Toradex
Linux QTアプリケーション
開発マニュアル
BSP5用

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
1.00	2022/11/07	BSP5系用に作成

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**を利用して**GUI**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 v15.5.7
Host OS： Windows 10 21H2
Guest OS： Ubuntu Desktop 20.04LTS 64bit(**英語版**)
BSP：v5.7
QT：v5.14
CPU**モジュール**：Verdin-iMX8M Plus Quad 4GB Wi-Fi / Bluetooth IT 1.1A
キャリアボード：**キャリアボード**：Verdin**開発ボード** Rev 1.1C + **アクセサリーキット**
ディスプレイ：10.1**インチディスプレイ**1.0A **静電式タッチパネル付き** + DSI to LVDS**アダプタ** 1.1

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。

3. 前提知識

Linux OSイメージ開発マニュアルの内容、TEZIによるOS書き込み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(パソコン)と実行環境(モジュール)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(パソコン)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(Linux) ：[Module]#
実行環境(モジュール)上で入力するコマンド(U-Boot)：[U-Boot]#

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

tdx-reference-multimedia-imageにはQTの機能が含まれています。このイメージを使用します。

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

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

QTCreatorがrsyncを使用するため下記を追記します。
IMAGE_INSTALL_append = " rsync"

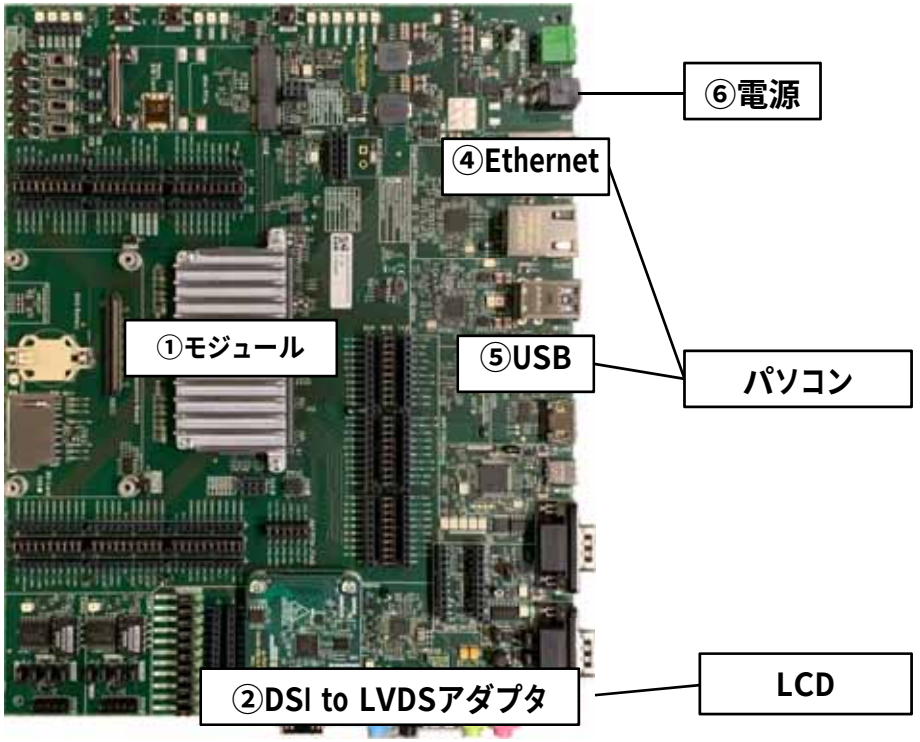
OSイメージを作成します。
[Ubuntu]$ bitbake tdx-reference-multimedia-image

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。

ハードウェアの準備

以下を行います。

①X1にモジュールを挿します。ヒートシンクも装着します。
②X21にDSItoLVDSアダプタとその先にLCDを接続します。 HDMIやTightVNC Viewerなどで代用できます。
③X34にUSBケーブルを差し込みパソコンと接続します。
④X25にEthernetケーブルを挿入します。パソコンと同じネットワークに接続してください。
⑤X58にACアダプタを接続します。



LCDを表示するためのセットアップ

開発ボードの電源を入れてLinuxを起動します。
初期状態ではNative HDMIがメインの出力先になっています。
Device Tree Overlay設定でLCDが表示されるように変更します。

[Module]$ vi /boot/overlays.txt
fdt_overlays=verdin-imx8mp_native-hdmi_overlay.dtbo verdin-imx8mp_lt8912_overlay.dtbo

->

fdt_overlays=touch-atmel-mxt_overlay.dtbo verdin-imx8mp_sn65dsi84-lt170410_overlay.dtbo verdin-imx8mp_sn65dsi84_overlay.dtbo

再起動します。
LCDにQTのデモプログラムが表示されます。

開発の邪魔になるためデモプログラムが起動しないように設定しておきます。
[Module]$ systemctl disable wayland-app-launch

デスクトップも消したい場合は下記を設定してwestonを再起動すれば消えます。
[Module]$ vi /etc/xdg/weston/weston.ini
[shell]
background-color=0x00FFFFFF
panel-position=none

設定後westonを再起動してください
[Module]$ systemctl restart weston@root

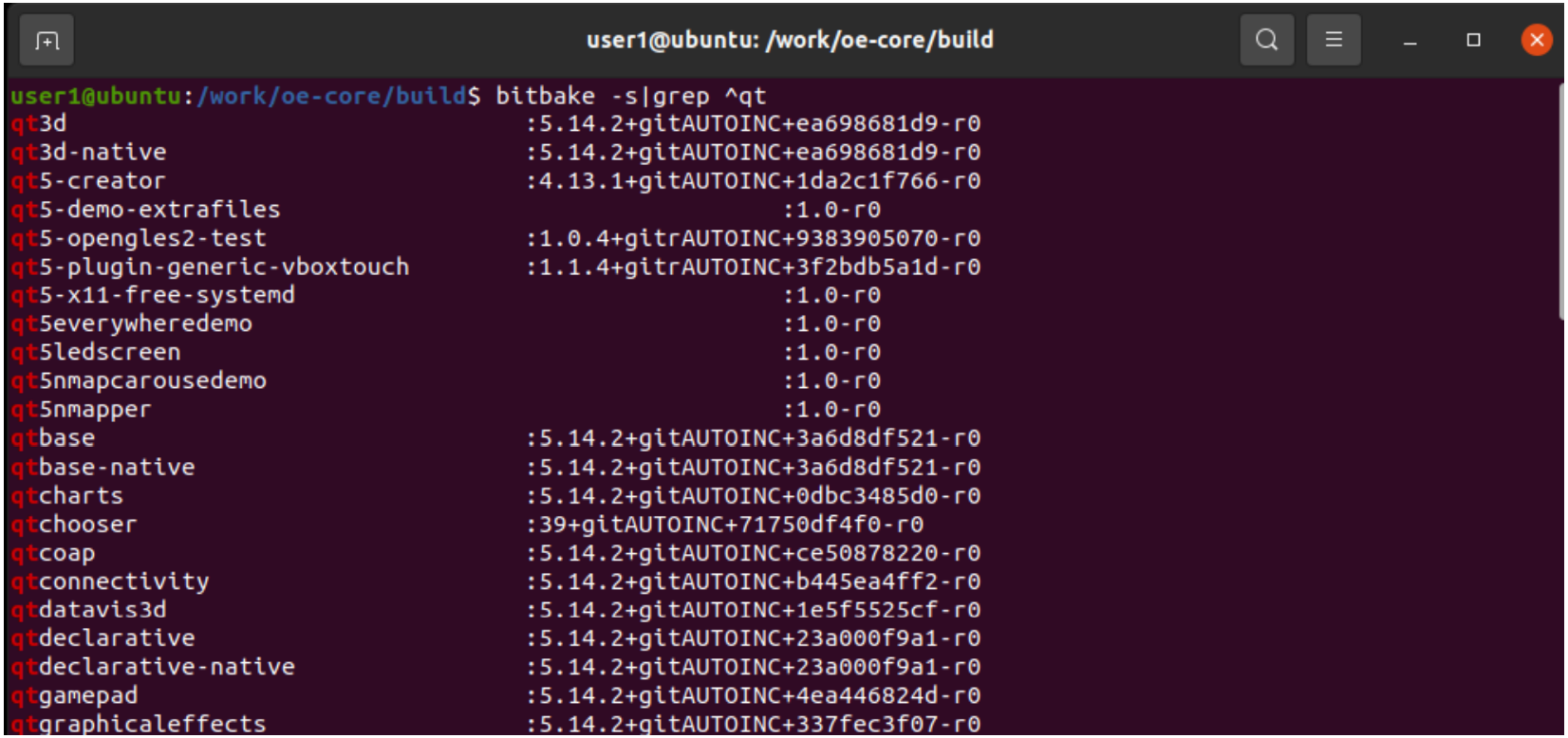
開発環境構築

Ubuntuで開発環境を構築します。

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。

最初にどんな機能が必要かわからない場合はそのまま進めて開発中に不足したことに気づいた時に再度OSを作り直してください。必要な機能を追加するときはlocal.confに追加します。

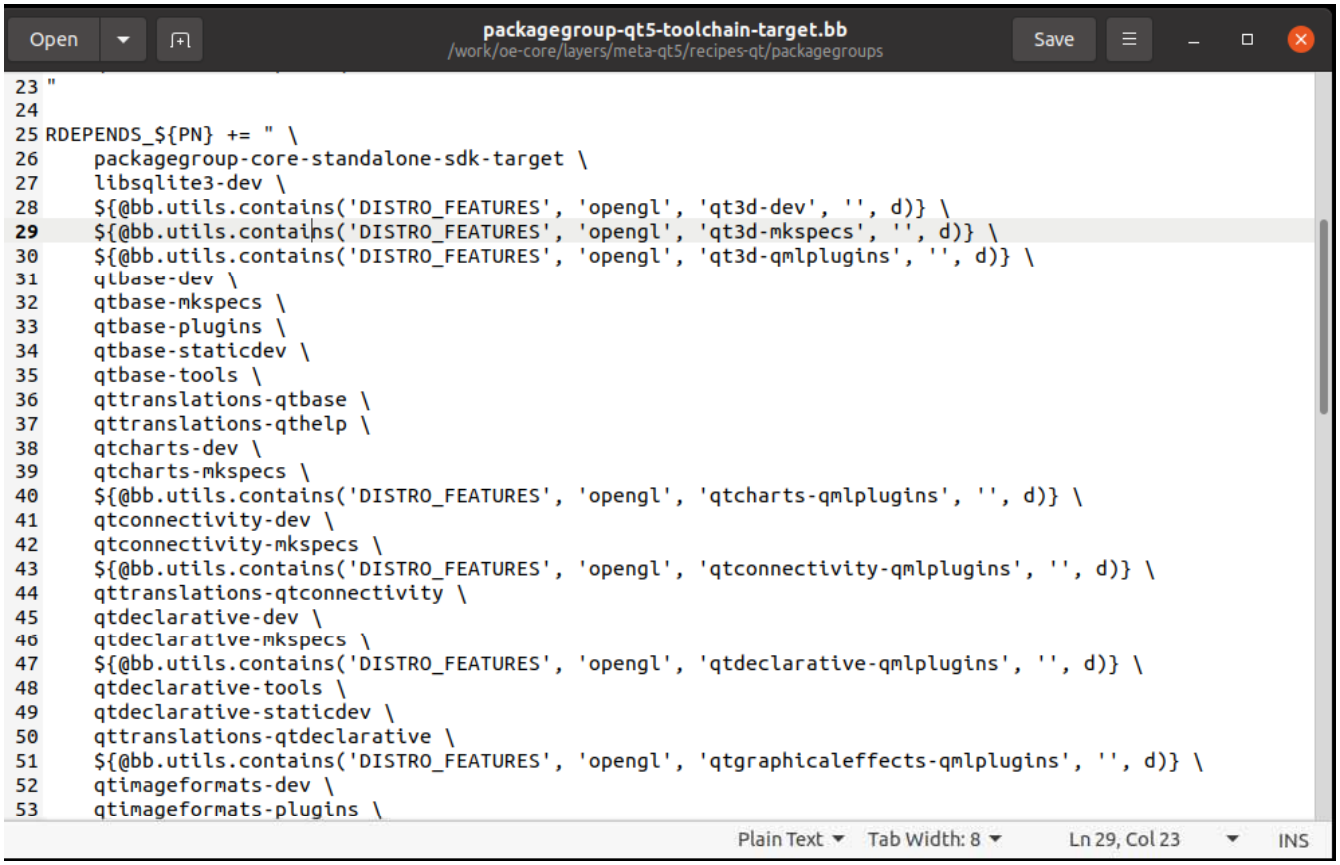
IMAGE_INSTALL_append=" <レシピ名>"



SDKの作成

QtCreatorが使用するコンパイラやデバッガ、ヘッダー、ライブラリなど一式をSDKとして出力することができます。
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。
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

QTの機能を追加した場合SDK用の機能も存在していなければ同様に追加してください。



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

/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を出力する場合に備えてSDKをqt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]$ sudo /work/oe-core/build/deploy/sdk/tdx-xwayland-glibc-x86_64-meta-toolchain-qt5-aarch64-verdin-imx8mp-toolchain-5.7.0.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-xwayland/5.7.0):
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-xwayland/5.7.0". Proceed [Y/n]?
Yを入力してEnterキーを押します。

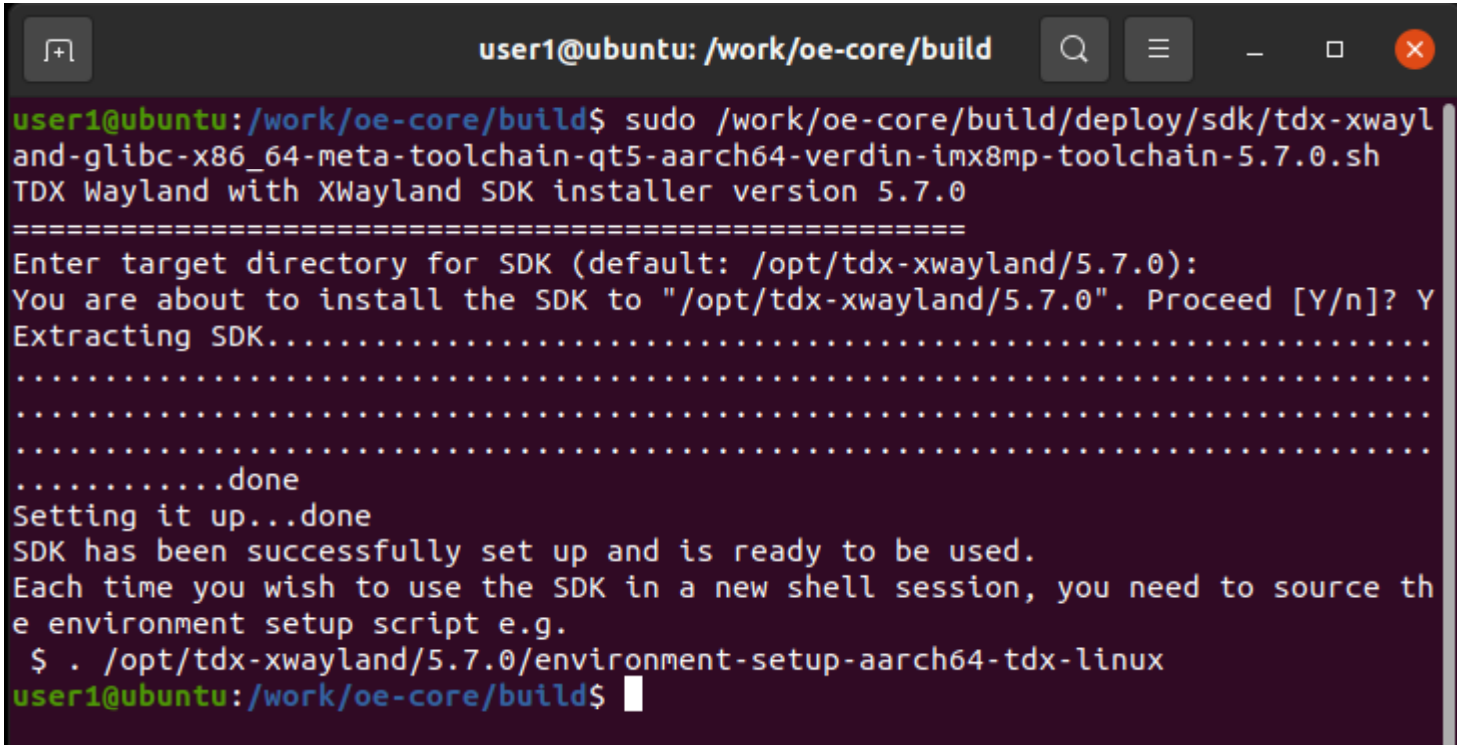
すでに存在していた場合は下記のように上書きするかどうかを問われますがその場合は一度シェルを停止してシェル実行前にディレクトリを削除しておいたほうが良いです。

If you continue, existing files will be overwritten! Proceed [y/N]?

[Ubuntu]$ sudo rm -rf /opt/tdx-xwayland/5.7.0

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-xwayland/5.7.0/environment-setup-aarch64-tdx-linux



QTの開発は最低限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。
BSP5.7.0のQTのバージョンは5.14.2です。

インストーラーのダウンロード
[Ubuntu]$ wget https://download.qt.io/archive/qt/5.14/5.14.2/qt-opensource-linux-x64-5.14.2.run

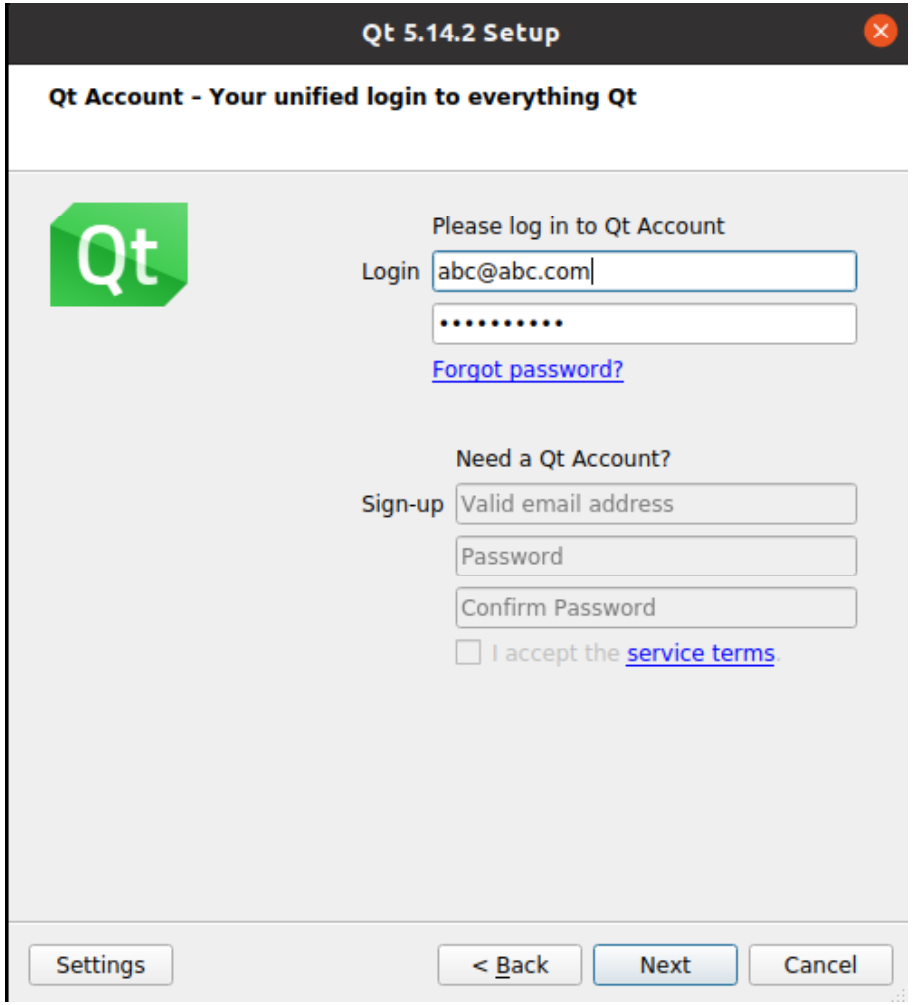
実行権限付与
[Ubuntu]$ chmod +x ./qt-opensource-linux-x64-5.14.2.run

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

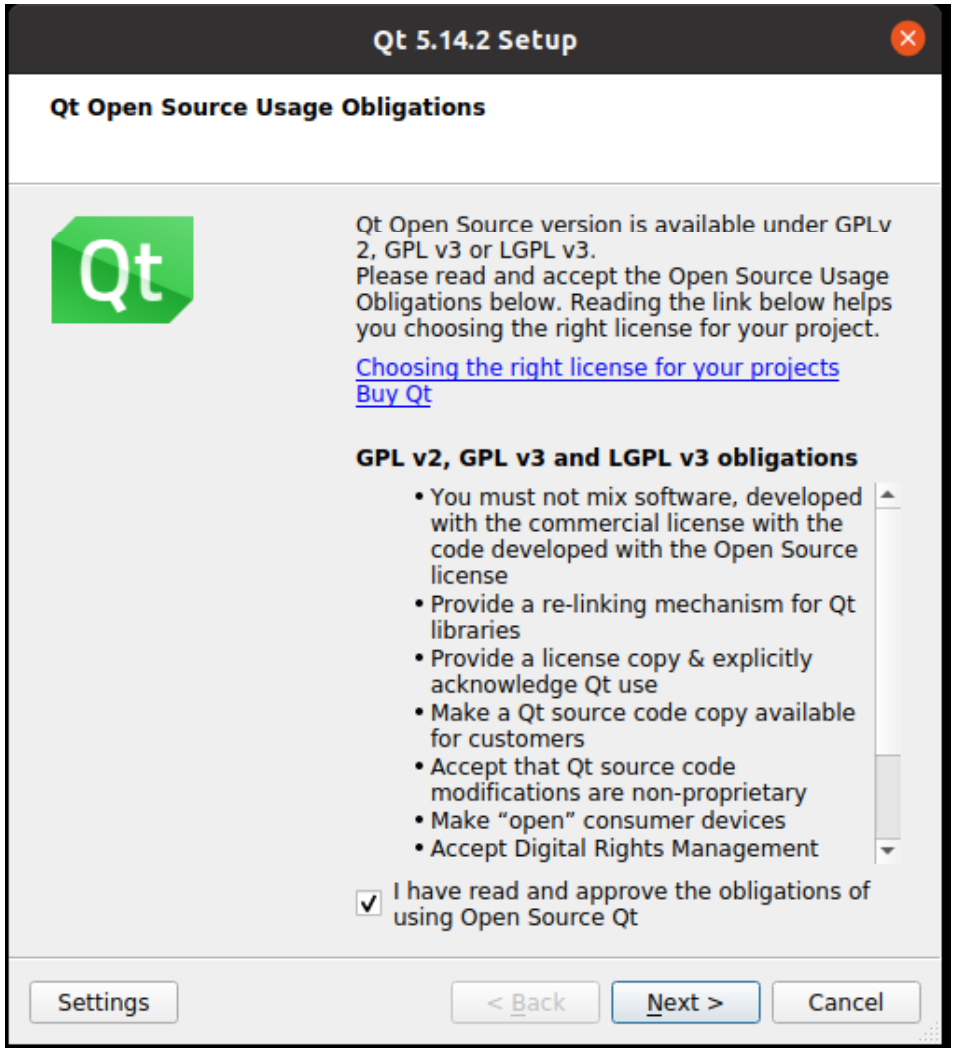
インストーラ実行
[Ubuntu]$ ./qt-opensource-linux-x64-5.14.2.run

Nextをクリックします。

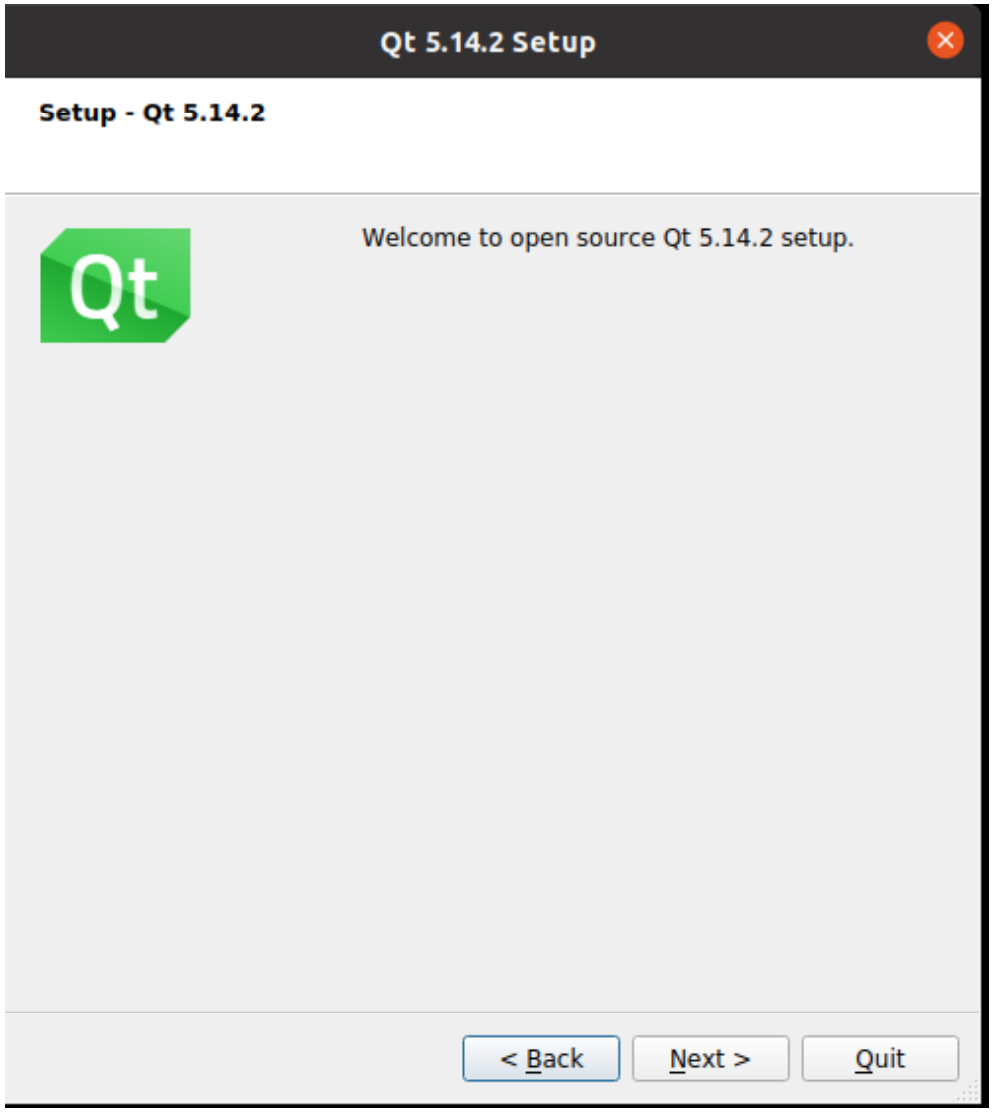
ログインIDとパスワードを入力してNextをクリックします。
QTのアカウントを持っていない場合はNeed a Qt Account?の下の項目を入力してアカウントを作成する必要があります。



