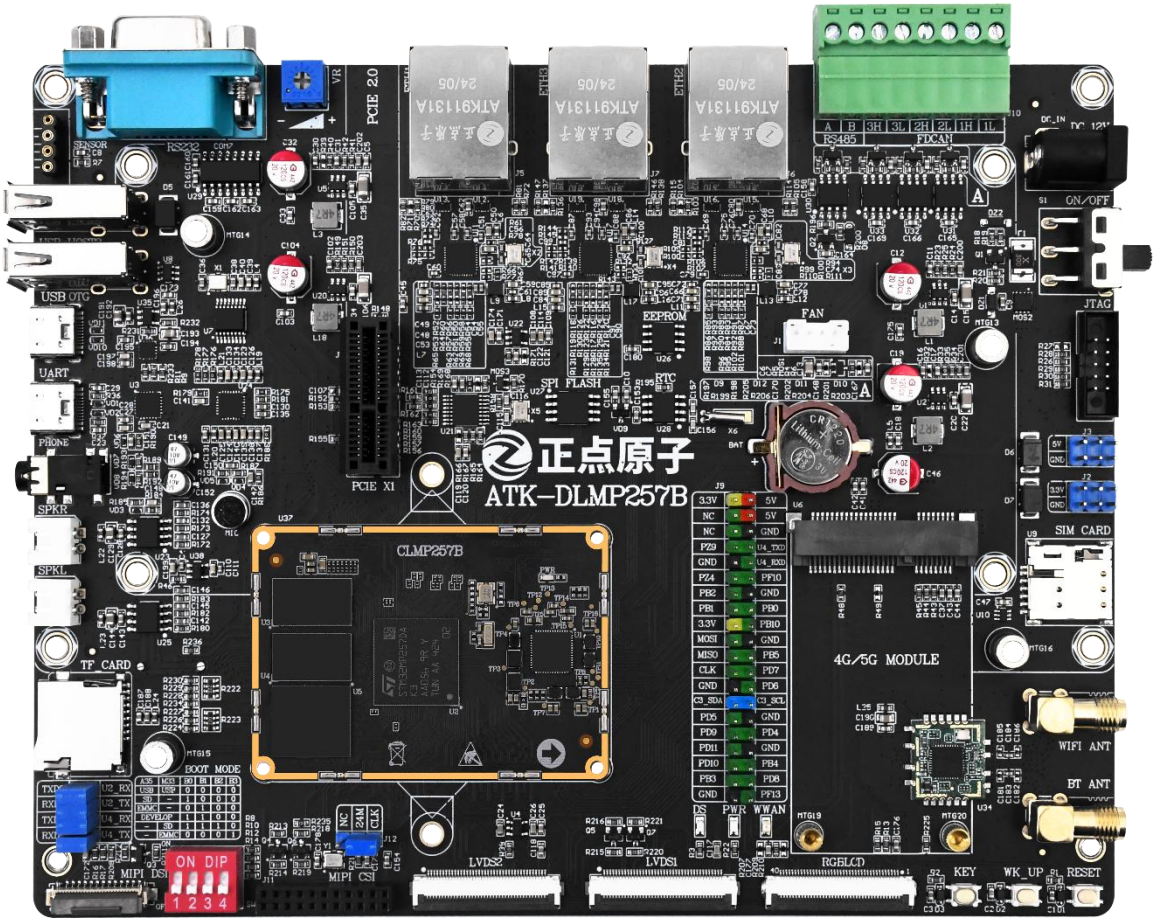
ATK-DLMP257B

Build Qt Environment Based on the factory system

V1.0



ALIENTEK

1. Shopping:

TMALL: https://zhengdianyuanzi.tmall.com

TAOBAO: https://openedv.taobao.com

2. Download

Address: http://www.openedv.com/docs/index.html

3. FAE

Website : www.alientek.com

Forum : http://www.openedv.com/forum.php

Videos : www.yuanzige.com

Fax : +86 - 20 - 36773971

Phone : +86 - 20 - 38271790

Disclaimer

The product specifications and instructions mentioned in this document are for reference only and subject to update without prior notice; Unless otherwise agreed, this document is intended as a product guide only, and none of the representations made herein constitutes a warranty of any kind.

In order to get the latest version of product information, please regularly visit the download center or contact the customer service of Taobao ALIENTEK flagship store. Thank you for your tolerance and support.

Revision History:

Version	Version Update Notes	Responsible person	Proofreading	Date
V1.0	release officially	ALIENTEK	ALIENTEK	2025.04.01

Catalogue

Introduction

This document is based on the factory system to build Qt development environment.

Chapter 1. Installing Qt on Ubuntu

Please check the network disk information under disk A basic information /10_user_manual / [ALIENTEK] Ubuntu to install Qt install manual V1.0.pdf.

Chapter 2. Installing the cross-compilation toolchain

Please check the network disk information under disk basic information /10_user_manual / [ALIENTEK] ATK-DLMP257B **I**nstalls the **C**ross-compilation **T**oolchain **B**ased on the factory system V1.0.pdf.

Chapter 3. Set up the Qt development environment

In this chapter, we will set up a cross-compile environment for Qt.

1. Compile Qt from the command line
2. Build a compilation suite to compile Qt projects

3.1 Command-line cross-compilation of Qt projects

Each time you use the compiler in a new terminal, you need to execute the following command:

```
source /opt/st/stm32mp2/5.0.3-snapshot/environment-setup-cortexa35-ostl-linux
```

3.1.1 See qmake and Qt versions

```
qmake -v
```

```
alientek@alientek:~/01_hello_world$ qmake -v
QMake version 3.1
Using Qt version 5.15.13 in /opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/lib
alientek@alientek:~/01_hello_world$
```

3.1.2 Building the Qt project

The following picture, the author has a 01_hello_world Qt project, in this path there is a 01_hello_world.pro file, this is the Qt project file. We use qmake to interpret it, and the Makefile will be generated for compilation.

```
alientek@alientek:~/01_hello_world$ ls
01_hello_world.pro  01_hello_world.pro.user  main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui
alientek@alientek:~/01_hello_world$
```

When there is only one pro file, qmake can be executed directly, and when there are multiple pros, qmake xx.pro.

```
qmake
```

```
alientek@alientek:~/01_hello_world$ qmake
Info: creating stash file /home/alientek/01_hello_world/.qmake.stash
alientek@alientek:~/01_hello_world$ ls
01_hello_world.pro  01_hello_world.pro.user  main.cpp  mainwindow.cpp  mainwindow.h  mainwindow.ui  Makefile
alientek@alientek:~/01_hello_world$
```

Then directly execute make, if there is no make command then install make first, sudo apt install make.

```
make -j 16 # The number after -j speeds up compilation
```

```
alientek@alientek:~/01_hello_world$ make -j 16
/opt/st/stm32mp2/5.0.3-snapshot/sysroots/x86_64-ostl_sdk-linux/usr/bin/uic mainwindow.ui -o ui_mainwindow.h
aarch64-ostl-linux-g++  -mcpu=cortex-a35+crc -mbranch-protection=standard --sysroot=/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux -c -pi
pe  -O2 -pipe -g -feliminate-unused-debug-types  --sysroot=/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux -O2 -std=gnu++11 -Wall -Wextra
-D_REENTRANT -fPIC -DQT_NO_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I. -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/inc
lude/QtWidgets -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/include/QtGui -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-os
tl-linux/usr/include/QtCore -I. -I. -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/lib/mkspecs/linux-oe-g++ -o main.o main.cpp
aarch64-ostl-linux-g++  -mcpu=cortex-a35+crc -mbranch-protection=standard --sysroot=/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux -pipe
 -O2 -pipe -g -feliminate-unused-debug-types  --sysroot=/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux -O2 -std=gnu++11 -Wall -Wextra -dM
 -E -o moc_predefs.h /opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/lib/mkspecs/features/data/dummy.cpp
aarch64-ostl-linux-g++  -mcpu=cortex-a35+crc -mbranch-protection=standard --sysroot=/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux -c -pi
pe  -O2 -pipe -g -feliminate-unused-debug-types  --sysroot=/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux -O2 -std=gnu++11 -Wall -Wextra
-D_REENTRANT -fPIC -DQT_NO_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB -I. -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/inc
lude/QtWidgets -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/include/QtGui -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-os
tl-linux/usr/include/QtCore -I. -I. -I/opt/st/stm32mp2/5.0.3-snapshot/sysroots/cortexa35-ostl-linux/usr/lib/mkspecs/linux-oe-g++ -o mainwindow.o mainwin
```

After compiling, the executable file is generated as shown below. Copy this file to the development board directly./01_hello_world!

```
alientek@alientek:~/01_hello_world$ ls
01_hello_world      01_hello_world.pro.user  main.o          mainwindow.h  mainwindow.ui  moc_mainwindow.cpp  moc_predefs.h
01_hello_world.pro  main.cpp                 mainwindow.cpp  mainwindow.o  Makefile       moc_mainwindow.o    ui_mainwindow.h
alientek@alientek:~/01_hello_world$
```

3.2 QtCreator to develop embedded Qt applications

Many developers write Qt applications and want to try them out on the development board. After all, the programs we develop under the Qt Creator on Ubuntu ultimately need to run on the development board. If you just develop Qt programs on Ubuntu, then you don't need to build Qt applications for embedded platforms.

