

Qt Widget Quick Start

Blake VERMEER

April 4, 2017

Date Performed: April 3, 2017
Company: Keysight Technologies

1 Overview

In this tutorial you will learn how to create a basic Qt Widget application from scratch and deploy it to the Keysight Hacking Platform development kit.

2 Creating a New Qt Widgets Project

- 1.) First we will use Qt Creator's new project wizard to create a new Qt Widgets project template. First open Qt Creator and then go to **File** → **New File or Project...**
- 2.) The new project wizard dialog box will pull up. Make sure the **Generic Linux Device Templates** option is selected in the drop-down menu in the upper right. Select **Applications** from the left hand menu under projects and then select the **Qt Widgets Application** project template and then click the **Choose...** button.

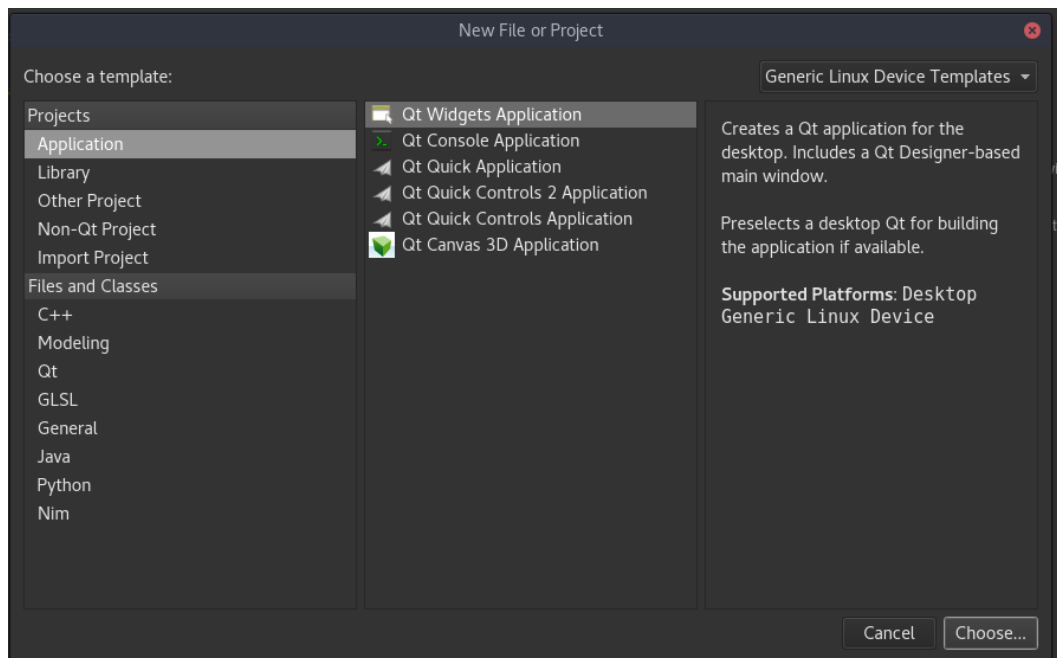


Figure 1: New Project Wizard Dialog Box

- 3.) In the next dialog box choose a name for the project and where to save it and then click the **Next** button.

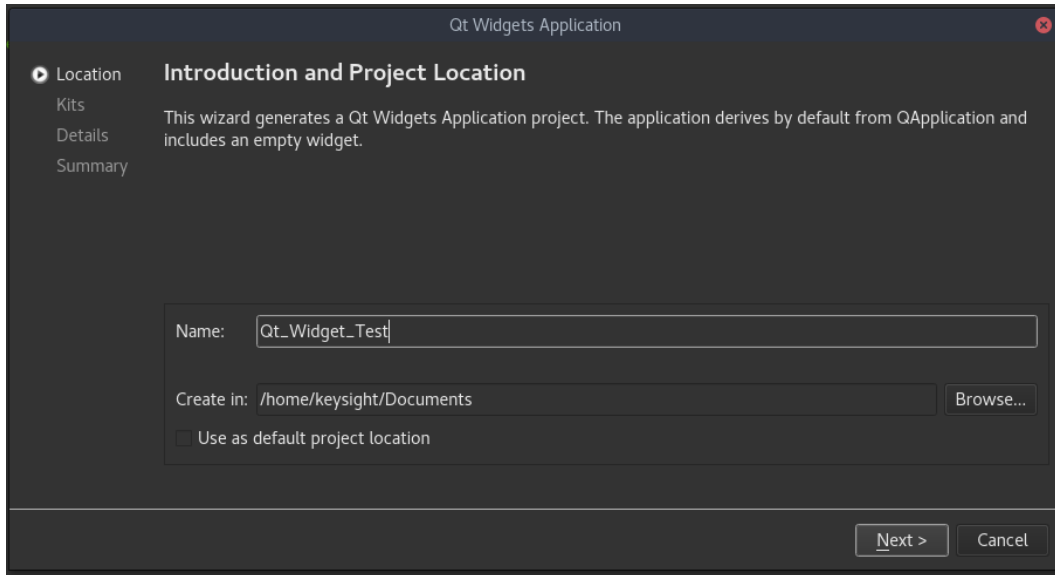


Figure 2: Name the Project

- 4.) In the next dialog box make sure that only the **RPI** box is selected since we are not trying to build a desktop application and then click the **Next** button.