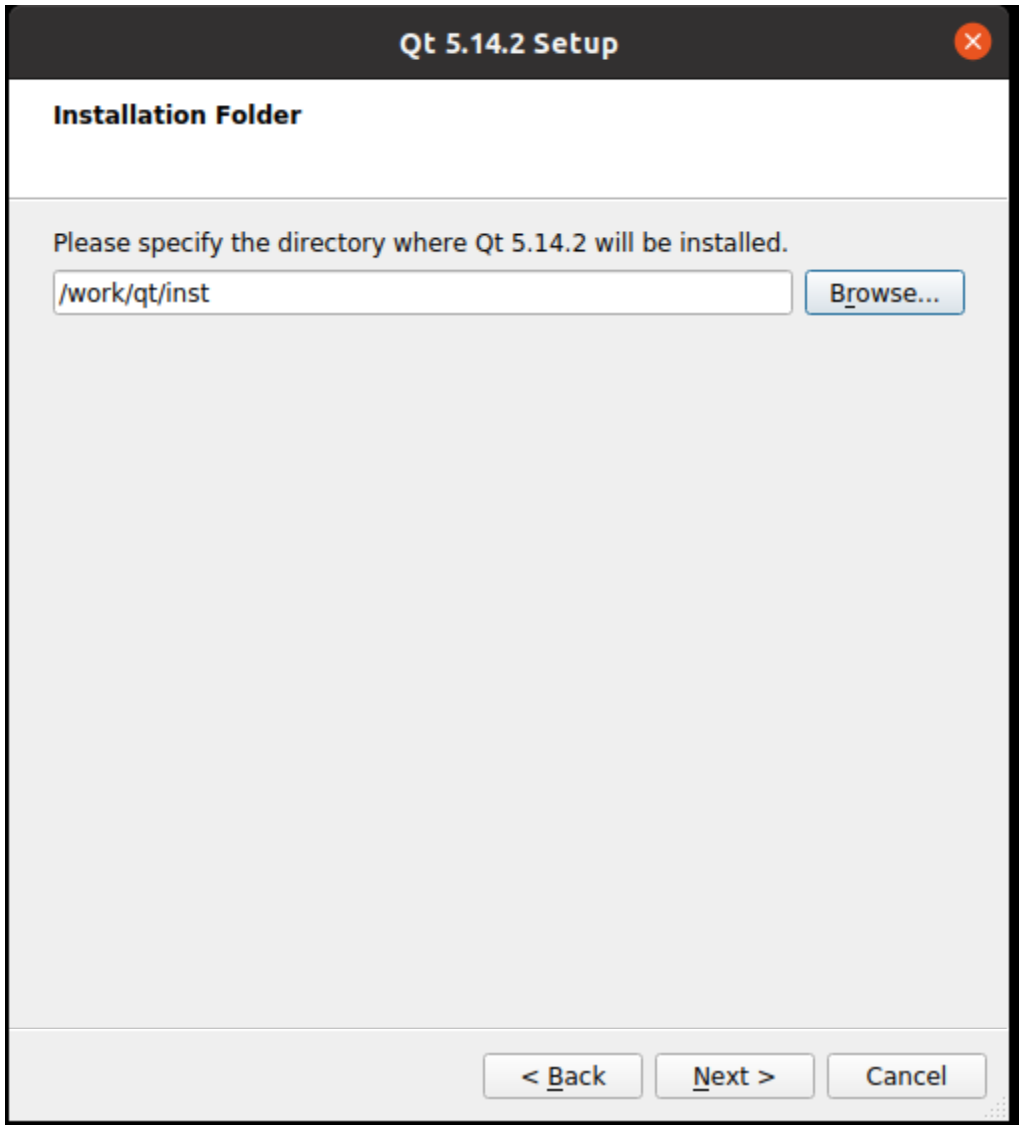
QTオープンソース版はGPLv2、GPLv3、LGPLv3の3つのライセンス形態があります。
これらの規約を受け入れなければ使用できません。
規約を受け入れる場合は「I have read...」の部分にチェックを入れてNextをクリックします。



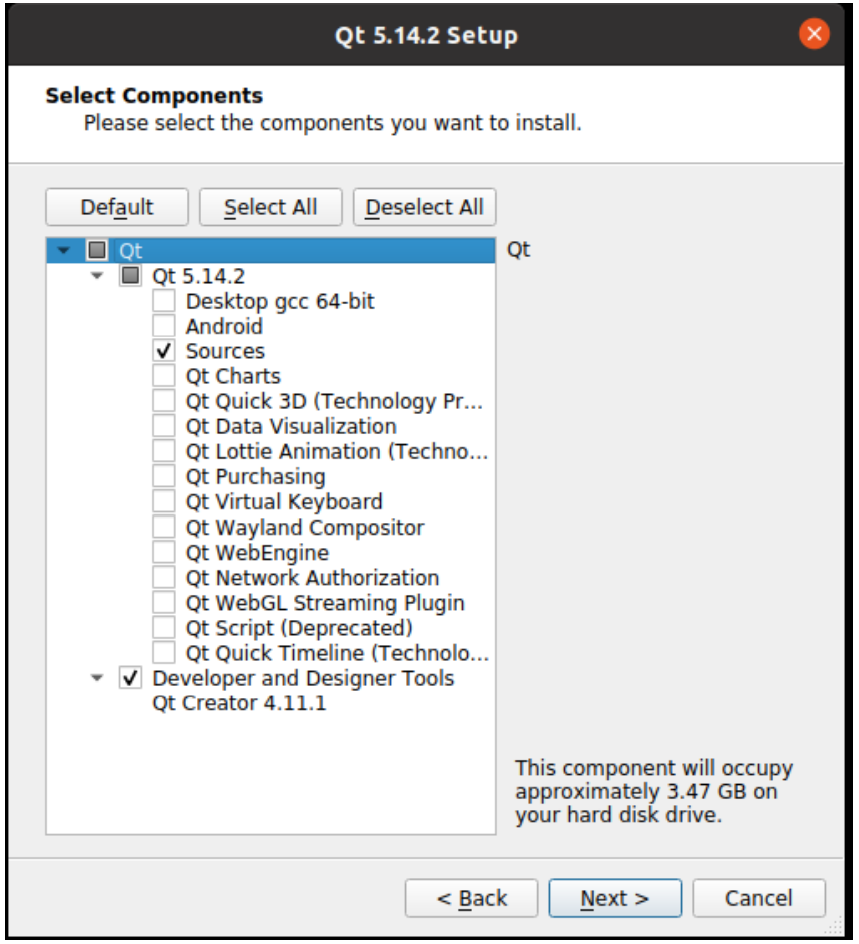
Nextをクリックします。



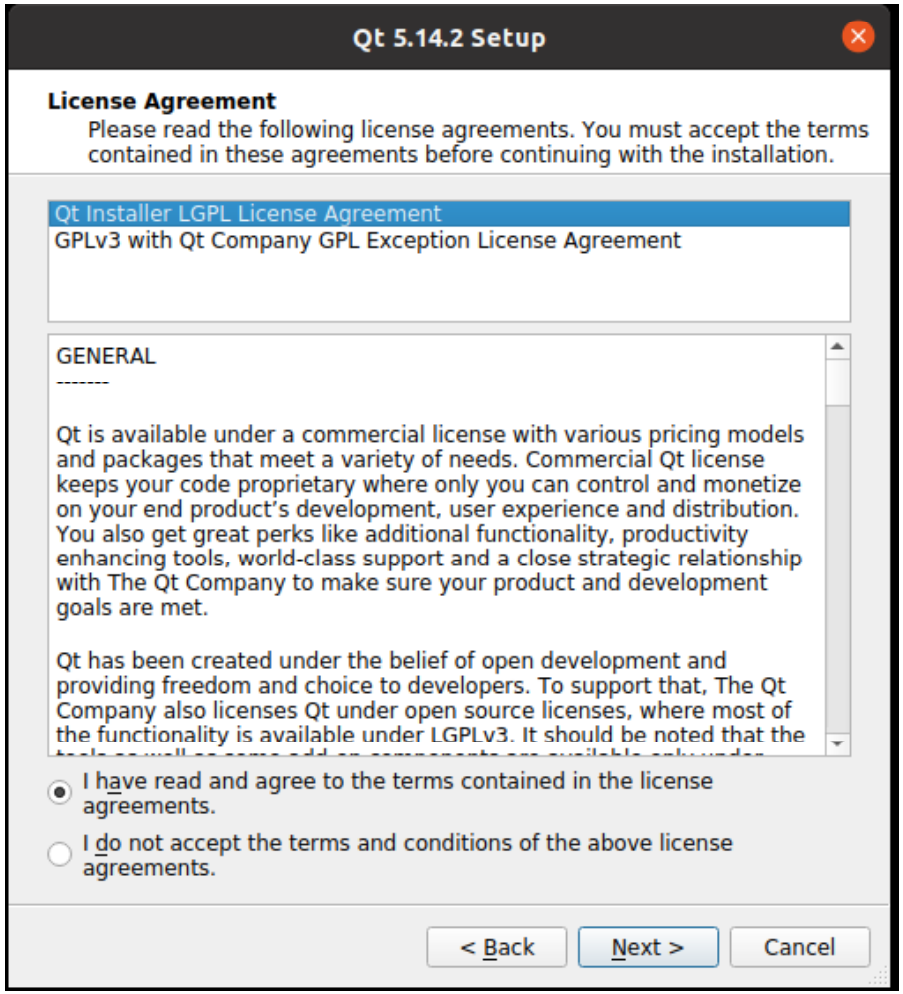
QTのインストールディレクトリを入力します。本マニュアルでは/work/qt/instです。入力後Nextをクリックします。



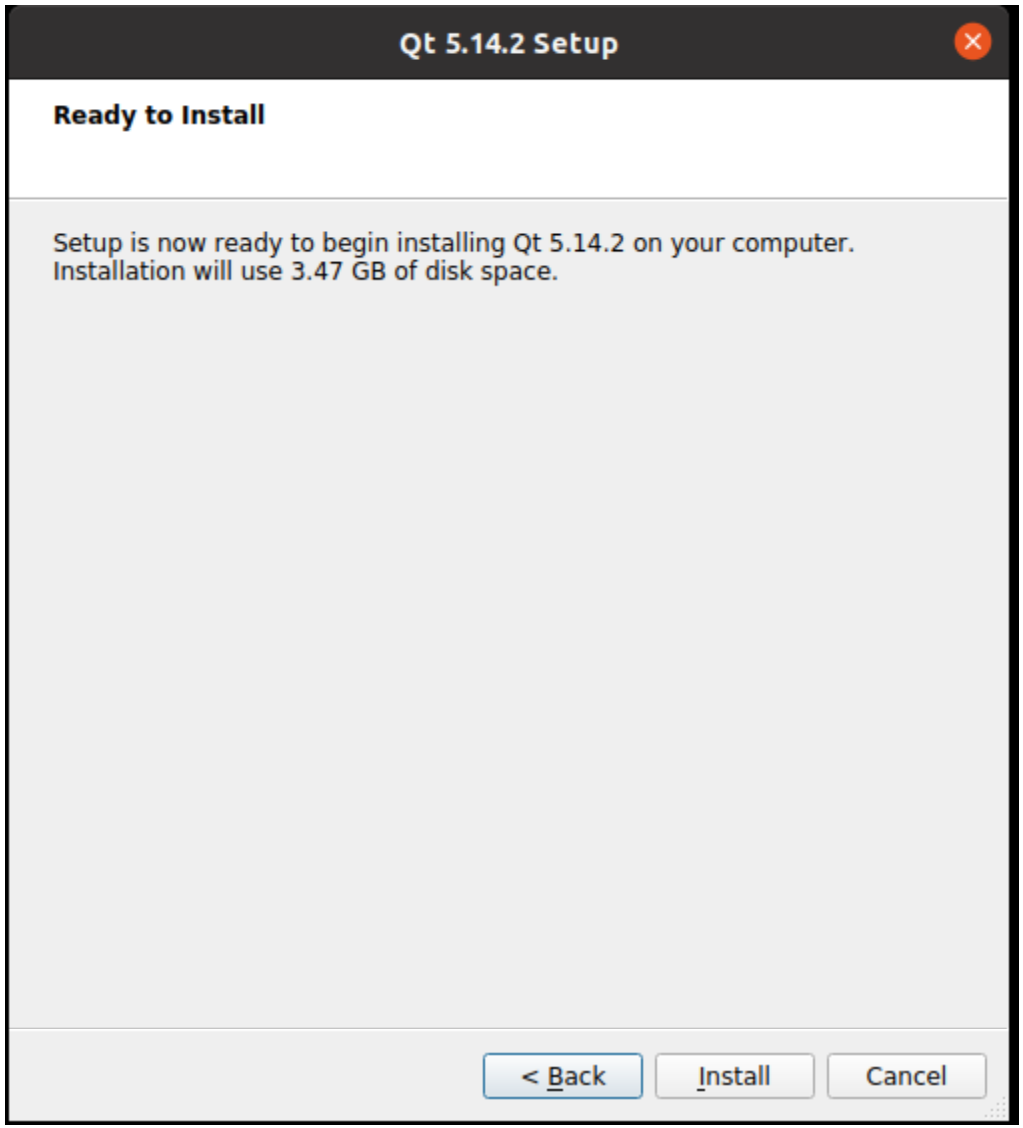
Qt Creatorにはデフォルトでチェックが入っています。
QT5.14.2のSourcesにチェックを入れてNextをクリックします。
Sourcesをインストールするのはサンプルコードを使用するためです。不要であればQTCreatorだけで問題ありません。



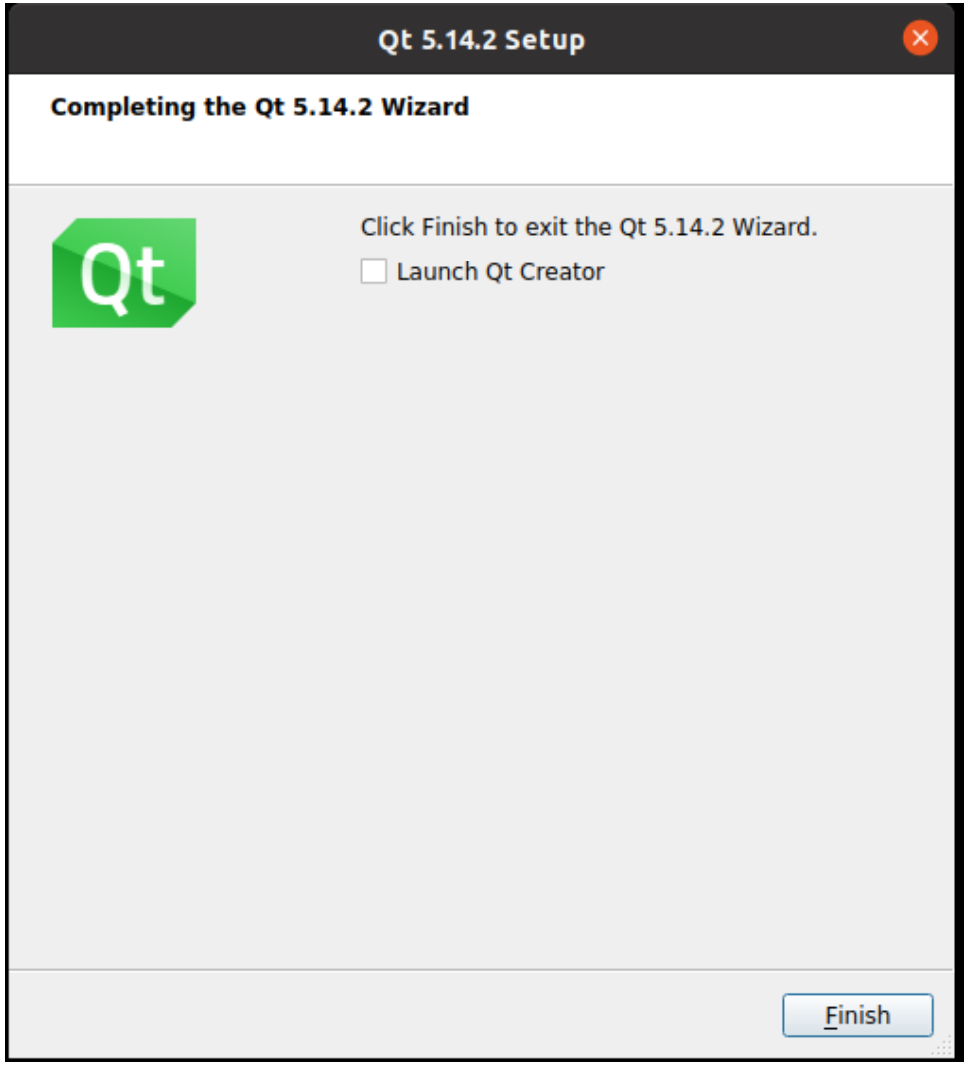
インストールするものによって必要な同意する規約の説明がでてきます。
ライセンス規約に同意できる場合は「I have read ...」にチェックを入れてNextをクリックします。
同意できない場合は使用できません。



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。**

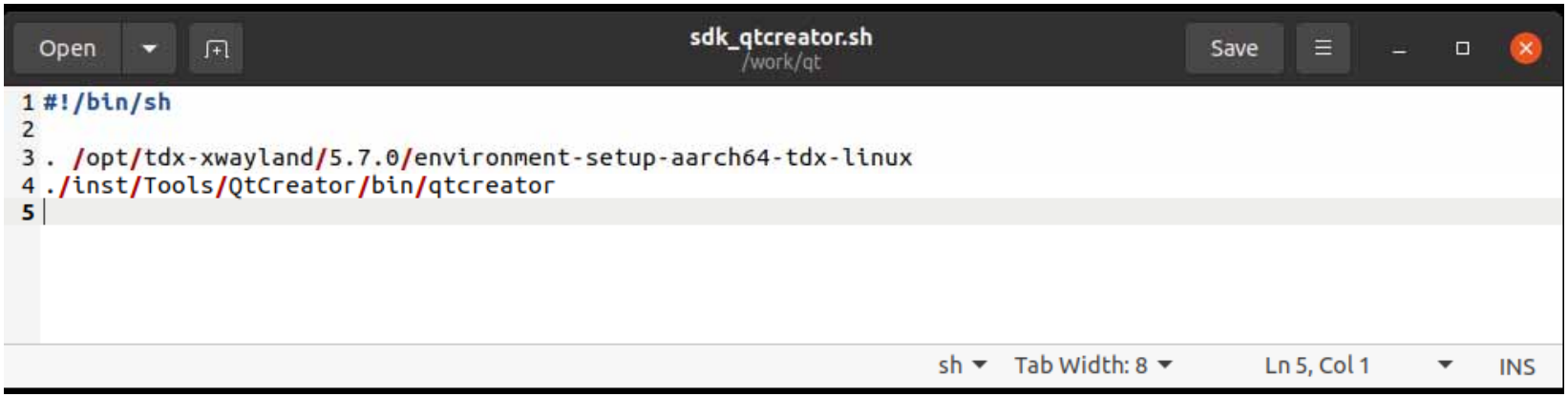


QT-Creator実行シェル作成

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-xwayland/5.7.0/environment-setup-aarch64-tdx-linux
./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

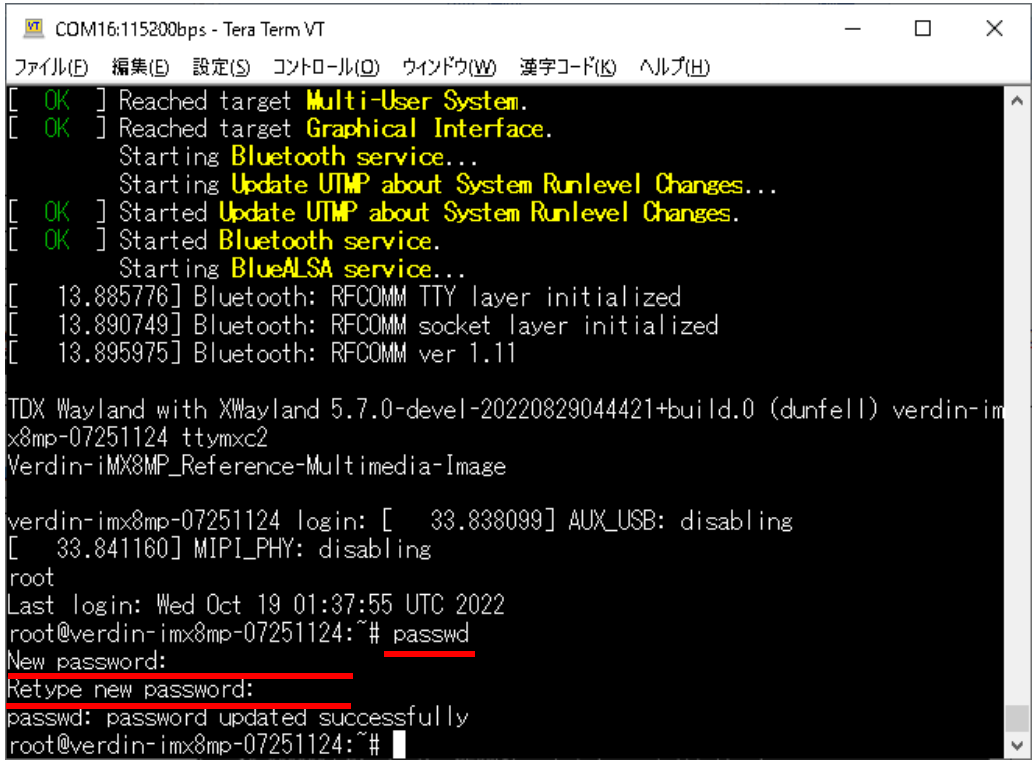
QT Creator実行
[Ubuntu]$ ./sdk_qtcreator.sh

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。
デバッグコンソールを使ってコマンドを入力します。
Teraterm(Ubuntuの場合minicomなど)を起動してから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。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)
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。
[Module]# passwd



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。

[Module]# 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。Verdin-iMX8M Plusはeth0とeth1があります。
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

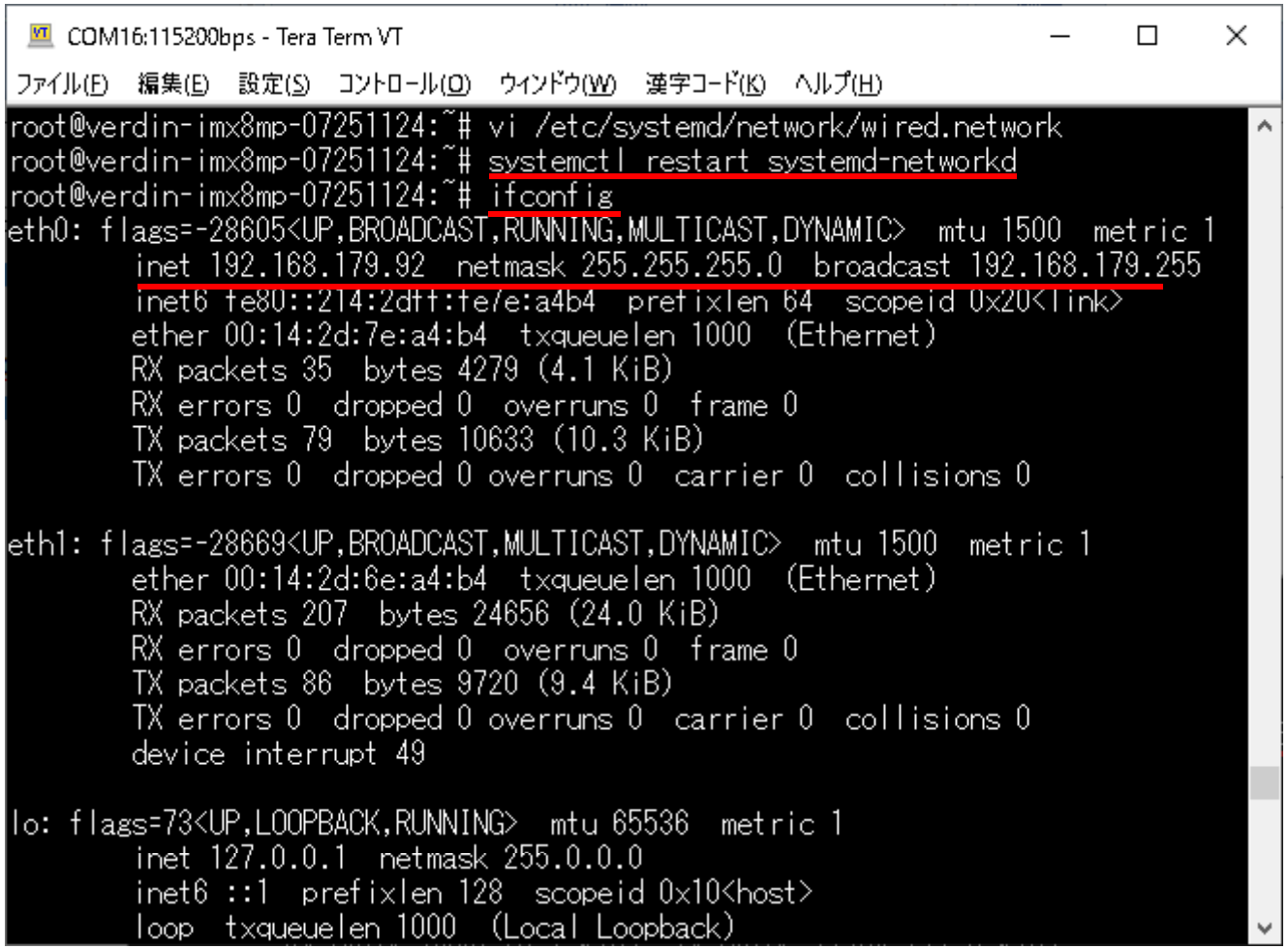
[Match]
Name=eth0

[Network]
Address=192.168.179.92/24
Gateway=192.168.179.1

192.168.179.92/24はIPアドレス192.168.179.92 サブネットマスク255.255.255.0を意味します。
IPアドレスは環境に応じて設定してください。

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。
[Module]# systemctl restart systemd-networkd

