Toradex
Linux QTアプリケーション
開発マニュアル

本マニュアルは岡本無線電機株式会社が独自作成したものでありメーカーが保証した内容ではありません。万が一本マニュアルに間違いがあり、事故が生じたとしても岡本無線電機株式会社は一切の責任を問われないものとさせていただきます。

変更履歴

バージョン	更新日付	変更内容
2.00	2020/08/19	BSP3.0.4向けに作成

本マニュアルについて

本マニュアルはトラデックスのCPUモジュール上で動作するQTアプリケーションを作成する手順を記述しています。
参考:
https://developer.toradex.com/knowledge-base/how-to-set-up-qt-creator-to-cross-compile-for-embedded-linux

1. 実行環境

本マニュアルの実行環境は下記です。

仮想化ソフト:VMWARE Player v15.5.6
Host OS: Windows 10 1909
Guest OS: Ubuntu Desktop 18.04LTS 64bit(英語版)
BSP:v3.0.4
QTバージョン:5.11.3

CPUモジュール:Colibri-i.MX7D 512MB V1.1C
キャリアボード:Colibri 評価ボード Rev 3.2 + アクセサリーキット
液晶:感圧式7インチタッチパネル付き液晶(EDT 7.0)

本マニュアルとは異なるモジュールや評価ボード以外のキャリアボードを使われても大雑把には同じ操作となります。

インターネット接続環境が必要になります。

2. 事前準備

本マニュアルはLinux OSイメージ開発マニュアルの内容をすべて終えた状態で進めています。
OpenEmbeddedのディレクトリは/work/oe-coreです。

液晶も使用するためOSカスタムマニュアルのカーネルとブートローダーのカスタマイズ内容を適用してLCDの設定を変更していることが望ましいです。LCDを評価ボードのX34に接続してください。VGAなどで代用することも可能ですがタッチパネルの機能は使用できません。

3. 前提知識

Linux OSイメージ開発マニュアルの内容をご理解いただいた状態を前提としています。

4. 注意点

オープンソース系を利用した開発に共通することですがすべてを理解しようとするときりがなく開発効率を損ないます。必要なタイミングで必要な知識を身につけるというスタンスで理解することを推奨いたします。

開発環境と実行環境の違いをわかりやすくするためにコマンドの表記の前に下記をつけています。
開発環境(PC)上で入力するコマンド：[ubuntu]$
実行環境(モジュール)上で入力するコマンド：[colibri-imx7]#

コピーについて
本マニュアル内のコマンドなどをコピーした場合、改行が入ったり「-」が抜けてしまうことがあるのでご注意ください。一度テキストエディタなどに張り付けてコピーした内容をご確認ください。

OSイメージ作成

最初にQTを含んだOSイメージを作成する必要があります。
OSイメージを作るにあたりQt Platform Abstractionというバックエンド機能の違いによって作成方法が変わってきます。

1．描画のバックエンド機能にOpenGLを使用するのかソフトウェアレンダリングをするのか
　OpenGLを使うかソフトウェアレンダリングを行うかは使用するモジュールにGPUがありOpenGLを使用するプラグインが提供されているかどうかで決まります。基本的にOpenGLが使える場合はデフォルトでOpenGLを使う構成のレシピとなっています。
それ以外の場合ソフトウェアレンダリングとなります。

2．デスクトップ機能を使用するかどうか
　デスクトップ機能があるOSイメージはないものに比べ描画が遅くなります。デスクトップ機能なしでOSイメージを作成した方が高速になりますがミドルウェアによってはデスクトップ機能がないと動作しないものがありますのでその場合デスクトップ機能を使います。

上記2点の違いから4パターン存在します。
・OpenGL　デスクトップあり
・OpenGL　デスクトップなし
・ソフトウェアレンダリング　デスクトップあり
・ソフトウェアレンダリング　デスクトップなし

デスクトップなしOSイメージ

ソフトウェアレンダリングでデスクトップ機能なしの場合についての手順

Open Embeddedの準備
cd /work/oe-core/
. export

[ubuntu]$ gedit ./conf/local.conf

下記を追記します。
DISTRO_FEATURES_remove = "wayland x11"
IMAGE_INSTALL_append = " qtbase qtbase-plugins qtdeclarative liberation-fonts qtquickcontrols2"
IMAGE_INSTALL_append = " tslib-calibrate"
PACKAGECONFIG_append_pn-qtbase = "linuxfb tslib"
PACKAGECONFIG_FONTS_append_pn-qtbase = " fontconfig"

QTは機能ごとにレシピが用意されているため必要な機能を容易に選択可能です。上記以外にも必要な機能がある場合は該当するレシピを追加してください。本マニュアルでは感圧式タッチパネルも使用するためtslibも入れています。

OSイメージを作成します。
[ubuntu]$ bitbake console-tdx-image

モジュールに出来上がったOSイメージを書き込んでください。
下記のような名前のファイルが出力されています。
Colibri-iMX7_Console-Image-Tezi_3.0b4.tar

SDKの作成

SDKにはOSイメージに含まれる機能に関係なくデフォルトで多くの機能が付加されています。その分bitbakeにも時間がかかります。
どのようなものが入っているかは下記のレシピを見るとわかります。
/work/oe-core/layers/meta-qt5/recipes-qt/packagegroups/packagegroup-qt5-toolchain-target.bb

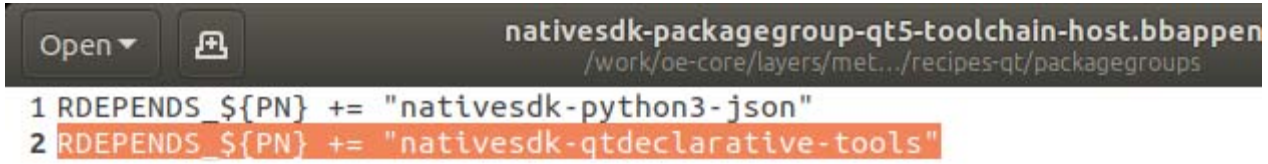
特にqt3d、qtwebkitに関しては開発環境のRAMが小さいとcompileに失敗します。(RAM16GB、swap8GB程度必要です)
不要の場合はbitbakeの対象から外します。bitbakeが早くなりSDKの容量も小さくなります。
本マニュアルの例ではOSイメージに入っていない機能であるため使用することはありません。
[ubuntu]$ gedit ./conf/local.conf

追記
RDEPENDS_packagegroup-qt5-toolchain-target_remove =" qt3d-dev qt3d-mkspecs qt3d-qmlplugins qtwebkit-dev "

QT5.11からQt Quick CompilerというQMLを事前にコンパイルする機能が使用できるようになりました。
その機能を使用するためにnativesdk-qtdeclarative-toolsの追加をします。

[ubuntu]$ gedit /work/oe-core/layers/meta-toradex-demos/qt5-layer/recipes-qt/packagegroups/nativesdk-packagegroup-qt5-toolchain-host.bbappend

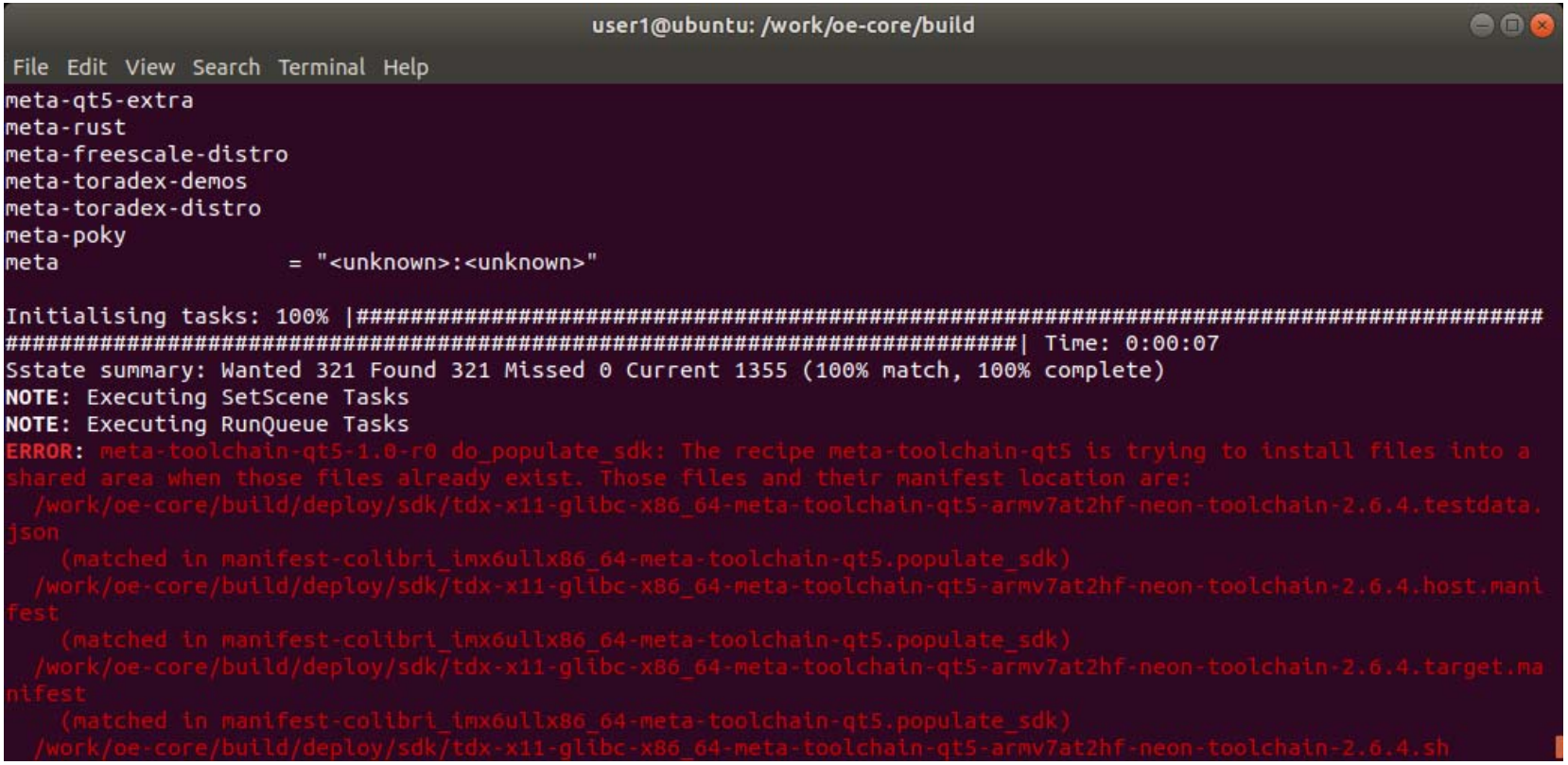
追記
RDEPENDS_${PN} += "nativesdk-qtdeclarative-tools"



SDKを作成します。
[ubuntu]$ bitbake meta-toolchain-qt5

ほかのモジュールでSDKを出力した後などSDK作成時は下記のようなSDKがすでに存在するというエラーが出るときがあります。

sdkを一度削除してから再度実行してください。
[ubuntu]$ rm -rf ./deploy/sdk/*



/work/oe-core/build/deploy/sdk配下にSDKが出力されます。

作業ディレクトリ作成
[ubuntu]$ mkdir /work/qt

再度sdkを出力するなどする場合に備えてSDKをapp配下にバックアップしておきます。(任意)
cp -rf /work/oe-core/build/deploy/sdk /work/qt/sdk

SDKのインストールをします。(モジュールやBSPのバージョンによってシェルの名前が変わります。)
[ubuntu]$ /work/oe-core/build/deploy/sdk/tdx-x11-glibc-x86_64-meta-toolchain-qt5-armv7at2hf-neon-toolchain-2.6.4.sh

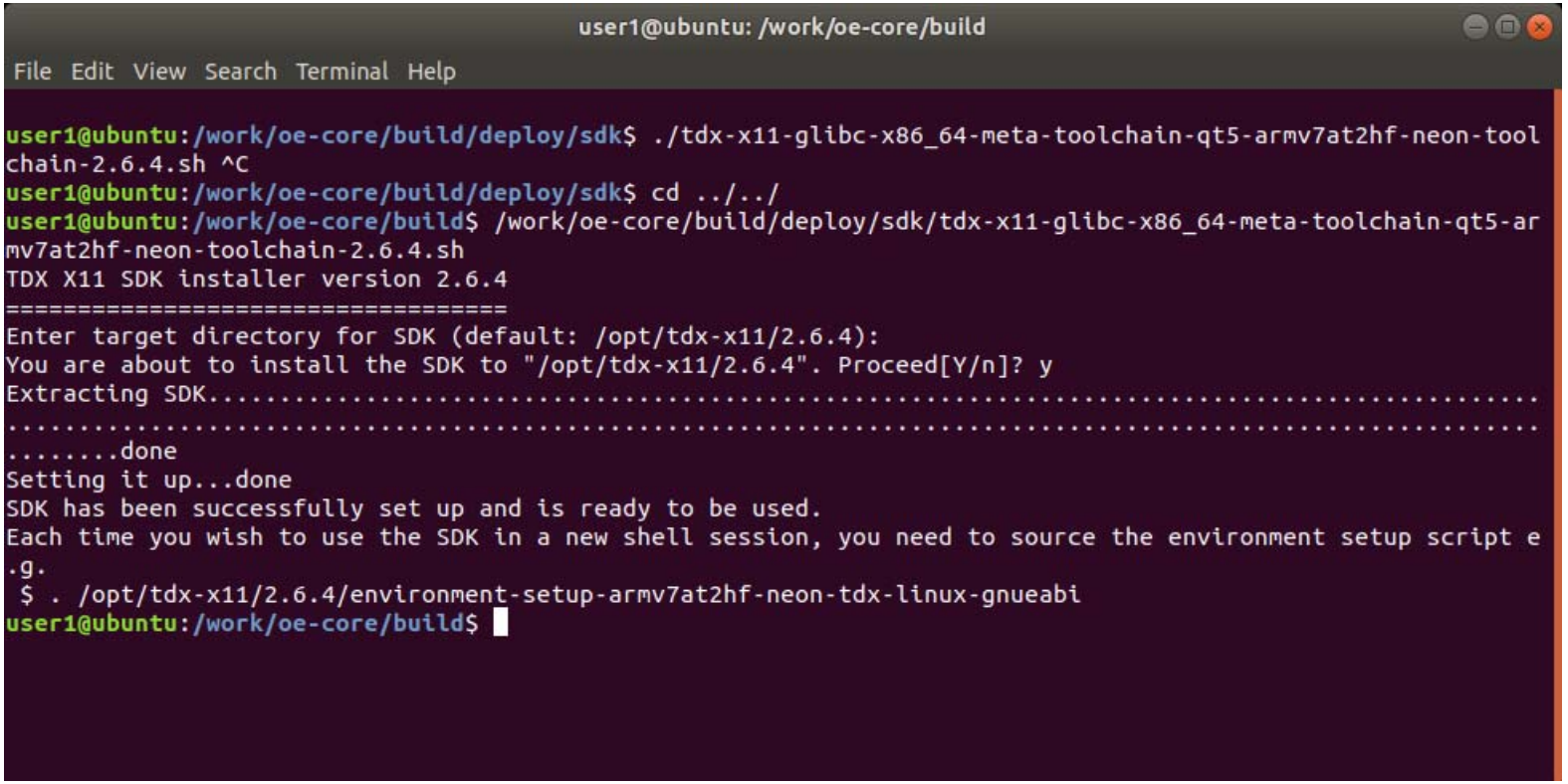
下記のようにSDKのインストールディレクトリを問われるので入力します。
[ubuntu]$ Enter target directory for SDK (default: /opt/tdx-x11/2.6.4):
本マニュアルではデフォルト設定のままインストールします。Enterキーを入力します。

下記のように問われますので
[ubuntu]$ You are about to install the SDK to "/opt/tdx-x11/2.6.4". Proceed[Y/n]?
Yを入力してEnterキーを押します。
管理者権限で実行しますので途中でパスワードを問われます。パスワードを入力してください。

すでに存在していた場合は下記のように上書きするかどうかを問われますがその場合は一度シェルを停止してシェル実行前にディレクトリを削除しておいたほうが良いです。
If you continue, existing files will be overwritten! Proceed[y/N]?

[ubuntu]$ sudo rm -rf /opt/tdx-x11/2.6.4

最後に下記のような環境変数設定シェルのパスの案内があります。このシェルは後の工程で使用します。
. /opt/tdx-x11/2.6.4/environment-setup-armv7at2hf-neon-tdx-linux-gnueabi



QT開発環境のインストール

QTの開発はQT Creatorのみで開発が可能です。本マニュアルではQT Creatorに加えてサンプルコードを入手します。
QTのサイトからQTのインストーラを入手してインストールします。

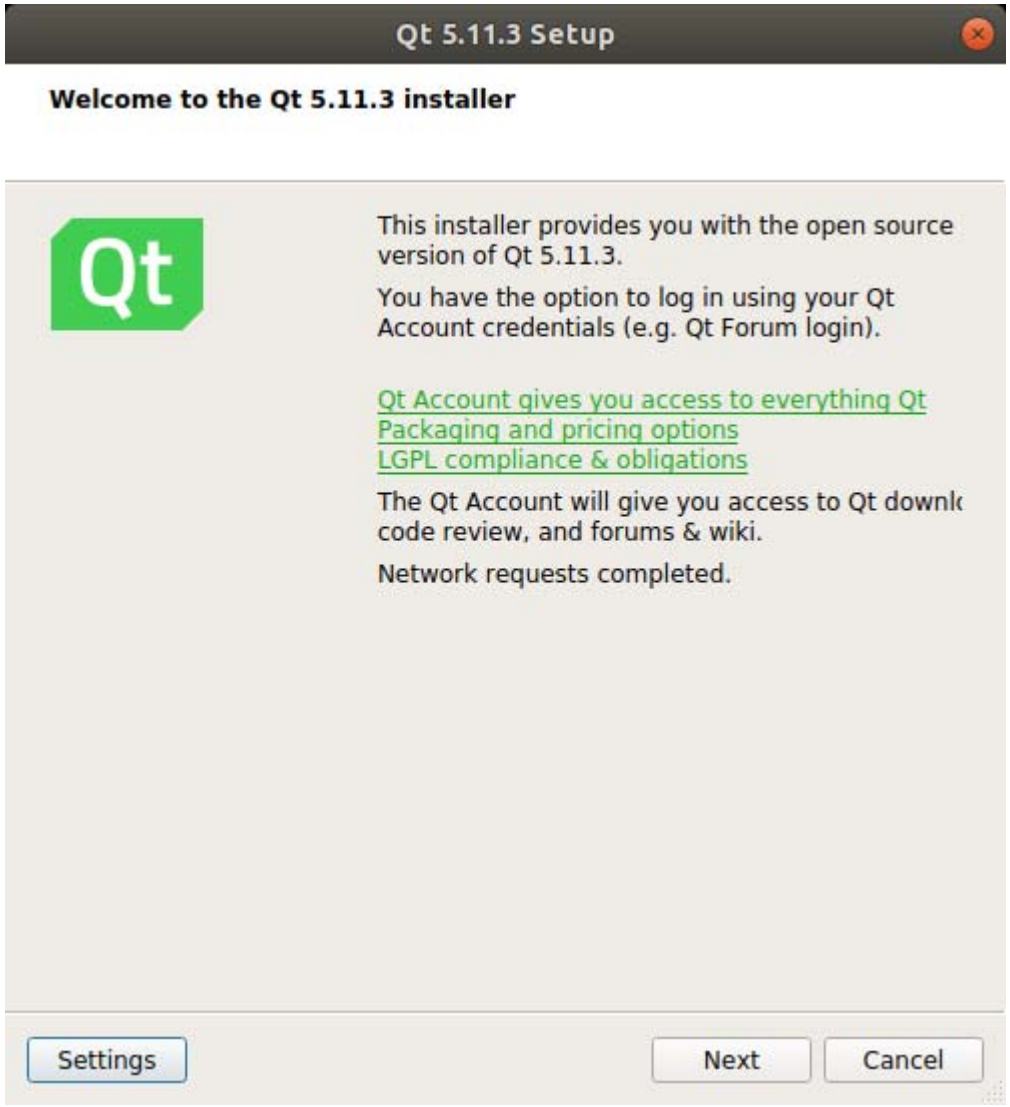
BSP3.0.4のQTのバージョンは5.11.3です。
インストーラーのダウンロード
[ubuntu]$ wget http://download.qt.io/new_archive/qt/5.11/5.11.3/qt-opensource-linux-x64-5.11.3.run

実行権限付与
[ubuntu]$ chmod +x ./qt-opensource-linux-x64-5.11.3.run

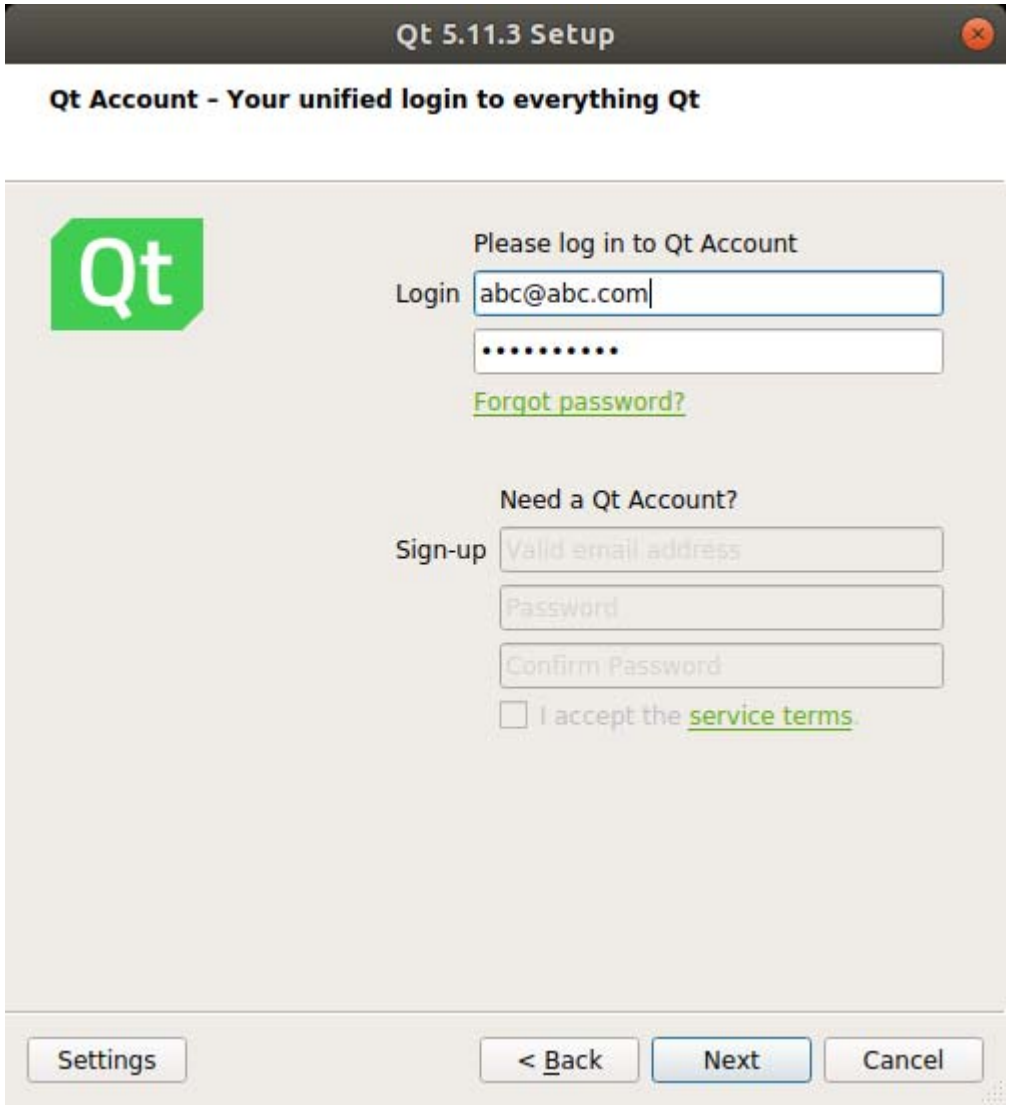
QTのインストール用ディレクトリを作成します。
[ubuntu]$ mkdir ./inst

インストーラ実行
[ubuntu]$ ./qt-opensource-linux-x64-5.11.3.run

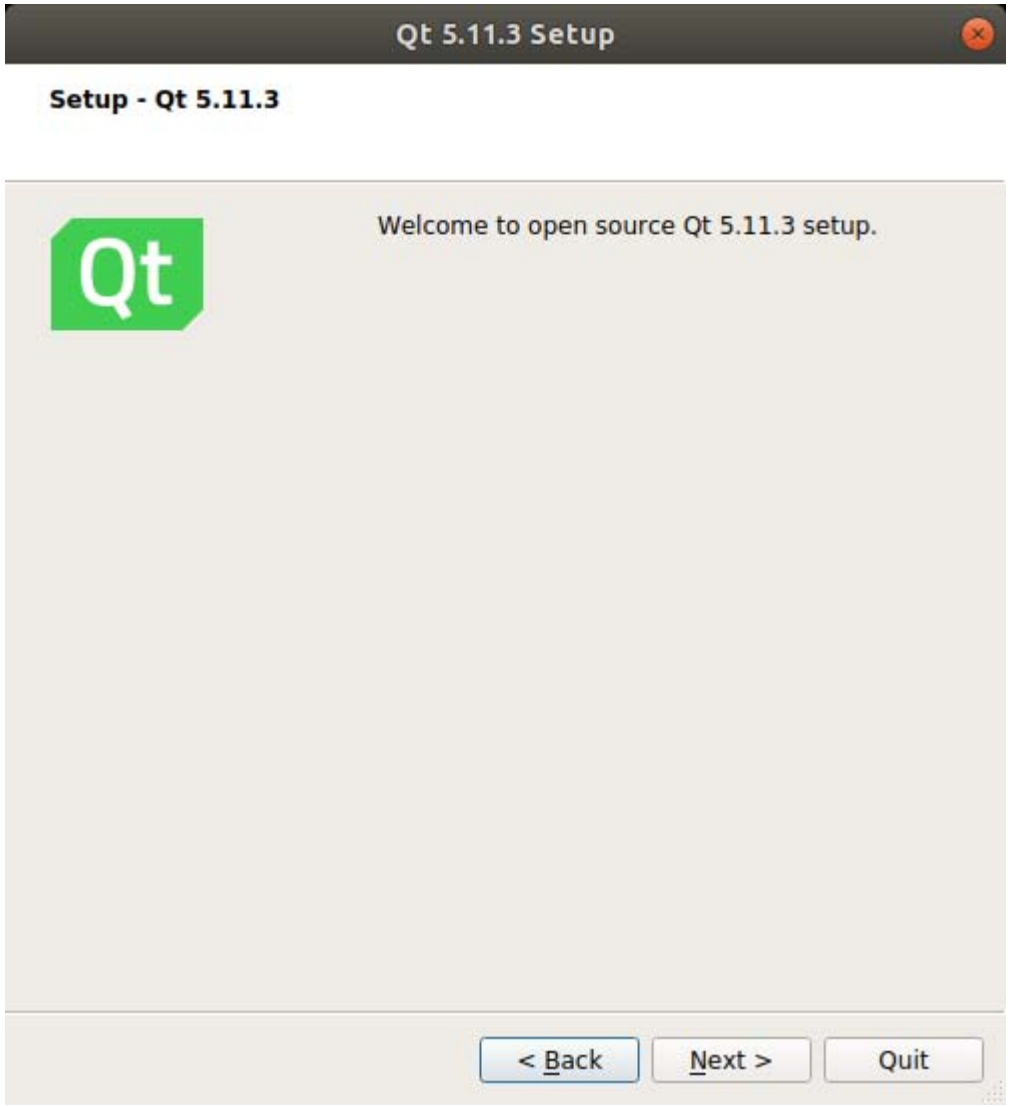
Nextをクリックします。



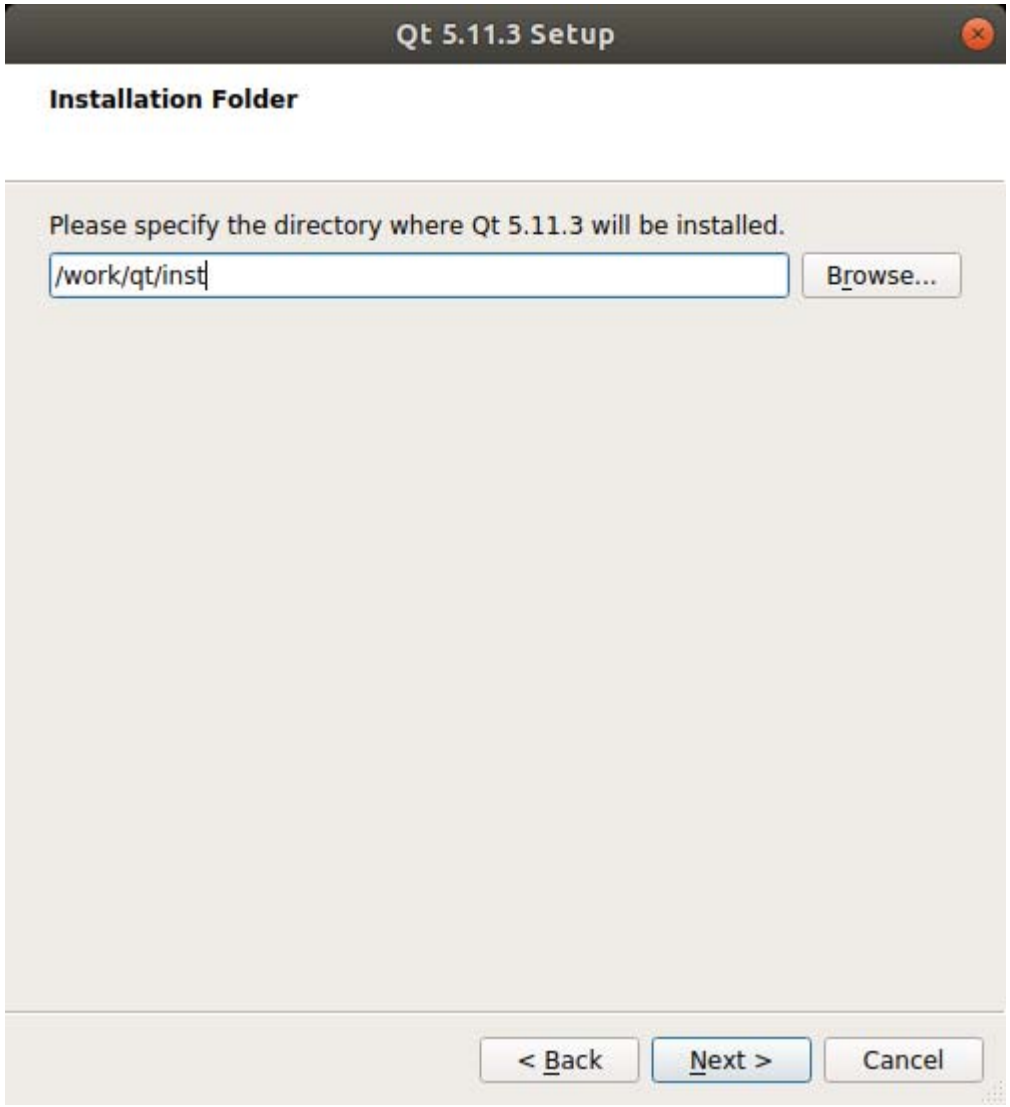
ログインIDとパスワードを入力してNextをクリックします。
QTのアカウントを持っていない場合はNeed a Qt Account?の下の項目を入力してアカウントを作成する必要があります。



