StarFive
赛昉科技

昉·惊鸿-7110 Linux Qt开发指南

版本：1.0

日期：2025/02/21

Doc ID：JH7110-DGCH-023

法律声明

阅读本文件前的重要法律告知。

版权注释

本文档中的说明均基于“视为正确”提供，可能包含部分错误。内容可能因产品开发而定期更新或修订。上海赛昉半导体科技有限公司 （以下简称“赛昉科技”）保留对本协议中的任何内容进行更改的权利，恕不另行通知。

赛昉科技明确否认任何形式的担保、解释和条件，无论是明示的还是默示的，包括但不限于适销性、特定用途适用性和非侵权的担保或条件。

赛昉科技无需承担因应用或使用任何产品或电路而产生的任何责任，并明确表示无需承担任何及所有连带责任，包括但不限于间接、偶然、特殊、惩戒性或由此造成的损害。

联系我们：

地址： 中国（上海）自由贸易试验区盛夏路61弄张润大厦2号电梯楼层5层（实际楼层4层）06室

网站： http://www.starfivetech.com

邮箱：

- sales@starfivetech.com（销售）
- support@starfivetech.com（支持）

Contents

List of Tables

List of Figures

前言

关于本指南和技术支持信息

关于本手册

本手册主要为开发者提供赛昉科技新一代SoC平台——昉·惊鸿-7110 Qt模块的编程基础，生成可在昉·惊鸿-7110 DevKit上正常运行的RISC-V端Qt镜像。

受众

本手册主要服务于与Qt相关驱动程序的开发人员。如果您正在开发其他模块，请与您的销售或支持顾问联系，获取我们在昉·惊鸿-7110上的完整文档。

修订历史

Table 0-1 修订历史

版本	发布说明	修订
1.0	2025/02/21	首次发布。

注释和注意事项

本指南中可能会出现以下注释和注意事项：

- **Tip:**
  建议如何在某个主题或步骤中应用信息。
- **Note:**
  解释某个特例或阐释一个重要的点。
- **Important:**
  指出与某个主题或步骤有关的重要信息。
- **CAUTION:**
  表明某个操作或步骤可能会导致数据丢失、安全问题或性能问题。
- **Warning:**
  表明某个操作或步骤可能导致物理伤害或硬件损坏。

1. 介绍

Qt是一种跨平台的C++图形应用程序开发框架，它提供了丰富的工具和库，用于构建高性能、可扩展且美观的应用程序。Qt最初由挪威的Trolltech公司（现在是Qt公司的一部分）开发，并于1995年首次发布。目前Qt 支持的操作系统包括Windows、嵌入式Linux、MS/Windows - 95、98、NT4.0、ME、2000、XP、Vista 等，这意味着使用Qt 您只需一次性开发应用程序，无须重新编写源代码，便可跨不同桌面和嵌入式操作系统部署这些应用程序。

本手册主要为开发者提供赛昉科技新一代SoC平台——昉·惊鸿-7110 Qt模块的编程基础，生成可在昉·惊鸿-7110 DevKit上正常运行的RISC-V端Qt镜像。

昉·惊鸿-7110 DevKit SDK基于buildroot框架，目前支持Qt5.15.2版本，目前支持以下Qt模块：

Table 1-1 支持的Qt模块

QT5BASE	QT5WEBSOCKETS	QT5DECLARATIVE	QT5MULTIMEDIA
QT5SVG	QT5SERIALBUS	QT5SERIALPORT	QT5QUICKCONTROLS
QT5WEBKIT	QT5WAYLAND	QT5WEBCHANNEL	QT5VIRTUALKEYBOARD
QT5CHARTS	QT5XMLPATTERNS	QT5WEBKIT_EXAMPLES	

2. 编译和运行SDK

2.1. 获取代码和编译

按照以下步骤，获取代码：

1. 执行以下命令，下载赛昉科技昉·惊鸿-7110 DevKit的代码：

```
$ git clone git clone https://glab.starfivetech.com/jh7110-devkits/devkits.git
```

2. 执行以下命令，切换分支：

```
$ cd devkits
$ git check --track origin/devkits-jh7110-devel
```

> **Tip:**
> 建议切换到分支`devkits-jh7110-devel` for 5.15 kernel或`devkits-6.6.y-devel` for 6.6 kernel。

3. 执行以下命令，下载其他模块：

```
$ git submodule update --init –recursive
```

4. 执行以下命令，编译代码：

```
$ export HWBOARD_CONFIG=qt
$ make -j$(nproc)
$ make buildroot_rootfs -j$(nproc)
$ make img
```

结果：

编译完成后，在`work/`目录下可找到以下两个固件：

- `sdcard.img`：TF卡烧录镜像
- `imag.fit`：U-Boot下TFTP启动镜像

> **Note:**
> 建议使用`sdcard.img`，可参考《昉·惊鸿-7110 DevKit快速参考手册》查看详细烧录步骤。

2.2. 运行Qt应用程序

按照以下步骤，运行Qt应用程序：

1. 按照SDK `README.md`里*Running on JH7110 Devkits Board via Network*和*APPENDIX I: Generate Booting SD Card*的说明将`imag.fit`或`sdcard.img`在昉·惊鸿-7110 DevKit上运行起来后，进入串口终端或ssh终端：
   - 账号：root
   - 密码：starfive
2. 执行以下命令，运行weston桌面：

```
$ ssh root@ip
$ cd /root/ && ./run_weston.sh &
$ export XDG_RUNTIME_DIR=/root
$ export WAYLAND_DISPLAY=wayland-1
$ export QT_QPA_PLATFORM=wayland-egl
```

2.3. 运行Qt Demo程序

以下提供了几个运行Qt的demo：

- 模拟时钟和布局相关的demo：

```
$ /usr/lib/qt/examples/gui/analogclock/analogclock
$ /usr/lib/qt/examples/widgets/layouts/basiclayouts/basiclayouts
```

- 运行qv4l2的demo：

> **Tip:**
> 需要提前在昉·惊鸿-7110 DevKit上接入传感器或USB摄像头，详细操作请查看《昉·惊鸿-7110 Camera开发和移植手册》。

```
$ /usr/bin/qv4l2
```

- 运行qtwebkit的demo：

```
/usr/lib/qt/examples/webkitwidgets/browser/browser
```

3. 基于QT库开发UI程序

本章主要介绍了以下两个方面：

3.1. Qt Creator环境搭建

本节包含以下两个方面：

3.1.1. 所需环境

在搭建环境前，请您做好以下准备：

- Ubuntu版本：18.04及以上版本
- 硬件平台：昉·惊鸿-7110 DevKit
- 软件：`qt-opensource-linux-x64-5.12.9.run`

3.1.2. 安装Qt Creator

请按照以下步骤，安装Qt Creator：

1. 点击进入Qt官网，下载Qt安装文件，此处我们使用的Qt版本是5.12.9，如下图所示：

   Figure 3-1 下载Qt

Name	Last modified	Size	Metadata
Parent Directory		-	
submodules/	16-Jun-2020 07:04	-	
single/	16-Jun-2020 07:04	-	
qt-opensource-windows-x86-5.12.9.exe	16-Jun-2020 18:07	3.7G	Details
qt-opensource-mac-x64-5.12.9.dmg	16-Jun-2020 14:49	2.7G	Details
qt-opensource-linux-x64-5.12.9.run	16-Jun-2020 14:48	1.3G	Details
md5sums.txt	16-Jun-2020 18:14	207	Details

