

XG-335x

Qtのためのルートファイルシステム作成方法

Rev1.0 2014/11/12

目次

1. 概要	1
1.1 はじめに.....	1
1.2 対象機種（開発キット）.....	1
1.3 Qtとは.....	1
1.4 Qtを使ったアプリケーションの稼動環境を作成するには.....	2
2. Qtの概要	3
2.1 システム概要.....	3
2.2 Qtシステム構成.....	3
3. ルートファイルシステム	4
3.1 Qtルートファイルシステム作成のための準備.....	4
3.2 日本語(IPA)フォントのインストール.....	6
3.3 設定ファイルの準備.....	9
3.4 ルートファイルシステムのカスタマイズ.....	11
3.5 WebKitに対応するためのカスタマイズ.....	18
3.5 ルートファイルシステムのビルド.....	21
4. SDルートファイルシステム	22
4.1 XG-3358におけるSDルートファイルシステムの作成.....	22
4.2 XG-3358におけるSD-Linuxシステムの起動.....	23
4.3 XG-BBEXTにおけるルートファイルシステムの作成.....	24
5. 動作確認	25
5.1 環境変数の確認.....	25
5.3 タッチパネルキャリブレーション.....	27
5.4 Qtの動作確認.....	28
5.5 sshの動作確認.....	30
6. アプリケーションの自動実行	32

表記

●バージョンに関する表記

弊社提供のソース等に関しては、弊社の管理するバージョン番号がファイル名やフォルダ名に付いている場合があります。そのバージョン番号に関しては、本ドキュメントでは、『X』を使用して表現しております。そのため、以下のような表記になりますので、その部分は読み替えてください。

例：

以下の表記がある場合

```
helloworld-X.X.tar.bz2
```

Ver1.0 での実際のファイル名は、以下になります。

```
helloworld-1.0.tar.bz2
```

●コマンドラインの表記

本ドキュメントには、コマンドラインで入力する操作手順が記載されております。操作は PC 及び XG ボードで行います。それぞれの記述について以下に記載します。

ゲスト OS(Ubuntu)での操作


プロンプトは、『\$』で記載します。

実際のプロンプトには、カレントディレクトリ等が表示されますが、本ドキュメントでは省略します。

なお、省略時には、コマンドプロンプトの前に、**省略** と表記します。

XG ボード上の Linux での操作


プロンプトは、『#』で記載します。

本ドキュメント中での入力では、以下のように表現し、入力の最後には、 があります。


例：ゲスト OS(Ubuntu)上で make コマンドを実行する場合の表記

```
省略 $ make 
```

コマンドによっては 1 つのコマンドが複数行で記載されている場合もあります。

その場合には、2 行目以降の入力では ENTER キーを押さずに続けて入力し、 の表記がある行の最後で ENTER キーを入力してそのコマンドを実行してください。

例：2 行続いてコマンド入力がある表記

```
省略 $ mkimage -A arm -O linux -T ramdisk -C gzip -d output/images/rootfs.cpio.gz output/i  
mages/uInitrd-xg3358 
```

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/Mac 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 シリーズに Qt の動作環境を構築する方法を説明します。



本ドキュメントでは、Ubuntu12.04LTS を含めた開発環境が PC にインストールされていることが前提となっています。開発環境をインストールされていない場合は、『Linux 開発 インストール マニュアル』に従って、先に開発環境の作成を行ってください。

また、XG-3358 では LCD-KIT に対応したカーネルも必要となります。カーネルの作成方法に関しては、各開発キットの『ソフトウェアマニュアル』で説明していますので、本ドキュメントを読む前にそちらのマニュアルも一通り行ってから本ドキュメントをお読みください。

1.2 対象機種（開発キット）

本アプリケーションノートは、以下の機種（開発キット）のバージョンを対象とします。

機種（開発キット）	バージョン
XG-3358(LK-3358-A01)	Rev1.1 以降
XG-BBEXT	Rev1.1 以降

バージョンは、付属 CD もしくは DVD のバージョン番号で表記しております。各ファイルの更新内容に関しては、機種（開発キット）の更新履歴をご確認ください。

1.3 Qt とは

Qt は一般的に GUI 開発ツールとして有名です。しかし、GUI 開発以外の機能も豊富であり非 GUI プログラムでも利用されています。また、Web アプリケーションなどでは、LCD などの表示機能がないシステムでも、クライアントのブラウザ向けに画像を生成するために利用することもあります。ここでは GUI アプリケーションを動かすための Qt のためのルートファイルシステム構築について説明します。

Qt を使用するに当たっては幾つかの注意が必要です。

第一に Qt の利用には、ライセンスについて注意が必要です。現時点で Buildroot の Qt パッケージをインストールした場合、Qt のバージョンは 4.8.5 となります。Qt4.5 以降は LGPL が利用できます。GPL/LGPL では、不都合な場合は SRA などの Qt のパートナー会社と契約を結ぶ必要があります。ライセンスの詳細は SRA のサイトの Qt ライセンスを参照してください。

第二に Qt のドキュメントやディレクトリのパス名、ファイル名、開発ツールのマクロ定義などに Trolltech、Qtopia など旧名称と新名称が混在しています。たとえば Qtopia は現在 Qt Extended となっています。各種ドキュメントや Qt を利用しているシステム、アプリケーションなどにおいて新旧の名称等で混乱することがありますので注意してください。

1.4 Qt を使ったアプリケーションの稼働環境を作成するには

Qt の稼働環境を作成するには、Qt のパッケージを選択しルートファイルシステムの作成をします。出来上がったルートファイルシステムを microSD などへ格納します。ここでは、例として microSD にルートファイルシステムを構築することにします。

ルートファイルシステムの作成において少量生産、実験装置など開発環境 = 最終出荷環境の場合を除けば、開発環境とは別に余分な機能をカットした構成で出荷環境用のルートファイルシステムを作成することがあります。出荷環境によっては対象メディアにルートファイルシステムが収まるかどうかを充分検討しておく必要があります。

2. Qt 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、ここでは一番コンパクトでオーバーヘッドの少ない framebuffer のみの構成は、以下となります。

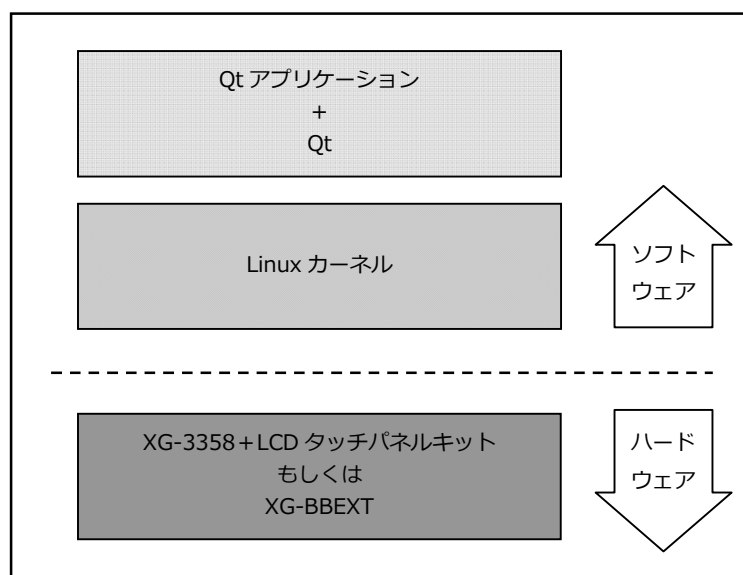


Fig 2.2-1 Qt システム例

以降では、上記構成で作成する方法を説明します。

3. ルートファイルシステム

Qt アプリケーションを稼動するためのルートファイルシステムを構築する方法は、以下の手順で行います。

- a) ルートファイルシステムを構築するため、初期化をします。
- b) スケルトンファイルシステムに、日本語フォントをインストールします。
- c) スケルトンファイルシステムに、Qt のための環境変数、および必要となるサーバ等を起動するスクリプトを用意します。
- d) buildroot で Qt 用のルートファイルシステムをビルドします。
- e) 作成されたルートファイルシステムをもとに、XG ボードにルートファイルシステムをインストールします。

ルートファイルシステムのスケルトンは、ルートファイルシステムをビルドするときの構造の基礎となる部分です。この部分に Qt で必要となるファイル、フォントをあらかじめセットしておきます。



microSD などの書き換え可能なデバイス上にルートファイルシステムを構築する場合には、後で書き換えることもできますが、RAM 上のルートファイルシステム等の場合は、あらかじめスケルトンによって構築する方法が有効です。

3.1 Qt ルートファイルシステム作成のための準備

ルートファイルシステムを Qt 用にビルドする準備として次の手順を進めます。

- ① フォントなどのダウンロードするための作業用フォルダを作成します。

```
省略 $ mkdir ~/qt-work ←入力
```

- ② output/target ディレクトリ下に配置したファイル、および変更したファイルがありましたら、必要に応じてバックアップを取ります。

- ③ ビルドルートシステムのディレクトリに移動します

```
省略 $ cd ~/xg3358-lk/buildroot-2013.11-xg3358-X.X ←入力
```

上記は、LK-3358-A01 の時のディレクトリ指定となります。移動先ディレクトリは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのディレクトリ名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

機種（開発キット）	ディレクトリ
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X
XG-BBEXT	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X

- ④ ルートファイルシステムを再構築するため、一度ビルドしてある output 下のディレクトリ、ファイルを以下のコマンドで削除します。

```
省略 $ make distclean ←入力
```



「make distclean」および「make clean」を実行すると、output ディレクトリ内のディレクトリ、ファイルが削除されてしまいますので、ご注意ください。

- ⑤ XG シリーズの buildroot のコンフィグレーションを行います。

```
省略 $ make xg3358_defconfig ←カ
```

上記は、LK-3358-A01 の時のコマンドとなります。コマンドは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのコンフィグレーション名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

機種（開発キット）	コンフィグレーション名
XG-3358(LK-3358-A01)	xg3358_defconfig
XG-BBEXT	xgbbext_defconfig



「make distclean」は「make clean」に加えて、Makefile, config.cache などのファイルも削除されます。Makefile のコンフィグレーション設定を変更した場合は「make distclean」を実行する必要があります。

3.2 日本語(IPA)フォントのインストール

- ① 日本語フォントをダウンロードします。

IPA フォントは「<http://ossipedia.ipa.go.jp/ipafont/index.html>」からダウンロードすることが出来ます。



ページをスクロールして「IPA フォントの詳細とダウンロード」をクリックします。



- ② 「IPA フォントライセンス」を確認し、ページをスクロールして「IPA サイトからダウンロード」の TTF ファイルの 4 書体パック「IPAFont00303.zip(19.1 MB)」をクリックしてダウンロードを開始します。
 ここでの保存先は「qt-work」とします。



- ③ フォントを展開します。

```

省略 $ cd ~
省略 $ unzip qt-work/IPAFont00303.zip
Archive: /home/guest/qt-work/IPAFont00303.zip
  inflating: IPAFont00303/IPA_Font_License_Agreement_v1.0.txt
  inflating: IPAFont00303/ipag.ttf
  inflating: IPAFont00303/ipagp.ttf
  inflating: IPAFont00303/ipam.ttf
  inflating: IPAFont00303/ipamp.ttf
  inflating: IPAFont00303/Readme_IPAFont00303.txt
    
```

