XG-335x
Qt（4.8.5)プログラミング

Rev1.0　2014/11/12

目　次

1.　概要

1.1　はじめに

Qt は主に Unix/Linux で利用されている GUI ツールキットです。Windows でも利用することができ Linux と Windows 両方に対応するアプリケーション開発などでしばしば利用されています。また、組込み系のシステムの GUI としても利用されています。
本ドキュメントでは、XG シリーズをターゲットとした Qt のアプリケーション開発方法を説明します。

本ドキュメントでは、アプリケーションノート『**Qt のためのルートファイルシステム作成方法**』によりファイルシステムの構築ができていること前提としています。もしお読みでない場合は、先にそちらをお読みください。

1.2　開発言語について

Qt は C++によって開発されており、多くのアプリケーションが C++によって開発されています。また、多くのコミュニティによって Qt バインディングが開発されており、多くの言語で Qt アプリケーションを開発することが可能です。本アプリケーションノートでは Qt Creator を使った、XG シリーズ用のクロスアプリ開発について解説します。

Qt はバインディングを利用することによって他の多くの言語で開発することも可能です。各種言語で Qt を利用するときはその言語で必要となるバインディングのためのライブラリ等をターゲットにインストールすることが必要となります。

本ドキュメントでは、Qt Creator および C++での開発を前提としたアプリケーション開発について説明します。
その他の言語での詳細については、一般書籍やインターネットから多くの情報を得られますので、それらを参考にしてください。

1.3　Qt の GUI 開発環境について

Qt Creator は Qt 用の GUI 開発環境です。Qt Creator は Ubuntu ソフトウェアセンターに登録されています。Ubutntu12.04LTS の場合 Qt Creator2.4.1(Qt 4.8.0)をインストールすることができますが、最新版を利用したい場合は別途インストール作業が必要となります。

2.　Qt Creator の概要

2.1　システム概要

Qt アプリを実行するには、バックエンドとして X.org、Kdrive(Xfbdev)、DirectFB 等の組み合わせ構成や、framebuffer のみの構成があります。
これらのどのバックエンドと組み合わせるかは、アプリケーションの稼動環境に応じて選択することになります。

2.2　Qt システム構成

Qt を動かすための構成は幾つかありますが、ここでは framebuffer のみの構成を例にして説明します。アプリの稼動環境がすべて Qt による独自開発のアプリケーションの場合は framebuffer のみの構成の方が GUI の動作が速くなります。X11 を前提としたアプリを移植する場合は Xfbdev をバックエンドにした方が有益ですので、必要に応じて環境を構築します。

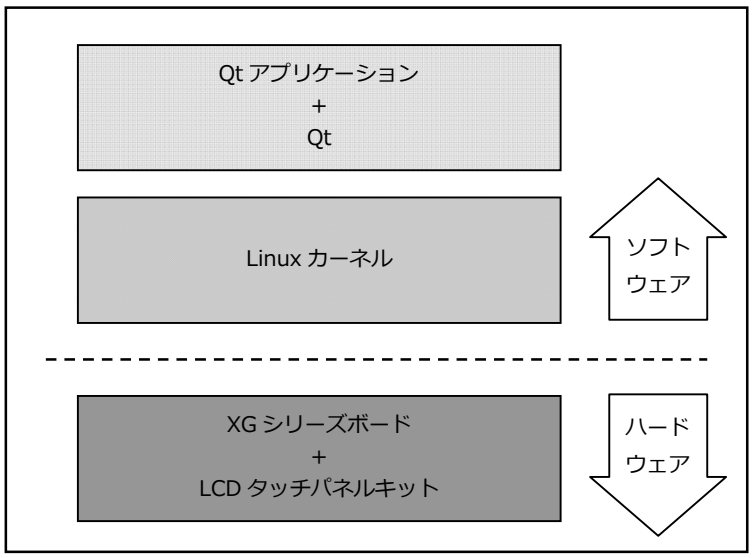


Fig 2.2-1　Qt システム例

3.　Qt Creator のインストール

Qt および Qt Creator をインストールするにはターミナルを開き以下のコマンドを入力します。

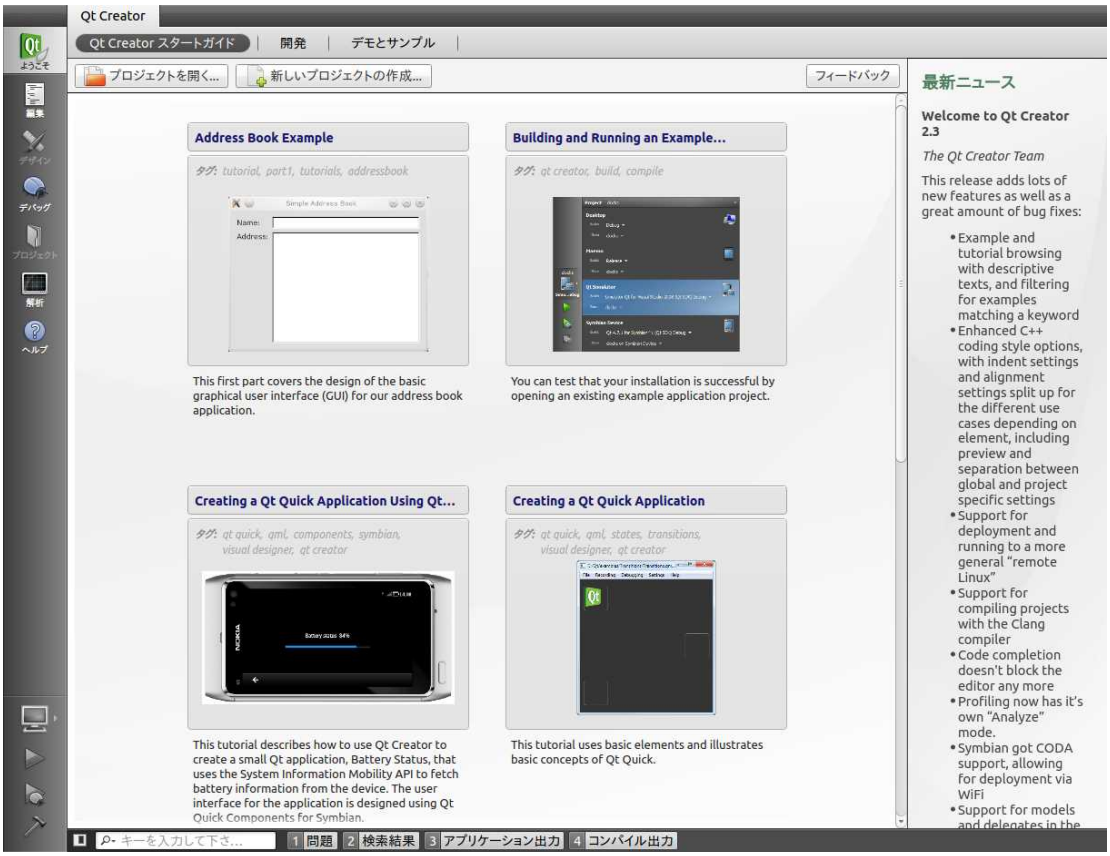
```
省略 $ sudo apt-get install qtcreator
```
入力

Ubuntu12.04LTS でインストールされる Qt Creator は 2.4.1(Qt 4.8.0)です。XG シリーズの Qt は 4.8.5 です。4.8.0 以降に追加された機能についての互換性が必要でしたら、Ubuntu 側を最新の Qt にする必要があります。

4.　Qt Creator によるサンプルアプリ作成

4.1　Qt Creator の起動

Ubuntu Dash ランチャーに Qt Creator のアイコンがある場合は、それをクリックして、Qt Creator を起動します。ランチャーに Qt Creator のアイコンが無い場合は Dash ホームアイコンをクリックし「検索」項目に Qt と入力して Qt 関連アプリを検索し「Qt Creator」をクリックしてください。

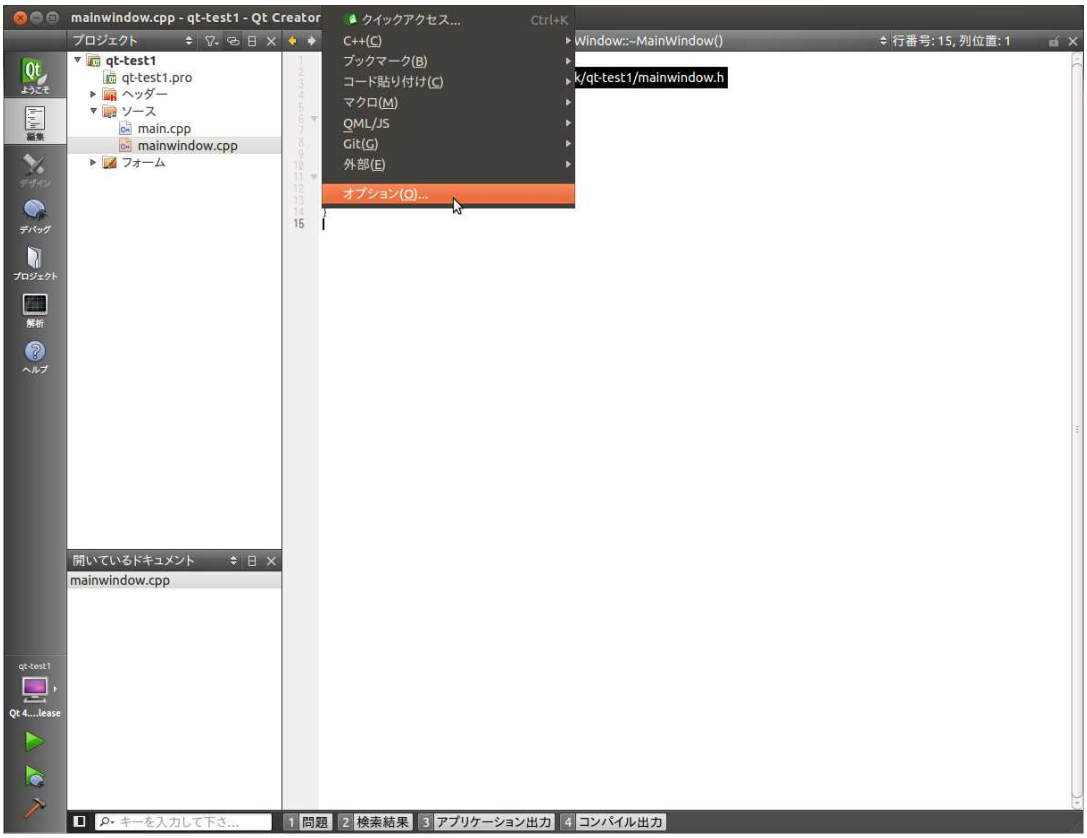


Dash ランチャーに Qt Creator が残っていない場合、Qt Creator 起動中にアイコンを右クリックし、ポップアップメニューにて「ランチャーに固定」をクリックすると Qt Creator が固定されます。