Using the command line has a bad place, after writing Qt project, you need to open the terminal, and then execute the command to compile, which is more troublesome. Is there a way to build a development Kit (Kit) method? Well, no nonsense, let's use Qt Creator to build and develop embedded platform Qt applications.

3.2.1 Configuring the aarch64 cross compiler

First we have to let QtCreator know where our compiler/environment is, so we have to let QtCreator know the compiler environment variables, source /opt/st/stm32mp2/5.0.3-snapshot/environment-setup-cortexa35-ostl-linux should be added to the QtCreator startup script.

sudo vi/opt/Qt5.12.12 / Tools/Qtcreator/bin/Qtcreator. Sh

At the top level of the script, write "source /opt/st/stm32mp2/5.0.3-snapshot/environment-setup-cortexa35-ostl-linux" as shown below.

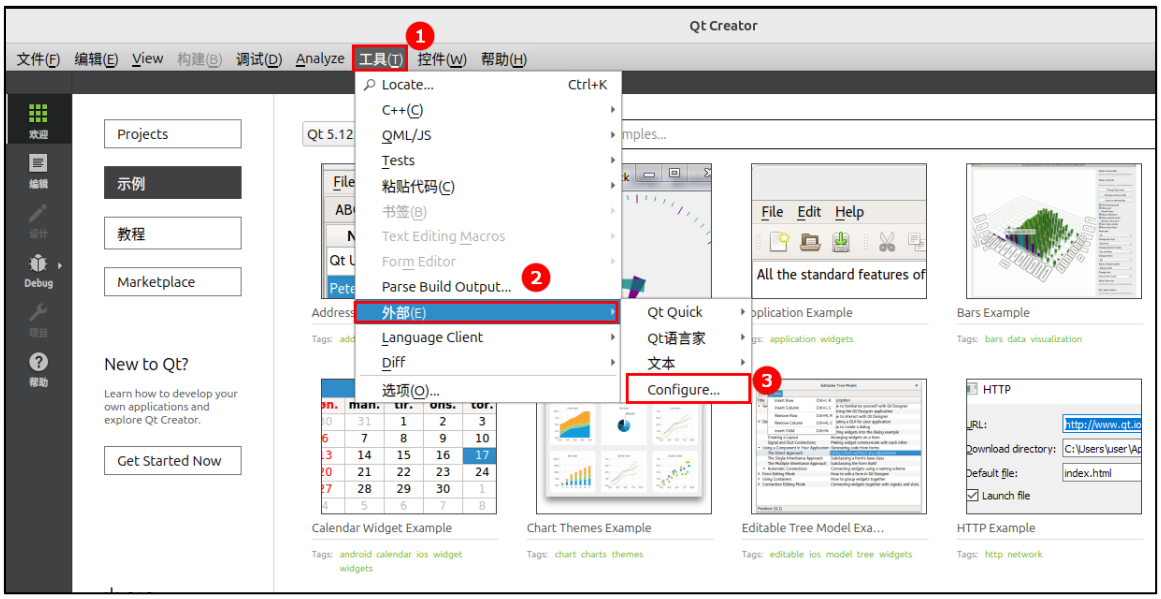
```
source /opt/st/stm32mp2/5.0.3-snapshot/environment-setup-cortexa35-ostl-linux
#! /bin/sh

# Use this script if you add paths to LD_LIBRARY_PATH
# that contain libraries that conflict with the
# libraries that Qt Creator depends on.

makeAbsolute() {
    case $1 in
        /*)
            # already absolute, return it
            echo "$1"
            ;;
        *)
            # relative, prepend $2 made absolute
            echo `makeAbsolute "$2" "$PWD"`/"$1" | sed 's,/\.$,,'
            ;;
    esac
}
```

After saving we have to use the script to start QtCreator, execute below command.

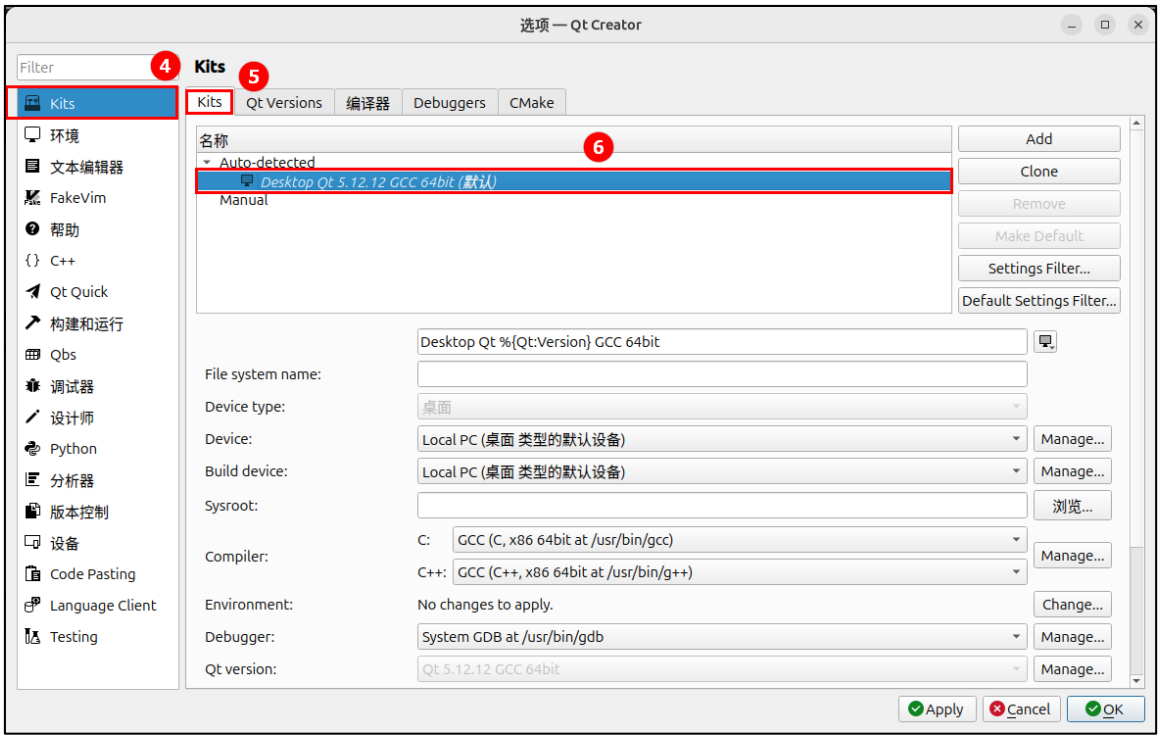
/ opt/Qt5.12.12 / Tools/Qtcreator/bin/Qtcreator. Sh & # & the background

```
alientek@alientek:~$ /opt/Qt5.12.12/Tools/QtCreator/bin/qtcreator.sh &
[1] 29383
alientek@alientek:~$ Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
```

To get started, open Qt Creator on Ubuntu and click "Tools", "External" and "configure" in the top menu bar.