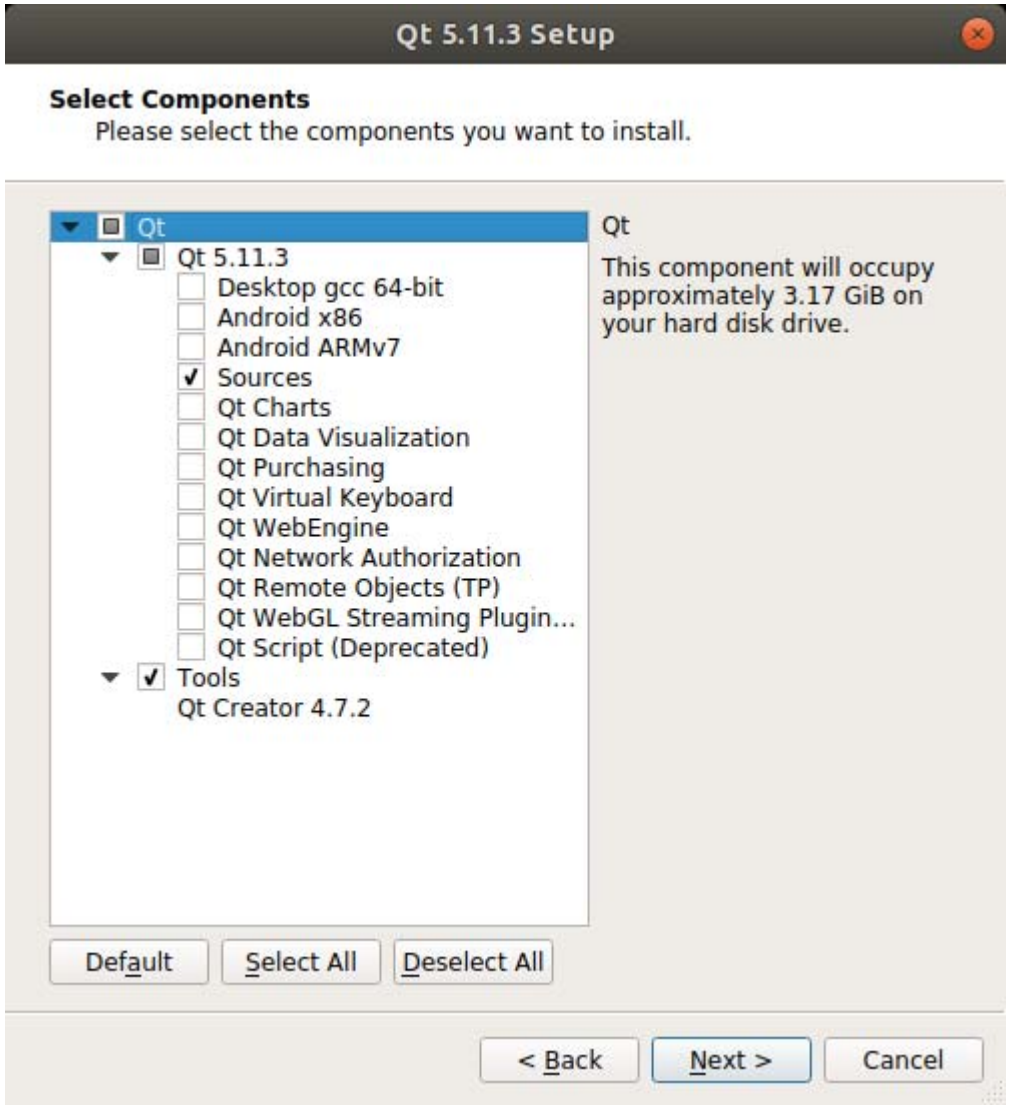
Nextをクリックします。



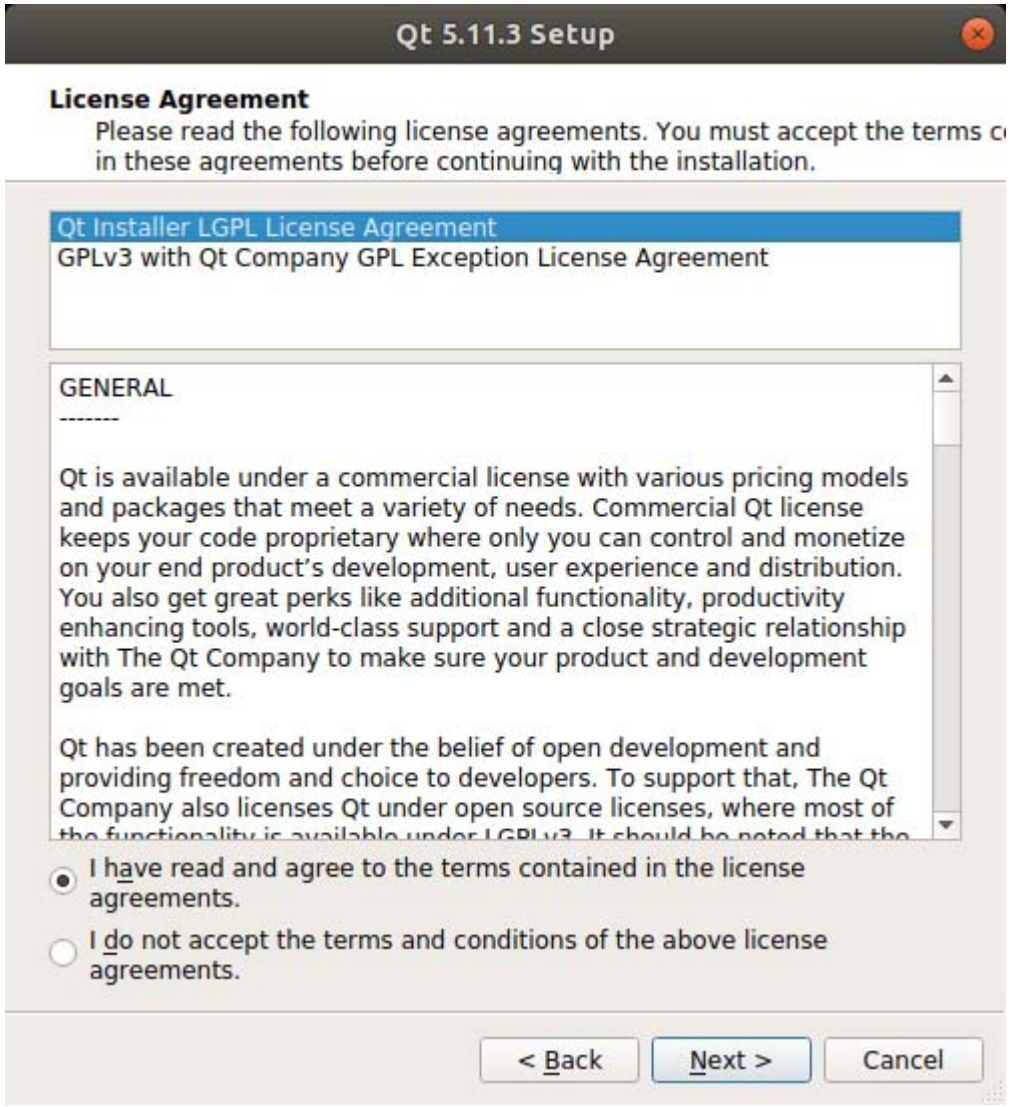
QTのインストールディレクトリを入力します。本マニュアルでは/work/qt/instです。入力後Nextをクリックします。



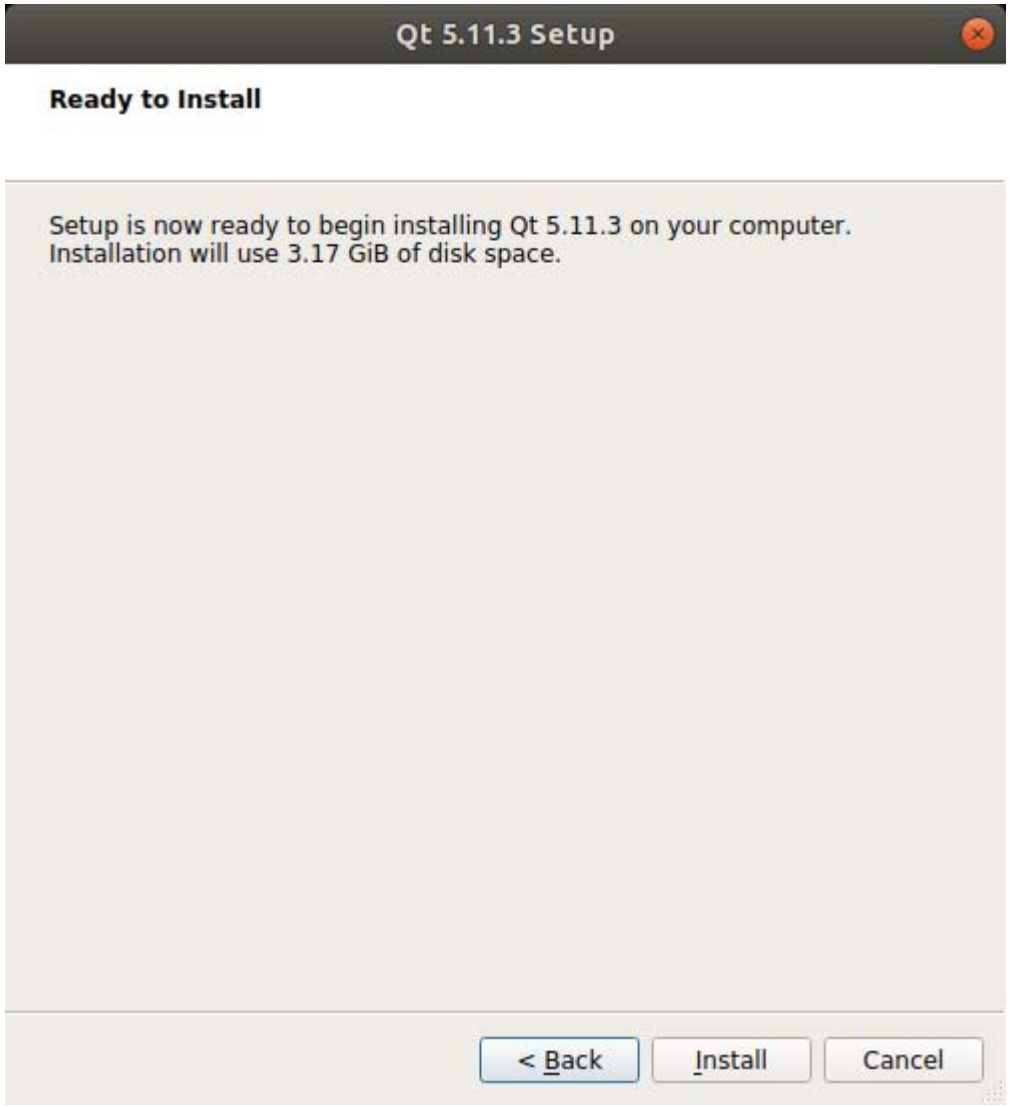
QT5.11.3のsourcesにチェックを入れてNextをクリックします。Qt Creator4.7.2にはデフォルトでチェックが入っています。



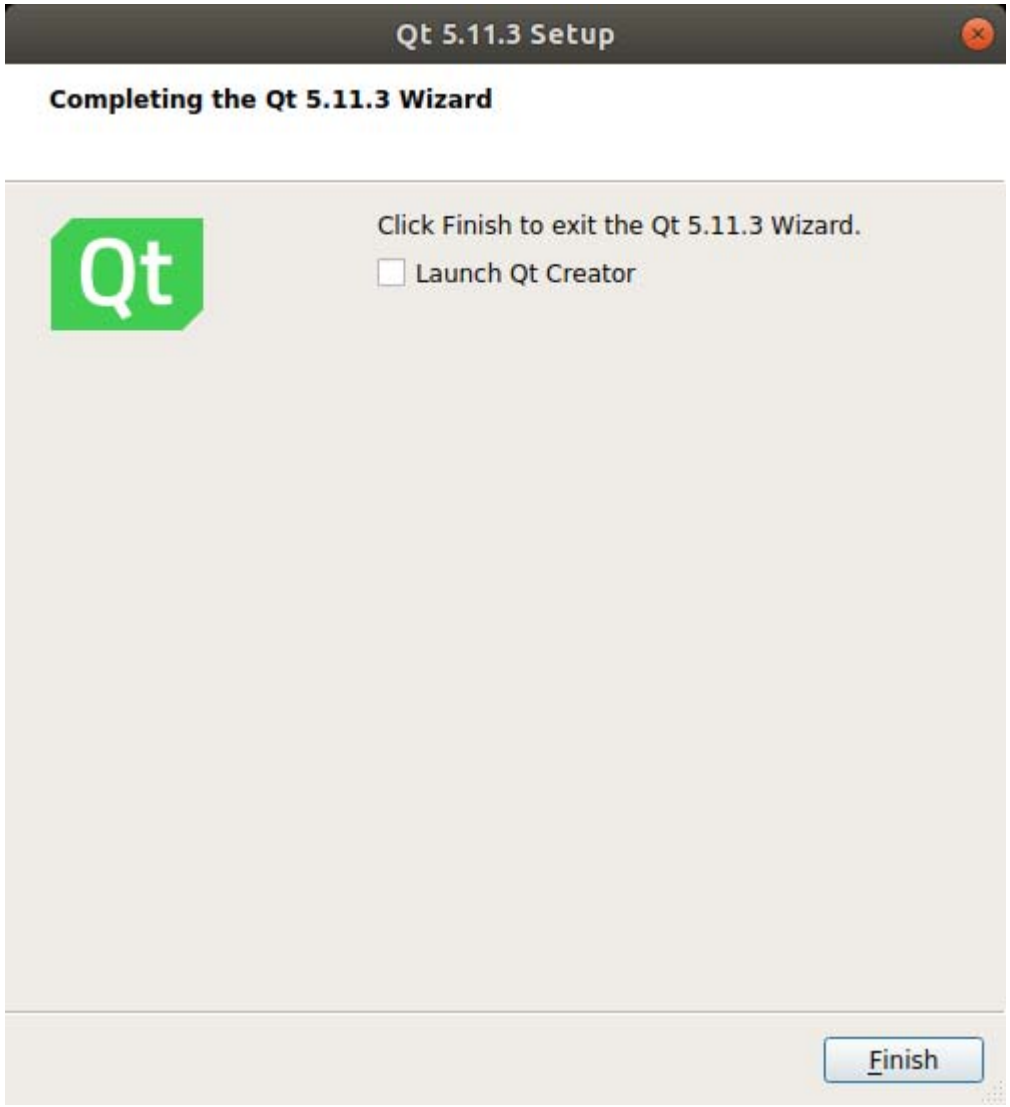
ライセンス規約に同意できる場合は「I have read and agree to the terms contained in the license agreements.」にチェックを入れてNextをクリックします。



Installをクリックします。



「Launch QT Creator」のチェックを外してFinishをクリックします。



QT-Creator実行シェル作成

QT Creatorは/work/qt/inst/Tools/QtCreator/bin/qtcreatorにインストールされています。
QT Creator起動前にSDK用環境変数を定義する必要があります。毎回起動前に環境変数の設定シェルを実行しないといけません。
設定漏れが起きないようにQT Creator起動用のシェルを作成します。
環境変数設定シェルは搭載するARMのアーキテクチャやBSPのバージョン、SDK出力ディレクトリなどによって変わります。(赤線部分)

```
[ubuntu]$ cd /work/qt/
[ubuntu]$ gedit ./sdk_qtcreator.sh
```

内容は下記です。

```
#!/bin/sh

. /opt/tdx-x11/2.6.4/environment-setup-armv7at2hf-neon-tdx-linux-gnueabi
./inst/Tools/QtCreator/bin/qtcreator
```

作成したファイルに実行権限を付与します。
[ubuntu]$ chmod +x ./sdk_qtcreator.sh

QTのワークスペース用ディレクトリを作成します。
[ubuntu]$ mkdir ./workspace

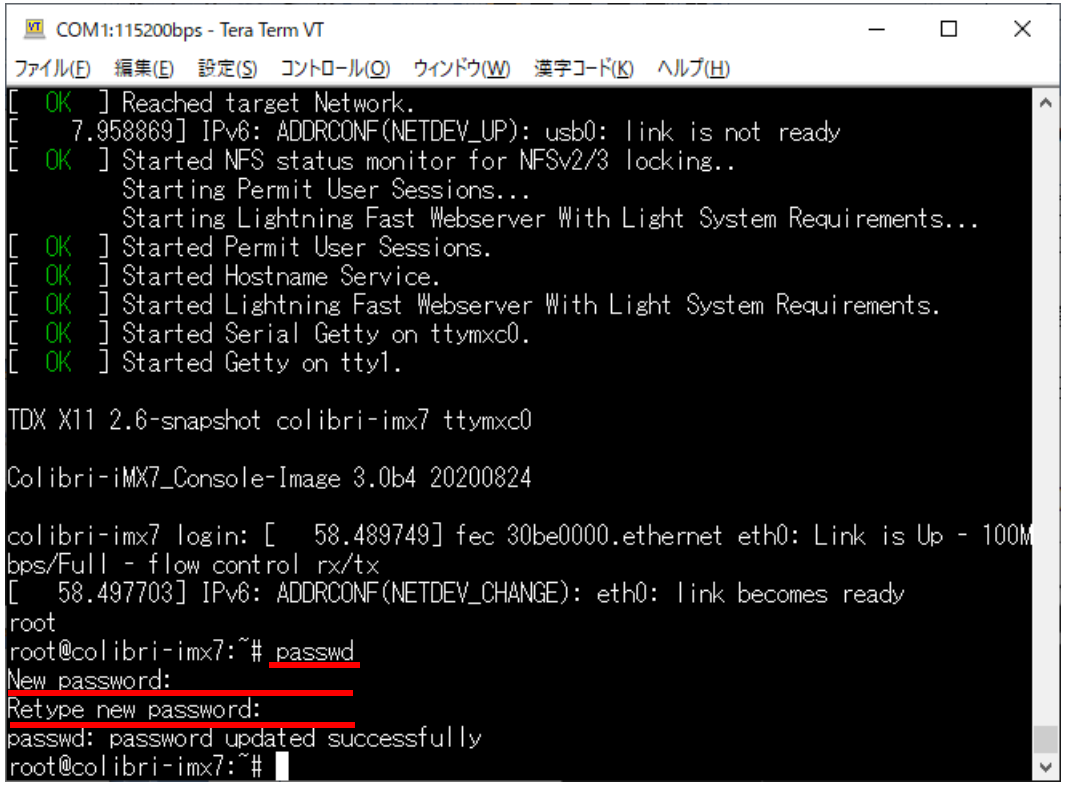
Eclipse実行
[ubuntu]$ ./sdk_qtcreator.sh

以後QT Creatorを起動する場合はこのシェルを使ってください。

次にQT Creatorの設定を行いますがQT CreatorはSSHを使用してデバッグを行います。
最初にモジュールとSSH接続できるようにモジュールにSSHの設定を行います。

SSH接続設定作成

デバッグに使用するGDBはEthernetで接続してSSHプロトコルを利用してデバッグを行います。
デバッグを行うためにモジュールとSSHで接続できるようにしておく必要があります。
モジュール側の設定を行います。
モジュールにEthernetケーブルを挿入し開発パソコンと同じサブネットに接続します。
Teratermを起動してモジュールを起動します。rootでログインしpasswdコマンドでパスワードを設定します。
本マニュアルではセキュリティを一切気にせず利便性のよいパスワード認証を使い、rootでSSHにログインできるようにします。
あくまでデバッグ目的で設定するだけです。
[colibri-imx7]# passwd
パスワードを2回入力するとパスワードが設定されます。(2回目は確認用)



次にモジュールのIPアドレスの設定を行います。何も設定していない場合はDHCPとなります。
設定ファイル/etc/systemd/network/wired.networkを新規作成します。

[colibri-imx7]# vi /etc/systemd/network/wired.network

DHCPの場合

```
[Match]
Name=eth0

[Network]
DHCP=ipv4
```

---

eth0はインターフェイス名です。モジュールやBSPのバージョンによって異なりますのでifconfigコマンドで調べてください。

固定IPの場合

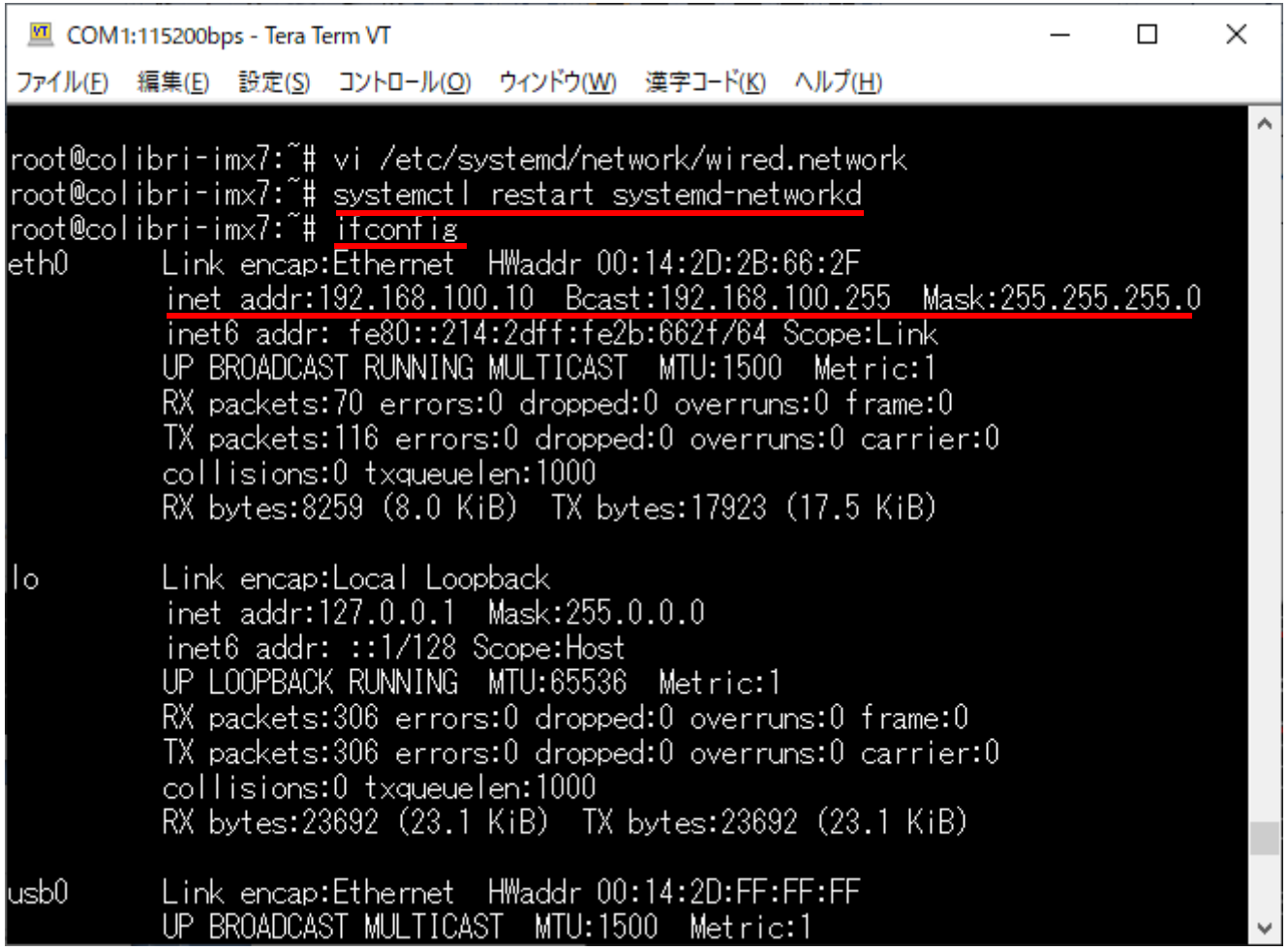
```
[Match]
Name=eth0

[Network]
Address=192.168.100.10/24
Gateway=192.168.100.254
```

192.168.100.10/24はIPアドレス192.168.100.10 サブネットマスク255.255.255.0を意味します。

下記コマンドでネットワークマネージャーを再起動します。
[colibri-imx7]# systemctl restart systemd-networkd

ifconfigで設定が反映されているのを確かめます。
[colibri-imx7]# ifconfig



SSH接続確認

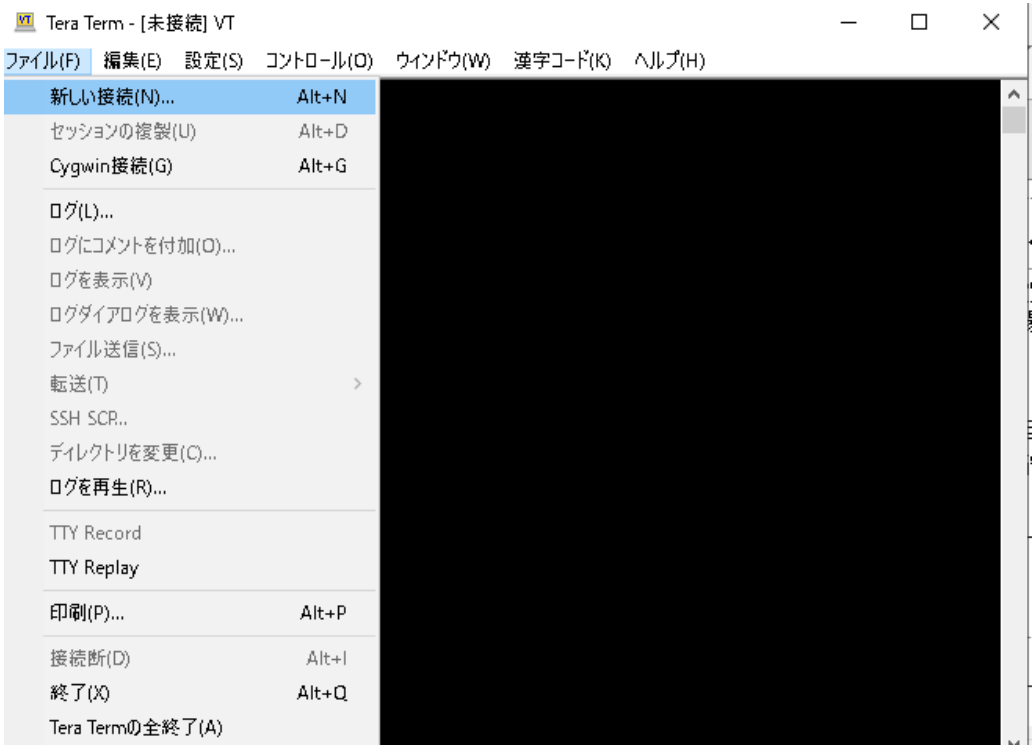
開発パソコンから接続できているか確認します。VMwareの設定がデフォルトのNATになっている場合、ホストOSのWindows10で接続できていればUbuntu側で接続できます。接続できない場合は何かしらの設定が間違っている可能性があります。