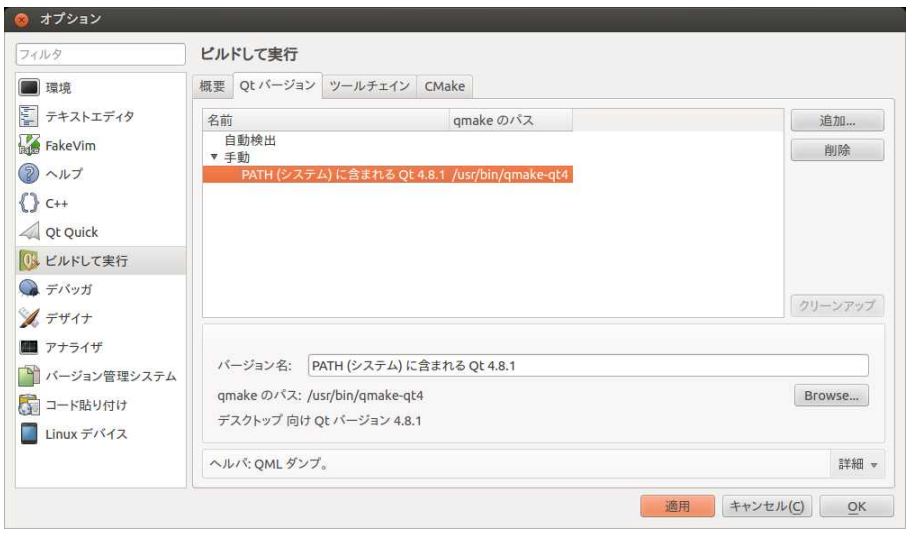
4.2　XG シリーズのためのデフォルト設定

XG-3358, XG-BBEXT 用のクロス開発用のデフォルト設定をします。デフォルト設定ですので一度設定するだけです。

① メニュー「ツール」－「オプション」にてオプションダイアログを開きます。

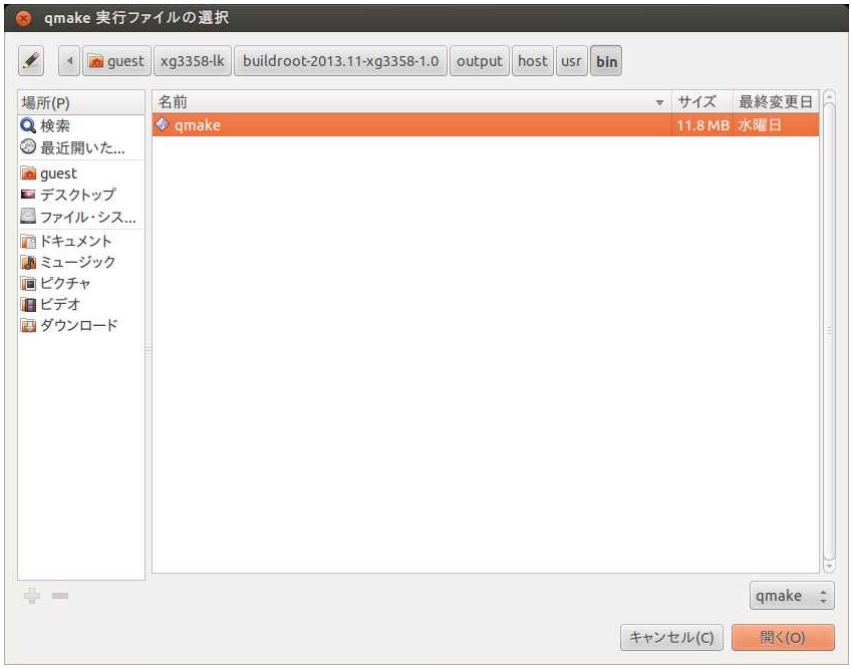


② 「ビルドして実行」を選択し「Qt バージョン」ページを開きます。
XG-3358 または XG-BBEXT の Qt4.8.5 が登録されていなければ「追加ボタン」をクリックします。
登録済みの場合は、手順⑤に進みます。

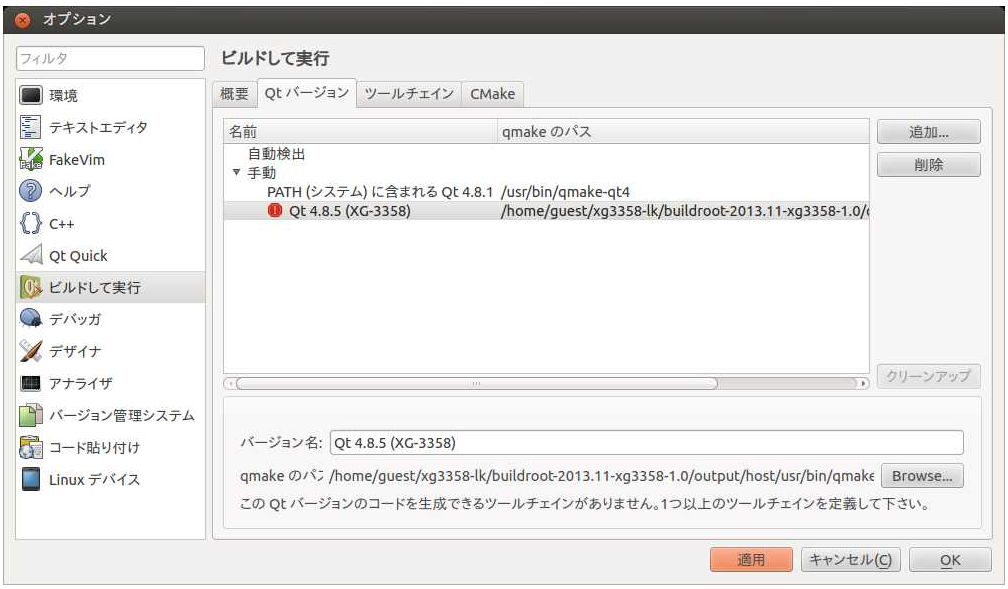


③　「qmake 実行ファイルの選択」ダイアログにて、XG-3358 または XG-BBEXT の Buildroot システムのホスト qmake を選択します。

機種（開発キット）	qmake のパス
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X/output/host/usr/bin/qmake
XG-BBEXT	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X/output/host/usr/bin/qmake



④　バージョン名を分かりやすいように「Qt 4.8.5(XG-3358)」または「Qt 4.8.5(XG-BBEXT)」に変更します。「この Qt バージョンのコードを生成できるツールチェインがありません。」と表示されますので、ページを「ツールチェイン」に切替えます。



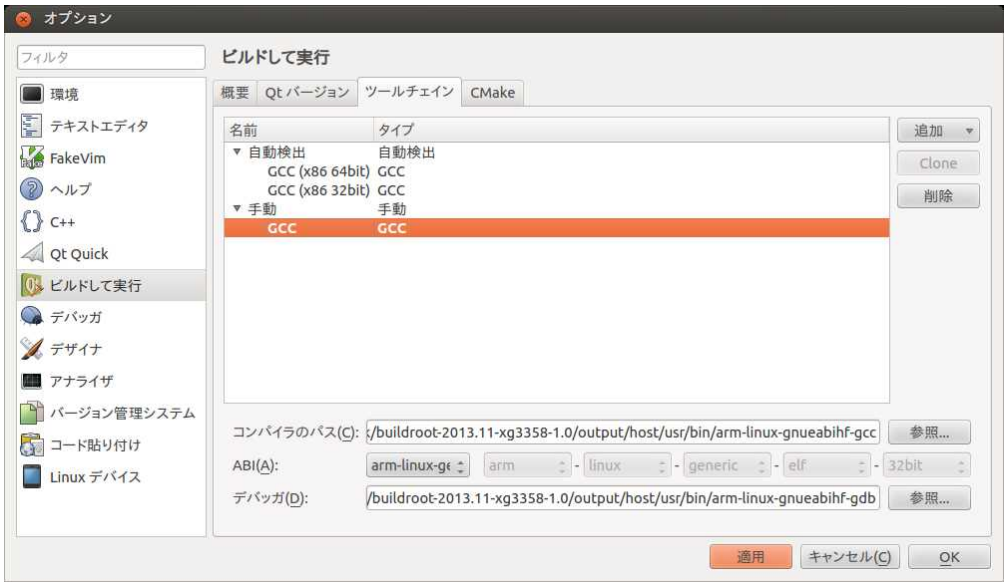
⑤　「ツールチェイン」ページにて追加ボタンをクリックし「gcc」を追加します。手動に追加された GCC 行を選択し、コンパイラのパス、デバッガを「参照」ボタンをクリックして入力します。「適用」ボタンをクリックしツールチェインの登録をし「Qt バージョン」のページに戻ります。

[XG-3358(LK-3358-A01)]

項目	コンパイラのパス
コンパイラ	~/xg3358-lk/buildroot-2013.11-xg3358-X.X/output/host/usr/bin/arm-linux-gnueabihf-gcc
デバッガ	~/xg3358-lk/buildroot-2013.11-xg3358-X.X/output/host/usr/bin/arm-linux-gnueabihf-gdb

[XG-BBEXT]

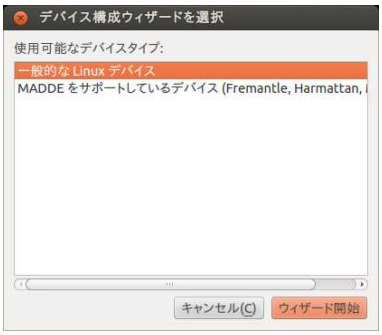
項目	qmake のパス
コンパイラ	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X/output/host/usr/bin/arm-linux-gnueabihf-gcc
デバッガ	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X/output/host/usr/bin/arm-linux-gnueabihf-gdb