- ④ スケルトンファイルシステムにフォント用ディレクトリを作成します。

```

省略 $ cd ~/xg3358-lk/buildroot-2013.11-xg3358-X.X/system/skeleton-xg3358
    
```

上記は、LK-3358-A01の時のディレクトリ指定となります。移動先ディレクトリは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのディレクトリ名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

機種（開発キット）	ディレクトリ
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X/system/skeleton-xg3358
XG-BBEXT	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X/system/skeleton-xg-bbext

- ⑤ Qt用フォントディレクトリを作成します。

```
省略 $ mkdir -p usr/lib/fonts
```

- ⑥ 作成したディレクトリにフォントファイルをコピーします。

```
省略 $ cd usr/lib/fonts
省略 $ cp ~/IPAFont00303/*.ttf .
省略 $ mkdir IPAFont00303
省略 $ cp ~/IPAFont00303/*.txt IPAFont00303/
省略 $ ls IPA* ipa*
ipag.ttf ipagp.ttf ipam.ttf ipamp.ttf

IPAFont00303:
IPA_Font_License_Agreement_v1.0.txt  Readme_IPAFont00303.txt
```



IPAのFAQに液晶つきの装置の利用について以下の解説があります。

『IPAフォントの再配布においては、契約書の添付が必要ですので、製品のHDやROM等の中にも契約書を入れて頂くこととなります。それを読む手段が提供されていないと意味がありませんので、複雑すぎない常識的な範囲の操作によって、利用者に契約書を表示できるようにしてください。また、製品の説明書等にIPAフォントが入っている旨と契約書または契約書のURL (http://ipafont.ipa.go.jp/ipa_font_license_v1.html)を掲載してください。派生プログラムの場合は、「IPAフォントライセンスv1.0」第3条(制限事項)の条件に従ってご利用ください。』



Qt以外のアプリケーションで日本語フォントを必要とする場合、usr/share/fonts/truetypeディレクトリ配下にフォントファイルが必要となります。その場合は「ttf-ipa-font」ディレクトリを作成し、その中にフォントファイルをコピーするか、シンボリックリンクを張って下さい。

3.3 設定ファイルの準備

Qt アプリケーションを稼働させるための環境設定用のファイルの準備をします。必要に応じてデバッグ用にネットワークの設定も準備します。また、Qt アプリケーションの実行に必要な、いくつかの環境変数の設定ファイルも準備します。

以下の手順で起動時のスクリプト、および関連ファイルを作成します。

- ① スケルトンファイルシステムのディレクトリに移動します。

```
省略 $ cd ~/xg3358-lk/buildroot-2013.11-xg3358-X.X/system/skeleton-xg3358
```

上記は、LK-3358-A01 の時のディレクトリ指定となります。移動先ディレクトリは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのディレクトリ名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

機種（開発キット）	ディレクトリ
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X/system/skeleton-xg3358
XG-BBEXT	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X/system/skeleton-xg-bbext

- ② 必要に応じてネットワーク設定します。

```
省略 $ vi etc/network/interfaces
```

- ③ etc/profile へ環境変数を追加します。（XG-BBEXT の場合は設定済みです）

Qt アプリケーションで必要となる環境変数は下図のとおりです。バックエンドによって必要な環境は異なります。

環境変数名	設定内容(XG-3358)	設定内容 (XG-BBEXT)
LANG	'ja_JP.UTF-8'	'ja_JP.UTF-8'
TZ	JST-9	JST-9
TSLIB_CONSOLEDEVICE	none	none
TSLIB_TSDEVICE	/dev/input/event0	/dev/input/event1
QWS_MOUSE_PROTO	'tslib:/dev/input/event0'	'tslib:/dev/input/event1'

```
省略 $ vi etc/profile
```

以下のテキストを「etc/profile」の環境変数定義の最後に追加します。

[XG-3358]

```
export LANG=' ja_JP.UTF-8'
export TZ=JST-9
export TSLIB_CONSOLEDEVICE=none
export TSLIB_TSDEVICE=/dev/input/event0
export QWS_MOUSE_PROTO=' tslib:/dev/input/event0'
```

[XG-BBEXT]

```
export LANG='ja_JP.UTF-8'  
export TZ=JST-9  
export TSLIB_CONSOLEDEVICE=none  
export TSLIB_TSDEVICE=/dev/input/event1  
export QWS_MOUSE_PROTO='tslib:/dev/input/event1'
```

3.4 ルートファイルシステムのカスタマイズ

Qt パッケージの必要な機能を選択し、ルートファイルシステムを再構築します。

- ① ビルドルートシステムディレクトリに移動します。

```
省略 $ cd ~/xg3358-lk/buildroot-2013.11-xg3358-X.X
```

上記は、LK-3358-A01 の時のディレクトリ指定となります。移動先ディレクトリは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのディレクトリ名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

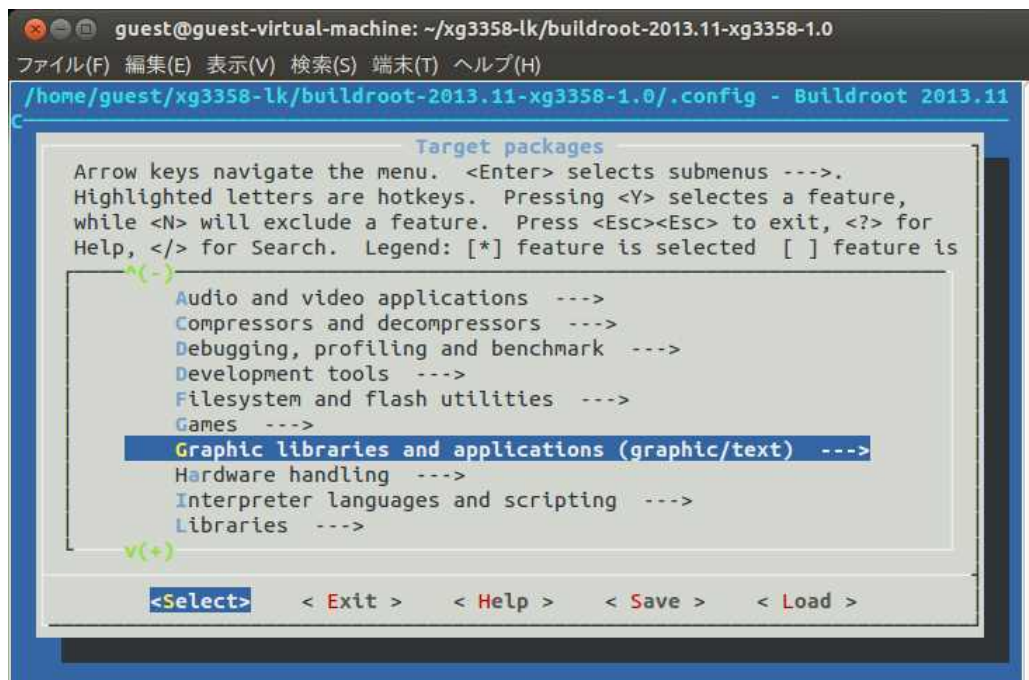
機種（開発キット）	ディレクトリ
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X
XG-BBEXT	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X

- ② ルートシステムのカスタマイズのためのメニューを起動します。

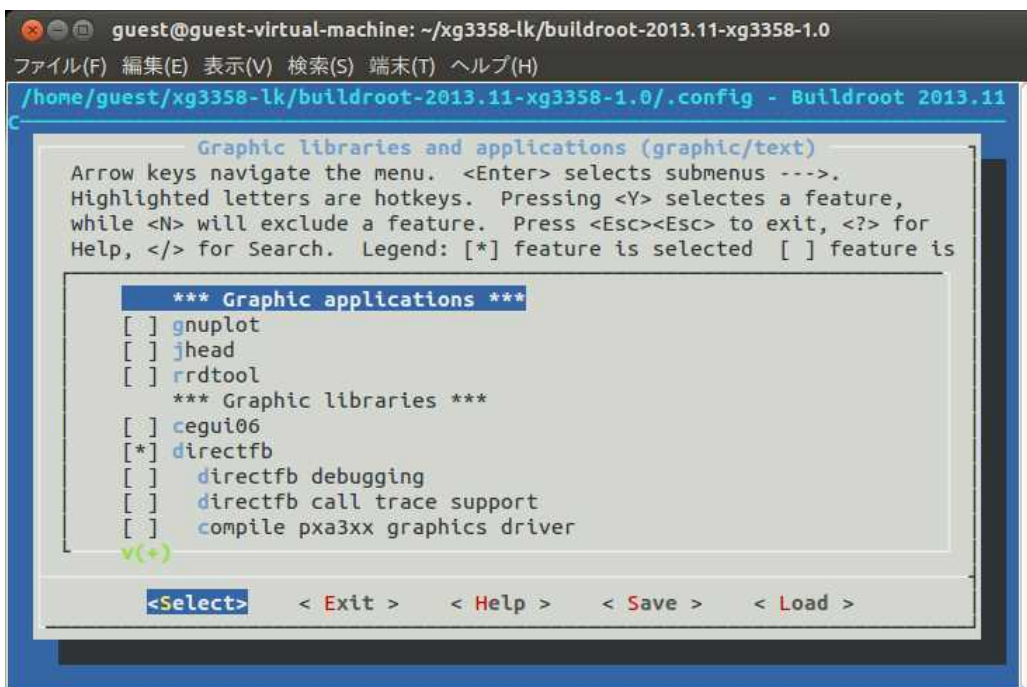
```
省略 $ make menuconfig
```

- ③ Qt に関する設定は「Target Packages」の中にありますので、それを選択し Enter キーを押します。

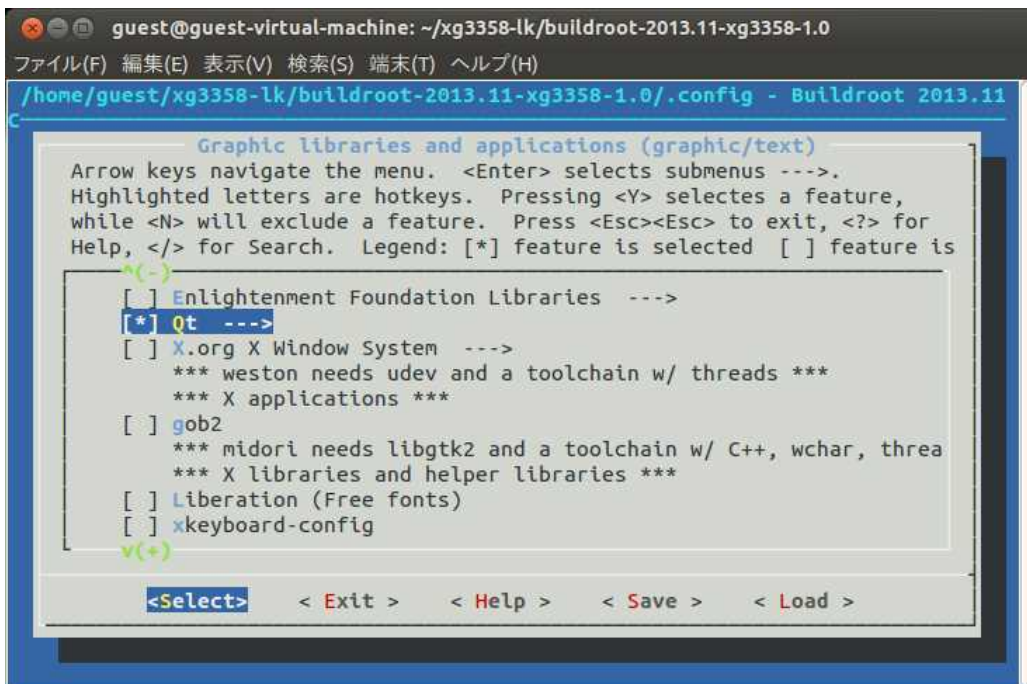
- ④ 「Graphic Libraries and applications (graphic/text)」に移動し、スペースキーあるいは Enter キーで選択します。