[ubuntu]$ ping 192.168.100.10

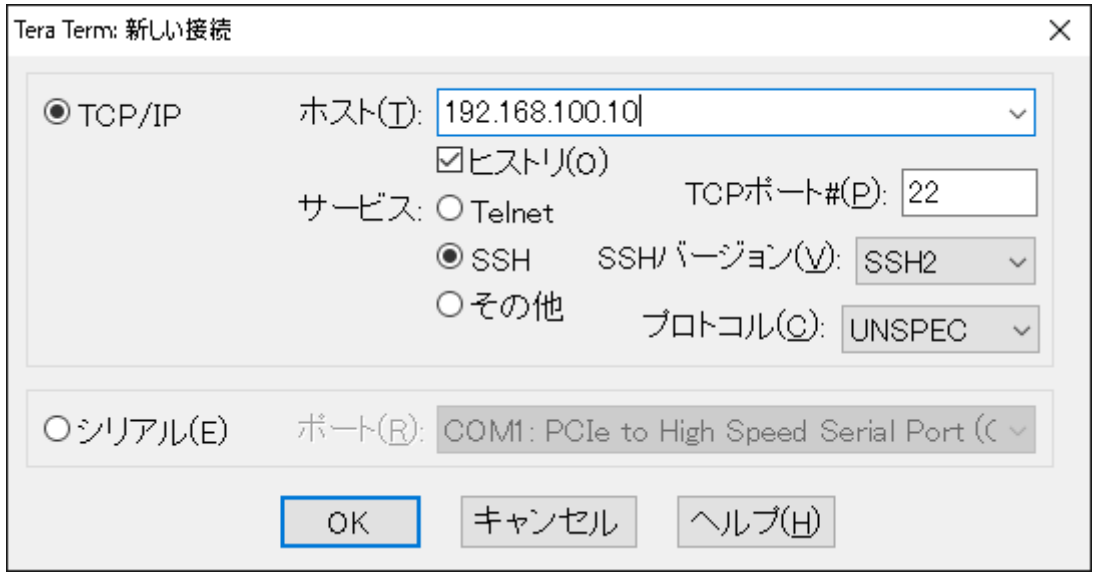
デフォルト設定のOSイメージではSSHサーバーが起動しています。

Teratermの新しい接続でSSH接続をして接続できるかどうか確かめてみます。

Teratermのメニューからファイル > 新しい接続を選択します。

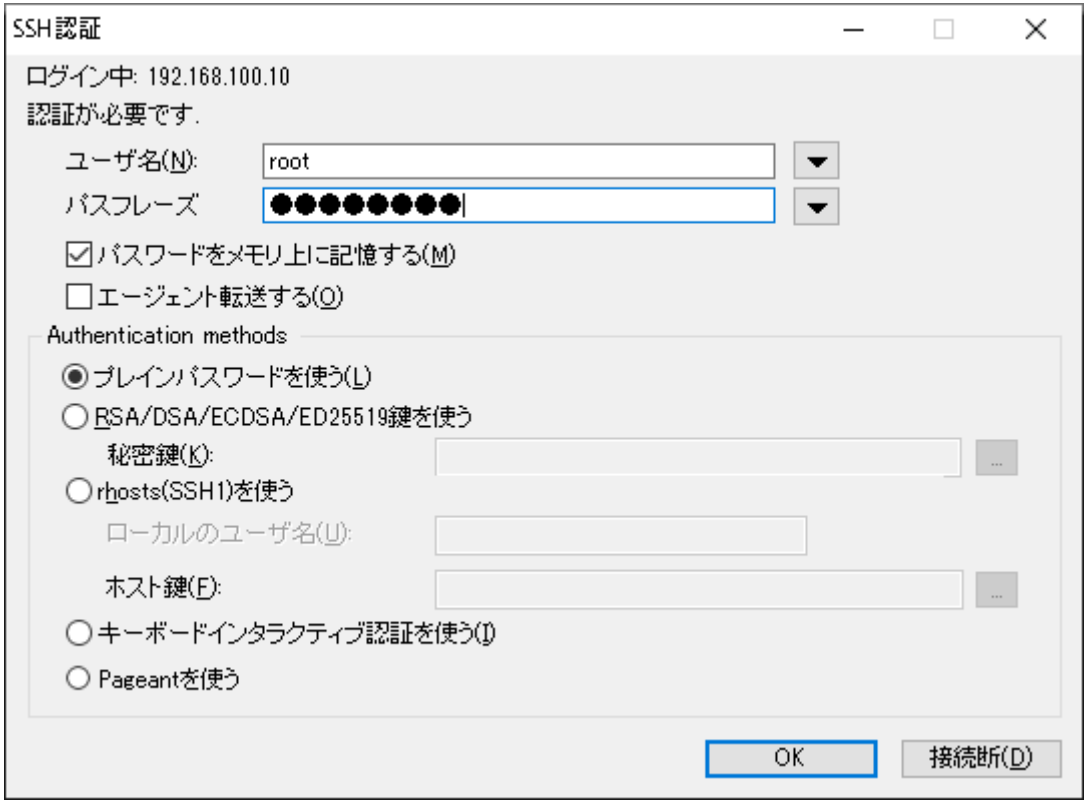


ホスト(T)にはモジュールに設定したIPを入力、サービスはSSHを選択してOKをクリックします。

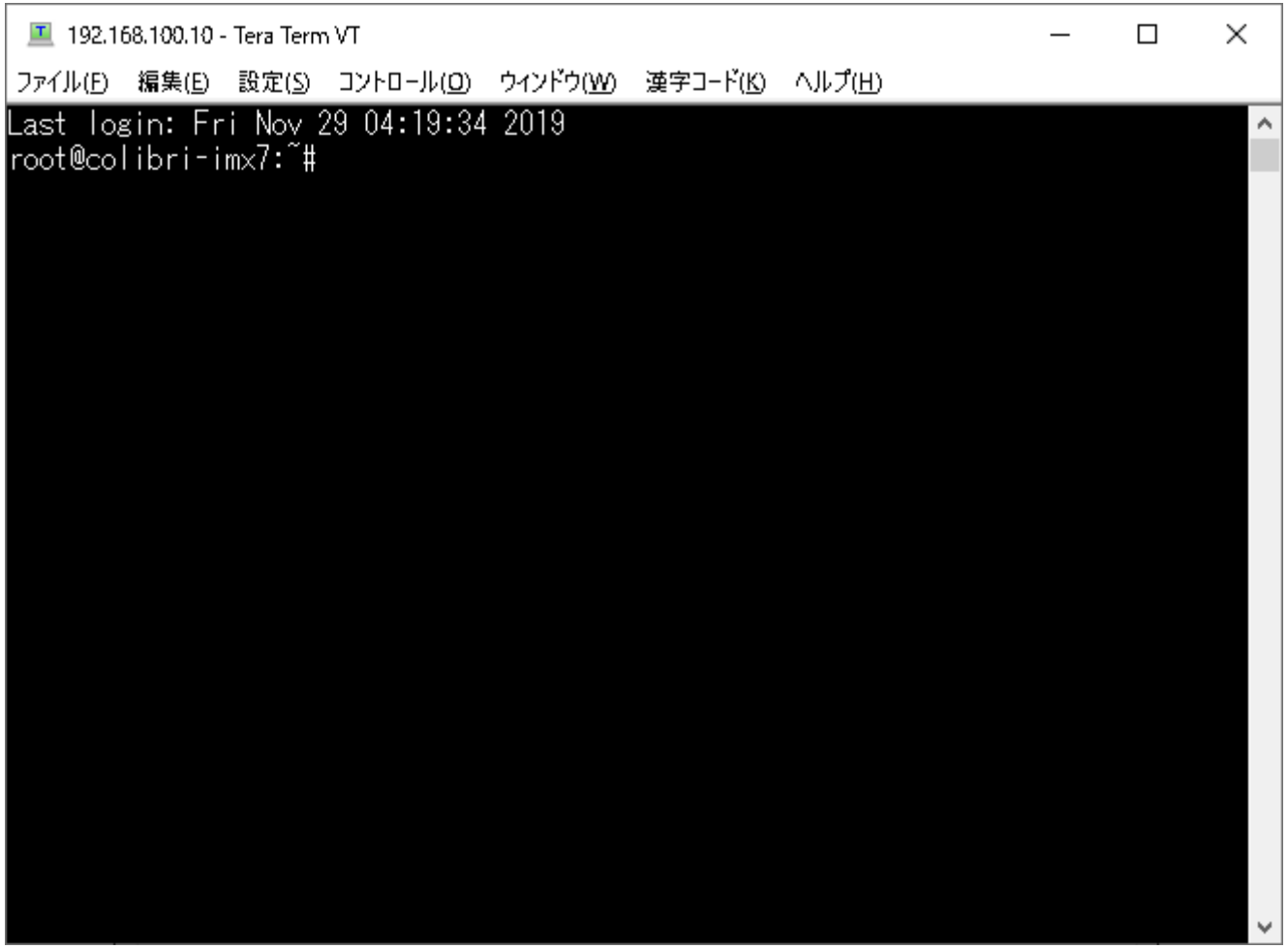


下記のような警告が出てきます。意図する接続先に間違いありませんので続行をクリックします。

ユーザ名はroot、パスフレーズはpasswdコマンドで設定したパスワードを入力します。
OKをクリックして接続します。

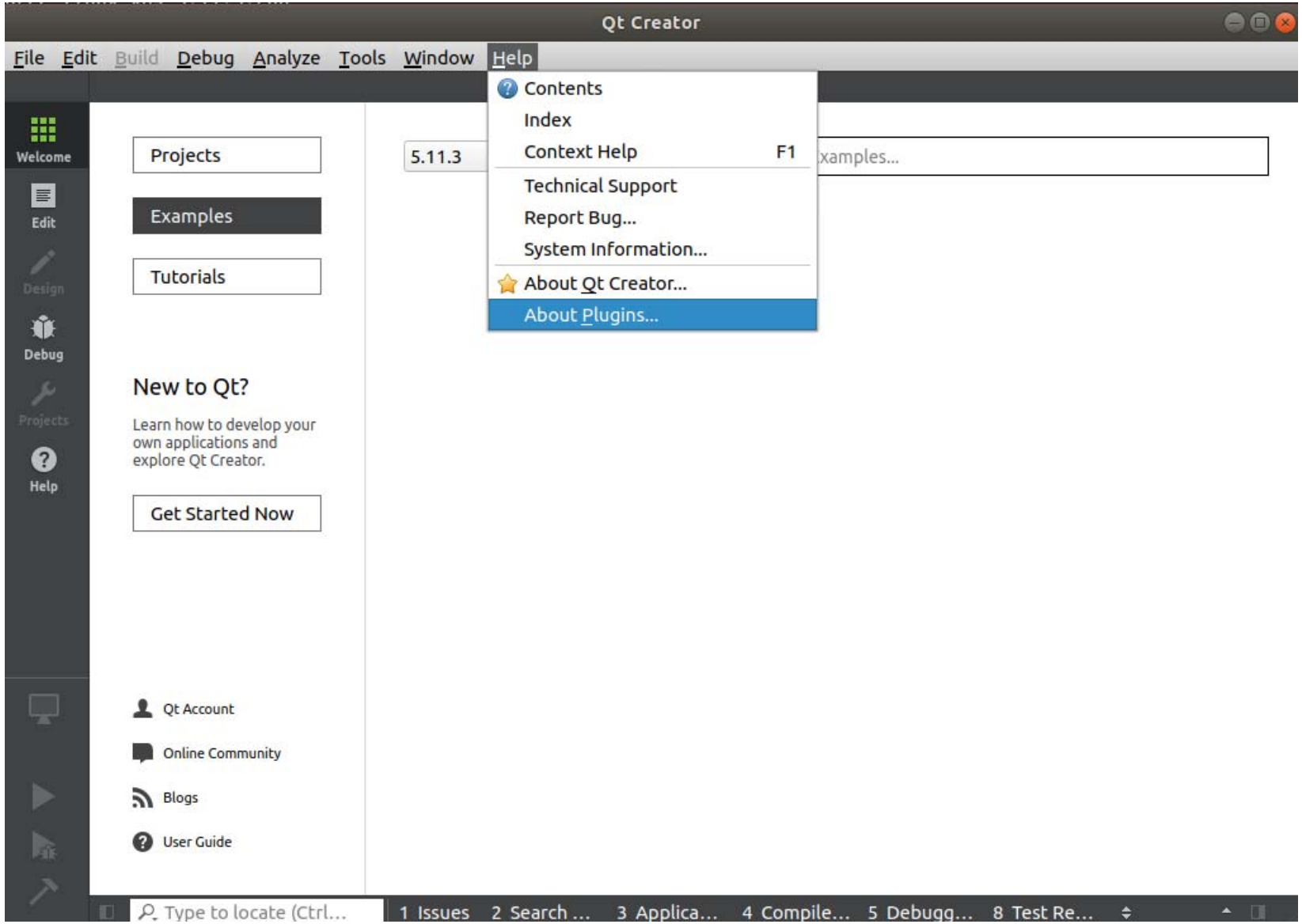


接続できた場合は下記のような画面になります。接続確認ができたらTeratermを終了します。

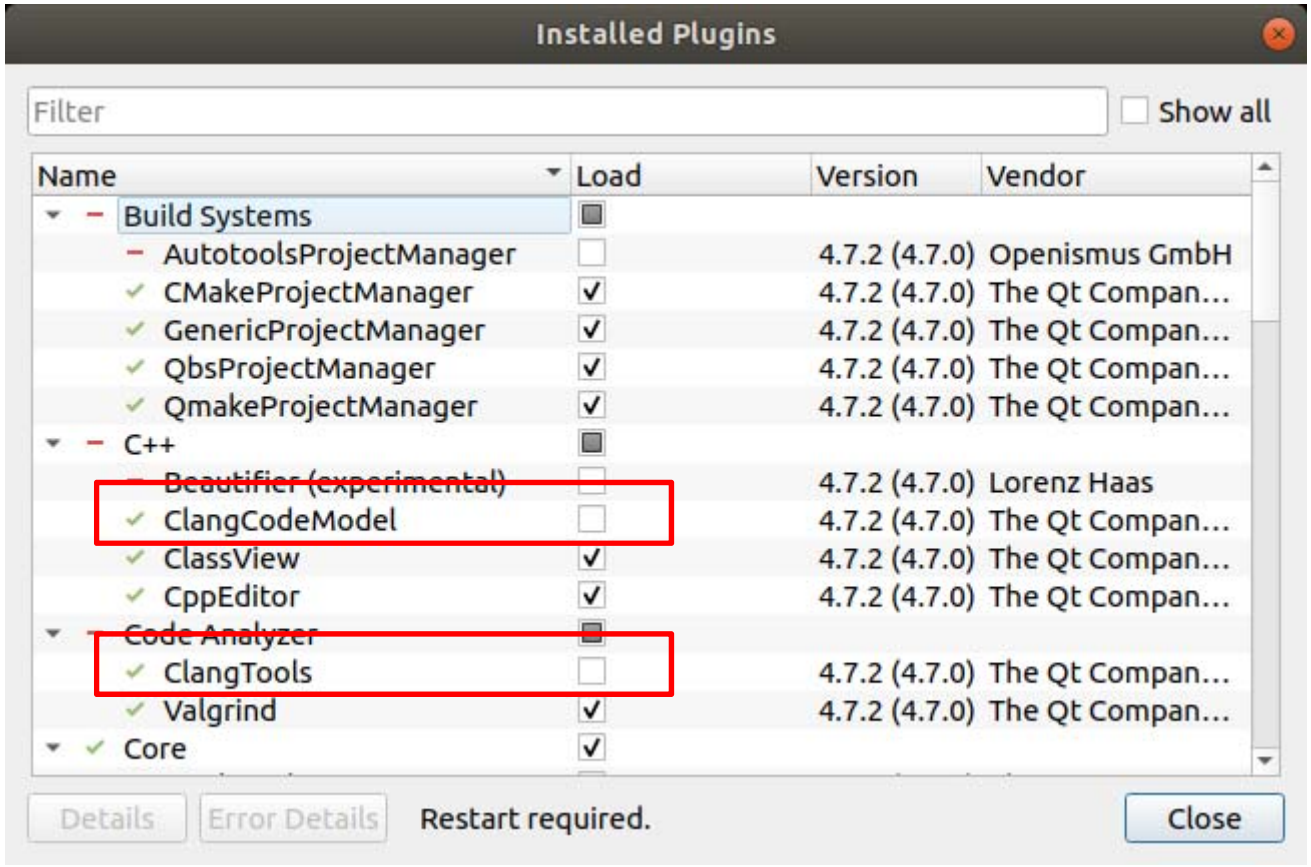


QT Creatorの設定

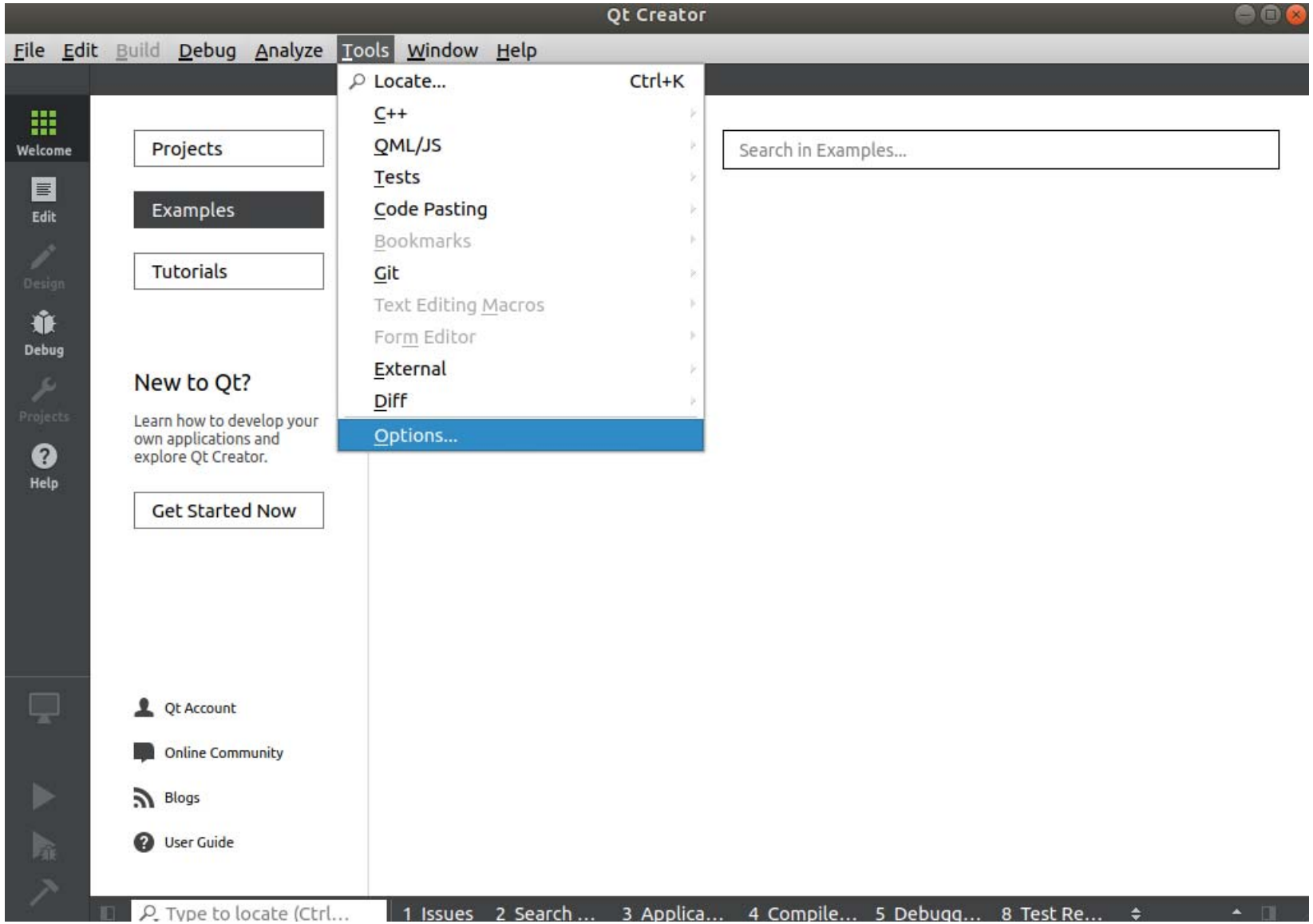
最初にClangのプラグインをoffにします。Help > About Pluginsを開きます。



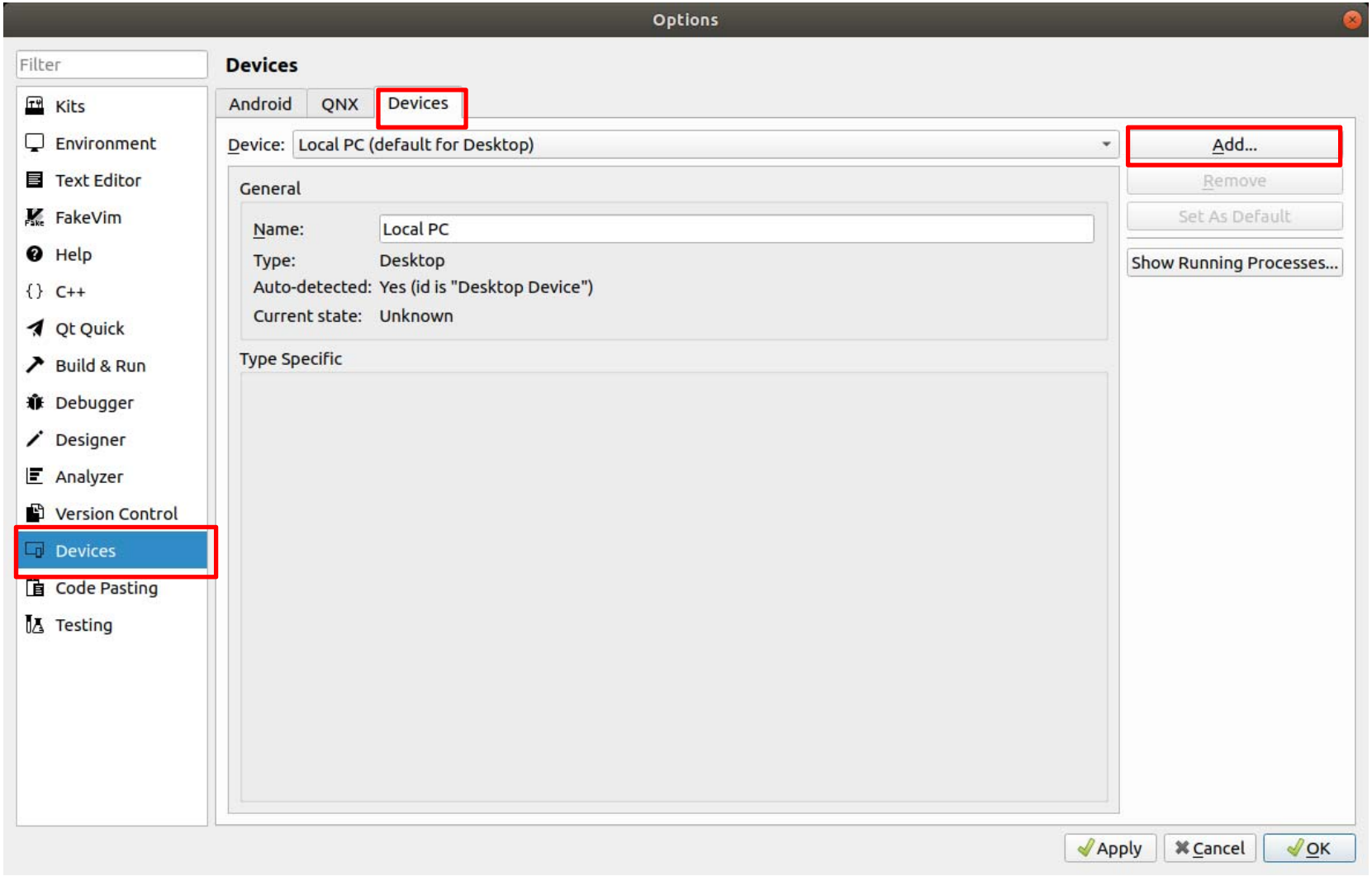
ClangCodeModelとClangToolsのチェックを外してCloseをクリックします。
その後、設定を反映するためにQT Creatorを一度再起動してください。
これを外さないとコードアナライザーが誤動作します。



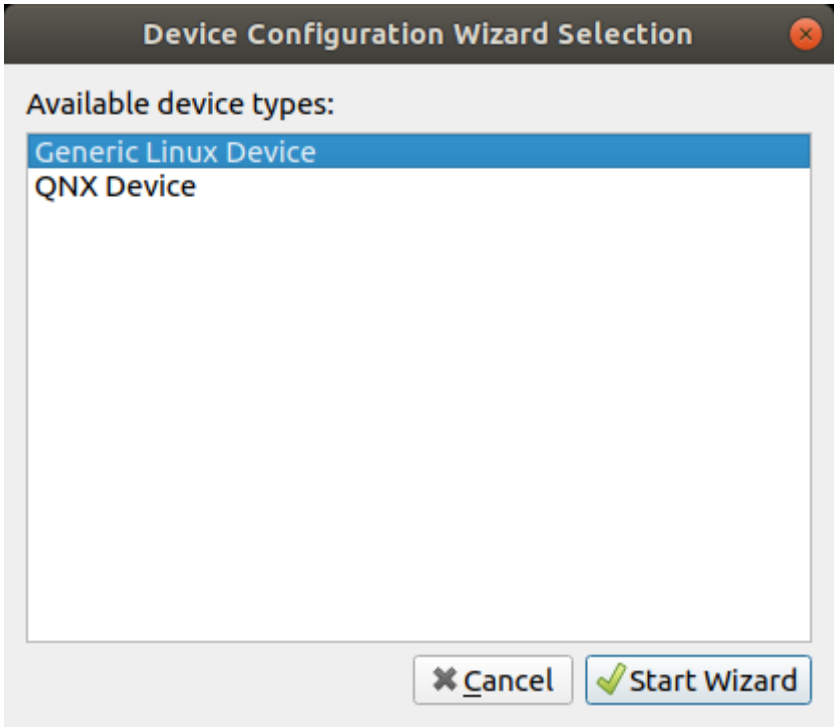
QT Creatorの設定を行います。Tools > Optionsを開きます。



左の欄からDevicesを選択しDevicesタブを選択します。
Addをクリックします。



Generic Linux Deviceを選択してStart Wizardをクリックします。



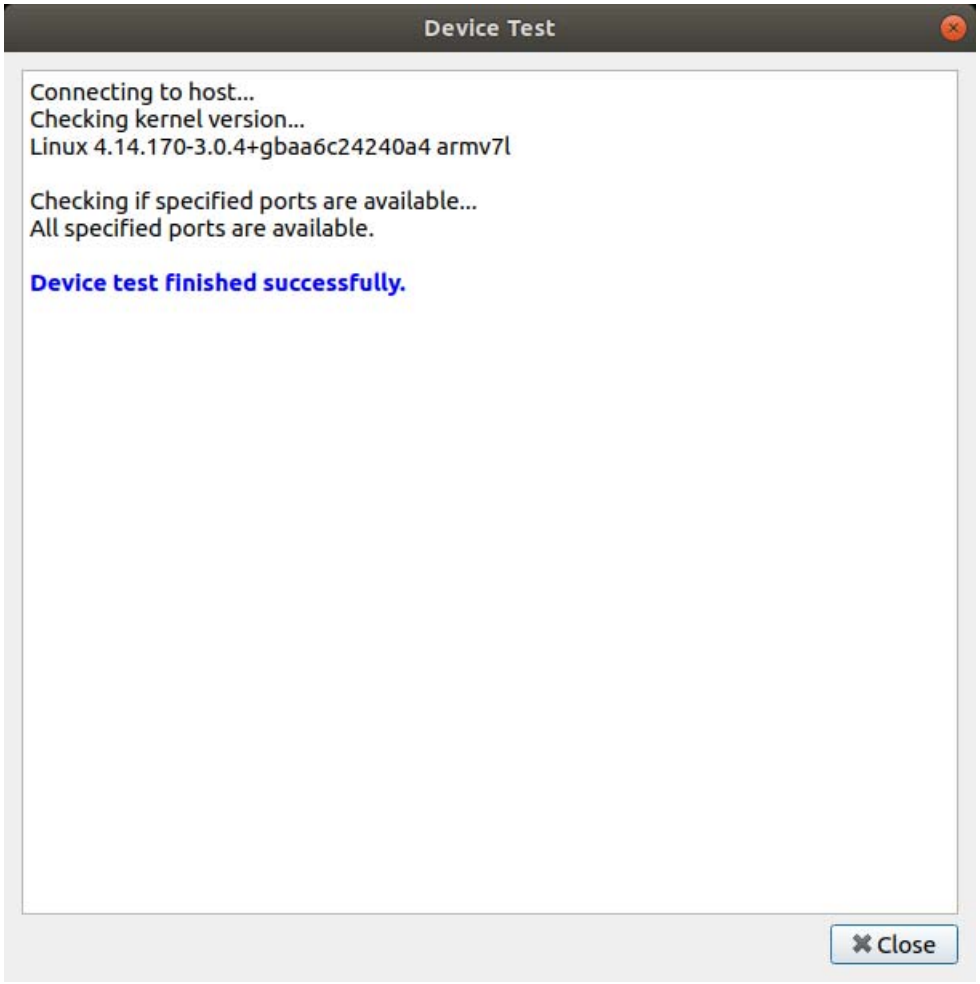
SSHで接続するモジュールの情報を入力します。本マニュアルでは下記の内容を設定しています。
接続名(わかりやすい名前をつけてください。):Colibri-iMX7
モジュールに設定したIPアドレス:192.168.100.10
ログインユーザー名:root
認証タイプ:パスワード
ログインパスワード:passwdコマンドで設定したパスワード