⑥　「Qt バージョン」ページにて先ほどのエラーメッセージが消え「qmlviewer がインストールされていません」というメッセージだけが表示されていることを確認し、次に「Linux デバイス」を選択します。

⑦　「Linux デバイス」の「デバイス構成」のページにて「追加」ボタンをクリックします。

⑧　「デバイス構成ウィザードを選択」ダイアログが表示されたら「一般的な Linux デバイス」を選択し「ウィザード開始」ボタンをクリックします。



⑨　「新しい一般的な Linux デバイス構成の設定」ダイアログにて接続データを入力し「次へ」ボタンをクリックします。「設定が完了しました」と表示されたら「完了」ボタンをクリックし「デバイス構成」画面に戻ります。

項目	qmake のパス
この構成を識別する為の名前	XG-3358 または XG-BBEXT
デバイスのホスト名か IP アドレス	ターゲットボードの IP アドレス「192.168.128.200」
デバイスにログインする為のユーザ名	root
認証方式	パスワード
パスワード	default に設定したパスワード「xg3358&bbext」

⑩　ウィザードを閉じると自動的にターゲットボートとの接続確認をします。

[コネクションができたとき]

[コネクションができなかったとき]

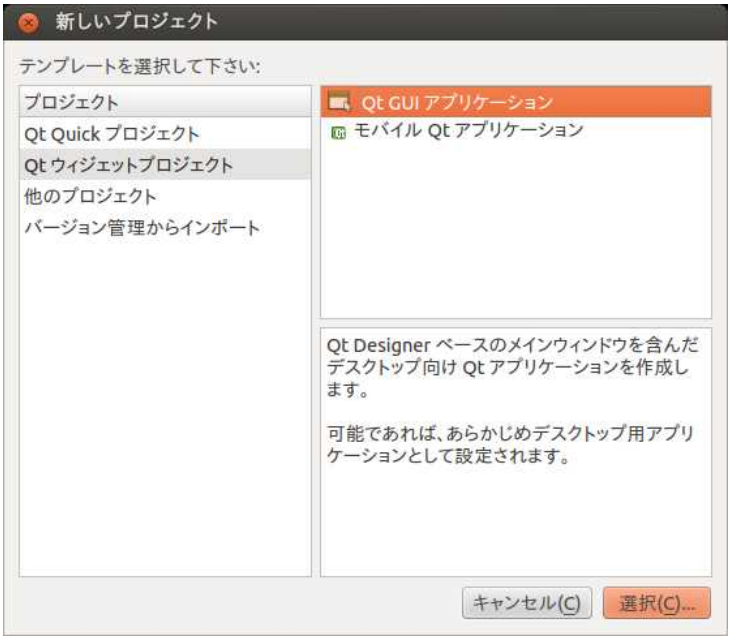
「デバイス構成」の「テスト」ボタンをクリックすれば再度接続テストができます。ターゲットの XG ボードとの接続が確認されましたら「OK」ボタンを押してオプション設定を終了します。

初めて接続するときなどは接続に時間が掛かることもあります。デバイステストが失敗した場合、他に原因が見当たらないようでしたら、タイムアウトの設定を大きな値にして再度テストしてください。

4.3　プロジェクトの作成

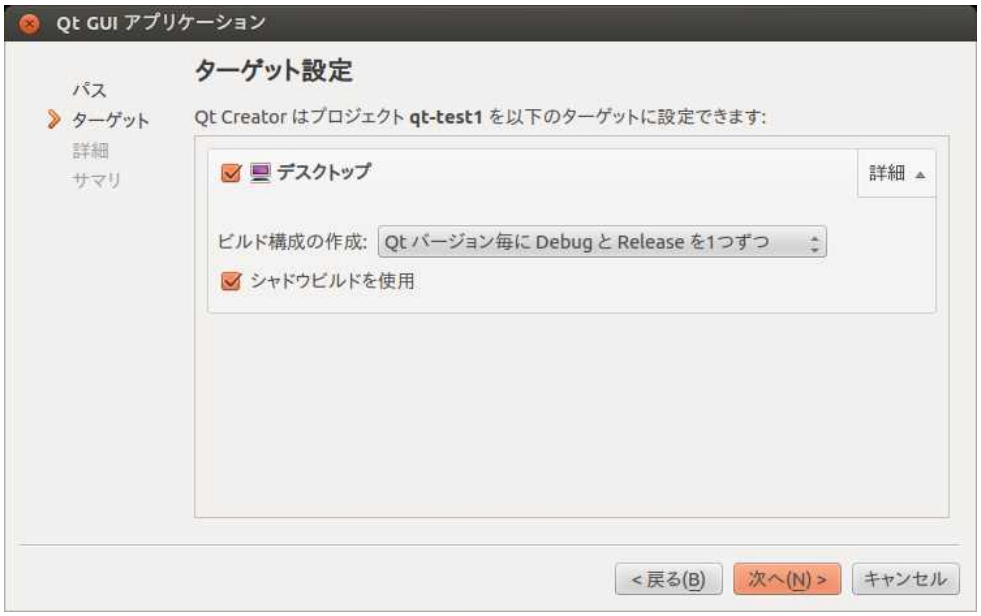
Qt アプリケーションを作成するには、最初に Qt プロジェクトを以下の手順で作成します。

① メニュー「ファイル」－「ファイル/プロジェクトの新規作成」を選択して「新しいプロジェクト」ダイアログを開きます。プロジェクトの中の「Qt ウィジェットプロジェクト」「Qt GUI アプリケーション」を選択し「選択(C)…」ボタンをクリックします。



② 名前に「qt-test1」、パスは参照ボタンをクリックしてディレクトリ「/home/guest/qt-work」を作成し「次へ」ボタンをクリックします。プロジェクトは「/home/guest/qt-work/qt-test1」に作成されます。

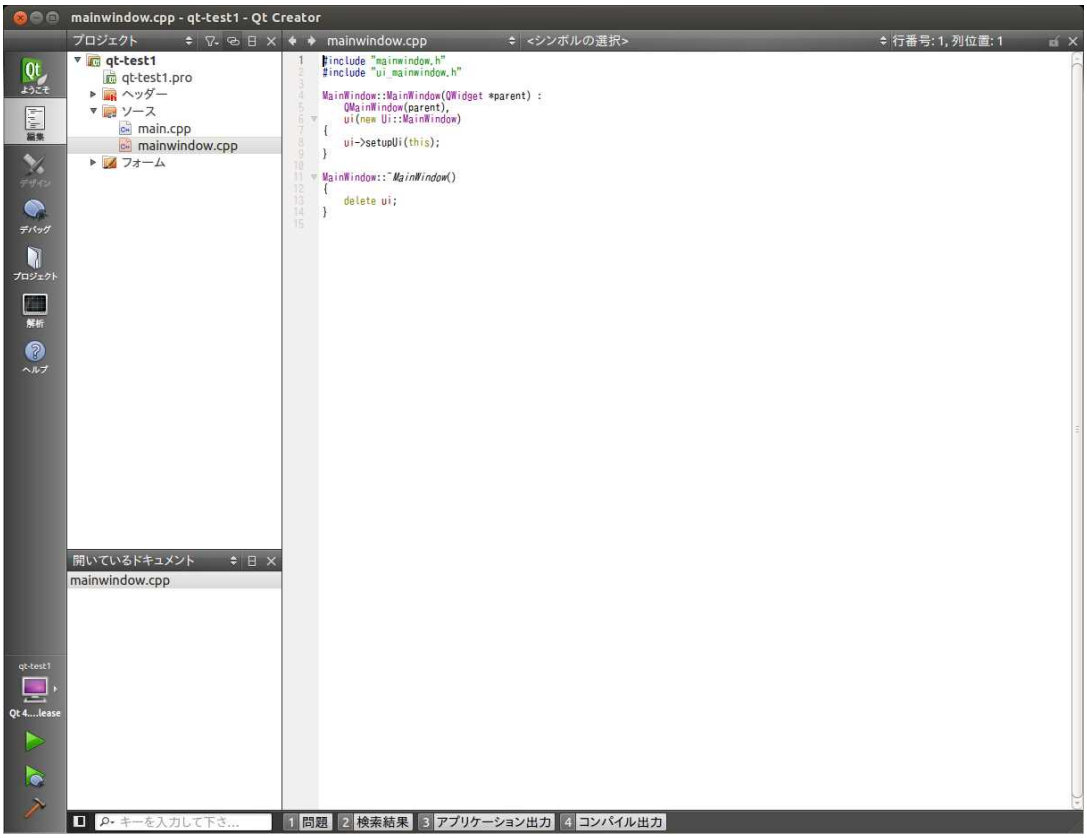
③　ターゲットの設定をします。　デスクトップのチェックを確認し、ビルド構成の作成は「Qt バージョンごとに Debug と Release を 1 つずつ」を選択します。



④　クラス情報にて、クラス名、ヘッダーファイル、ソースファイル、フォームファイルを設定します。ここではデフォルトのまま「次へ」ボタンをクリックします。。

④　プロジェクトに作成されるファイル一覧が表示されます。確認の上「完了」ボタンをクリックし、プロジェクトの作成を完了します。
バージョン管理システムを使用する場合は「バージョン管理システムに追加」項目でバージョン管理システムを選択してください。

⑤　qt-test1 プロジェクト　開発用に変わります。



4.4　プロジェクトの設定

Qt アプリケーションプロジェクトの設定をします。

①　プロジェクトをクリックし「ビルド設定」ページを選択します。

② 最初に Qt バージョンは「Qt 4.8.5(XG-3358)」を選択します。Qt バージョンの選択を終えると「ビルド構成を編集」は「Qt 4.8.5(XG-3358) Debug」または「Qt 4.8.5(XG-3358)Relase」になりますので「Qt 4.8.5(XG-3358) Debug」を選択します。

シャドウビルドが選択されていると、構成によってビルドされるディレクトリが別々になります。

③ ビルド後、出来上がった実行可能ファイルを/nfs ディレクトリに自動的にコピーするように設定します。
「ビルドステップを追加」ボタンをクリックすると、メニューが表れますので「独自プロセスステップ」を選択します。