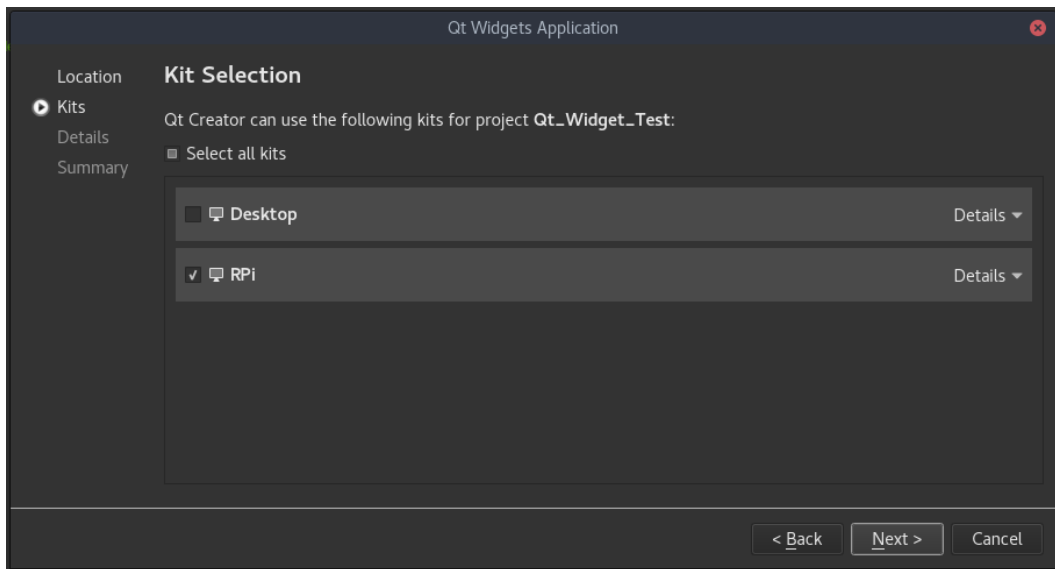


Figure 3: Kit Selection

- 5.) In the **Class Information** dialog box leave everything at their default values and click the **Next** button.

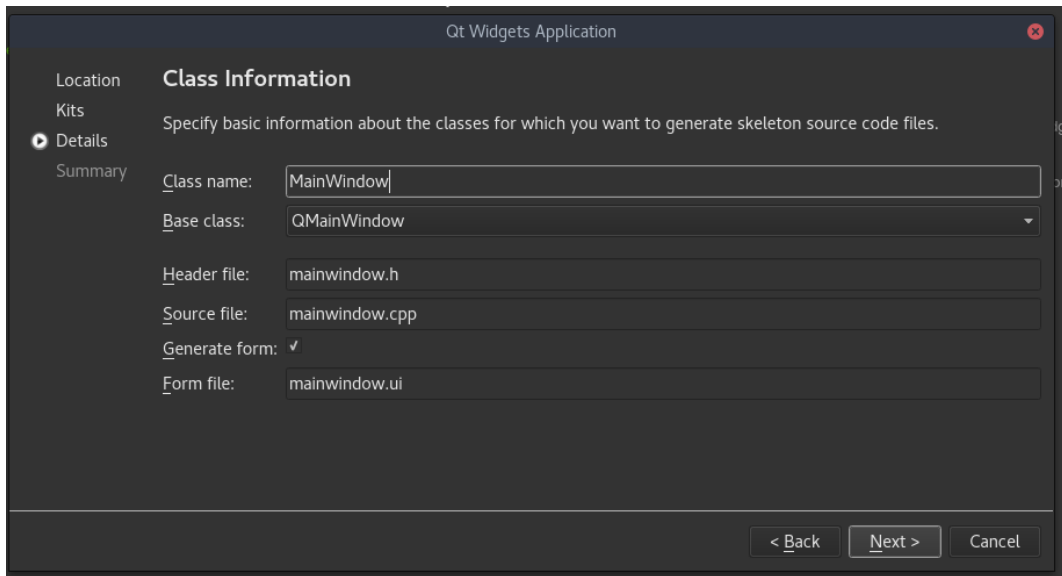


Figure 4: Class Information

6.) In the **Project Management** dialog box leave everything alone and click the **Finish** button.