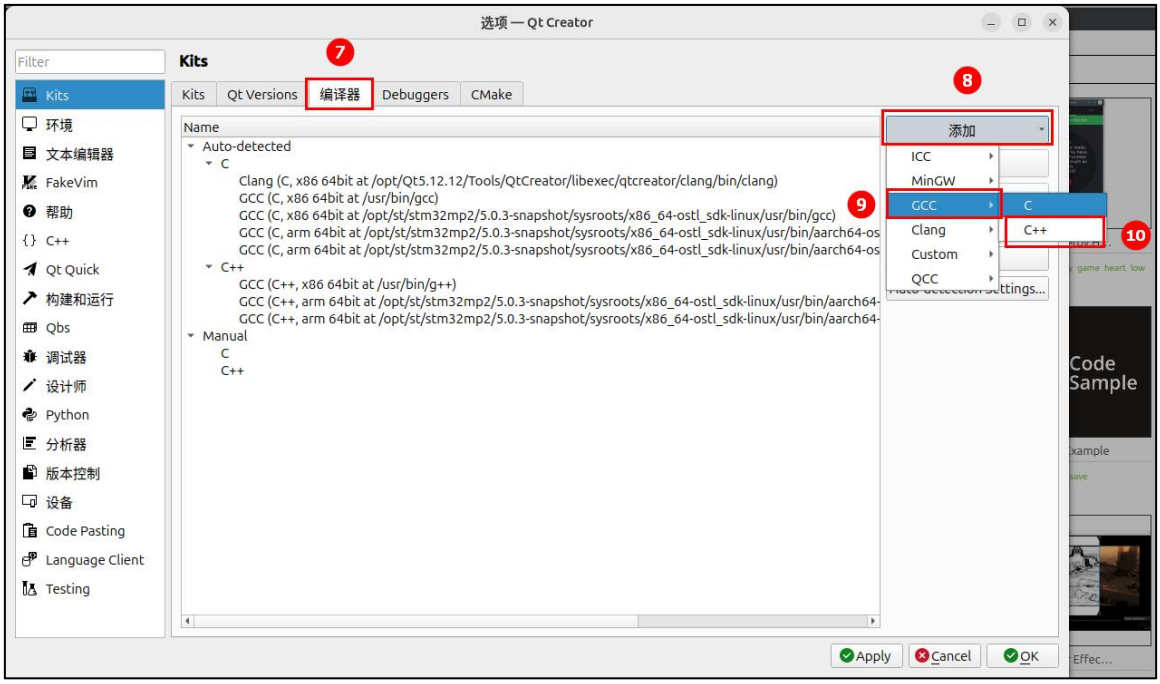


You can see ⑥, this is the Ubuntu desktop application development kit, that is, the Qt application can be run on Ubuntu. Usually we develop interfaces on Ubuntu. If there is some code that does not work on Ubuntu, that is, the code is related to the board hardware, then we have to compile the program to run on the board to see the results.

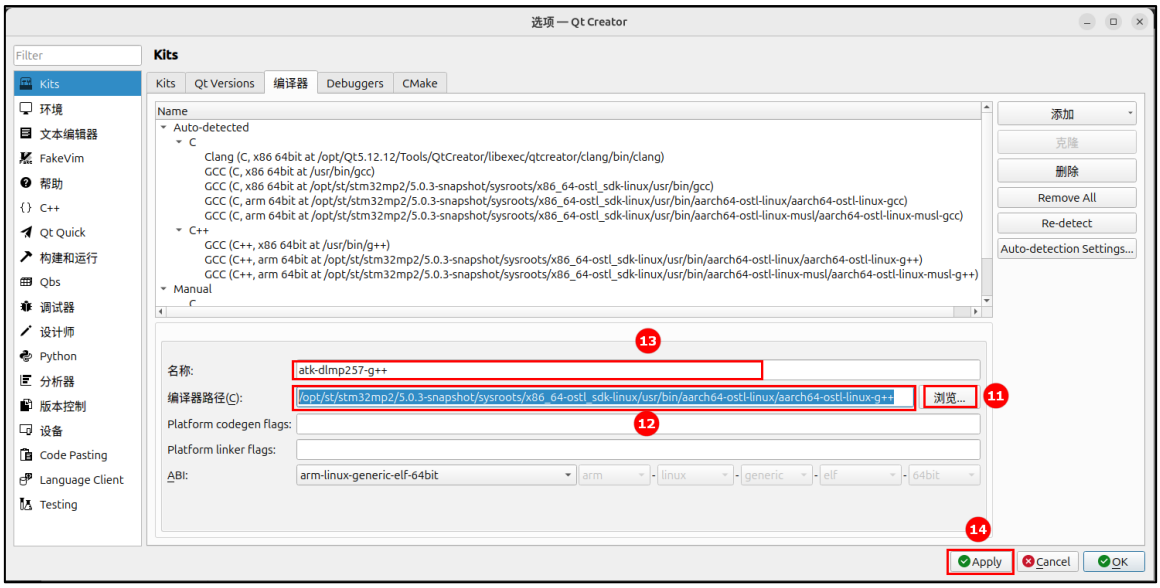


Now that we only have one kit, we need to add our ATK-DLMP257 configuration. Click on the compiler option, you can see that on Ubuntu there are many compilers, C compiler and C++ compiler, this is detected by the system. Then click the Add button to add the cross-compiler for our ATK-DLMP257 platform. In Chapter 2, you installed the cross-compiler. This is where the cross-compiler comes in.

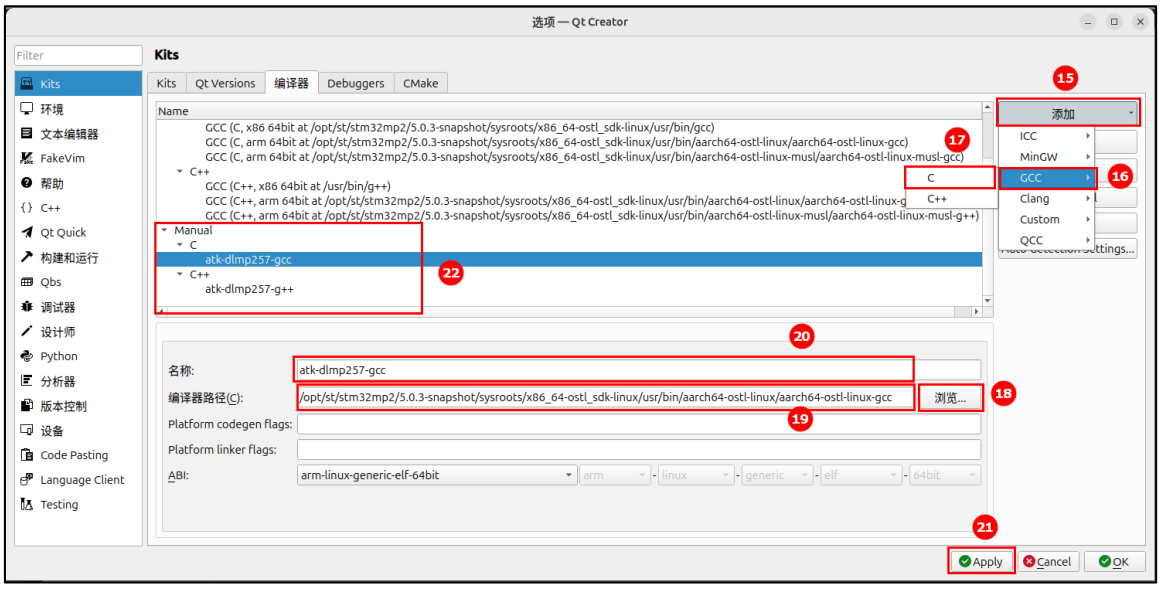
Click on compiler, we choose GCC, why GCC, because the name of the compiler we are using is GCC type, and then we first configure C++, which is the compiler that compiles C++. Note: QtCreator will also automatically detect our compiler! It already knows where our ATK-DLMP257 compiler is, but recognizes two?? We don't know which one? For the sake of completeness, let's do this manually.



After clicking, we need to make the screen that pops up a little bigger. Otherwise, you will lose sight of the alert path button. In the last step, the path is /opt/st/stm32mp2/5.0.3-snapshot/sysroots/x86_64-sdk-linux/usr/bin/aarch64-ostl-linux/aarch64-ostl-linux-g++. If your cross-compiler was installed at the same location as in Chapter 2, our results will be the same. In the last step, we need to define the name of the compiler. Let's call our ATK-DLMP257 development board ATK-DLMP257-G ++! Write LOWERcase, it is easy to recognize, of course, you can customize it for other names!