入力したらNextをクリックします。

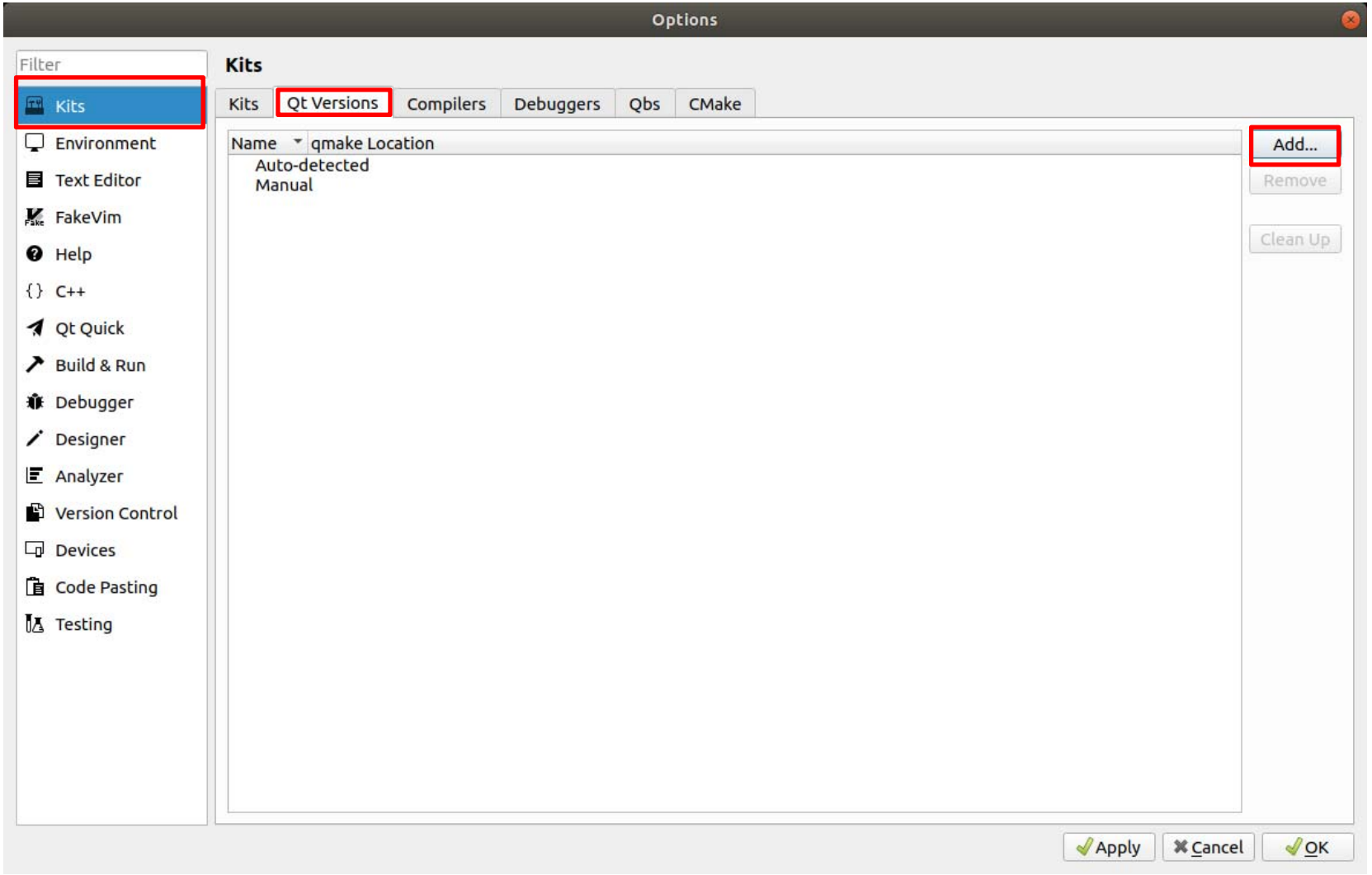
Finishをクリックします。

SSHの接続確認が行われます。成功すると下記のような画面になります。Closeをクリックします。

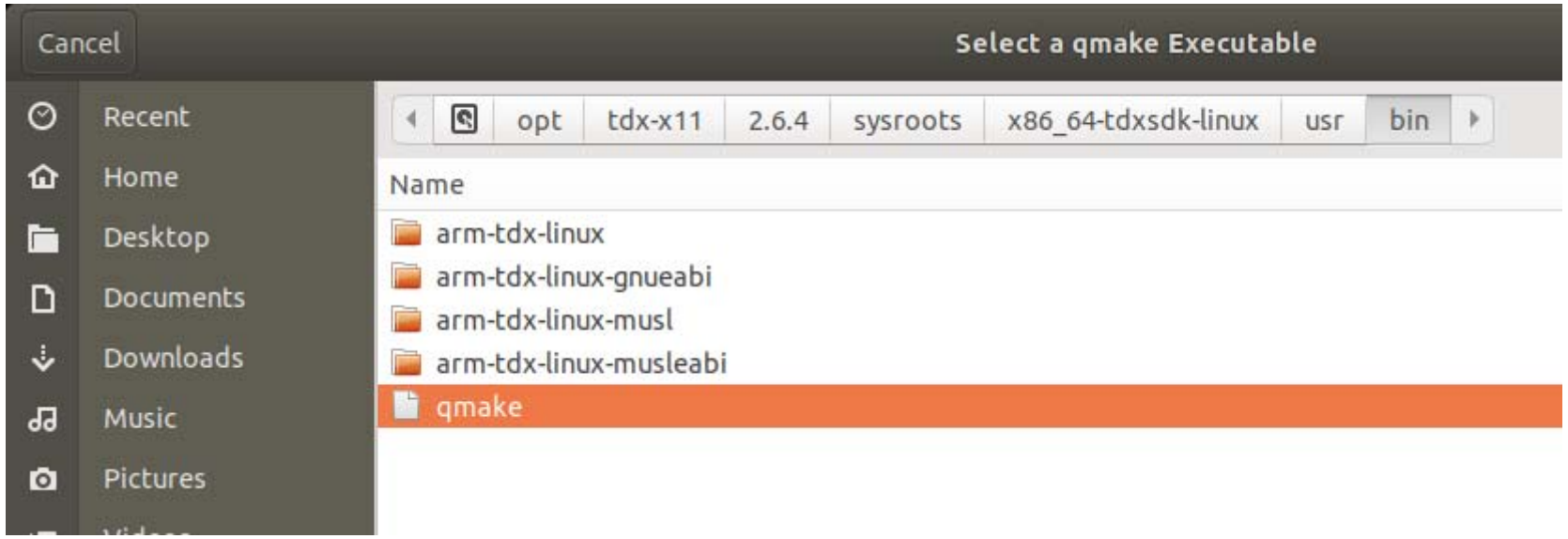
SSHの接続を行うと~/.ssh/known_hostsにホストのキーを保存します。
ホストのキーが保存された後にモジュールのOSを書き換えた場合、再度接続するとホストのキーが違うというエラーが発生します。
その場合ssh-keygen -f "~/.ssh/known_hosts" -R "192.168.100.10"で一旦古いキーを削除してください。



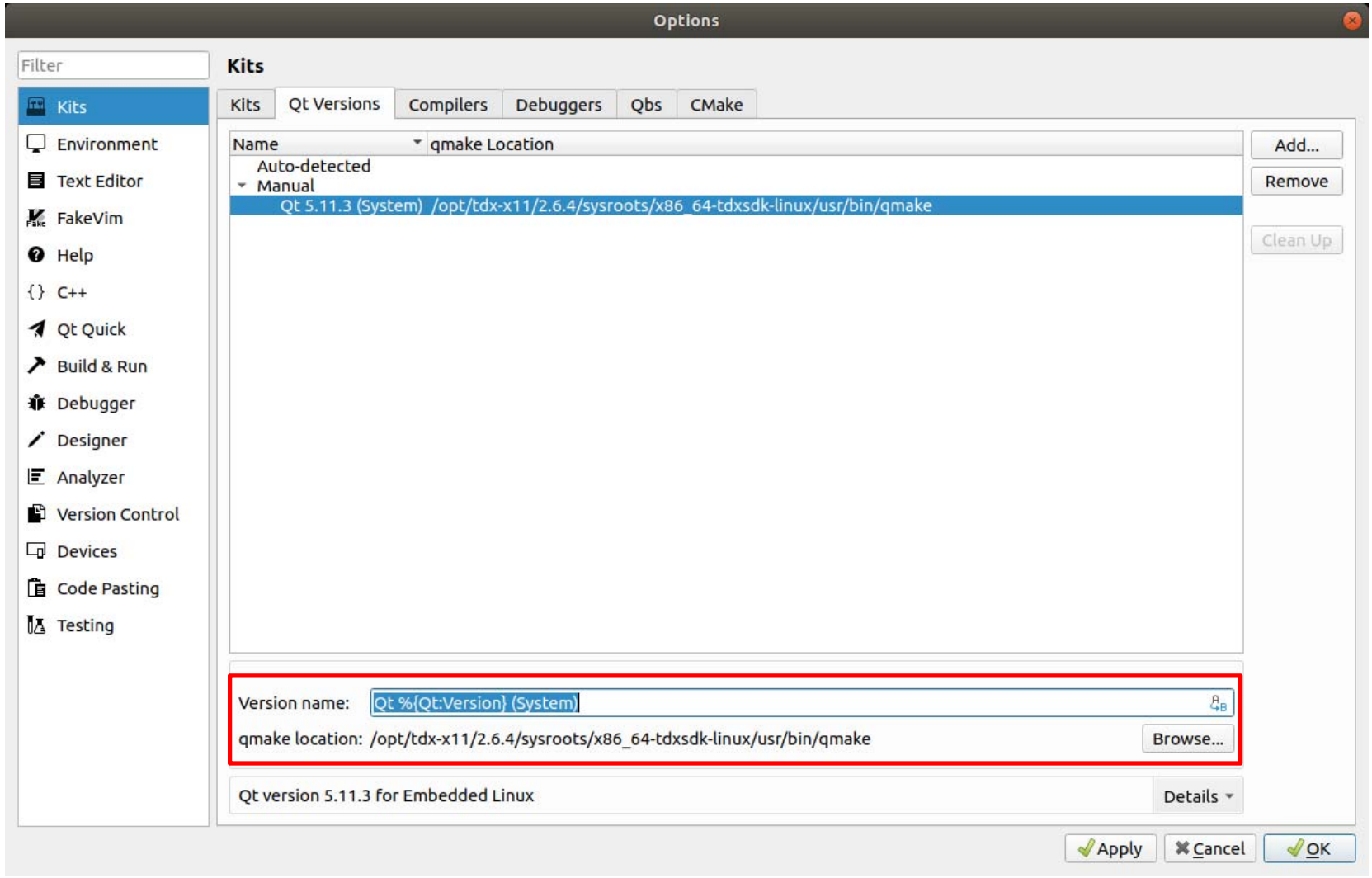
Optionsに戻ります。左の欄からKitsを選択しQT Versionsタブを選択します。Addをクリックします。



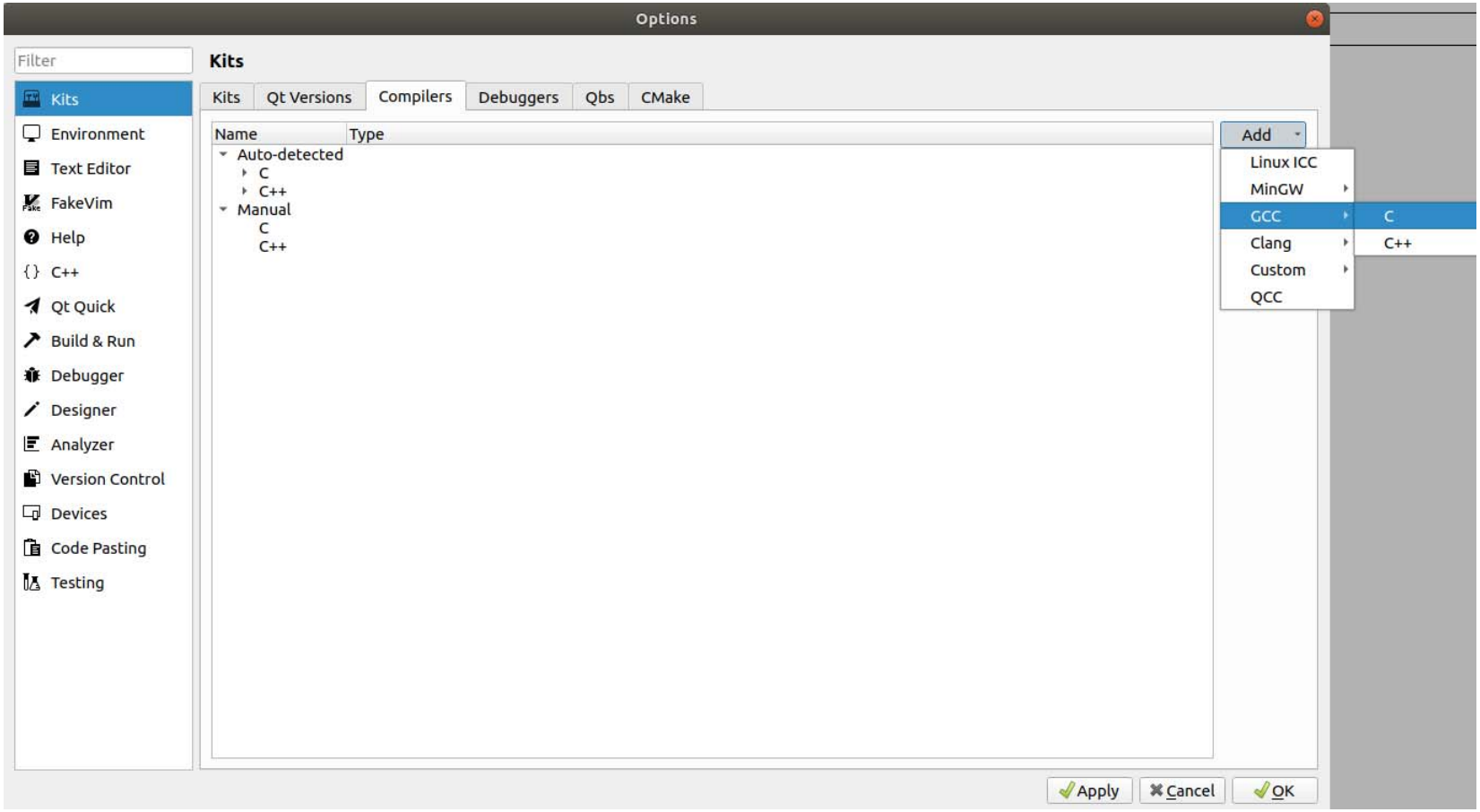
qmakeのパスを選択します。選択したら画面右上のOpenボタンをクリックします。
(パスはSDK出力先やモジュールやBSPのバージョンなどによって異なります。)
デフォルト設定の場合は下記になります。
/opt/tdx-x11/2.6.4/sysroots/x86_64-tdxsdk-linux/usr/bin/qmake



下記のようになります。
Version Nameは自動的に作成されます。QT5.11.3(System)という名前になります。

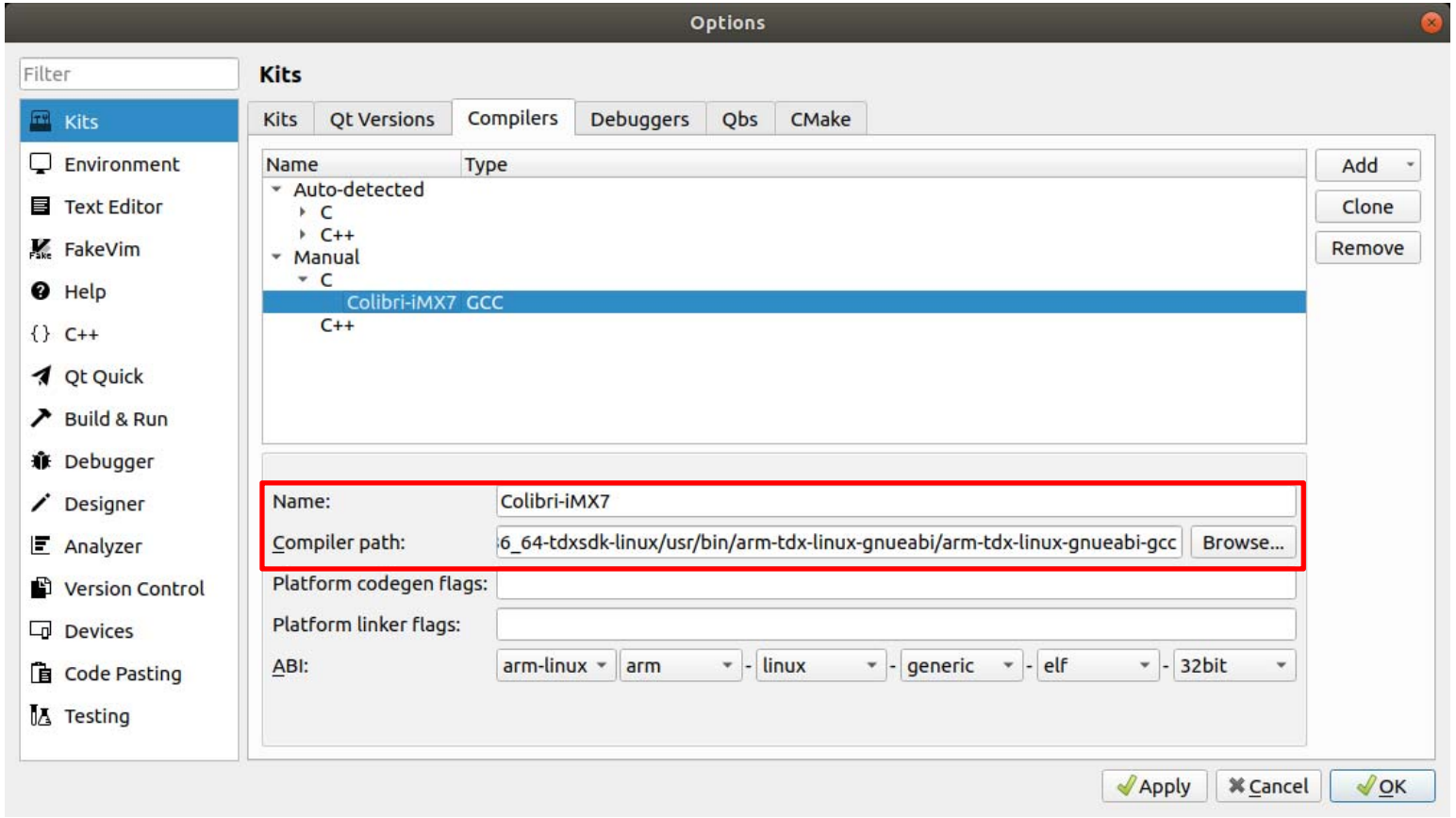


Compilersタブを選択しAddからGCC > Cを選択します。

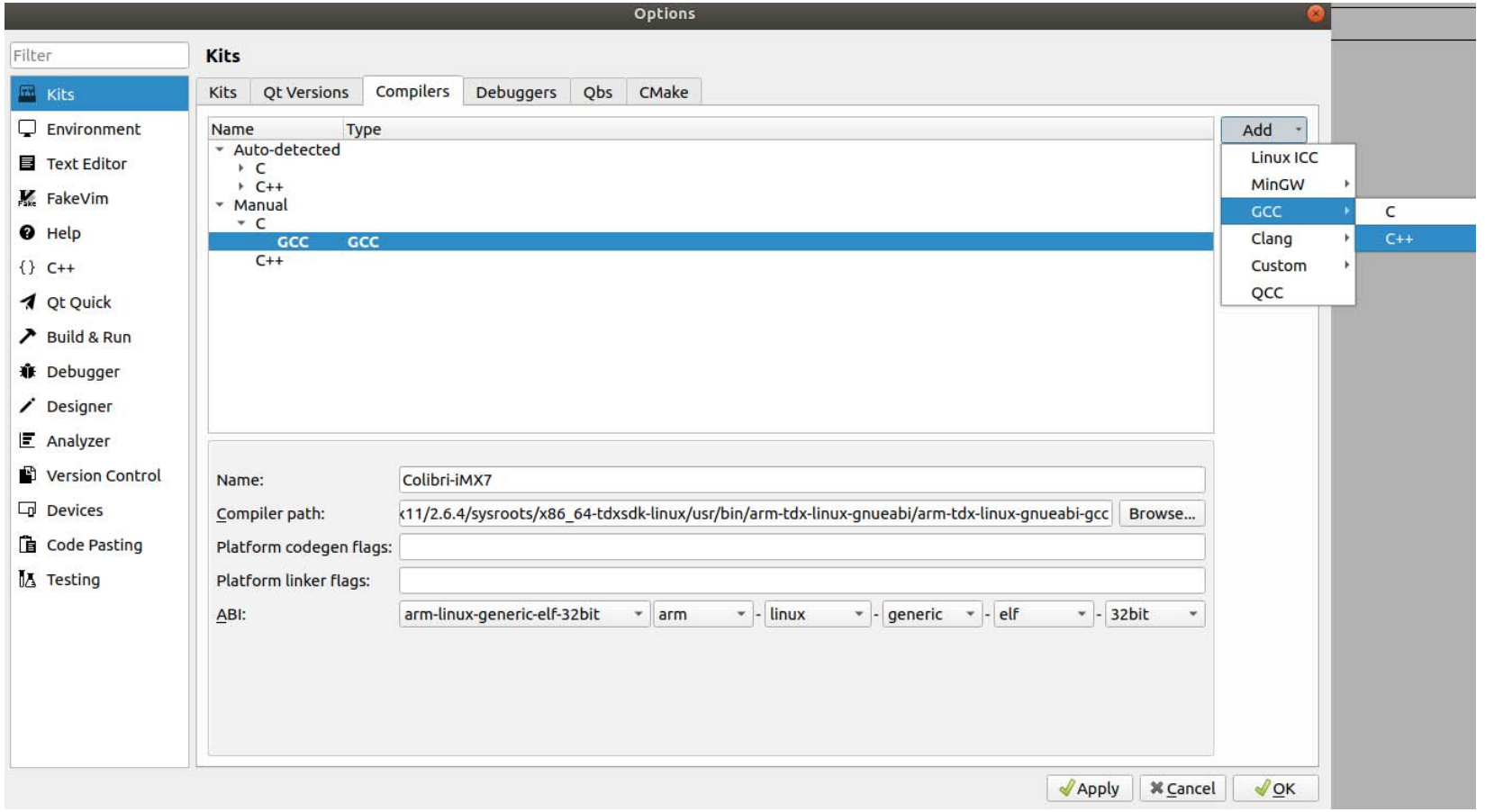


Name:にコンパイラの設定の名前を入力します。本マニュアルではColibri-iMX7としています。
Compiler Pathに下記を入力します。(パスはSDK出力先やモジュールやBSPのバージョンなどによって異なります。)
デフォルト設定の場合は下記になります。
/opt/tdx-x11/2.6.4/sysroots/x86_64-tdxsdk-linux/usr/bin/arm-tdx-linux-gnueabi/arm-tdx-linux-gnueabi-gcc

Applyをクリックます。

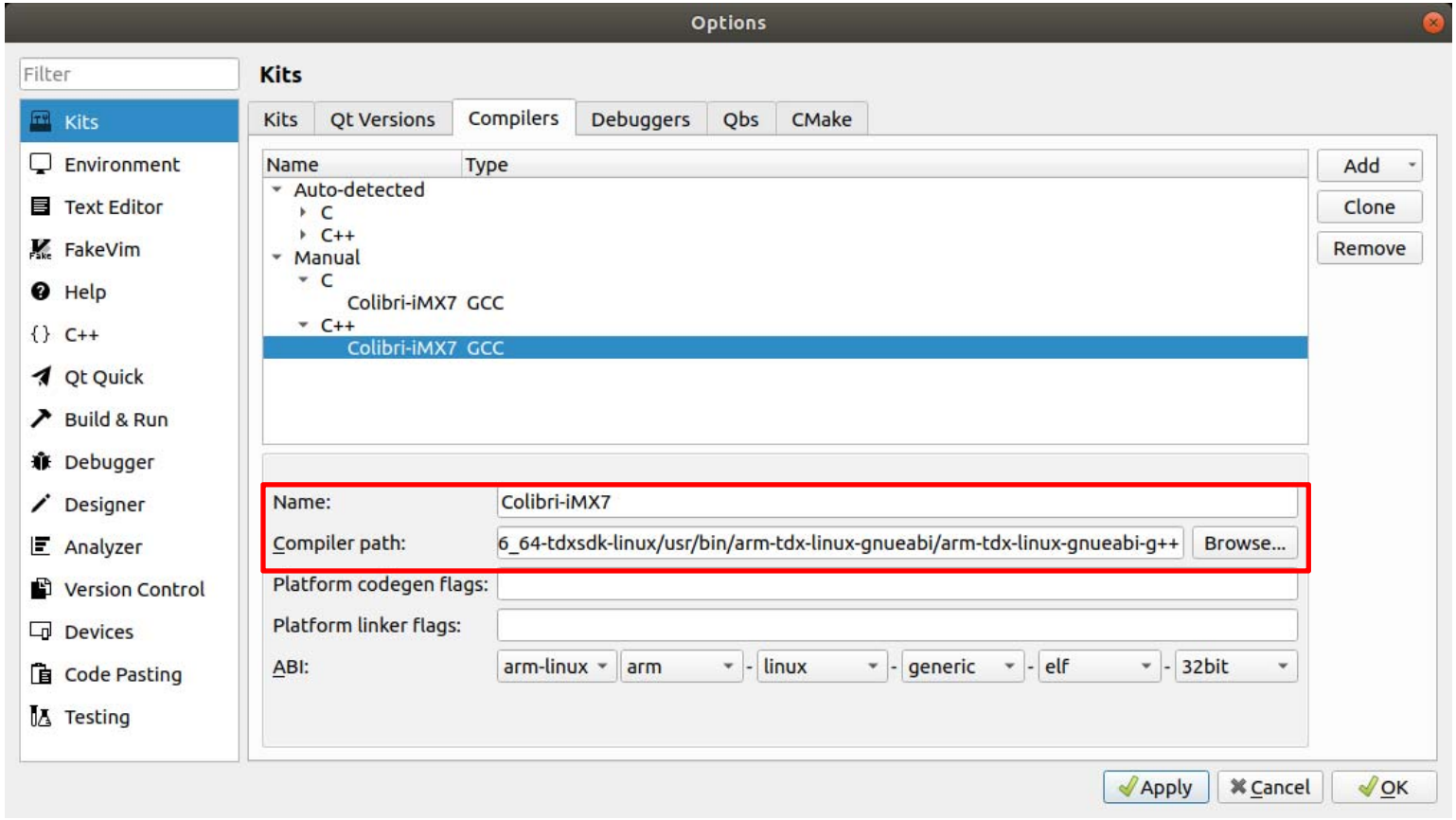


Compilersタブを選択しAddからGCC > C++を選択します。

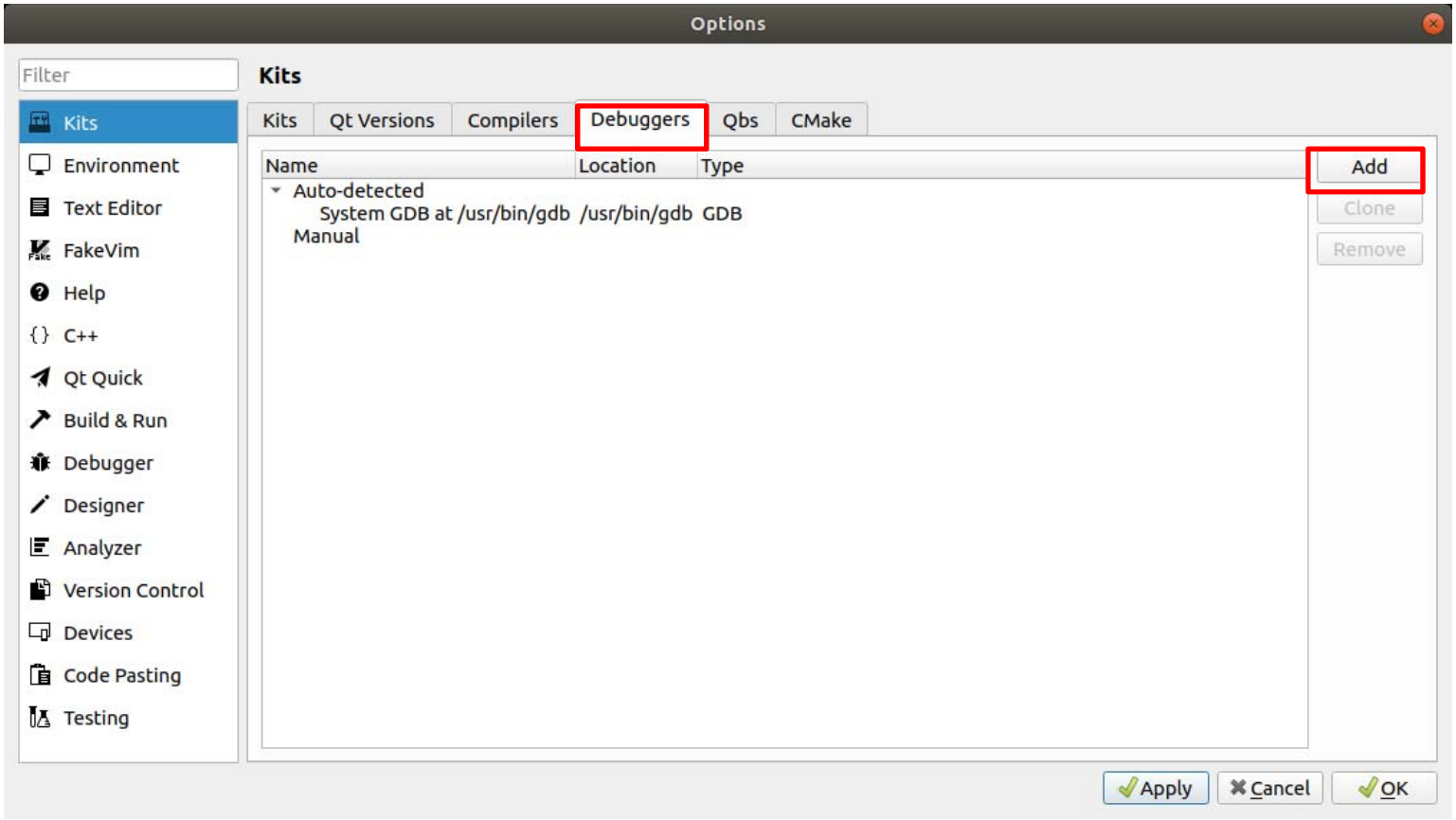


Nameにコンパイラの設定の名前を入力します。本マニュアルではColibri-iMX7としています。
Compiler Pathに下記を入力します。(パスはSDK出力先やモジュールやBSPのバージョンなどによって異なります。)
デフォルト設定の場合は下記になります。
/opt/tdx-x11/2.6.4/sysroots/x86_64-tdxsdk-linux/usr/bin/arm-tdx-linux-gnueabi/arm-tdx-linux-gnueabi-g++

