XG シリーズ
Qt のためのルートファイルシステム作成方法

Rev1.1　2023/10/02

目　次

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、 入力 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  入力
```

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、 入力 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/images/uInitrd-xg3730  入力
```

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、VMware Player を含めた開発環境が WindowsPC 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、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　対象開発キット

本アプリケーションノートは、以下の開発キットのバージョンを対象とします。

開発キット	バージョン
LK-1808-A01(XG-1808)	Rev1.1 以降
LK-3517-A01(XG-3517)	Rev1.2 以降
LK-3730-A01(XG-3730)	Rev1.1 以降

バージョンは、付属 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、各開発キットの更新履歴でご確認ください。

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。此処では 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 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 4.7.4 です。Qt4.5 以降は LGPL が利用できます。GPL,LGPL では、不都合な場合は SRS 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。

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動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 などに格納します。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 や Kdrive(Xfbdev)、DirectFB、framebuffer と組み合わせるなどの方法があります。
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 directFB をバックエンドにした構成を例にとって説明します。

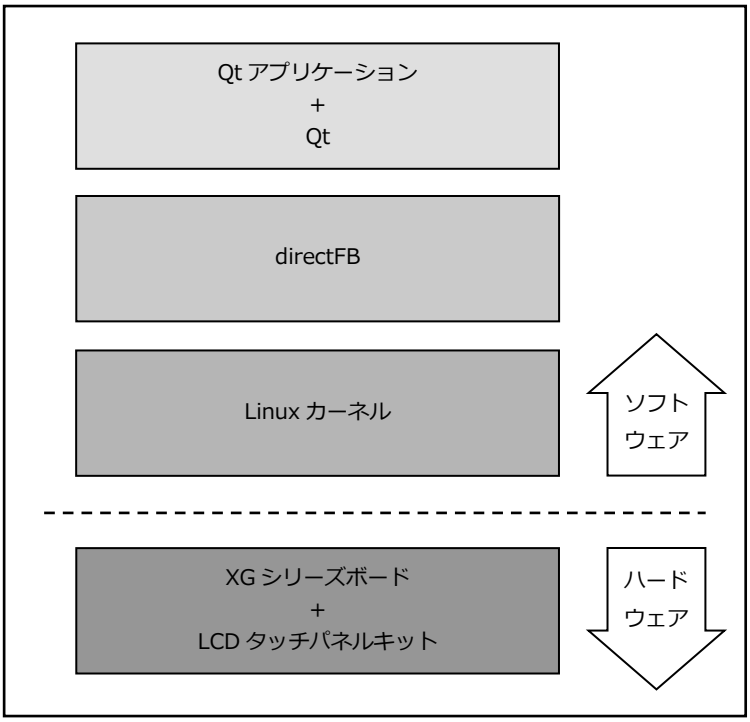


Fig 2.2-1　Qt システム例

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ルートファイルシステム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は、buildroot でファイルシステムを構築後、書き換えることも可能です。)

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 用にビルドするする準備として次の手順で進めます。

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 ~/xg3730/buildroot-2011.11-xg3730-X.X    入力
```

上記は、LK-3730-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、各開発キットのソフトウェアマニュアルを参照ください。

開発キット	ディレクトリ
LK-1808-A01(XG-1808)	~/xg1808/buildroot-2011.11-xg1808-X.X
LK-3517-A01(XG-3517)	~/xg3517-lk/buildroot-2011.11-xg3517-X.X
LK-3730-A01(XG-3730)	~/xg3730/buildroot-2011.11-xg3730-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 distcclean」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ます。

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 xg3730_defconfig  入力
```

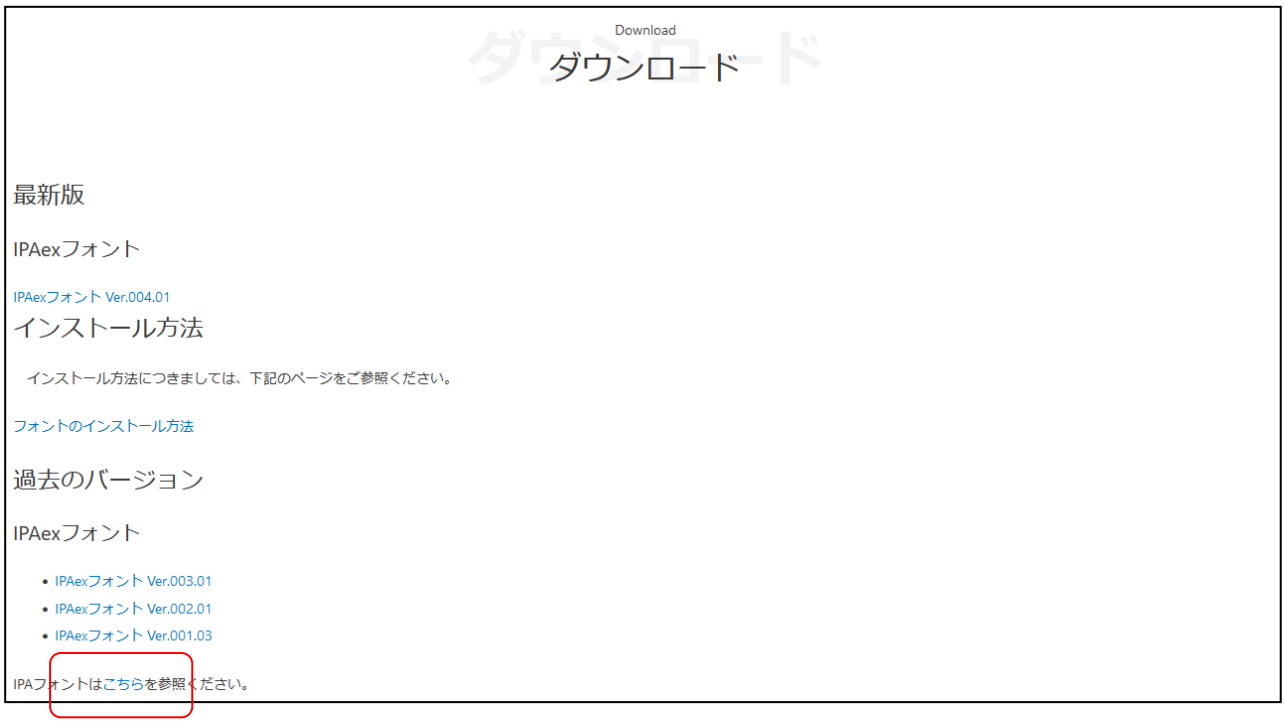
上記は、LK-3730-A01 の時のコマンドとなります。コマンドは開発キットごとに異なりますので適宜変更してください。なお、以下の表に各開発キットごとのコンフィグレーション名を記載しますが、バージョンによって異なる場合がありますので、各開発キットのソフトウェアマニュアルを参照ください。