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

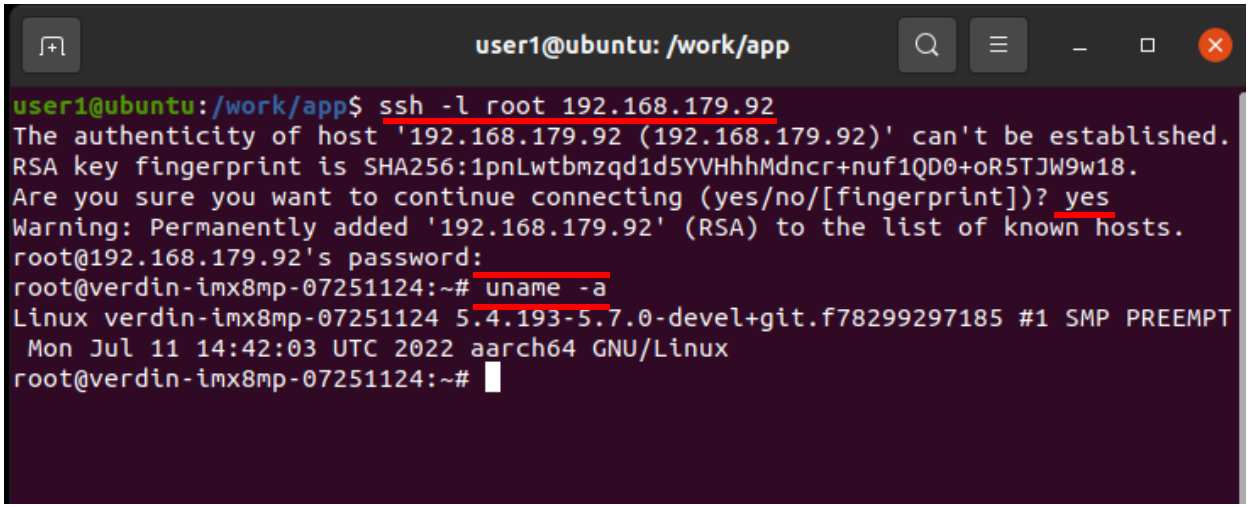


SSH接続確認

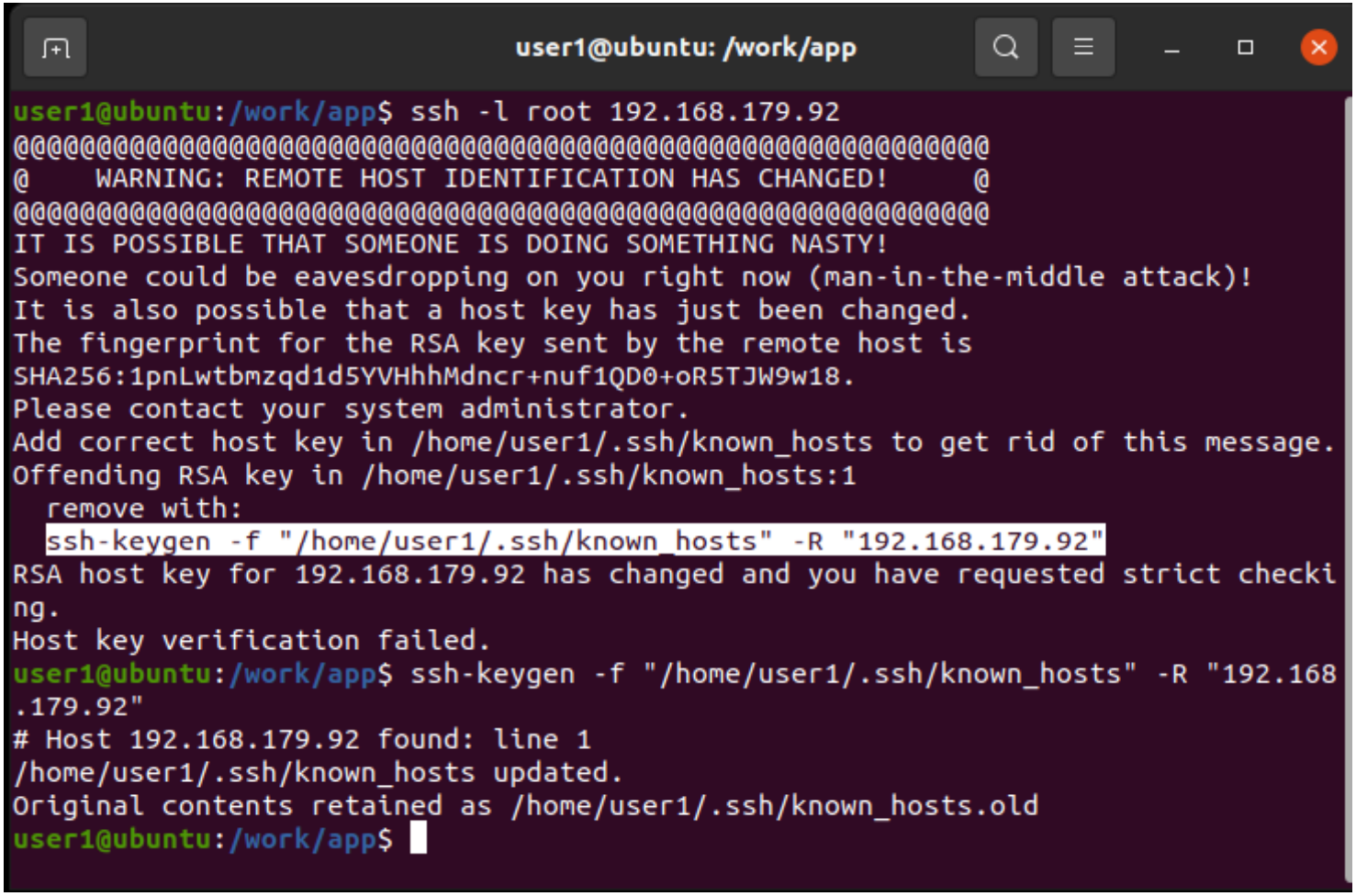
開発パソコンからpingで接続できているか確認します。
接続できない場合は何かしらの設定が間違っている可能性があります。
[Ubuntu]$ ping XXX.XXX.XXX.XXX

sshコマンドで接続できるか試します。
[Ubuntu]$ ssh -l root XXX.XXX.XXX.XXX

最後にunameコマンドでログインできているかを確かめています。
[Module]# uname -a

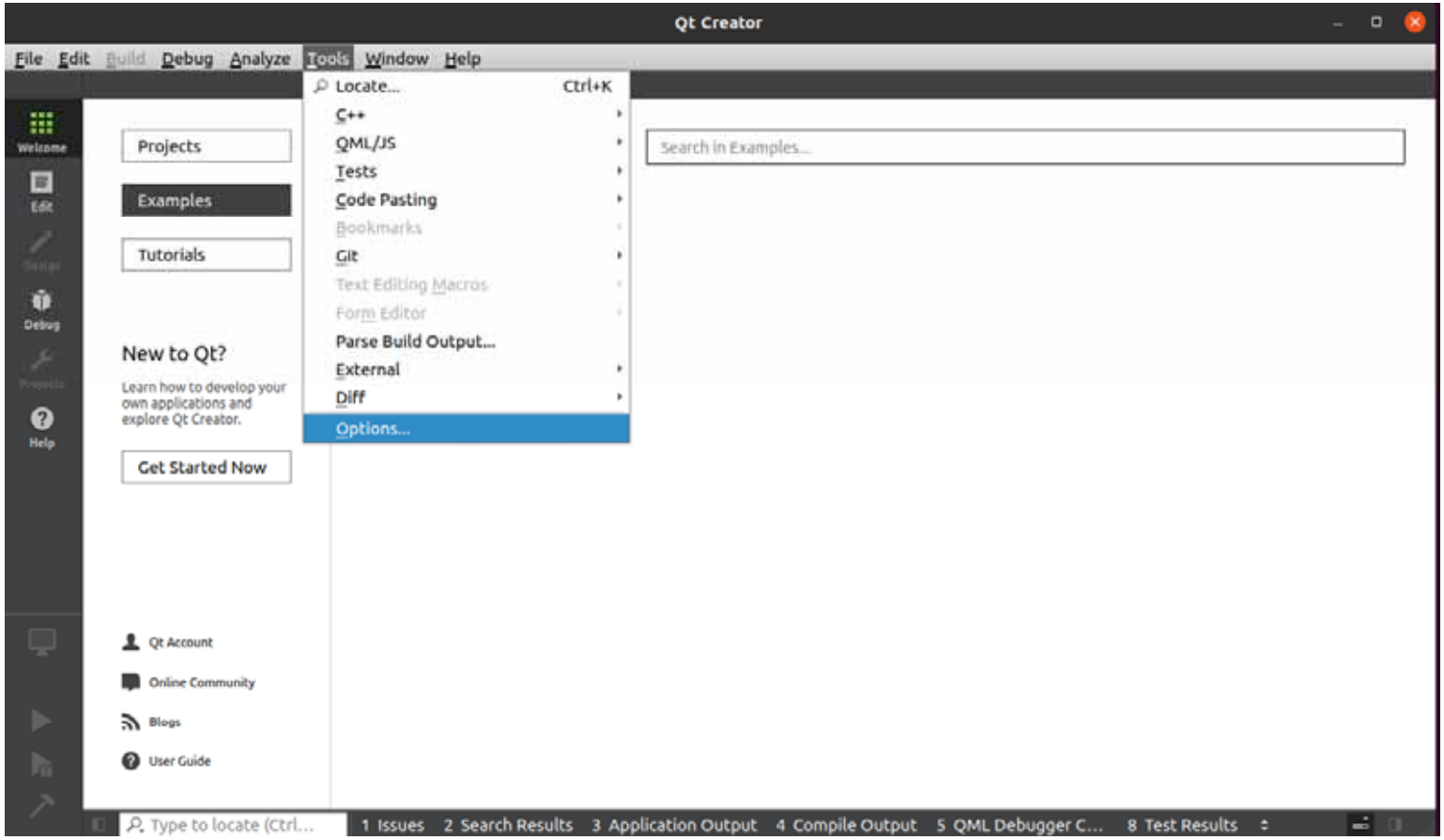


sshコマンドで下記のようなエラーが出た場合
ssh -keygenでキーを一度削除してください。

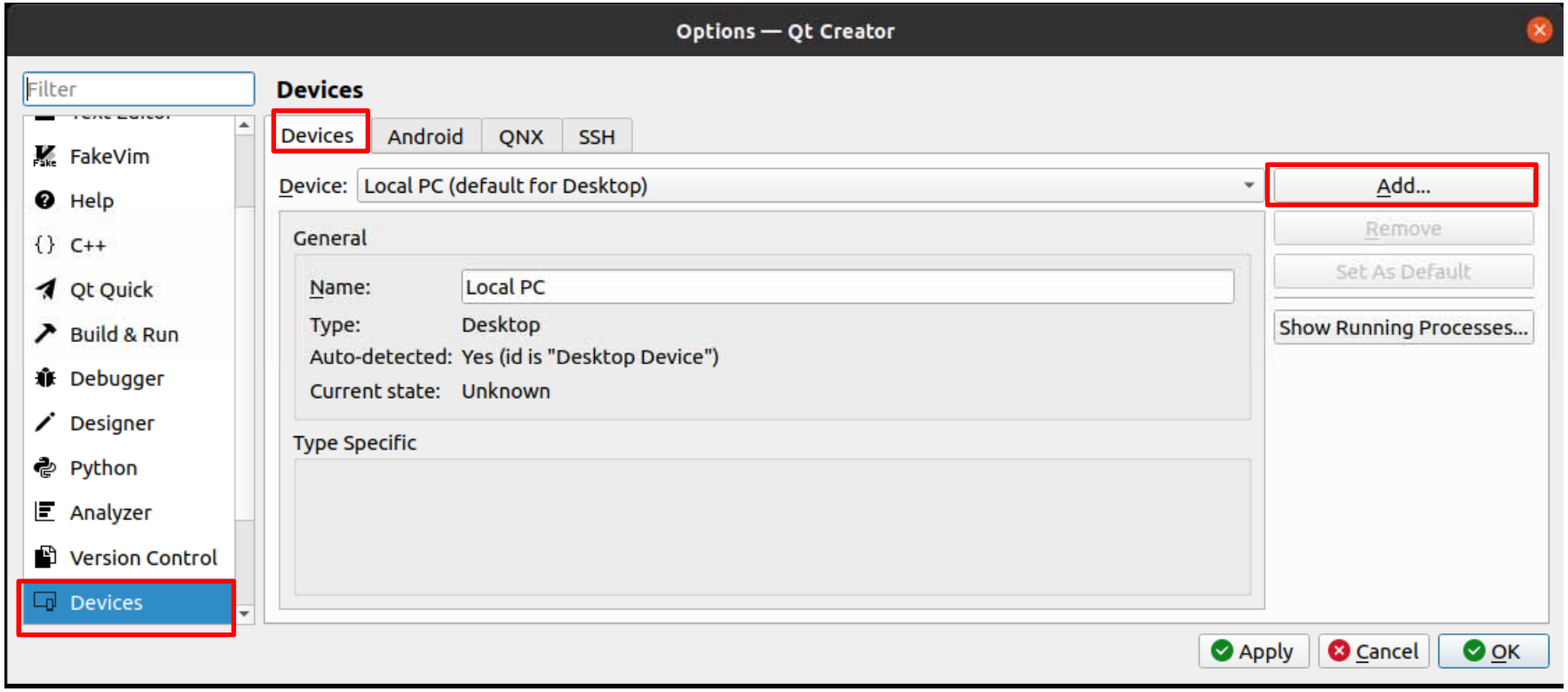


QT Creatorの設定

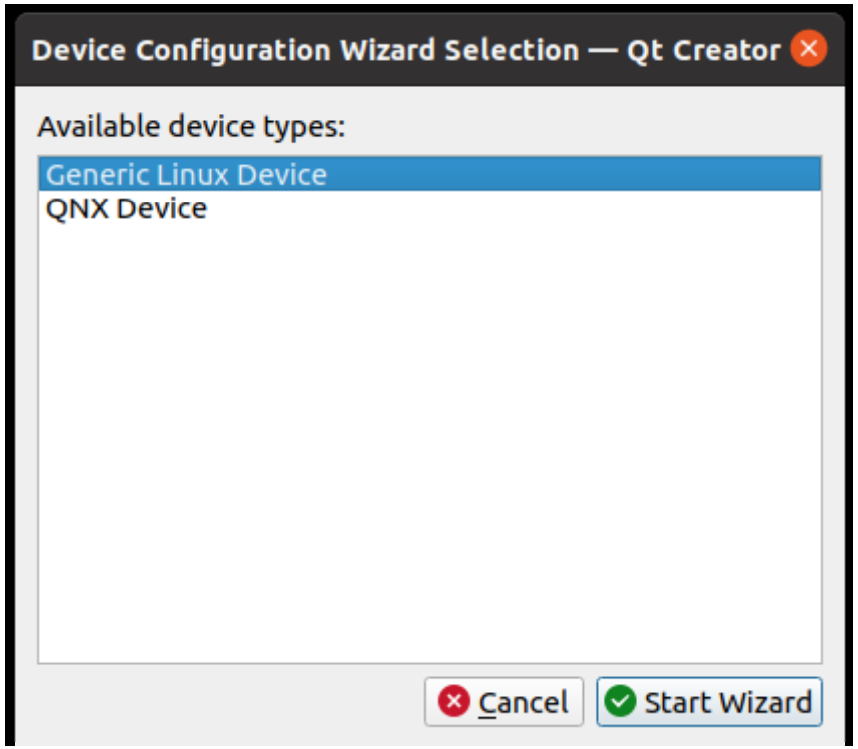
QT Creatorの設定を行います。Tools > Optionsを開きます。



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。

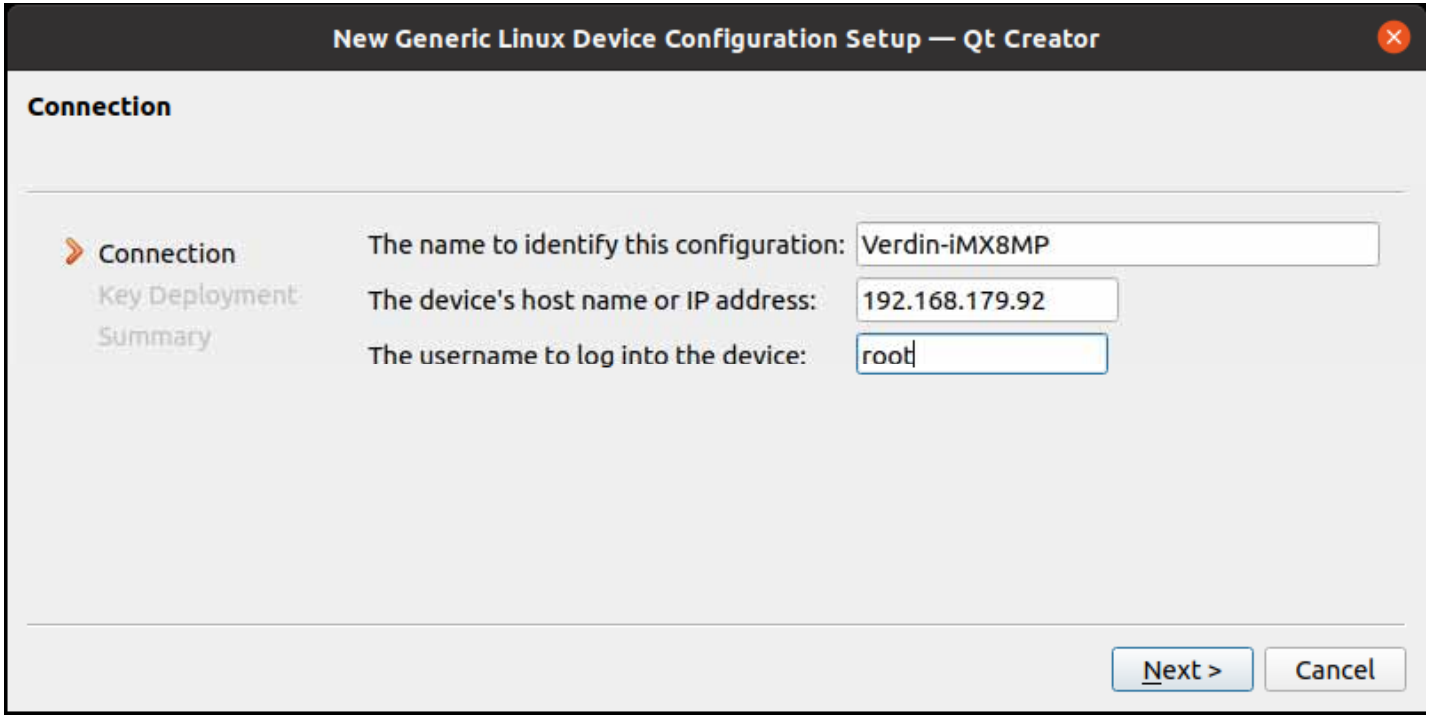


Generic Linux Deviceを選択してStart Wizardをクリックします。

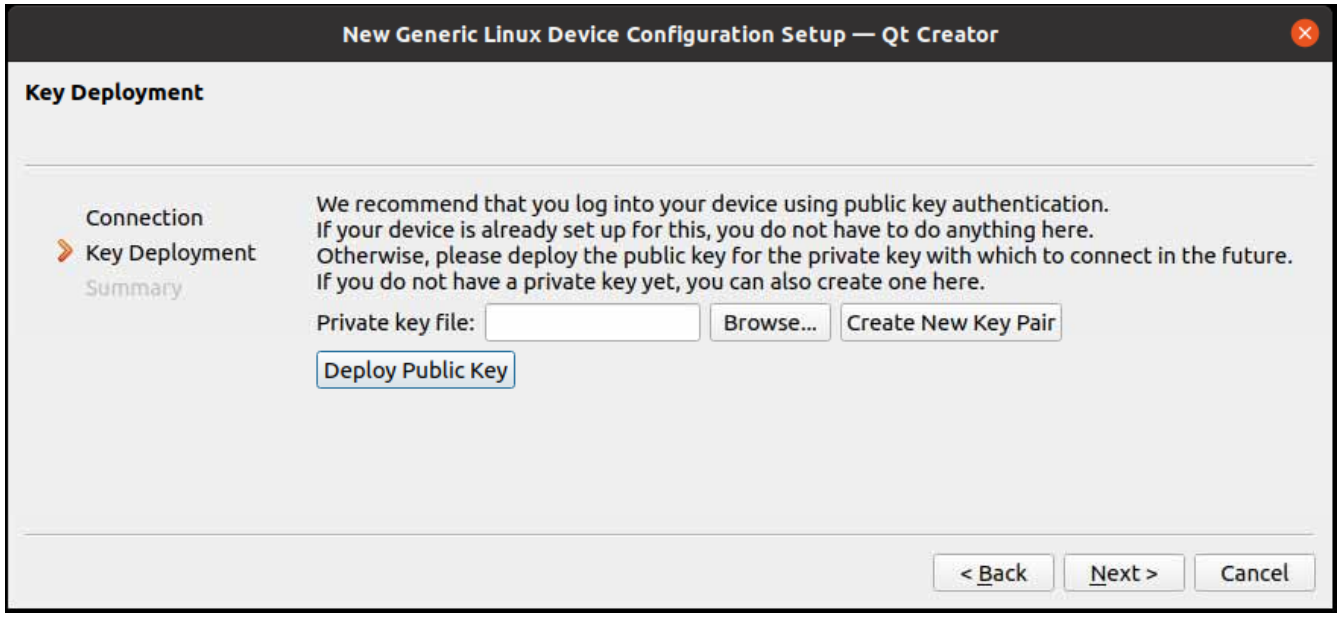


SSHで接続するモジュールの情報を入力します。本マニュアルでは下記の内容を設定しています。
接続名(わかりやすい名前をつけてください。)：Verdin-iMX8MP
モジュールに設定したIPアドレス：192.168.179.92
ログインユーザー名：root