2. 下载完成后，将`qt-opensource-linux-x64-5.12.9.run`拷贝至虚拟机开发环境的任意目录下，并执行以下命令：

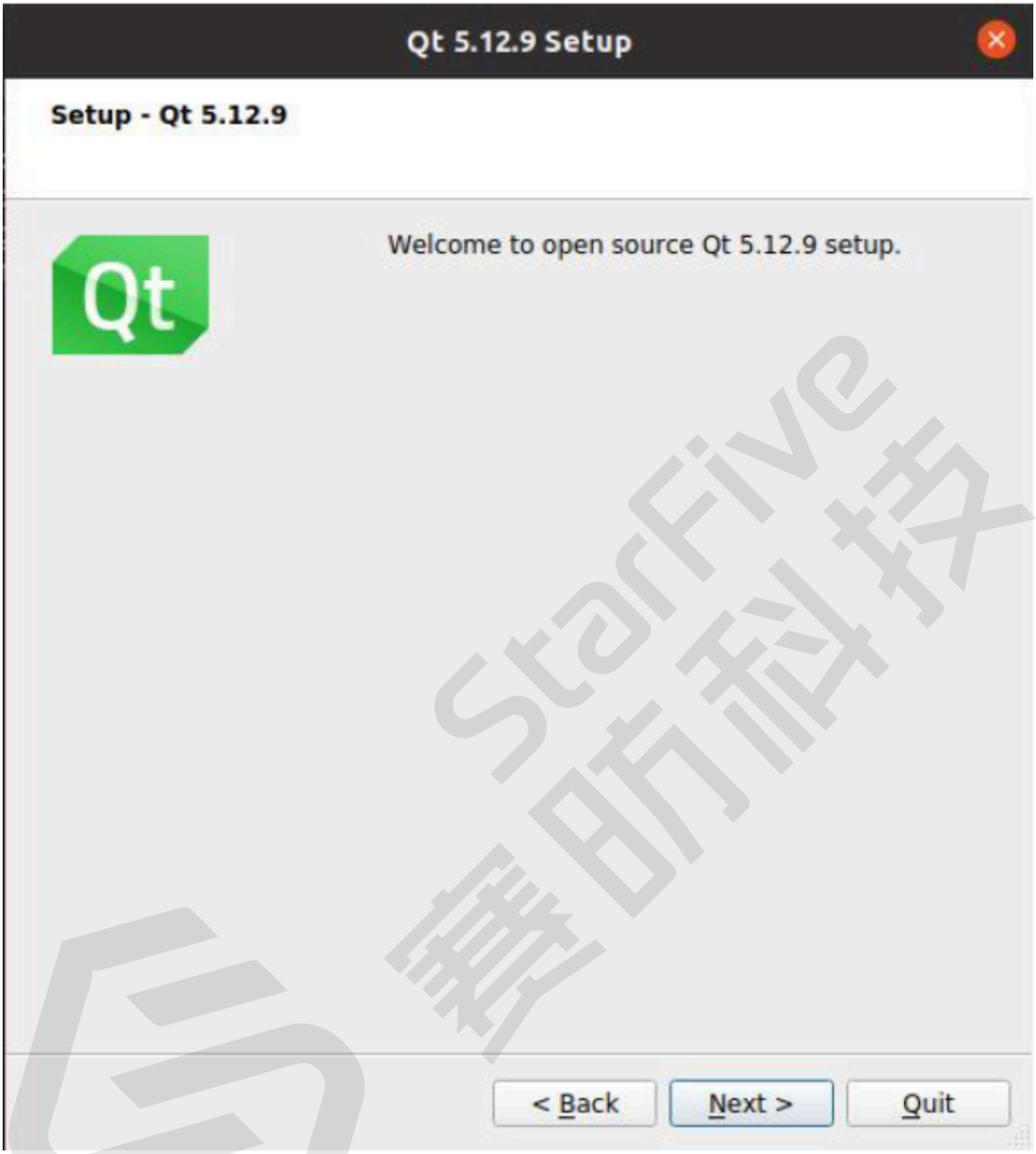
```
$ chmod 777 qt-opensource-linux-x64-5.12.9.run
$ ./qt-opensource-linux-x64-5.12.9.run
```

3. 执行完成后，系统会弹出以下界面：

Figure 3-2 安装界面

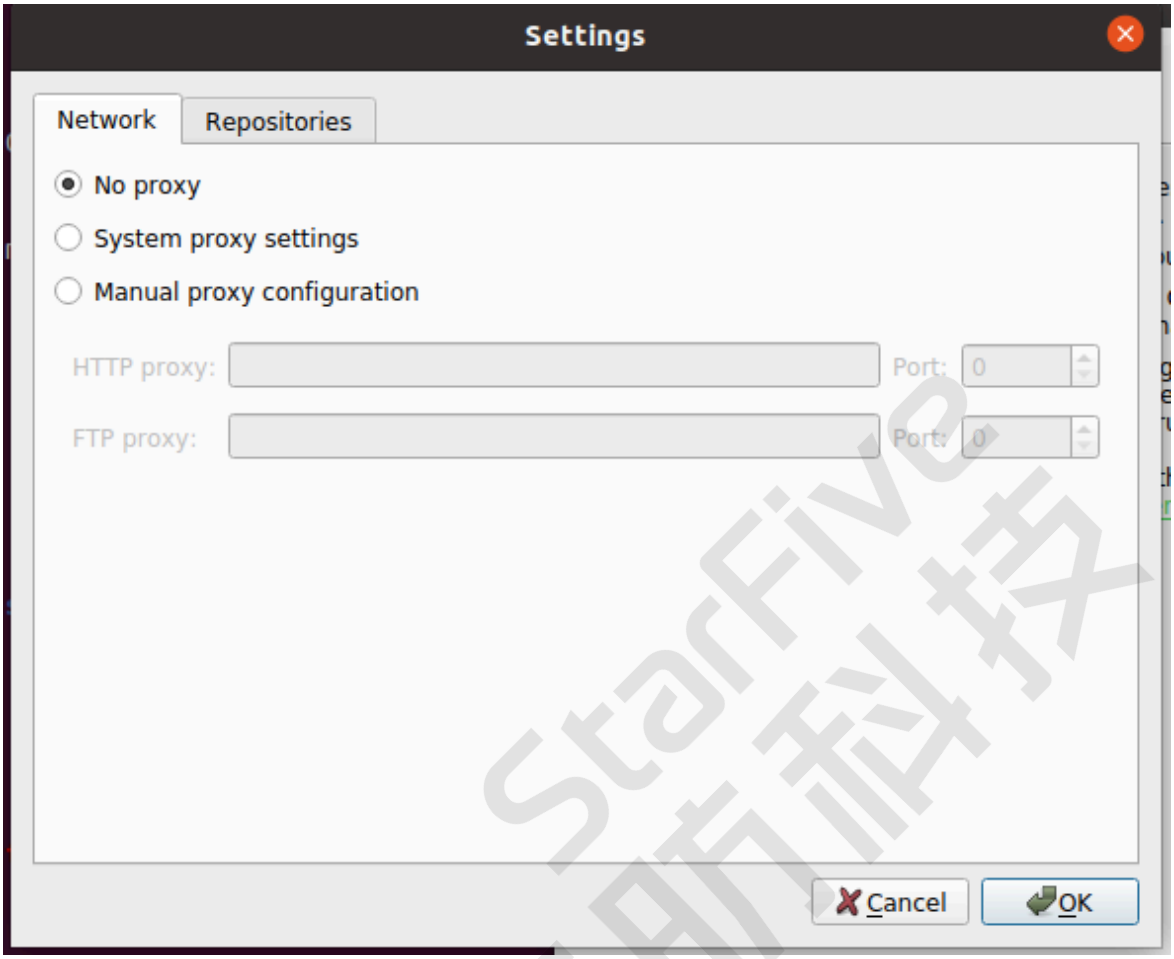
4. 在弹窗中一直点击**Next**进行安装:

Figure 3-3 点击Next



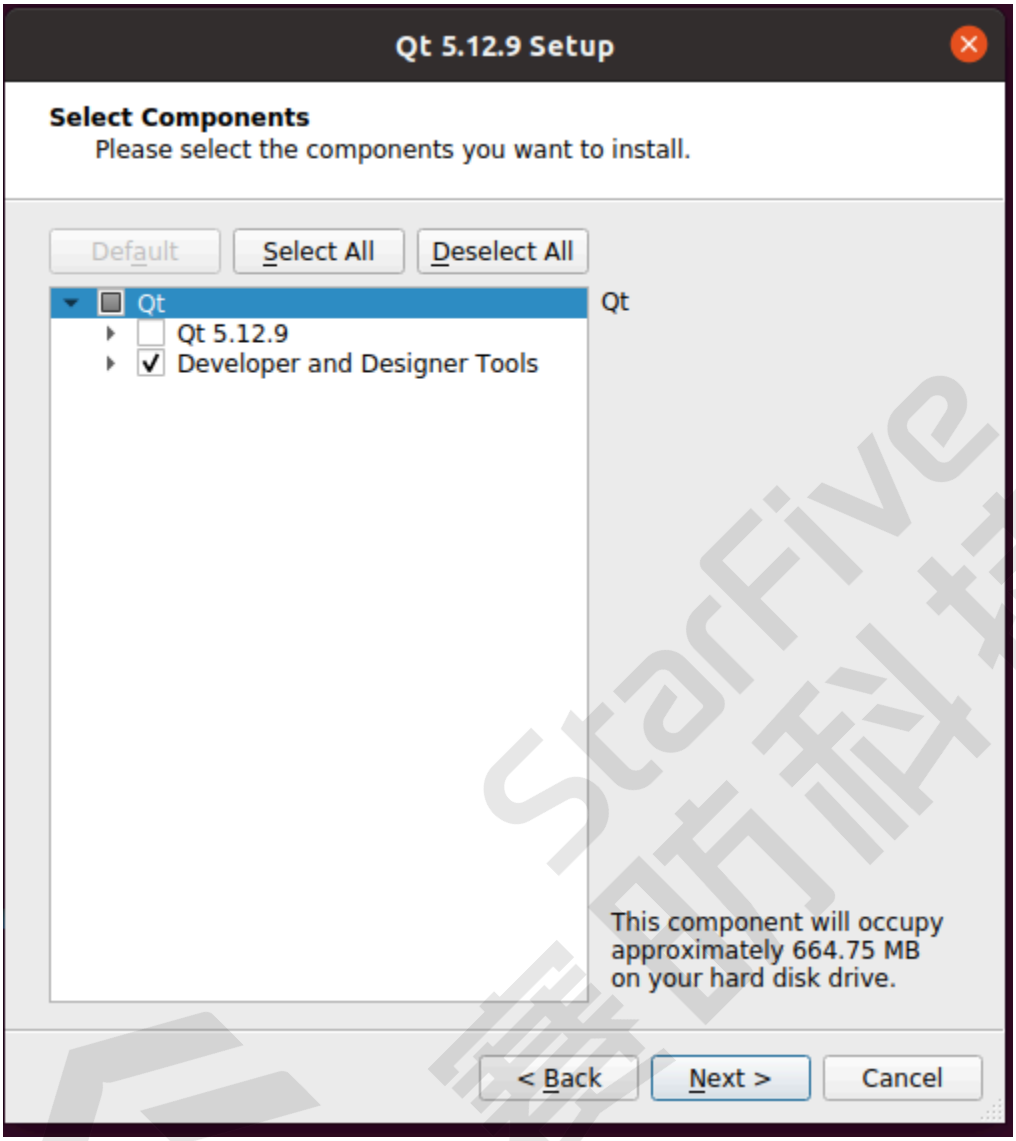
5. 在如下**Settings**界面中，点击**Browse...**选择Qt creator的安装路径，选择完成后，点击**Next**进入下一步：

Figure 3-4 Setting



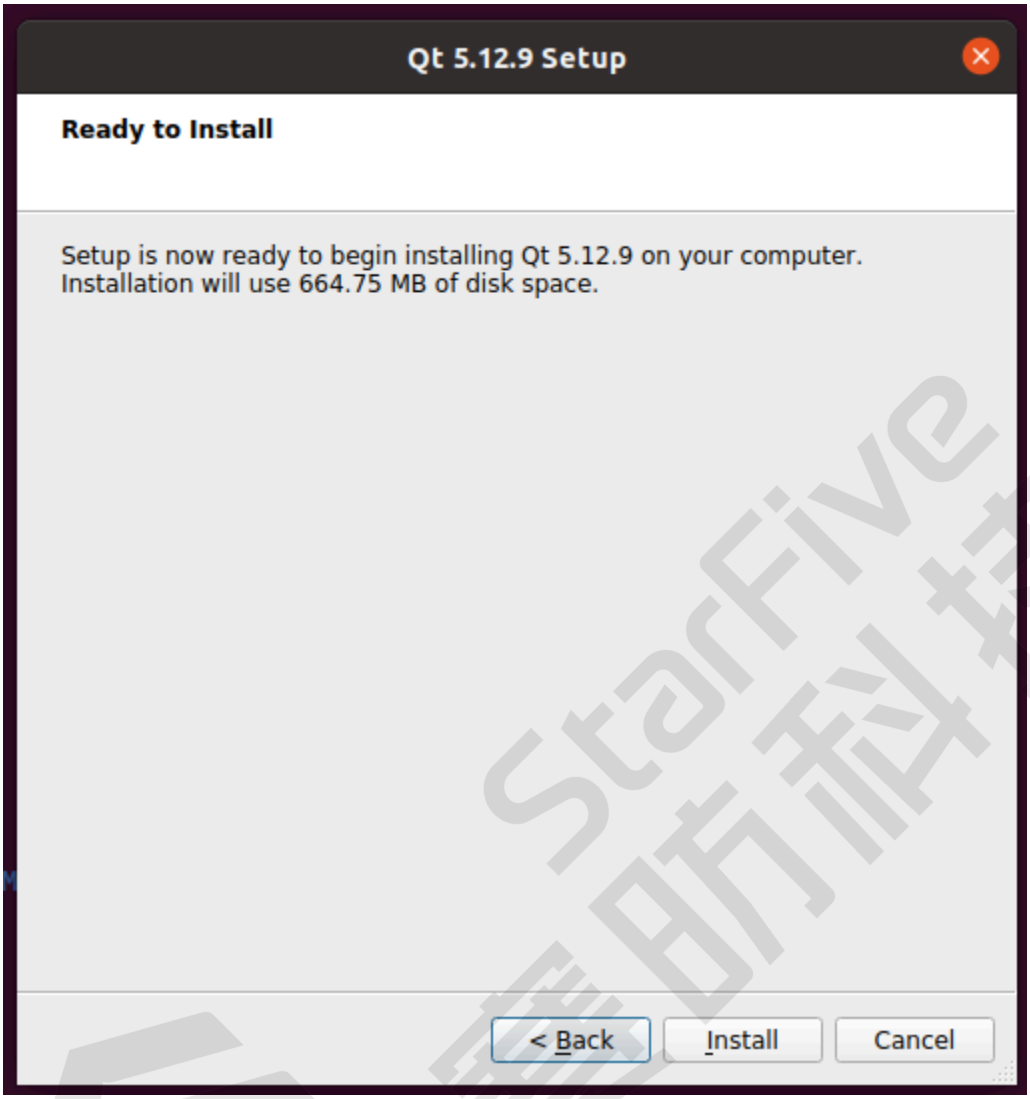
6. 在如下界面中，勾选**Developer and Designer Tools**组件，然后点击**Next**进入下一步：

Figure 3-5 选择组件



7. 点击**Install**开始安装:

Figure 3-6 开始安装



8. 安装完成后，显示如下界面，取消选项**Launch Qt Creator**的勾选，点击**Finish**，成功安装Qt Creator。

Figure 3-7 完成安装

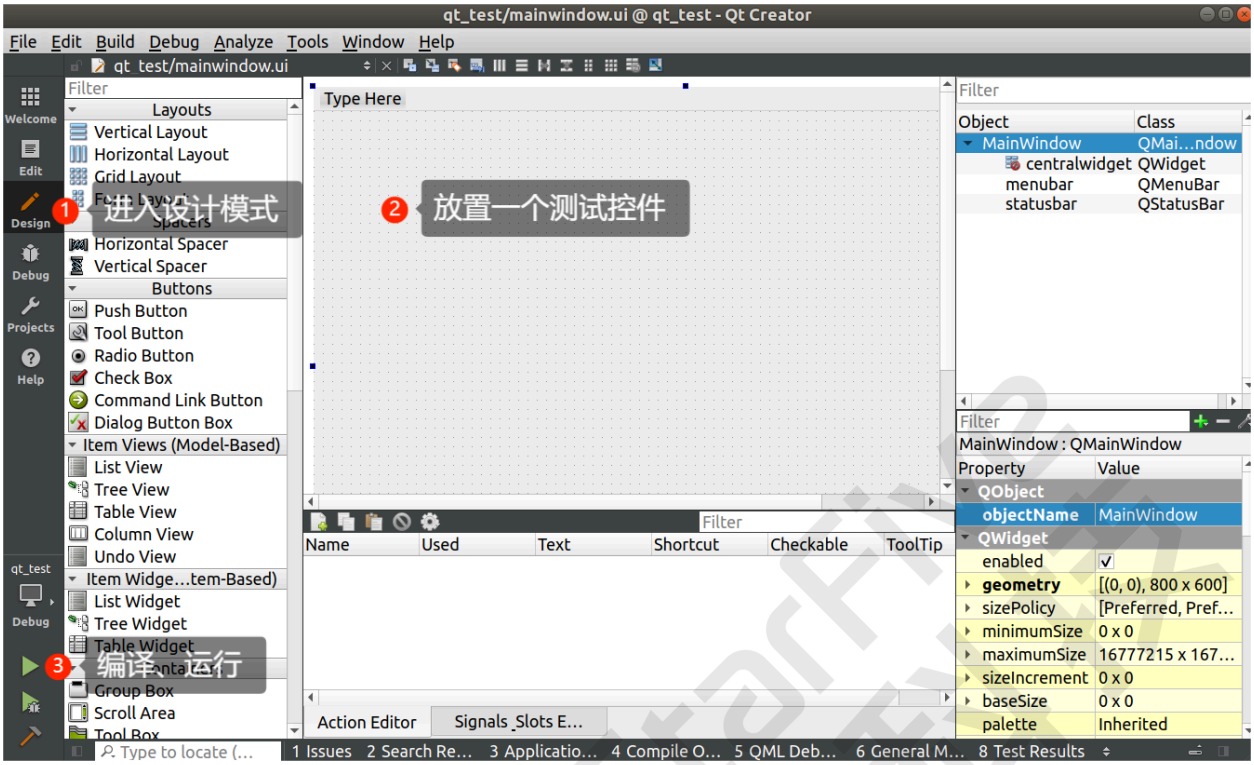
3.2. Qt Creator开发UI代码工程

Qt界面库的一大优点就是跨平台，您可以使用Qt Creator设计出好看的UI，然后移植进目标开发板上，以下内容介绍怎么使用Qt Creator来设计布局：

Qt Creator的布局支持使用`QtWidget`文件名为`*.ui`和`QtDeclarative`模块使用`*.qml`，这里介绍QtWidget方式。

直接点击如上的`mainwindow.ui`可以打开UI Design界面：

Figure 3-8 UI Design界面



在这个界面上，您可以拖拽左边的控件来添加您窗体的控件，具体操作方法可以详见Qt Creatot官网 上的丰富教程。

3.3. Qt UI代码工程编译

本节主要包含以下三个方面：

- 基于Qt Creator工具编译 *(on page 17)*
- 基于Buildroot环境编译 *(on page 25)*
- 基于昉·惊鸿-7110 DevKit板上编译 *(on page 26)*

3.3.1. 基于Qt Creator工具编译

Qt Creator 是一个用于Qt 开发的轻量级跨平台集成开发环境。与Qt 语言相辅相成，Qt Creator 能够跨平台运行，目前支持的系统包括Linux（32 位及64 位），Mac OS X 和Windows，Qt Creator 的设计使得开发人员能够利用Qt 这个应用程序框架更加快速和轻易的开发务。

3.3.1.1. Qt Creator交叉编译器配置

本节主要介绍了以下几个内容：

- 配置交叉编译器 *(on page 17)*
- 配置Debuggers *(on page 20)*
- 配置Qt Version *(on page 21)*
- 配置Kits *(on page 22)*