開発キット	コンフィグレーション名
LK-1808-A01(XG-1808)	xg1808_defconfig
LK-3517-A01(XG-3517)	xg3517_defconfig
LK-3730-A01(XG-3730)	xg3730_defconfig

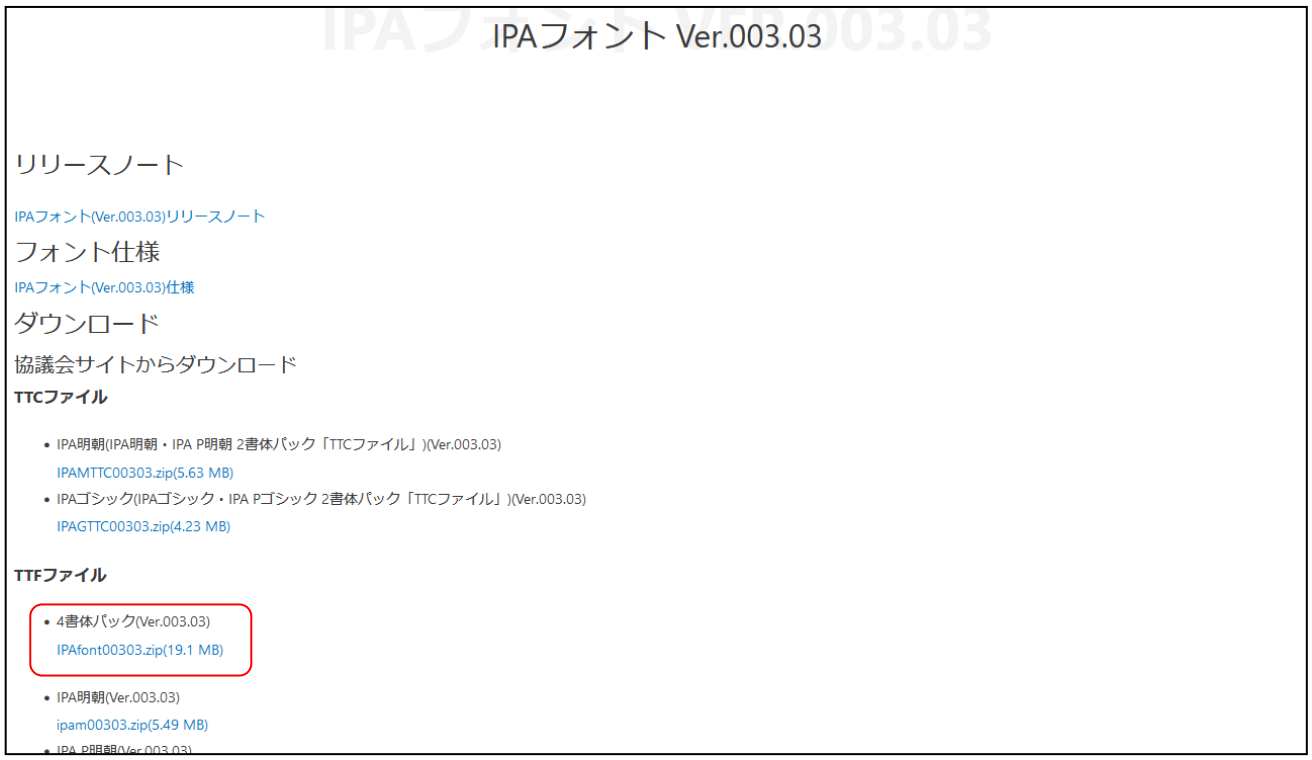
「make distclean」は「make clean」に加えて、Makefile, config.cache などのファイルも削除されます。Makefile のコンフィグレーション設定を変更した場合は「make distclean」を実行する必要があります。

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「https://moji.or.jp/ipafont」からダウンロードすることが出来ます。
「ダウンロード ＞ IPA フォントはこちらを参照ください。」と辿ってください。



② IPA フォントの 4 書体パック IPAfont00303.zip(19.1 MB)をダウンロードします。保存先は「qt-work」とします。



③ フォントを展開します。

```
省略 $ cd ~                                          入力
省略 $ unzip qt-work/IPAfont00303.zip                入力
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 ~/xg3730/buildroot-2011.11-xg3730-X.X/fs/skeleton-xg3730    入力
```