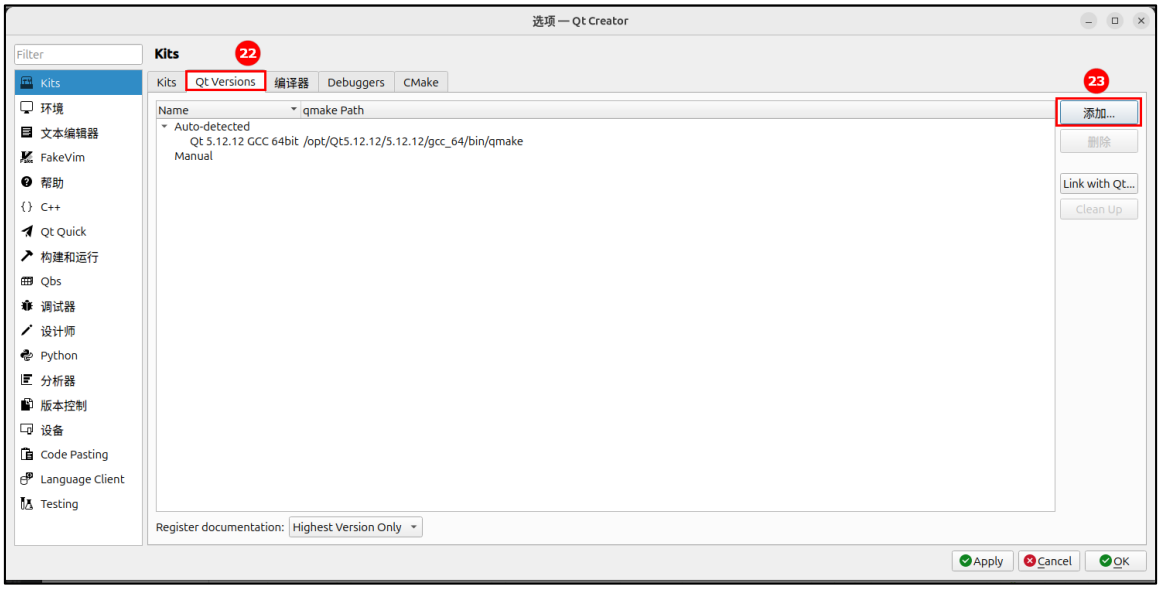


After configuring C++ above, we must point to Apply. Now let's configure C. Repeat the steps above. Add a C compiler. In the same way, gcc The path is /opt/st/stm32mp2/5.0.3-snapshot/sysroots/x86_64-sdk-linux/usr/bin/aarch64-ostl-linux/aarch64-ostl-linux-gcc. The custom name is atk-dlmp257-gcc. Click Apply. As you can see in step 22, we've added two compilers.

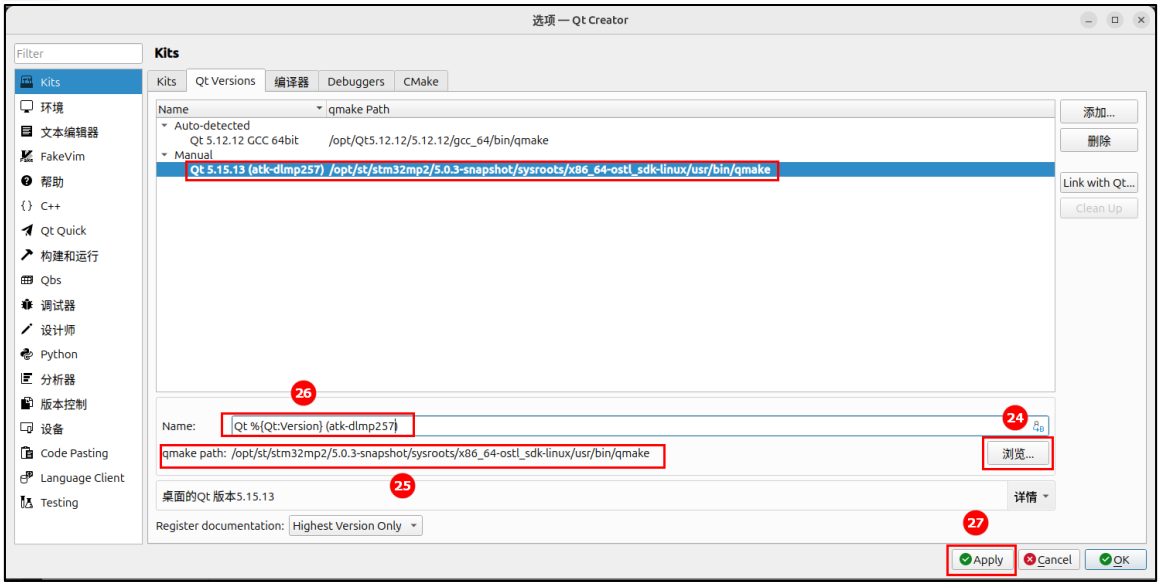


3.2.2 Configuring qmake

Now that you've configured the cross-compiler, let's configure Qmake.You can configure qmake first and then the cross-compiler. Click Qt version, then click Add.



In the following figure, add qmake, if your cross-compiler is the same as my installation path, The path for qmake is /opt/st/stm32mp2/5.0.3-snapshot/sysroots/x86_64-ostl sdk-linux/usr/bin/qmake. We change the name to Qt %{Qt:Version} (atk-dlmp257) at step 26. Hit Apply! Don't forget!



3.2.3 Configuring Kit

Different versions of QtCreator may display different views here, so we will use the unified QtCreator version which is QtCreator5.0.2 from Qt5.12.12.

At this point we need to add our own cross-compiled suite. Click Add. (Note: This screen needs to be zoomed-in with the mouse, otherwise we won't see the full screen.)

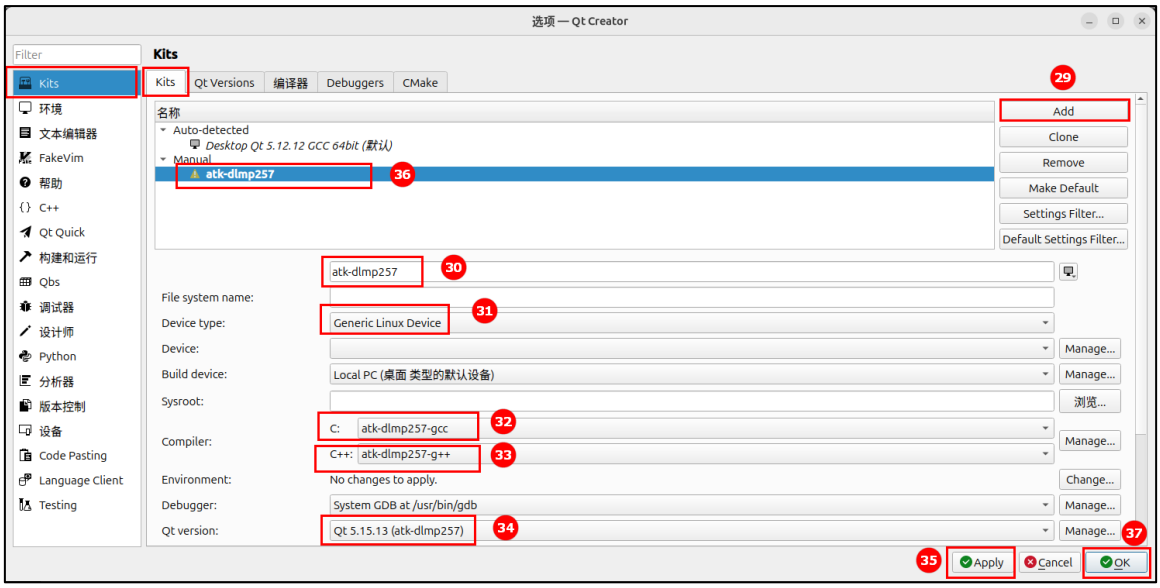
Serial number 30: Here we named atk-dlmp257, this is the name of our development board, write lowercase feel more pleasing to the eye.

Serial number 31: We select Generic Linux Device.