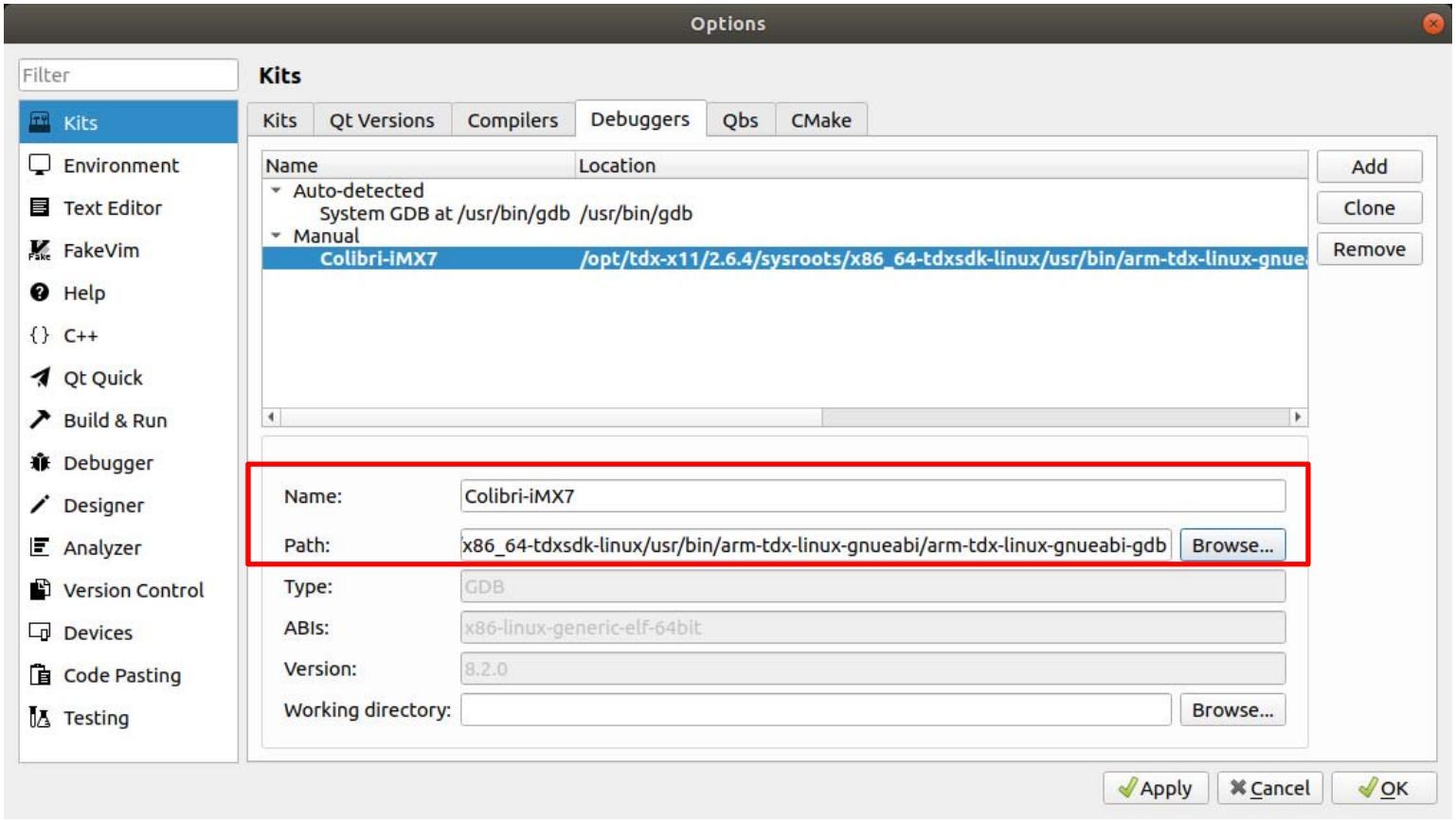
Applyをクリックます。



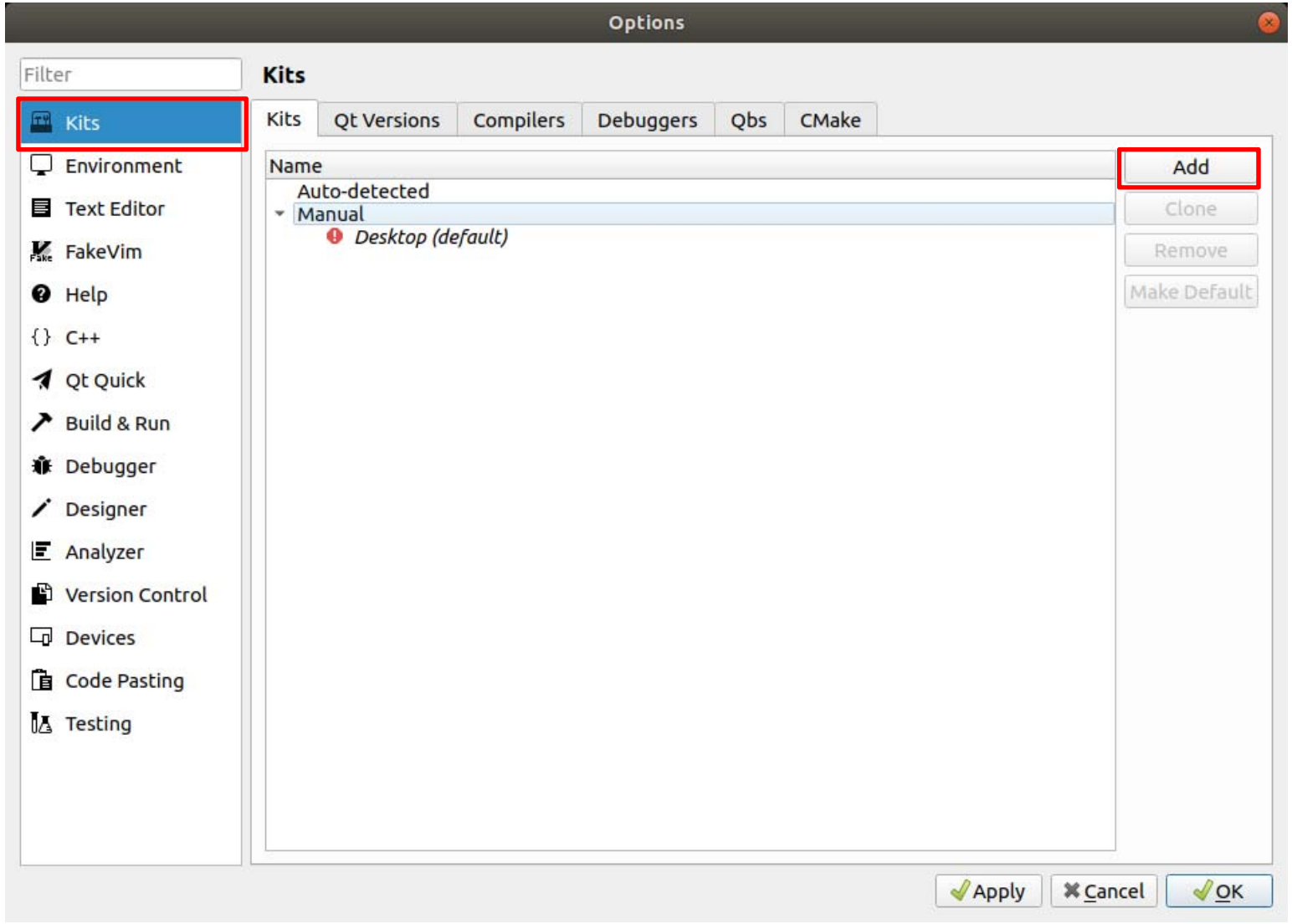
Debuggersタブを選択しAddをクリックします。



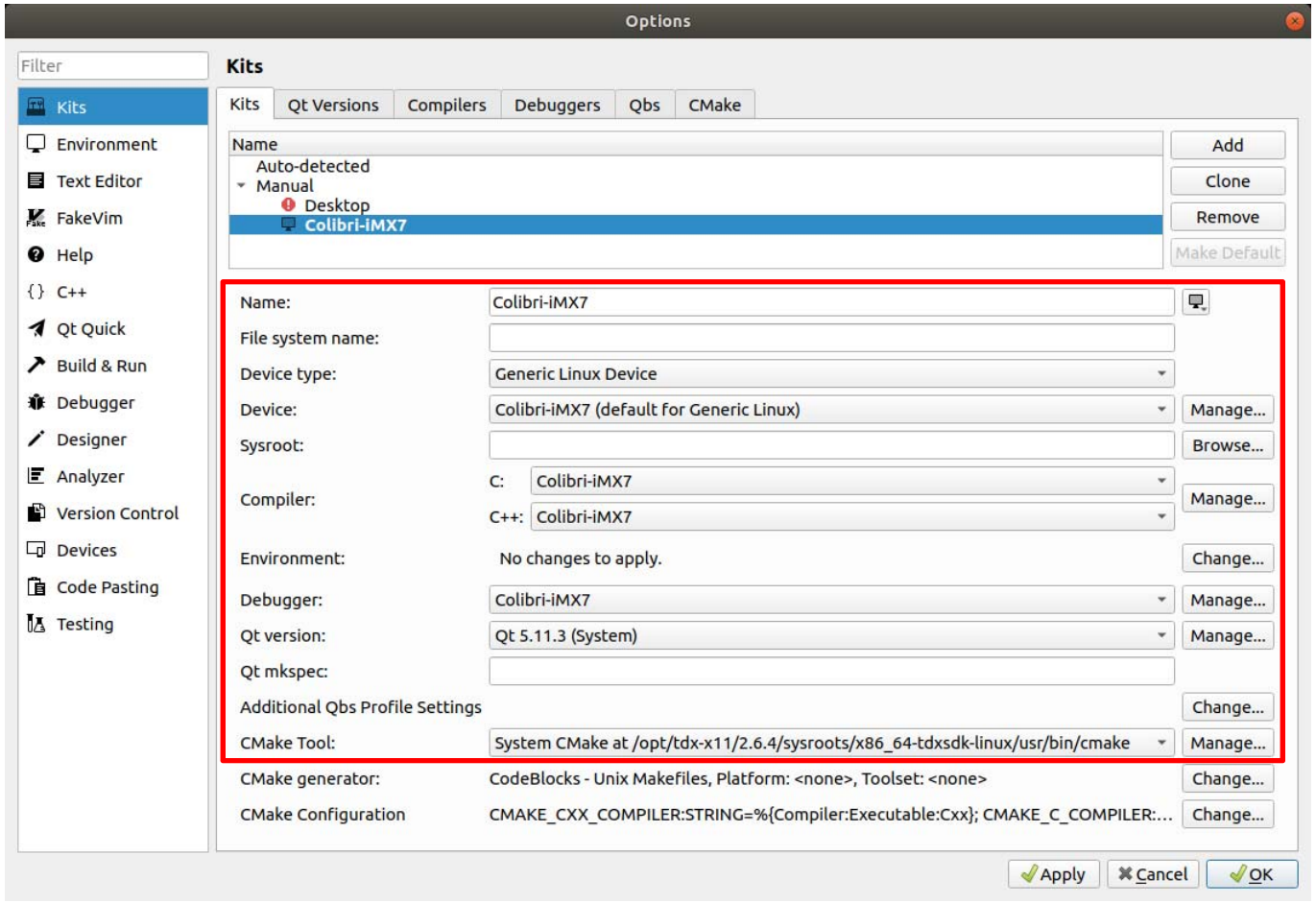
Nameにデバッガーの設定の名前を入力します。本マニュアルではColibri-iMX7としています。
Pathに下記を入力します。(パスはSDK出力先やモジュールやBSPのバージョンなどによって異なります。)
デフォルト設定の場合は下記になります。
/opt/tdx-x11/2.6.4/sysroots/x86_64-tdxsdk-linux/usr/bin/arm-tdx-linux-gnueabi/arm-tdx-linux-gnueabi-gdb



Kitsタブを選択しAddをクリックします。

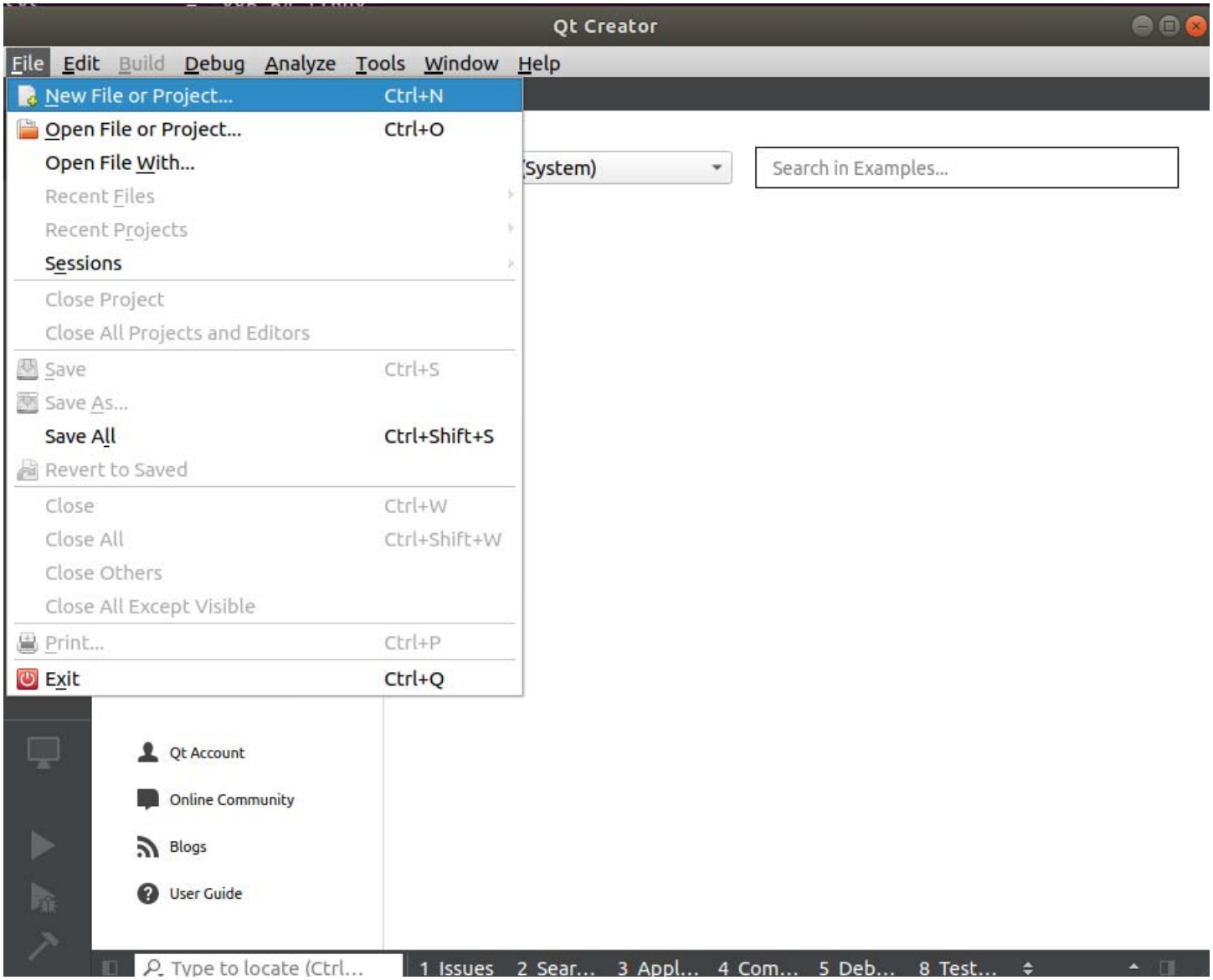


Nameにデバッガーの設定の名前を入力します。本マニュアルではColibri-iMX7としています。
Device type：Generic　Linux Device
Device,QT Version,Compilers,Debuggerに関してはそれぞれDevices,QT Version,Compilers,Debuggerで作成した設定を選択します。CMake Toolは自動で入力されます。
OKをクリックします。

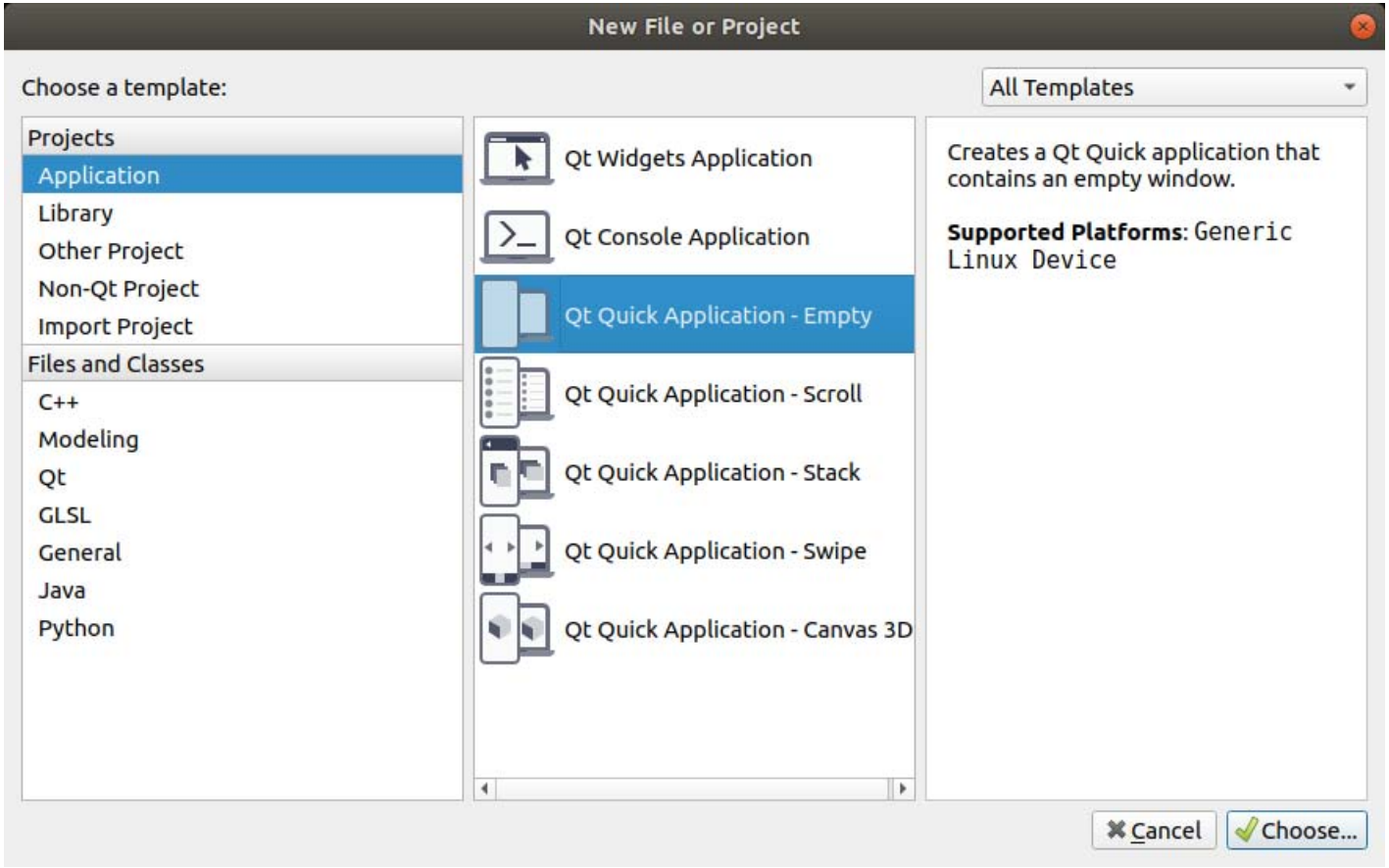


プロジェクト作成からデバッグまで

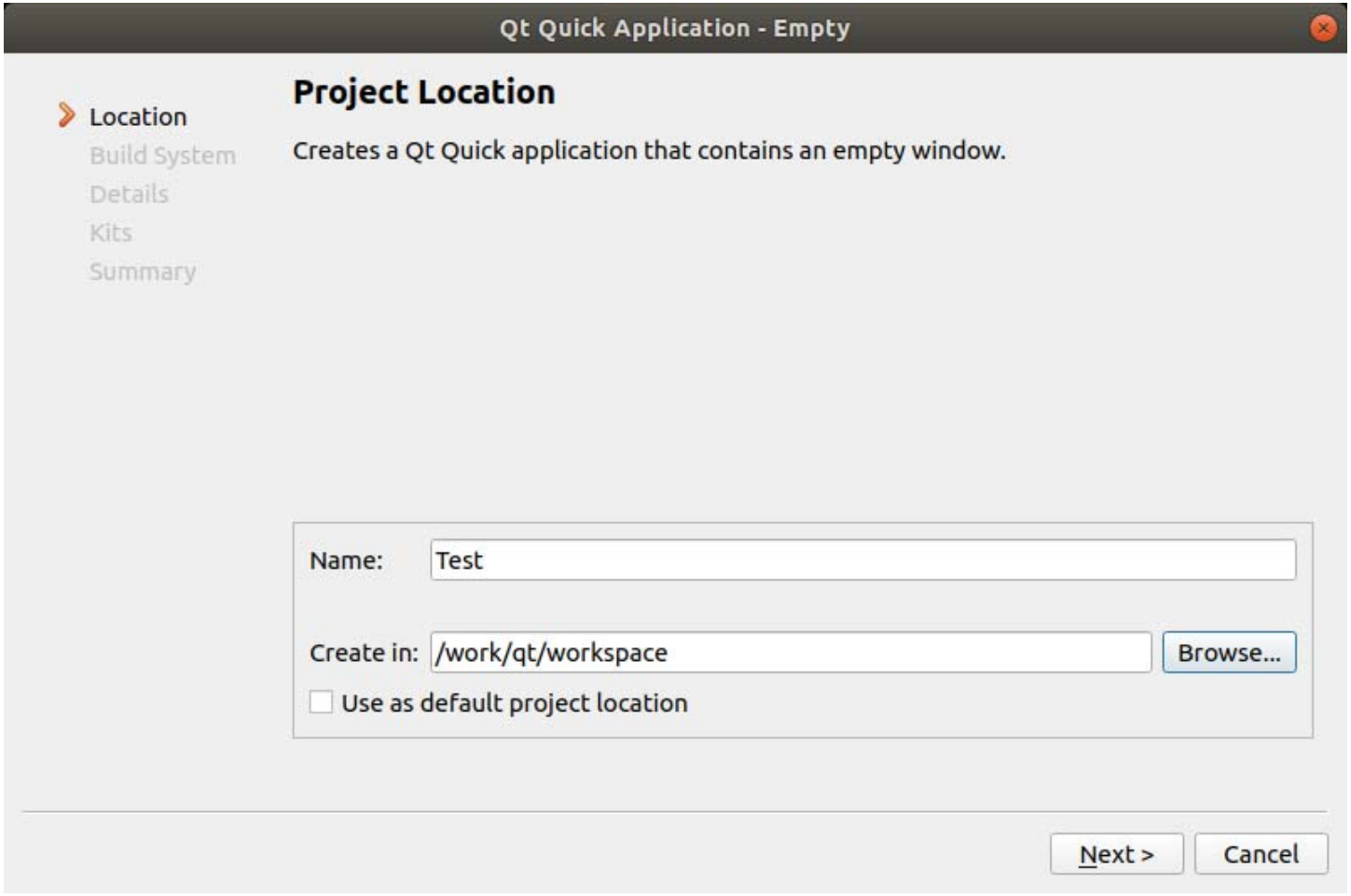
メニューからFile > New Fileor Projectを選択します。



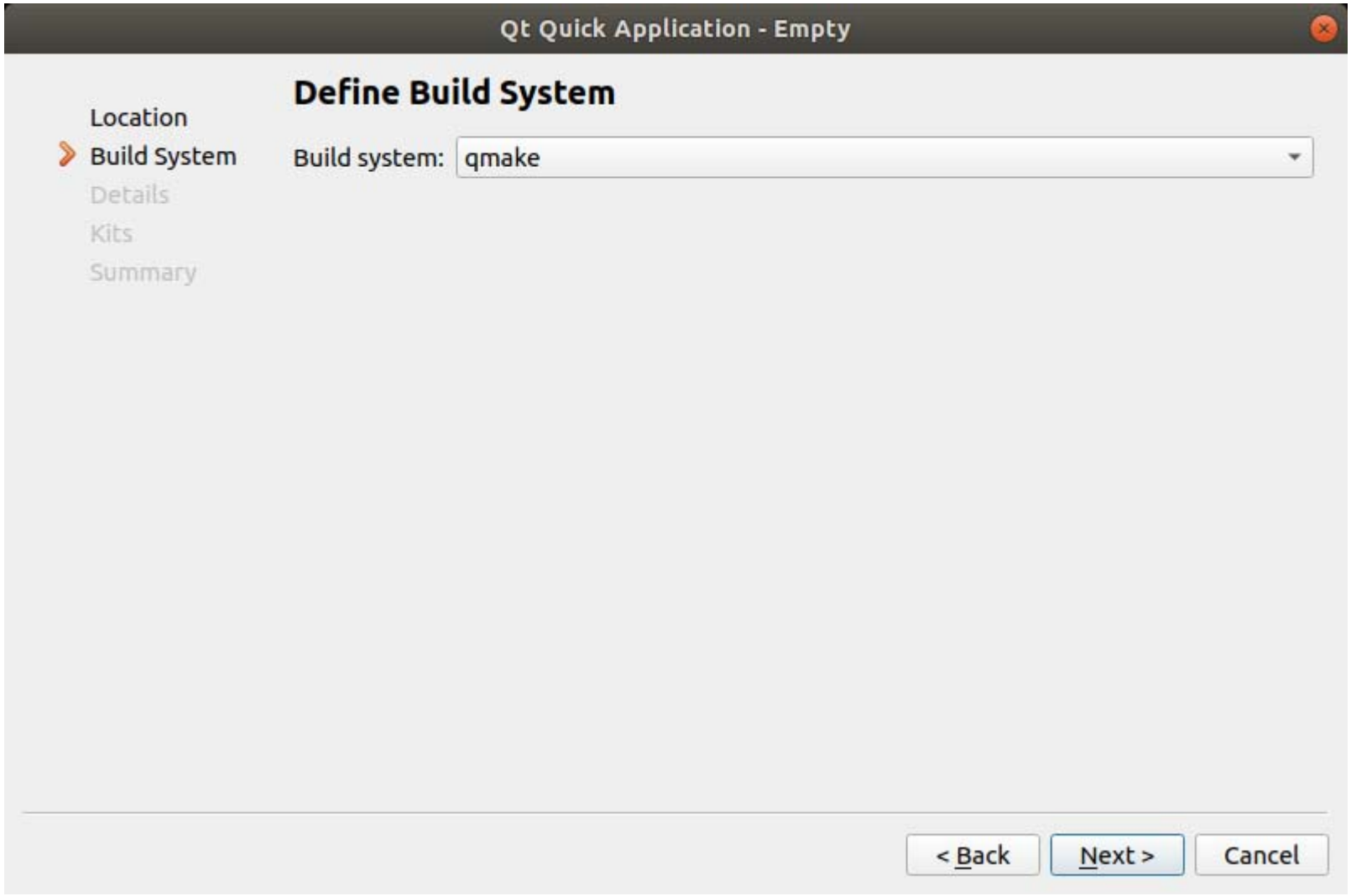
簡単なアプリケーションを作成します。
Projectsの項目はApplicationを選択します。
本マニュアルでQt Quick Application - Emptyを選択します。
Chooseをクリックします。



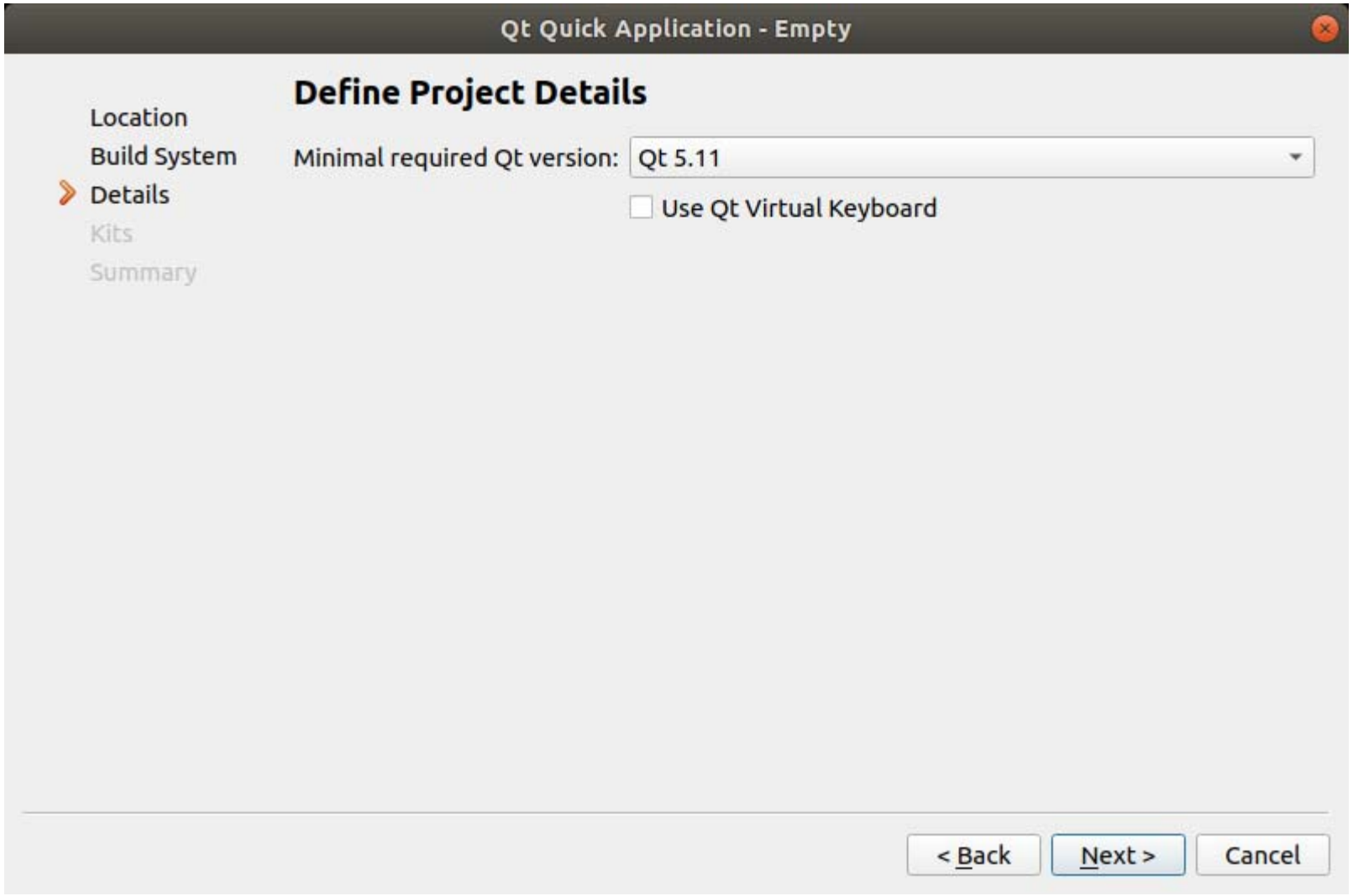
NameにProjectの名前を入力します。本マニュアルではTestとしています。
Projectのパスは事前に作成した$/work/qt/配下のworkspaceを入力します。
Nextをクリックします。