上記は、LK-3730-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、各開発キットのソフトウェアマニュアルを参照ください。

開発キット	ディレクトリ
LK-1808-A01(XG-1808)	~/xg1808/buildroot-2011.11-xg1808-X.X/fs/skeleton-xg1808
LK-3517-A01(XG-3517)	~/xg3517-lk/buildroot-2011.11-xg3517-X.X/fs/skeleton-xg3517
LK-3730-A01(XG-3730)	~/xg3730/buildroot-2011.11-xg3730-X.X/fs/skeleton-xg3730

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    入力
```

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。ここでは IPA ゴシックをインストールします。

```
省略 $ cd usr/lib/fonts    入力
省略 $ cp ~/IPAfont00303/ipag.ttf .    入力
省略 $ ls -l    入力
合計 6092
-rw-r--r-- 1 guest guest 6235344 2012-05-07 18:04 ipag.ttf
```

インストールするフォントはメモリサイズなどを考慮してください。microSD では 4 書体インストールも可能です。

ライセンスの関係で必要であれば以下のファイルもコピーします。
IPA_Font_License_Agreement_v1.0.tx
Readme_IPAfont00303.tx

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動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となる、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 ~/xg3730/buildroot-2011.11-xg3730-X.X/fs/skeleton-xg3730  入力
```

上記は、LK-3730-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、各開発キットのソフトウェアマニュアルを参照ください。

開発キット	ディレクトリ
LK-1808-A01(XG-1808)	~/xg1808/buildroot-2011.11-xg1808-X.X/fs/skeleton-xg1808
LK-3517-A01(XG-3517)	~/xg3517-lk/buildroot-2011.11-xg3517-X.X/fs/skeleton-xg3517
LK-3730-A01(XG-3730)	~/xg3730/buildroot-2011.11-xg3730-X.X/fs/skeleton-xg3730

②　必要に応じてネットワーク設定します。

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

③　必要に応じて FTP 用の設定をします。

```
省略 $ echo 21 stream tcp nowait root /usr/sbin/ftpd ftpd -w /home > etc/inetd.conf  入力
```

⑥ etc/profile へ環境変数を追加します。

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用な環境は異なります。

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

```
省略 $ vi etc/profile  入力
```

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。

```
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'
```