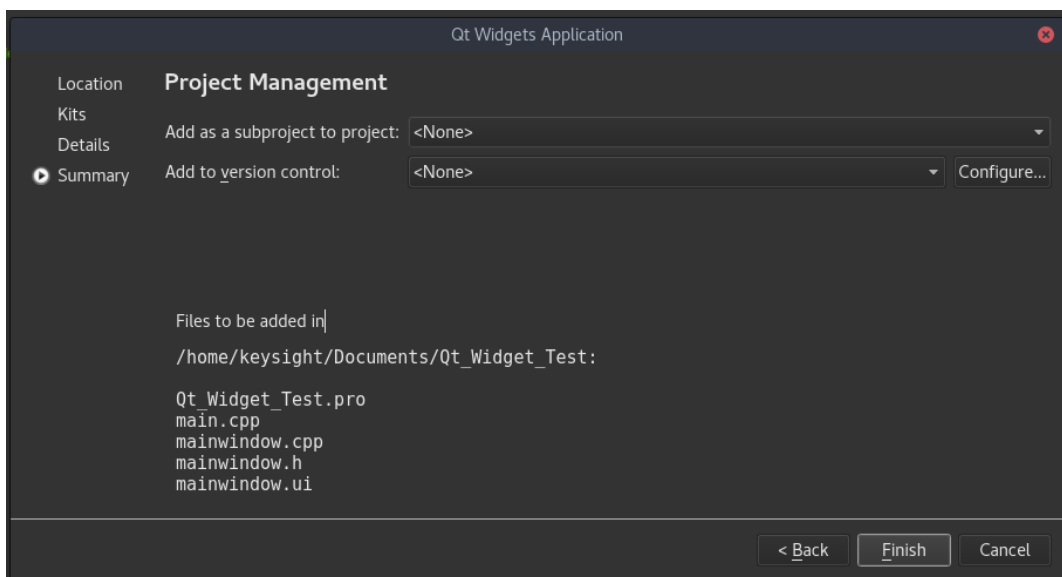


Figure 5: Project Management

3 Qt Widgets Project Overview

The Qt Widgets project wizard creates several different files. This section gives a brief overview of the files created and their purpose.

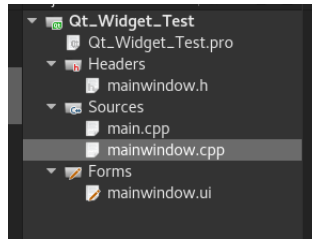


Figure 6: Project Files Created for a Project Name "Qt_Widget_Test"

Here is a brief overview of the files created for a project called Qt_Widget_Test and their purpose:

- **Qt_Widget_Test.pro** - The main project file. The project file defines the source files used by the project, the name of the application, where to deploy the application on the target device, and various other settings.
- **mainwindow.h** - A header file for the *mainwindow* class.
- **main.cpp** - The main source file for the application. It creates a *QApplication* object need for Qt Widget programs and then creates a *mainwindow* object and displays it.
- **mainwindow.cpp** - The source file for the *mainwindow* class. This class inherits from *Q_OBJECT* and defines the actions done by the various *mainwindow* GUI events.
- **mainwindow.ui** - The UI for for the *mainwindow* widget. This file defines the GUI layout for the application.

4 Preparing the Project File

The default created project file is almost fully complete. The only thing that needs to be added is a target path definition that will tell Qt Creator where to install the application on the target device when deploying it. It is important that the user account you are using on the target device has write access to the target path. Add the three lines as shown in Figure 7.

```
#-----
#
# Project created by QtCreator 2017-04-03T19:13:53
#
#-----

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Qt_Widget_Test
TEMPLATE = app

SOURCES += main.cpp\
           mainwindow.cpp

HEADERS  += mainwindow.h

FORMS    += mainwindow.ui

# Target install path
target.path = /home/keysight/bin
INSTALLS += target
```

Figure 7: Define the target install path

5 Creating the GUI

This section will explain how to make a basic GUI. Double-click the *mainwindow.ui* file. It will automatically open in the design view.