- ⑤ 「↓」カーソルキーで移動し、Qt の項目まで移動します。

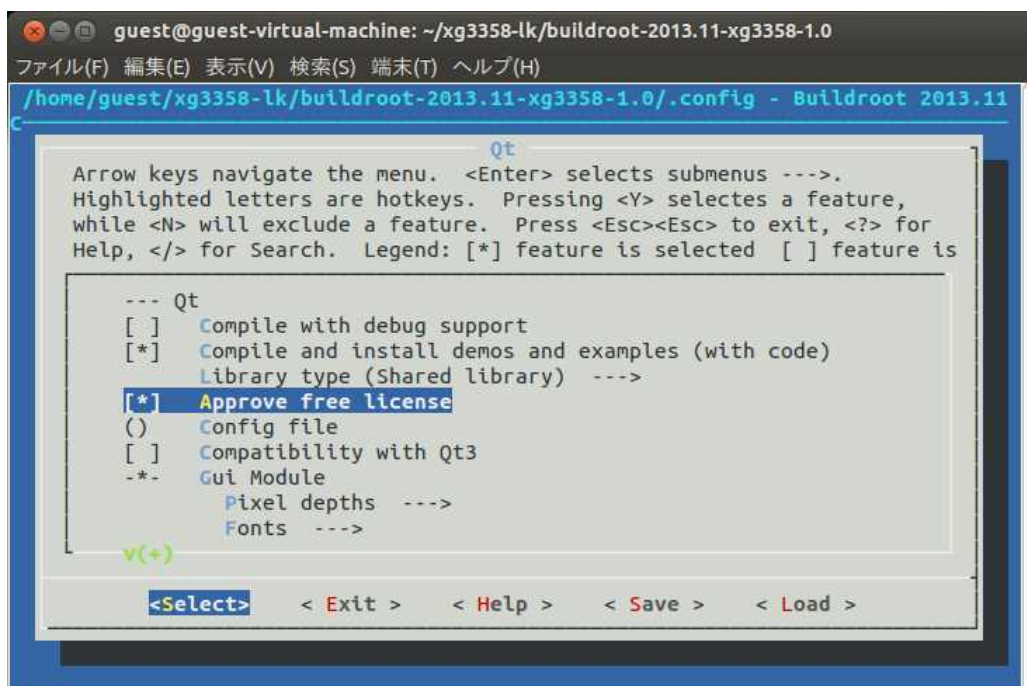


- ⑥ スペースキーを押して Qt 項目にチェックをいれます (* マークがつきます)。Qt の詳細設定をするために、Enter キーを押します。

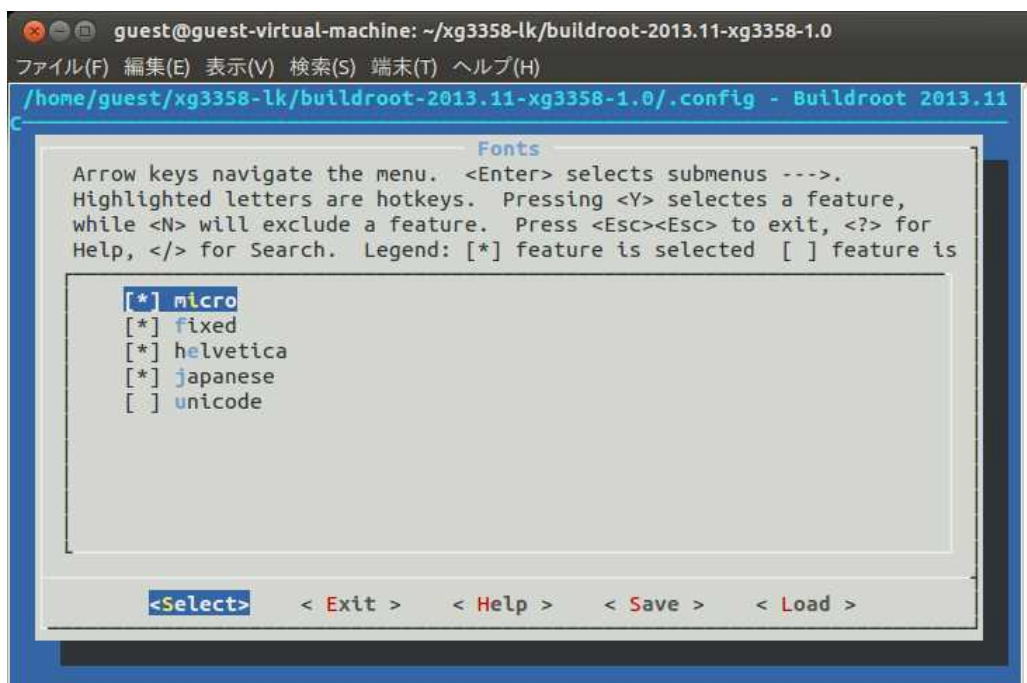


- ⑦ Qt のデモと例で動作テストをしたい場合には、コンパイル&インストールするように「Compile and install demos and examples (with code)」にチェックを入れます。

「Approve free license」にチェックを入れます。チェックが入っていないときは、make 中にライセンスの確認のための質問があります。

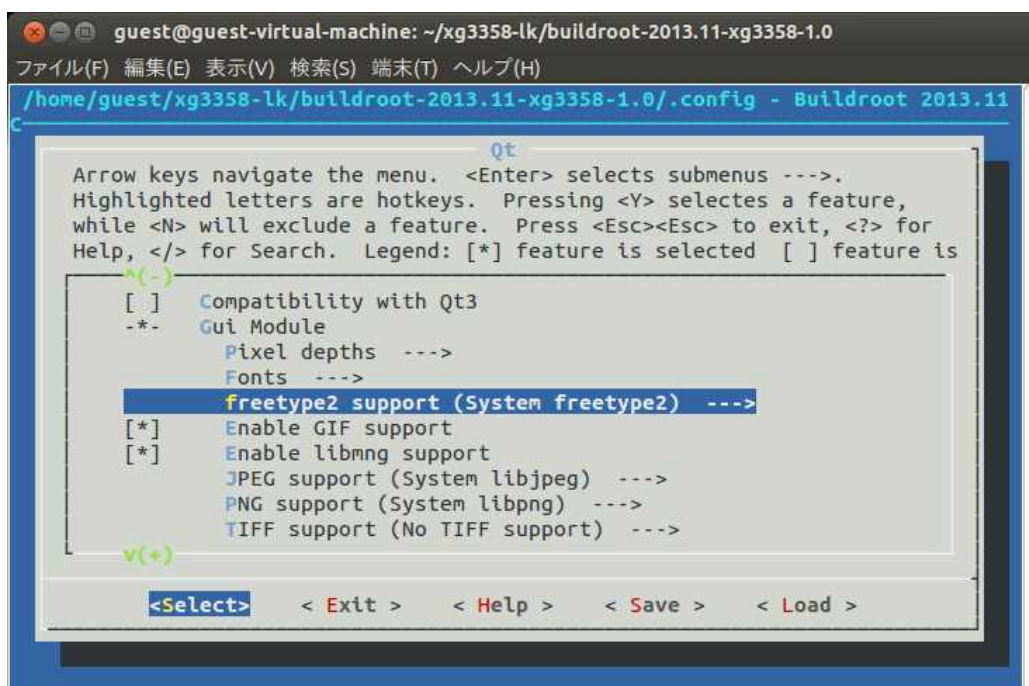


- ⑧ 「Gui Module」がチェックされていることを確認し、その中の Fonts を選択し Enter キーを押して、Fonts メニューにて japanese を追加選択します。

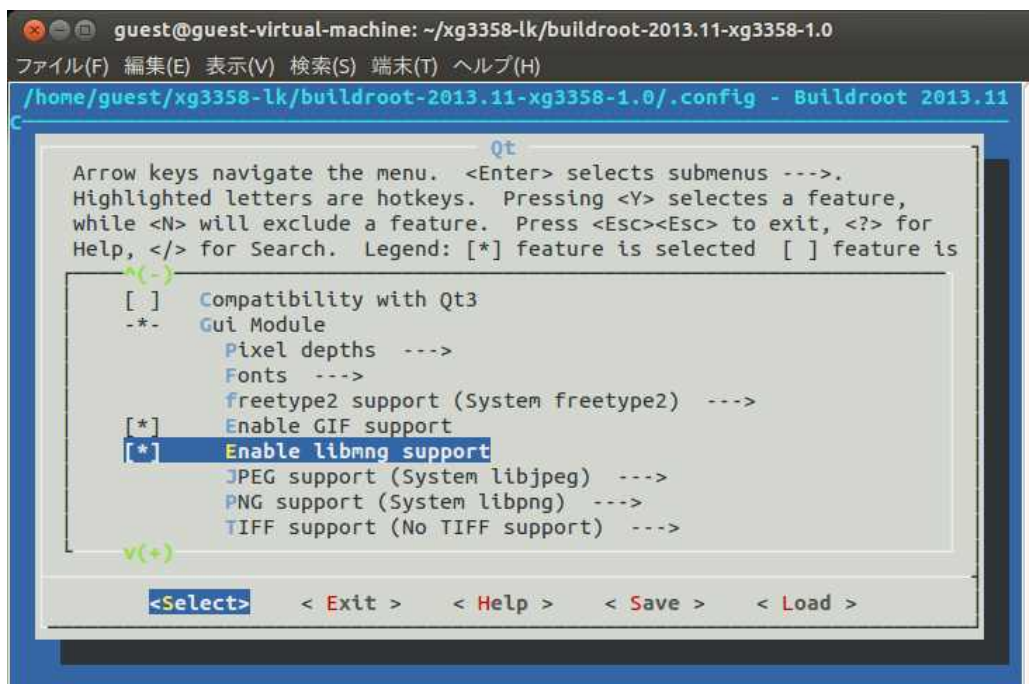


ESC-ESC で一段上のメニューに戻ります。

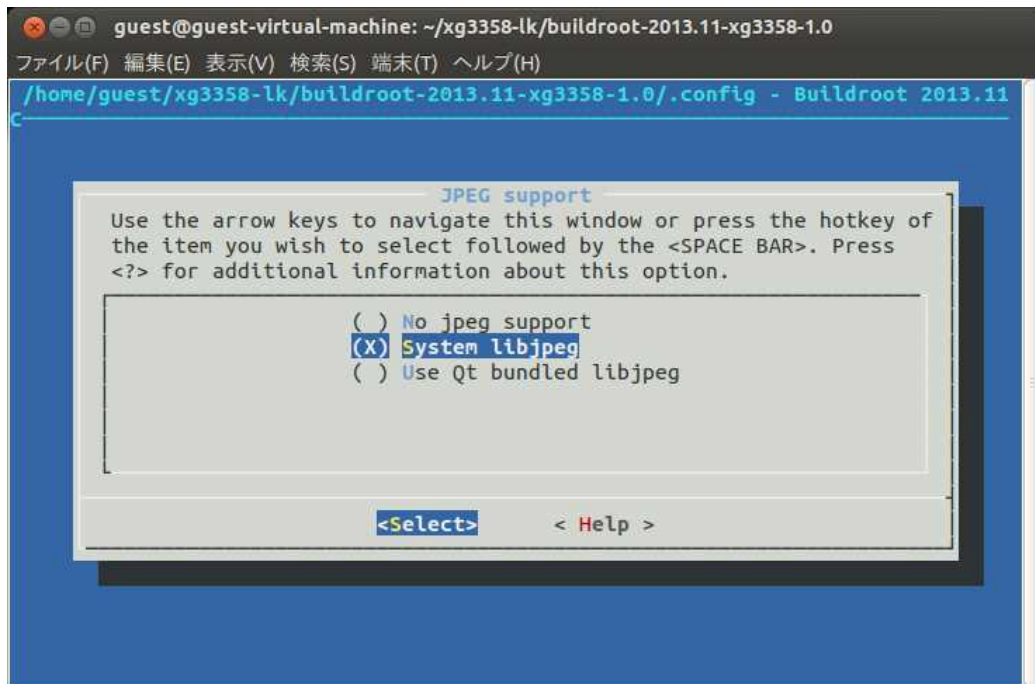
- ⑨ 「freetype2 support」を「system freetype2」に設定します。