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 ~/xg3730/buildroot-2011.11-xg3730-X.X
```
入力

上記は、LK-3730-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、各開発キットのソフトウェアマニュアルを参照ください。

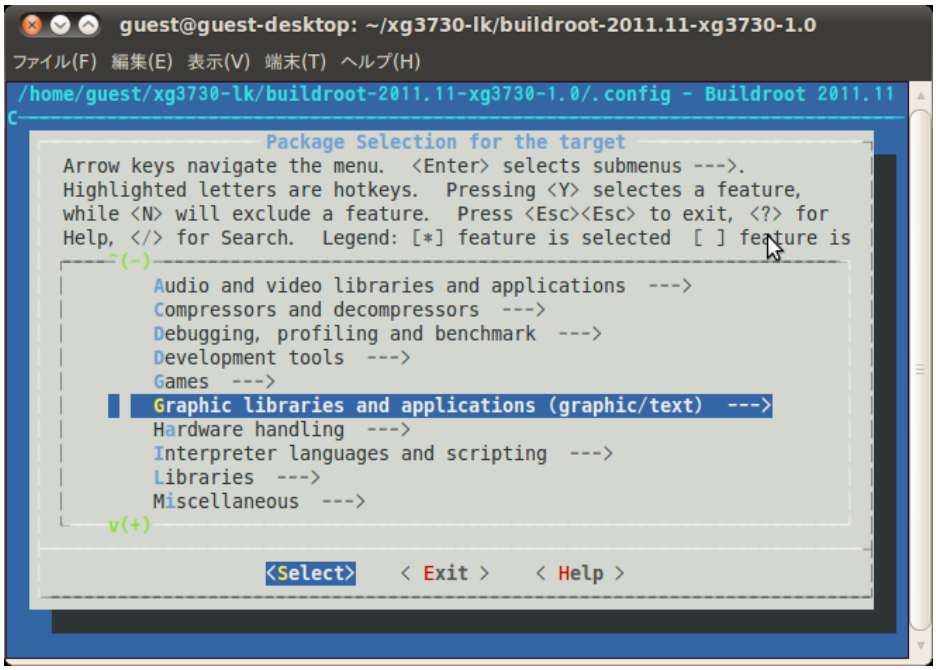
開発キット	ディレクトリ
LK-1808-A01(XG-1808)	~/xg1808/buildroot-2011.11-xg1808-X.X
LK-3517-A01(XG-3517)	~/xg3517-lk/buildroot-2011.11-xg3517-X.X
LK-3730-A01(XG-3730)	~/xg3730/buildroot-2011.11-xg3730-X.X