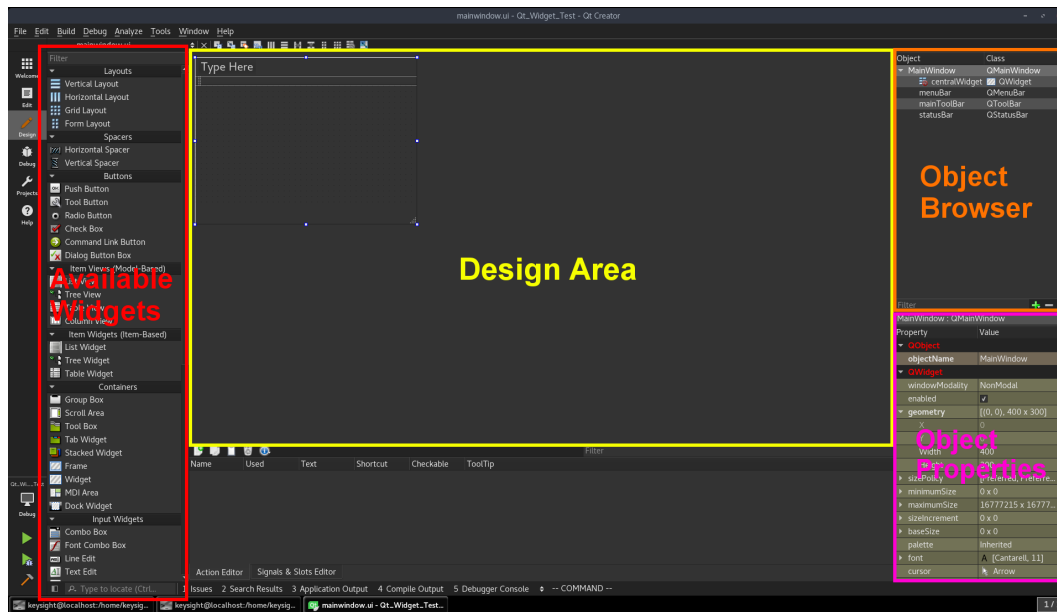


Figure 8: Qt Designer Overview

After opening up the design view the first thing to do is to delete the design objects that are not needed in our UI design. In the **Object Browser** right-click on the **statusBar**, **mainToolBar**, and **menuBar** object and click remove for each of these. Since we are designing an app which will only run on a device fullscreen with no window manager, we don't need these elements.

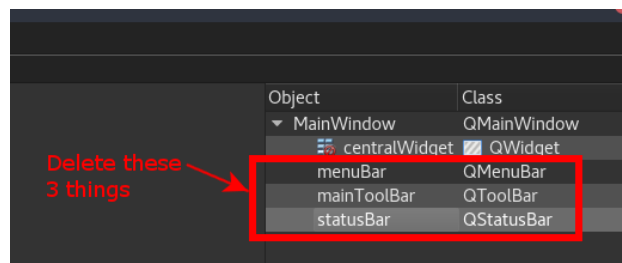


Figure 9: Delete unneeded GUI elements

Next we need to change the main window dimensions to the dimensions of our screen. Select the **Main-Window** from the object browser and then in the object properties dialog box expand the **geometry** section and change the **Width** to **320** and the **Height** to **240**.

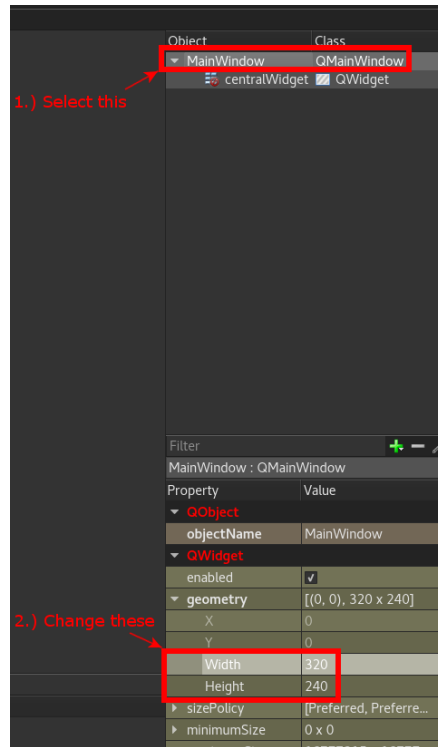


Figure 10: Change the main window dimensions to match the screen

At this point we can choose the color theme for our application. Select the **MainWindow** from the object browser and then go select **palette** from the object properties box and then press the **Change Palette** button next to it.