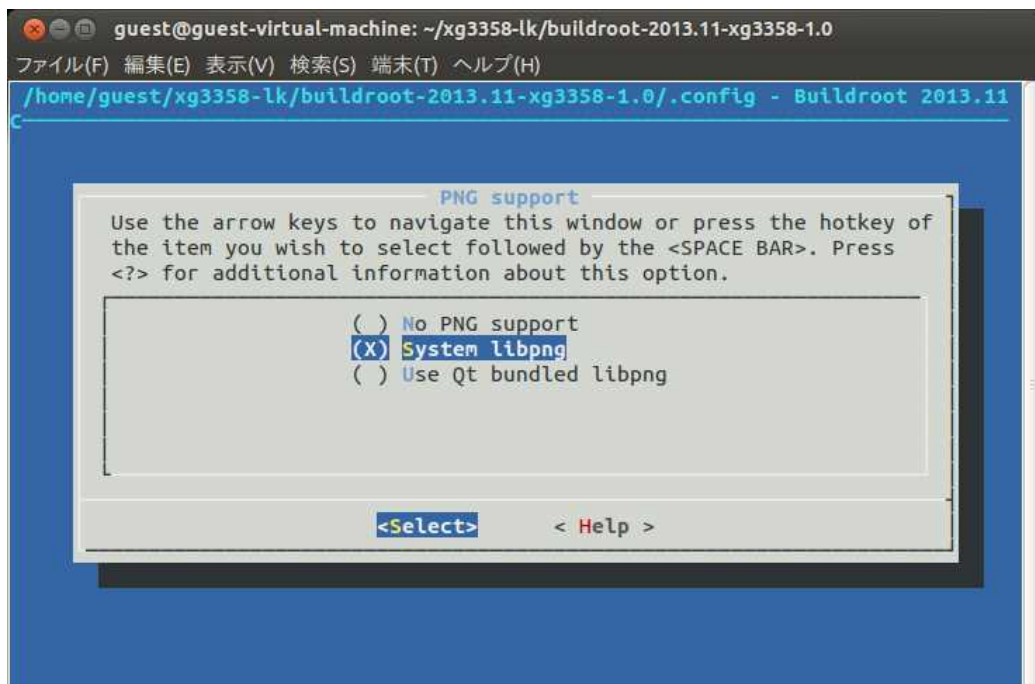
- ⑩ Enable GIF support を選択しスペースを押してチェックを入れます。
png 画像のアニメーションをサポートするときは「Enable libmng support」にもチェックを入れます。



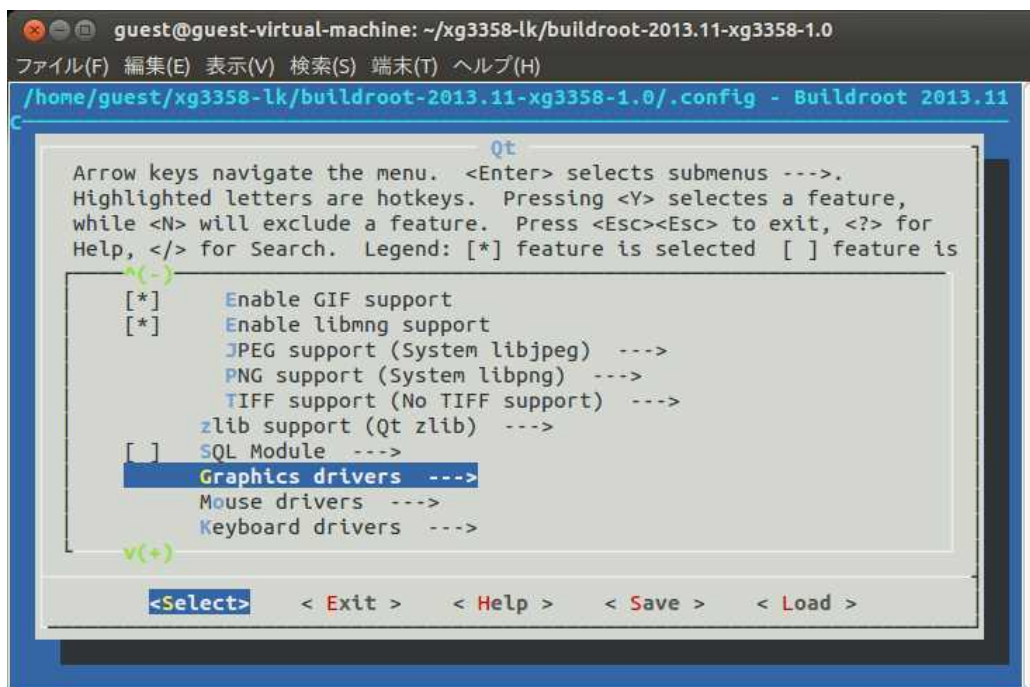
- ⑪ JPEG support を選択し、JPEG support メニューにて libjpeg を選択します。



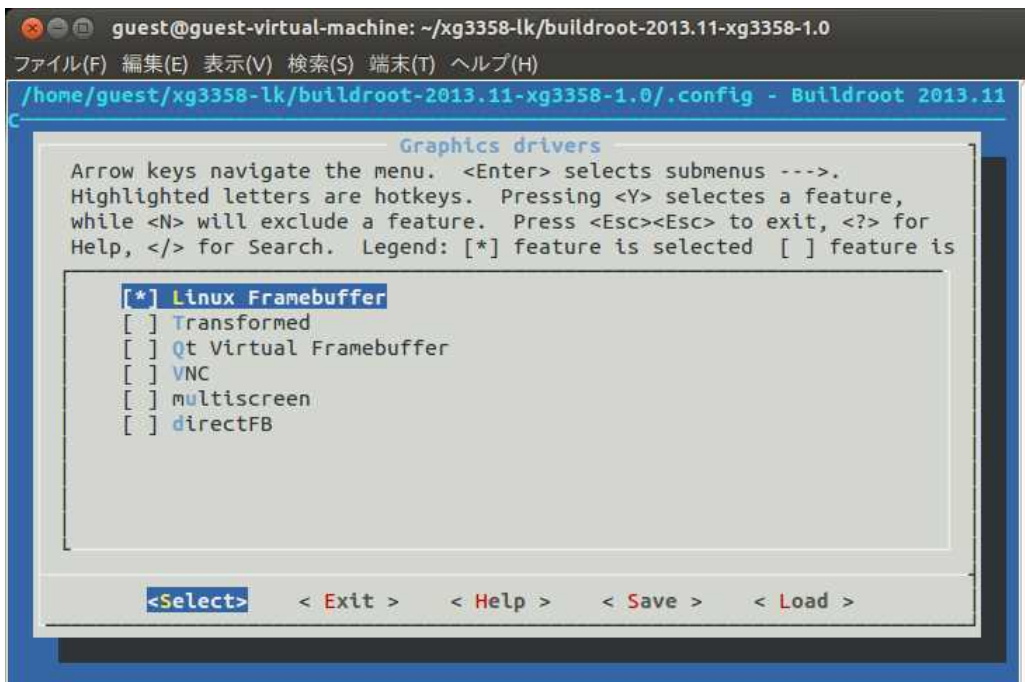
- ⑫ PNG support を選択し、PNG support メニューにて System libpng を選択します。



