WT-EVK6ULX Application Development Guide

Revise history:

Version	Date	Log
V0.9	2019/01/10	Create
V1.0	2019/01/16	Publish V1.0
V1.1	2019/07/05	1. Add header, footer, revision history 2. Update 2.1 Extraction code to: drh4

Content

1. ARM-Linux Application Development Environment Construction

1.1 ARM-Linux general developing methods

Developing applications on ARM-Linux system usually need to use another host for cross-compiling ,and then download the application to the target board through serial port or network. The general development process is shown in the following figure.

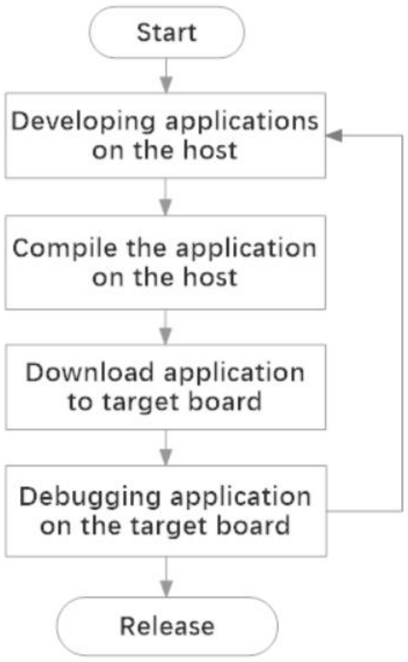

Start
Developing applications on the host
Compile the application on the host
Download application to target board
Debugging application on the target board
Release

Figure 1.1. General process of developing on ARM-Linx

The general model of embedded Linux development is shown in Figure 1.2. Usually, a PC host is required, in which various software required for cross-compilation is installed, and connected through a serial port, or an Ethernet to the target board. The program is edited and compiled on the host, and the obtained executable file is downloaded to the target board through the serial port or Ethernet for debugging.

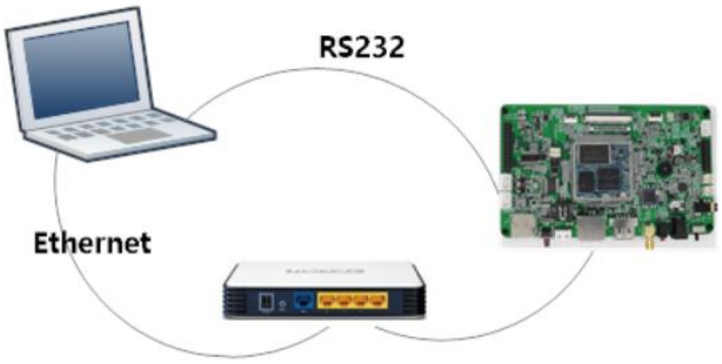


Figure1.2 Embedded Linux development model

1.2 Prepare working

Host	Option A (default): Windows+Vmware (ubuntu 12.04,64bit or upper)
	Option B: Ubuntu (12.04 ,64 bit or upper)
Host tools(windows)	Console tools: XShell or putty or Secure CRT
	File transfer: winscp
Host tools(Linux)	minicom ssh scp

2. Build a ubuntu development environment

2.1 Install cross-compile development toolchain

1) Download tool chain

Two versions of sdk available:

Download path1 : https://releases.linaro.org/components/toolchain/binaries/4.9-2017.01/arm-linux-gnueabihf/gcc-linaro-4.9.4-2017.01-i686_arm-linux-gnueabihf.tar.xz

Download path2(**for QT**) : https://pan.baidu.com/s/1yjjSyAoV6LVS008MJQS2mw ,

Extraction code： drh4

2) Install & configuration environment variables

➢ For standard SDK

Method 1：

a) Decompression:

```
sudo tar xjvf gcc-linaro-4.9.4-2017.01-i686_arm-linux-gnueabihf.tar.xz -C /opt/industio
```