Sequence numbers 32 and 33: We choose the cross-compiler type configured in 3.2.1

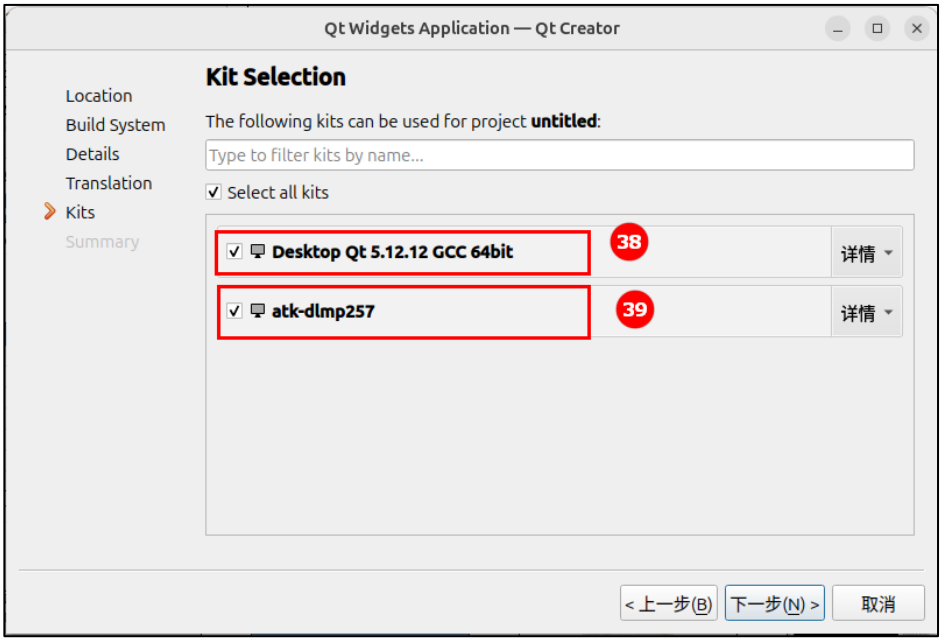
Sequence 34: We choose qmake as configured in 3.2.2

It can be seen that the kit in the following picture only has a yellow exclamation mark (the yellow exclamation mark is just that our development board device has not been configured yet, so the yellow exclamation mark appears), which means that we have successfully configured it without any errors. Then we hit Apply Apply and OK!

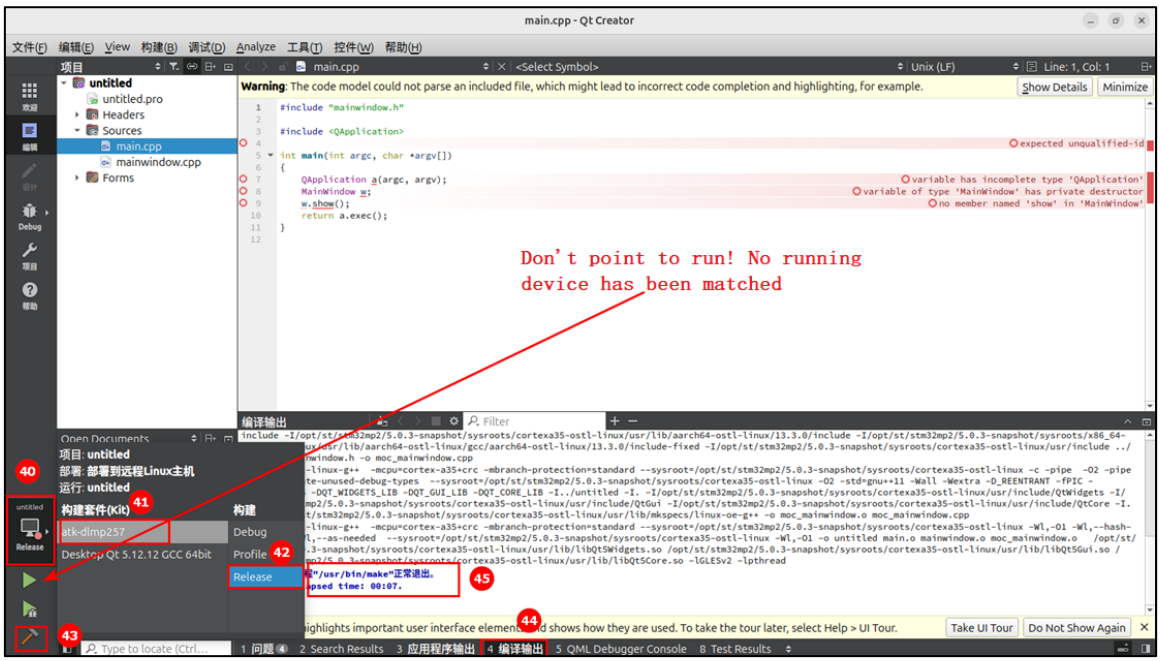


3.2.4 Compiling the tests

Now let's create a simple project to test. I quickly created a project, when we create the project, we need to select the build suite we set up in the previous step. Here we choose both, we can compile Qt applications to run on Ubuntu, and we can compile Qt applications to run on ATK-DLMP257 platform.



Click on the build suite in the bottom left corner, we select atk-dlmp257 build suite, select Release or Debug version, note that we click the small hammer in step 43 to build, do not click Run (green triangle). In the build output window, you can see that step 45 uses the cross compiler we installed in Section 3.2.1 and compiles without errors!



The compiled executable is under build-project name -atk_dlrmp257-Release/. Check it out for yourself. Copy to the development board system to execute!

3.2.5 Deploying Qt applications remotely

In section 3.3.3, remember when we configured kit with a yellow exclamation mark? Because we did not configure the running device, the Qt program you compiled has no device to run, so it reported yellow. Default Desktop Qt 5.12.2 GCC 64bit This compilation suite is configured to run on the running device, which is the Ubuntu desktop, and the programs it compiles will run on the Ubuntu desktop. The atk-dlmp257 compilation suite we configured needs to configure the running device, which is our development board.

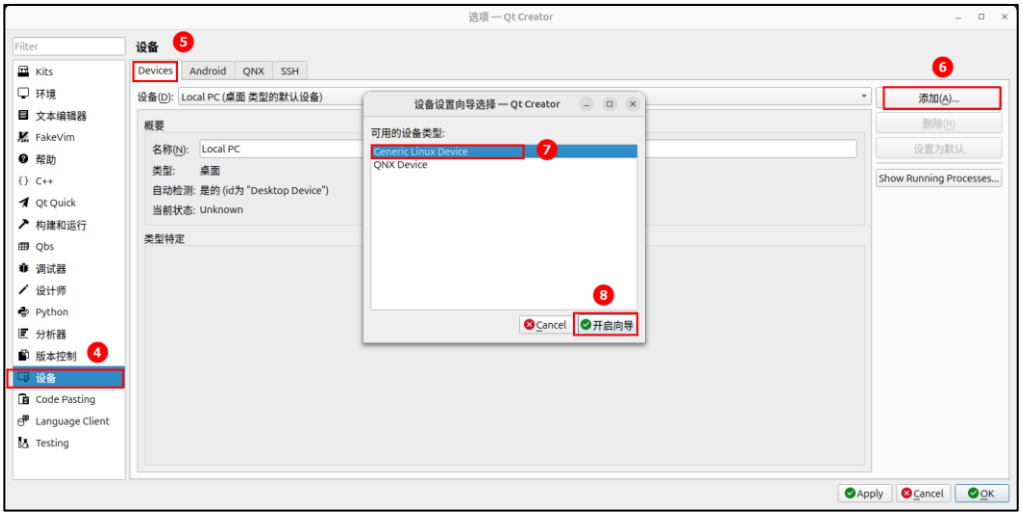
Ok, now let's create the running device. This running device is not on our Ubuntu, so how do we connect our ATK-DLMP257 development board device? The answer is network!

Similarly, we click on the Tools' External 'configuration options entry as follows.

Before creating the device, ensure that the ATK-DLMP257 development board is currently running the factory system and that the ETH2 network port of the development board is connected to the router device through a network cable (no router? Please refer to the Resources user manual to find out how to set the static IP of Ubuntu and the development board, but only the most general ones!) .