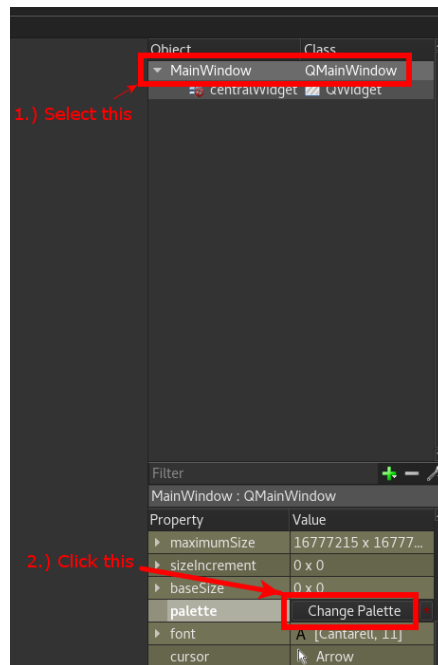


Figure 11: Change the color scheme for the app

This will pull up the **Edit Palette** dialog box. From this window you can change choose the colors of each individual window element or you can just click on the box to the right of the **Quick** label and have a color theme automatically generated based on the chosen color. Set up your color scheme and then press OK.

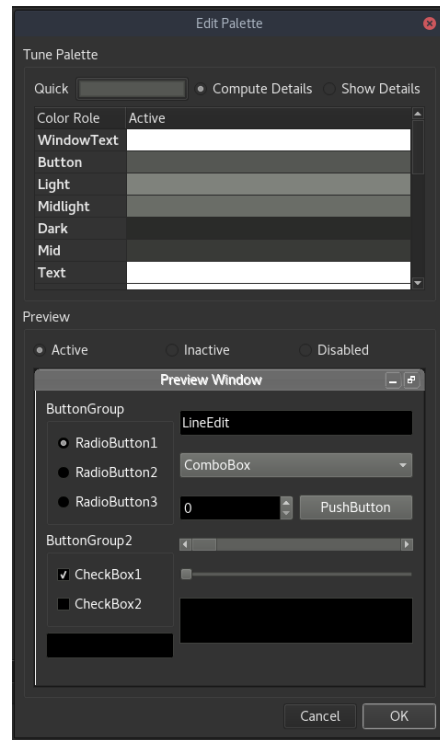


Figure 12: Edit the application color scheme

Now we are going to add three buttons to the UI. Find the **Push Button** widget in the widget browser and drag three of them to the UI design view. Place them as shown in Figure 13.

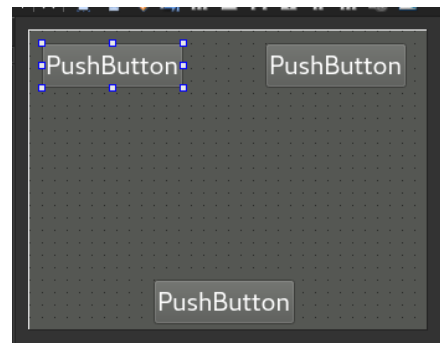


Figure 13: Place the push buttons

Now right-click on the upper left button and choose the **Change objectName...** option. Rename the upper left button to **plus_Button**. Now right-click on the upper left button again and this time choose **Change text...**. The text on the upper left button to **+**. Rename and change the text on the other two buttons as shown in Figure 14.

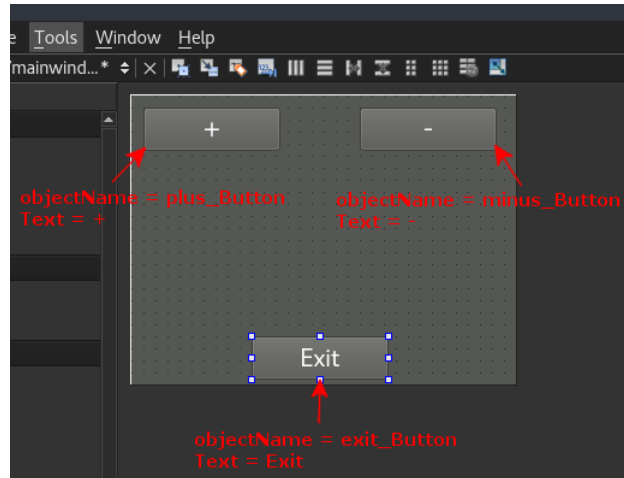


Figure 14: Set the objectNames and text for the buttons

Now find the **LCD Number** widget from the widget browser and drag it to the middle of the UI as shown in Figure 15. Resize the LCD display as you see fit.