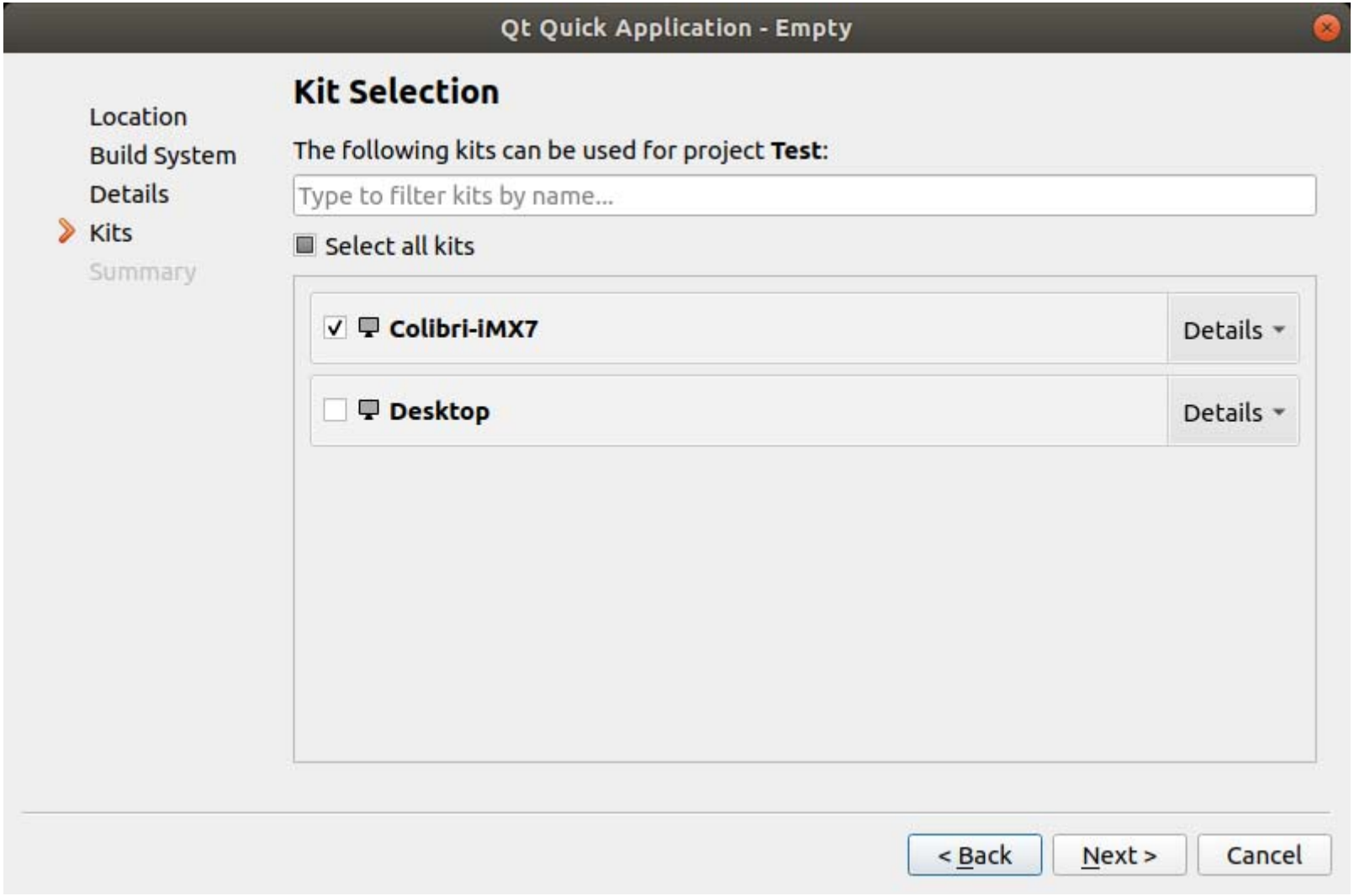
Nextをクリックします。



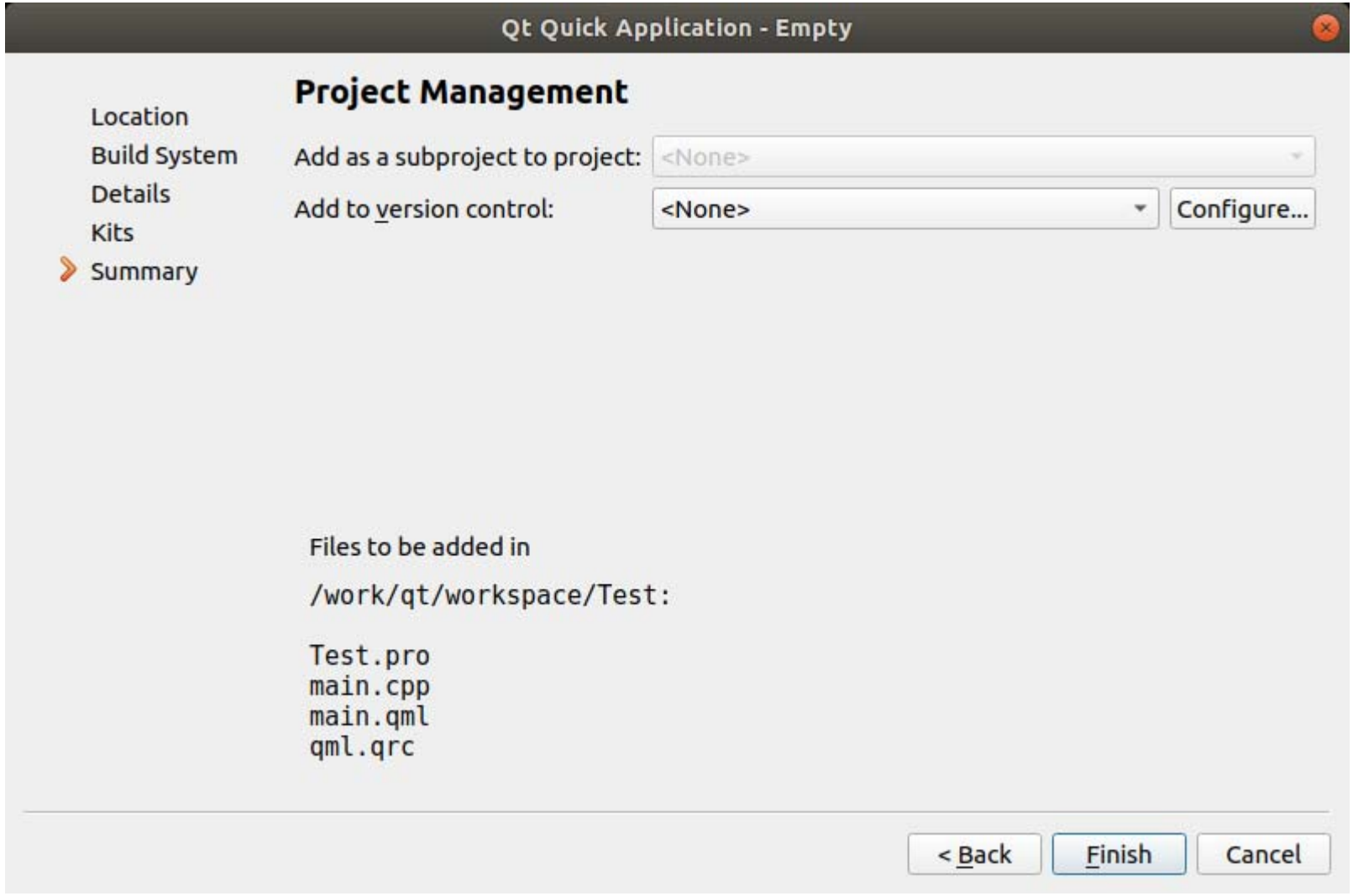
本プロジェクトが動作するQTの最低バージョンを指定します。本マニュアルでは5.11としています。
Nextをクリックします。



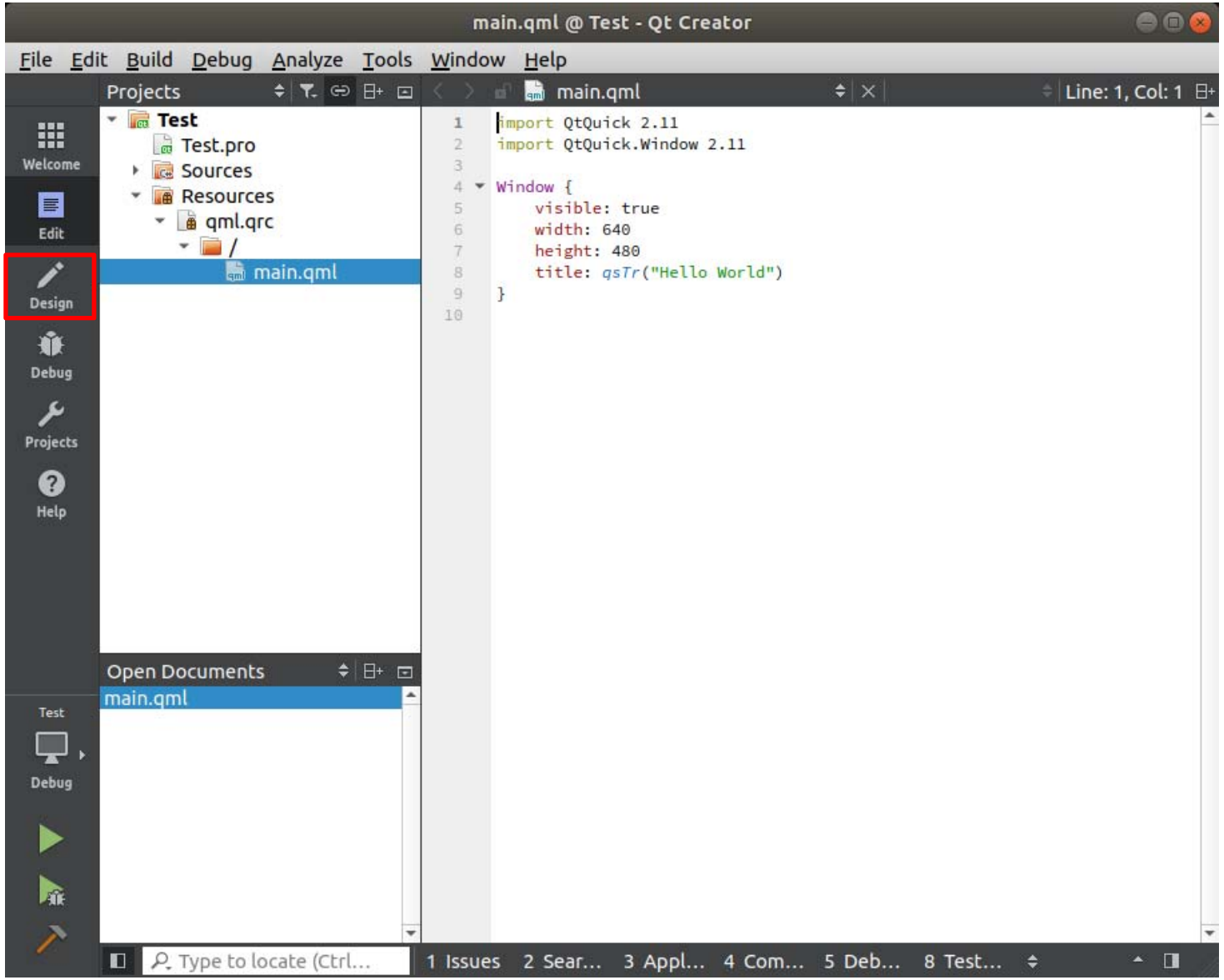
事前に作成したKitが選択されていることを確認します。
Nextをクリックします。



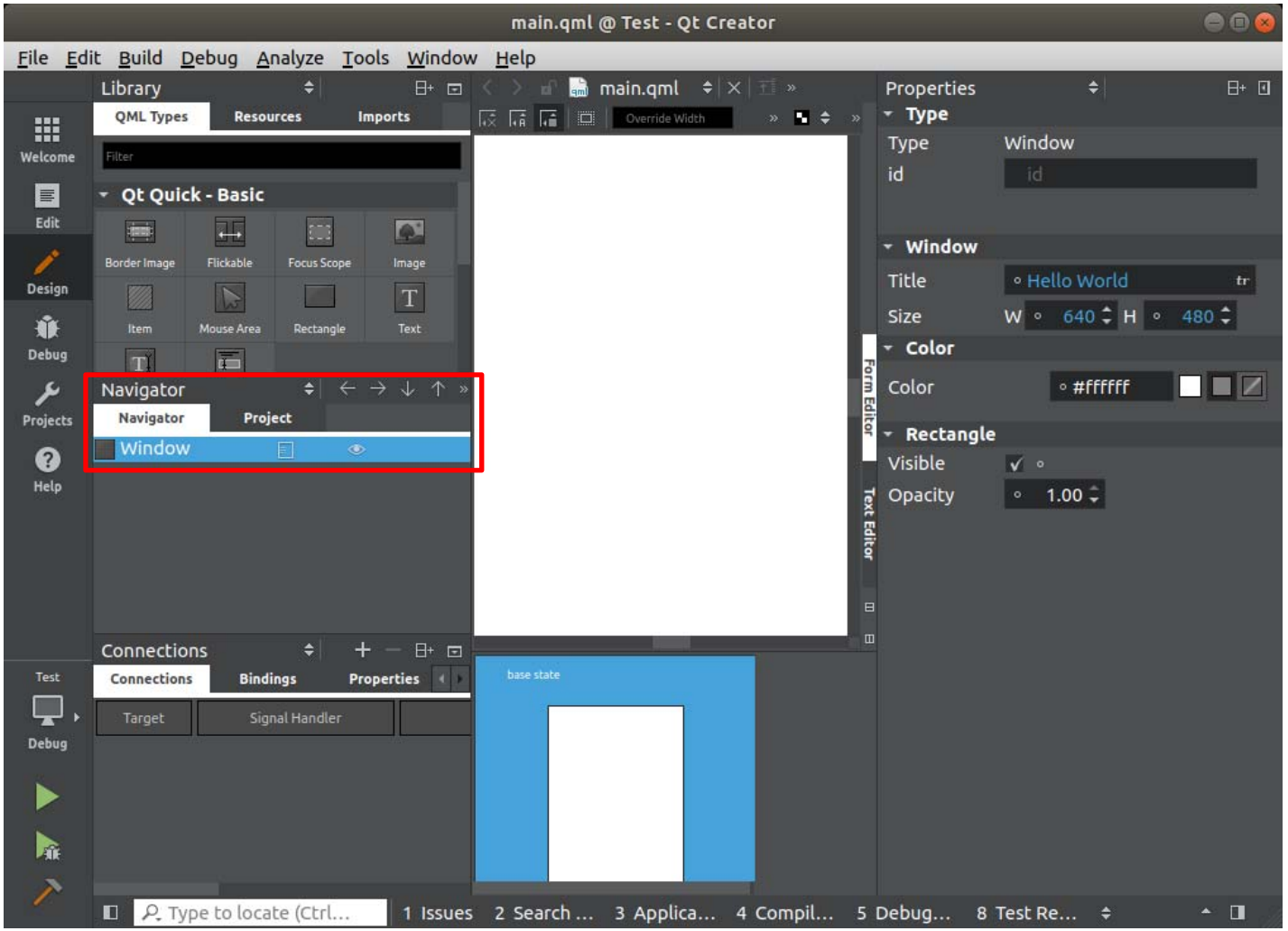
本マニュアルではバージョン管理の設定は行いません。Finishをクリックします。



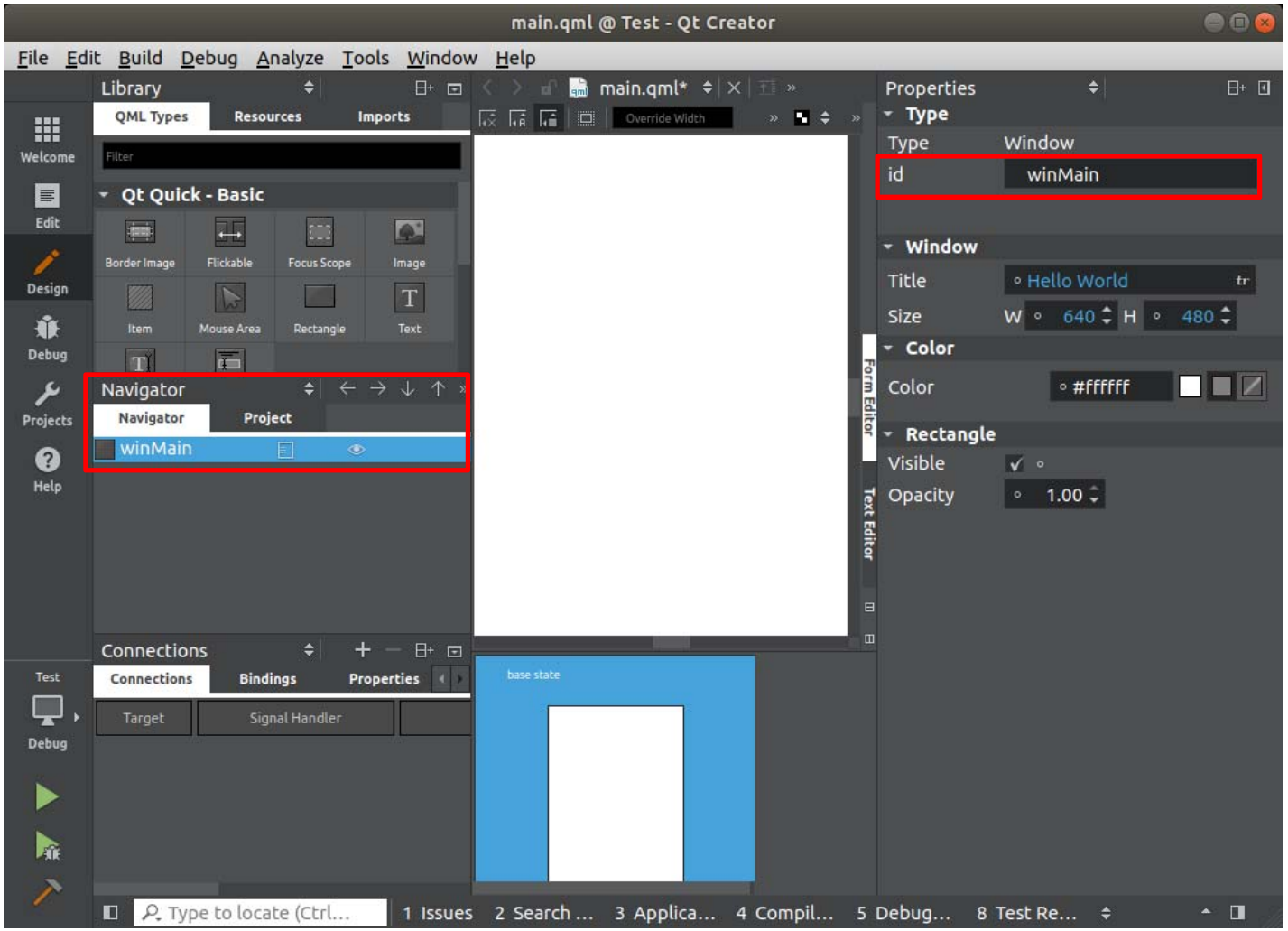
プロジェクトが作成されます。Designを開きます。



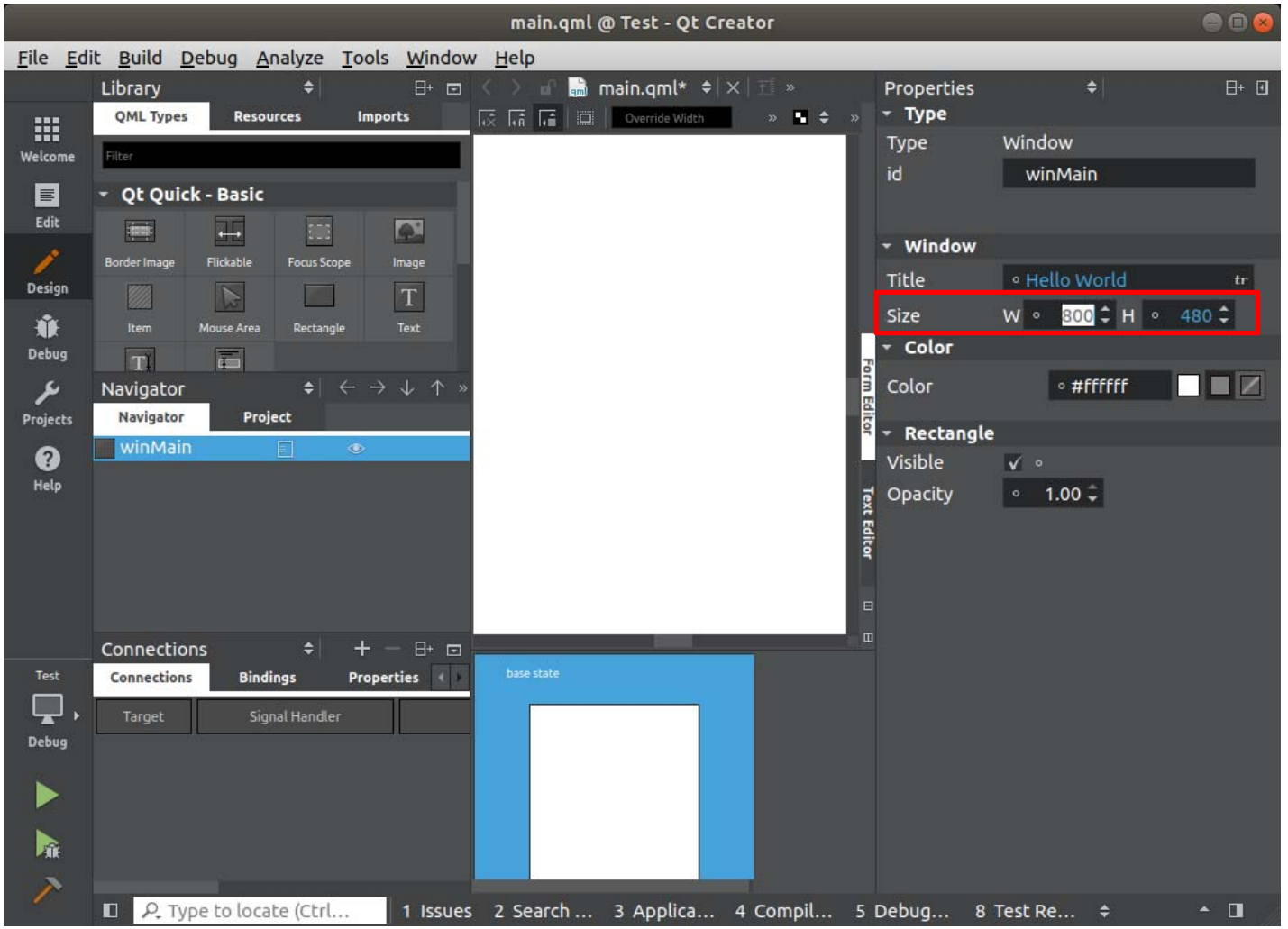
QT CreatorではGUIを容易に編集することができます。
画面左中央あたりのNavigatorの枠内にコンポーネントの一覧があります。最初はWindowという名前のWindow(Type)があります。



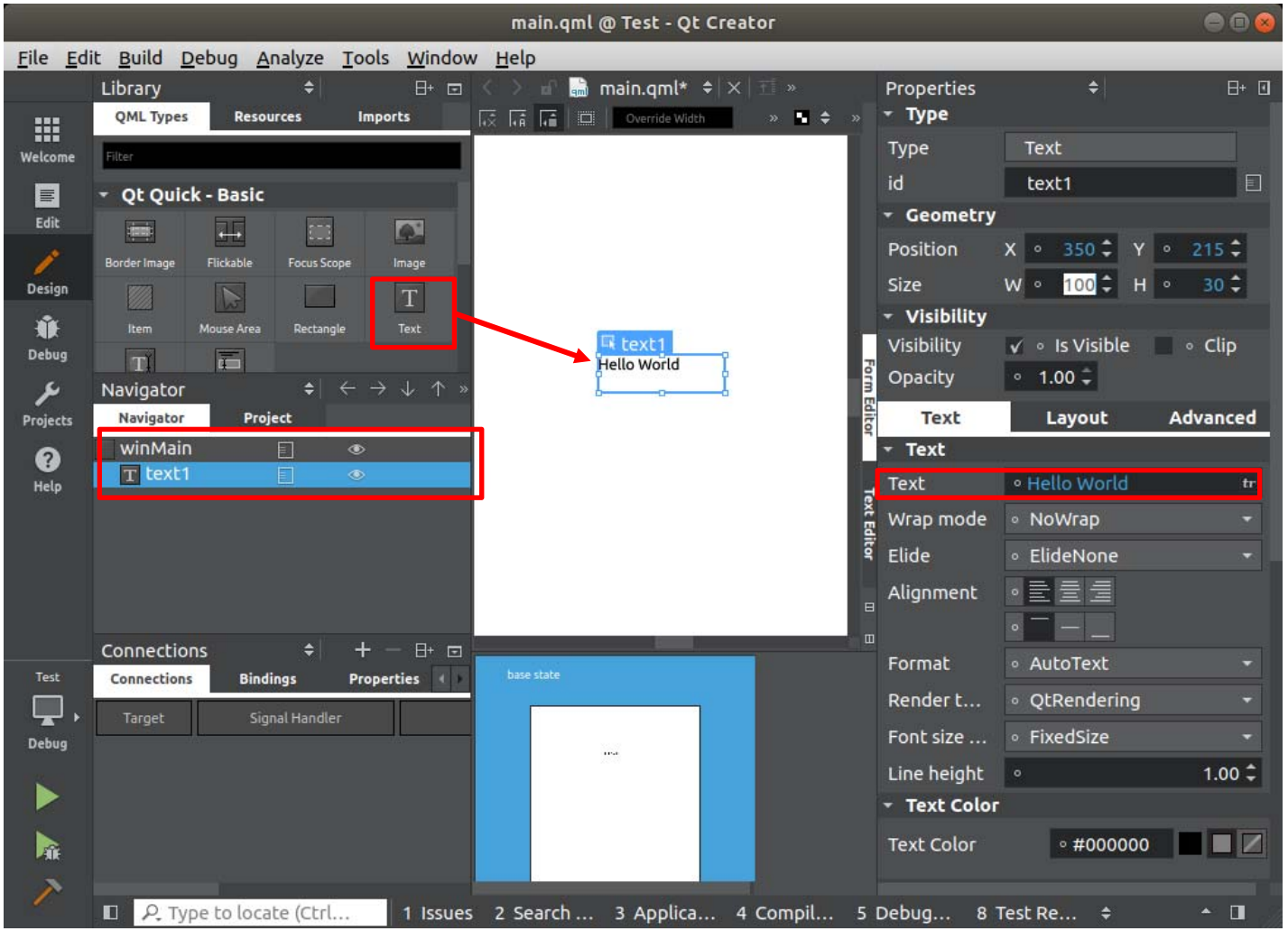
Navigatorの枠内でWindowを選択した状態で画面右側のProperties内のidを変更します。
本マニュアルではwinMainという名前に変更しています。変更するとNavigatorにあるアイテムも変わります。



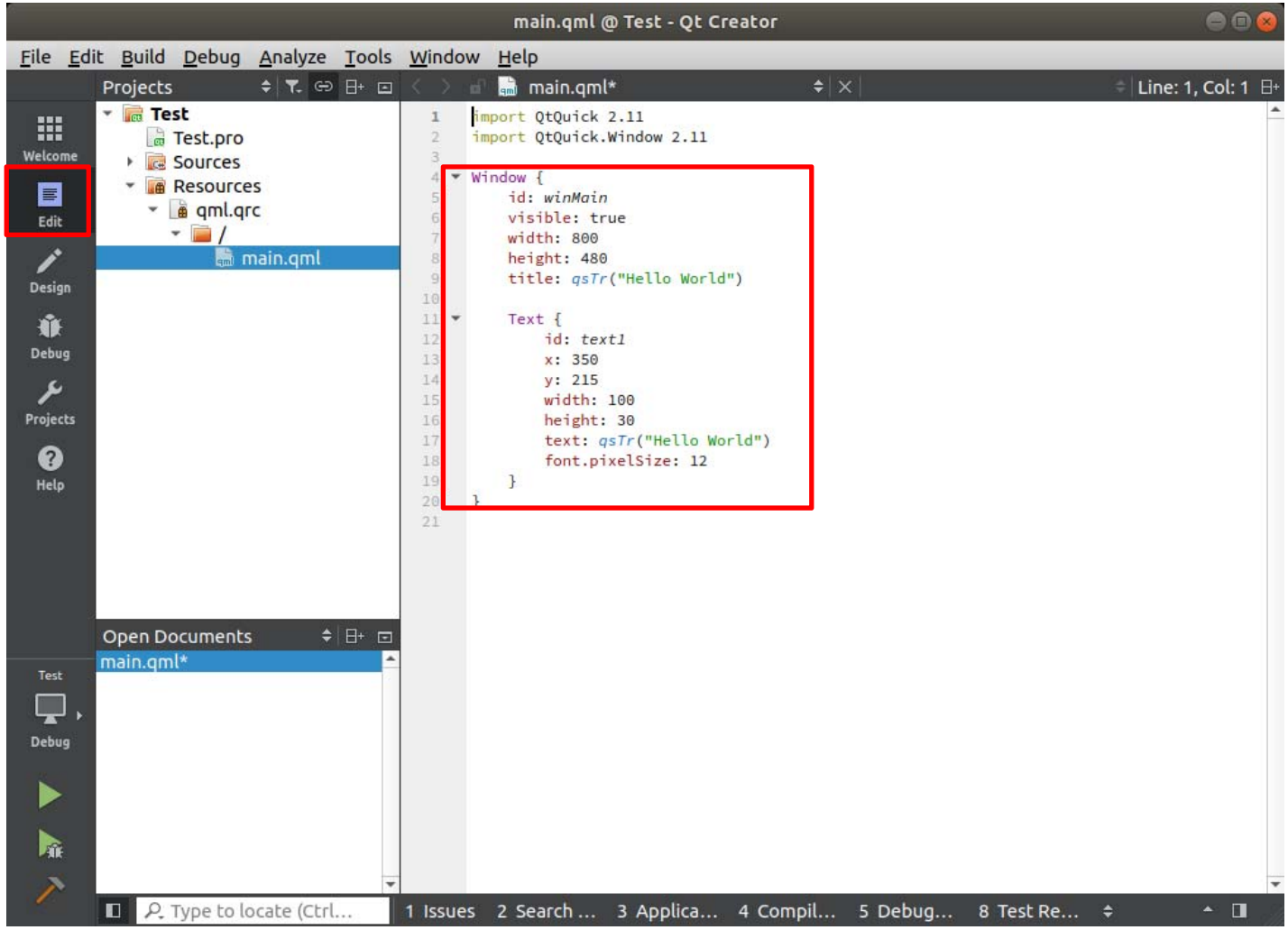
Sizeのプロパティを変更して液晶のサイズに合わせます。本マニュアルで使う7インチ液晶は800x480です。