To start creating a device, click the device entry and click Add. To add a generic Linux device, perform the following steps:

To start creating a device, click the device entry and click Add. To add a generic Linux device, perform the following steps:

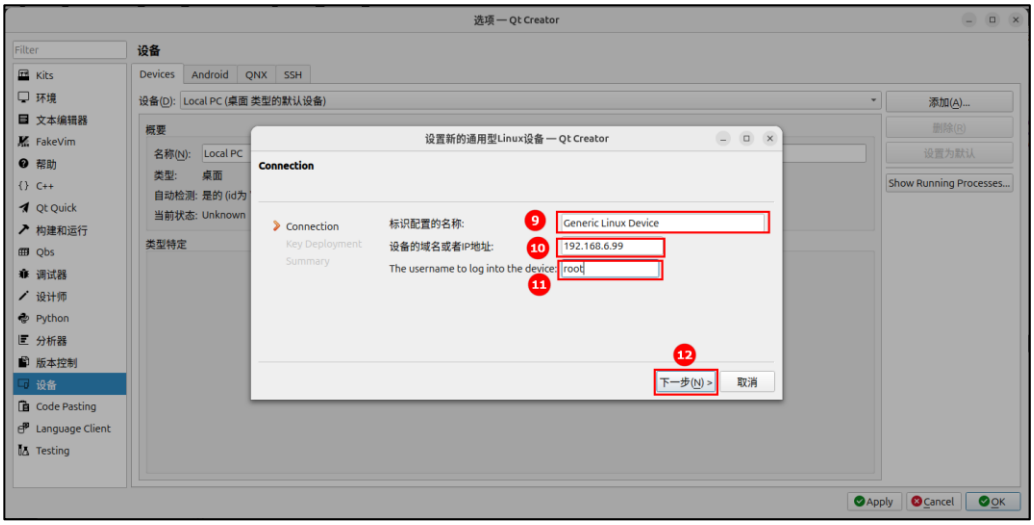


Step 9 Name the device xxx, name whatever you like, the default here.

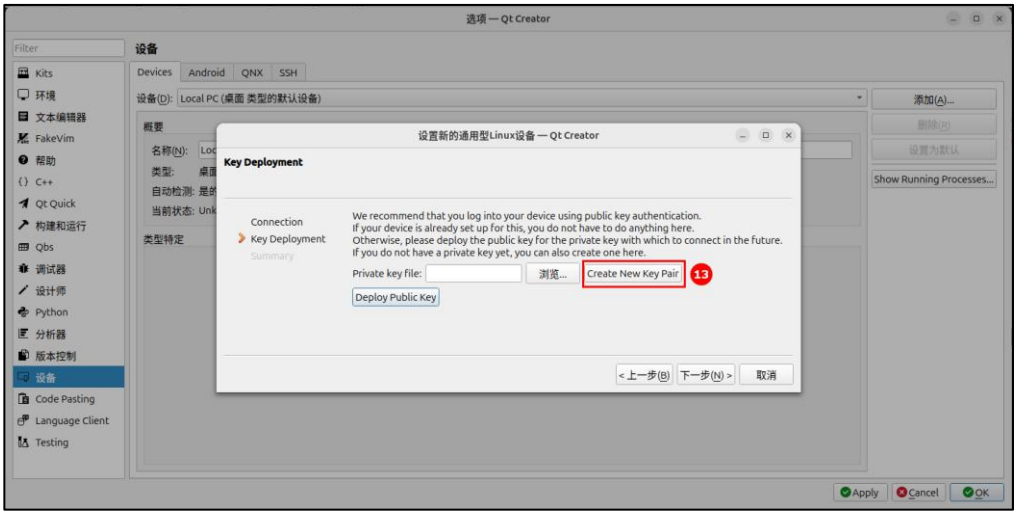
Step ⑩ Fill in the ip address of the board, and fill in the ip address of your board.

The second step is that the device logs on to the development board as the root user. The default board has the root user.

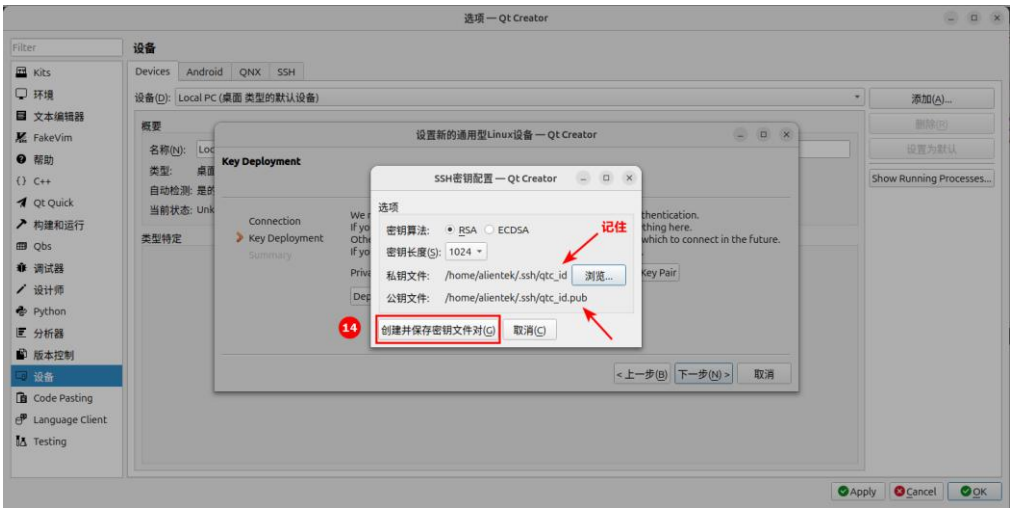
Click Next.



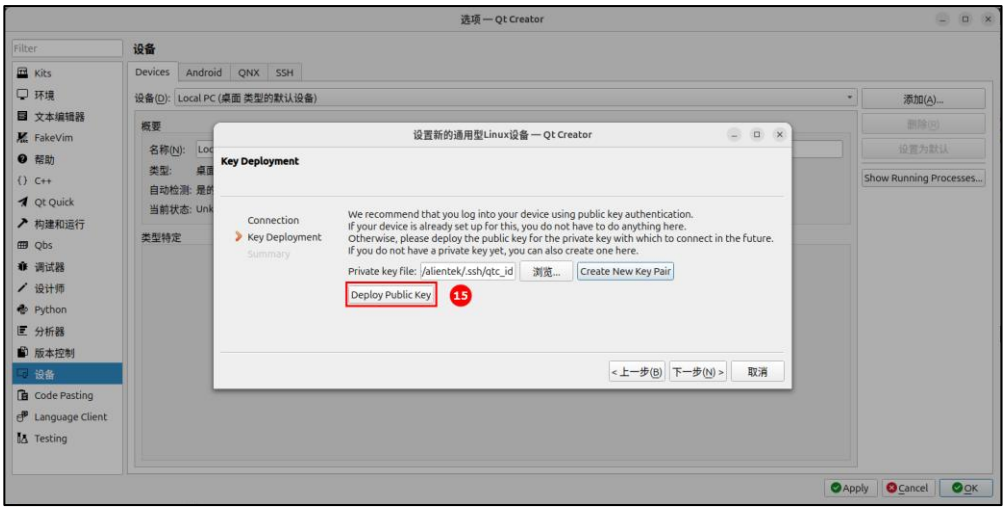
Click Create Key File.



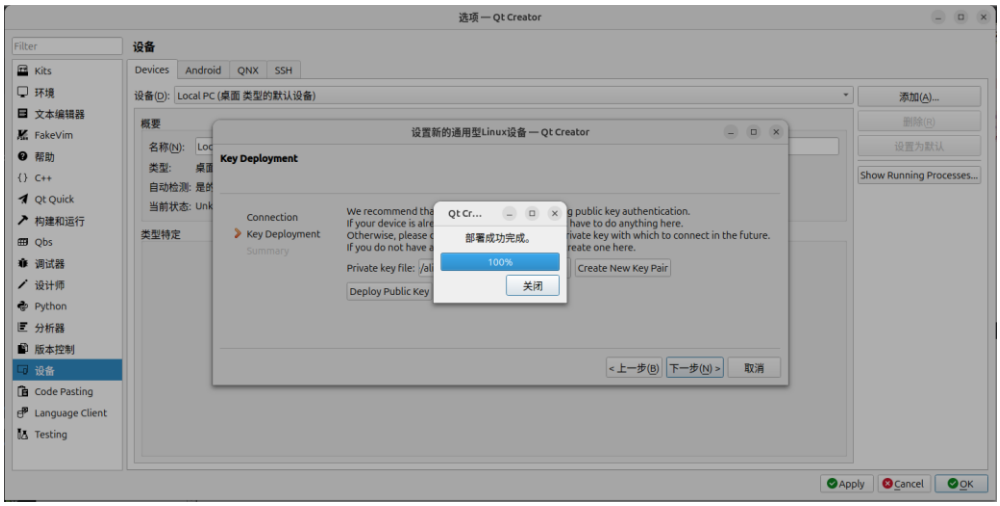
The default option is sufficient, and then click to create a key file pair. Remember the key pair location so you don't have to repeat it when creating a new device.