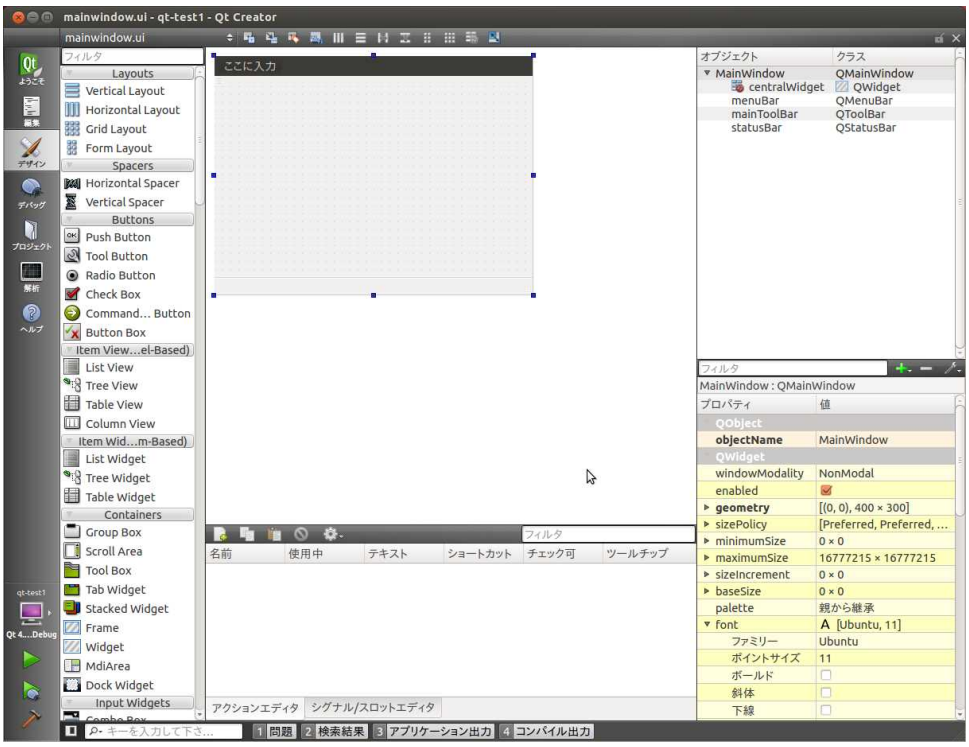
④　独自プロセスの「独自プロセスステップの有効化」にチェックを入れ、コマンドに「cp」、コマンド引数に「./qt-test1 /nfs」と入力します。

⑤　編集ボタンをクリックして、編集モードに戻します。

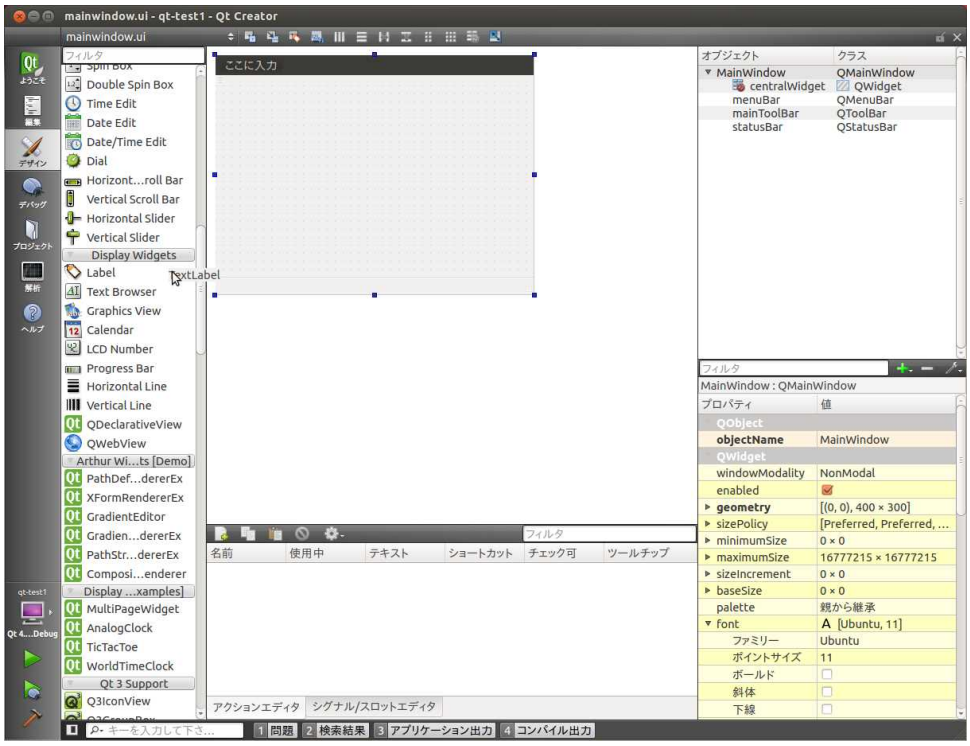
4.5　サンプル GUI アプリ作成

GUI のデザインは次のような手順にて行います。

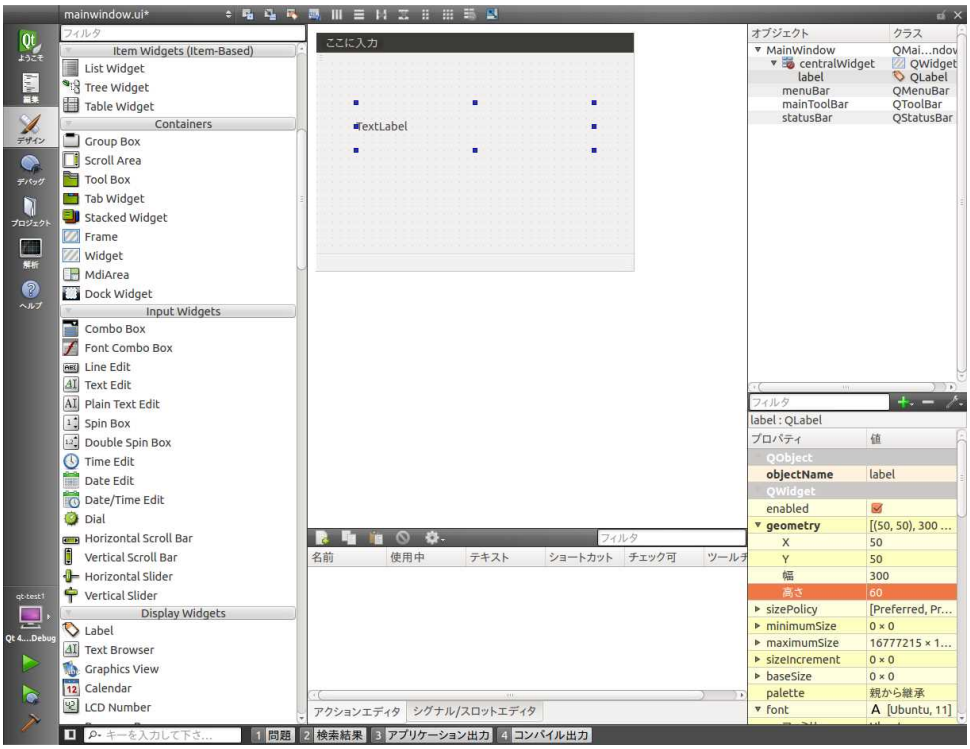
①　「qt-test1」－「フォーム」を展開し「mainwindo.ui」をダブルクリックし GUI デザイン用に切り替えます。



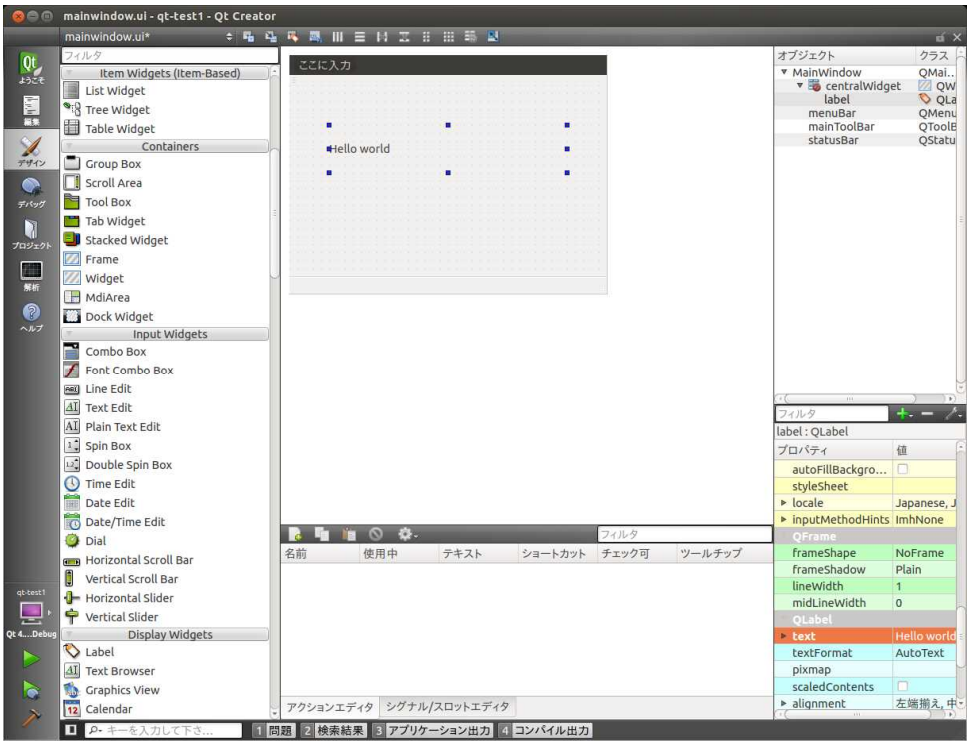
②　Display Widgets から Label をドラッグ＆ドロップでフォームに配置します。