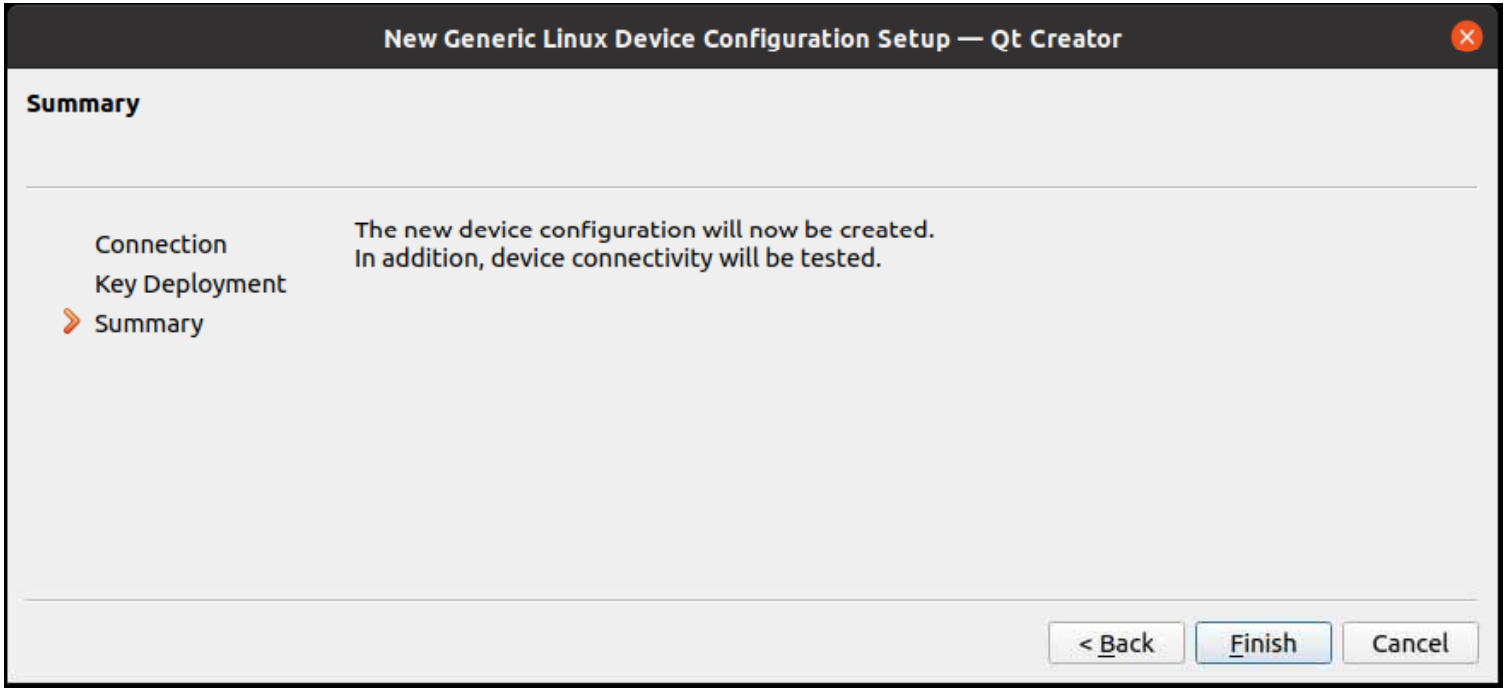
入力したらNextをクリックします。



SSHで認証キーを使うことを推奨されますが本マニュアルでは使用しません。パスワードのみ使用します。
Nextをクリックします。

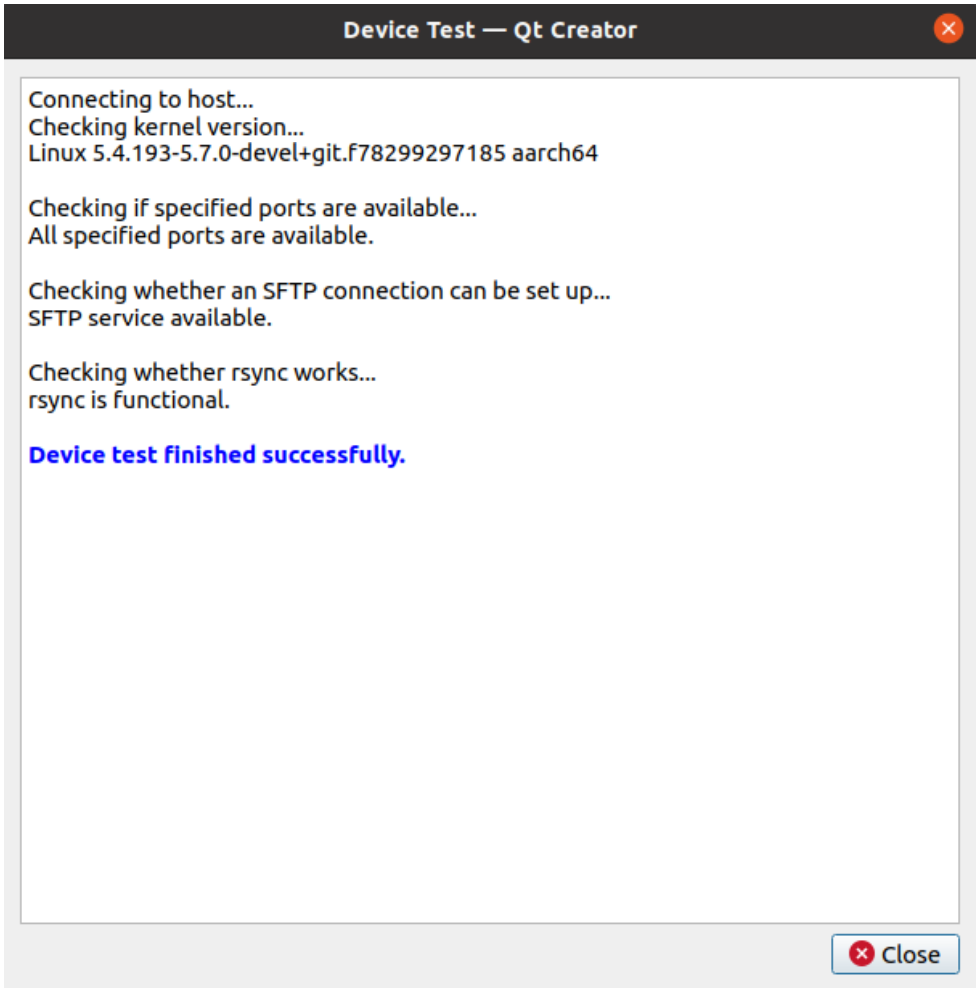


Finishをクリックします。

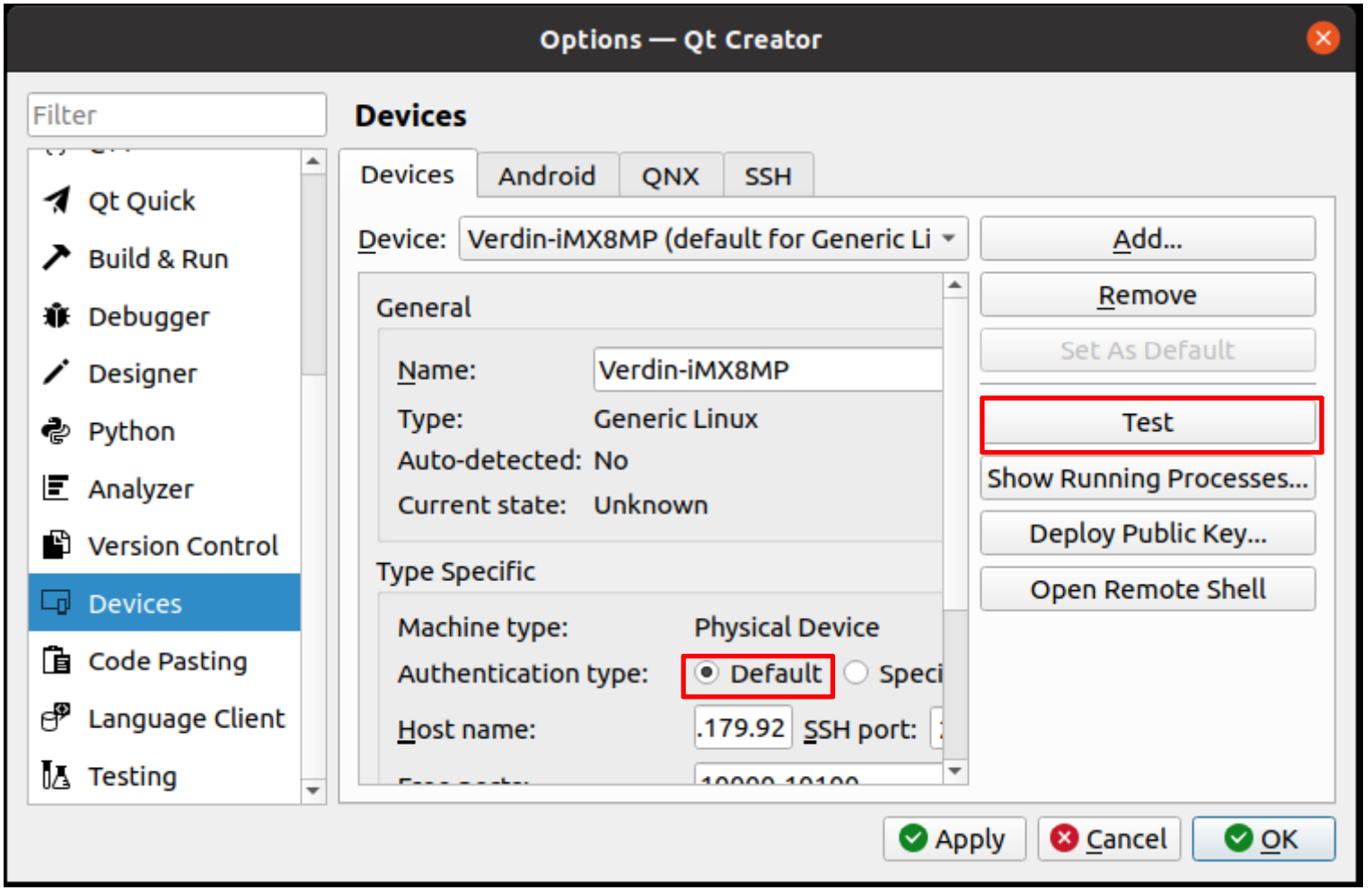


SSHの接続確認が行われます。成功すると下記のような画面になります。Closeをクリックします。

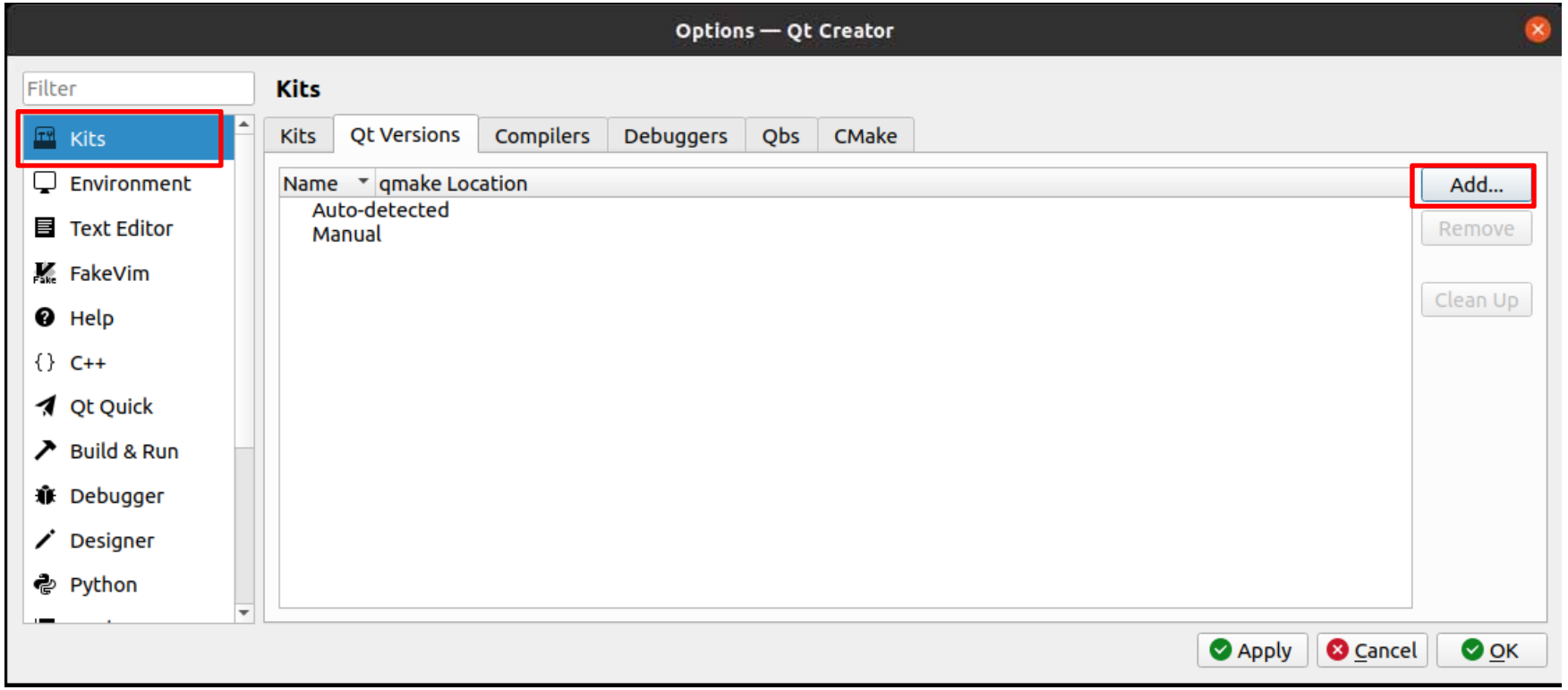
SSHの接続は~/.ssh/known_hostsのキーを使用します。
ホストのキーが保存された後にモジュールのOSを書き換えた場合、再度接続するとホストのキーが違うというエラーが発生します。
その場合ssh-keygen -f "~/.ssh/known_hosts" -R "192.168.179.92"で一旦古いキーを削除してください。



接続失敗後などに再度接続を試す場合はAuthentication type**を**Default**にしてかた**Test**ボタンをクリックしてください。**



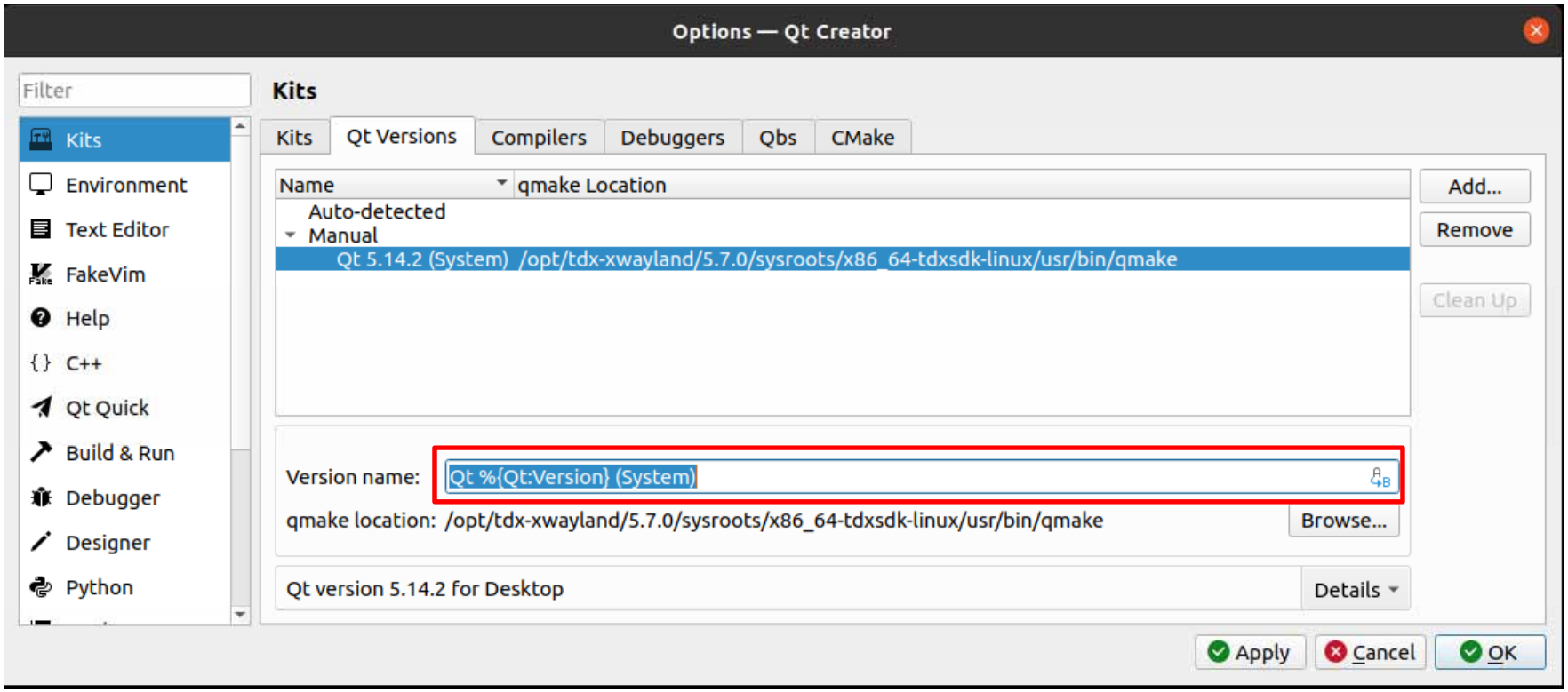
Optionsに戻ります。左の欄からKitsを選択しQT Versionsタブを選択します。Addをクリックします。
最初からqmakeが存在している場合はこの手順は不要です。



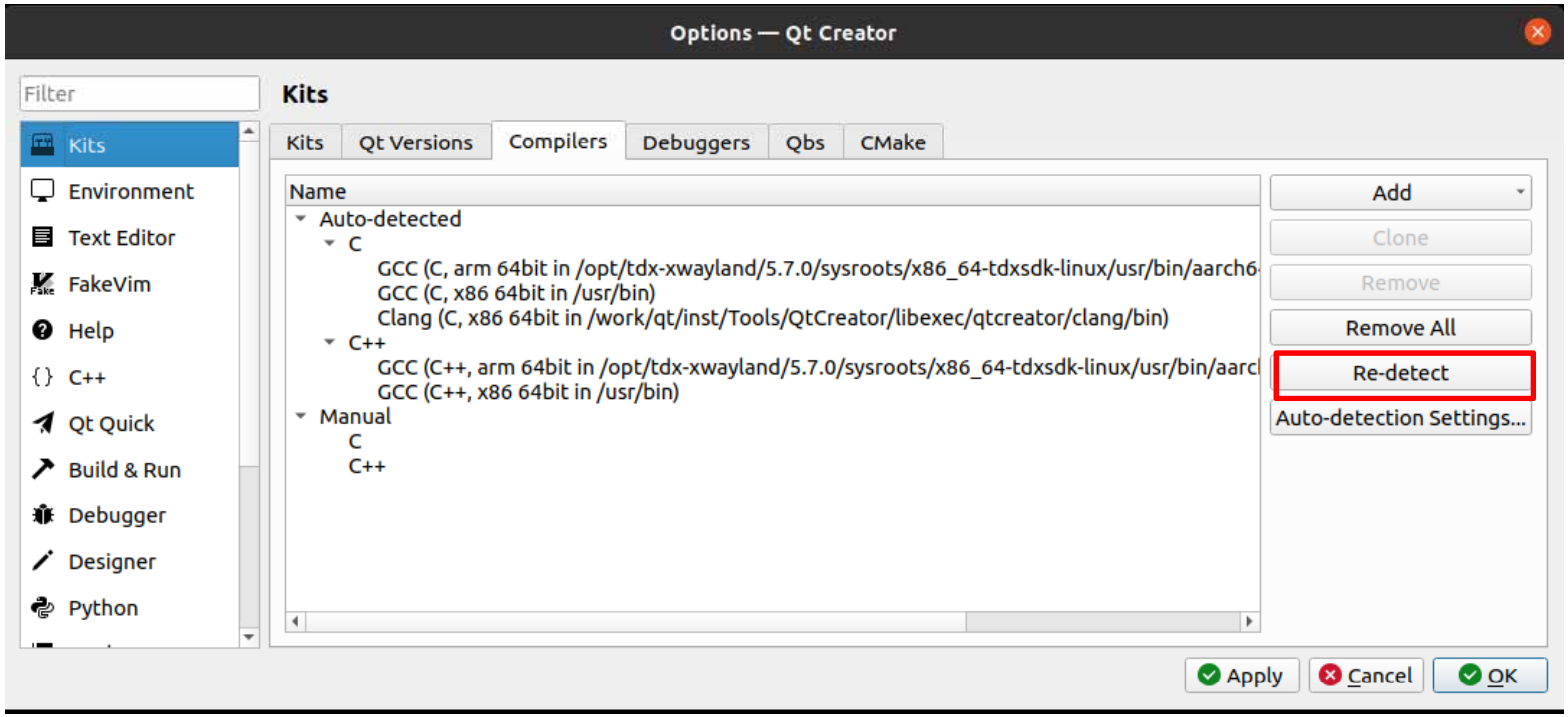
qmakeのパスはデフォルト設定の場合は下記になります。
/opt/tdx-xwayland/5.7.0/sysroots/x86_64-tdxsdk-linux/usr/bin/qmake

ダブルクリックでqmakeのパスを選択します。
(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。)

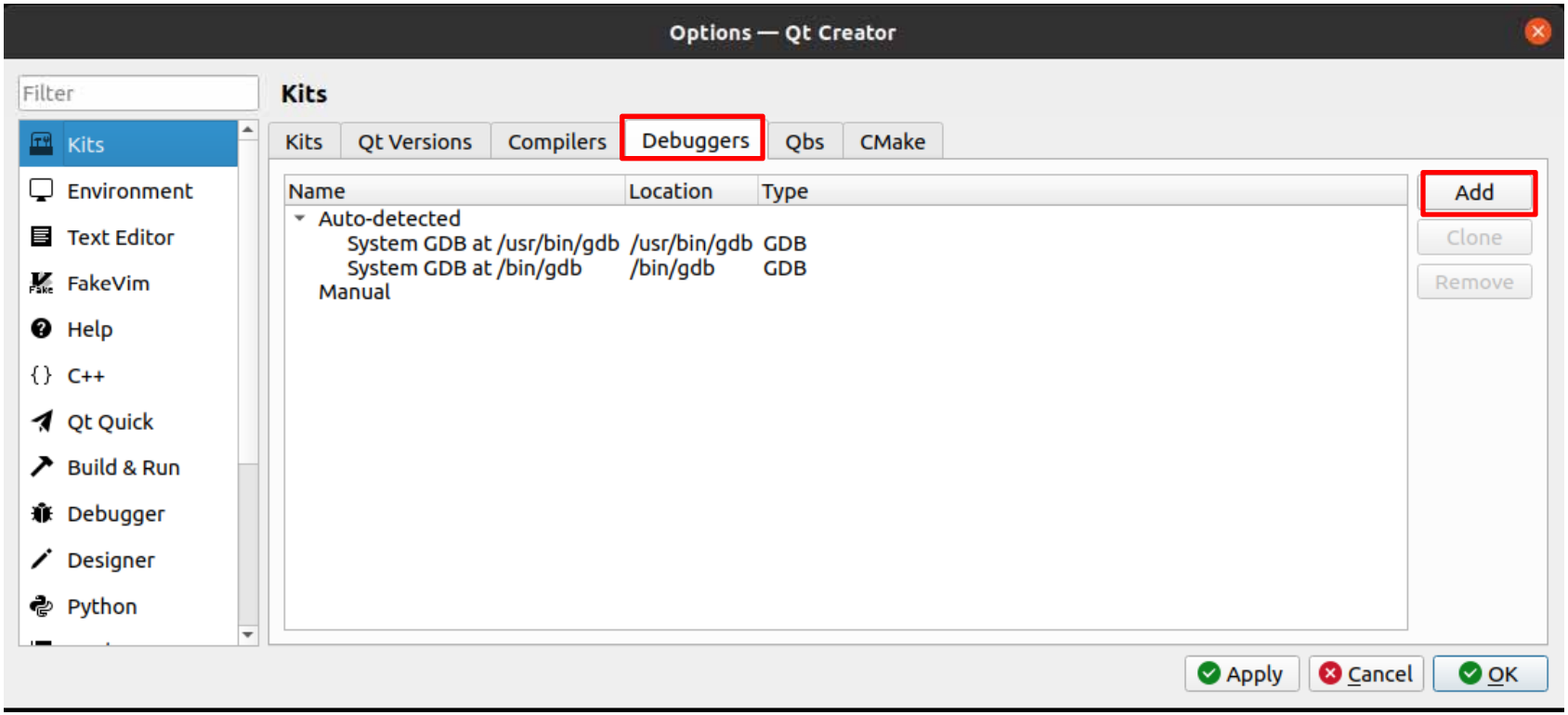
下記のようになります。Version Name**は自動的に作成されます。**



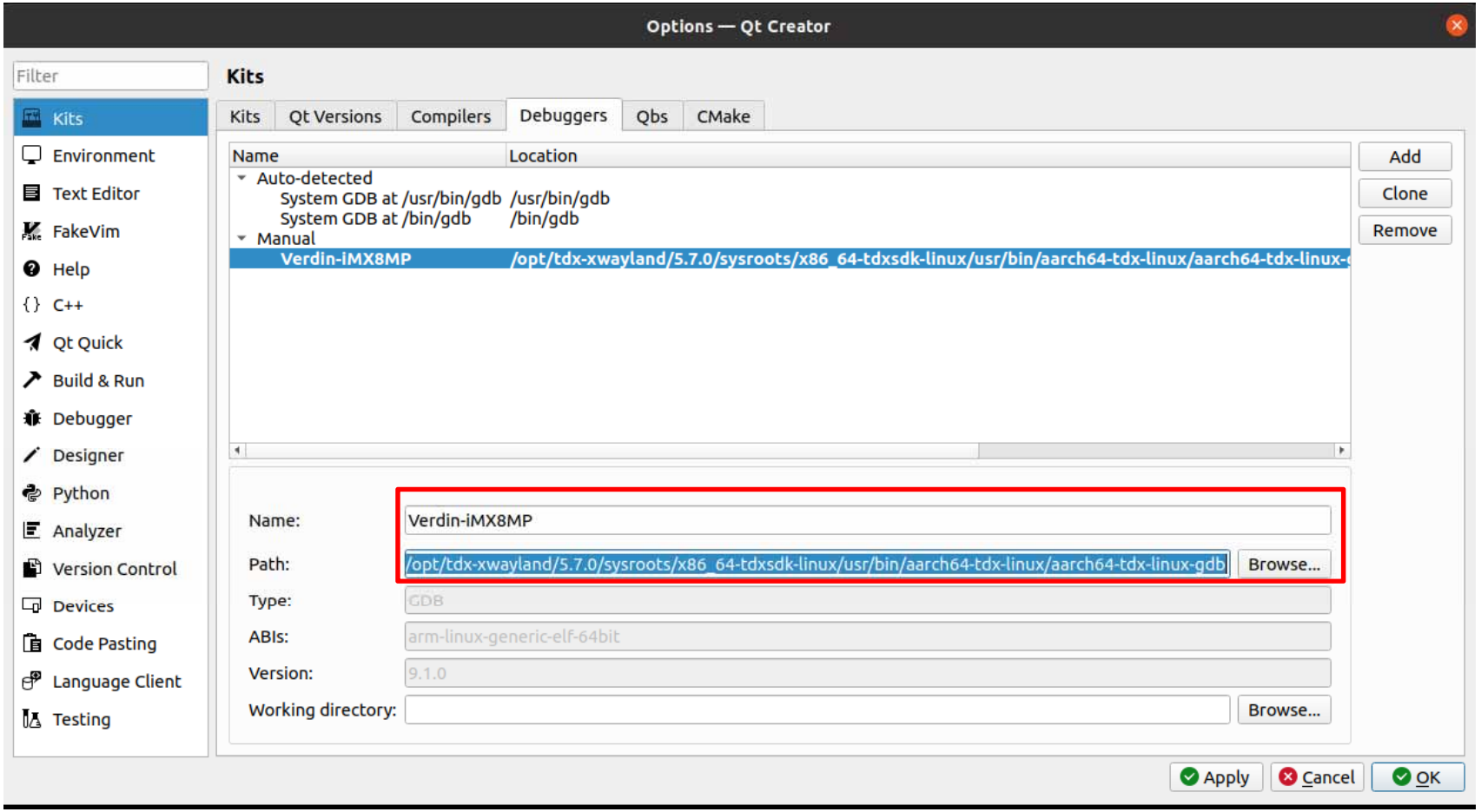
コンパイラは自動で設定されていますがもし消してしまったりした場合はRe-detectボタンをクリックして再検知してください。



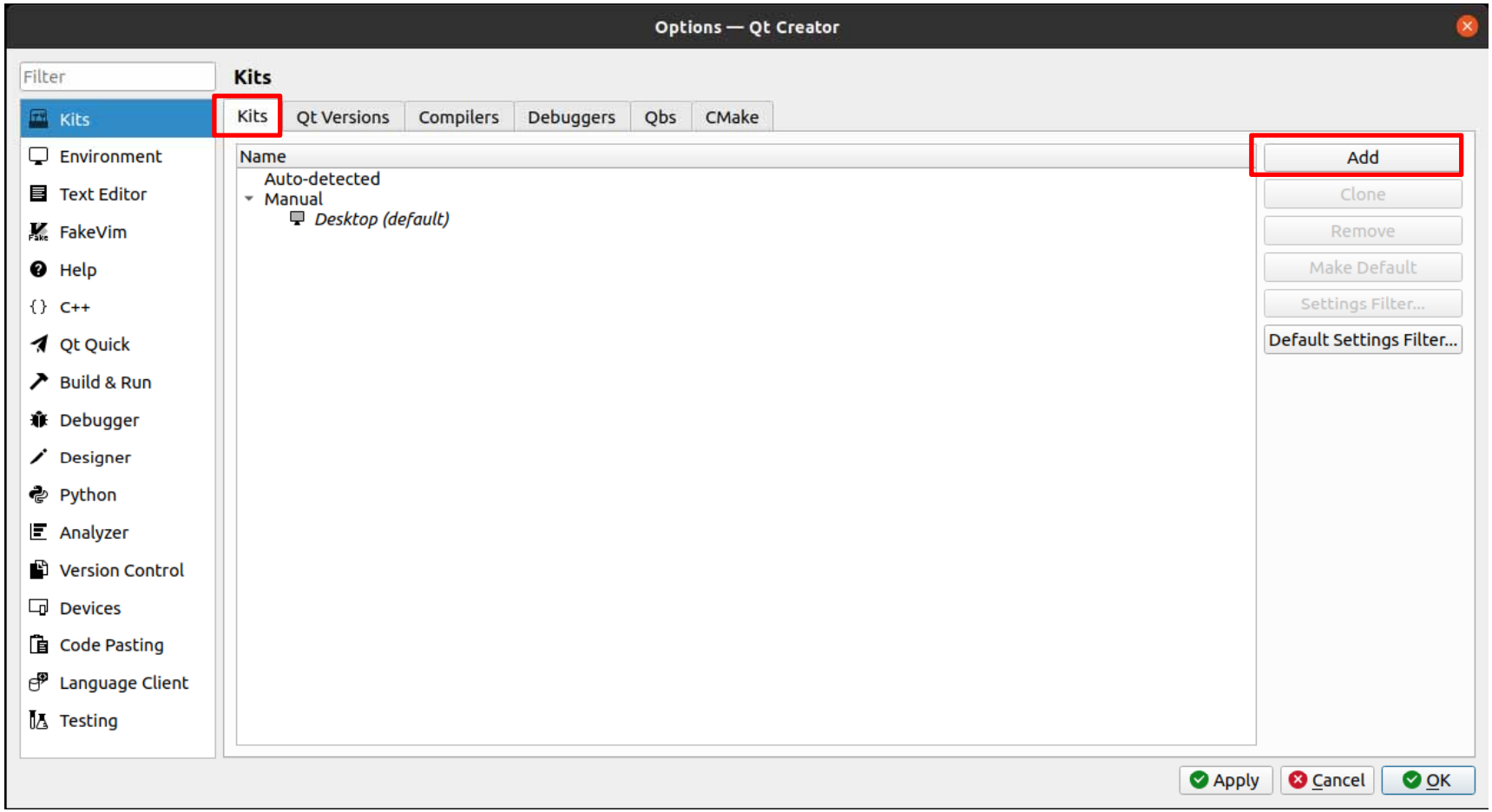
Debuggersタブを選択しAddをクリックします。



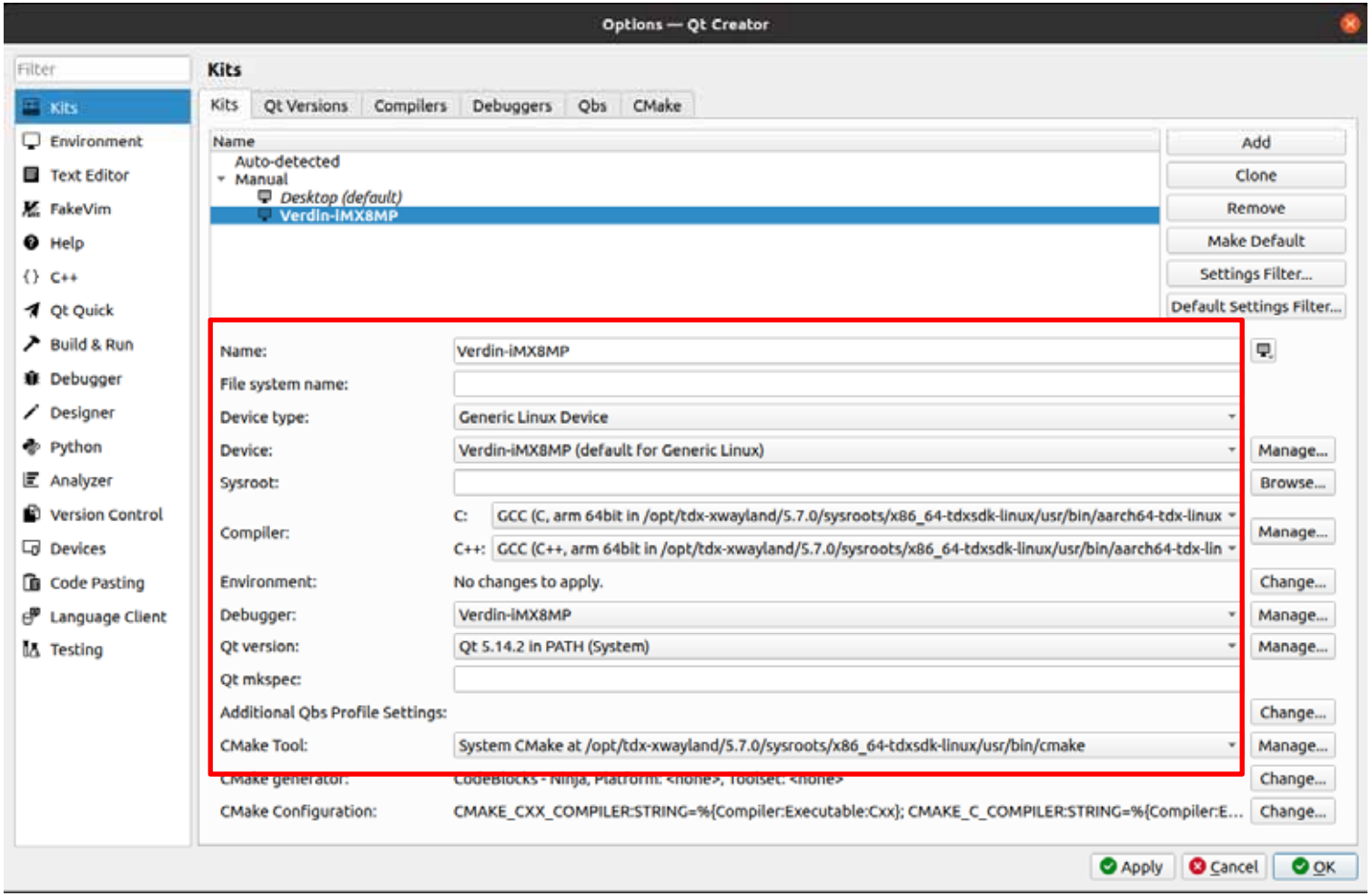
Nameにデバッガーの設定の名前を入力します。本マニュアルではVerdin-iMX8MPとしています。
Pathに下記を入力します。
デフォルト設定の場合は下記になります。
/opt/tdx-xwayland/5.7.0/sysroots/x86_64-tdxsdk-linux/usr/bin/aarch64-tdx-linux/aarch64-tdx-linux-gdb
(パスはSDK出力先やモジュールやBSPのバージョンなどによって異なります。)