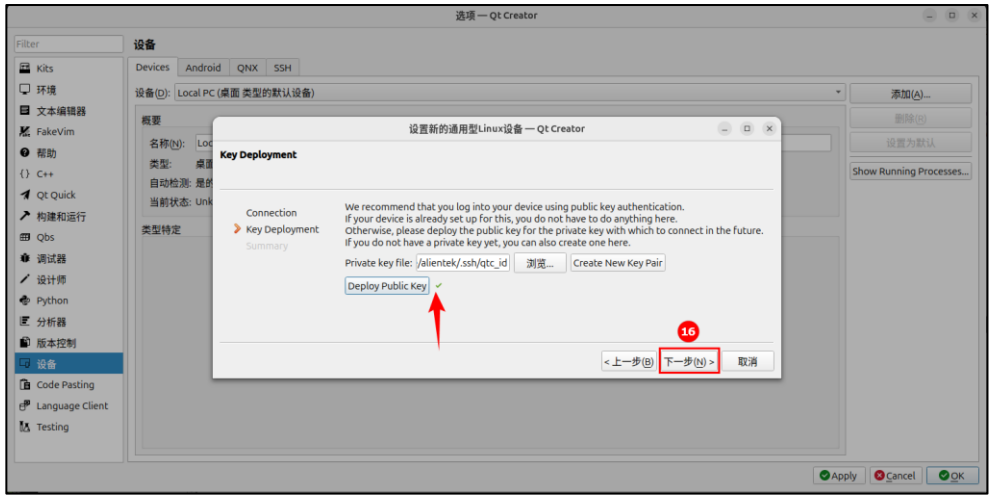
Click Deploy Public Key. In the previous step, we created the key pair, which is like a lock and key. The public key represents the key, which is deployed on the board, and the lock and key are "paired" so that they have permission to communicate.



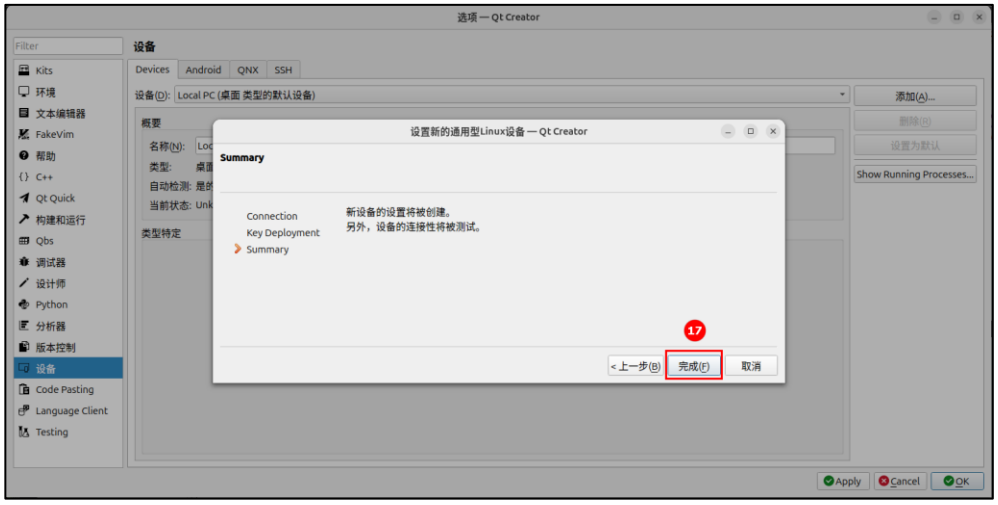
In general, Yocto system is the root user, no password, direct deployment is successful. Click Close.



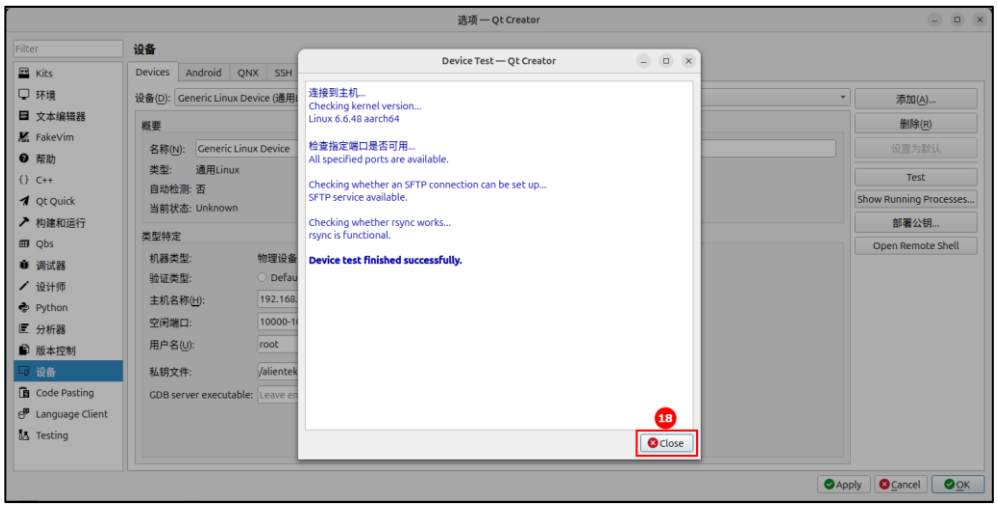
Click Next



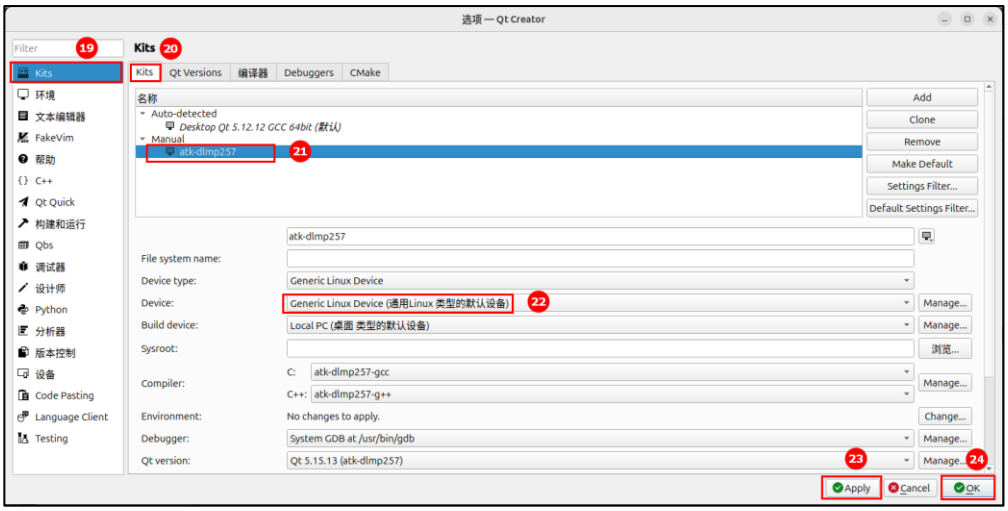
After clicking done, the device connectivity will be tested.



As you can see below, the device test was successful. If there are any errors, please check whether the ip address of the development board is correct, and see (Device test finished successfully) Device test finished successfully! If any other errors can be ignored, click close.