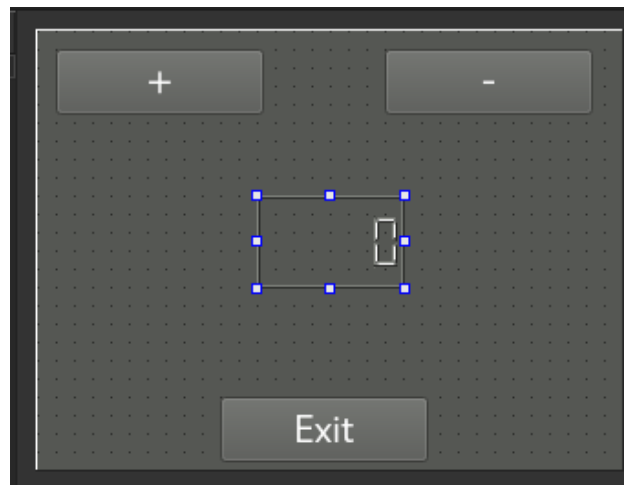


Figure 15: Add a LCD widget to the UI

6 Create Button Signals

Now we are going to change the design view to the **Edit Signals/Slots** view to edit the events that are called when the buttons are pressed. In the bar directly above the design area click the **Edit Signals/Slots** button as shown in Figure 16.

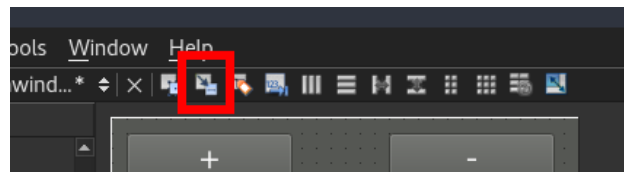


Figure 16: Edit Signals/Slot Button

Click and drag from the exit button until a red wire with a ground symbol appears and then release your mouse. This will bring up a **Configure Connection** dialog box for the **Exit** button.

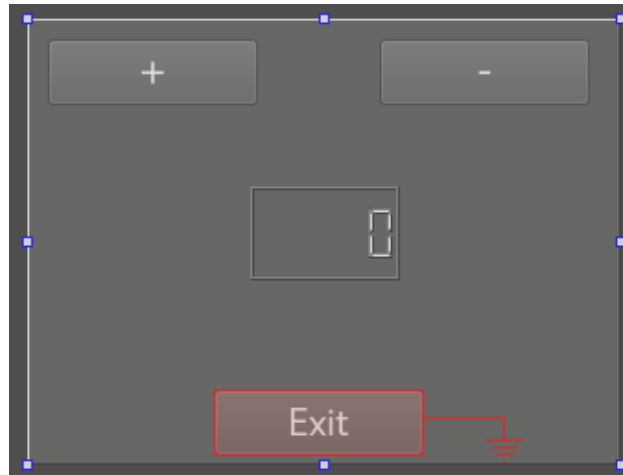


Figure 17: Create a new signal for the Exit button

In the **Configure Connection** dialog box choose **clicked()** for the exit.Button signal, then check the box to 'Show signals and slots inherited from QWidget', and then choose **close()** for the event. This will cause the application to close when the exit button is pressed.

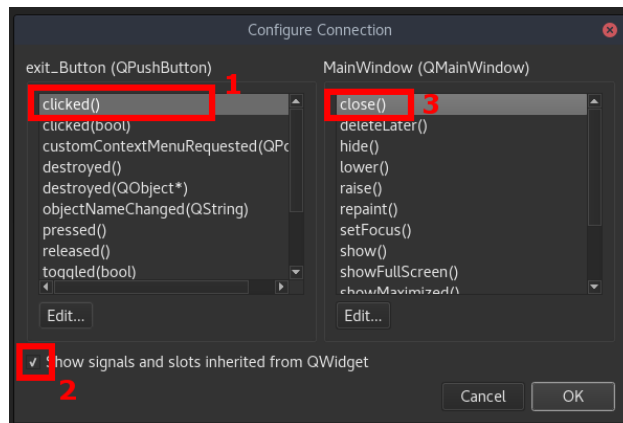


Figure 18: Set the Exit button to close the application

Now we will go back to the **Edit Widgets** view by clicking the **Edit Widgets** button in the toolbar above the design area to add signal function stubs for the other two buttons.

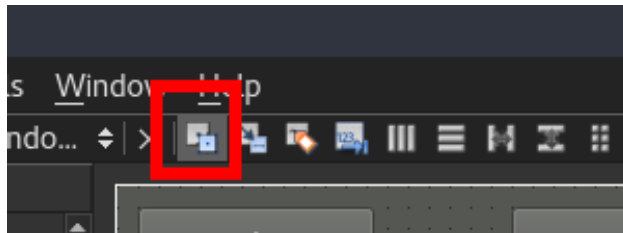


Figure 19: Switch back to Edit Widgets mode

Now select the **Plus Button** and right click it and choose **Go to slot....** This will pull up a dialog box and in it select the **clicked()** signal and then press OK. This will create a function stub for us in the MainWindow class that is called when the plus button is clicked.