Kitsタブを選択しAddをクリックします。

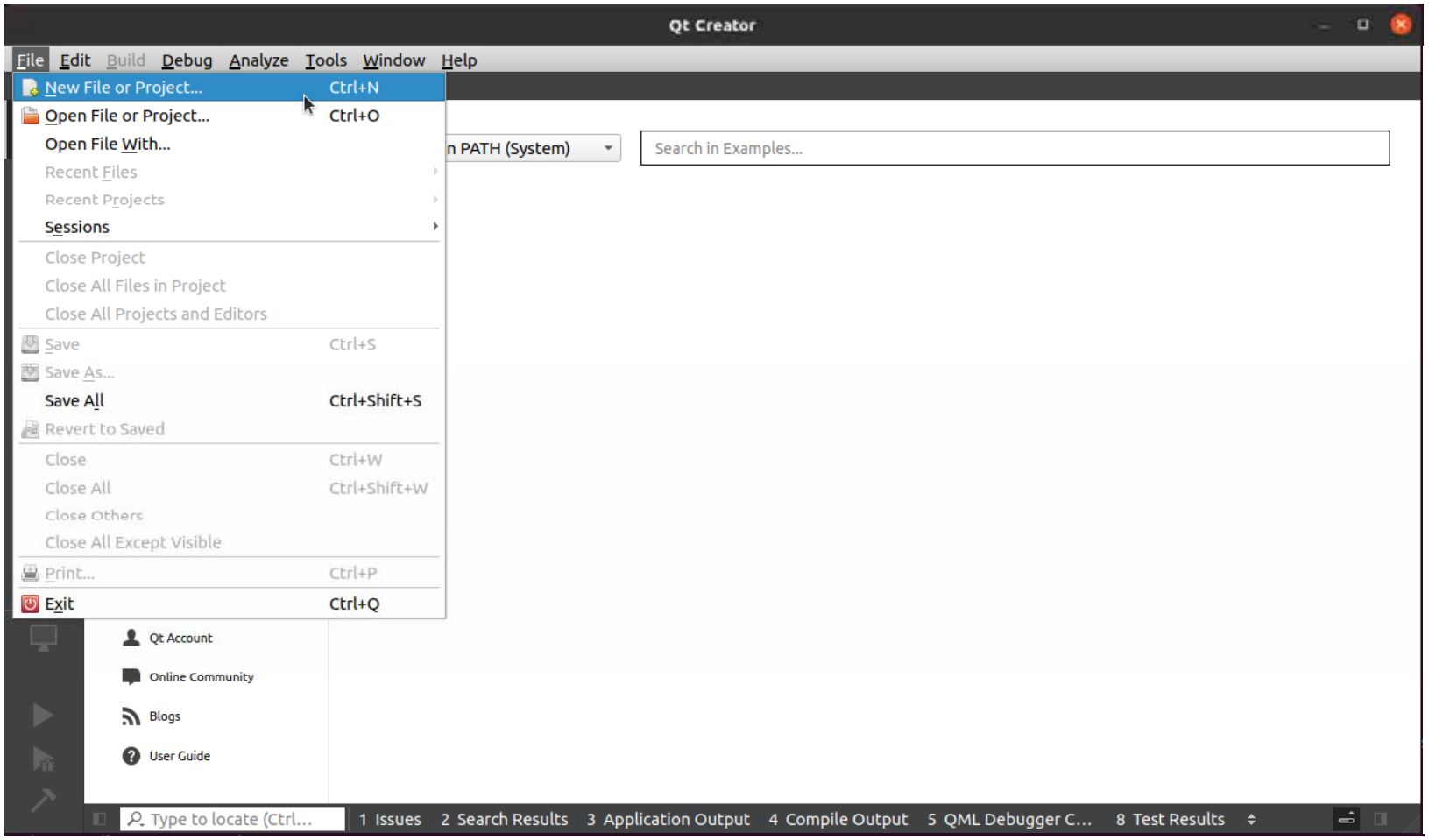


NameにKitsの設定の名前を入力します。本マニュアルではVerdin-iMX8MPとしています。
Device typeにGeneric　Linux Deviceを選択するとDeviceが自動的に選択されます。
QT Version,Compilers,Debuggerに関してはそれぞれ作成した設定を選択します。
CMake Toolは自動で入力されます。
OKをクリックします。

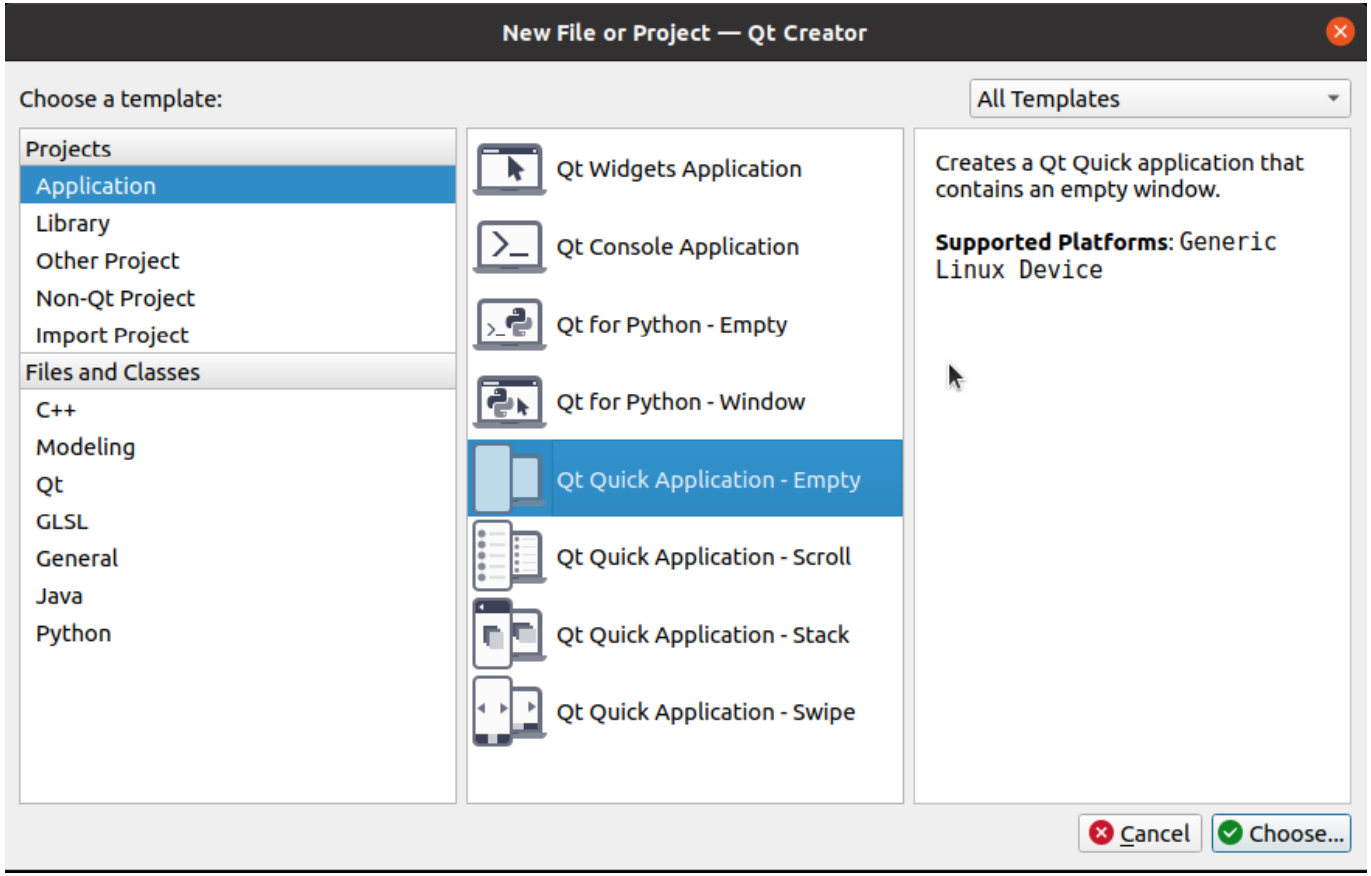


プロジェクト作成からデバッグまで

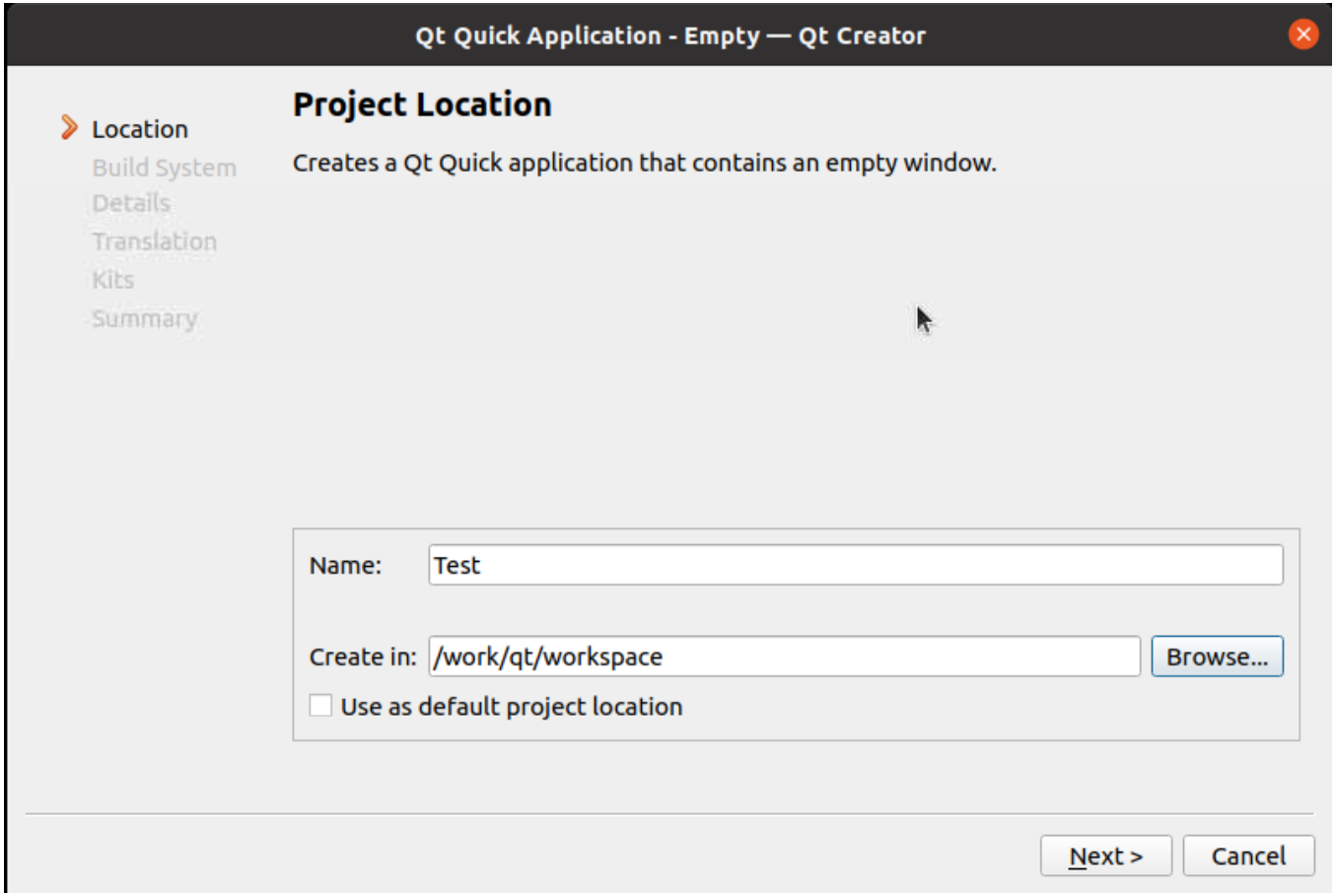
メニューからFile > New File or Projectを選択します。



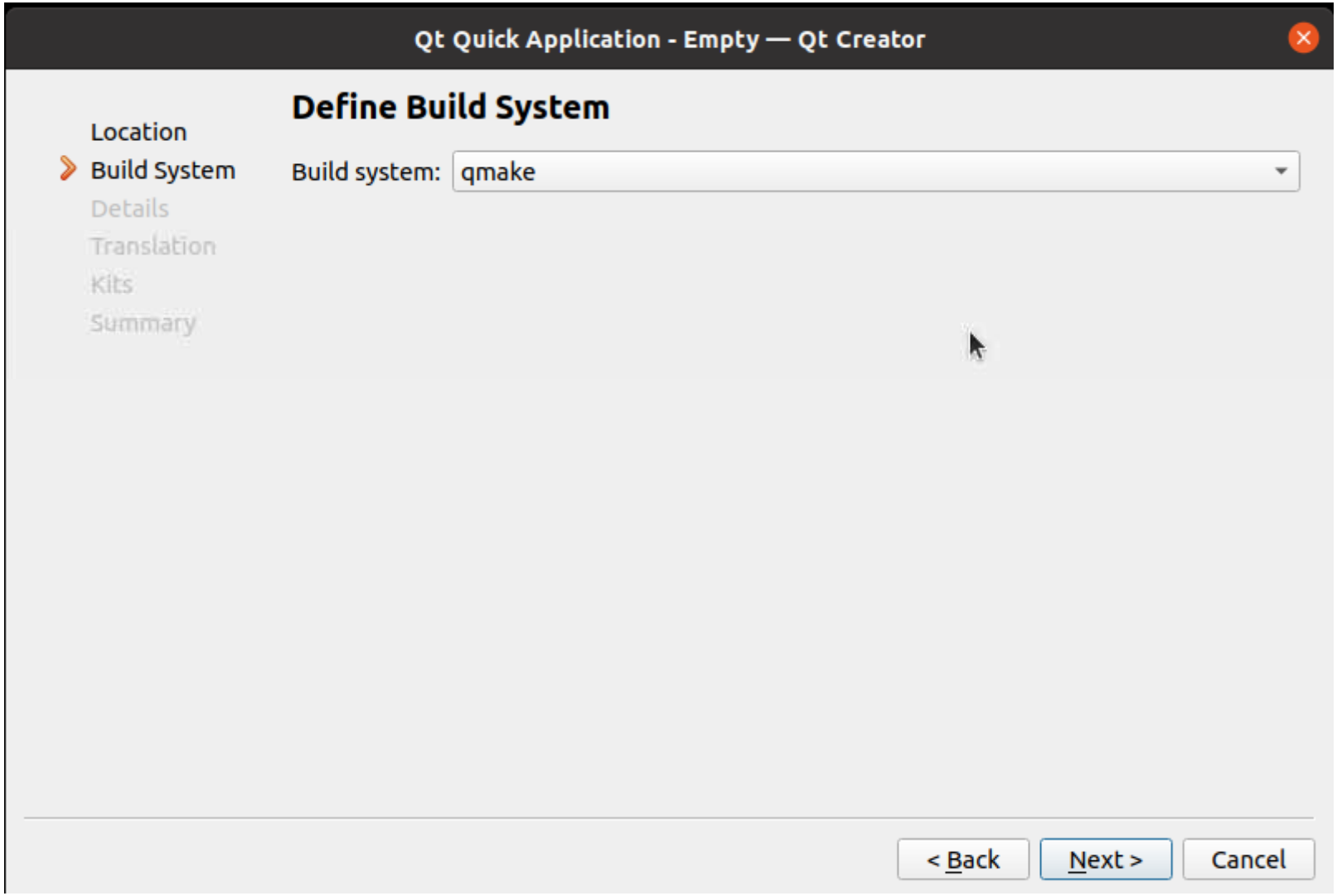
簡単なアプリケーションを作成します。
Projectsの項目はApplicationを選択します。
本マニュアルではQt Quick Application - Emptyを選択します。
Chooseをクリックします。



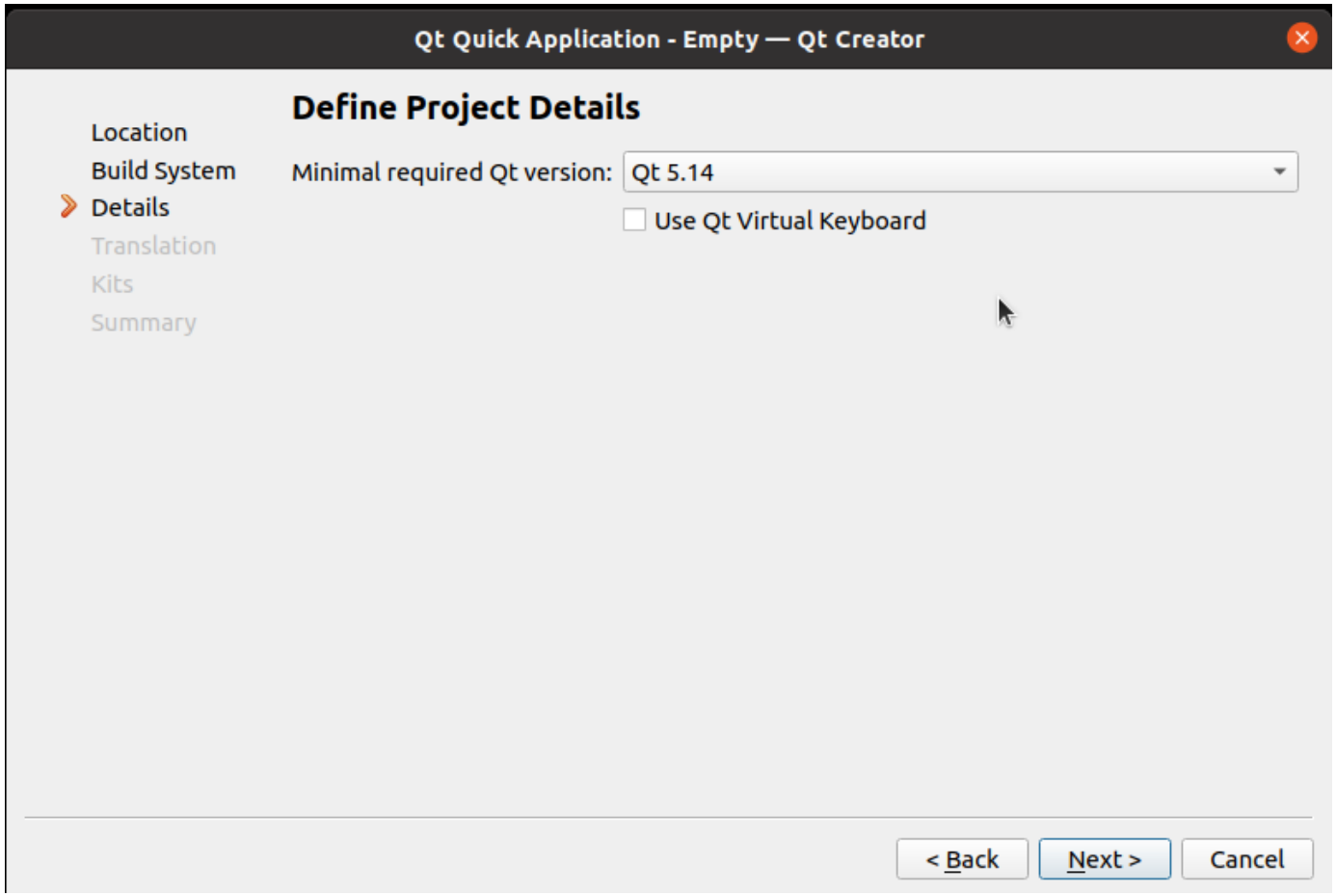
NameにProjectの名前を入力します。本マニュアルではTestとしています。
Projectのパスは事前に作成した/work/qt/配下のworkspaceを入力します。
Nextをクリックします。



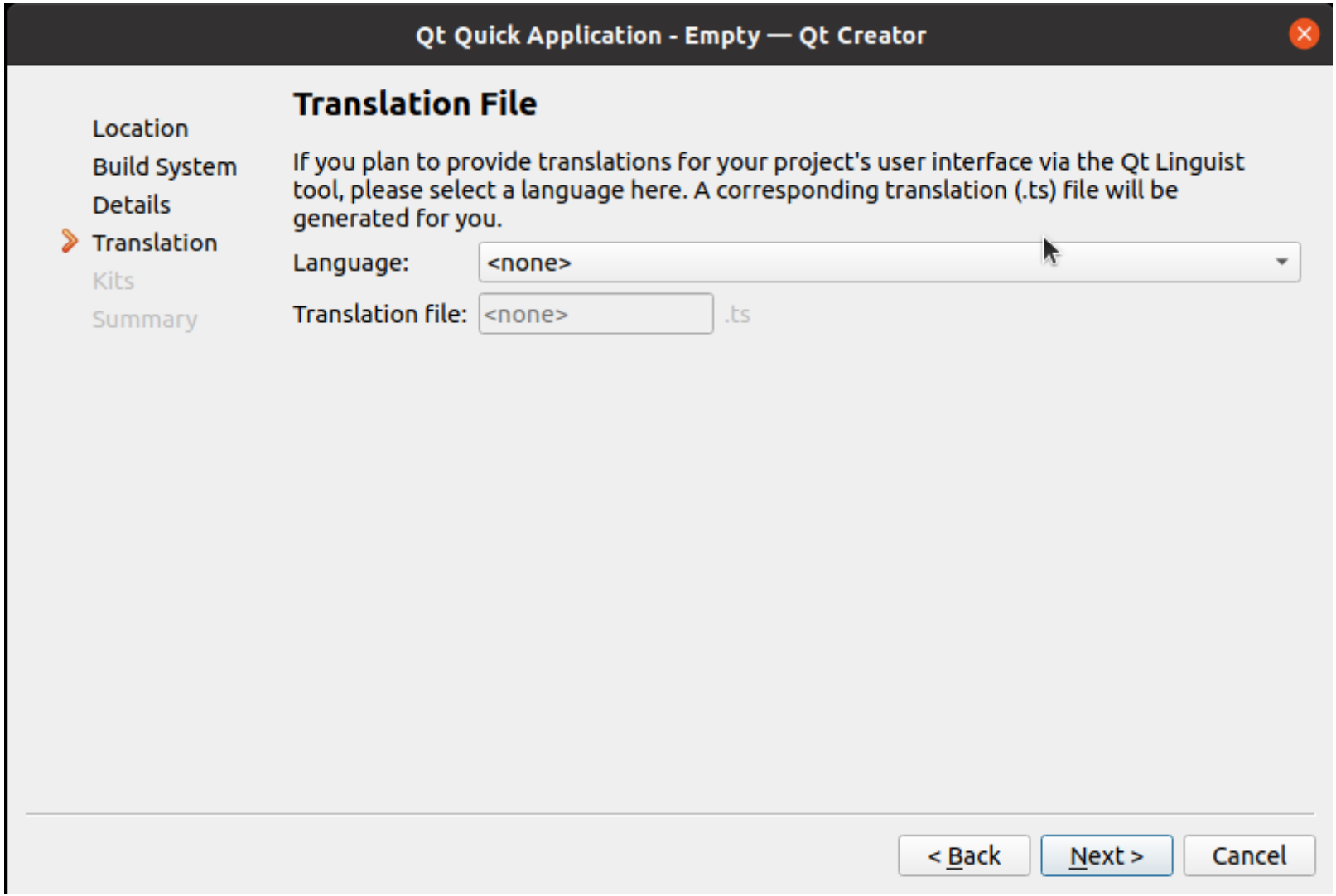
Nextをクリックします。



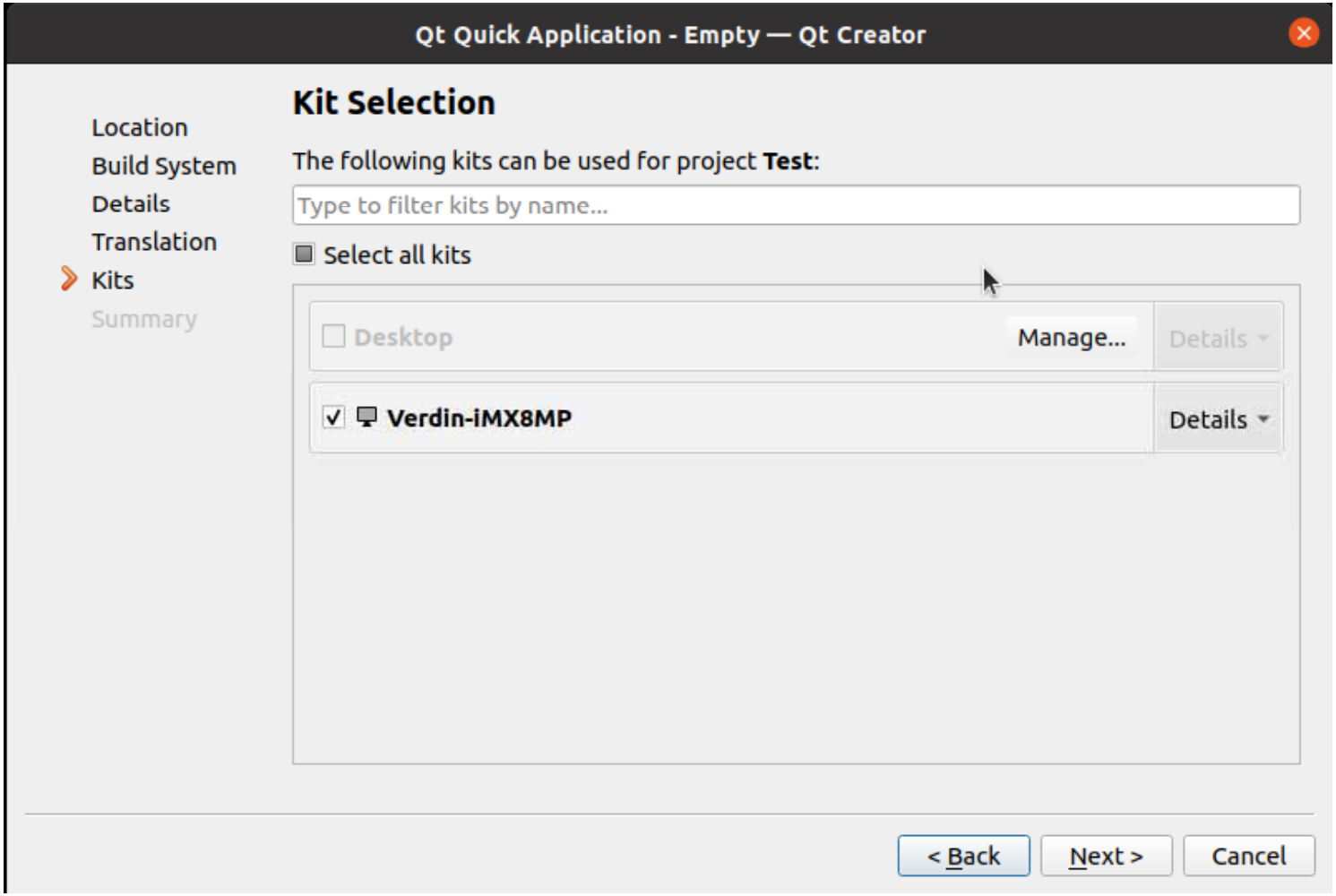
本プロジェクトが動作するQTの最低バージョンを指定します。本マニュアルでは5.14としています。
Nextをクリックします。