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。

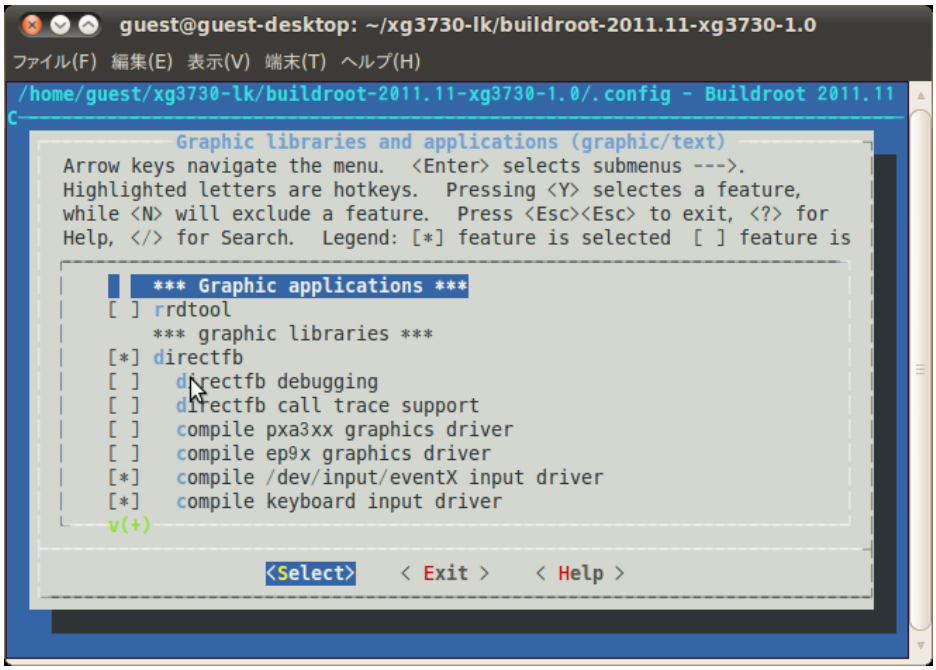
```
省略 $ make menuconfig
```
入力

③ Qt に関する設定は「Package Selection for the target」の中にありますので、それを選択し Enter キーを押します。

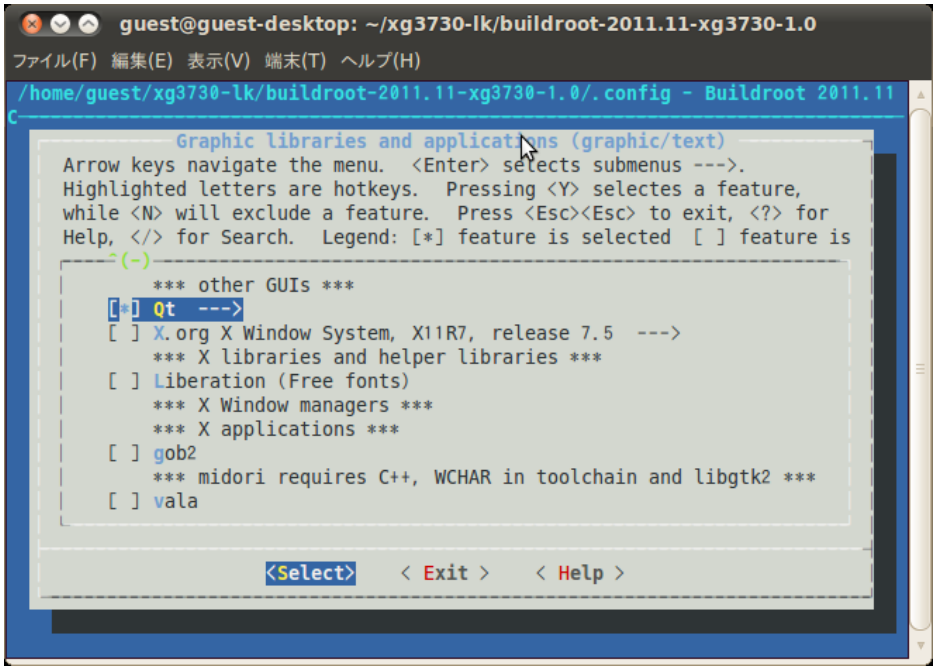
④ 「Graphic Libraries and applications (graphic/text)」に移動し、スペースキーあるいは Enter キーで選択します。



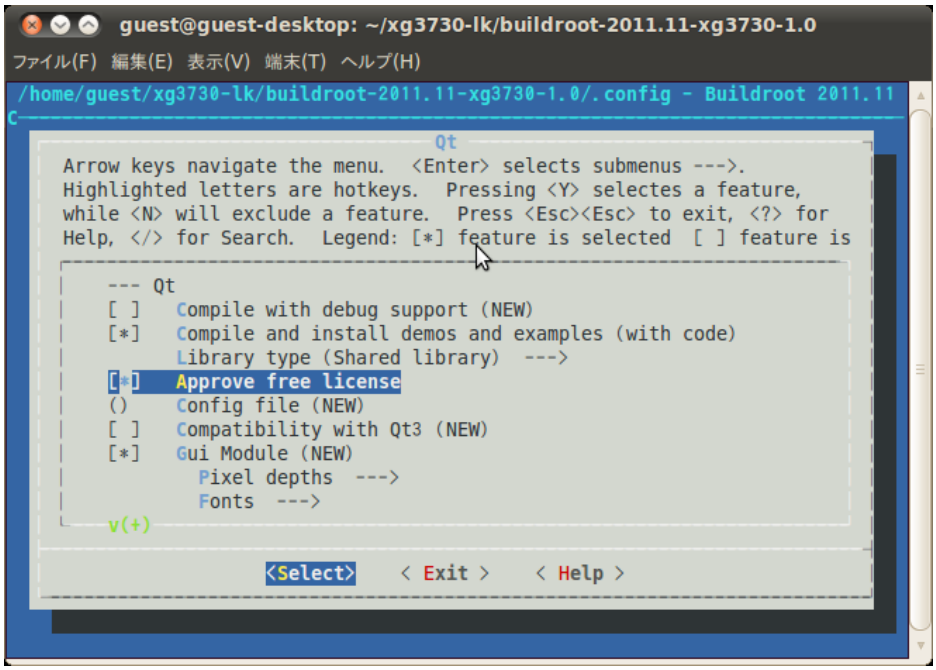