- ⑬ 「Graphics drivers」を選択し、Enter キーを押します。

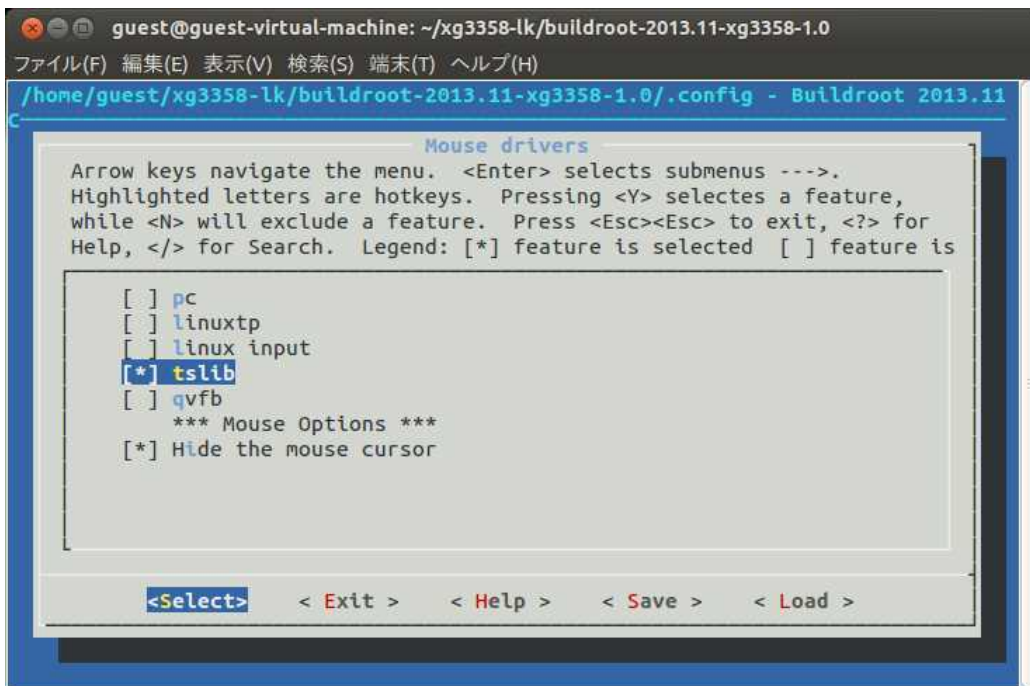


- ⑭ 「Linux Framebuffer」を選択します。



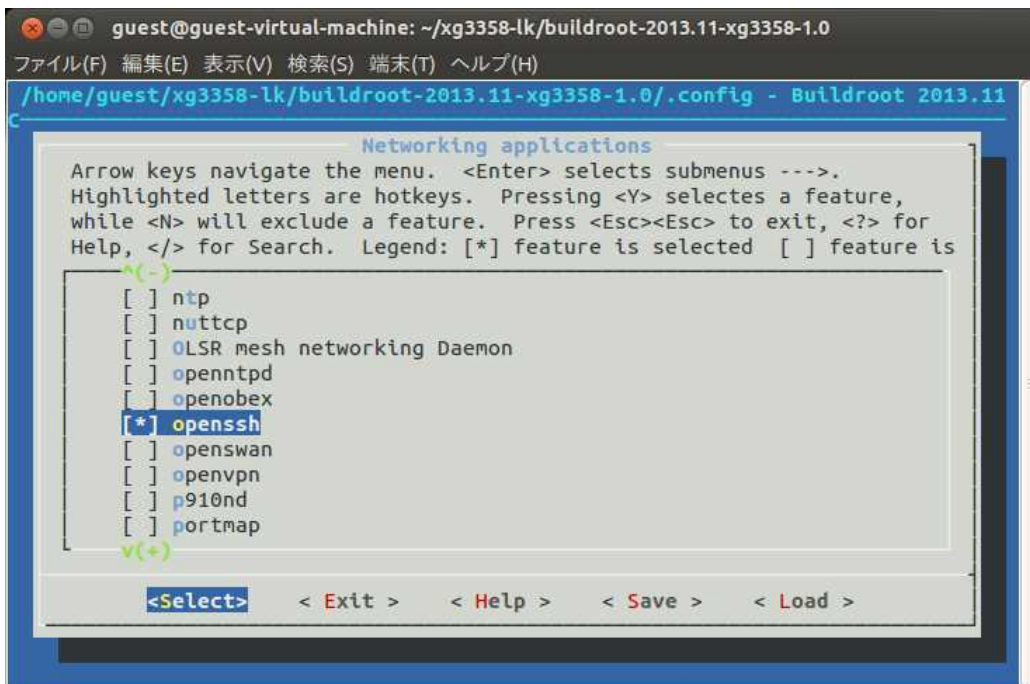
ESC-ESC で Qt のメニューに戻ります。

- ⑮ Mouse drivers を選択し Enter キーを押し Mouse drivers メニューにて、tslib を選択します。マウスカーソルを表示したくない場合は「Hide the mouse cursor」を選択します。



ESC-ESC、ESC-ESC で Target packages のメニューに戻ります。

- ⑯ Openssh を有効にします。「Target packages」 - 「Newtworking applications」メニューを開いて「openssh」を選択します。



- ⑰ 一通りの設定が終了したら ESC-ESC を複数回押し保存問い合わせが出るまで戻り、Yes を選択して設定内容を保存、menuconfig を終了します。

3.5 WebKit に対応するためのカスタマイズ

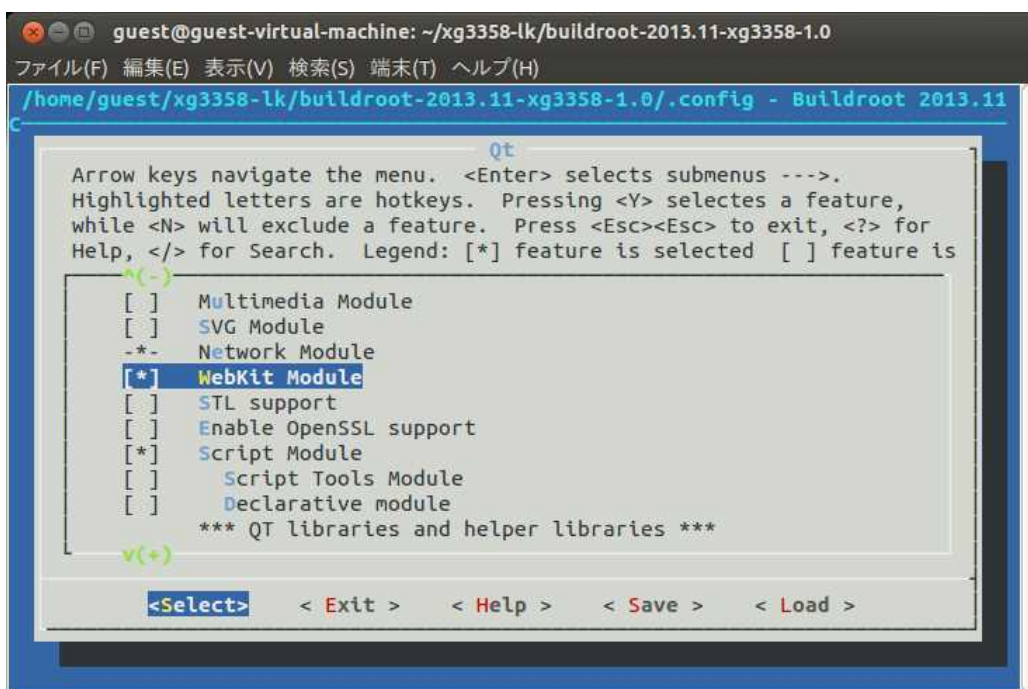
WebKit は Web コンテンツのレンダリングと編集を行うエンジンで、Qt で Web 対応のアプリケーションを作成することができます。Qt にはアプリケーション例として fancybrowser という WebKit を使用したブラウザがあります。

WebKit を使用することによって HTML5 や jQuery を使った Web アプリも可能となります。

WebKit を導入するに当たっては Web アクセス関連の機能だけでなくマルチメディアの機能や HTML, XML, XML2 などの機能も必要となり多くのパッケージの導入が必要です。WebKit を選択しただけでは関連パッケージが全てビルドされるわけではありませんので一部は追加指定をしなければなりません。

ここではよく使われる機能についてビルドするための手順を説明します。

- ① 「WebKit Module」を選択します。

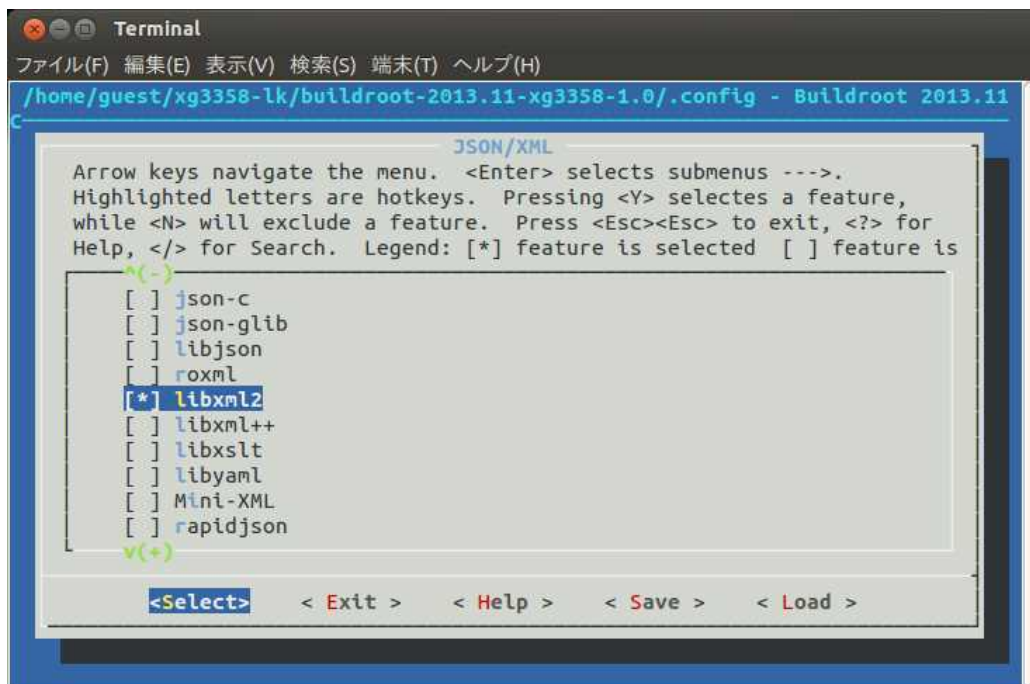


この他にも Qt で使用する機能について必要なものを選択します。以下のものを除く Qt 関連のほとんどの機能を選択してください。

Qtの設定において、選択しない項目一覧	
Compatibility with Qt3	通常Qt3との互換性は必要ありません。
grantlee	Djangoテンプレートフレームワーク
qwt	Qt GUIアプリケーションフレームワーク用のグラフィック拡張です。

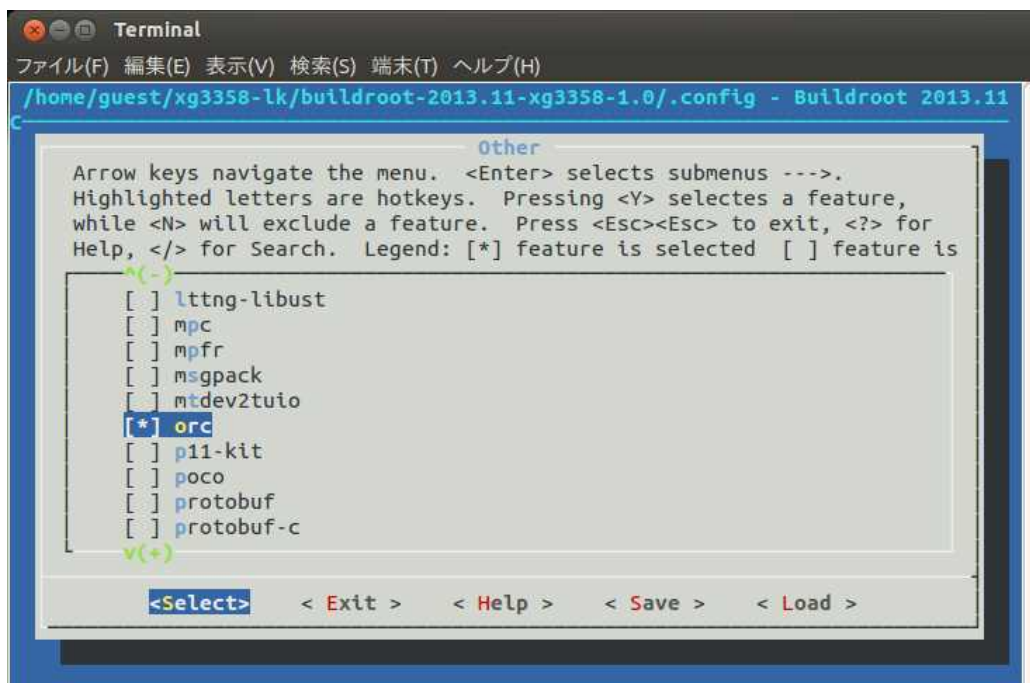
Qt の設定が終了したら ESC-ESC、ESC-ESC を押して「Target Packages」メニューに戻ります。

- ② webkit を使う場合は libxml2 も選択します。
 「Target packages」 - 「Libraries」 - 「JSON/XML」メニューを開き「libglib2」を選択します。