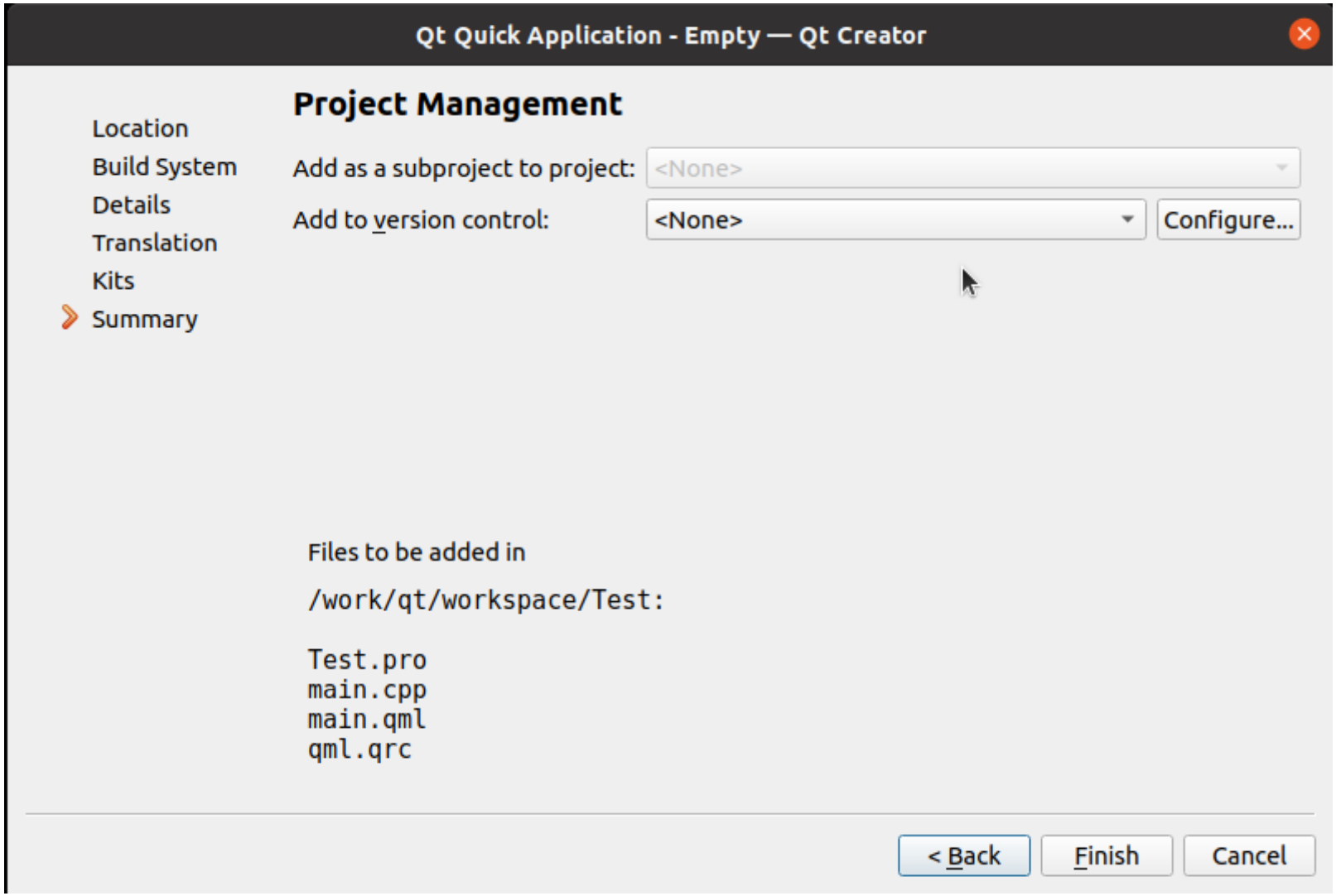
Translation Fileを使用しないためそのままNextをクリックします。



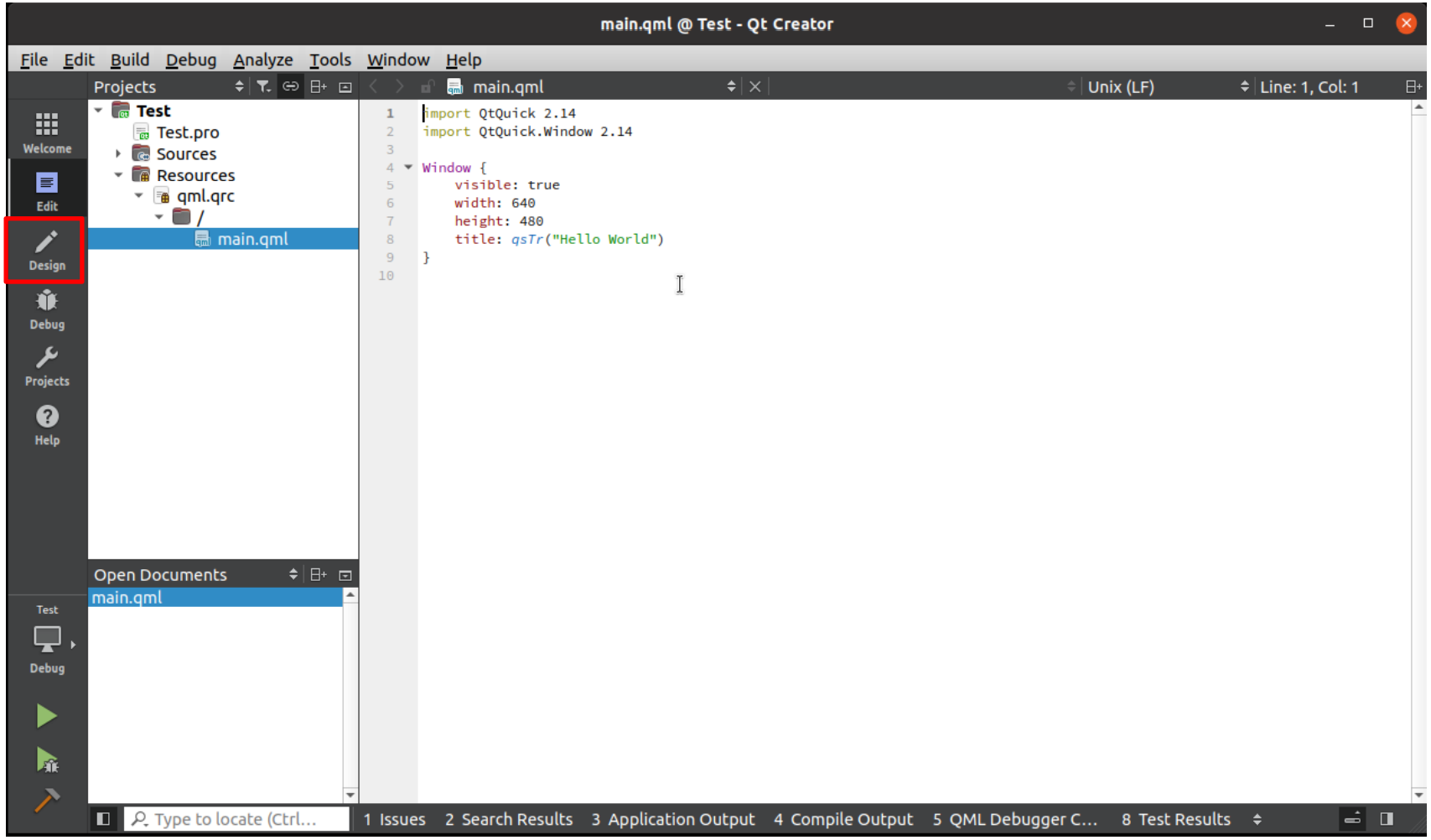
さきほど作成したKitを選択します。Nextをクリックします。



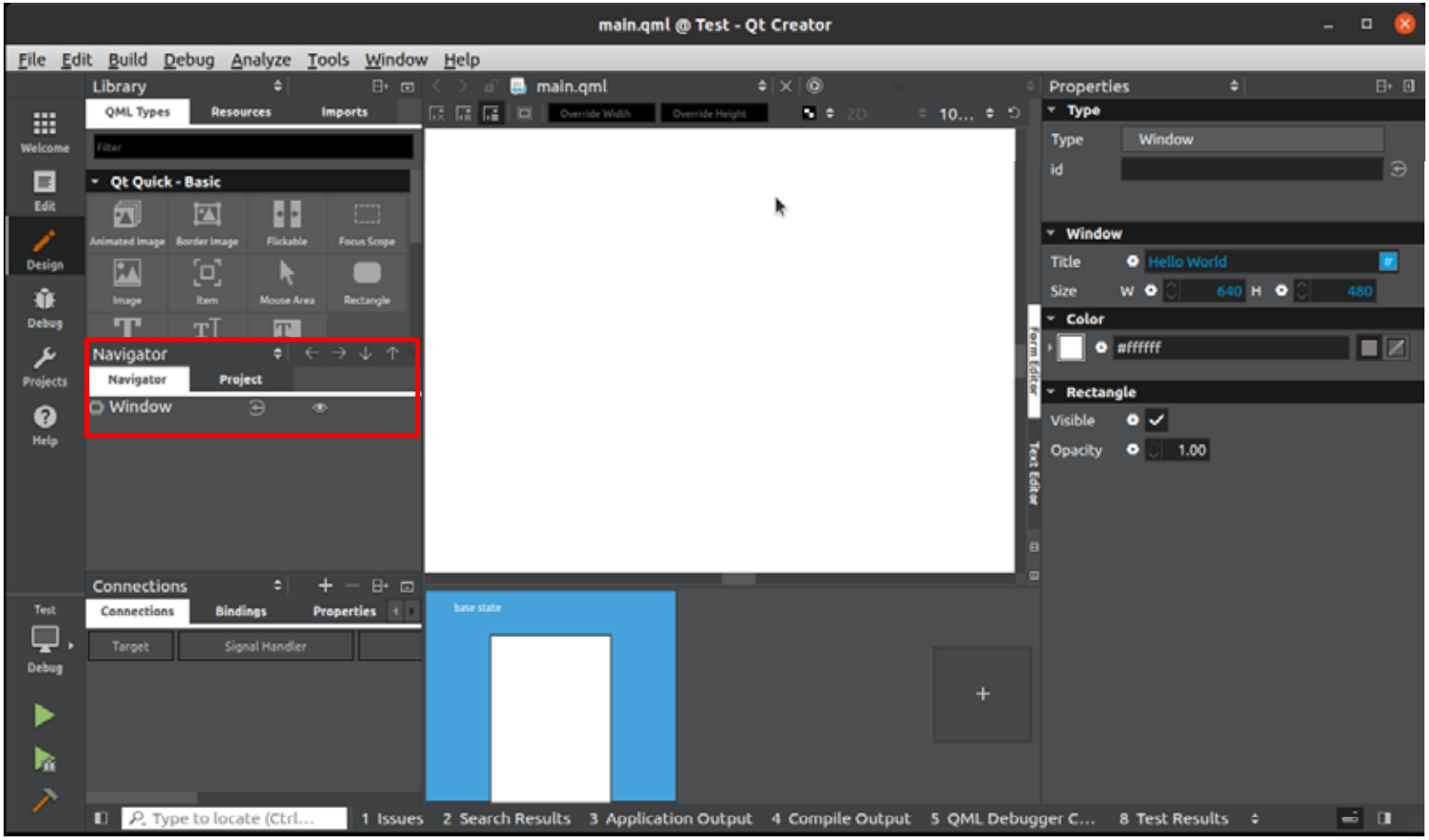
本マニュアルではバージョン管理の設定は行いません。Finishをクリックします。



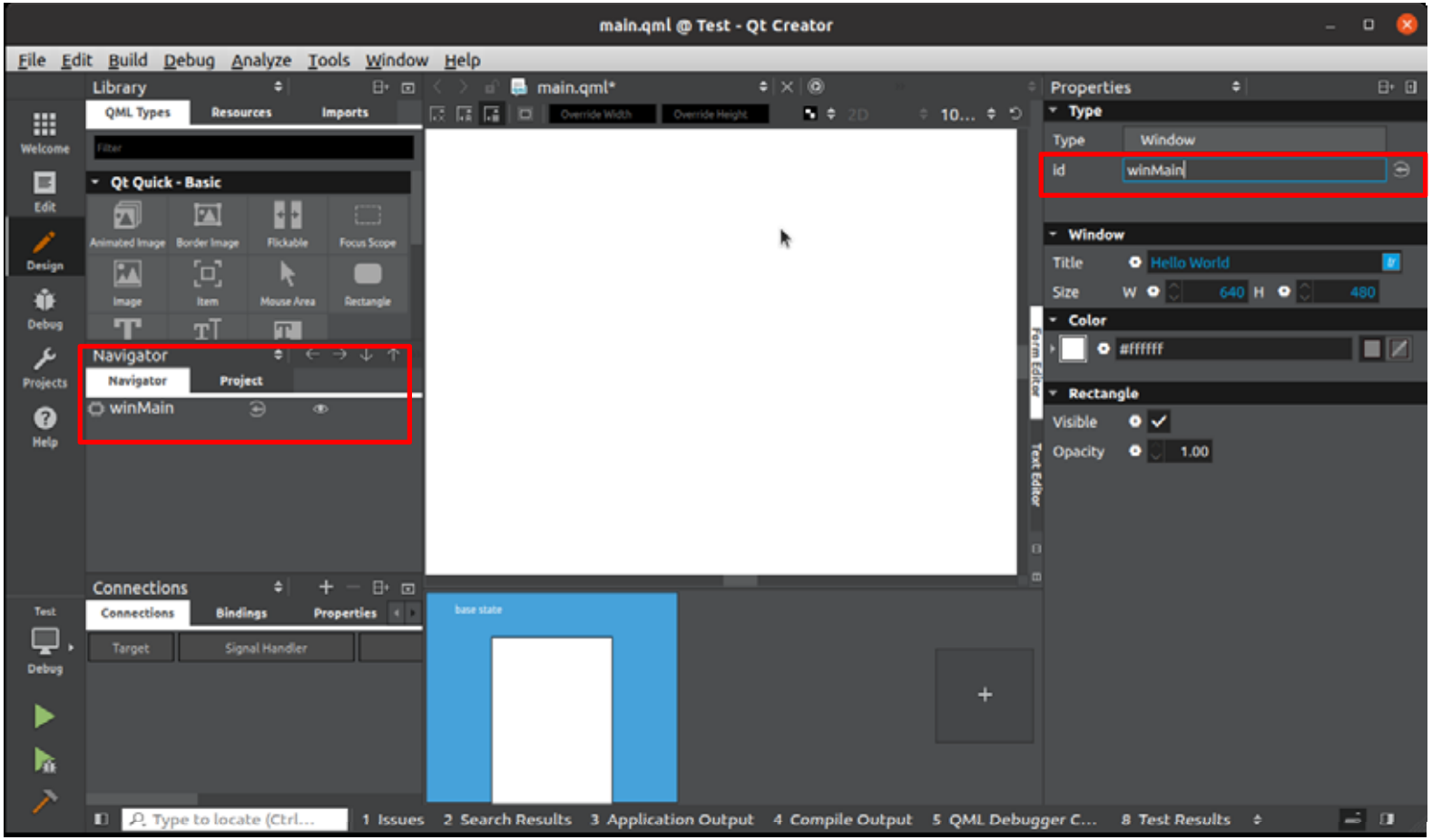
プロジェクトが作成されます。Designを開きます。



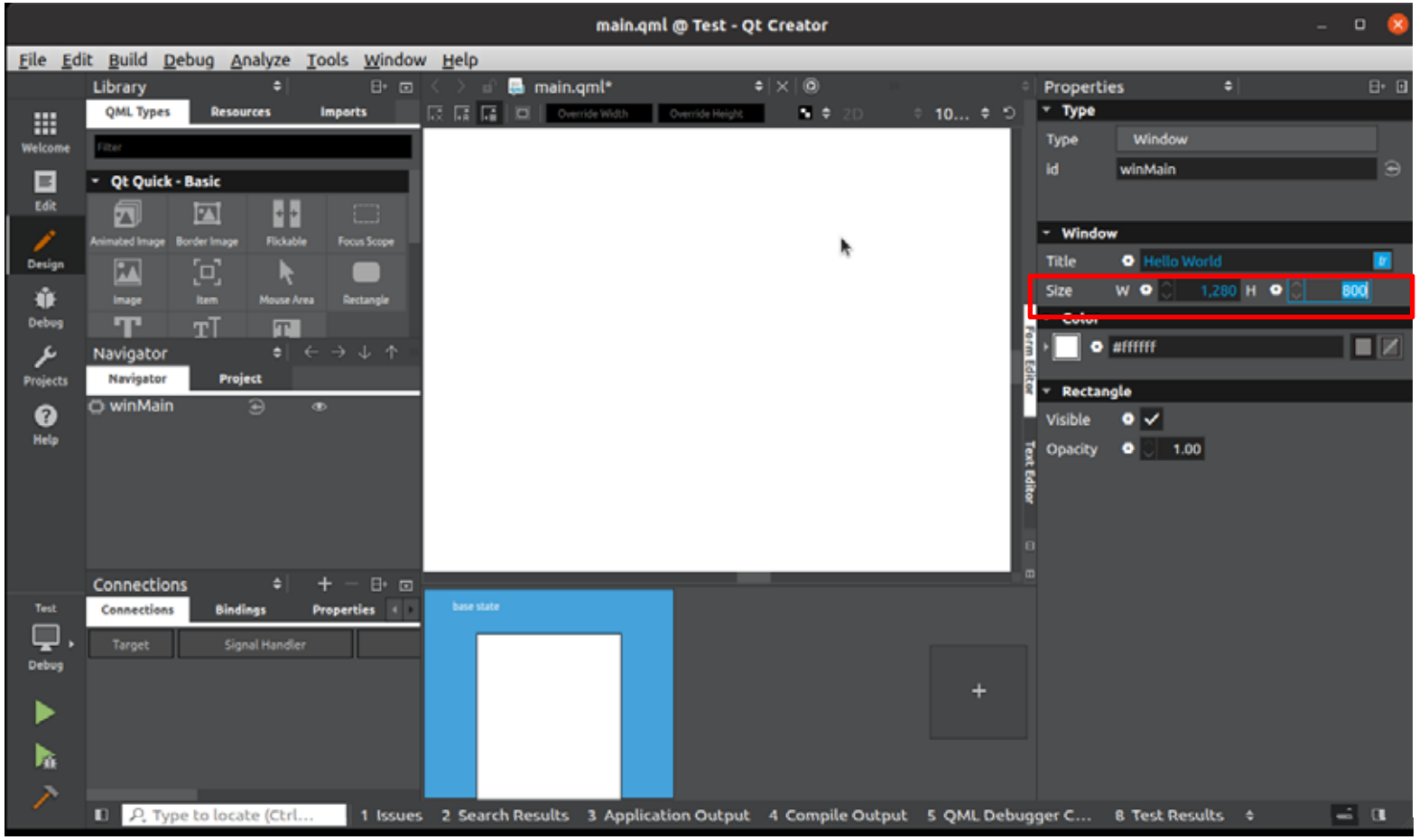
QT Creatorではを容易に編集することができます。
画面左中央あたりのNavigatorの枠内にコンポーネントの一覧があります。最初はWindowという名前のWindow(Type)があります。



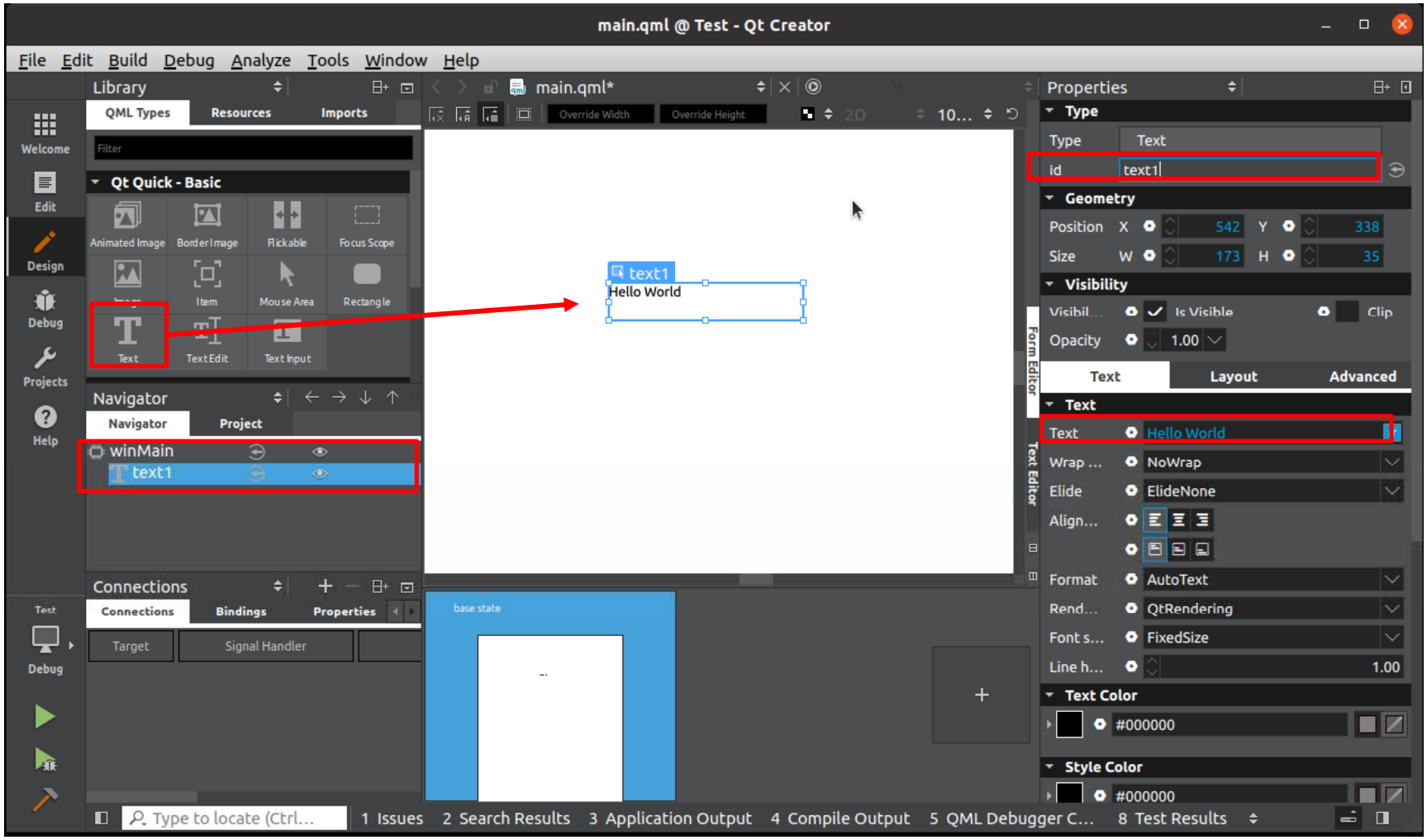
Navigatorの枠内でWindowを選択した状態で画面右側のProperties内のidを変更します。
本マニュアルではwinMainという名前に変更しています。変更するとNavigatorにあるアイテムも変わります。



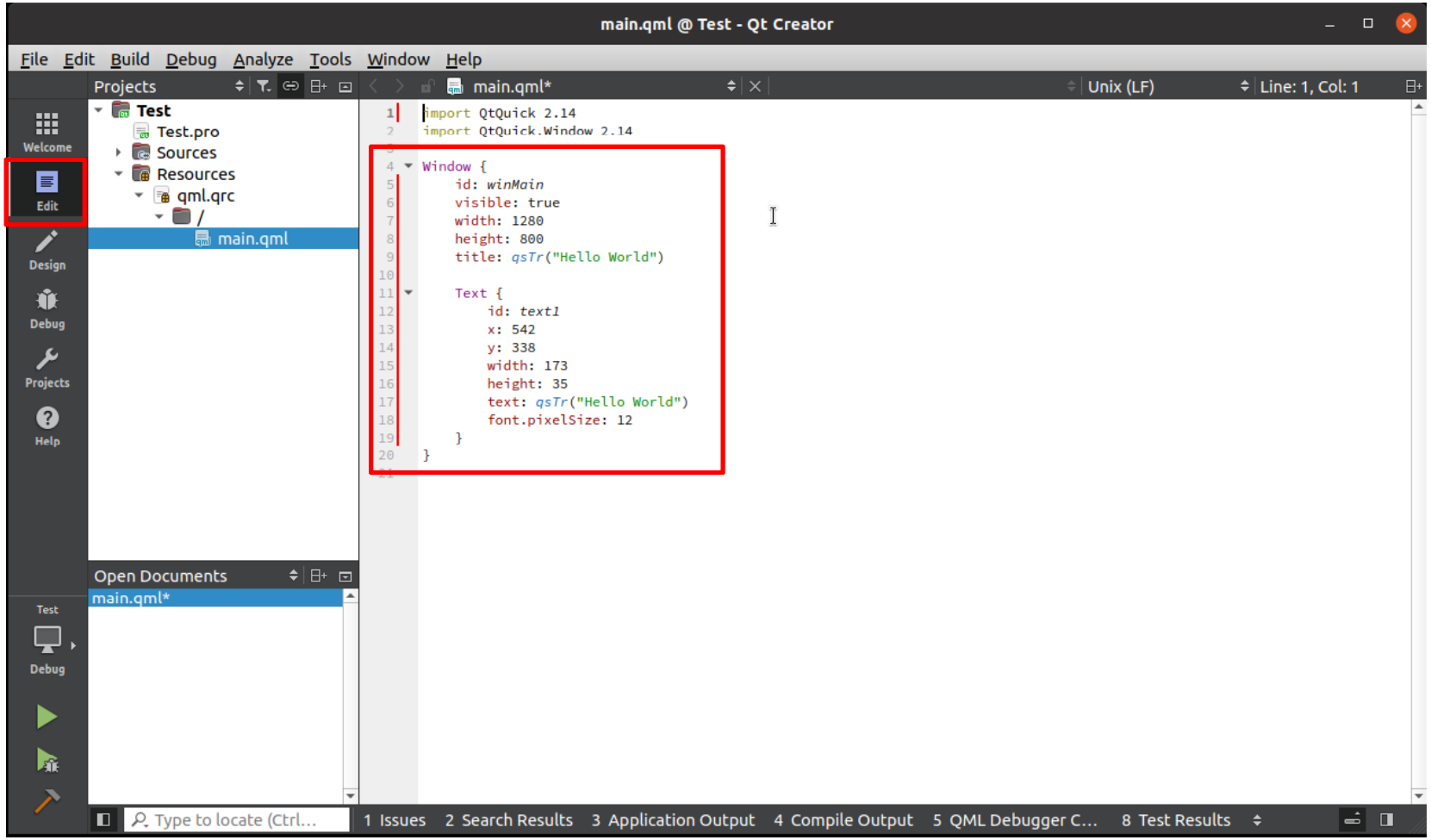
Sizeのプロパティを変更して液晶のサイズに合わせます。本マニュアルで使う10.1インチ液晶は1280x800です。



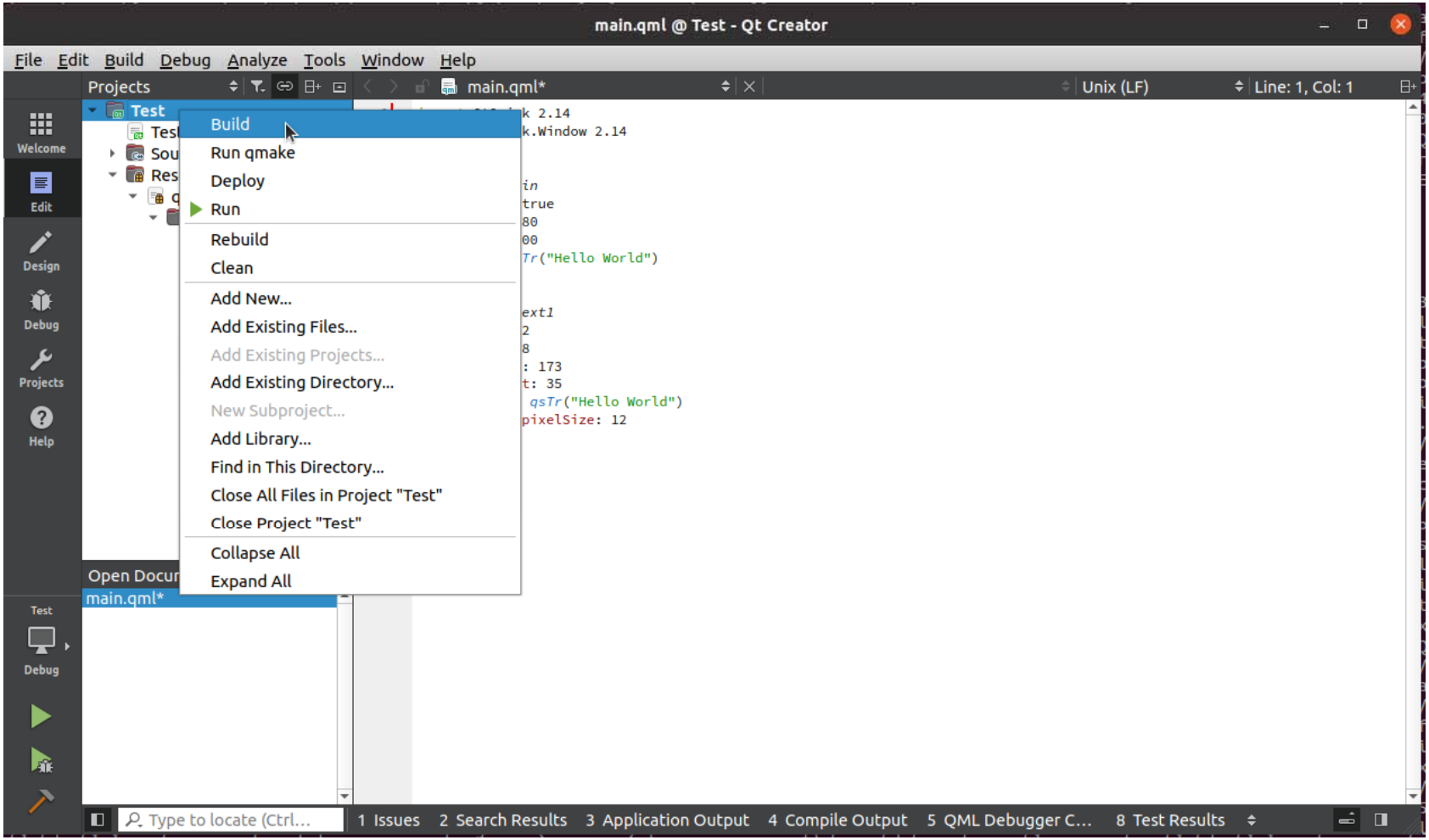
本マニュアルではHello worldという表示をするテキストボックスを追加します。
Qt Quick -BasicからTextをドラッグ＆ドロップして真ん中の白い部分(main.qmlの画面)に持っていきます。
その後、Navigatorで追加テキスト(element)を選択した状態でidにtest1と入力します。
textプロパティにHello Worldと入力します。