⑤　「↓」カーソルキーで移動し、Qt の項目まで移動します。



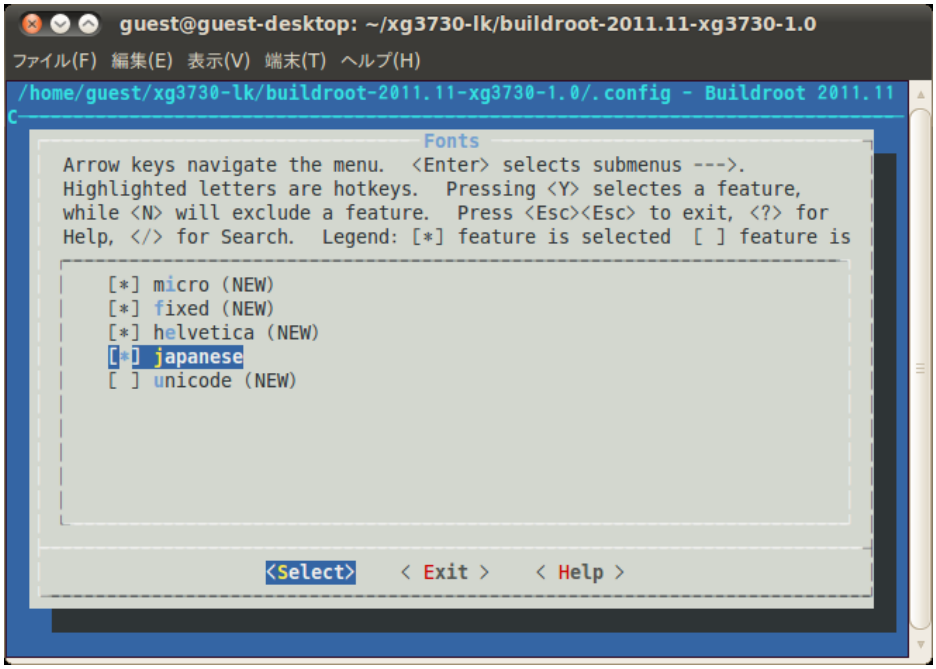
⑥　スペースキーを押して Qt 項目にチェックをいれます（＊マークがつきます）。Qt の詳細設定をするために、Enter キーを押します。



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。

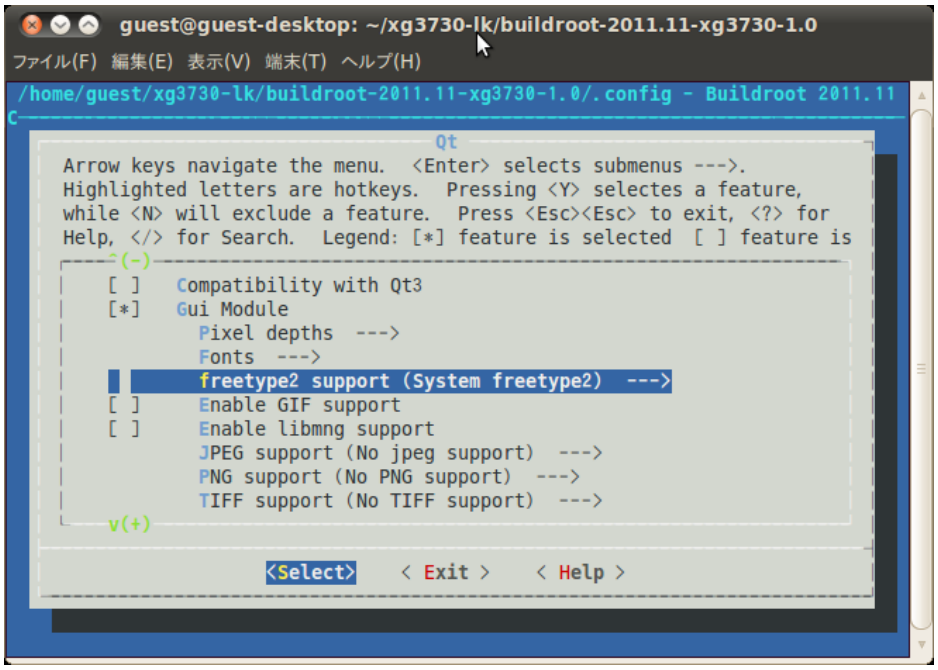


⑧　「Gui Module」がチェックされていることを確認し、その中の Fonts を選択し Enter キーを押して、Fonts メニューにて japanese を追加選択します。

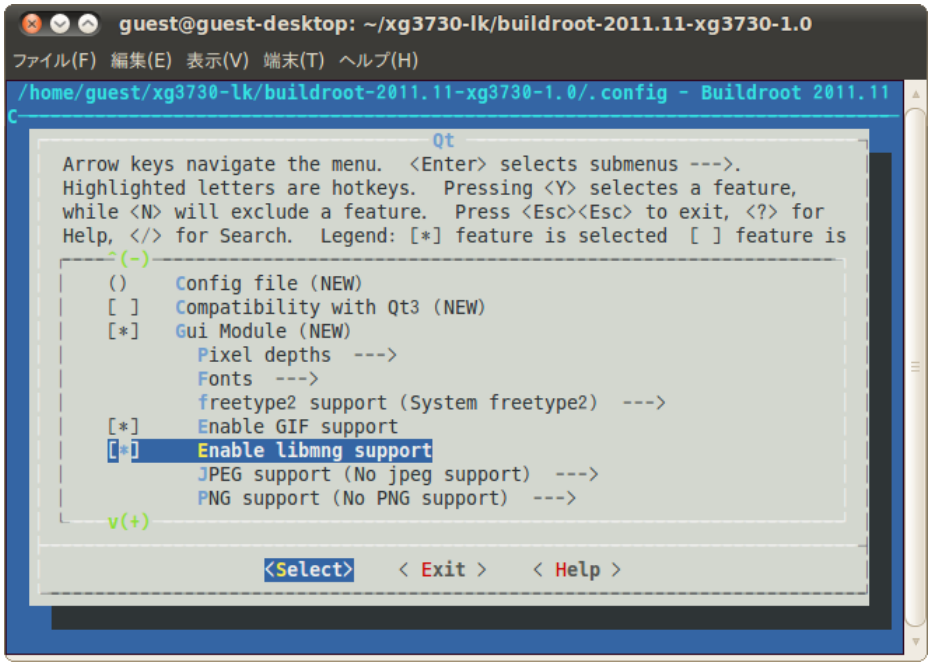


ESC-ESC で一段上のメニューに戻ります。