設定が終了したら ESC-ESC、ESC-ESC を押して「Target Packages」メニューに戻ります。

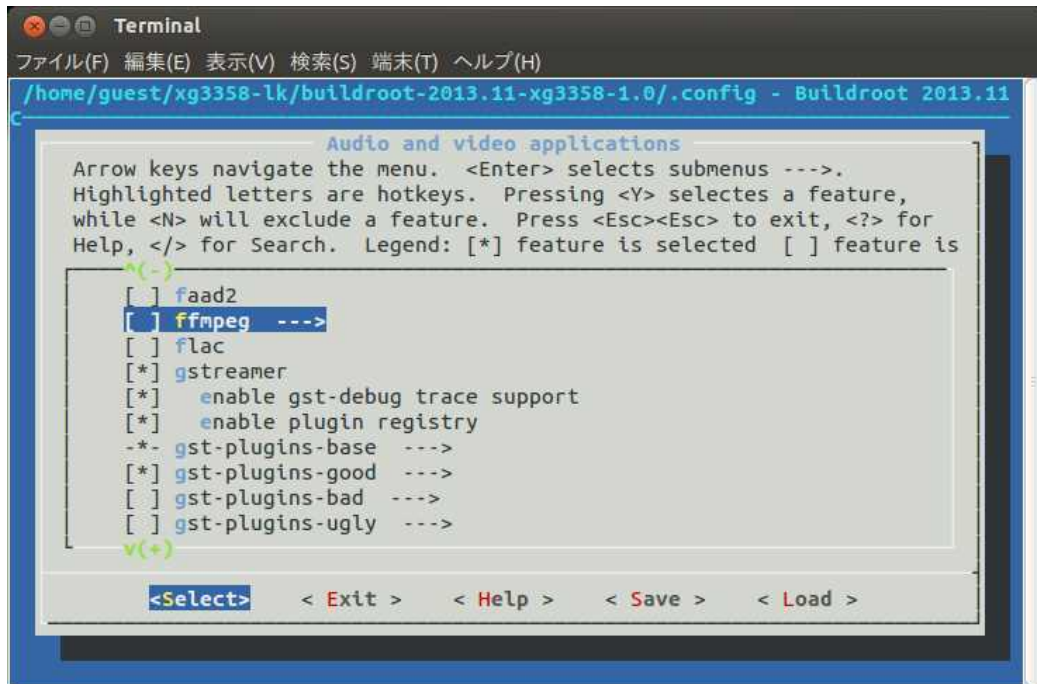
- ③ orc を選択します。
 「Target packages」 - 「Libraries」 - 「Other」メニューを開き「orc」を選択します。



- ④ gstreamer を選択します。

Webkit などマルチメディアを扱うパッケージの場合下位層で gstreamer が使われていますので、それを指定します。

「Target packages」 - 「Audio and video applications」にて「gstreamer」関連項目を指定します。gstreamer に沢山のプラグインがあります。各プラグイン (base,good,bad,ugly)を選択し必要とする項目を選択します。



3.6 ルートファイルシステムのビルド

- ① menuconfig が終了したら、以下のコマンドでルートファイルシステムのビルドを行います。
Qt の選択した項目によってはビルドに数時間程度かかることもあります。

```
省略 $ make ←
```

- ② ルートファイルシステムの make が正常に終了していると、『./output/images』ディレクトリに『rootfs.tar.gz』ファイルが作成されています。

```
省略 $ ls output/images/rootfs.tar.gz ←
output/images/rootfs.tar.gz
```



config の内容によっては、必要なパッケージのソフトをダウンロードすることになりますので、ネットワークに接続した環境にて make を行います。
make の途中でエラーが発生することがあります。エラーメッセージを参考にエラー原因を取り除いて再度 make します。よくある原因としては、以下の内容があります。

- ・開発 PC に必要なパッケージ・ライブラリがインストールされていない。
- ・開発 PC の Linux、バージョンが適応しない。(Ubuntu14.04 など)
- ・Toolchain のバグ (最適化レベルを変えてみたりして回避します)
- ・ダウンロードサイトがメンテナンス中あるいは移動した。
- ・menuconfig で選択したパッケージが足りない。



Buildroot のあるバージョンのリリース後にダウンロード先の URL が変更になりダウンロードが失敗しビルドエラーになることがあります。
この場合の対処方法は、

- 1) 他のマシンにダウンロードしたファイルがある場合はそれを dl ディレクトリにコピーする。
- 2) 手動でダウンロードし、dl ディレクトリにコピーする。
- 3) package/パッケージ名/パッケージ名.mk を編集し、ダウンロードサイトを修正する。

手動でダウンロードする場合のダウンロードファイル名を知るにはビルド時のログメッセージを調べるか、「package/パッケージ名/パッケージ名.mk」から該当ファイル名を読み解きます。
セキュリティアップデートする場合は、バグ回避のため別バージョンにアップデートまたはダウンロードする場合もこのファイルを修正します。しかしながら、他のパッケージとの関係でなんらかの影響が出ることがよくありますので十分配慮する必要があります。

4. SD ルートファイルシステム

4.1 XG-3358 における SD ルートファイルシステムの作成

microSD カードの構成はパーティションが 1 つ存在し、その箇所に SD ルートファイルシステムを作成する手順で説明します。

- ① microSD カードをホスト PC の SD カードスロットに挿入して、Ubuntu 上で操作できるようにします。



Ubuntu で microSD カードを認識した場合、自動でマウントされる場合があります。その場合には、すべてアンマウントしてから行うようにしてください。また、microSD カードのデバイス名がわからない場合には、『`sudo fdisk -l`』等を使用して事前に確認してください。

- ② microSD カードの第 1 パーティションを EXT3 でフォーマットします。
(以下のコマンドでは、microSD カードが『`/dev/sdb1`』として認識している場合です。)

```

省略 $ sudo mke2fs -j /dev/sdb1
[sudo] password for guest:
mke2fs 1.41.11 (14-Mar-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)

:
途中省略
:

This filesystem will be automatically checked every 21 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

- ③ フォーマットした領域をマウントします。

```

省略 $ sudo mount /dev/sdb1 /mnt
[sudo] password for guest:
```

- ④ sd ルートファイルシステムを展開します

```

省略 $ sudo zcat output/images/rootfs.tar.gz | sudo tar -xf - -C /mnt
[sudo] password for guest:
```

- ⑤ アンマウントします。

```

省略 $ sudo umount /mnt
[sudo] password for guest:
```

4.2 XG-3358 における SD-Linux システムの起動

U-Boot を使用し、ソフトウェアマニュアル『タッチパネル LCD キットの使用』で作成した Linux カーネル『**uImage-xg3358-lcdkit**』をネットワーク経由(TFTP)でダウンロードし、SD ルートファイルシステムを作成した microSD カードを使用して SD-Linux システムを起動する方法を示します。

カーネルのロードアドレスは各機種（開発キット）によって異なります。下記の手順は LK-3358-A01 の操作例です。その他の機種（開発キット）についてはアドレス、ファイル名を以下の表から読み替えて行ってください。

機種（開発キット）	ロードアドレス	カーネルイメージファイル名
XG-3358(LK-3358-A01)	0x83000000	uImage-xg3358-lcdkit

① Linux カーネルイメージ『**uImage-xg3358-lcdkit**』を RAM 上にダウンロードします。

```
=> tftp 83000000 uImage-xg3358-lcdkit
link up on port 0, speed 100, full duplex
Using cpsw device
TFTP from server 192.168.128.210; our IP address is 192.168.128.200
Filename 'uImage-xg3358-lcdkit'.
Load address: 0x83000000
Loading: #####
#####
#####
#####
#####
#####
#####
#####
#####
###
237.3 KiB/s
done
Bytes transferred = 2675864 (28d498 hex)
```

② 環境変数『bootargs』に『root=/dev/mmcb1k0p1 rootwait』のパラメータを追加します。

```
=> setenv bootargs $bootargs root=/dev/mmcb1k0p1 rootwait
```

③ ダウンロードしたカーネルと microSD カードで起動します。

```
=> bootm 83000000
## Booting kernel from Legacy Image at 83000000 ...
Image Name: Linux-3.2.0-xg3358-1.0
Image Type: ARM Linux Kernel Image (uncompressed)
:
途中省略
:
Welcome to Buildroot
xg-3358 login:
```

SD-Linux システムの場合には、ボードの電源を切る前に終了処理を行う必要があります。
以下のコマンド『halt』を実行し、『System halted.』を確認してから電源を切ってください。

```
# halt
The system is going down NOW!
Sent SIGTERM to all processes
Sent SIGKILL tomusb-hdrc musb-hdrc.1: remove, state 1
usb usb1: USB disconnect, device number 1
musb-hdrc musb-hdrc.1: USB bus 1 deregistered
System halted.
```



SD-Linux システムで自動的に Linux を起動するには次のように U-Boot の環境変数を設定し保存します。

1. U-Boot、Kernel を NOR フラッシュから U-Boot、Kernel を起動する場合
ソフトウェアマニュアル「11.ボードの初期化」を参考にタッチパネルに対応したカーネル
uImage-xg3358-lcdkit を書込んでおきます。

```
=> setenv bootargs "console=${console} root=/dev/mmcblk0p1 rootwait"
=> setenv bootcmd "bootm ${kaddr}"
=> saveenv
```

2. U-Boot、Kernel を SD カードから起動する場合 (MMC ブート)
MMC ブートにて起動された U-Boot では saveenv コマンドが使えないためデフォルトで SD
カードから kernel を起動するように U-Boot のソースの環境変数を変更し u-boot.img をビルド
します。

ソフトウェアマニュアル「10.MMC ブート」を参考にして第一パーティションを FAT32 で
フォーマットし MLO,u-boot.img,uImage-xg3358-lcdkit を格納します。

Linux ルートファイルシステムはソフトウェアマニュアル「5.3 ルートファイルシステムの作成
- sd ルートファイルシステムの作成」を参考にして第二パーティションを ext3 でフォーマット
しルートファイルシステムを格納します。

4.3 XG-BBEXT におけるルートファイルシステムの作成

XG-BBEXT では BeagleBone Black 本体の eMMC あるいは micoroSD にルートファイルシステムを作成します。
手順については「XG-BBEXT Linux Software manual」を参考にしてください。

5. 動作確認

5.1 環境変数の確認

ビルドしたルートファイルシステムが Qt に必要な環境で設定されているかどうか確認します。root でログインし、`printenv` で環境変数がセットされているかどうか確認します。

[XG-3358 の例]