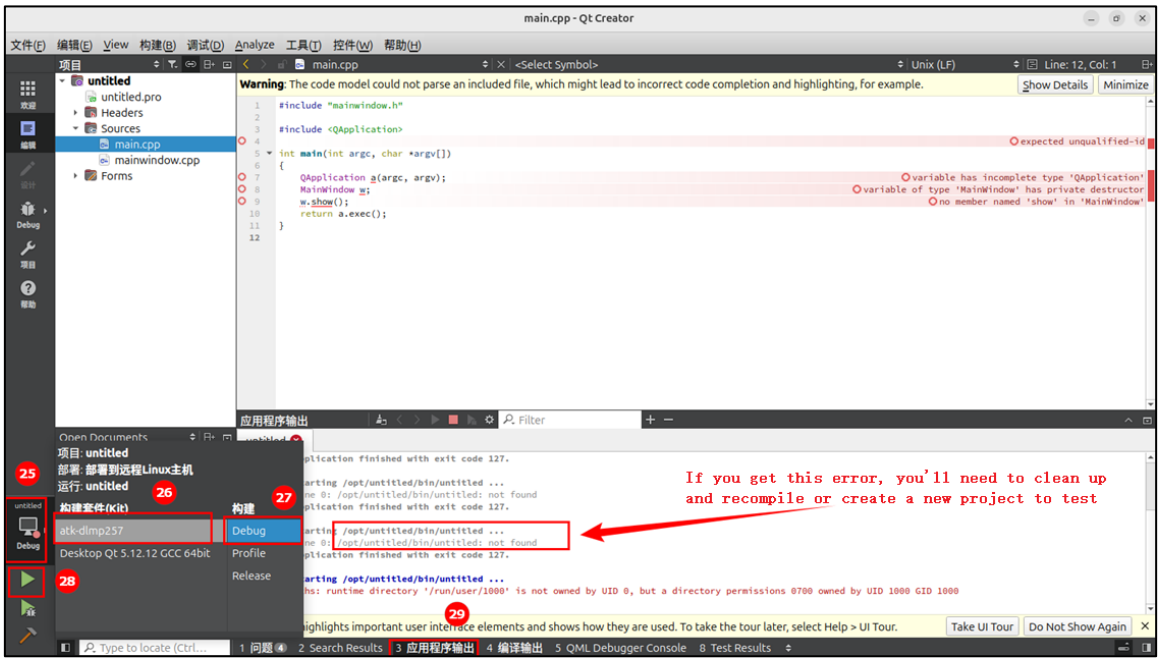
You can see that the small yellow exclamation mark in front of your atk-dlmp257 is gone. In step 22, the system automatically selects the Generic Linux Device we just configured and click OK, which completes the configuration.



3.2.6 Run the device tests

Since the ATK-DLMP257 development board comes with a Qt main interface, please exit the main interface according to your personal needs before running the device test.

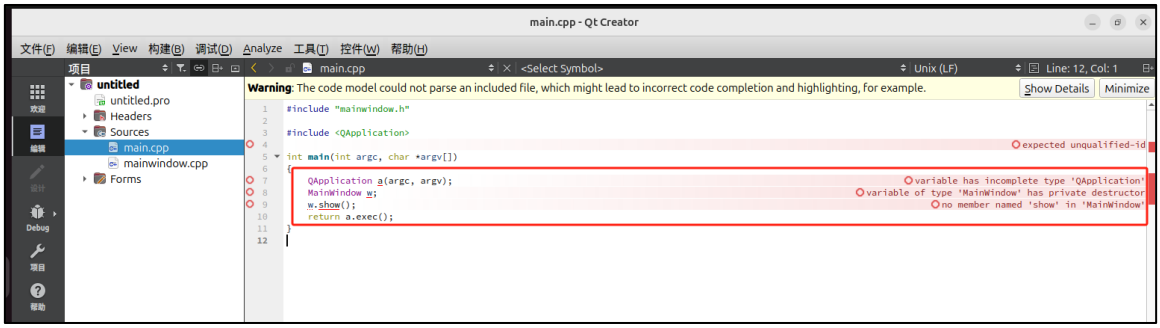
Select our atk-dlmp257 compilation suite in the following figure, select Debug or Release, click the green triangle, or the shortcut key "Ctrl + r" to run our Qt program on the ATK-DLMP257 development board. Since our example is a blank window, you should see a white window with nothing else on the screen. Run successfully in the compiler output window to see the following information in Figure 29. If you encounter a running error, please create a new project and test it as follows.



At this point, the embedded Qt development environment has been built!

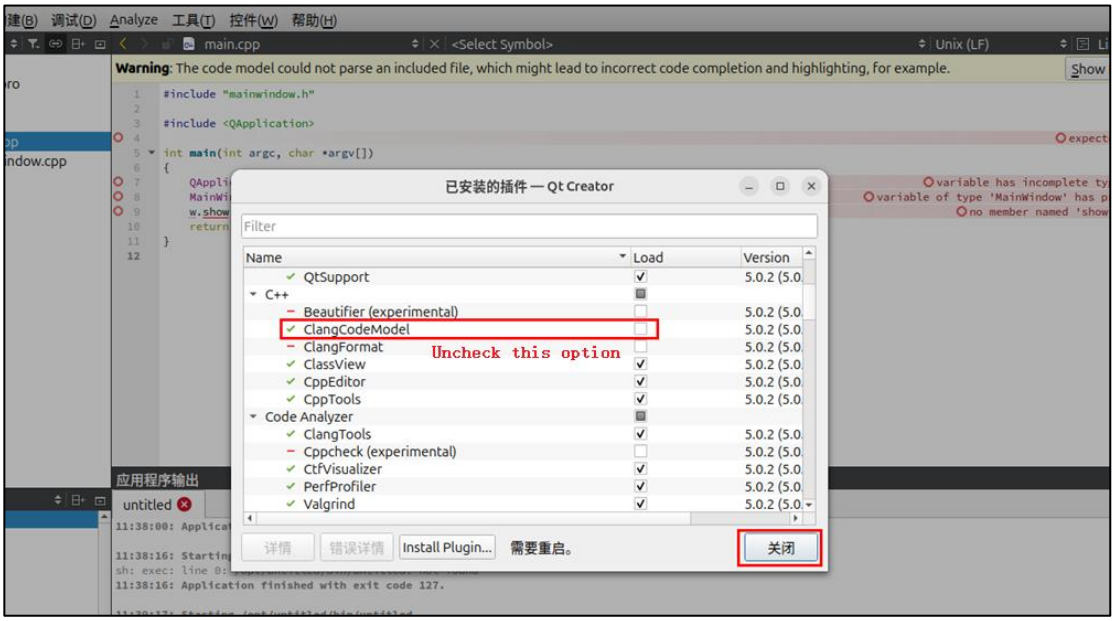
3.3 QtCreator does not recognize member functions problem solved

As shown, choosing atk-dlmp257 to compile the suite will result in a red flag.

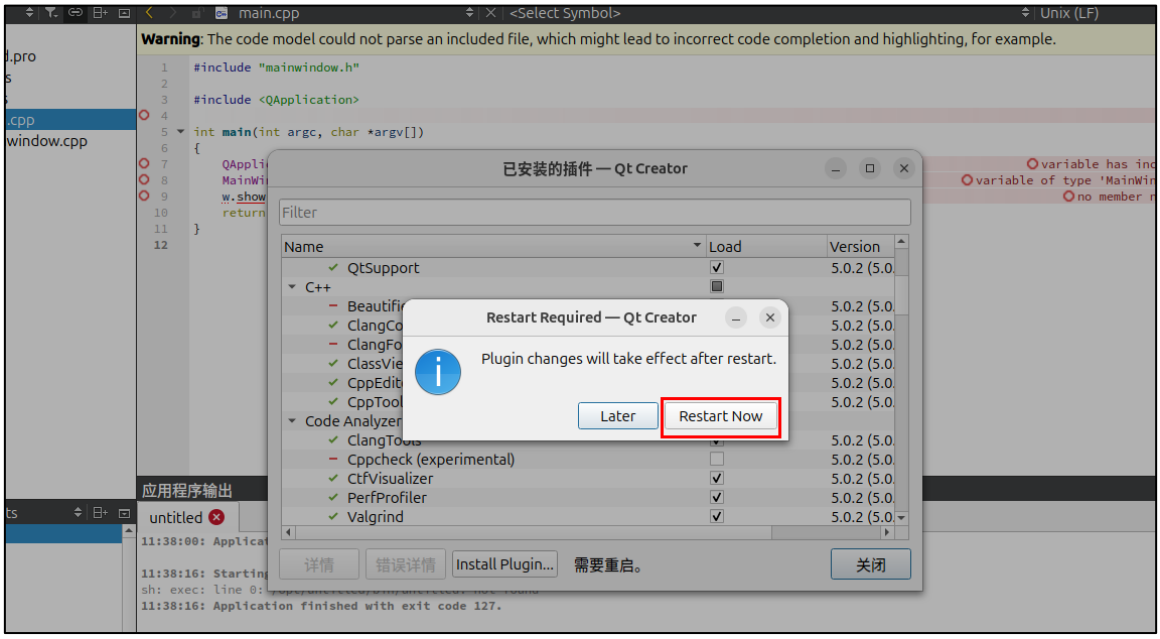


In my experience, it's because ClangCodeMode is checked in the plugin and click Help - "About plugins".

Uncheck the ClangCodeMode plugin.



Click Restart now.



No red after restart.