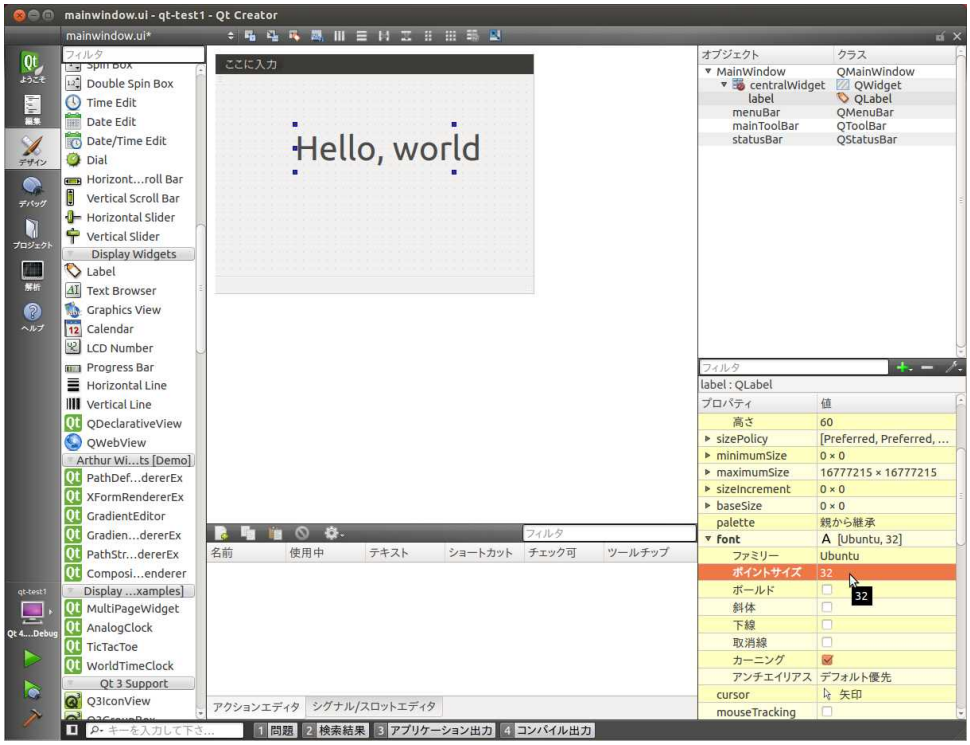
③　配置したラベルを選択し、プロパティの「QWidget」－「geometry」を展開し、 X に 50、Y に 50、幅に 300、高さに 60 をセットします。



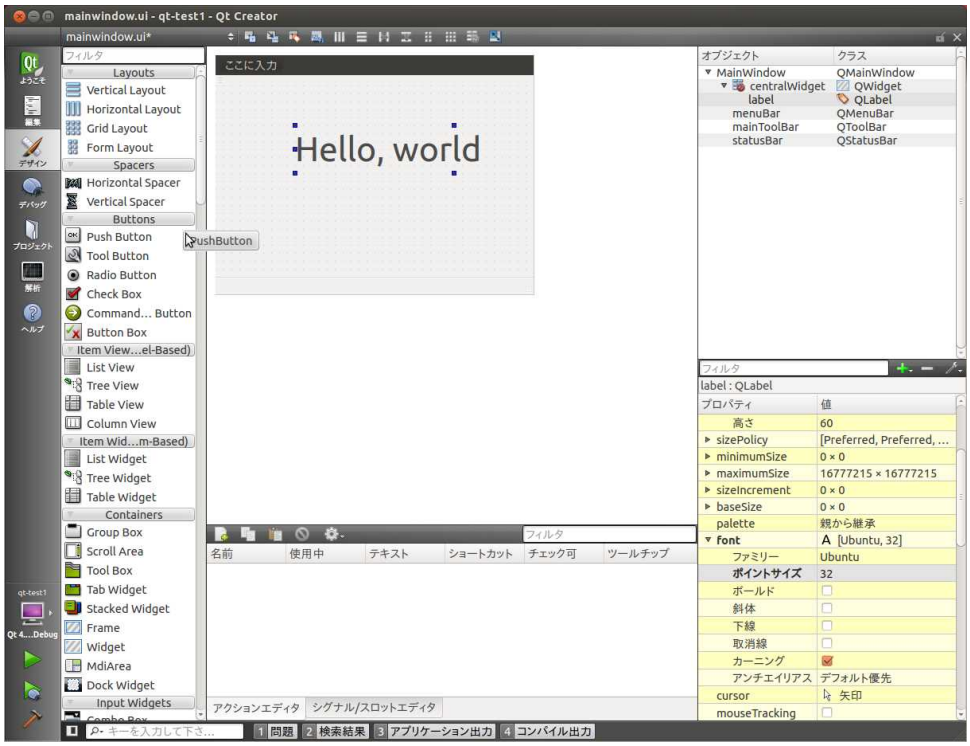
④　プロパティで「QLabel」－「text」に「Hello, world」と入力します。



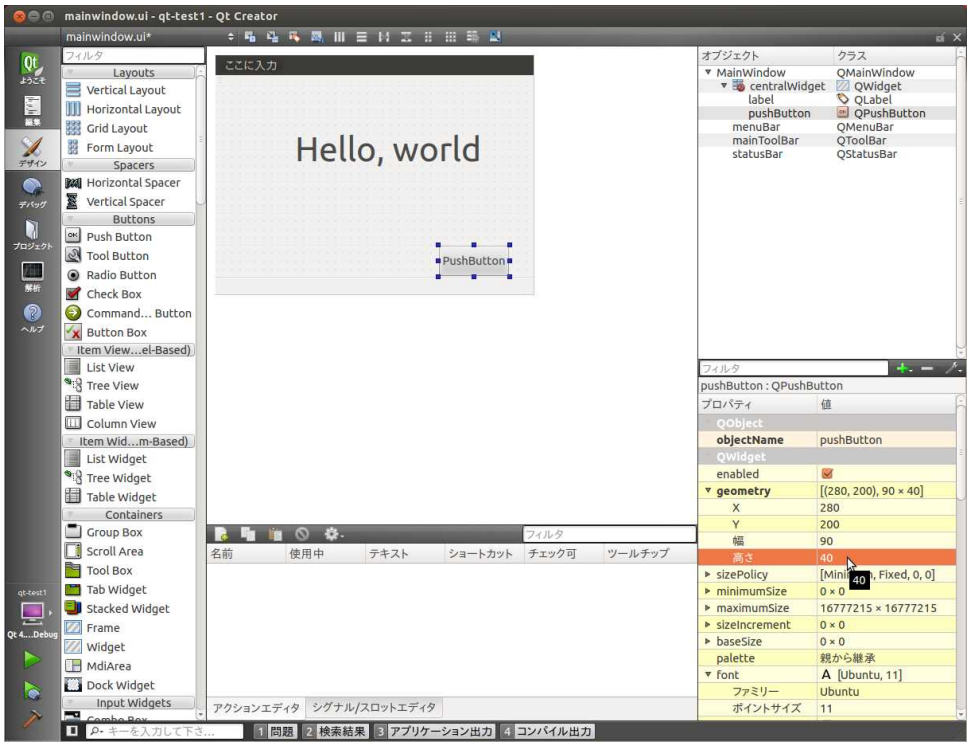
⑤　プロパティの「QWidget」－「font」－「ポイントサイズ」を 32 にします。



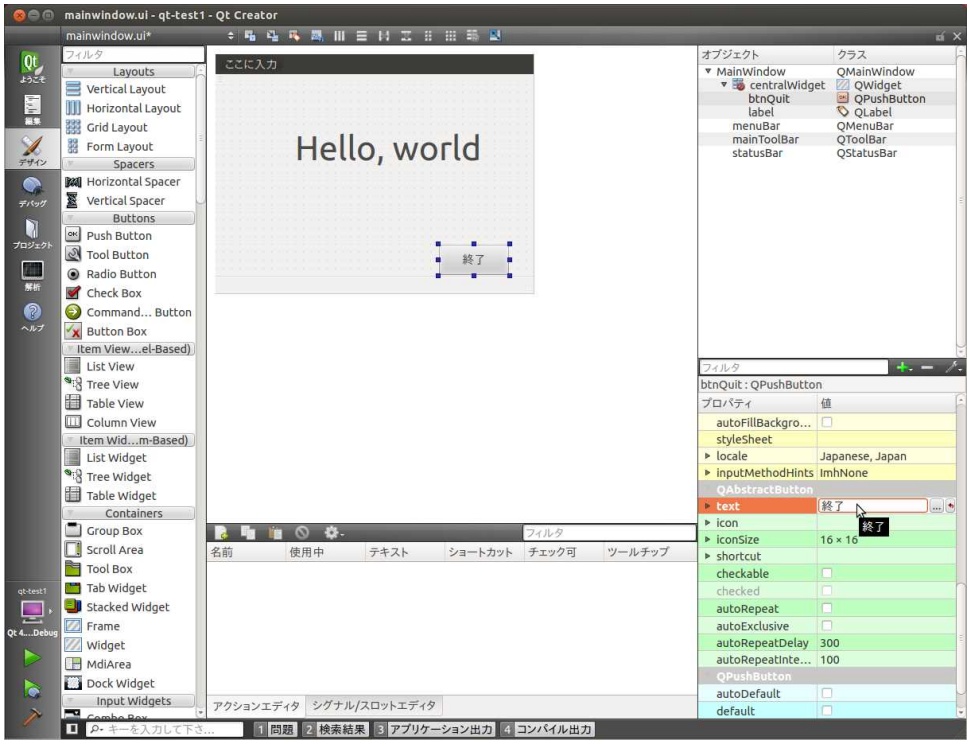
⑥　次にアプリケーションの終了ボタンを追加します。Buttons の「Push Button」をフォームの右下端にドラッグ＆ドロップで配置します。