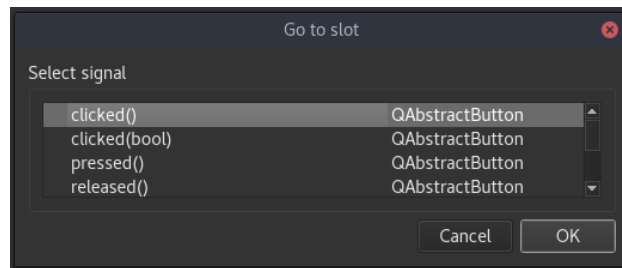


Figure 20: Go to slot dialog

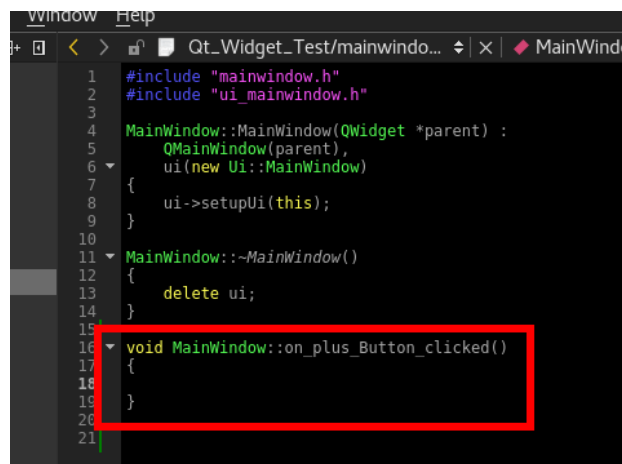


Figure 21: New signal function stub

For now leave the function stub alone and double-click on the **mainwindow.ui** file under Forms to go back to the GUI designer.

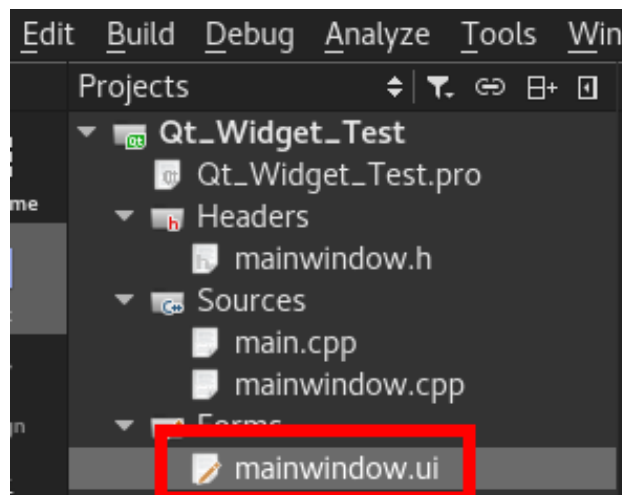
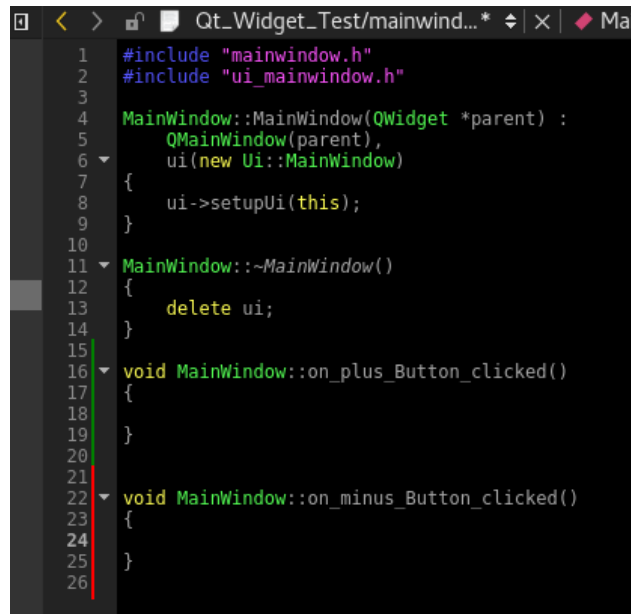