3.3.1.1.1. 配置交叉编译器

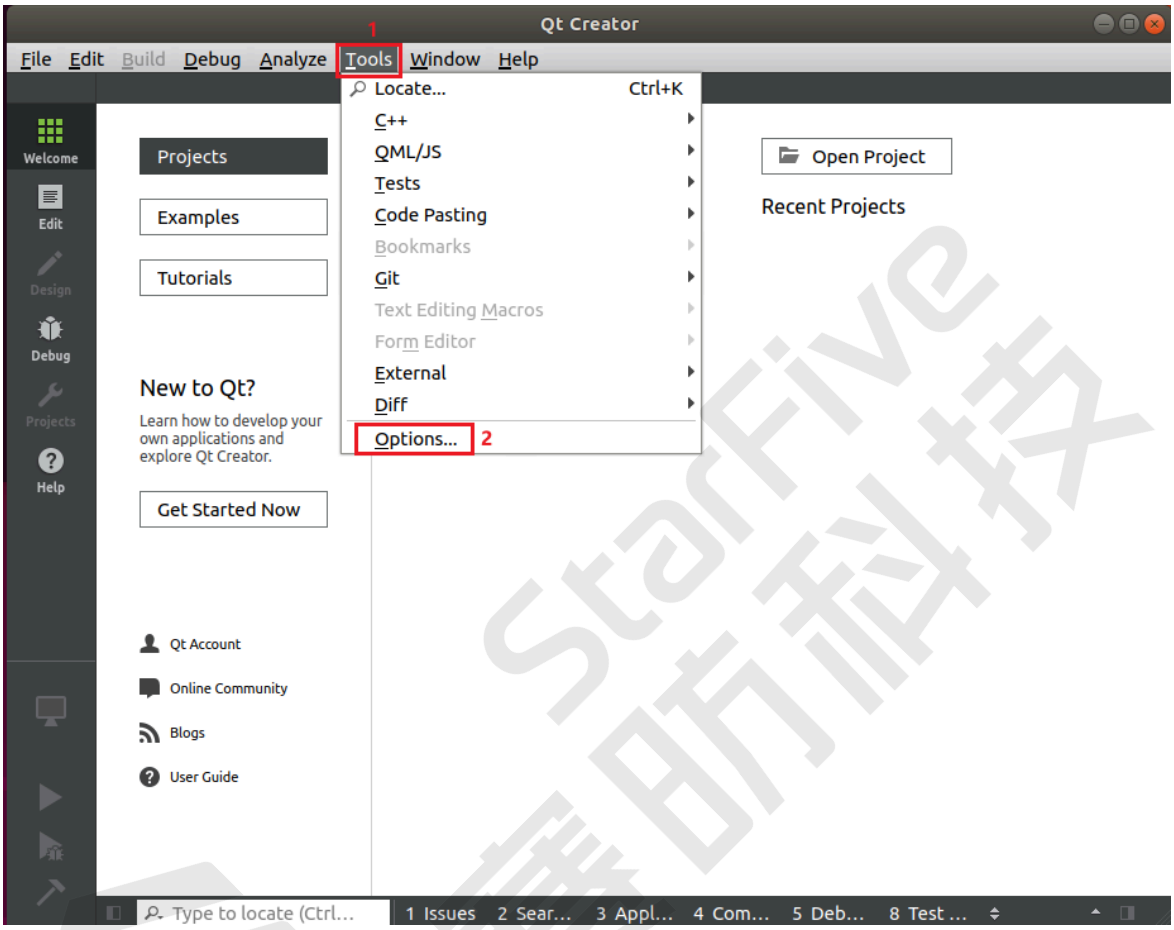
按照以下步骤，配置交叉编译器：

1. 执行以下命令，启动Qt Creater，会出现Qt开发界面：

```
sudo ./qtcreator
```

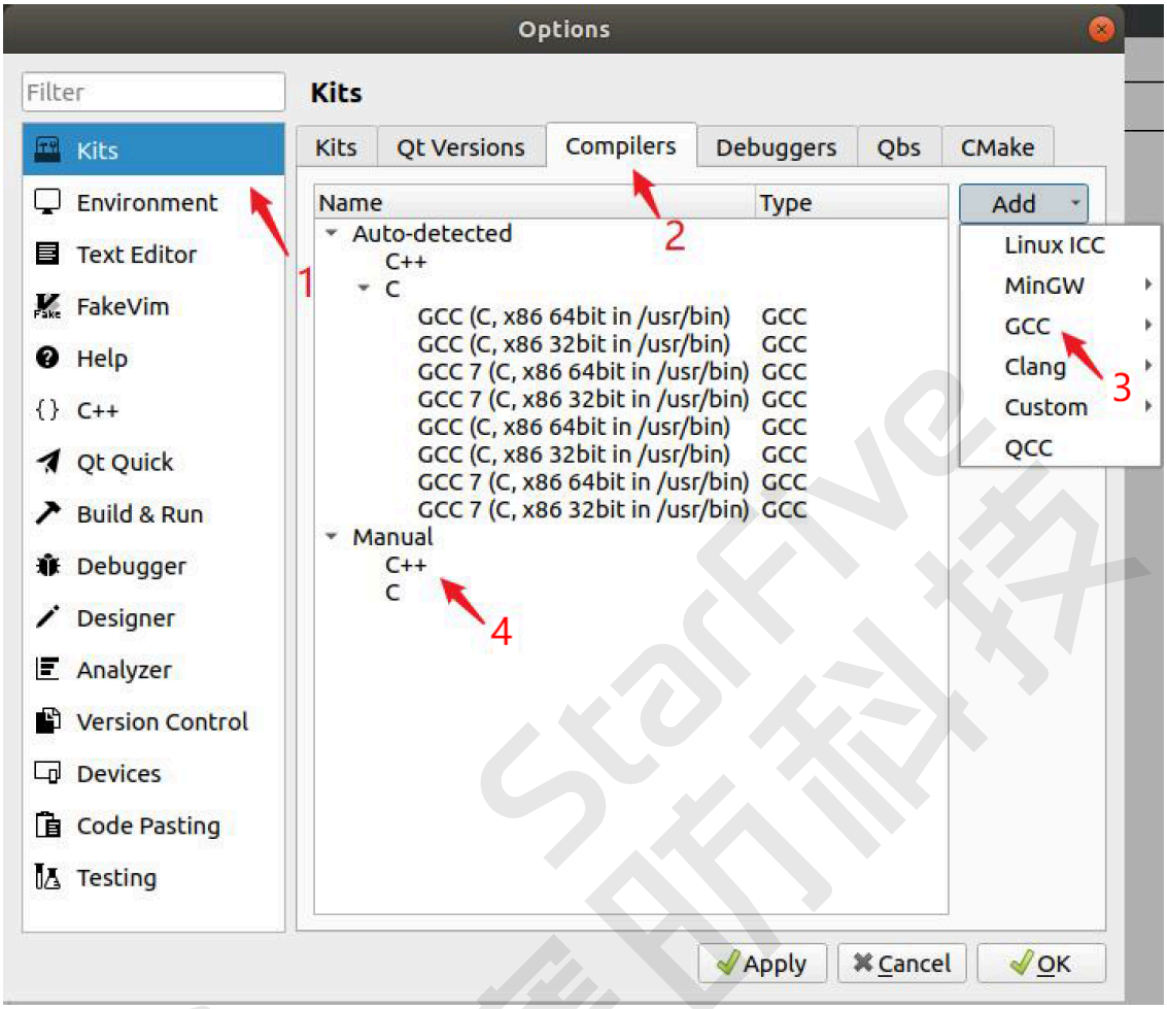
2. 在Qt开发界面中，点击**Tools > Options**选项，如下图所示：

Figure 3-9 选择Options



3. 进入到**Options**界面，按照**Kits > Compilers > GCC > C++**顺序选择C++，如图所示：

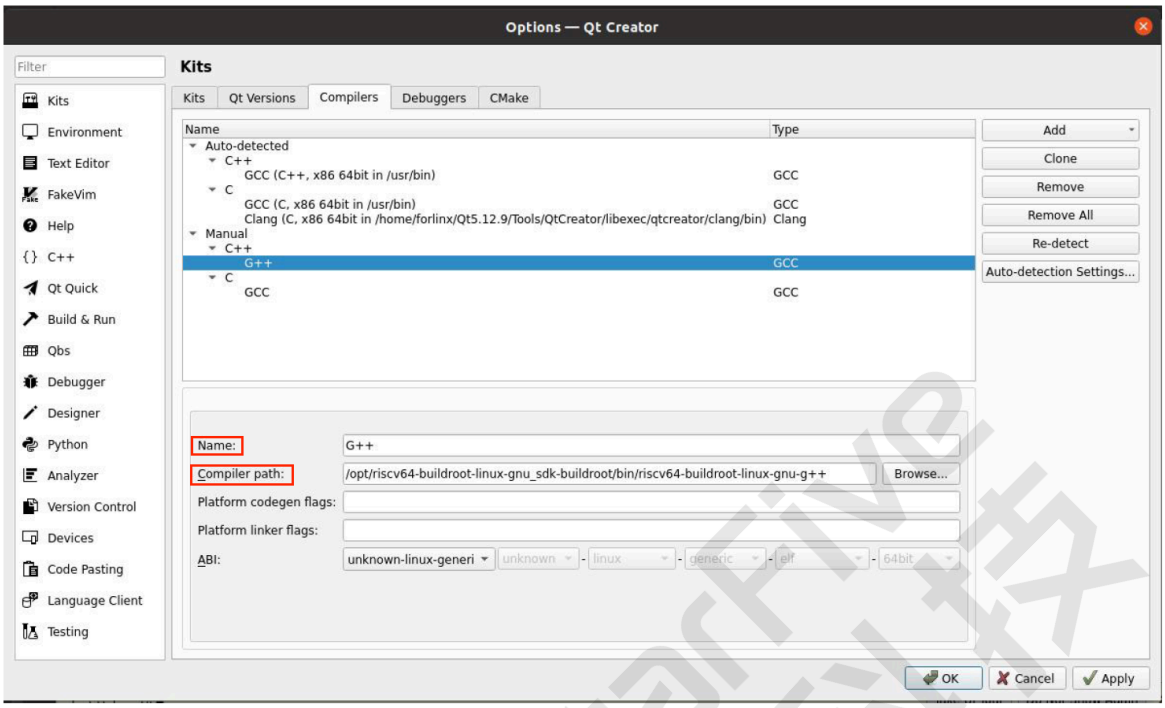
Figure 3-10 选择C++



4. 下图为弹出的窗口，添加G++编译器。在**Compiler path**填入`/opt/riscv64-buildroot-linux-gnu_sdk-buildroot/bin/riscv64-buildroot-linux-gnu-g++`，并将**Name**修改为`G++`，如下图所示：

> **Tip:**
> 为避免混淆，您也可以将**Name**修改为其他名称，如`jh7110-g++`。

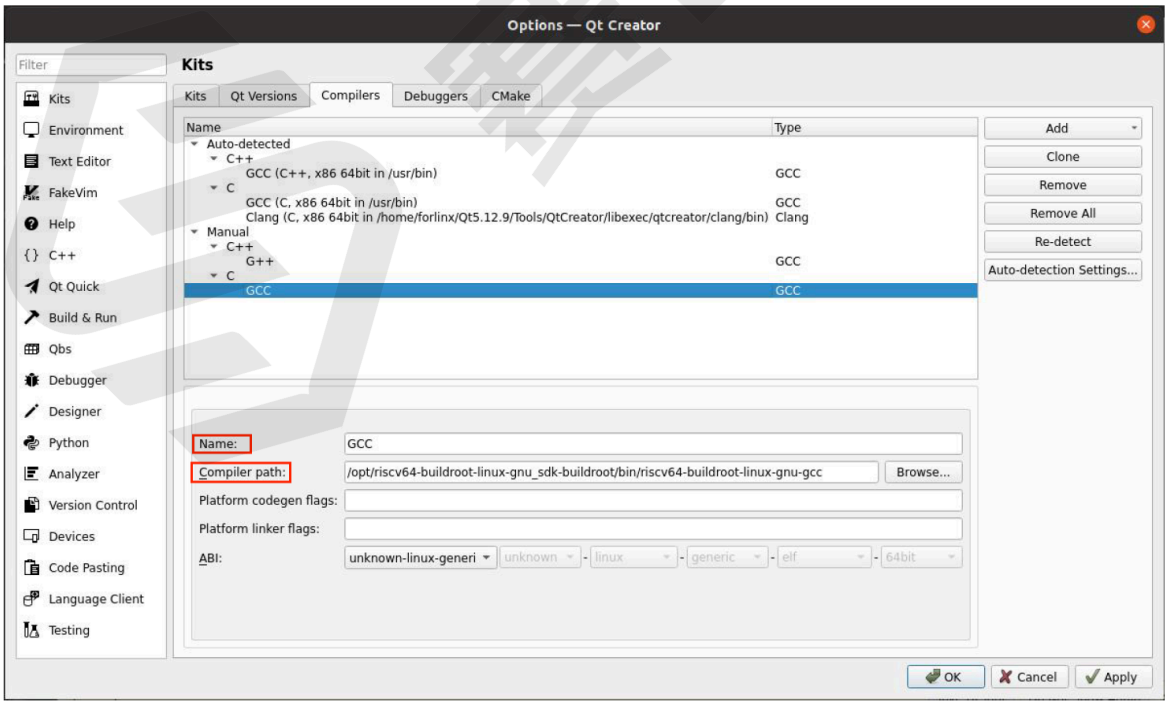
Figure 3-11 添加G++编译器



5. 按照第3步 *(on page 18)*描述的同样的方法，添加GCC编译器。点击右侧**Add > GCC > C**，在**Compiler path**填入`/opt/riscv64-buildroot-linux-gnu_sdk-buildroot/bin/riscv64-buildroot-linux-gnu-gcc`，并将**Name**修改为`GCC`，如下图所示：