Editを選択すると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。

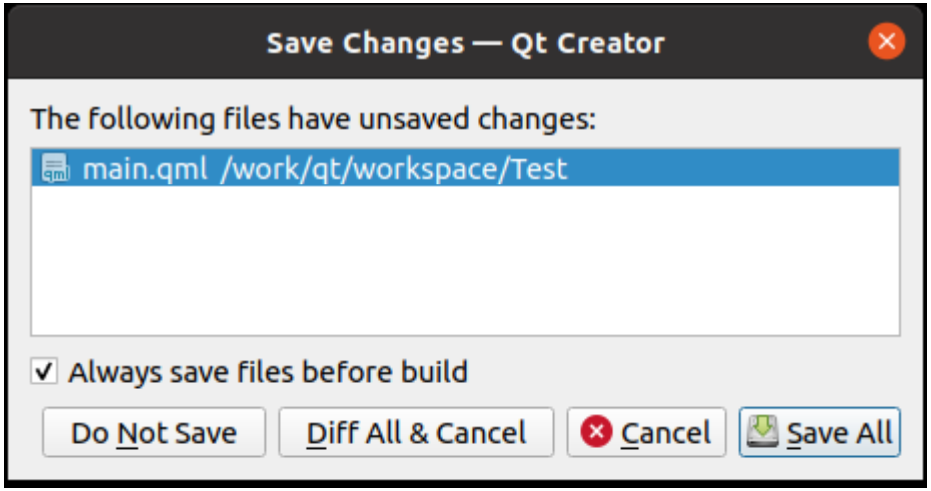


プロジェクトを右クリックしてBuildを選択します。



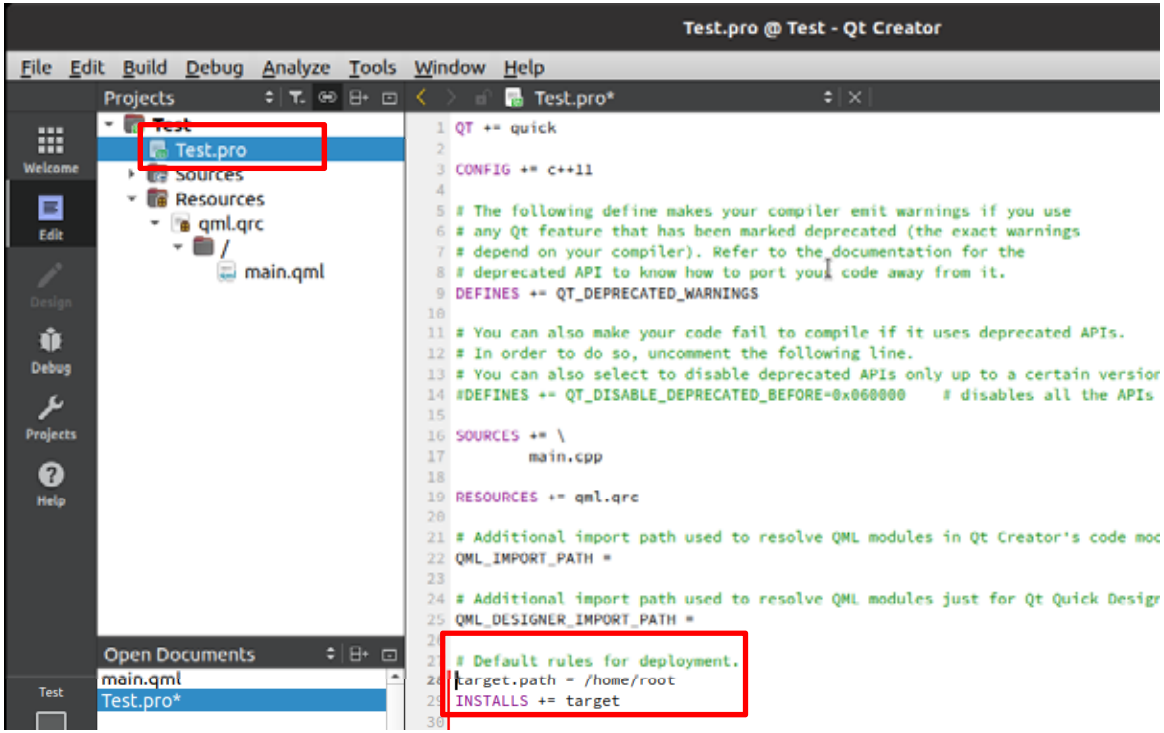
下記のような保存していない内容を保存するかどうか問われます。
Save Allをクリックします。