b) Configuring environment variables, editing configuration scripts environment-setup_hf,

```
#vi environment-setup_hf
```

```
GCC_PATH=/opt/industio/gcc-linaro-4.9.4-2017.01-i686_arm-linux-gnueabihf
GCC_CC=arm-linux-gnueabihf

export ARCH=arm
export CROSS_COMPILE=$GCC_CC-
export PATH=$GCC_PATH/bin:$GCC_PATH/bin/$GCC_CC:$PATH
```

c) Effective environment variable

```
#source environment-setup_hf
```

Method 2： Add environment variables to the end of profile file.

```
#vi .profile    // Add the following to the end of the file
```

```
export PATH=$PATH:/opt/industio/gcc-linaro-4.9.4-2017.01-i686_arm-linux-gnueabihf/bin
export ARCH=arm
export CORESS_COMPILE= arm-linux-gnueabihf
```

Check the tool chain:

```
#which arm-linux-gnueabihf-gcc
```

➢ For SDK supports QT:

Confirm the standard SDK(V.4.9.4-2017.01) has installed.

Method 1:

d) Decompression:

sudo unzip xjvf imx6ull_qt4.8.7_sdk_hf.zip -C /opt/industio

e) Configuring environment variables, editing configuration scripts environment-setup_qt,

```
#vi environment-setup_qt
```

```
GCC_PATH=/opt/industio/gcc-linaro-4.9.4-2017.01-i686_arm-linux-gnueabihf
GCC_CC=arm-linux-gnueabihf


export ARCH=arm
export CROSS_COMPILE=$GCC_CC-
export
PATH=/opt/industio/imx6ull_qt4.8.7_sdk_hf/bin:$GCC_PATH/bin:$GCC_PATH/bin/$GCC_CC:
$PATH
```

f) Effective environment variable

```
#source environment-setup_qt
```

3. Hello world demo

Edit the simplest Hello world program on ubuntu which print the “Hello world “string to console. The sample program is as follows:

```
#include <stdio.h>

int main(void)
{
    int i;
    printf(“Hello world!\n”);
    return 0;
}
```

Enter the hello program directory and execute the command to compile the hello.c file.:

```
# arm-linux-gnueabihf-gcc hello.c -o hello
```

Use winscp to download the excutable hello file to the EVK and run the hello program.:

```
# ./hello
Hello World!
```

4. QT demo

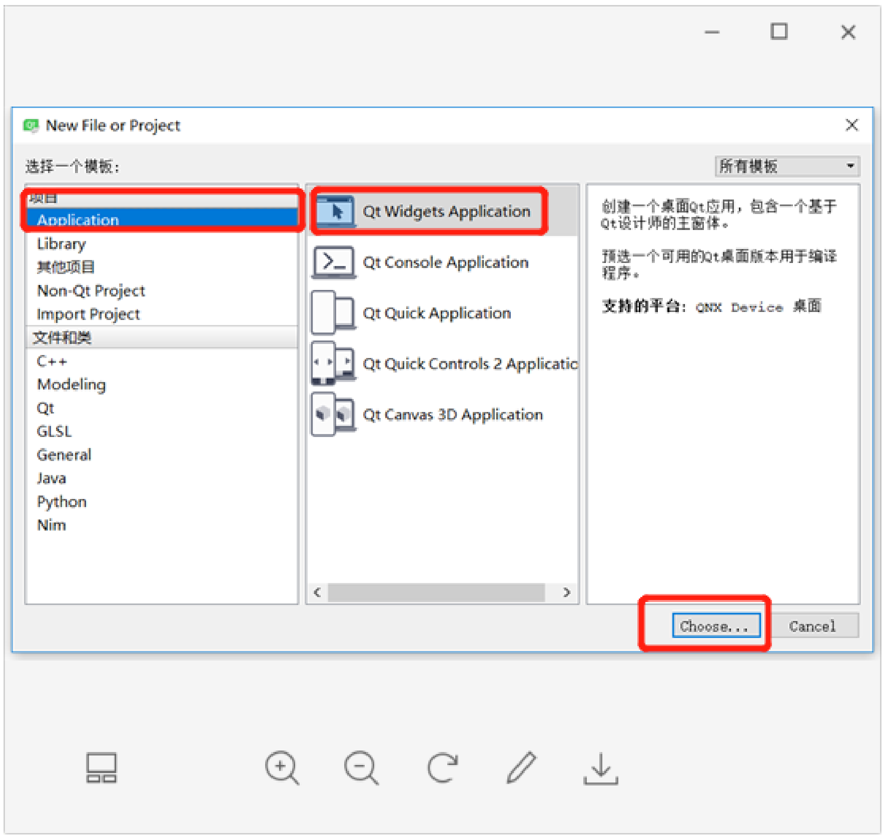
4.1 QT demo development

We can first complete the qt UI interface and application development on the Windows PC, and then package the project code (remove the build related files) into the ubuntu cross-compilation environment.