Tip:
为避免混淆，您也可以将**Name**修改为其他名称，如`jh7110-GCC`。

Figure 3-12 添加GCC编译器

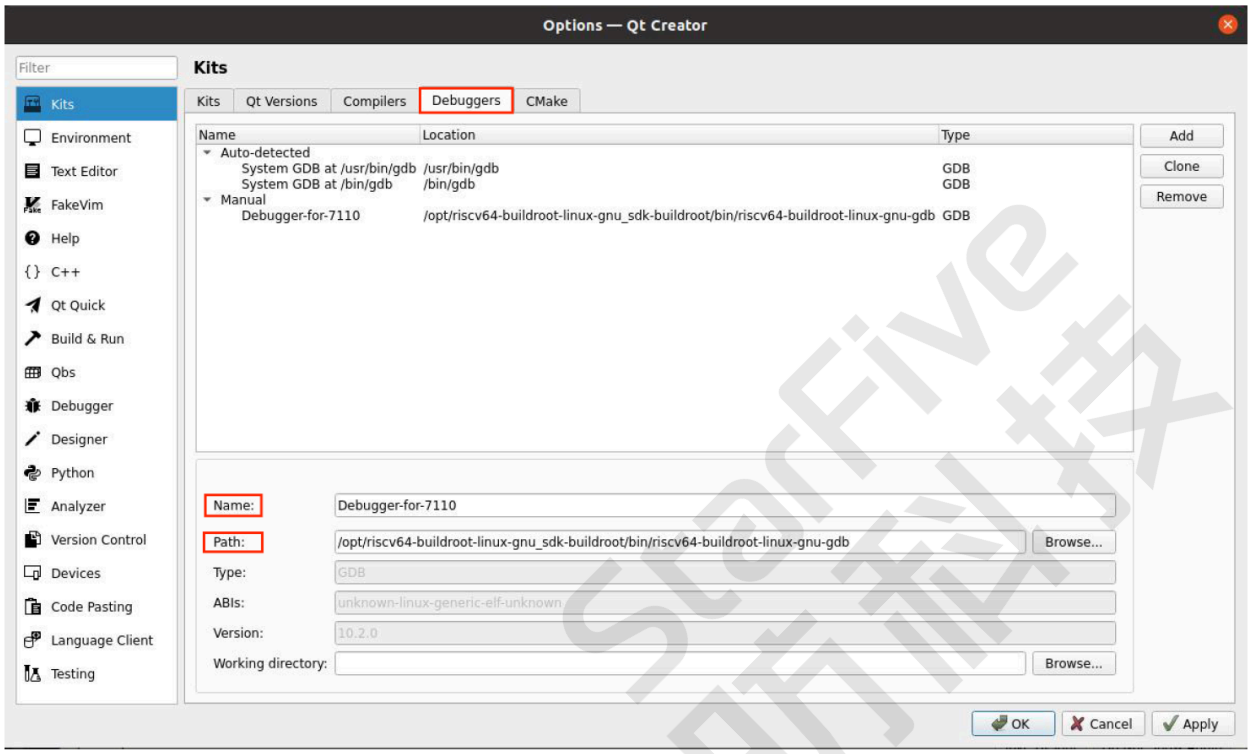


3.3.1.1.2. 配置Debuggers

配置Debuggers

在弹出的如下窗口，点击**Debuggers**选项卡，在Path中填入`/opt/riscv64-buildroot-linux-gnu_sdk-buildroot/bin/riscv64-buildroot-linux-gnu-gdb`，选中后点击Open，并将**Name**修改为`Debugger-for-7110`，如下图所示：

Figure 3-13 配置Debuggers

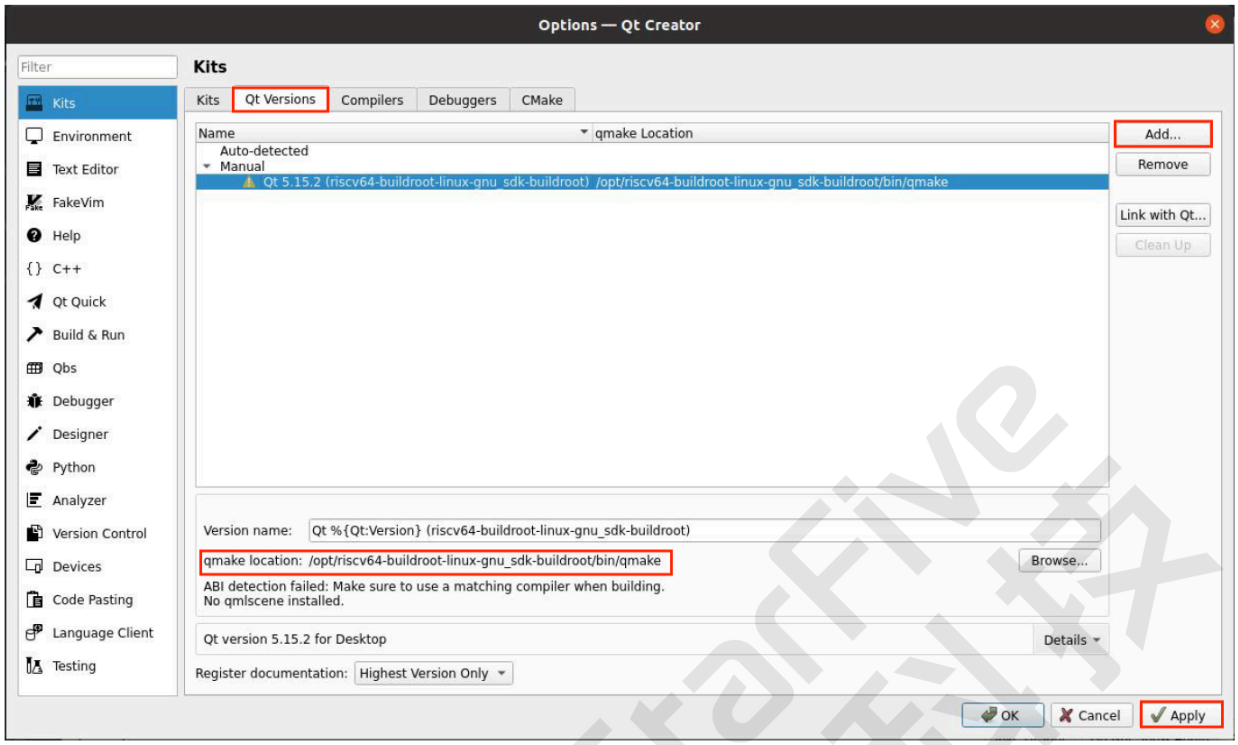


3.3.1.1.3. 配置Qt Version

配置Qt Version

点击**Qt Versions**选项卡，点击右侧的**Add**，在`/opt/riscv64-buildroot-linux-gnu_sdk-buildroot/bin/`的目录下找到`qmake`，选中后点击Open，添加后显示下图，点击**Apply**。