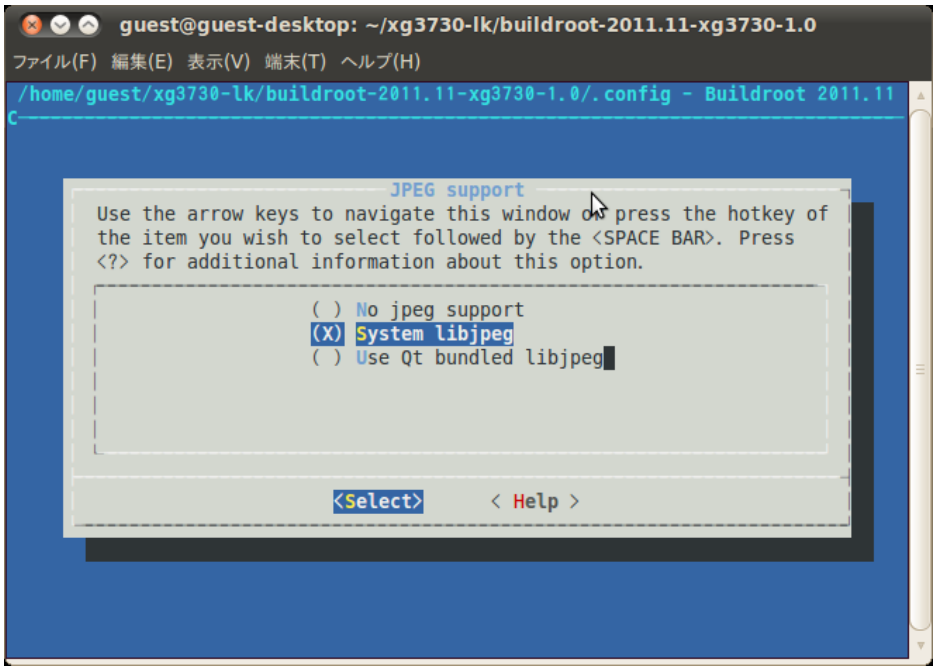
⑨　「freetype2 support」を「system freetype2」に設定します。



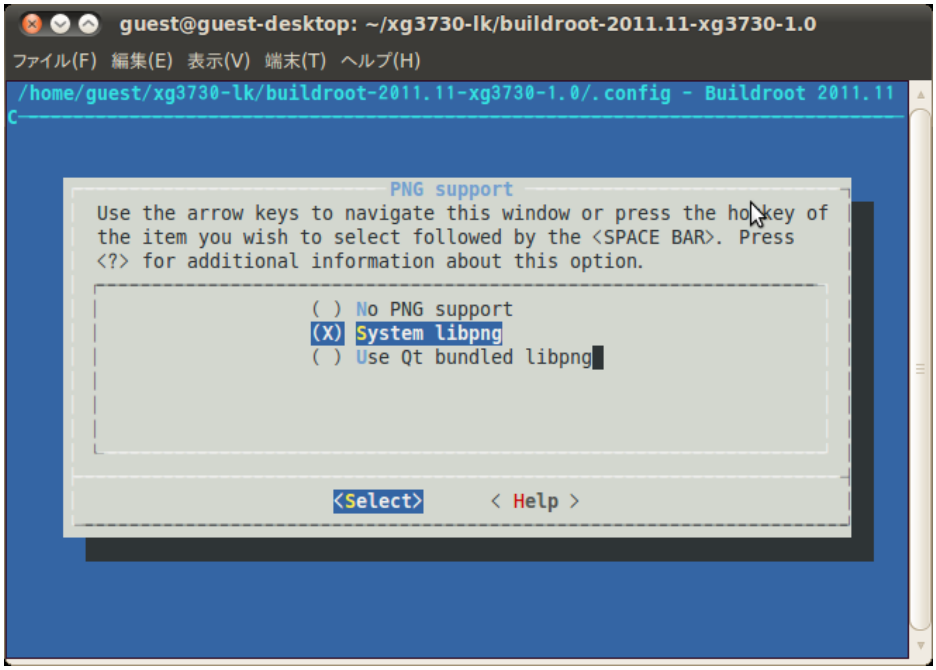
⑩　Enable GIF support を選択しスペースを押してチェックを入れます。
png 画像のアニメーションをサポートするときは「Enable libmng support」にもチェックを入れます。



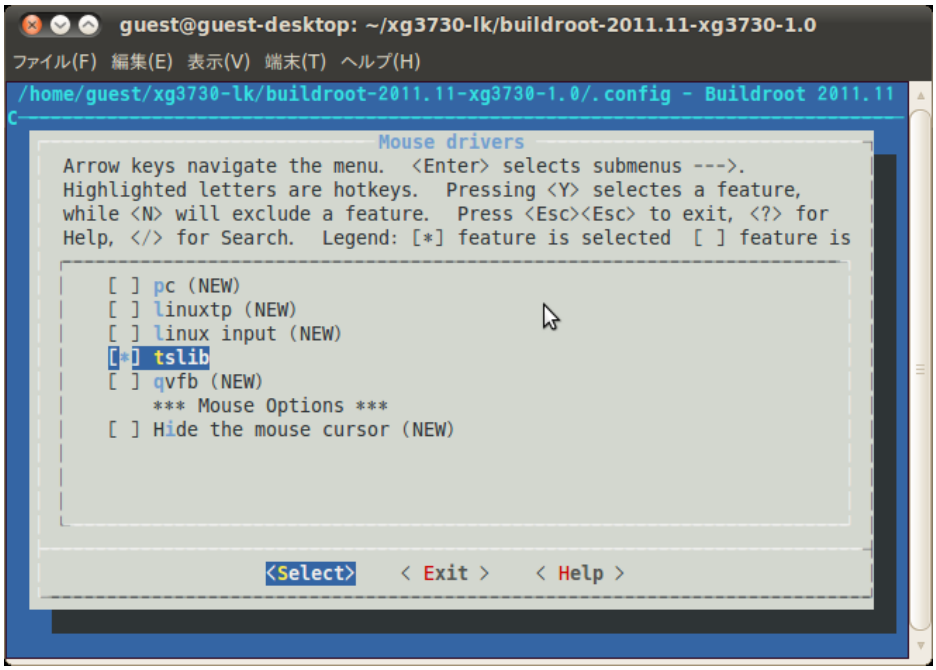
⑪　JPEG support を選択し、JPEG support メニューにて libjpeg を選択します。



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