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。

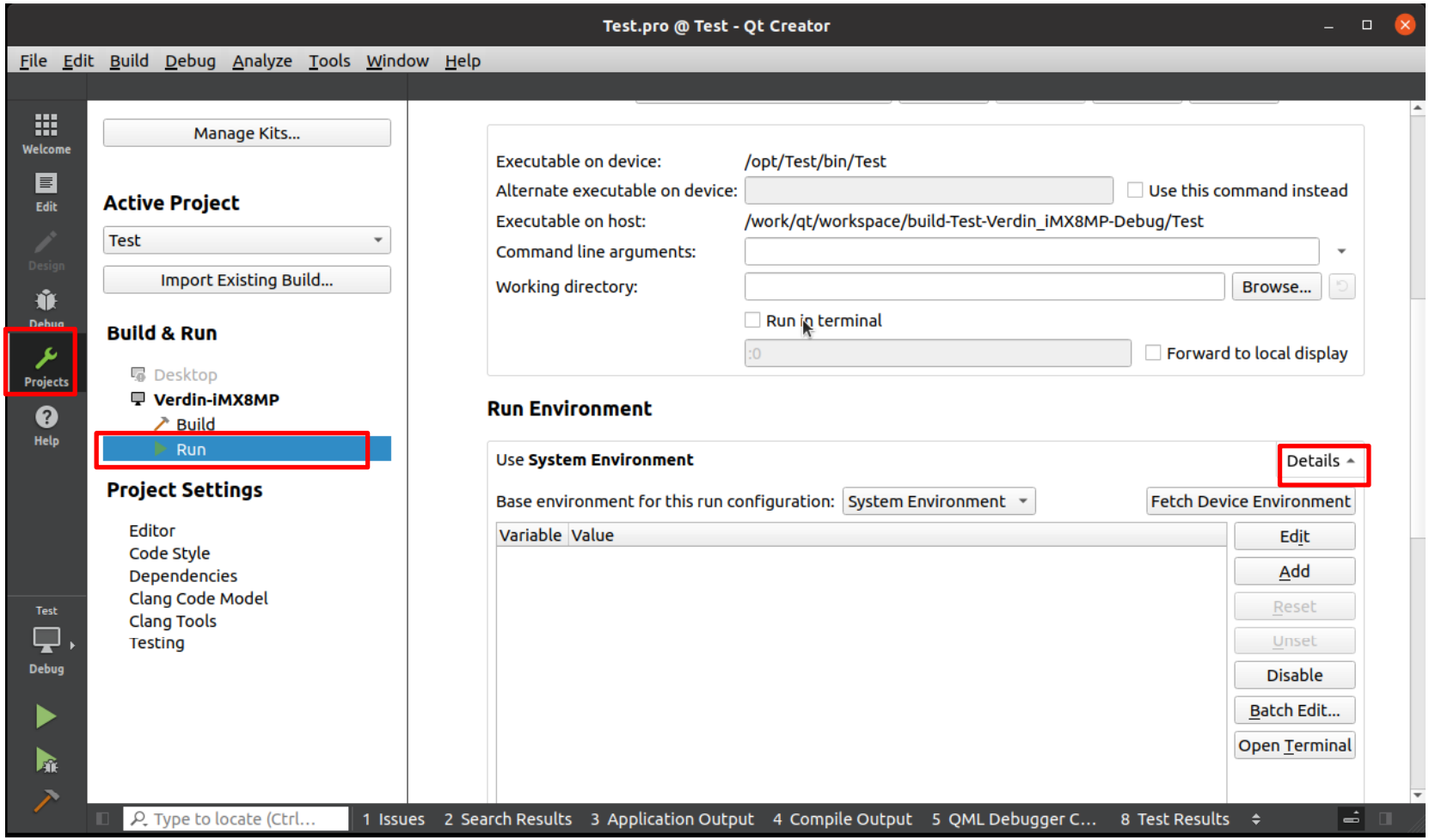


Test.proを開いてデプロイのディレクトリの修正をします。下記はモジュール上の/home/rootに変更する例です。

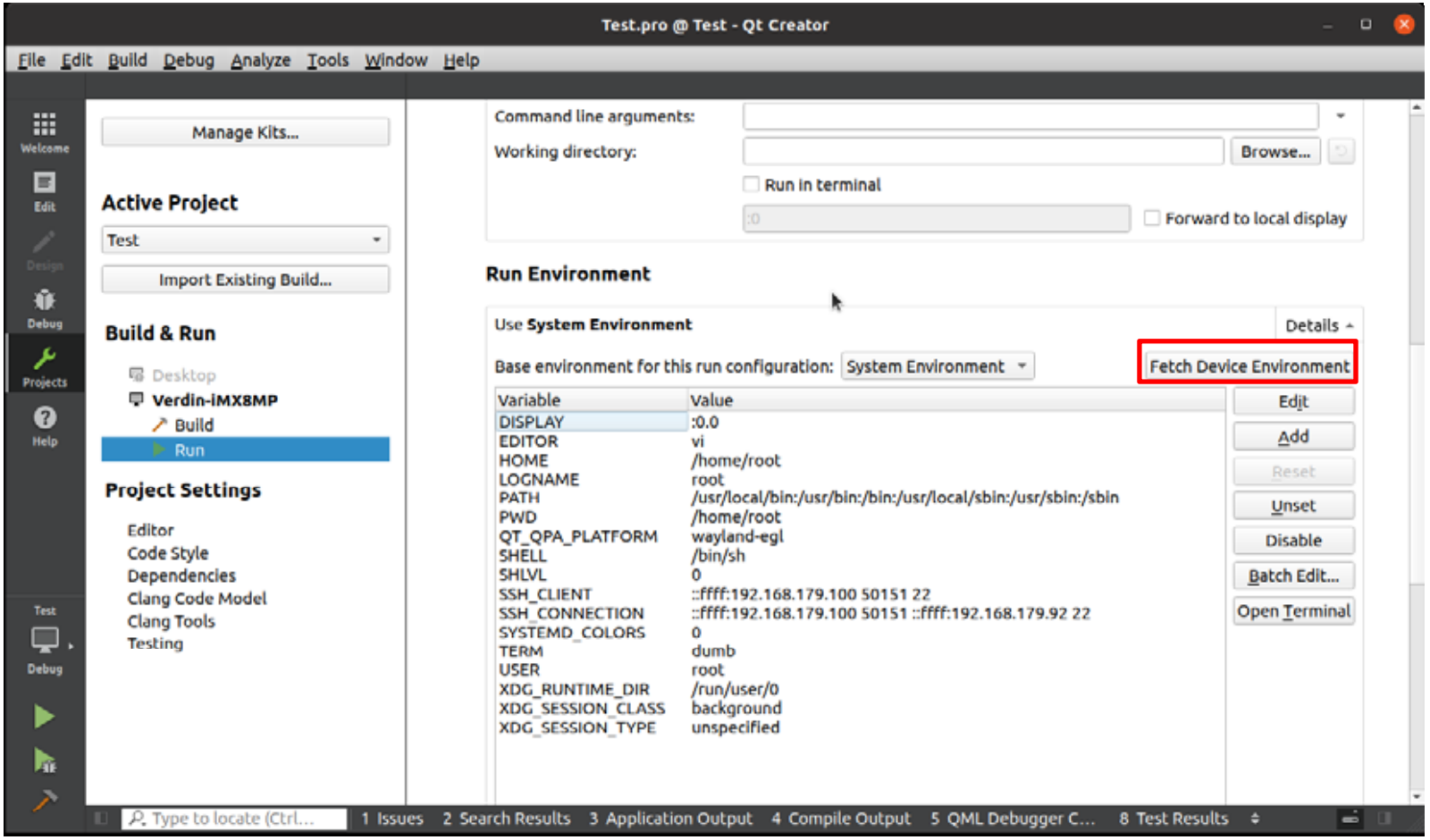
```
# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
->

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



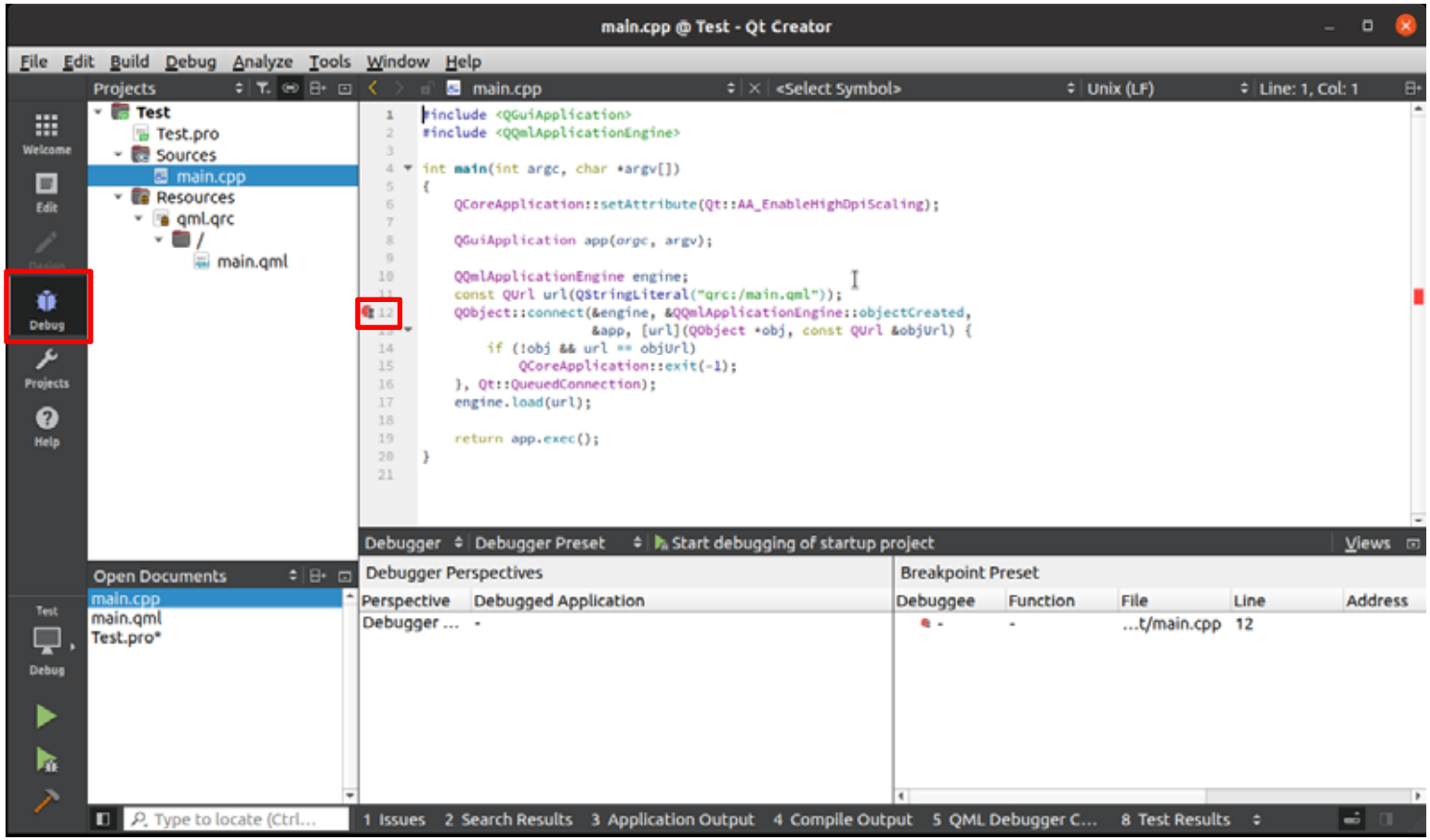
Projectsを開きRunを選択します。Run EnvironmentのDetailsをクリックします。



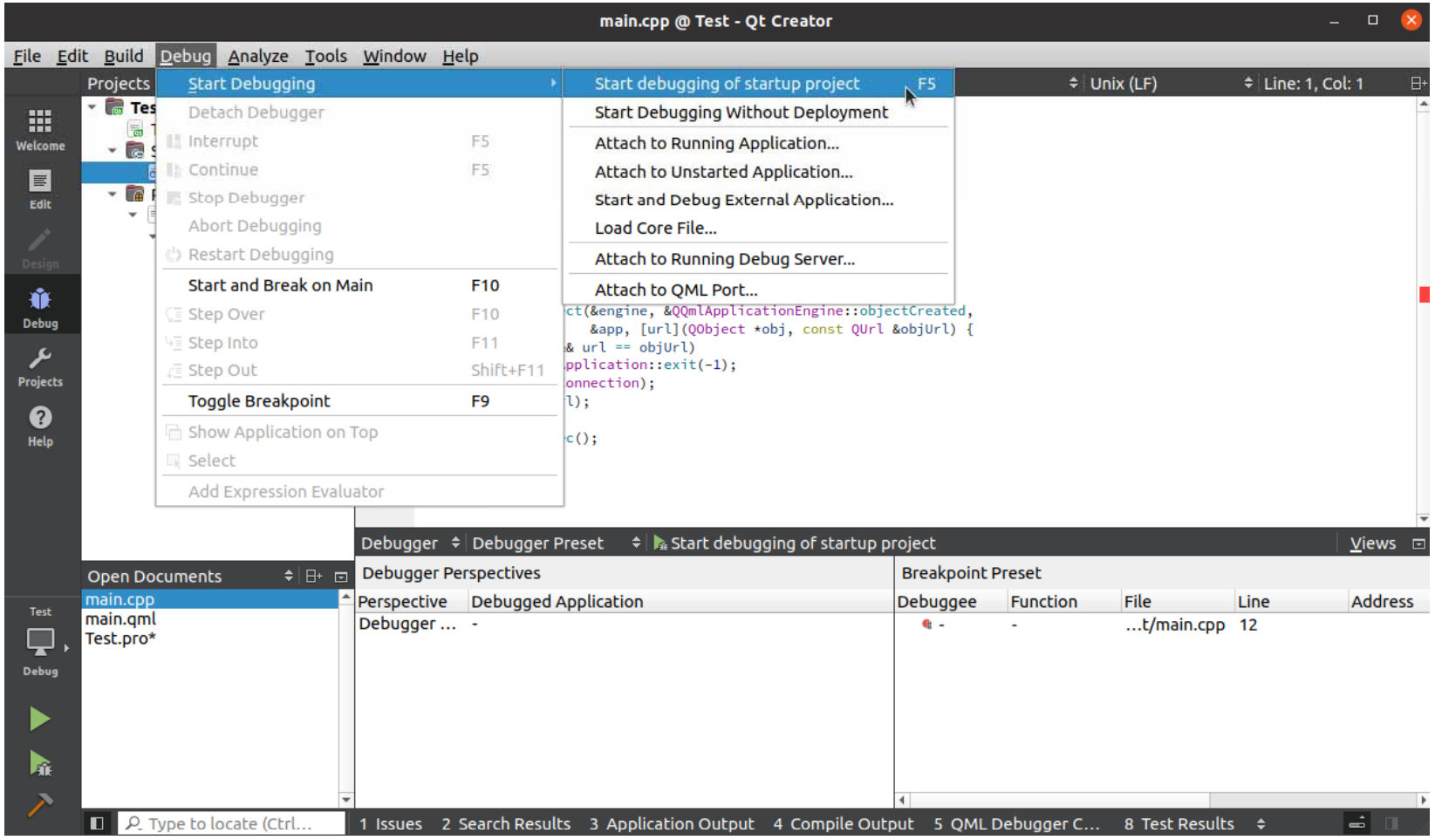
Fetch Device Environmentをクリックしてモジュールの環境変数を取得してデバッグ実行環境に設定します。
モジュールとSSH通信できる状態になっている必要があります。
tdx-reference-multimedia-imageはQTが実行できるようにQT_QPA_PLATFORMにwayland-eglが定義されています。



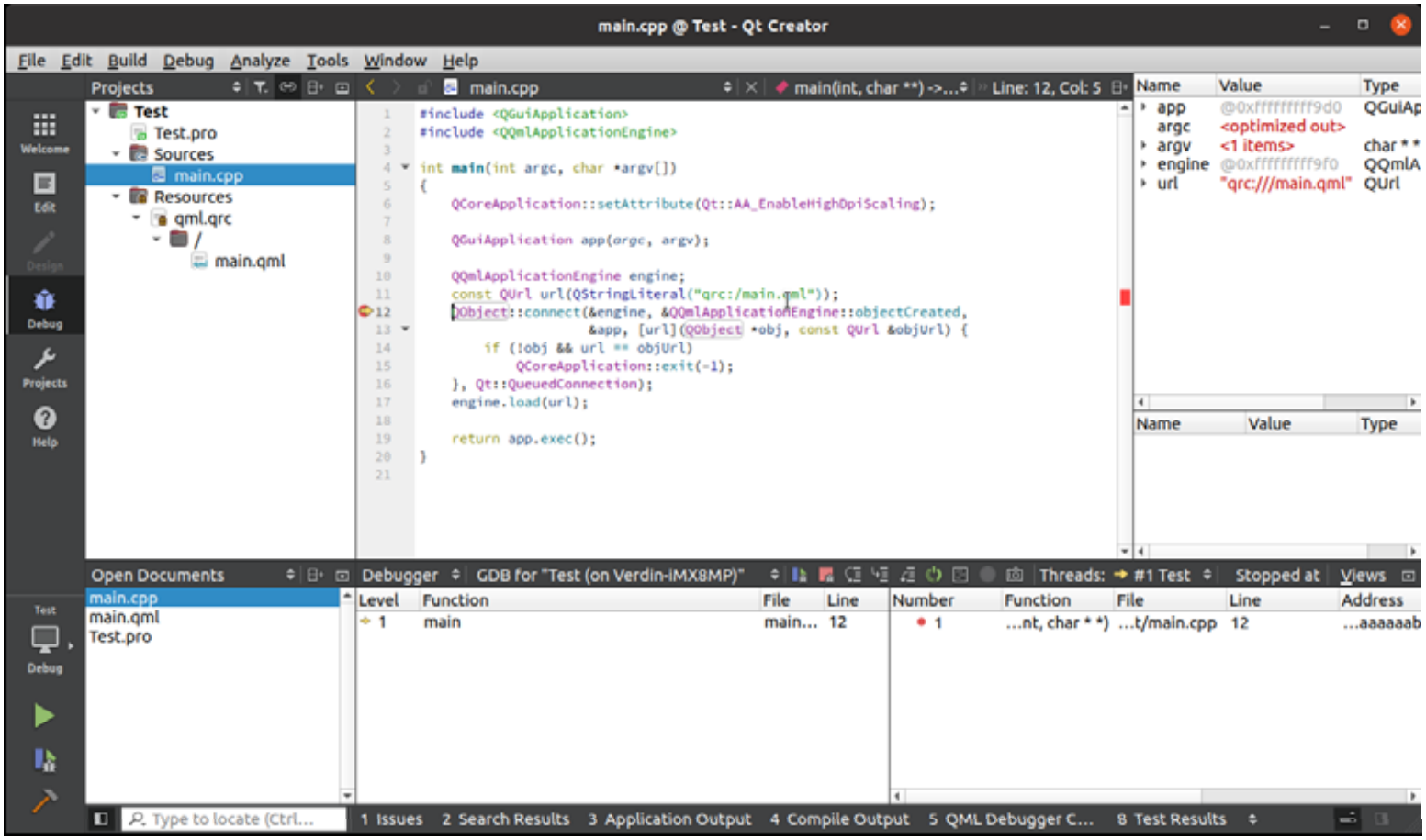
Debugの項目を開き、Projectsからmain.cppを開きます。
行番号の左側をクリックしてブレイクポイントを作ります。ブレイクポイントができると赤丸が出てきます。



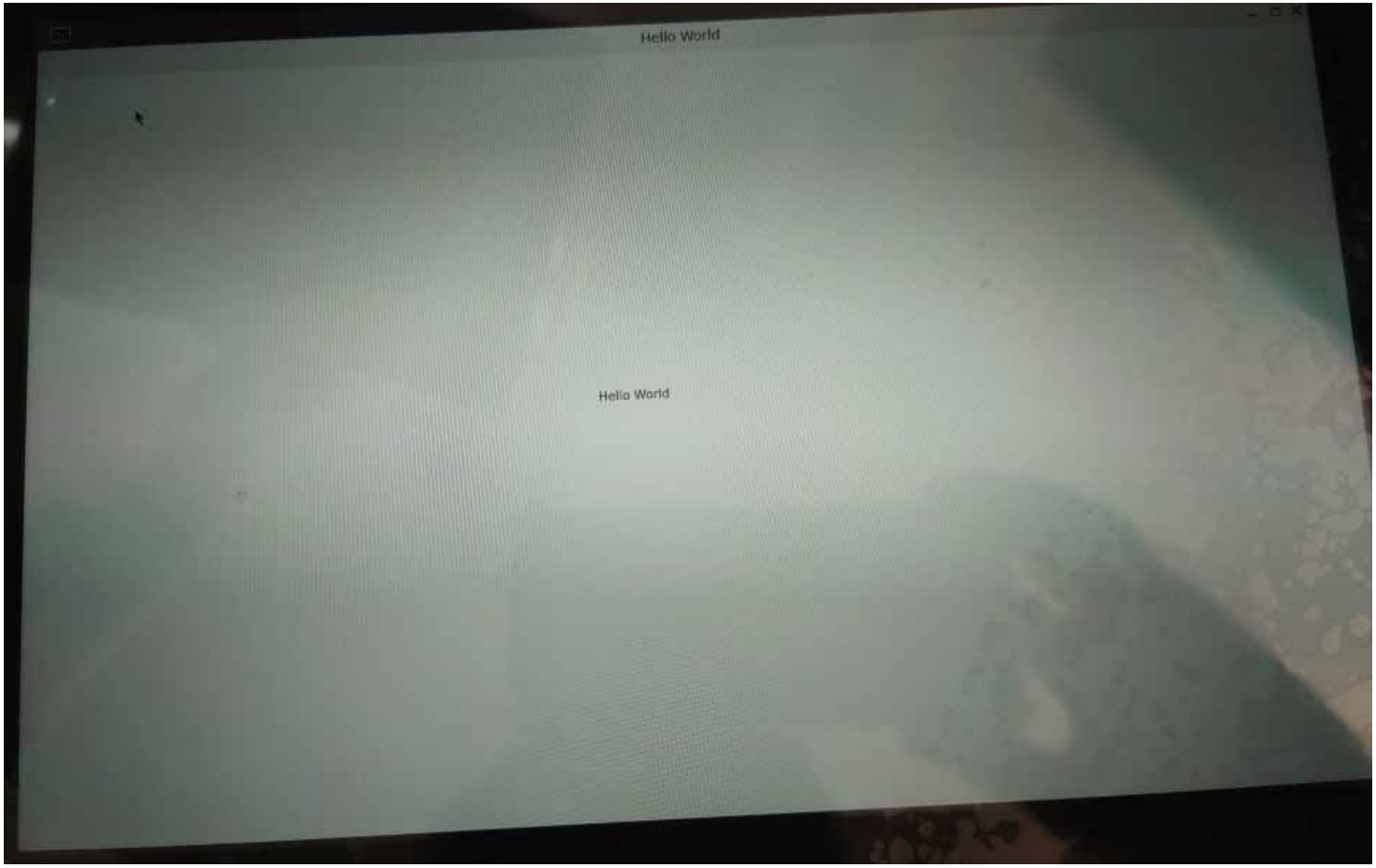
メニューのDebugからStart Debugging > Start debugging of startup projectを選択します。



デバッグが開始してブレイクポイントで止まります。

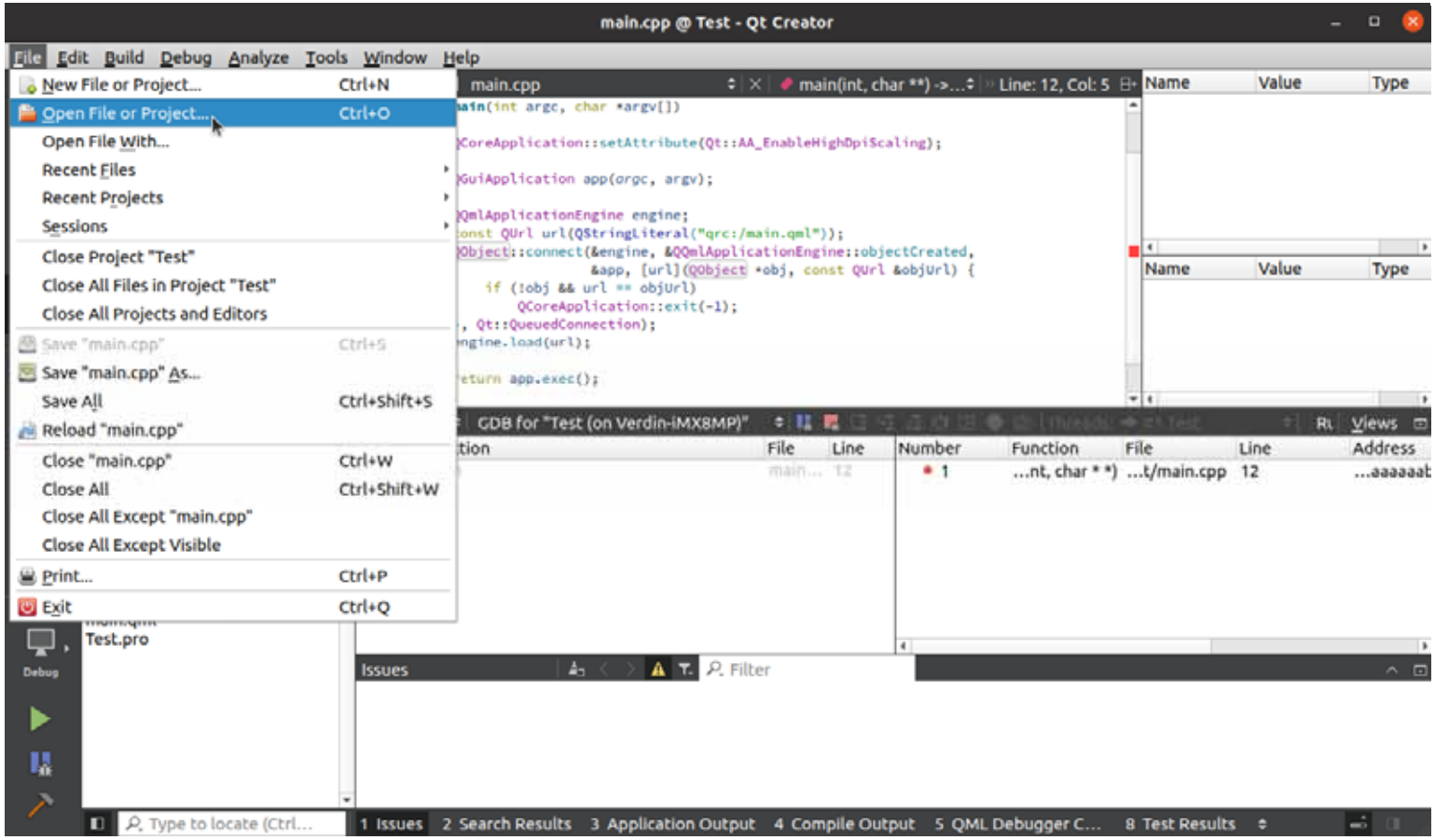


F5でデバッグを続行すると下記のような画面が表示されます。

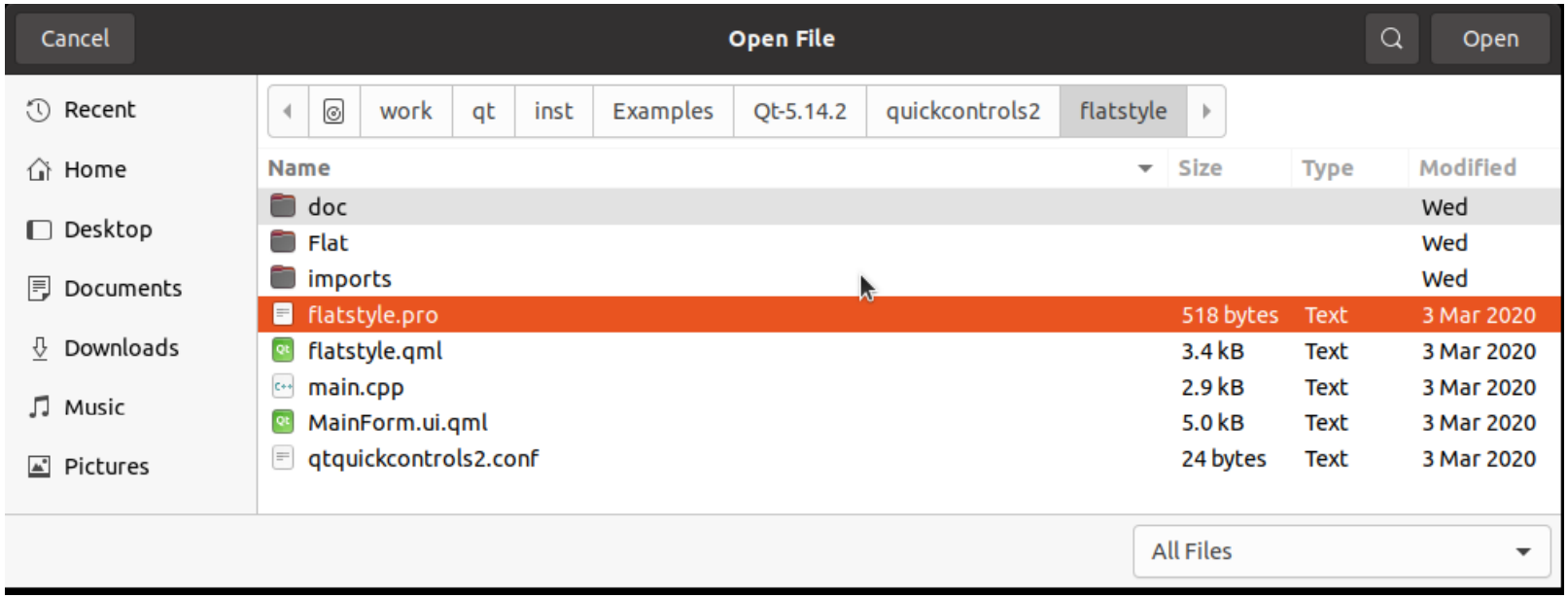


サンプルプログラムの読み込み

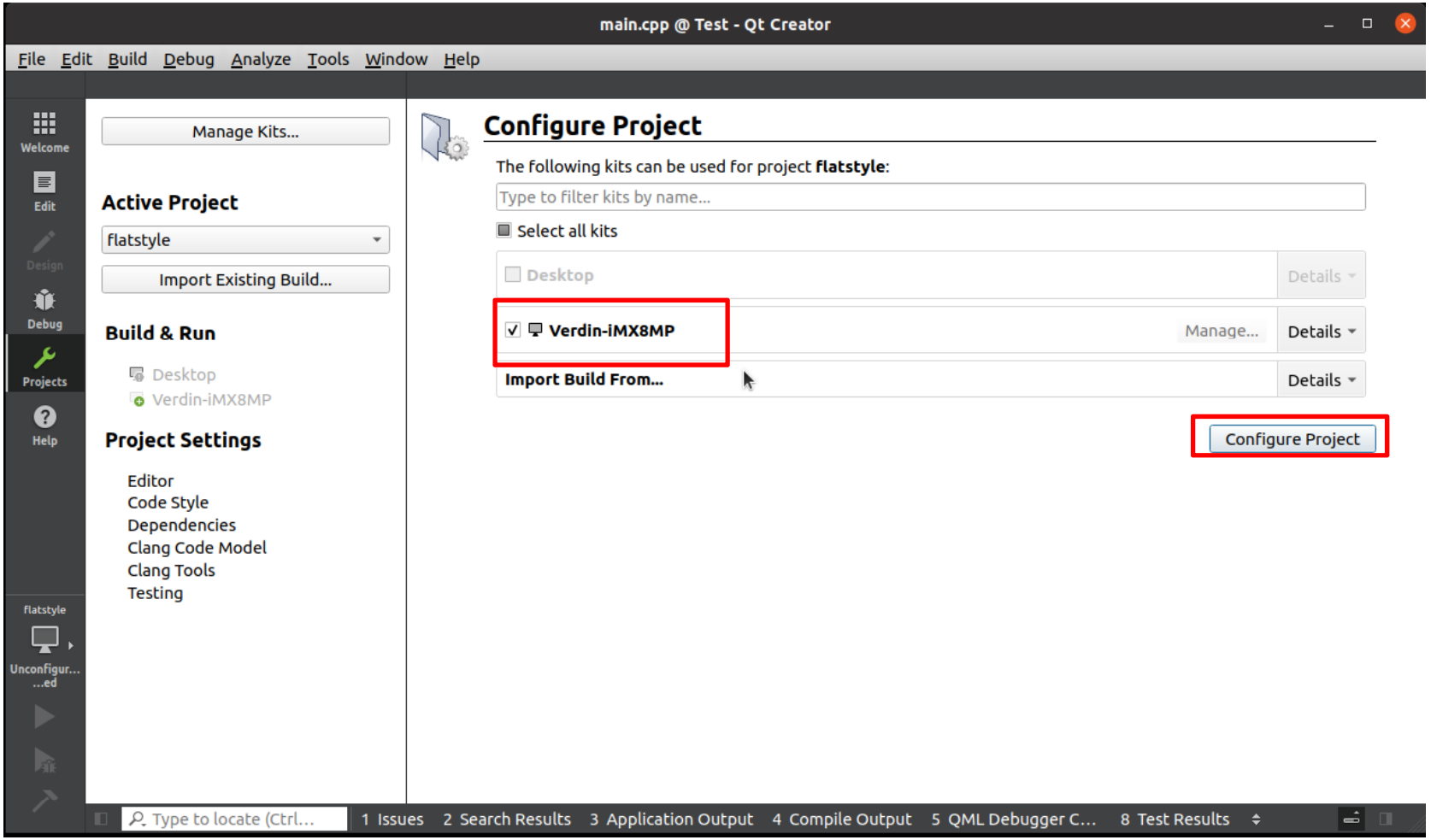
既存のプロジェクトと読み込む場合はFile > Open File or Project**を選択します。**



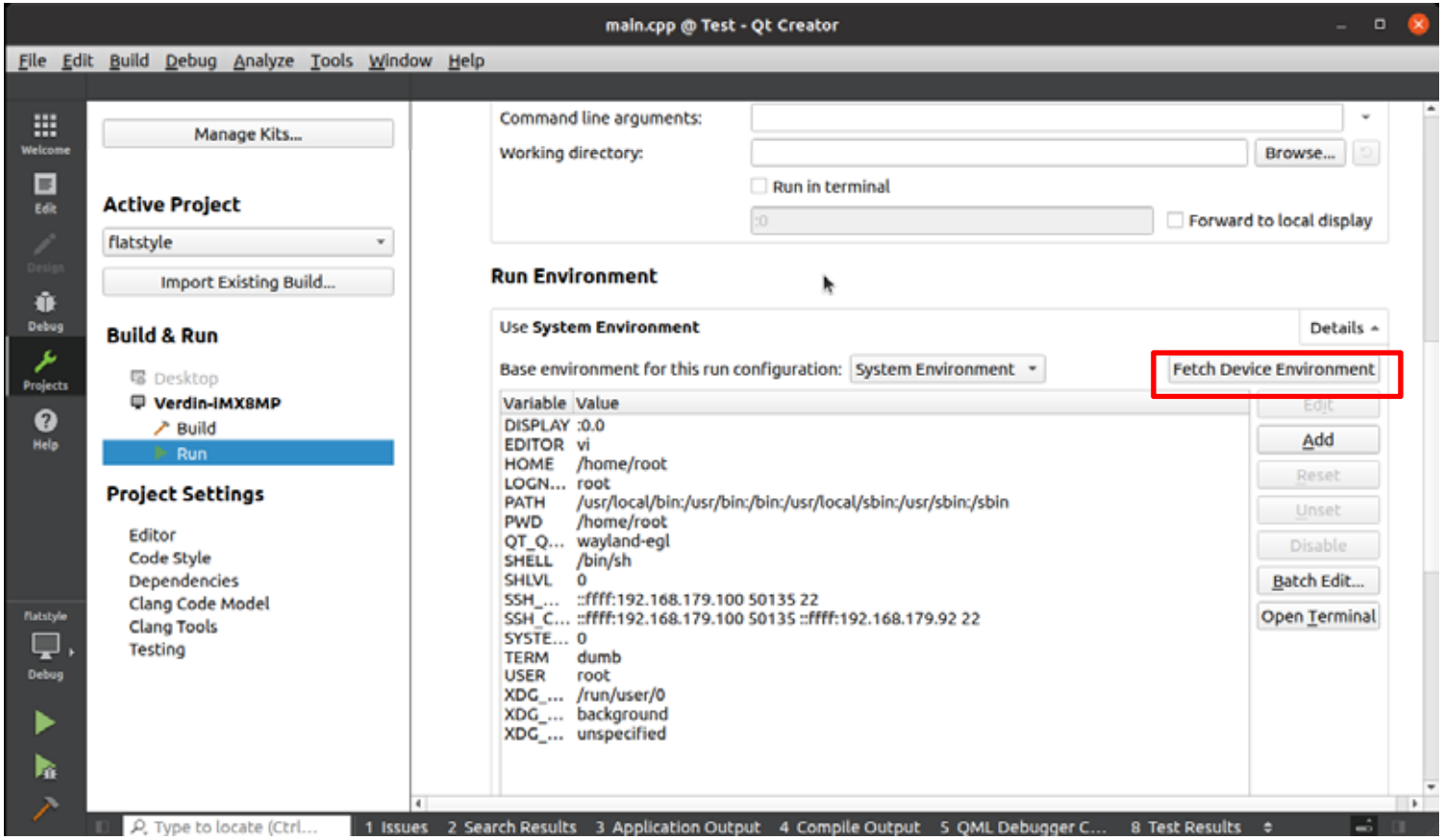
本マニュアルではタッチパネルの動作確認がしやすいflatstyle**というサンプルコードを実行します。**
/work/qt/inst/Examples/Qt-5.14.2/quickcontrols2/flatstyle/flatstyle.pro**を選択します。**



プロジェクトを読み込むと使用するKitsを選択します。
作成したVerdin-iMX8MPにチェックを入れてConfigure Projectをクリックします。

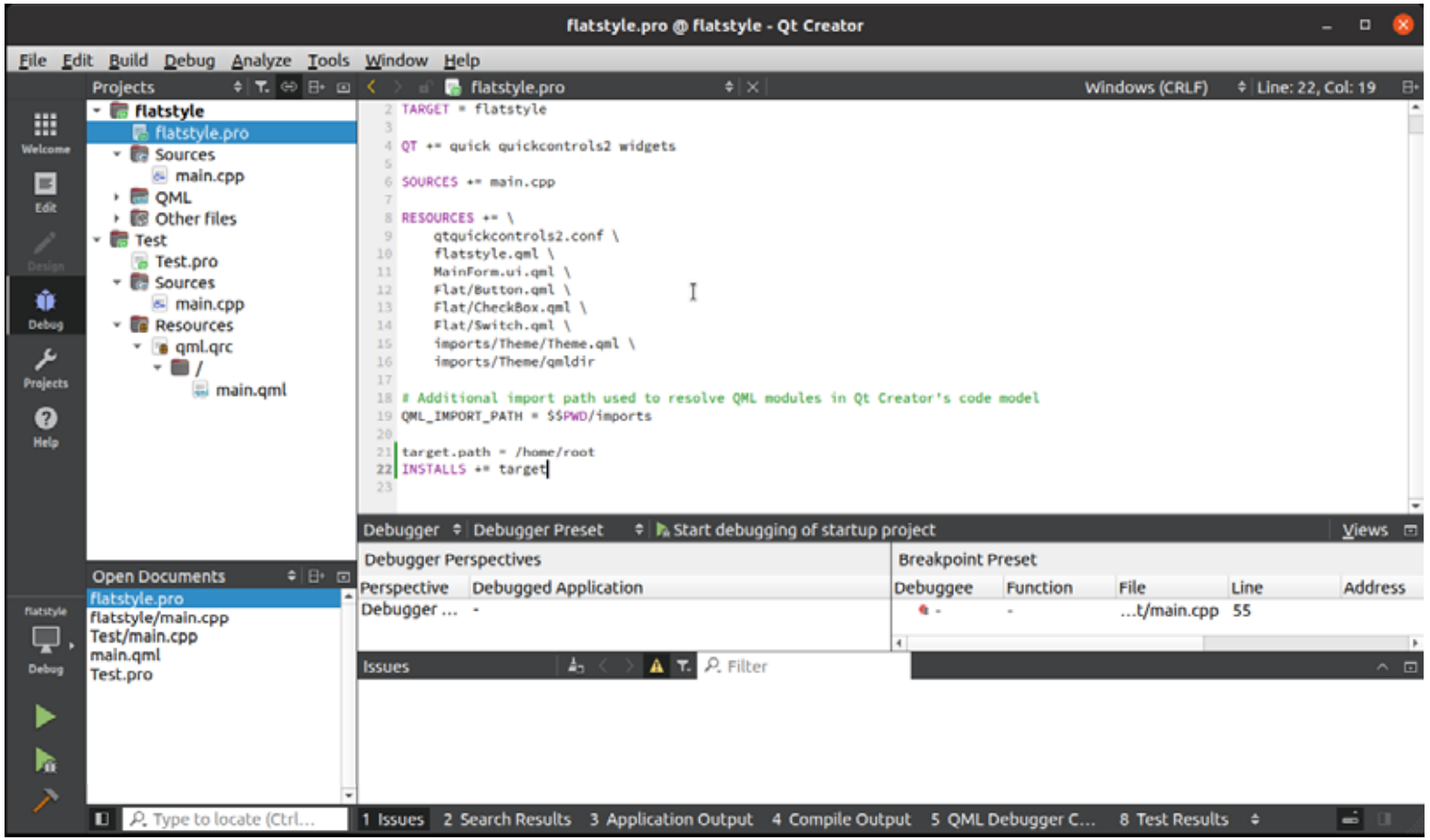


プロジェクトを作成した時と同様Fetch Device Environment**をクリックして設定を取得します。**

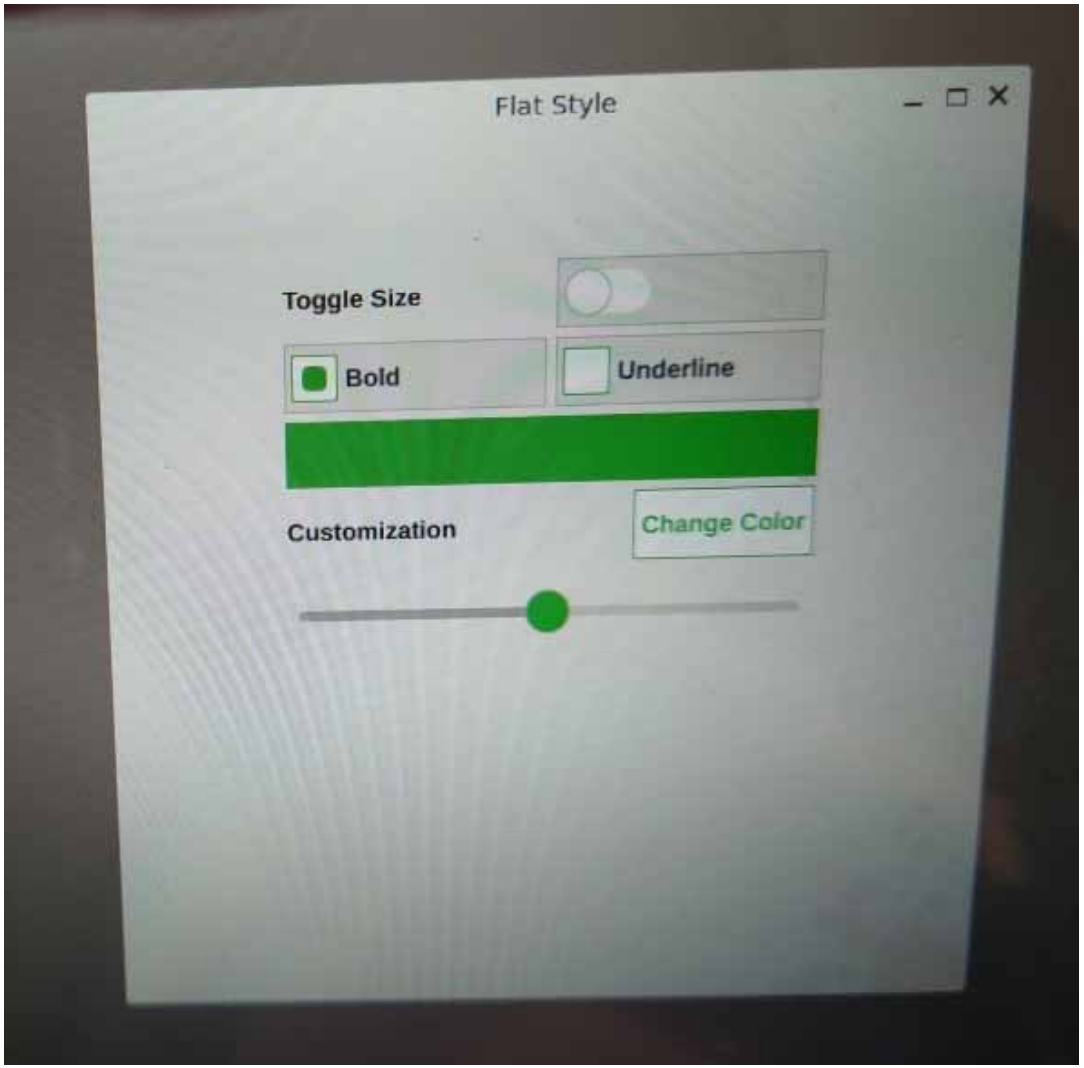


flatstyle.proを開いてtarget.pathを書き換えます。

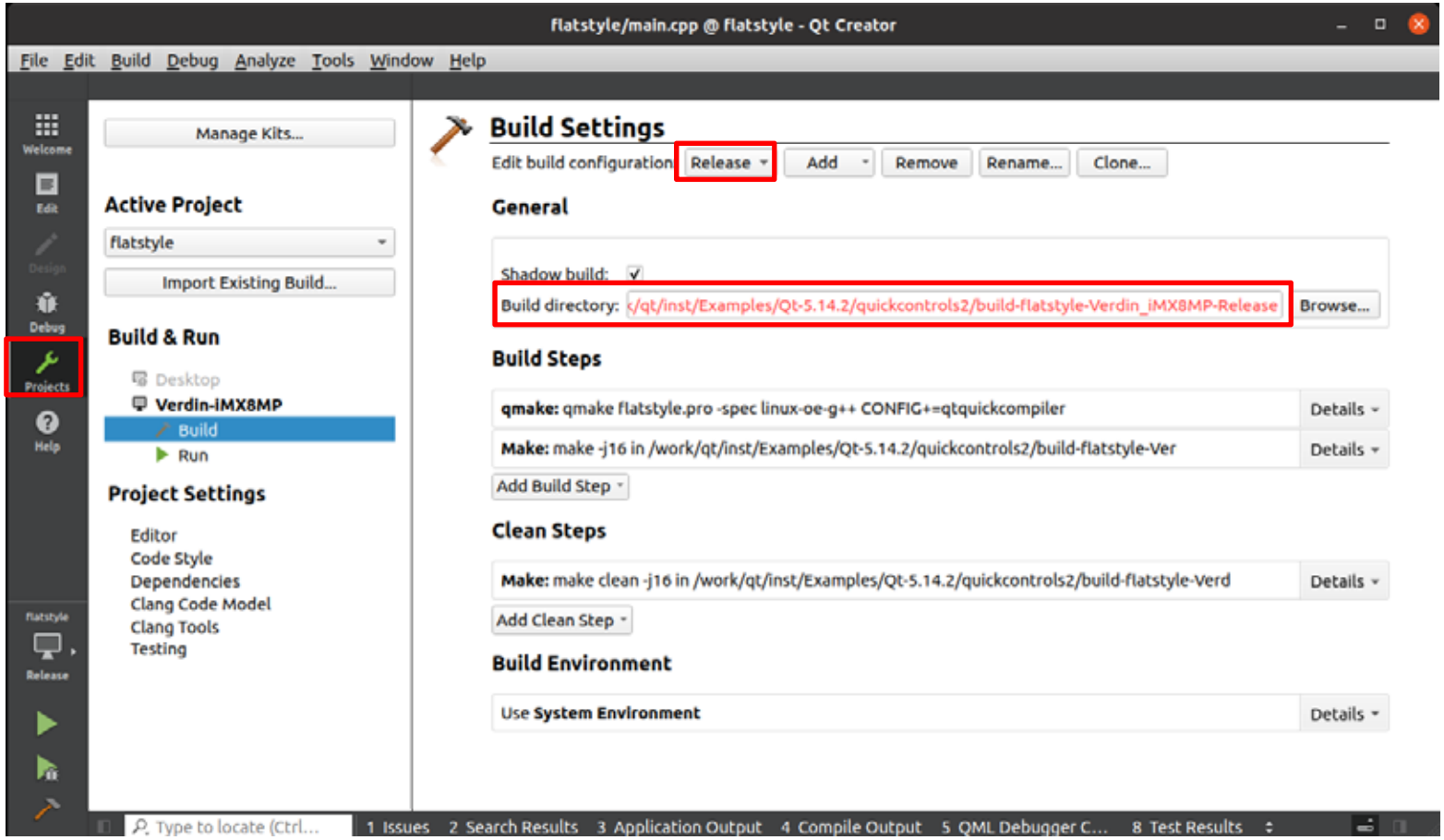
target.path = $$[QT_INSTALL_EXAMPLES]/quickcontrols2/flatstyle
->
target.path = /home/root



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。
最終的にはOSイメージのファイルシステムに組みこむことを推奨します。
以上がQT開発の環境構築からデバッグ、実行ファイル作成の手順です。
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。