```
xg-3358 login: root ←入力
# printenv ←入力
HISTFILESIZE=1000
INPUTRC=/etc/inputrc
TSLIB_TSDEVICE=/dev/input/event0   タッチパネルデバイスを指定
USER=root
HOSTNAME=xg-3358
HOME=/root
PAGER=/bin/more
PS1=#
TSLIB_CONSOLEDEVICE=none
LOGNAME=root
TERM=vt100
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/bin/X11:/usr/local/bin
LANG=ja_JP.UTF-8   言語は日本語、コードは UTF-8
DMALLOC_OPTIONS=debug=0x34f47d83, inter=100, log=logfile
HISTSIZ=1000
SHELL=/bin/sh
PWD=/root
TZ=JST-9   タイムゾーンは日本時間に
QWS_MOUSE_PROTO=tslib:/dev/input/event0
EDITOR=/bin/vi
#
```

[XG-BBEXT の例]

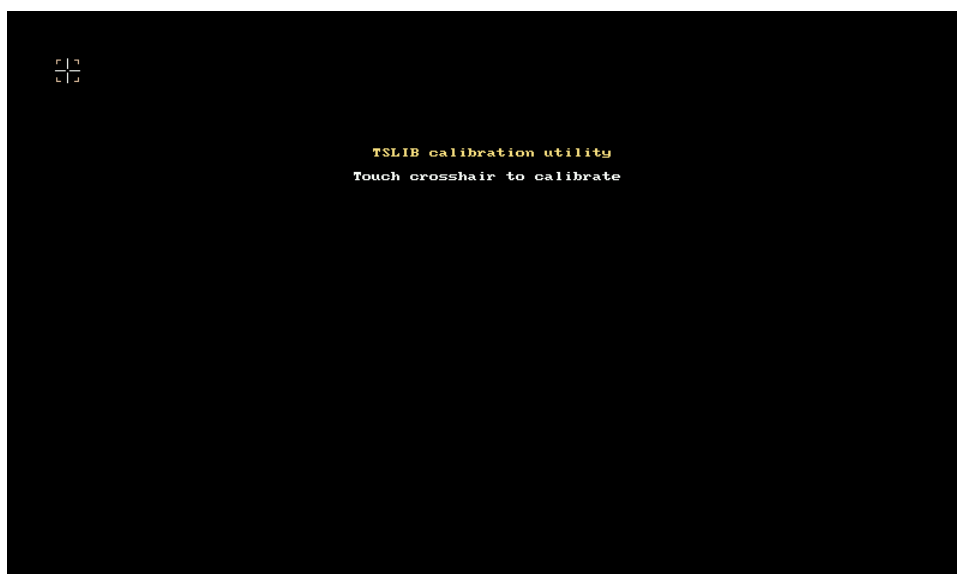
```
xg-bbext login: root ←入力
# printenv ←入力
HISTFILESIZE=1000
INPUTRC=/etc/inputrc
TSLIB_TSDEVICE=/dev/input/event1   タッチパネルデバイスを指定
USER=root
HOSTNAME=xg-bbext
HOME=/root
PAGER=/bin/more
PS1=#
TSLIB_CONSOLEDEVICE=none
LOGNAME=root
TERM=vt100
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/bin/X11:/usr/local/bin
LANG=ja_JP.UTF-8   言語は日本語、コードは UTF-8
DMALLOC_OPTIONS=debug=0x34f47d83, inter=100, log=logfile
HISTSIZE=1000
SHELL=/bin/sh
PWD=/root
TZ=JST-9   タイムゾーンは日本時間に
QWS_MOUSE_PROTO=tslib:/dev/input/event1
EDITOR=/bin/vi
```

5.2 タッチパネルキャリブレーション

次にタッチパネルのキャリブレーションを行います。

```
# ts_calibrate ←
```

「+」が表示されますので、順にその位置をタッチしてください。タッチパネルのキャリブレーション情報は「/etc/pointercal」に格納されます。このファイルが無いと Qt アプリケーションを動かすことができませんので、必ずキャリブレーションをしてください。



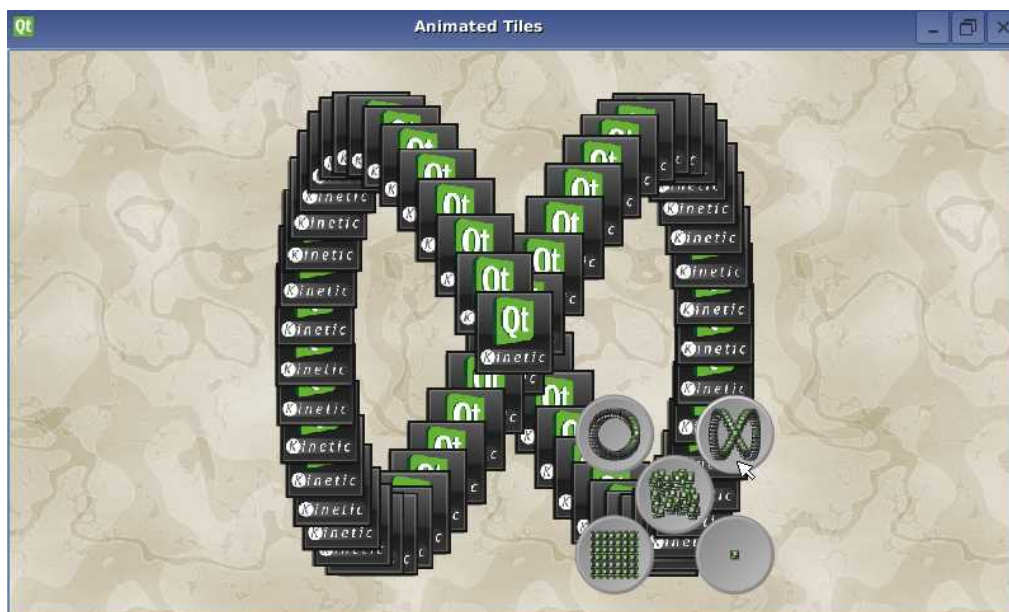
5.3 Qt の動作確認

ルートファイルシステムの menuconfig にて、Qt デモおよび例をインストールした場合は Qt の動作確認が出来ます。これらデモおよび例は、/usr/share/qt に格納されています。ただし、組み込み用の Qt embedded 用として特別に用意されているデモおよび例ではありませんので、組み込みシステムでは利用できないものもたくさん含まれています。

- ① 画面表示とタッチパネルの動作確認をします。

ここではタイルのアニメーション例で動作確認します。

```
# cd /usr/share/qt/examples/animation/animatedtiles
# ./animatedtiles -qws
```



画面上の 5 つのボタンを押すと、Qt タイルが移動しボタンのイメージの形に整列します。

デモ、例はソースコードも一緒にインストールされていますので Qt プログラミングの参考にしてください。



Qt アプリケーションを動かす場合、Qt Embedded Linux Server が必要となります。オプション「-qws」と付けることによって、プログラムをサーバとして起動することが出来ます。

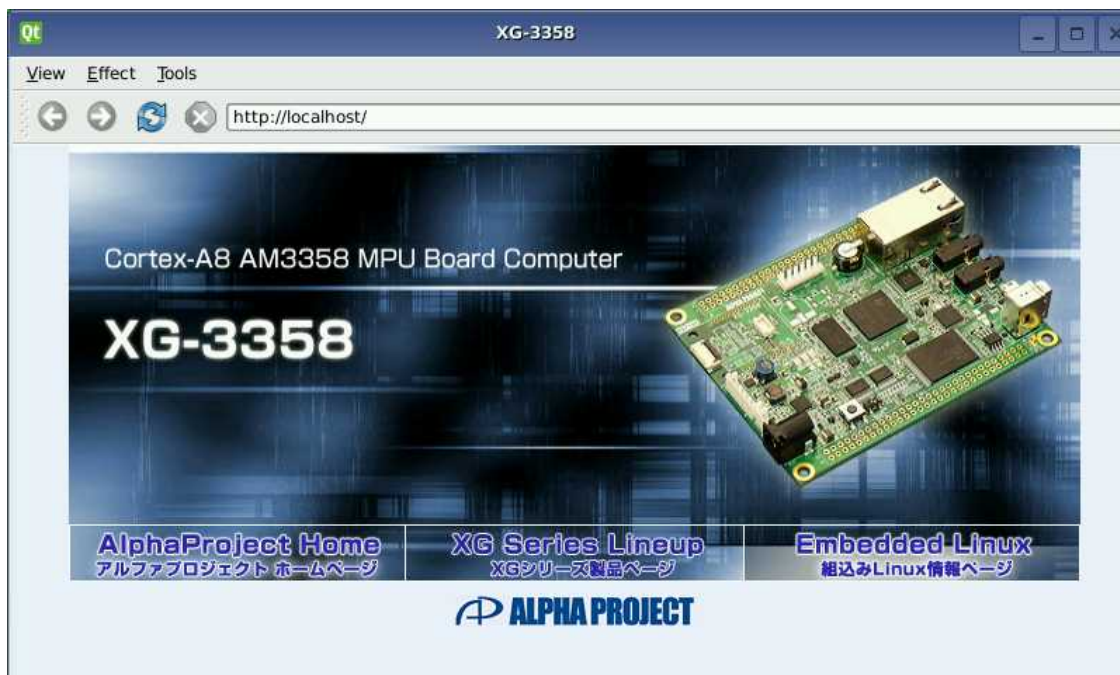
サーバとして起動している Qt アプリケーションがすでに存在している場合、別の Qt アプリケーションを起動するときには「-qws」オプションは必要ありません。サーバとして起動している Qt アプリケーションを先に閉じると、そのほかの Qt アプリケーションもエラーで終了します。

後で起動した Qt アプリケーションに「-qws」オプションが付加されていると GUI イベントを取り合う形になり、操作が正常に出来なくなります。

- ② WebKit 対応の設定でビルドした場合は WebKit の動作確認をします。
ここでは簡単なブラウザサンプルを起動し XG-3358 または XG-BBEX のデフォルトで起動している Web サーバに接続します。

```
# cd /usr/share/qt/examples/webkit/fancybrowser ←カ  
# ./fancybrowser -qws -geometry 800x480+0+0 http://localhost ←カ
```

ウィンドウの表示サイズを 800×480 ドットのフルスクリーンで表示しています。



5.4 ssh の動作確認

リモート PC から XG-3358、XG-BBEXT を操作するには、シリアルあるいはネットワーク接続が必要になります。統合開発環境下でデバッグする場合通常ターゲットボードとの接続はセキュリティの関係上 ssh が用いられます。「3.4 ルートファイルシステムのカスタマイズ」にて OpenSSH を有効にしてルートファイルシステムをビルドすれば ssh をはじめとするコマンドが利用できるようになります。ssh 関連のコマンドではターゲットボードとの通信は暗号化されます。Linux の最初の起動時に RSA の鍵の生成が行われます。

[RSA 鍵が生成されたときのログ]

```

Generating RSA Key...
/usr/bin/ssh-keygen: /usr/lib/libcrypto.so.1.0.0: no version information
available (required by /usr/bin/ssh-keygen)
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh_host_key.
Your public key has been saved in /etc/ssh_host_key.pub.
The key fingerprint is:
af:36:e3:f4:8e:c7:f2:82:1d:23:fb:9f:25:93:97:8d
The key's randomart image is:
+--[RSA1 2048]-----+
|
|          S
|       . o . . +
|      = . += E .
|     o . B+o*
|     +=X*
|-----+-----+
Generating RSA Key...
/usr/bin/ssh-keygen: /usr/lib/libcrypto.so.1.0.0: no version information
available (required by /usr/bin/ssh-keygen)
Generating public/private rsa key pair.
Your identification has been saved in /etc/ssh_host_rsa_key.
Your public key has been saved in /etc/ssh_host_rsa_key.pub.
The key fingerprint is:
11:28:51:a5:5d:33:74:9f:f2:01:c7:62:e9:17:c4:48

~ 略 ~

|          S .
|-----+-----+
Starting sshd: /usr/sbin/sshd: /usr/lib/libcrypto.so.1.0.0: no version
information available (required by /usr/sbin/sshd)
OK

```

- ① ネットワーク経由でログインできるようにするために、root のパスワードを設定します。
 XG-3358、XG-BBEXT とシリアル接続されたコンソールから「passwd」コマンドを使ってパスワードを設定します。
 ここでは便宜上パスワードは「xg3358&bbext」とします。此処で設定したパスワードは Qt のデバイス構成で使用します。