本マニュアルではHello worldという表示をするテキストボックスを追加します。
Qt Quick -BasicからTextをドラッグ＆ドロップして真ん中の白い部分(main.qmlの画面)に持っていきます。
その後、Navigatorでtext1を選択した状態でTextプロパティにHello Worldと入力します。

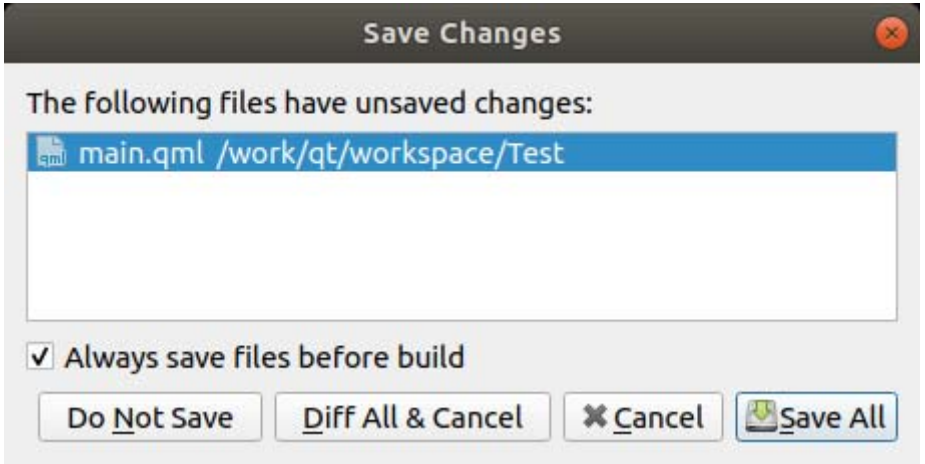


Editを選択してソースコードが表示される画面に戻ります。
main.qmlにDesignで編集した内容のコードが追加されているのが分かります。
プロジェクトを右クリックしてBuildを選択します。



下記のような保存していない内容を保存するかどうか問われます。
Save Allをクリックします。

Always save files before buildにチェックを入れておくと以後ビルド時に自動的に保存されます。

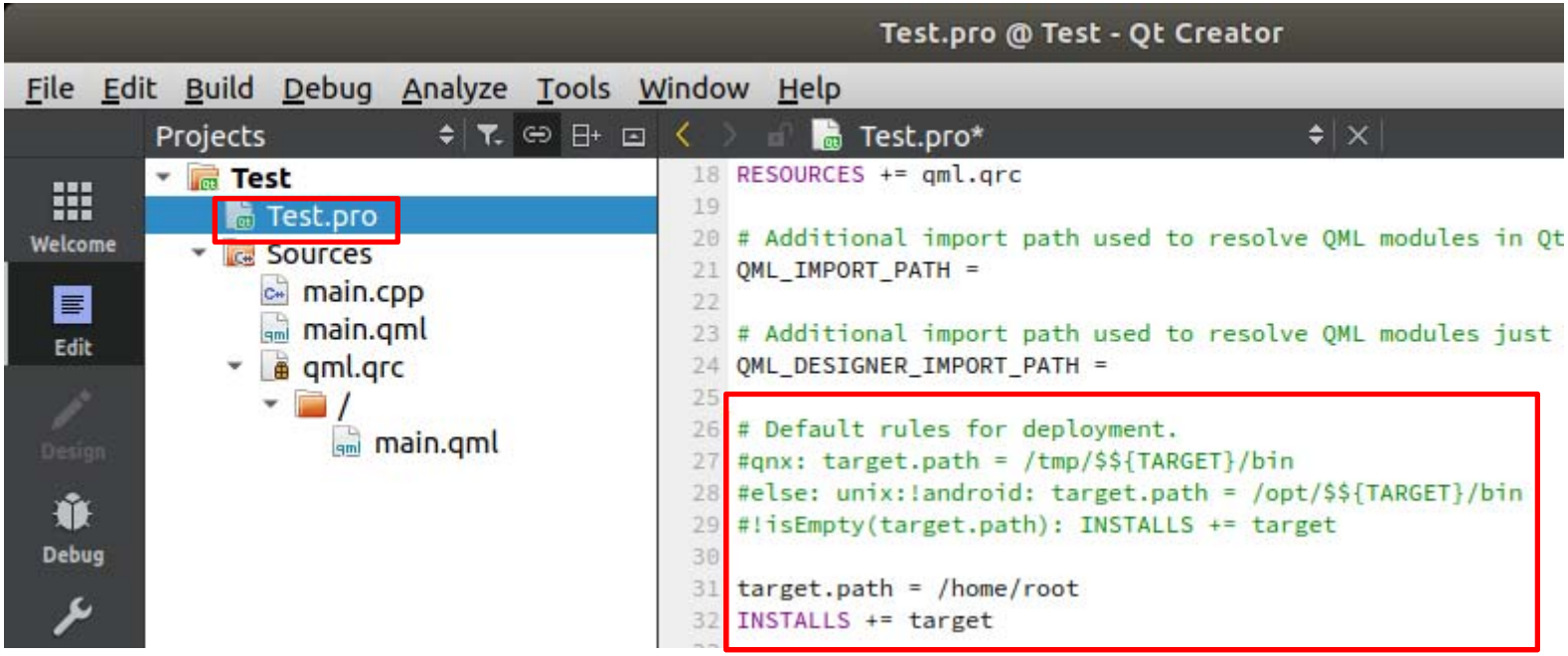


Test.proを開いてデプロイのディレクトリの修正をします。下記はモジュール上の/home/rootに変更する例です。

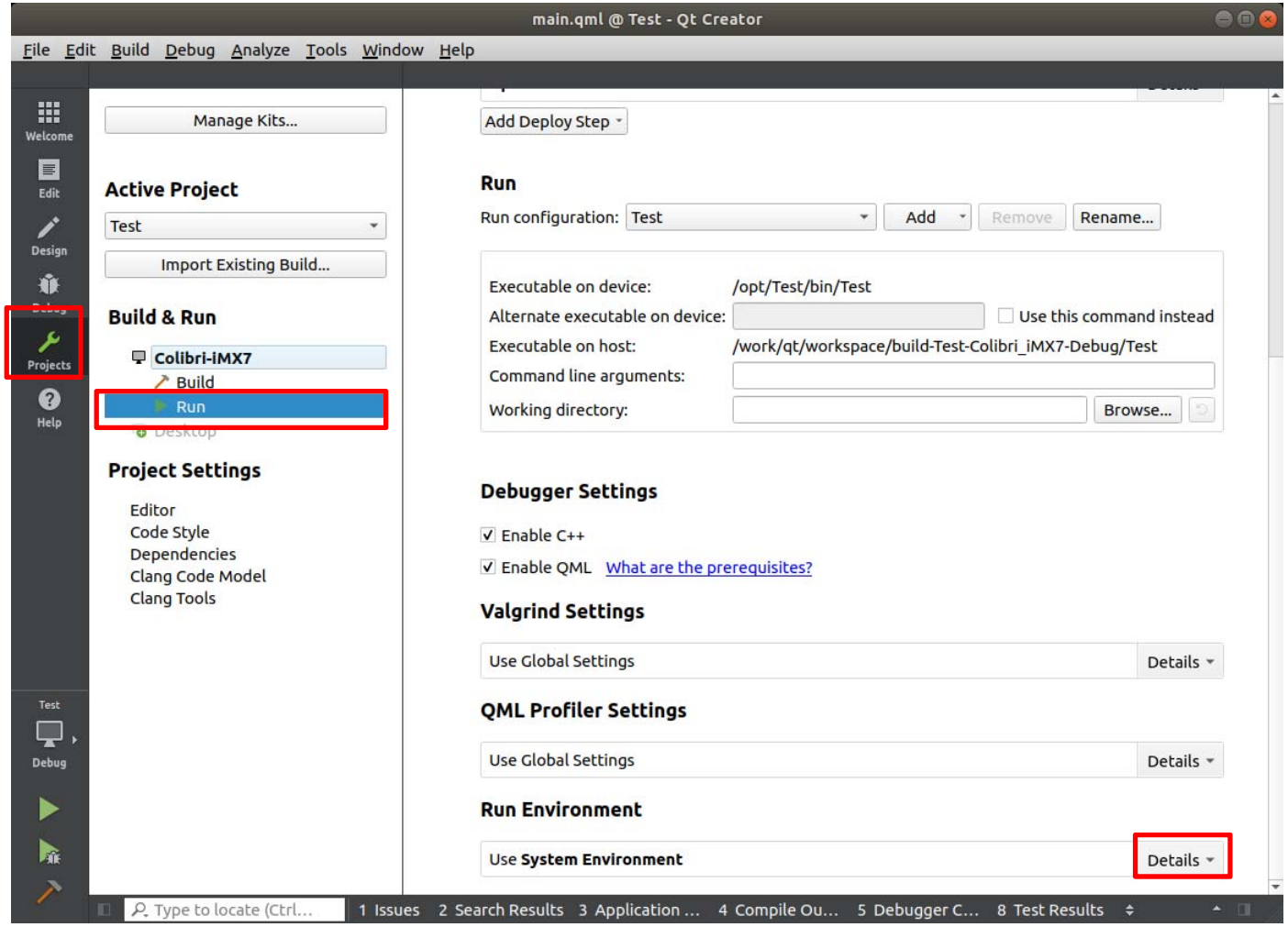
```
# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

->