Figure 3-14 配置Qt Version

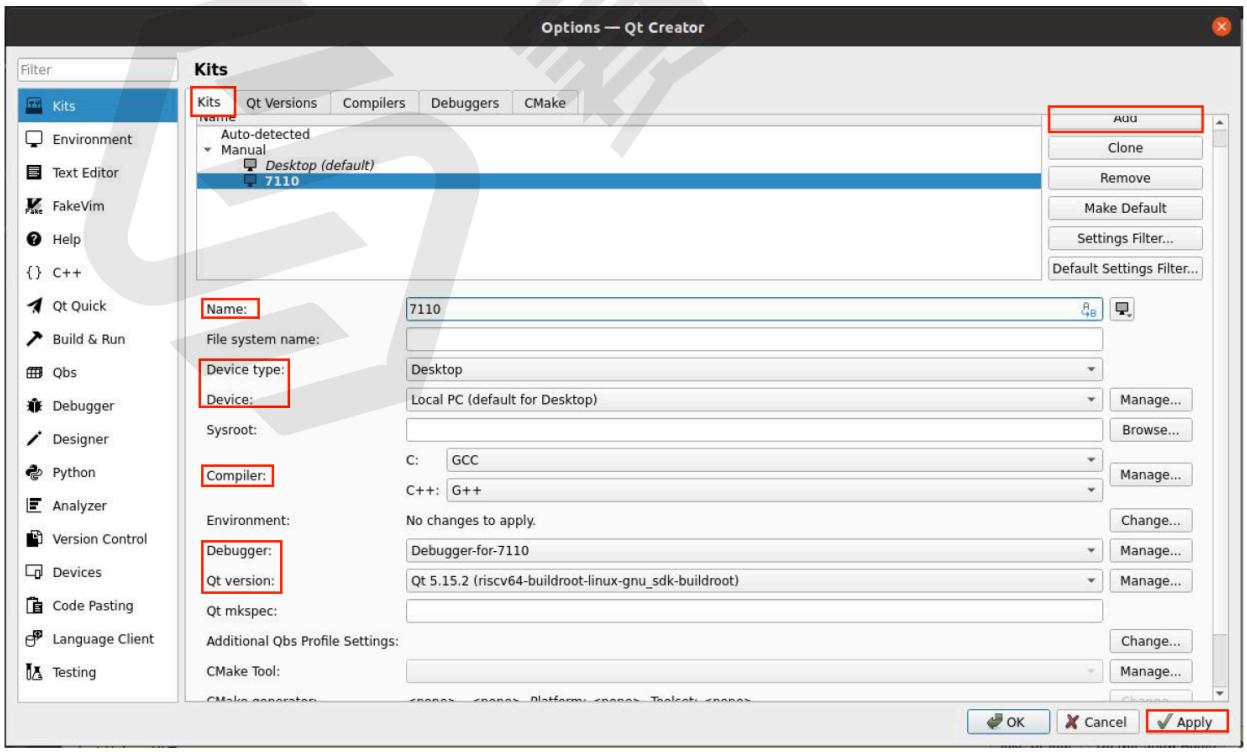


3.3.1.1.4. 配置Kits

配置Kits

点击**Kits**选项卡，点击右侧**Add**，添加一个新的Kits，按照下图的内容进行修改，点击**Apply**。

Figure 3-15 配置Kits



3.3.1.2. 新建项目

按照以下步骤，新建Qt Creator项目：

1. 执行以下命令，进入QT Creator，启动QT Creator：

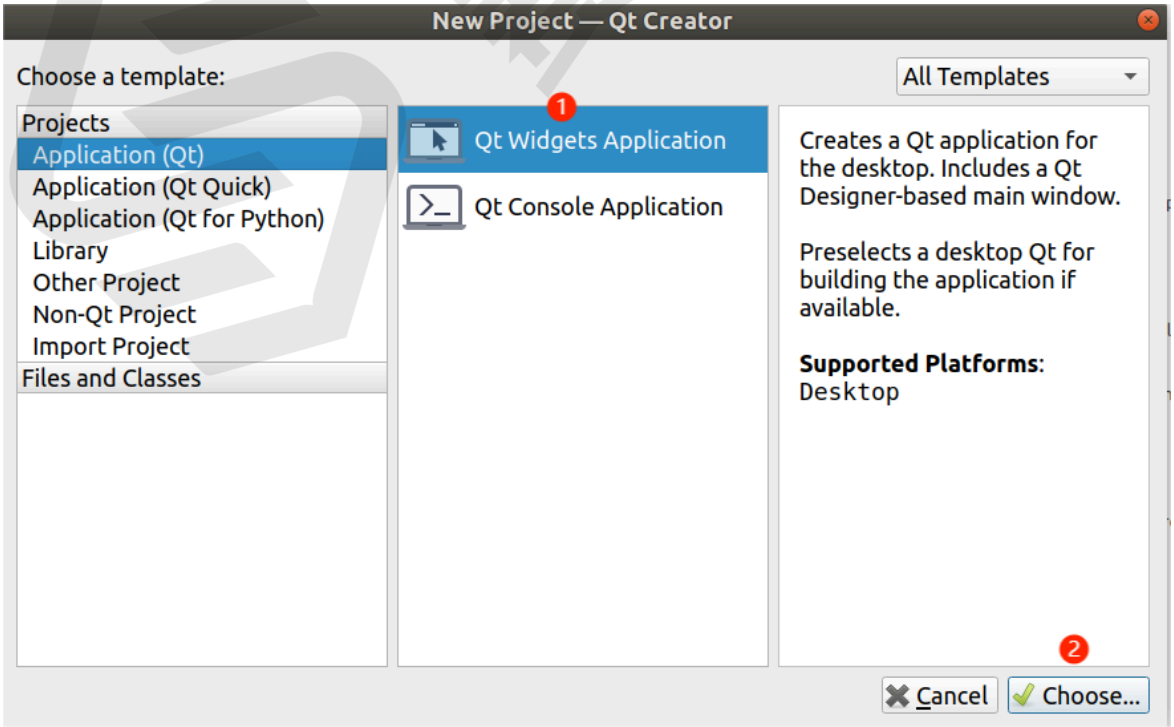
```
cd /home/riscvdev/Qt5.12.9/Tools/QtCreator/bin/
sudo ./qtcreator
```

示例输出：

Figure 3-16 Qt Creator首页

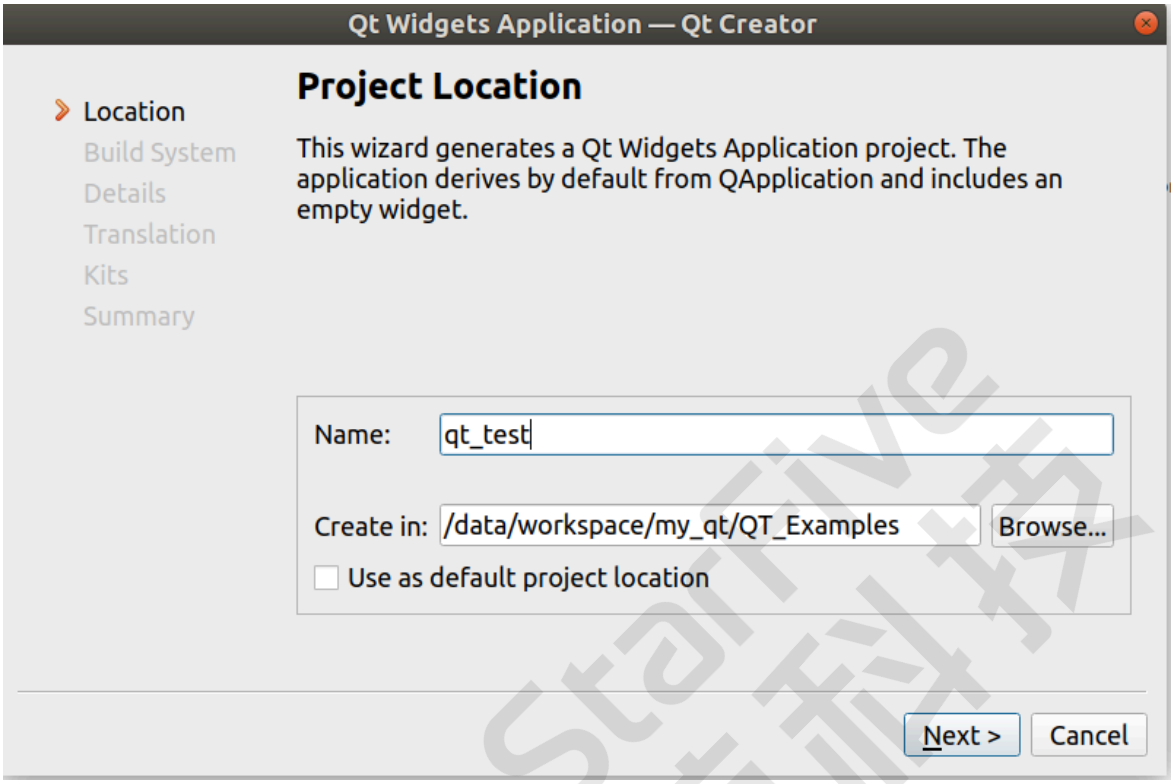
2. 点击**New**后，进入如下界面选择**Qt Widgets Application**，然后选择**Choose**，配置工程选项：