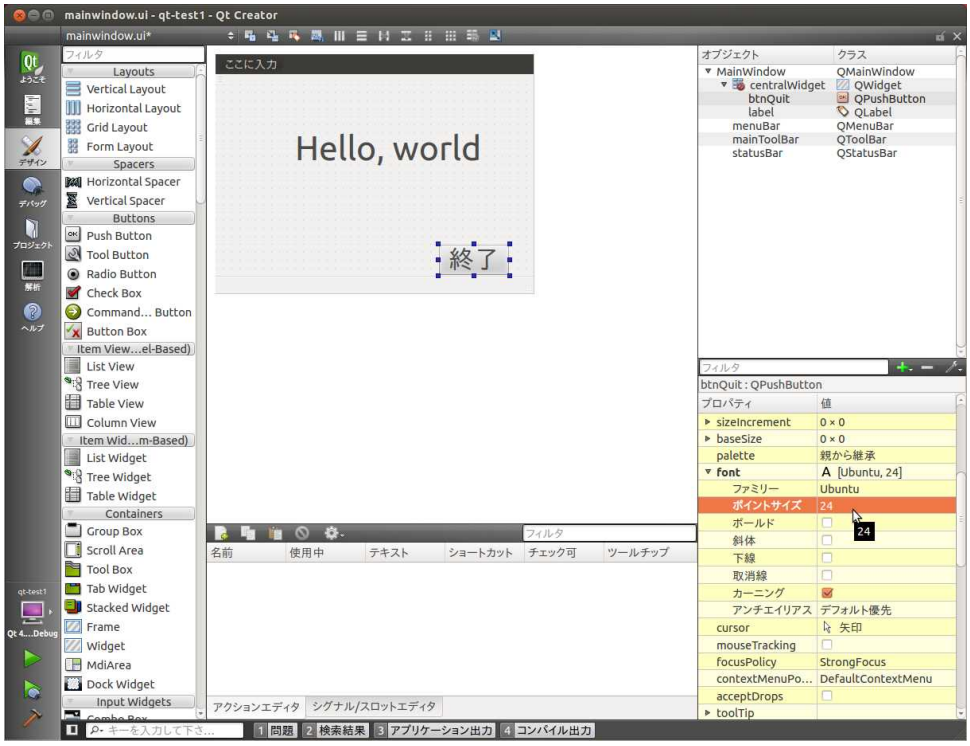
⑦　配置したラベルを選択し、プロパティの「QWidget」－「geometry」を展開し、 X に 280、Y に 200、幅に 90、高さに 40 をセットします。



⑧　プロパティにて QObject の Object Name を「btnQuit」に、QAbstructButton の Text プロパティを「終了」にします。



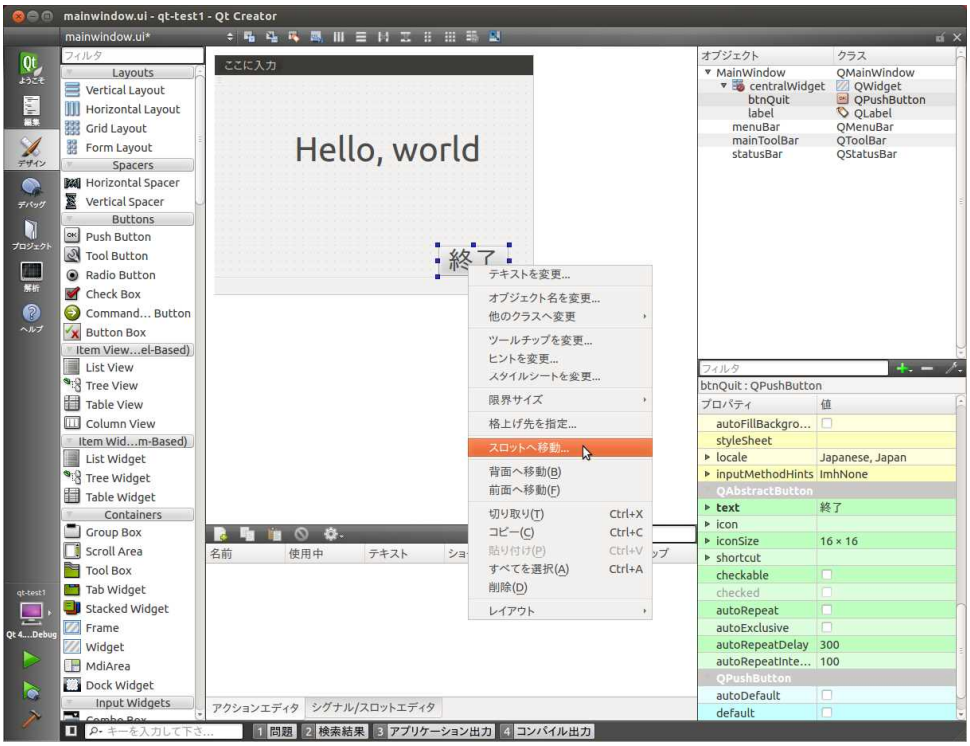
⑨　プロパティの「QWidget」－「font」－「ポイントサイズ」を 24 にします。



画面上で同じフォントを利用する場合、デフォルトとして登録しておくと便利です。

MainWindow の font プロパティにフォントを設定しておけば、MainWindow 上に配置された font プロパティを持った widgets のデフォルトは Mainwindow の font プロパティが用いられます。

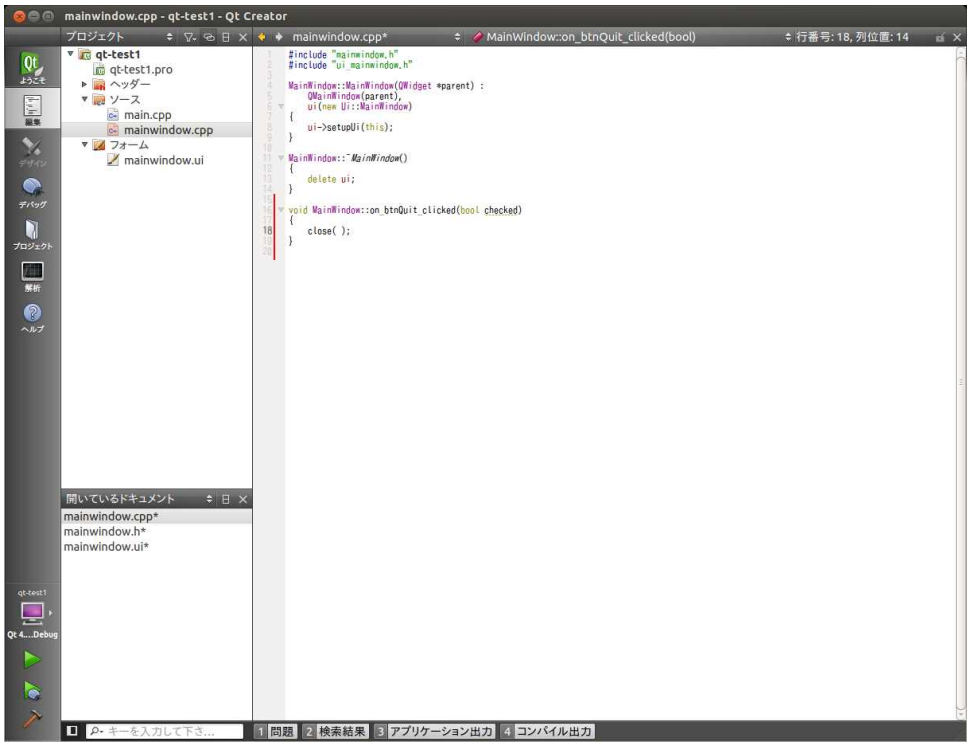
⑩　Quit ボタンをタップしたときに、qt-test1 アプリケーションが終了するようします。そのためには btnQuit にてマウス右ボタンをクリックしポップアップメニューの「スロットへ移動…」を選択します。



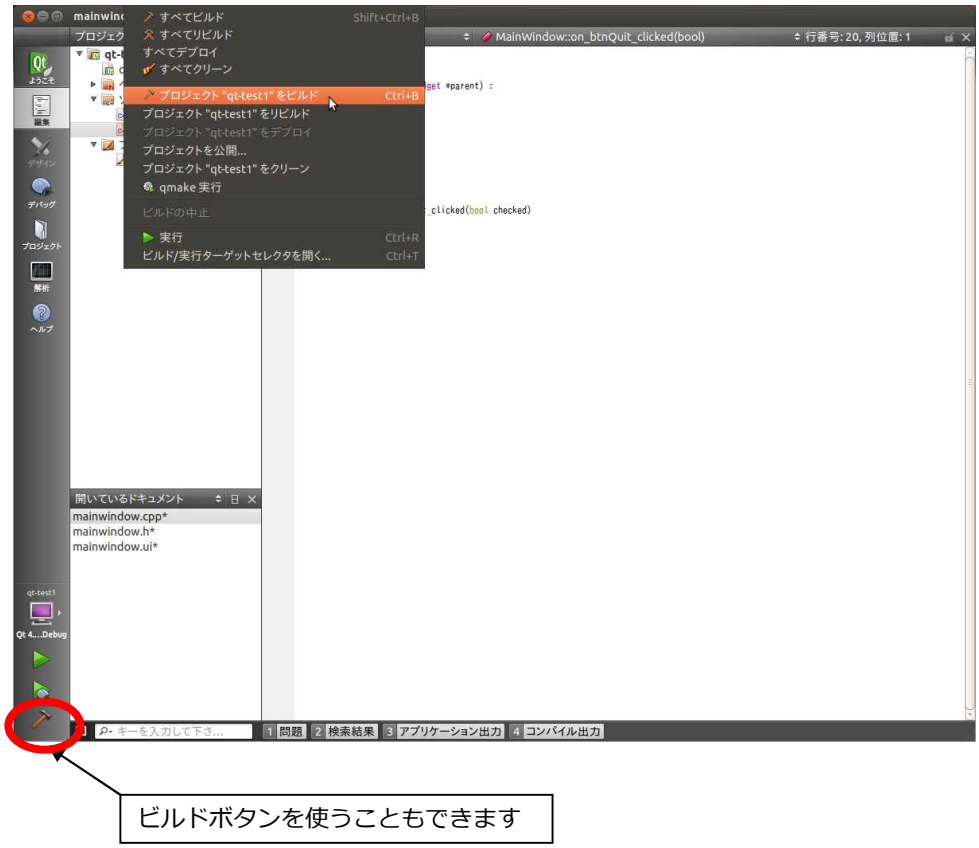
⑪　ここでは単にタップ（クリック）するだけなので、「clicked()」を選択し「OK」ボタンをクリックします。

⑫　自動的にコードエディタに切り替わり「MainWindow::on_btnQuit_clicked()」が自動的に生成されますので、その中に実際の処理のコードを記述します。ここではアプリケーションを終了するために、以下のコードを入力します。

```
close();
```