```
target.path = /home/root
INSTALLS += target
```



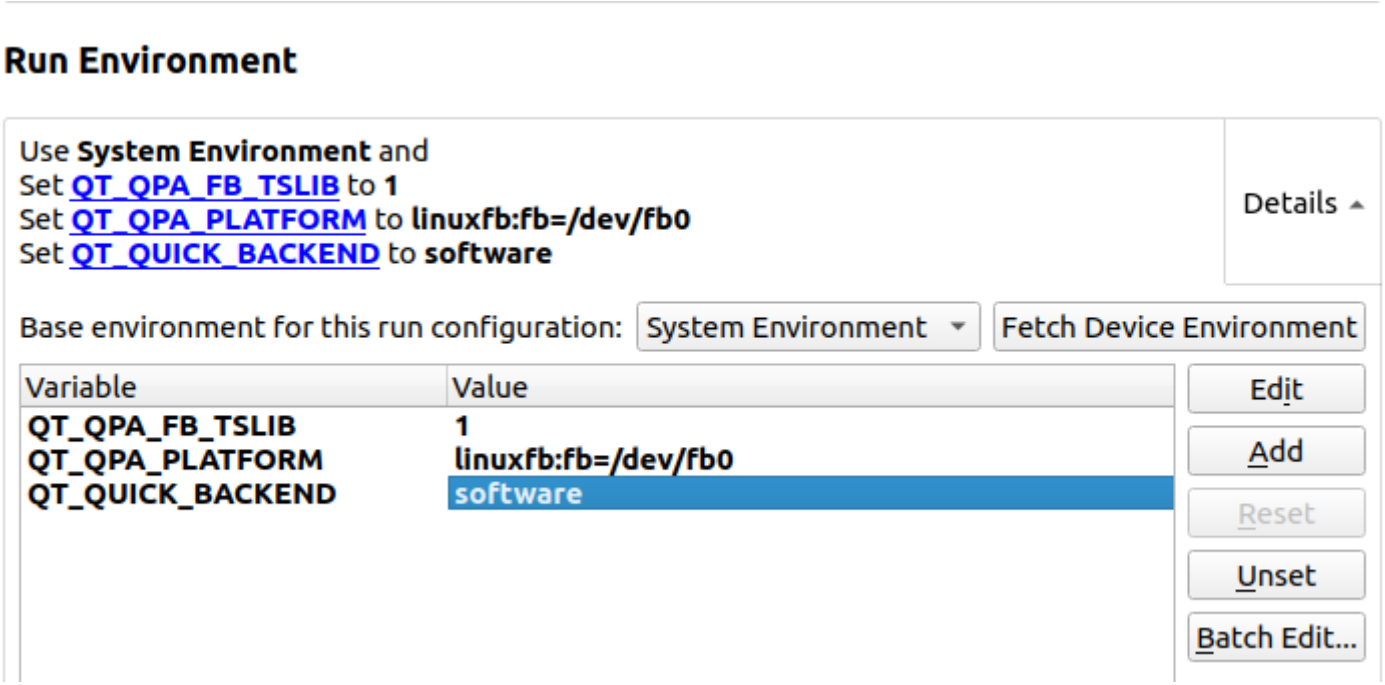
Projectsを開きRunを選択します。Run EnvironmentのDetailsをクリックします。



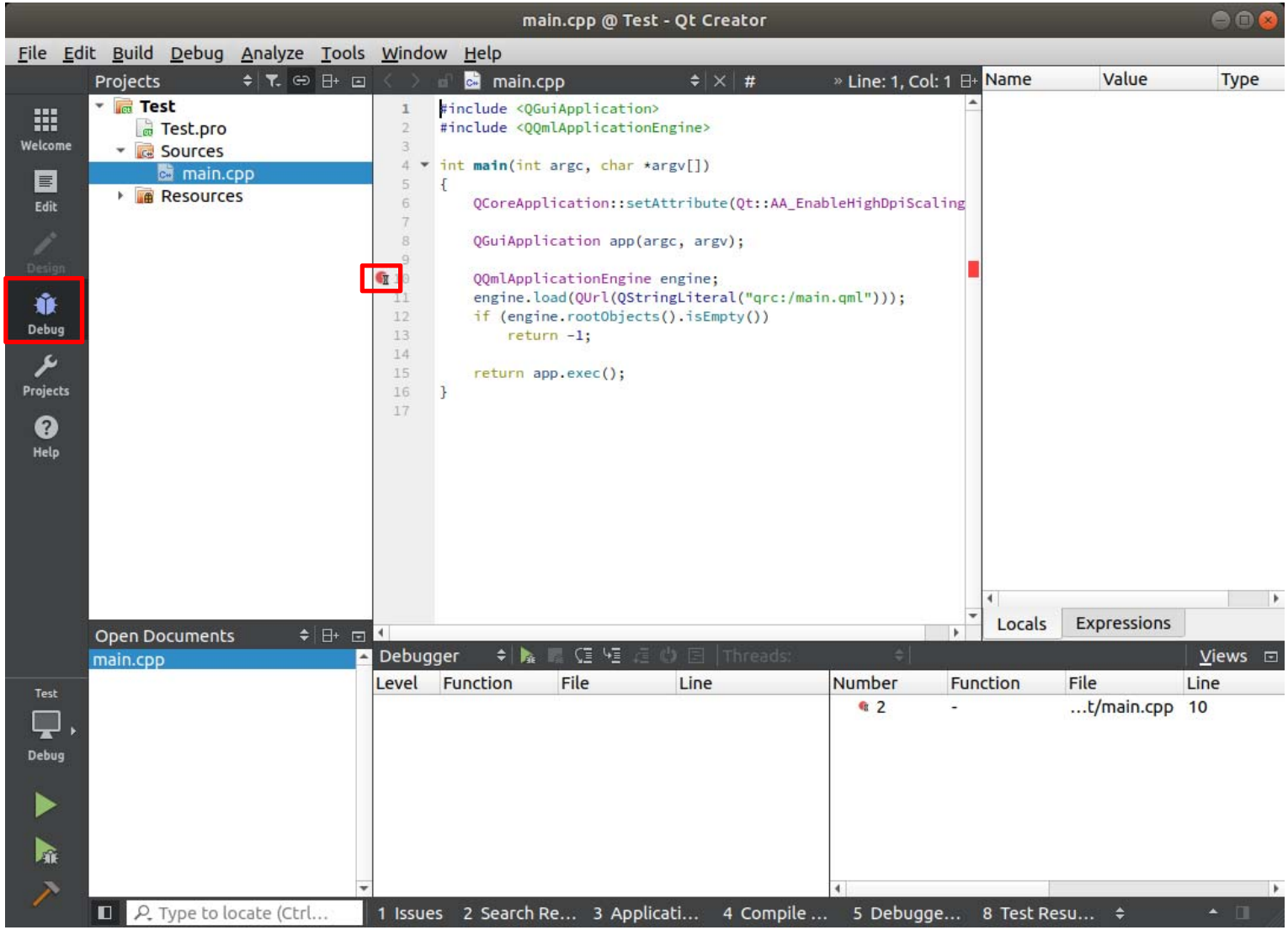
Addをクリックして下記3つの設定を追加します。
QT_QPA_FB_TSLIB　1
QTでTSLIBを使用する設定

QT_QPA_PLATFORM　linuxfb:fb=/dev/fb0
バックエンドがlinuxfbでフレームバッファのパスが/dev/fb0を意味します。

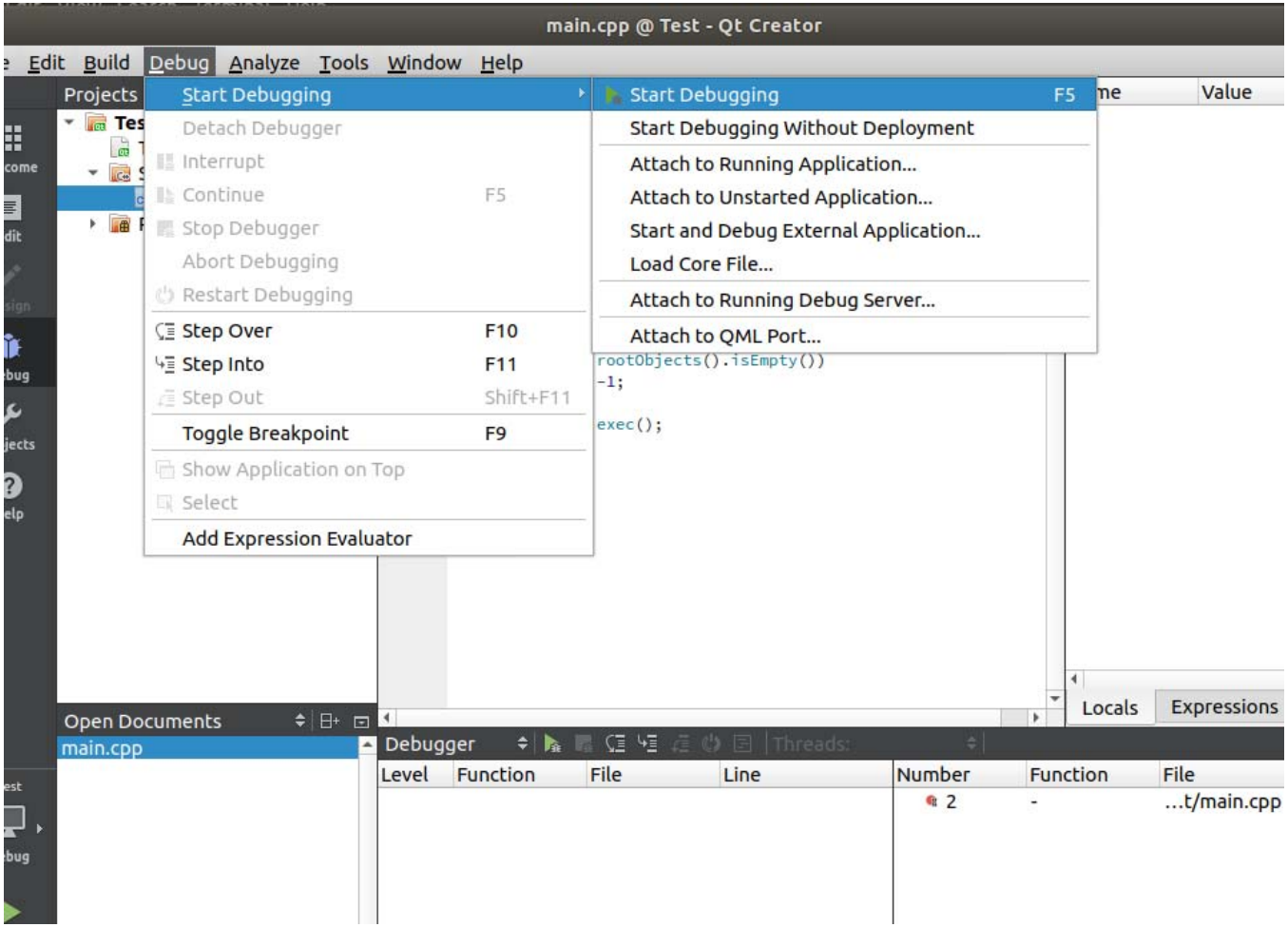
QT_QUICK_BACKEND　software
ソフトウェアレンダリングを意味します。



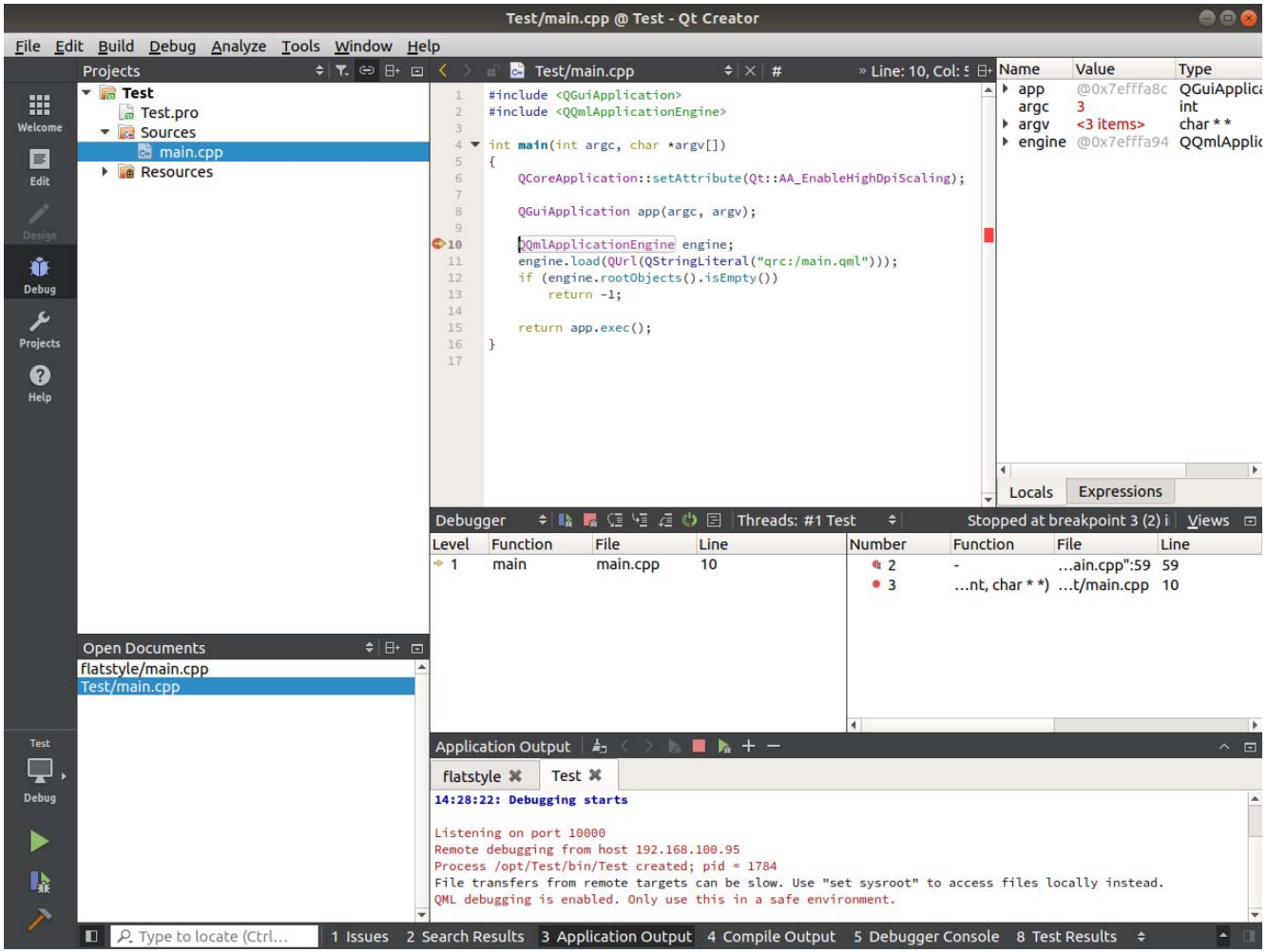
Debugの項目を開き、Projectsからmain.cppを開きます。
行番号の左側をクリックしてブレイクポイントを作ります。ブレイクポイントができると赤丸が出てきます。



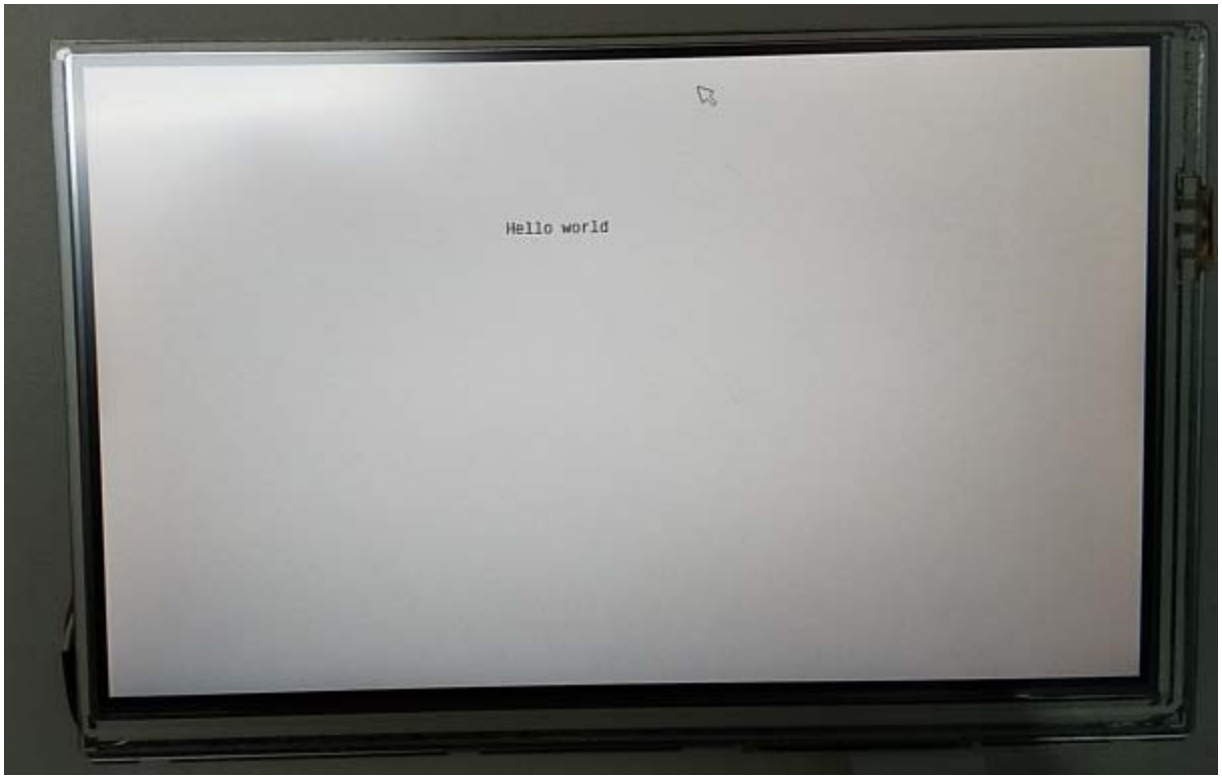
メニューのDebugからStart Debugging > Start Debuggingを選択します。



デバッグが開始してブレイクポイントで止まります。

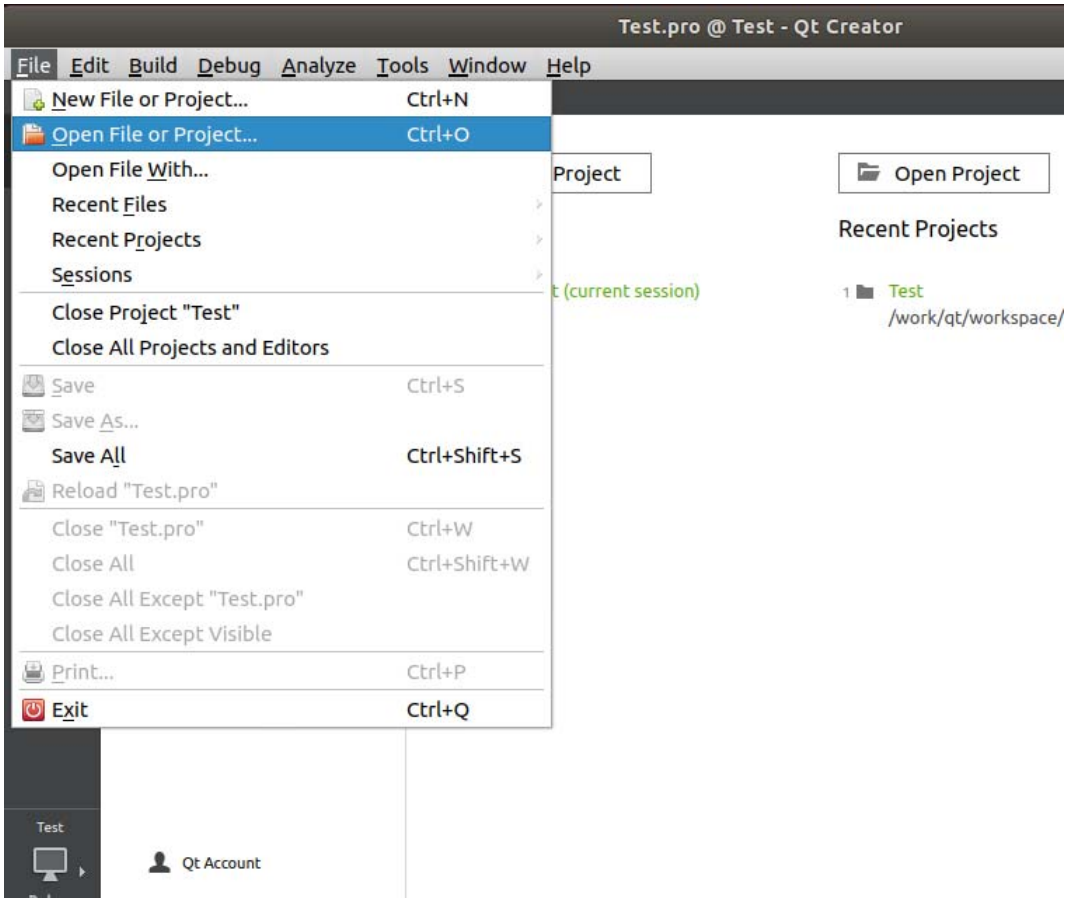


F5でデバッグを続行すると下記のような画面が表示されます。

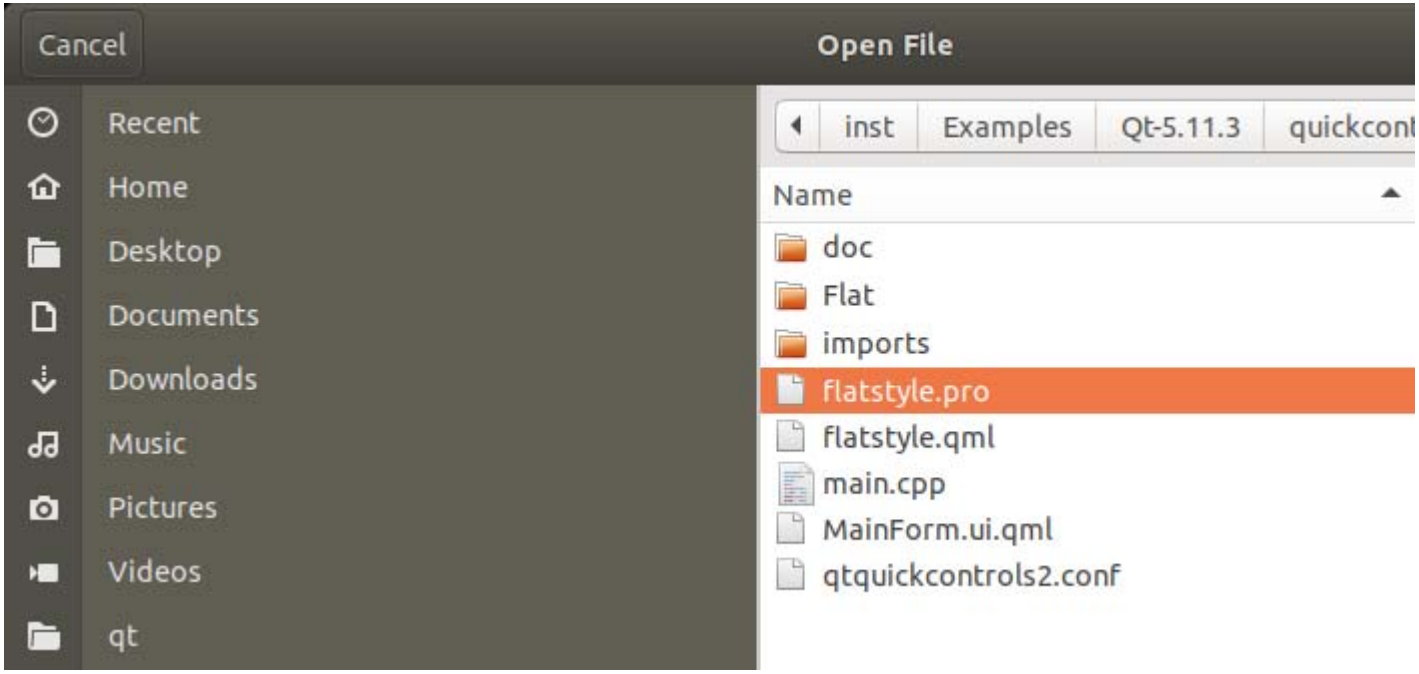


サンプルプログラムの読み込み

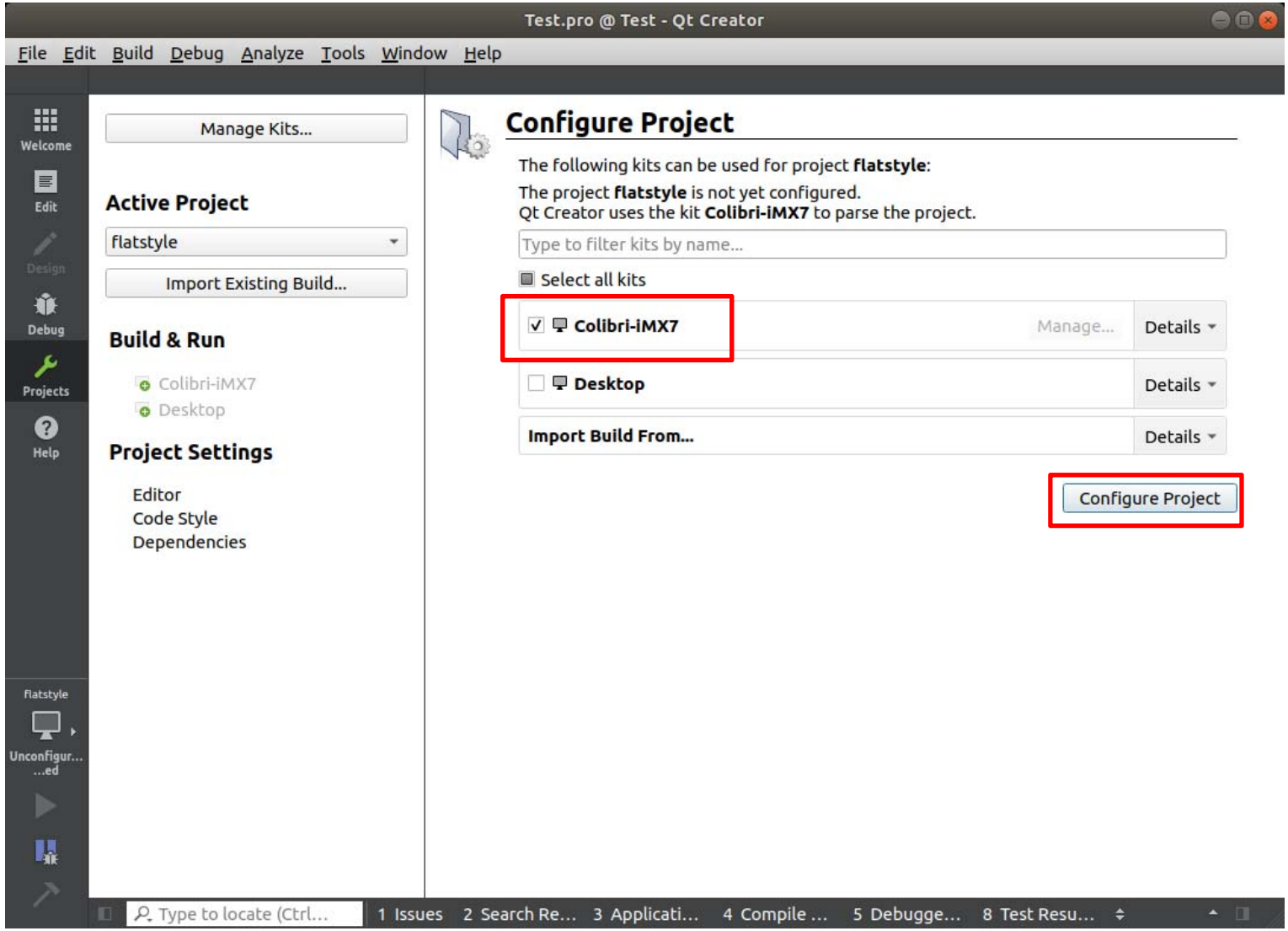
既存のプロジェクトと読み込む場合はFile > Open File or Projectを選択します。



本マニュアルではタッチパネルの動作確認がしやすいflatstyleというサンプルコードを実行します。
/work/qt/inst/Examples/Qt-5.11.3/quickcontrols2/flatstyle/flatstyle.proを選択します。

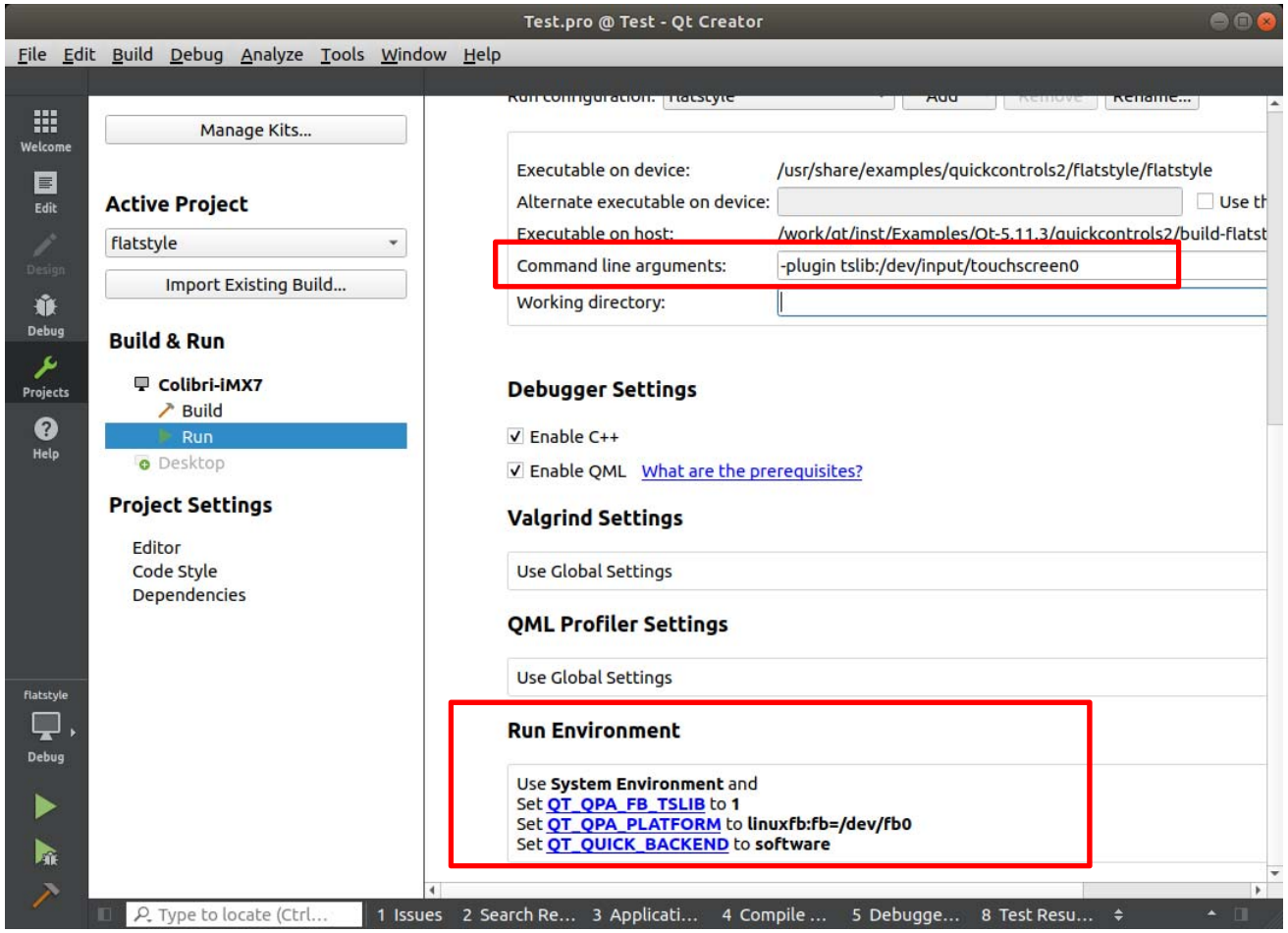


プロジェクトを読み込むとそのKitsで使用するかを問われます。
作成したColibri-iMX7にチェックを入れてConfigure Projectをクリックします。

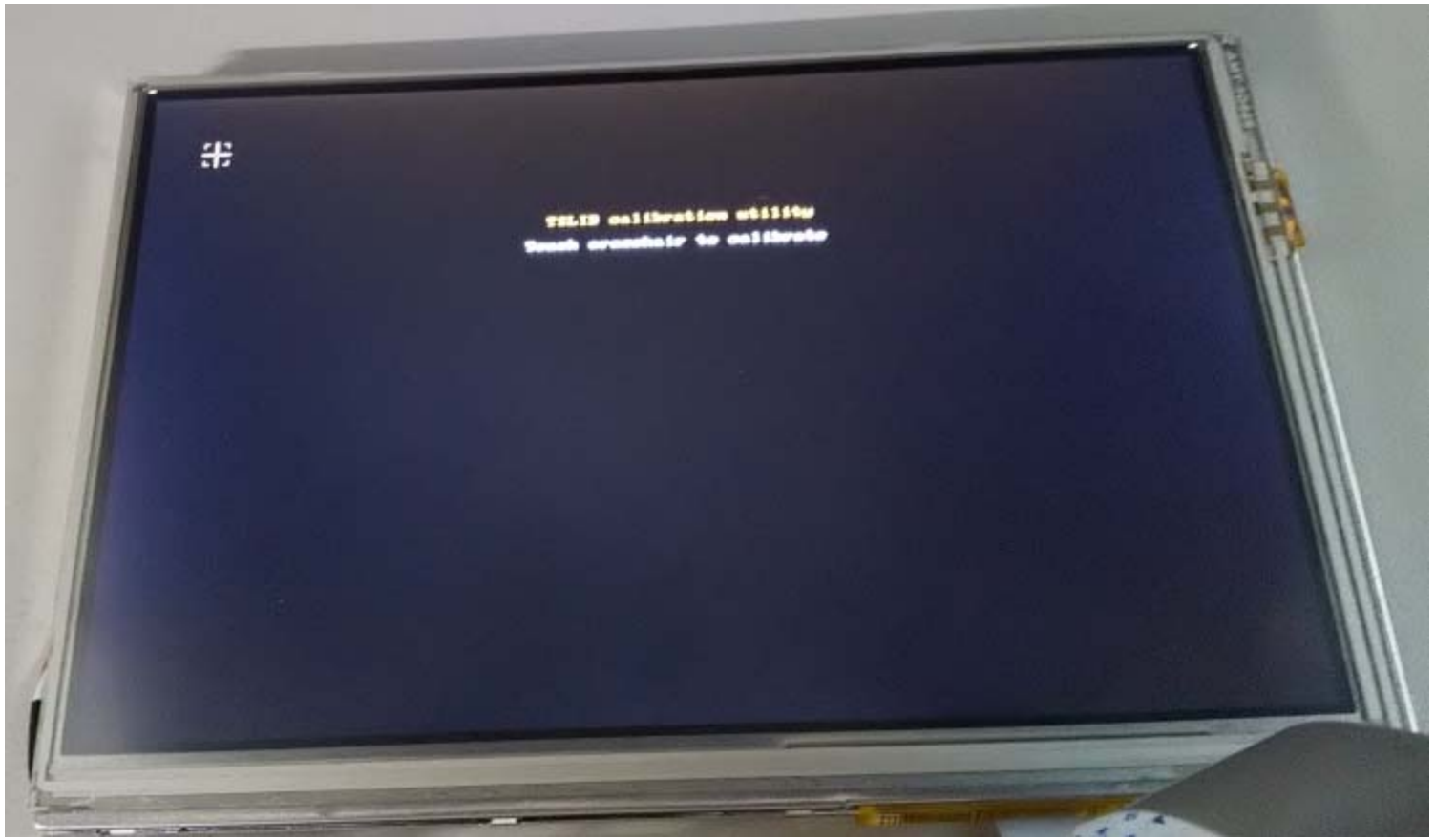


プロジェクトを作成した時と同様Run Environmentに下記3つを設定してください。
QT_QPA_FB_TSLIB　1
QT_QPA_PLATFORM　linuxfb:fb=/dev/fb0
QT_QUICK_BACKEND　software

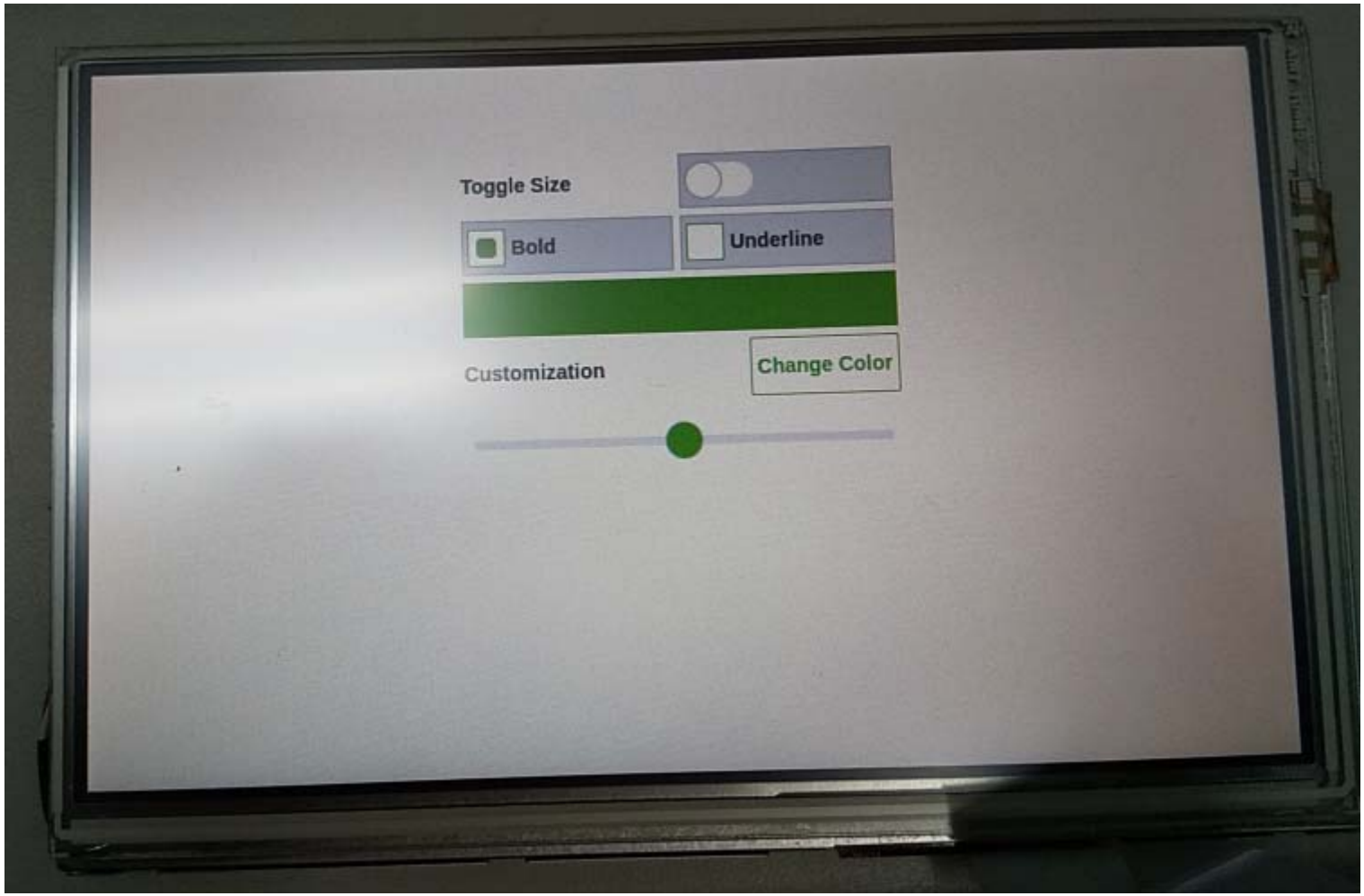
Command Line argumentsに下記を設定してください。タッチパネルを動かすための起動オプションです。
-plugin tslib:/dev/input/touchscreen0



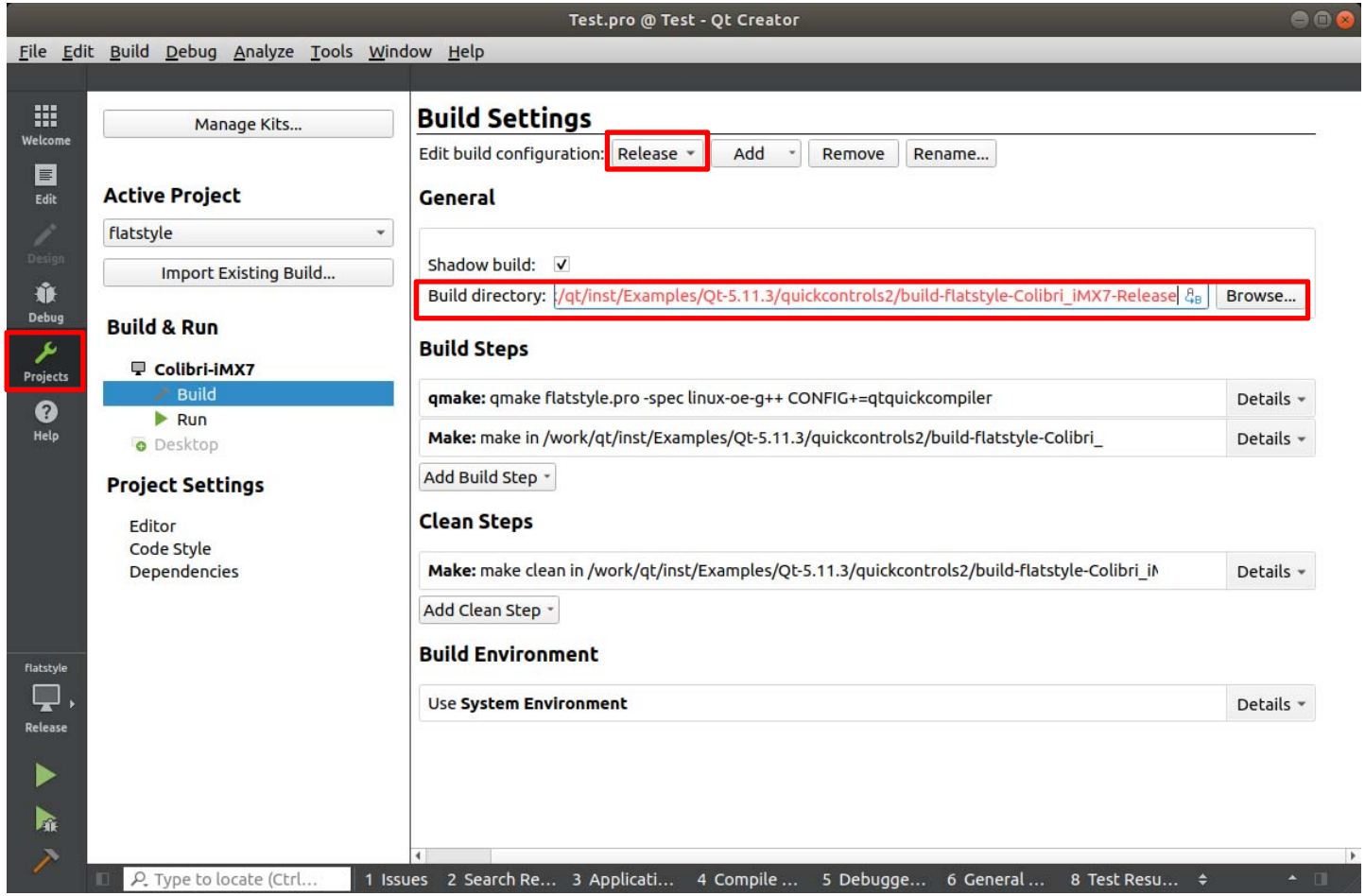
デバッグ前にts_calibrateでタッチパネルのキャリブレーションを行います。
モジュールをリブートしてからts_calibrateを実行します。(QTアプリケーション起動後は動作しません。)
[colibri-imx7]# ts_calibrate
タッチしてキャリブレーションを行ってください。



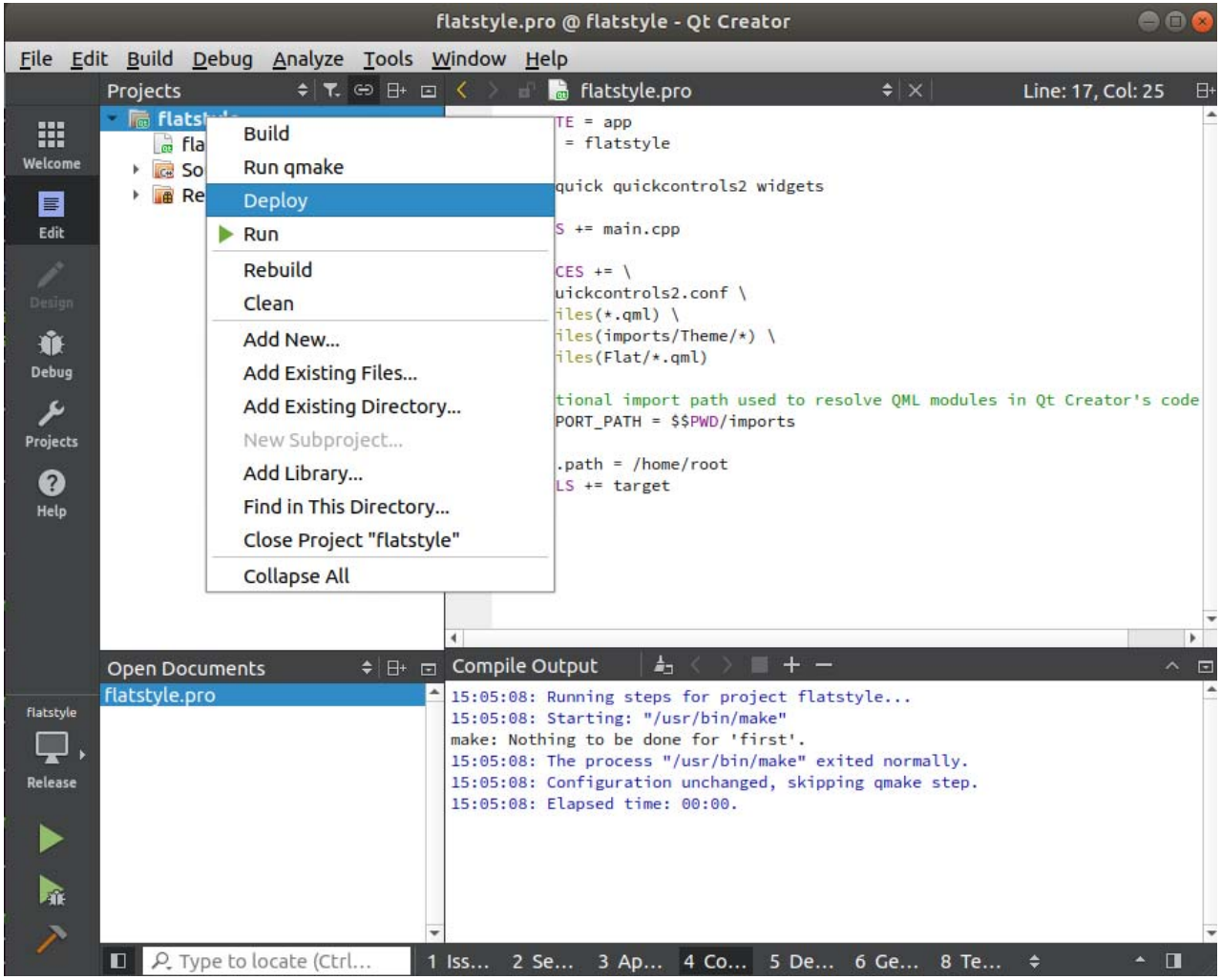
デバッグを開始すると下記のような画面が表示されます。スライダーやチェックボックスなどをタッチパネルで操作することができます。



デバッグではなく最終製品向けの実行をする場合はリリースビルドに変更します。Projectsの項目を開きDebugをReleaseに変更してからビルドを行います。Build Directoryに設定されたディレクトリに実行ファイルが出力されます。



プロジェクト右クリック > DeployでBuildした実行ファイルをデプロイできます。
.proファイルにtarget.path = /home/rootを定義していた場合/home/rootにデプロイされます。



モジュール上で実行するにはデバッグ時と同様に環境変数の設定が必要です。
下記3つを実行します。
export QMLSCENE_DEVICE=softwarecontext
export QT_QPA_PLATFORM=linuxfb:fb=/dev/fb0
export QT_QPA_FB_TSLIB=1

/etc/profile　に設定しておくと毎回設定しなくても起動時に自動的に設定されます。
設定後は反映させるために一度リブートしてください。

アプリケーションを実行することができるようになります。
[colibri-imx7]# ./flatstyle -plugin tslib:/dev/input/touchscreen0

以上がQT開発の流れです。
QTのアプリケーション開発はサンプルコードやインターネット上の情報をもとに行ってください。