Figure 3-17 配置界面



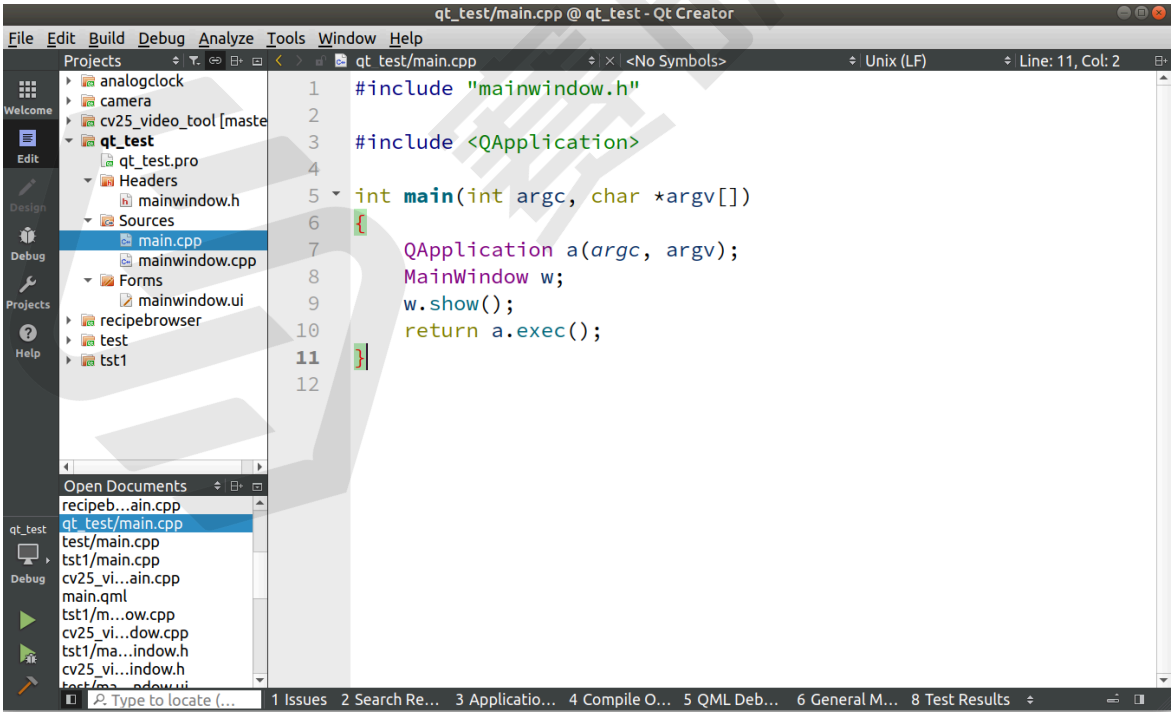
3. 输入您的项目名称，选择项目的保存路径，并点击**Next**：

Figure 3-18 项目名称和路径



4. 之后一直点击**Next**即可，项目创建成功后会出现如下界面：

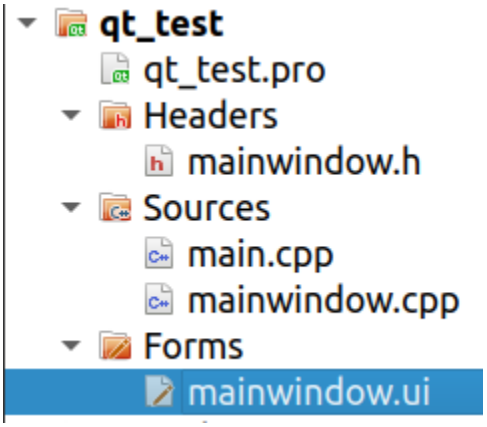
Figure 3-19 创建成功



3.3.1.3. 项目源码结构介绍

在默认新建的项目中，会包含以下这4类文件：

Figure 3-20 文件



以下是对这4类文件的介绍：

- `.pro`项目文件是qmake编译所需要的，具体编写方法见下文。
- `.cpp`文件和`.h`文件是c++的资源文件。
- `.ui`文件是设计师设计界面文件，使用xml来编写。

3.3.2. 基于Buildroot环境编译

本节介绍在Buildroot中添加自己编写的Qt，下面介绍添加进Buildroot的方法，可以参考已有的Qt package：`buildroot/package/starfive/qt_test`：

。

1. 在`buildroot/package/starfive`目录下，新建`package`文件夹，在文件夹中添加两个文件：
   - `Config.in`，该文件内容如下：

```
config BR2_PACKAGE_xxx
bool "xxx"
help
对于项目的描述
```

> **Tip:**
> *xxx*表示项目文件名。

   - `xxx.mk`，该文件内容如下：

```
QT_TEST_VERSION:=1.0.0
QT_TEST_SITE=$(TOPDIR)/package/starfive/qt_test/src
QT_TEST_SITE_METHOD=local
QT_TEST_DEPENDENCIES = qt5base

define QT_TEST_INSTALL_TARGET_CMDS
cd $(@D); $(TARGET_MAKE_ENV) $(MAKE) INSTALL_ROOT=$(TARGET_DIR) install
endef

$(eval $(qmake-package))
```

2. 准备Qt源码。在创建Buildroot `package`文件夹后，需要把前面在QT Creator里创建的源码 *(on page 24)*添加到SDK中去编译，并将整个源码复制到`buildroot/package/starfive/xxx/src/`目录下。

> **Note:**
> 将`*.pro`文件里的`install`目录改为其他目录，如`target.path = /root/bin`。

3. 编译。在 SDK 顶层目录，执行以下命令：

- For **image.fit**

```
$ make -C ./work/buildroot_initramfs/ O=./work/buildroot_initramfs qt_test-dirclean
$ make -C ./work/buildroot_initramfs/ O=./work/buildroot_initramfs qt_test-rebuild
```

- For **sdcard.img**

```
$ make -C ./work/buildroot_rootfs/ O=./work/buildroot_rootfs qt_test-dirclean
$ make -C ./work/buildroot_rootfs/ O=./work/buildroot_rootfs qt_test-rebuild
```

3.3.2.1. Buildroot Qt相关编译选项说明

Table 3-1 编译选项说明

编译选项	说明
BR2_PACKAGE_QT5BASE	是否支持QT，添加qtbase（必选）
BR2_PACKAGE_QT5CHARTS	是否支持数据可视化和图表的功能
BR2_PACKAGE_QT5DECLARATIVE	是否支持QML图形设计方式
BR2_PACKAGE_QT5MULTIMEDIA	是否添加多媒体类
BR2_PACKAGE_QT5QUICKCONTROLS	是否引入Qt Quick Controls服务组件
BR2_PACKAGE_QT5SERIALBUS	是否提供串行总线通信的功能，如CAN、Modbus
BR2_PACKAGE_QT5SERIALPORT	是否提供串口通信的功能
BR2_PACKAGE_QT5SVG	是否支持位图格式图片格式
BR2_PACKAGE_QT5VIRTUALKEYBOARD	是否支持虚拟键盘
BR2_PACKAGE_QT5WAYLAND	是否添加wayland显示服务组件
BR2_PACKAGE_QT5WEBCHANNEL	可用于在Web应用程序和Qt应用程序之间进行通信和交互
BR2_PACKAGE_QT5WEBKIT	可提供基于WebKit引擎的Qt模块，可以用于浏览web 页面
BR2_PACKAGE_QT5WEBKIT_EXAMPLES	可提供 Qtwebkit的一些example，如browser
BR2_PACKAGE_QT5WEBSOCKETS	可用于用于在应用程序中实现WebSocket协议的通信功能
BR2_PACKAGE_QT5XMLPATTERNS	基于XQuery和XPath语言，用于在Qt应用程序中进行XML数据处理和查询

3.3.3. 基于昉·惊鸿-7110 DevKit板上编译

3.3.3.1. Debian OS下编译

按照以下步骤，在Debian OS下编译Qt：

Note:
Debian OS下`root`目录切换命令，请使用`su root`。

1. 执行以下命令，安装Perl：

```
sudo apt install perl gcc g++ make        # 更新到最新的即可
```

2. 执行以下命令，安装python：

```
sudo apt install python2      # 更新到最新的即可
```

Tip:
编译过程中有可能会报错，如未知的成员变量，只需要在当前报错文件上引入头文件`#include <limits>` 即可；如果报python相关的错误，请执行`cp /usr/bin/python3 /usr/bin/python`。

3. 在命令行下先切换到工程目录下，编译Qt。
   - 在命令行下，执行以下命令，生成的`qmake`文件：

     ```
     qmake
     ```

   - 执行以下命令，生成相关二进制可执行文件：

     ```
     make
     ```

4. QT+TP触摸配置

目前SDK里默认已经支持触摸功能，只需要连接带触摸功能的屏幕，步骤同上面即可实现触摸功能，触摸功能准确无偏差。