⑬　Qt アプリケーションのビルドは、Qt Creator メニュー「ビルド」－「プロジェクト"qt-test1"をビルド」を選択しビルドを開始します。

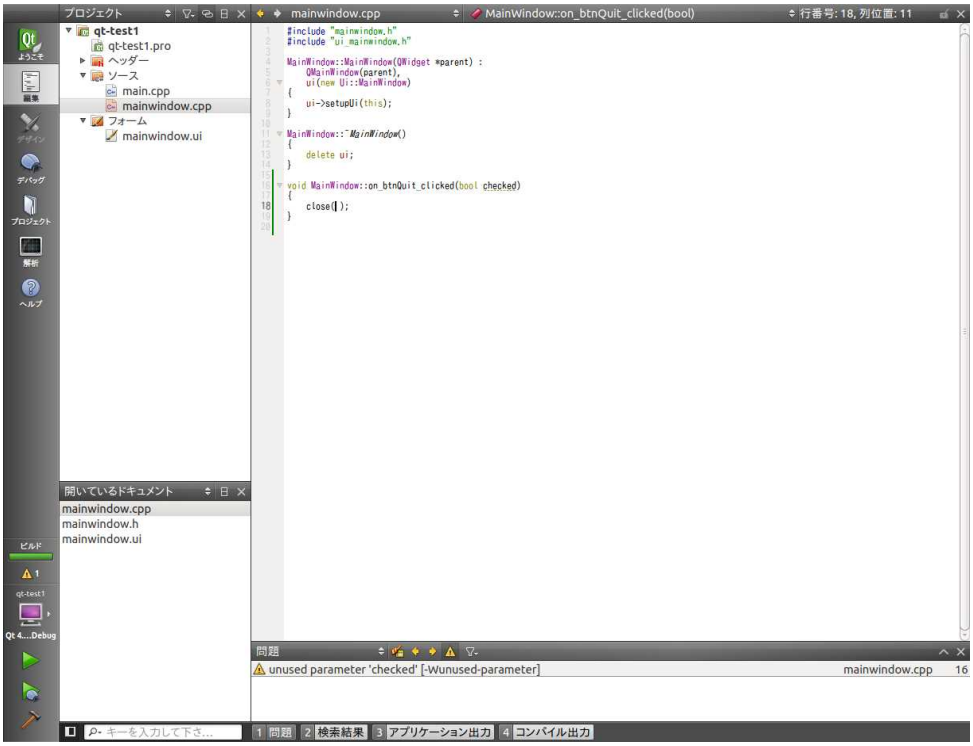


保存してないファイルがあるときは、以下のように問い合わせがあります。その場合は「すべて保存」をクリックします。「ビルド前にすべてのファイルを保存する」にチェックを入れておけば自動的に保存されてからビルドされるようになります。

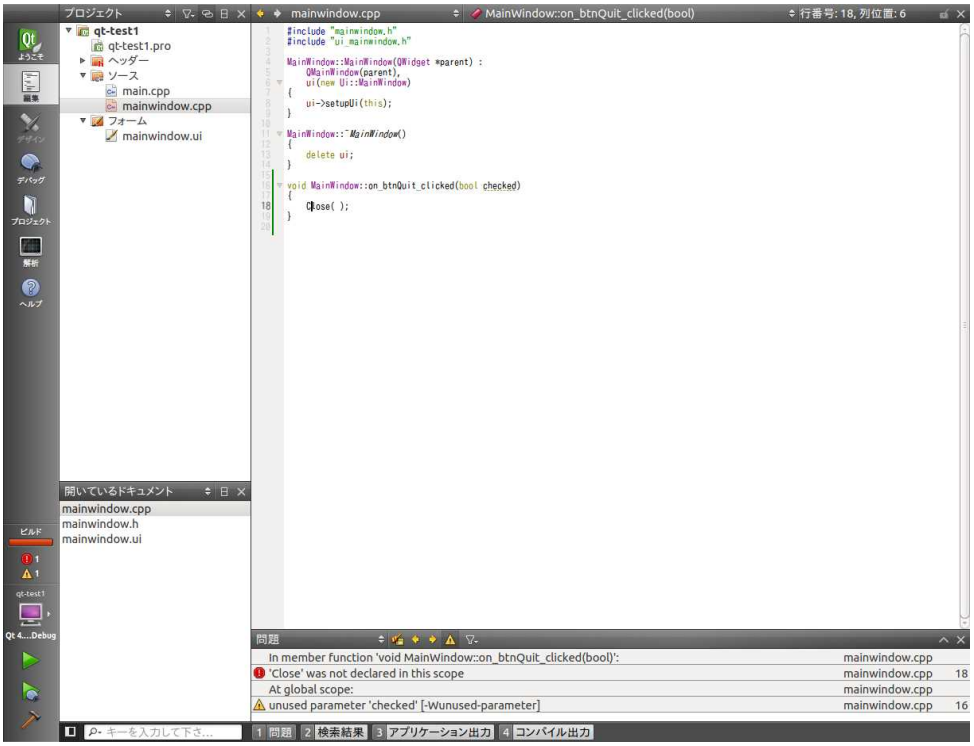
ビルドが正常に終了すると、ディレクトリ「/nfs」に「qt-test1」がコピーされます。

ビルドが始まるとビルド進行インジケータが表示されます。ビルドの結果エラーが無いときは次のように数秒間画面左下のビルドの部分が緑色になります。

問題に「unused parameter 'checked'[-Wunused-paramter」とウォーニングが出ていますが、これは終了ボタンを押したときの処理「on_btnQuit_clicked」で check 引数を使用してないためです。



エラーがある場合は、ビルド進行インジケータが赤色になり、次のようにエラーメッセージが表示されます。



4.6　サンプルの実行

ビルドが正常に終わると開発Linuxサーバの/nfsディレクトリに実行可能なアプリケーションファイルをコピーされています。
ターゲット側の XG ボードで NFS マウントしてあればそのまま、/mnt/nfs ディレクトリに移動して実行する事ができます。

①　開発用 PC の NFS サーバに接続してないときは、mount します。

/etc/init.d に「K95debug」スクリプトがある場合は

```
# /etc/init.d/K95debug start      入力
```

コマンドで直接 NFS マウントする場合は以下のようにします。

```
# mount -t nfs -o nolock 192.168.128.210:/nfs /mnt/nfs      入力
```

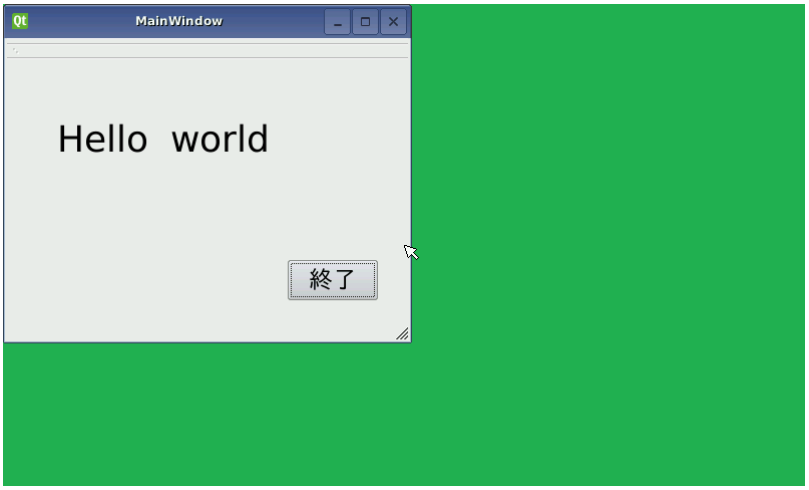
②　qt-test1 が存在することを確認します。

```
# cd /mnt/nfs      入力
# ls qt-test1      入力
qt-test1
```

③　qt-test1 を実行します。

```
# ./qt-test1 -qws      入力
```

④　次のような画面が表示され「終了」ボタンをタッチすると終了します。



5　リモートデバッガ

5.1　リモートデバッガの設定

XG-3358 または XG-BBEXT 上で稼動する Qt アプリケーションをデバッグできるようにするための設定を行います。

① 「プロジェクト」－「実行時の設定」ページに切替えます。初期状態ではデプロイ、実行ともリモートでバッグ用に設定されていません。

② デプロイ（配布）の追加をします。

「デプロイ」-「メソッド」の「追加」をクリックして「リモート Linux ホストにデプロイ」を選択します。

「名前を変更」をクリックして名前を「XG-3358 にデプロイ」または「XG-BBEXT にデプロイ」にします。

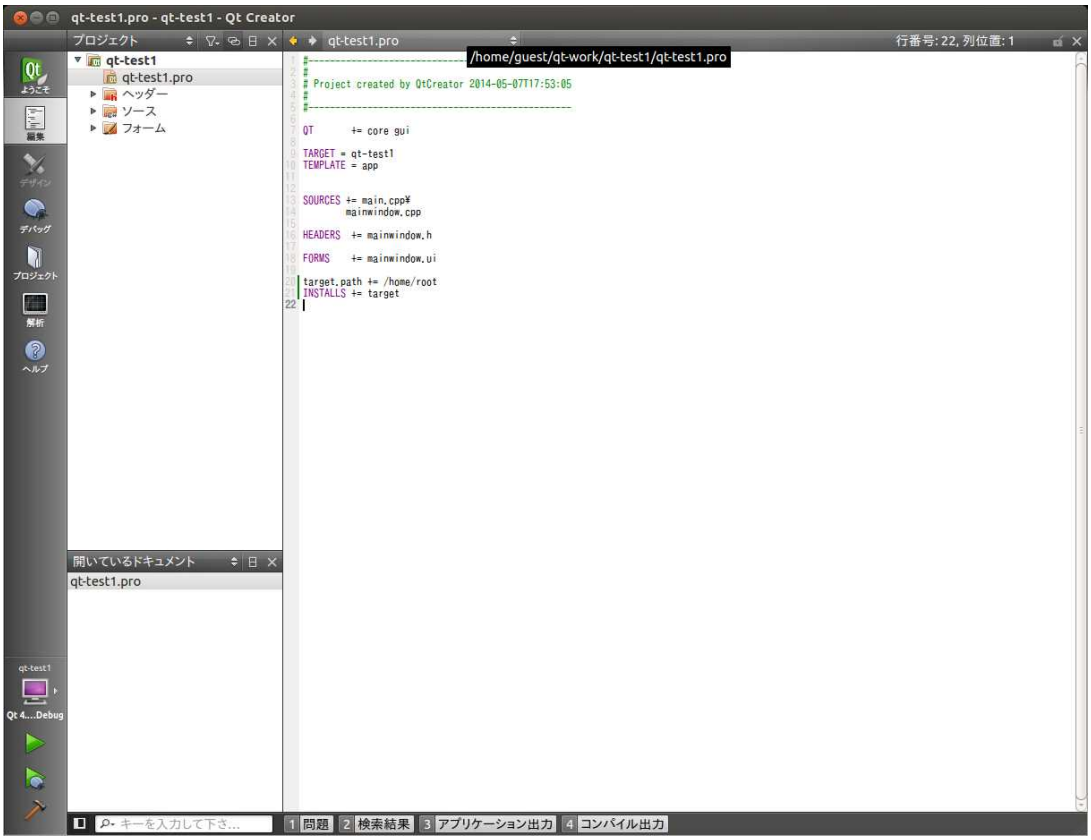
デプロイ設定をするとビルドが成功すればターゲットボードに実行ファイルを自動的に配布することができます。