First install the QT desktop version (4.8.7 version or upper)on Windows. For the installation method, please refer to the official website operation guide and other online materials. Let's start by creating a hello world program.

1) Create a hello world project with QT.

➔ Create hello world project ， File->New file or project， like figure 4.1。



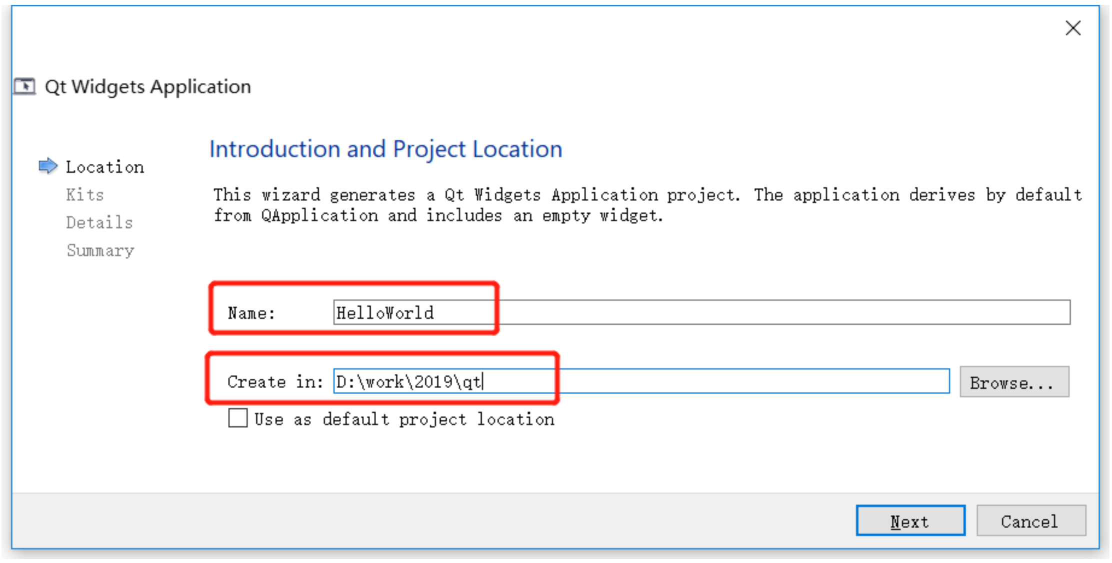
创建一个桌面Qt应用，包含一个基于Qt设计师的主窗体。

预选一个可用的Qt桌面版本用于编译程序。

支持的平台：QNX Device 桌面

Figure 4.1 create a new project

➔ Set the project name and project path:


Qt Widgets Application

Introduction and Project Location

This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.