Figure 22: Go back to editing the UI file

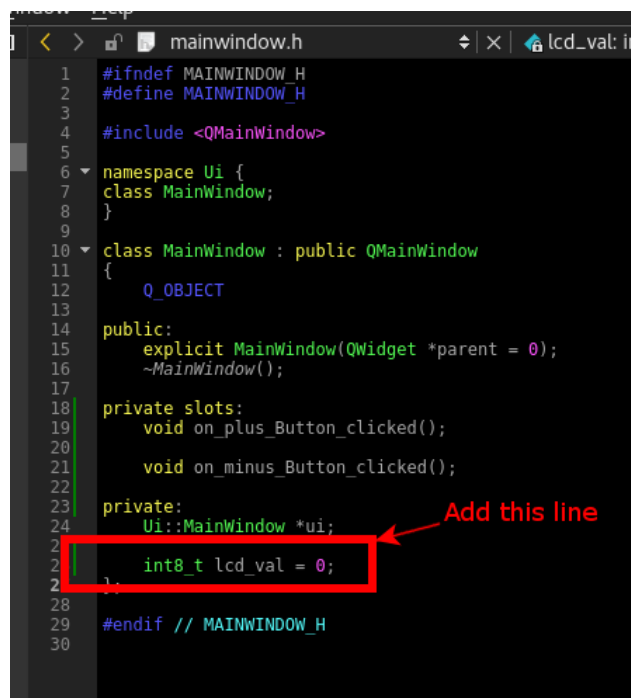
Right-click on the **minus button** and click **Go to slot...** and chose the clicked option similar to what was done with the plus button. At this point you should be seeing the **mainwindow.cpp** file and it should look like Figure 23.



```
1  #include "mainwindow.h"
2  #include "ui_mainwindow.h"
3
4  MainWindow::MainWindow(QWidget *parent) :
5      QMainWindow(parent),
6      ui(new Ui::MainWindow)
7  {
8      ui->setupUi(this);
9  }
10
11 ~MainWindow()
12 {
13     delete ui;
14 }
15
16 void MainWindow::on_plus_Button_clicked()
17 {
18 }
19
20
21
22 void MainWindow::on_minus_Button_clicked()
23 {
24 }
25
26
```

Figure 23: mainwindow.cpp with function prototypes for the buttons

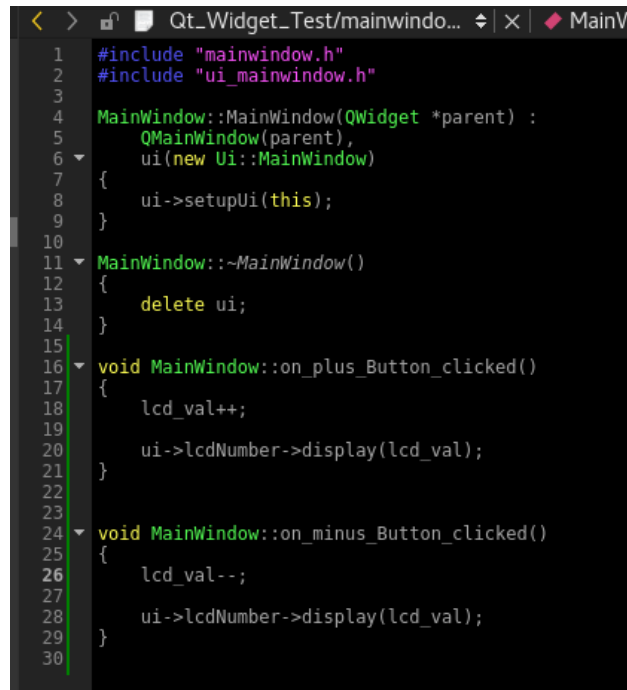
Now double-click the **mainwindow.h** file to open it. Add a new private variable to keep track of the LCD display number as shown in Figure 24.



```
1  #ifndef MAINWINDOW_H
2  #define MAINWINDOW_H
3
4  #include <QMainWindow>
5
6  namespace Ui {
7      class MainWindow;
8  }
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     explicit MainWindow(QWidget *parent = 0);
16     ~MainWindow();
17
18 private slots:
19     void on_plus_Button_clicked();
20
21     void on_minus_Button_clicked();
22
23 private:
24     Ui::MainWindow *ui;
25     int8_t lcd_val = 0;
26
27 };
28
29 #endif // MAINWINDOW_H
30
```

Figure 24: Add a private variable called lcd_val

Now double-click on the **mainwindow.cpp** file to open in again. Fill out the button function as shown in Figure 25.



```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9 }
10
11 MainWindow::~MainWindow()
12 {
13     delete ui;
14 }
15
16 void MainWindow::on_plus_Button_clicked()
17 {
18     lcd_val++;
19
20     ui->lcdNumber->display(lcd_val);
21 }
22
23
24 void MainWindow::on_minus_Button_clicked()
25 {
26     lcd_val--;
27
28     ui->lcdNumber->display(lcd_val);
29 }
30
```

Figure 25: Completed button functions

At this point the program is complete and is ready to be deployed to the Raspberry Pi to be tested. Click the **Start Debugging** button in the lower left hand side of the screen (or press the F5 key) to deploy the application to the Raspberry Pi and open up a debugging session for the application.

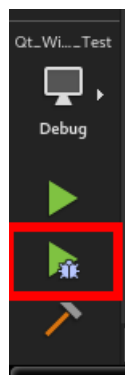


Figure 26: Start Debugging Button