「4.5　サンプルの実行」で行った Ubuntu 側の/nfs ディレクトリに実行ファイルをコピーして、ターゲットボード側で NFS マウントして実行する必要はなくなります。
実行ファイル以外に沢山の画像、サウンドファイルやデータファイルなどを配布する必要がある場合は、NFS を利用した方が便利ですので、デバッグ環境にあわせて NFS の使用は検討してください。

③　ターゲットボードの配布先ディレクトリを指定します。
「<ターゲットパス未設定>」となっておりマウスカーソルを近づけると「項目の追加・削除にはプロジェクトファイルを編集して下さい。」とヒントが表示されます。「編集」にてプロジェクトファイル qt-test1.pro をクリックしてプロジェクトファイルを開き、ターゲットにデプロイするパスを指定するために以下の 2 行を追加します。

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

プロジェクトファイルの編集が終わりましたらメニュー「ファイル」-「"qt-test1.pro"の保存」でプロジェクトファイルを保存します。



プロジェクトに戻ると先ほど赤字で出ていた「<ターゲットパス未設定>」は「/root」に変更されています。

④　リモートでの実行環境の設定をします。
「実行構成」の「追加」をクリックし「qt-test1(リモートの一般的な Linux ホスト上)」を選択します。
「名前を変更」をクリックして「qt-test1 (リモートデバイス上)」を「XG-3358」または「XG-BBEXT」に変更します。
Qt アプリケーションは qws サーバが無いと実行できませんので、ターゲットボードですでに qws サーバが動いている場合を除き、引数に「-qws」を追加します。

以上でリモートデバッグの設定が完了しました。

5.2　リモートデバッグ

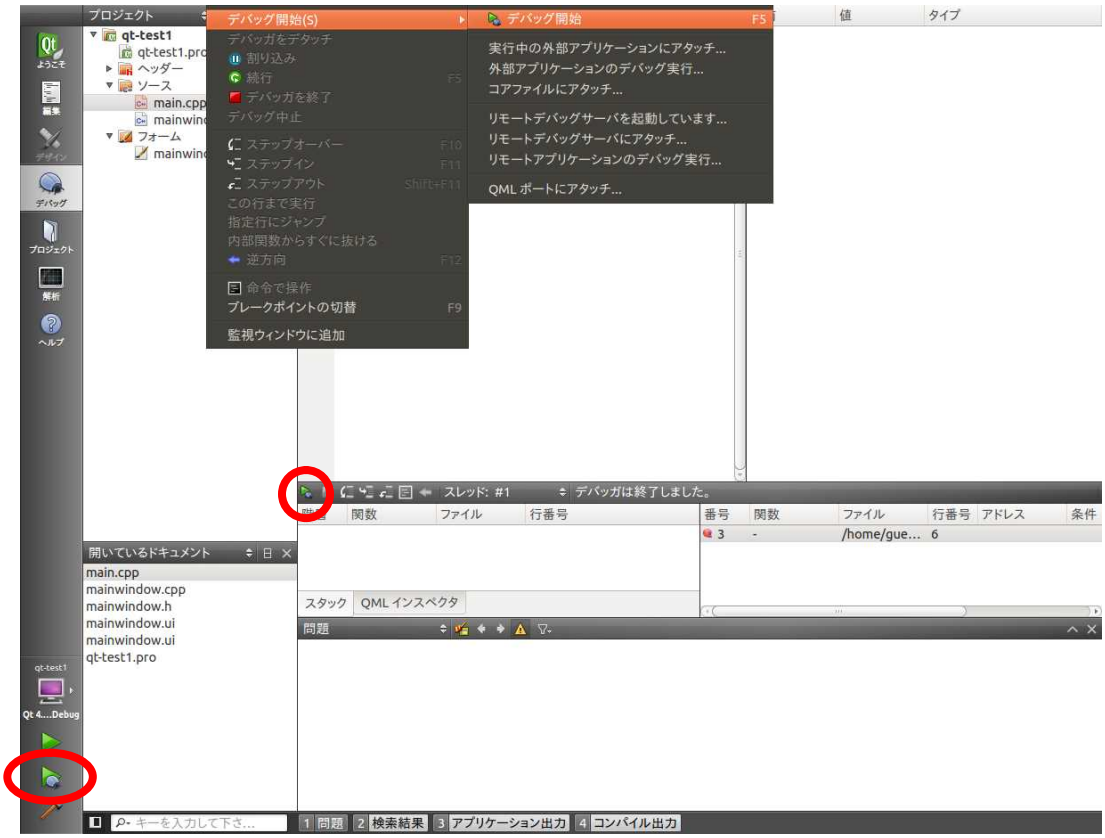
① デバッグを開始します。

デバッグを開始するには以下の方法があります。

a. メニューの「デバッグ」－「デバッグ開始」－「デバッグ開始」でデバッグを開始します。

b. 「デバッグ」画面に切替えて、デバッグ開始ボタンをクリックします。

c. 画面左下のデバッグ開始ボタンをクリックします。



デバッグ関係のスピードボタンには次のものがあります。

左から、

実行／続行
デバッガを終了
ステップオーバー
ステップイン
ステップアウト
命令操作モード切替

となっています。

またデバッグメニューにはスピードボタンにある機能以外のものも登録されています。

ブレークポイントの設定／解除は行番号の左側をクリックします。

6.　Qt プログラミング Tips

6.1　フォントファミリの指定

Qt Creator を使って GUI デザインするときにフォントファミリ指定で困ることがあります。たとえば XG-3358 また XG-BBEXT ターゲットボードに IPA の明朝とゴシックフォントをインストールした場合、GUI デザイン設計においてウィジェットのフォントごとに明朝、ゴシック、明朝（プロポーショナル）、ゴシック（プロポーショナル）を指定したいことがあります。

通常フォントファミリは、プロパティの「font-ファミリ」で設定できますが、Ubuntu 上の QtCreator では選択するフォントファミリー覧には日本語のファミリ名しか列挙されません。そしてそのまま日本語のフォントファミリ名を選択、XG ターゲット用の実行ファイルを生成して XG ボードで動かすと期待したフォントで表示されないことがあります。

なぜならば XG ボード側ではフォントファミリは英字のみで検索されるからです。そのため日本語フォントファミリ名で指定してあるとフォントが見つからないのでデフォルトフォントが使用されます。困ったことに QtCreator では「font-ファミリ」にはフォントファミリ名を直接入力することができません。したがって回避方法としては一旦日本語のフォントファミリ名を選択し Qt Creator を終了した後、プロジェクト内の「*.ui」ファイルを編集します。

「IPA P 明朝」を指定したときの mainwindow.ui ファイル

```
<property name="font">
 <font>
  <family>IPA P 明朝</family>
  <pointsize>36</pointsize>
 </font>
</property>
```

XG-3358、XG-BBEXT のフォントファミリ名に変えるため以下のように編集します。

```
<property name="font">
 <font>
  <family>IPAPMincho</family>
  <pointsize>36</pointsize>
 </font>
</property>
```

再度、QtCreator を起動しプロジェクトを開くと、Ubuntu 側に該当フォントがインストールされていれば指定されたフォントで表示されます。フォントファミリ名は英字が基本ですので QtCreator や LibreOffice で表示される日本語のフォント名で指定しなくても、QtCreator 上では同じフォントで表示されます。

フォントファミリ名を知りたいときは、フォントファイルをクリックするとフォントビューワーが開きフォントの表示例と明細が表示されます。フォントファミリ名は Name 欄に記載されています。

フォントファイルは、Qt の場合、次のディレクトリに保存されています。
~/xg3358-lk/buildroot-2013.11-xg3358-X.X/output/target/usr/lib/fonts
または
~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X/output/target/usr/lib/fonts

Ubuntu にこのフォントがインストールされていない場合は、右下の「Install Font」ボタンをクリックすることによってインストールができます。

7.　関連情報

Qt Creator に関しての情報は以下の書籍および、サイトが参考になります。

名前	著者	出版社
実践 Qt プログラミング	Mark Summerfield	オライリージャパン
入門 Qt 4 プログラミング	Jasmin Blanchette 他	オライリージャパン
Qt Quick ではじめる クロスプラットフォーム UI プログラミング	折戸 孝行	アスキー書籍

Table 7-1 参考書籍

サイト名	URL
digia	http://www.digia.com
Qt	http://qt.digia.com
Qt wiki(日本語)	http://qt-project.org/wiki/Wiki_Home_Japanese
SRA Qt	http://www.sra.co.jp/qt/licence/
Sitara Linux Training: Hands on with QT	http://processors.wiki.ti.com/index.php/Sitara_Linux_Training:_Hands_on_with_QT

Table 7-2 参考 Web サイト

ご注意

商標について

ALPHA PROJECT Co.,LTD.
株式会社アルファプロジェクト
〒431-3114
静岡県浜松市東区積志町 834
http://www.apnet.co.jp
E-MAIL : query@apnet.co.jp