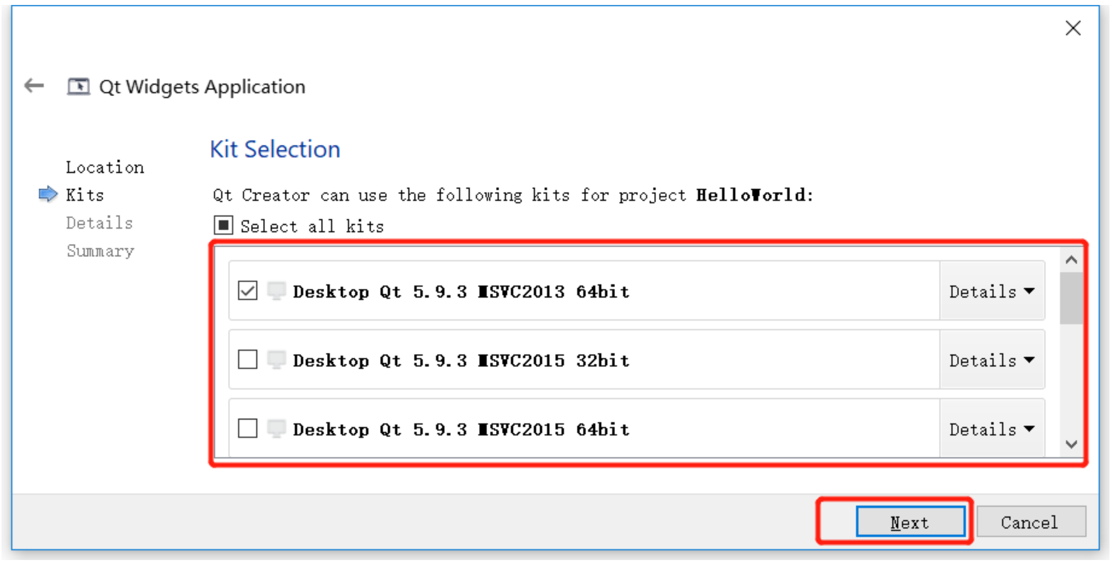
Name: HelloWorld

Create in: D:\work\2019\qt

☐ Use as default project location

Figure 4.2 Set the project name and project path

➔ Configure Kit Selection


Qt Widgets Application

Kit Selection

Qt Creator can use the following kits for project **HelloWorld**:

■ Select all kits

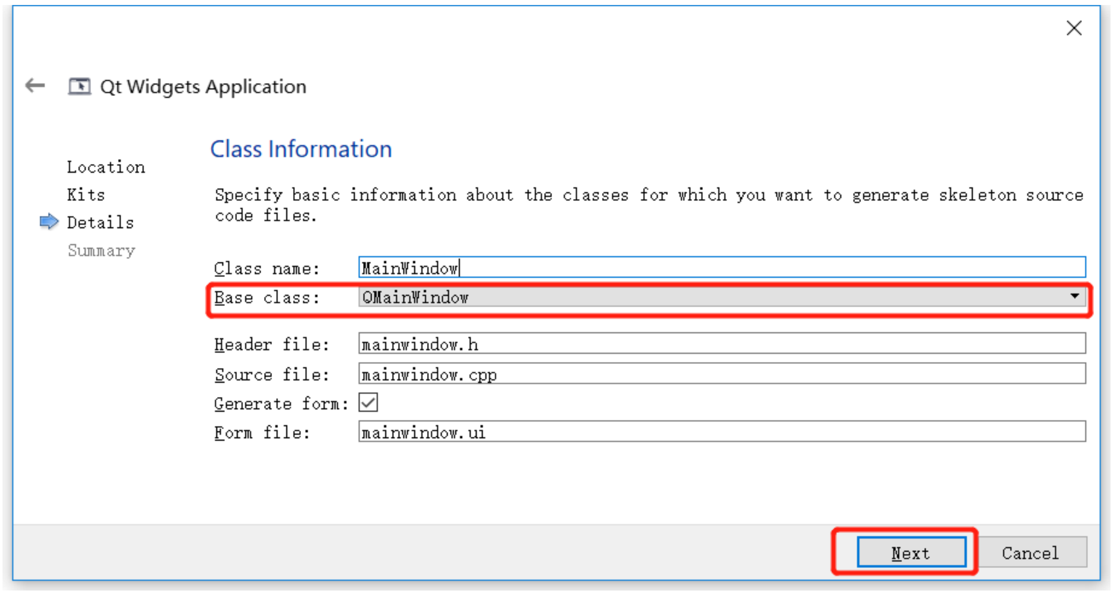
☑ **Desktop Qt 5.9.3 MSVC2013 64bit**

☐ **Desktop Qt 5.9.3 MSVC2015 32bit**

☐ **Desktop Qt 5.9.3 MSVC2015 64bit**

Figure 4.3 Kit Selection

➔ Configure Class Information


Qt Widgets Application

Location
Kits
Details
Summary

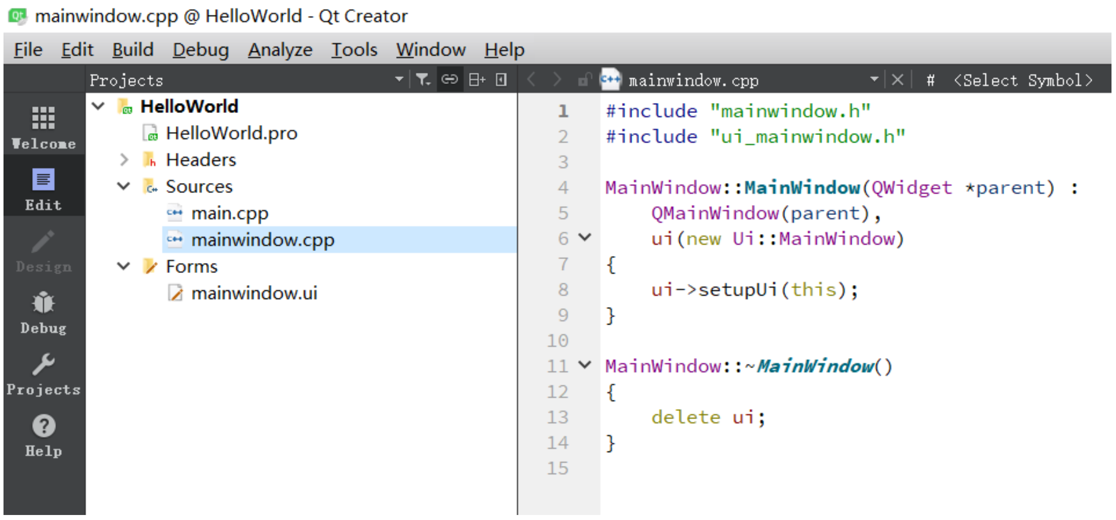
Class Information

Specify basic information about the classes for which you want to generate skeleton source code files.

Class name: MainWindow
Base class: QMainWindow
Header file: mainwindow.h
Source file: mainwindow.cpp
Generate form: ☑
Form file: mainwindow.ui

Figure 4.4 Class Information

➔ Create successfully.


mainwindow.cpp @ HelloWorld - Qt Creator

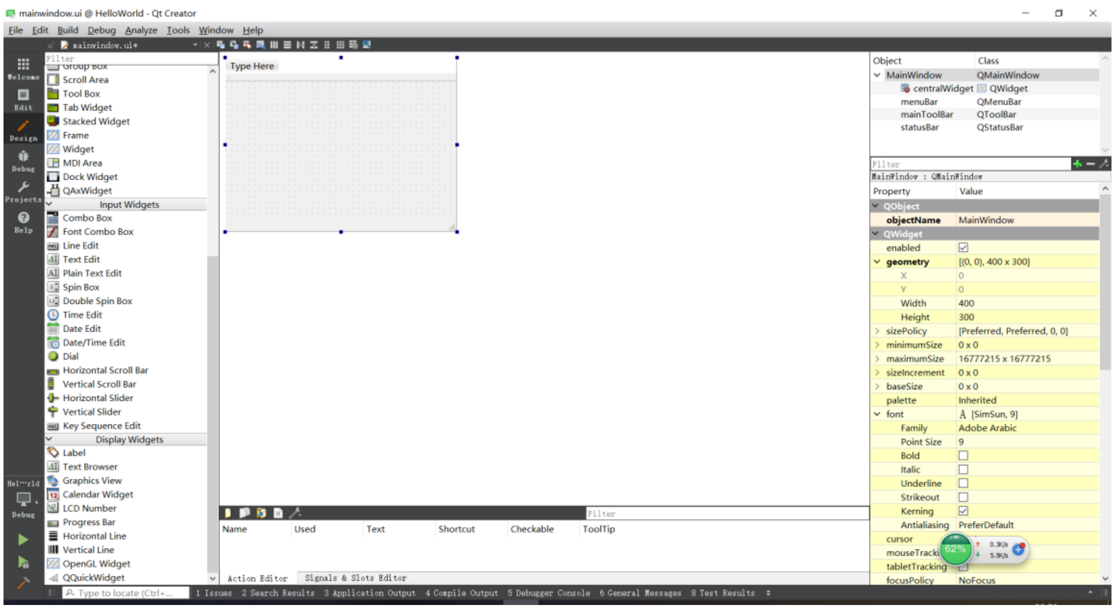
```
HelloWorld
    HelloWorld.pro
  Headers
  Sources
    main.cpp
    mainwindow.cpp
  Forms
    mainwindow.ui
```