```
# passwd ←カ
Changing password for root
New password: ←カ
Retype password: ←カ
Password for root changed by root
```

- ② ssh にてログインできることを確認します。
 PC(Ubuntu) から ssh コマンドでログインします。最初のログインでは認証のための問い合わせがありますので「yes」と入力します。接続ができればパスワードを聞いてきますのでパスワードを入力します。

```
省略 $ ssh root@192.168.128.200 ←カ
The authenticity of host '192.168.128.200 (192.168.128.200)' can't be established.
ECDSA key fingerprint is 89:8f:1b:c3:3d:91:e7:59:43:fc:20:00:0a:d0:d4:e5.
Are you sure you want to continue connecting (yes/no)? yes ←カ
Warning: Permanently added '192.168.128.200' (ECDSA) to the list of known hosts.
root@192.168.128.200's password: ←カ
$
```



設定が同じ IP の別のターゲットボードボードに接続しようとした場合、PC(Ubuntu)側から見ると、偽のボードで不正をしようとしているのではと警告が表示され接続は拒否されます。

デバッグなどでは同じ IP で行う場合ことはよくあります。
 この場合は警告の文章の中に対処するためのコマンドが表示されていますのでその部分をコピー & ペーストして実行し「~/ssh/known_host」のホスト情報を削除します。

例 : ssh-keygen -f "/home/guest/.ssh/known_hosts" -R 192.168.128.200

6. アプリケーションの自動実行

アプリケーションを起動時に自動実行するには「/etc/init.d」に起動のためのスクリプトを用意します。

また開発段階および現場サイドでのデバッグにおいては、一つのコマンドでデバッグのための環境設定およびサービスの起動が簡単に出来ると便利です。開発・デバッグの期間だけ起動時に自動設定・起動するようにすればさらに便利です。そのため
のスクリプトも用意します。

① スケルトンファイルシステムのディレクトリに移動します。

```
省略 $ cd ~/xg3358-lk/buildroot-2013.11-xg3358-X.X/system/skeleton-xg3358
```

上記は、LK-3358-A01 の時のディレクトリ指定となります。移動先ディレクトリは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのディレクトリ名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

機種（開発キット）	ディレクトリ
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X/system/skeleton-xg3358
XG-BBEXT	~/xgbbext-lk/buildroot-2013.11-xgbbext-X.X/system/skeleton-xg-bbext

② 自動起動スクリプトを追加します。

シリアル接続、telnet,ssh 接続してログインしたときは/etc/profile で設定した環境変数は有効でありませんが、環境変数も指定します。

```
省略 $ vi etc/init.d/S97appstart
```

以下のテキストを入力します。(XG-BBEXT の場合 S97appstart は用意されています)

[XG-3358]

```
#!/bin/sh

export LANG='ja_JP.UTF-8'
export TZ=JST-9
export TSLIB_CONSOLEDEVICE=none
export TSLIB_TSDEVICE=/dev/input/event0
export QWS_MOUSE_PROTO='tslib:/dev/input/event0'

APPPATH=アプリケーションのパス( "/"で終わること)
APPNAME=Qt アプリケーション名

case "$1" in
  start)
    # タッチパネルの補正データファイルがなければ、校正を開始する。
    if [ ! -f /etc/pointercal ]
    then
      ts_calibrate
    fi

    # 自動起動するアプリケーションのスク립トを記述します。
    echo "Starting $APPNAME: "
    ${APPPATH}/${APPNAME} -qws &
    echo "done $APPNAME"
    ;;

  stop)
    echo "Stopping $APPNAME "
    killall -9 $APPNAME
    echo "done $APPNAME"
    ;;

  restart)
    echo "Restarting $APPNAME "
    killall -9 $APPNAME
    ${APPPATH}/${APPNAME} -qws &
    echo "done $APPNAME"
    ;;

  *)
    echo "Usage: /etc/init.d/S97appstart {start|stop|restart}" >&2
    exit 1
    ;;

esac
```

[XG-BBEXT]

```

#! /bin/sh

export LANG='ja_JP.UTF-8'
export TZ=JST-9
export TSLIB_CONSOLEDEVICE=none
export TSLIB_TSDEVICE=/dev/input/event1
export QWS_MOUSE_PROTO='tslib:/dev/input/event1'

APPPATH=アプリケーションのパス("/"で終わること)
APPNAME=Qt アプリケーション名

case "$1" in
start)
    # CPUクロックの設定 (300000, 600000, 720000, 800000, 1000000)
    echo 1000000 >/sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed

    # オーディオ設定 再生レベル設定
    amixer set 'PCM' 127
    amixer set 'HP DAC' 118
    # オーディオ設定録音レベル設定
    amixer set 'PGA' 64

    # タッチパネルの補正データファイルがなければ、校正を開始する。
    if [ ! -f /etc/pointercal ]
    then
        ts_calibrate
    fi

    # 自動起動するアプリケーションのスク립トを記述します。
    echo "Starting $APPNAME: "
    ${APPPATH}/${APPNAME} -qws &
    echo "done $APPNAME"
    ;;

stop)
    echo "Stopping $APPNAME "
    killall -9 $APPNAME
    echo "done $APPNAME"
    ;;

restart)
    echo "Restarting $APPNAME "
    killall -9 $APPNAME
    ${APPPATH}/${APPNAME} -qws &
    echo "done $APPNAME"
    ;;

*)
    echo "Usage: /etc/init.d/S97apstart {start|stop|restart}" >&2
    exit 1
    ;;

esac

```



自動起動したアプリケーションが終了しないと、コンソールからログインできません。アプリケーション稼動中にデバッグ等でコンソールを使用するときは、例のようにアプリケーション起動時に「&」を付加してバックグラウンドでアプリケーションを稼動させてください。

アプリケーションの実行結果に基づいてアプリケーションの再起動、システムのリポート、停止 (halt) をする場合には、アプリケーション終了時に実行結果を返し、スクリプト側で「\$?» で実行結果を判断しアプリ再起動、システムリポート、システム停止をするようにします。



S97appstart などのファイルを編集した場合、使用するエディタによってはバックアップファイル「S97appstart~」などが作成されることがあります。/etc/init.d にこのバックアップファイルが残っていると、起動時に S97appstart と S97appstart~ の両方が実行されることとなります。そのためアプリ終了後再度アプリが起動したり、アプリの二重起動などの問題が発生することがあります。

⑤ デバッグ用の起動時処理を追加します。

```
省略 $ vi etc/init.d/K95debug ←入力
```

大文字の「S」で始まるスクリプト以外は起動時自動実行されませんが、通常は「K」で始まる名前とします。
ユーザがログイン後「/etc/init.d/K95debug start」のようにコマンド実行します。start で起動した内容を止めるには「/etc/init.d/K95debug stop」とします。

起動時に自動的に NFS マウントするするには以下のようにします。
デバッグ期間中だけ「S95debug」とリネームすれば起動時に自動的に NFS マウントがされています。

以下のテキストを入力します。(デバッグ環境に応じて修正してください)

```
#!/bin/sh

case "$1" in
  start)
    mount -t nfs -o nolock 192.168.128.210:/nfs /mnt/nfs
    ;;
  stop)
    echo -n "Stopping $NAME: "
    umount /mnt/nfs
    echo "done"
    ;;
  *)
    echo $"Usage: $0 {start|stop}"
    exit 1
esac

exit $?
```



K95debug を S95debug にリネームすることにより起動時に自動実行されます。ssh をインストールしてある場合は sftp,ssh を使用しますので ftp,telnetd などは必要ありません。自動起動する場合は開発用のボードだけ設定してください。製品として出荷するものには SSH (sshd) をインストールすることをお勧めします。

- ⑥ S97appstart、K95debug スクリプトの実行権限を与えます。

ファイルシステムをビルド後、ターゲットシステムを起動し chmod コマンドで実行権限を与えることも出来ませんがここでは、ルートファイルのビルド時に実行権限を与えておきます。

省略 \$ cd ~/xg3358-lk/buildroot-2011.11-xg3358-X.X/target/generic ←入力

上記は、LK-3358-A01 の時のディレクトリ指定となります。移動先ディレクトリは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのディレクトリ名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

機種（開発キット）	ディレクトリ
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X/system
XG-BBEXT	~/xgbbext-lk/buildroot-2013.11-xg33bbext-X.X/system

省略 \$ vi device_table.txt ←入力

以下の2行を追加します。

/etc/init.d/S97appstart	f	755	0	0	-	-	-	-	-
/etc/init.d/K95debug	f	755	0	0	-	-	-	-	-

以上設定が完了しましたら、ルートファイルシステムをビルドしなおします。

- ⑦ ビルドルートのディレクトリに移ります。

省略 \$ cd ~/xg3358-lk/buildroot-2013.11-xg3358-X.X ←入力

上記は、LK-3358-A01 の時のディレクトリ指定となります。移動先ディレクトリは機種（開発キット）ごとに異なりますので適宜変更してください。

なお、以下の表に各機種（開発キット）ごとのディレクトリ名を記載しますが、バージョンによって異なる場合がありますので、各機種（開発キット）のソフトウェアマニュアルを参照ください。

機種（開発キット）	ディレクトリ
XG-3358(LK-3358-A01)	~/xg3358-lk/buildroot-2013.11-xg3358-X.X
XG-BBEXT	~/xgbbext/buildroot-2013.11-xgbbext-X.X

- ⑧ ビルドします。

スケルトン部分の変更は make 単独では反映されませんので、make clean から行います。

省略 \$ make clean ←入力

省略 \$ make ←入力

7. 関連情報

Qt に関する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/

Table 7-2 参考 Web サイト

ご注意

- ・本文書の著作権は、株式会社アルファプロジェクトが保有します。
- ・本文書の内容を無断で転載することは一切禁止します。
- ・本文書に記載されているサンプルプログラムの著作権は、株式会社アルファプロジェクトが保有します。
- ・本文書に記載されている内容およびサンプルプログラムについての技術サポートは一切受け付けておりません。
- ・本文書の内容およびサンプルプログラムに基づき、アプリケーションを運用した結果、万一損害が発生しても、弊社では一切責任を負いませんのでご了承下さい。
- ・本文書の内容については、万全を期して作成いたしました。が、万一ご不審な点、誤りなどお気付きの点がありましたら弊社までご連絡下さい。
- ・本文書に記載されているプログラム、データ等は執筆時点のライセンス規定をもとに解説してあります。ライセンス規定は変更になる場合もありますのでそれらソフトウェア、データをご利用の都度ライセンス規定をご確認ください。
- ・本文書の内容は、将来予告なしに変更されることがあります。

商標について

- ・ Windows®の正式名称は Microsoft®Windows®Operating System です。
Microsoft、Windows、Windows NT は、米国 Microsoft Corporation.の米国およびその他の国における商標または登録商標です。
Windows®7、Windows®Vista、Windows®XP は、米国 Microsoft Corporation.の商品名称です。
本文書では下記のように省略して記載している場合がございます。ご了承ください。
Windows®7 は、Windows 7 もしくは Win7
Windows®Vista は、Windows Vista もしくは WinVista
Windows®XP は、Windows XP もしくは WinXP
- ・ その他の会社名、製品名は、各社の登録商標または商標です。



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