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

```

Figure 4.5 Create successfully

➔ Create successfully.

Clicking on the mainwindow.ui in the sidebar of the project, start the visual editor.



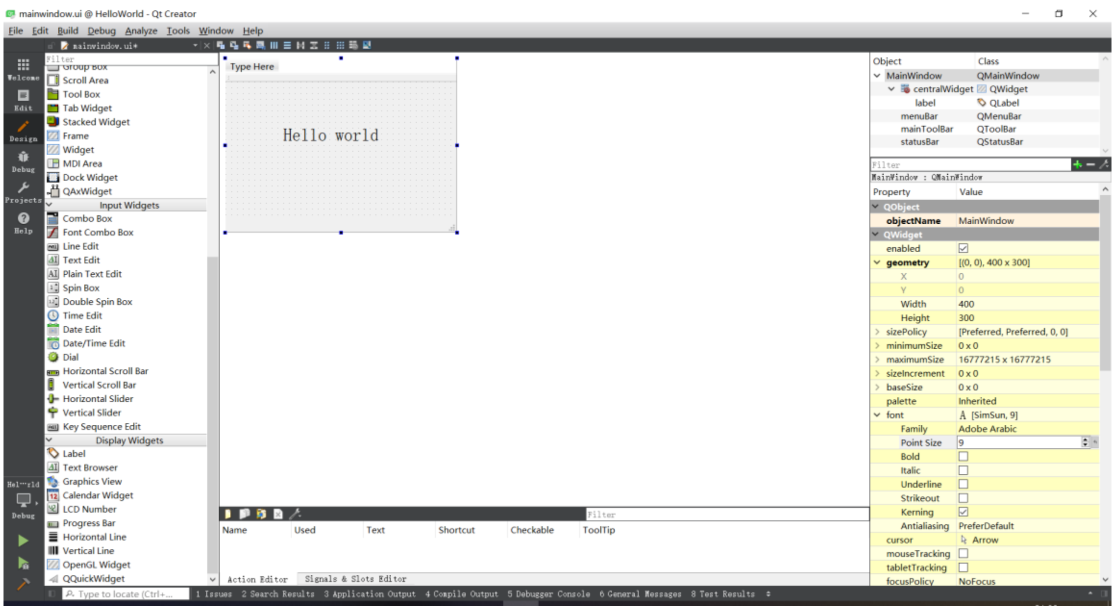
Object	Class
MainWindow	QMainWindow
centralWidget	QWidget
menuBar	QMenuBar
mainToolBar	QToolBar
statusBar	QStatusBar

MainWindow : QMainWindow

Property	Value
QObject	
objectName	MainWindow
QWidget	
enabled	✓
geometry	[(0, 0), 400 x 300]
X	0
Y	0
Width	400
Height	300
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited
font	A [SimSun, 9]
Family	Adobe Arabic
Point Size	9
Bold	☐
Italic	☐
Underline	☐
Strikeout	☐
Kerning	✓
Antialiasing	PreferDefault
cursor	
mouseTracking	
tabletTracking	☐
focusPolicy	NoFocus

Figure 4.6 Visual interface editor

➔ Design the program interface in a WYSIWYG manner by dragging the controls in the sidebar of the control to the main page of the program. Drag a QLabel widget to the main page of the program and set the text on QLabel to "Hello World".



Object	Class
MainWindow	QMainWindow
centralWidget	QWidget
label	QLabel
menuBar	QMenuBar
mainToolBar	QToolBar
statusBar	QStatusBar

MainWindow : QMainWindow

Property	Value
QObject	
objectName	MainWindow
QWidget	
enabled	✓
geometry	[(0, 0), 400 x 300]
X	0
Y	0
Width	400
Height	300
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited
font	A [SimSun, 9]
Family	Adobe Arabic
Point Size	9
Bold	☐
Italic	☐
Underline	☐
Strikeout	☐
Kerning	✓
Antialiasing	PreferDefault
cursor	Arrow
mouseTracking	☐
tabletTracking	☐
focusPolicy	NoFocus

Figure 4.7 Add a QLable widget

➔ Save the project, package the project code (remove the build related files) into the ubuntu cross-compilation environment.

2) Cross-compile hello project

```
#source environment-setup_qt
```

Enter into the hello directory, run:

```
#qmake
#make
```

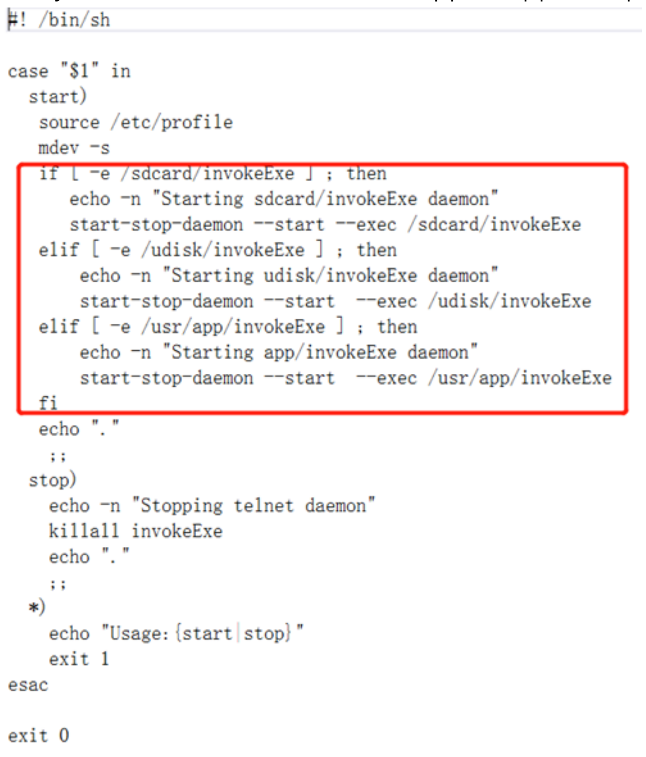
3) Run hello program.

To run the qt program on the EVK, you first need to set the environment variable of qt.:

```
#Resistive touchscreen
export TSLIB_TSDEVICE=/dev/input/event1
export TSLIB_CONFFILE=/etc/ts.conf
export TSLIB_PLUGINDIR=/usr/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal

#capacitive touch screen
export QWS_MOUSE_PROTO="LinuxInput:/dev/input/event0"
```

5. Run your app automatically on boot.

The system run the /etc/init.d/S51app-wrapper script automatically. Content as follow.



```
#! /bin/sh

case "$1" in
  start)
   source /etc/profile
   mdev -s
   if [ -e /sdcard/invokeExe ] ; then
      echo -n "Starting sdcard/invokeExe daemon"
      start-stop-daemon --start --exec /sdcard/invokeExe
   elif [ -e /udisk/invokeExe ] ; then
       echo -n "Starting udisk/invokeExe daemon"
       start-stop-daemon --start  --exec /udisk/invokeExe
   elif [ -e /usr/app/invokeExe ] ; then
       echo -n "Starting app/invokeExe daemon"
       start-stop-daemon --start  --exec /usr/app/invokeExe
   fi
   echo "."
    ;;
  stop)
    echo -n "Stopping telnet daemon"
    killall invokeExe
    echo "."
    ;;
  *)
    echo "Usage:{start|stop}"
    exit 1
esac

exit 0
```

After the system is powered on, the USB flash drive will be automatically mounted to the /sdcard directory. The TF card will be automatically mounted to the /udisk directory. After running the script, “invokeExe” program on the u disk or TF card or in the file system /usr/app directory will be run by default. If you want to run your own application, you can modify the contents of the red box of the script.