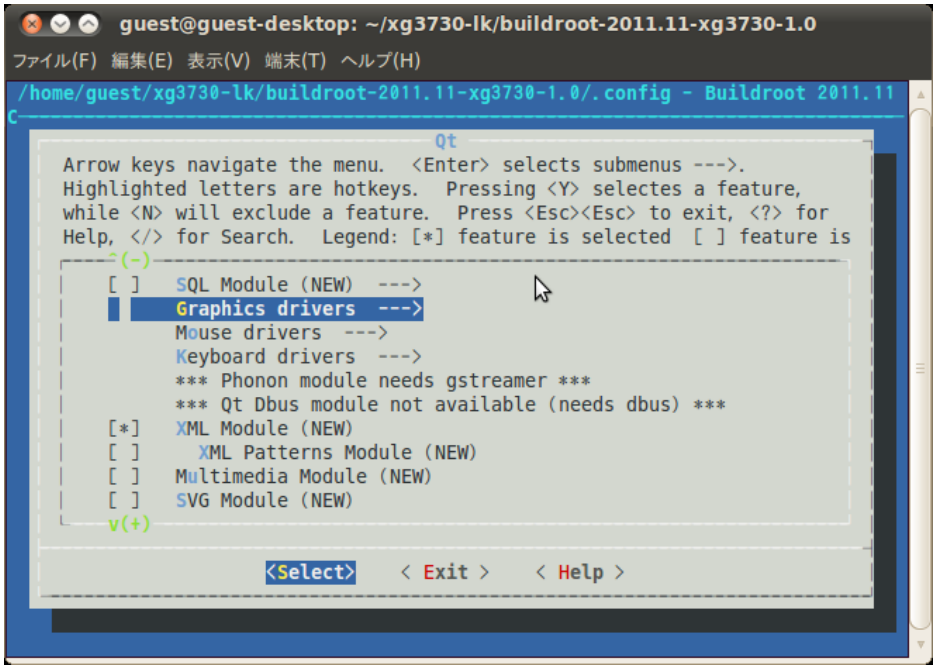


⑬　Mouse drivers を選択し Enter キーを押し Mouse drivers メニューにて、tslib を選択します。

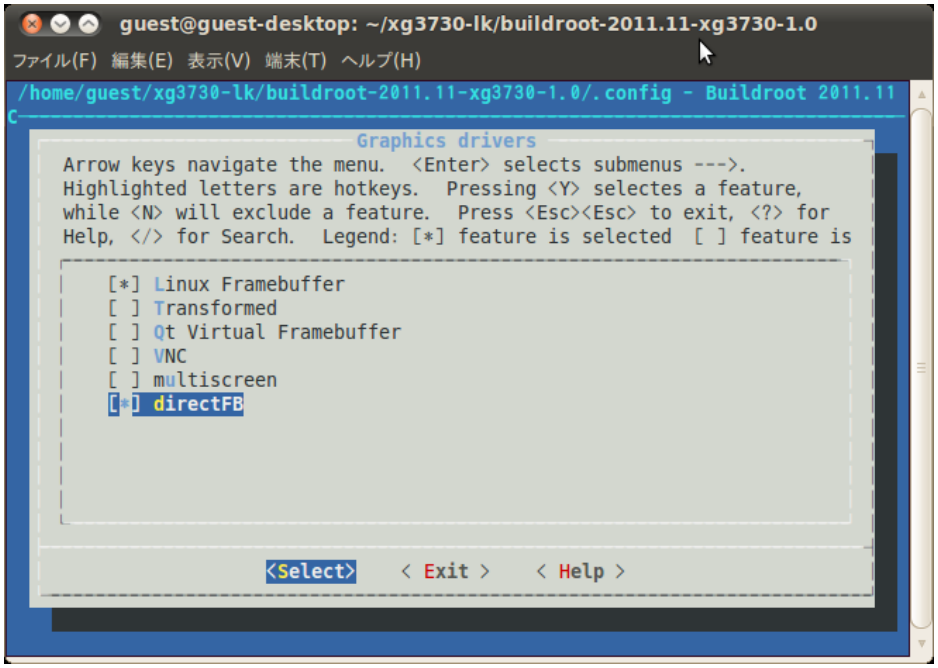


ESC-ESC で Qt のメニューに戻ります。

⑭　「Graphics drivers」を選択し、Enter キーを押します。

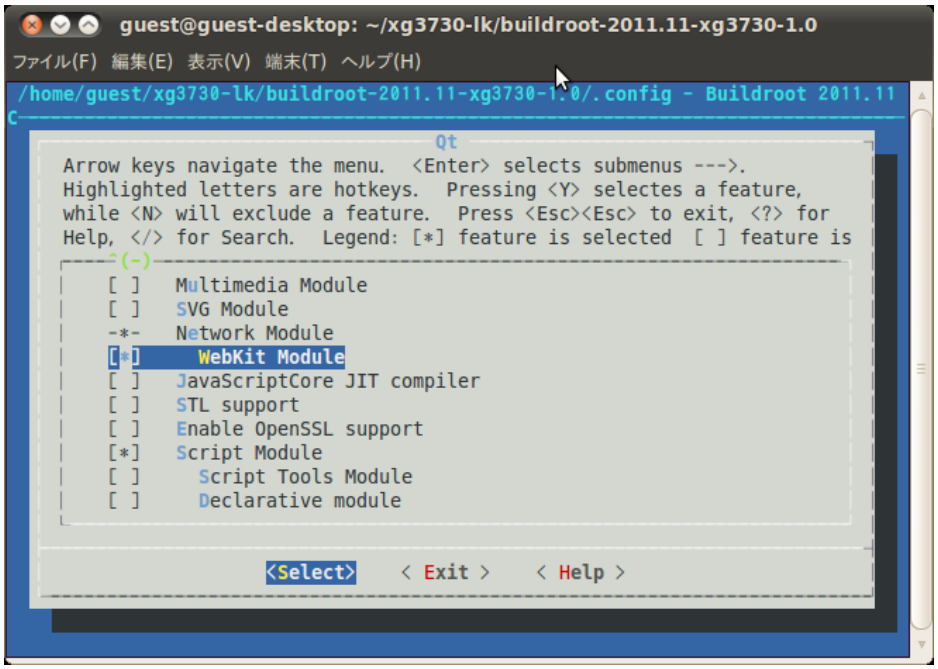


⑮　「Linux Framebuffer」と「directFB」を選択します。



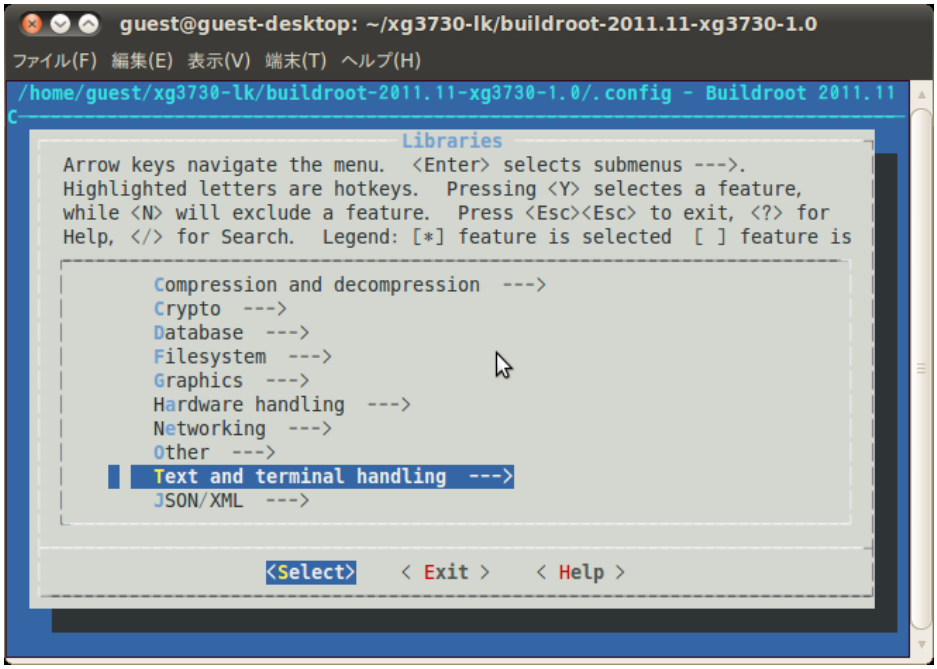
ESC-ESC で Qt のメニューに戻ります。

⑯　「WebKit Module」を選択します。

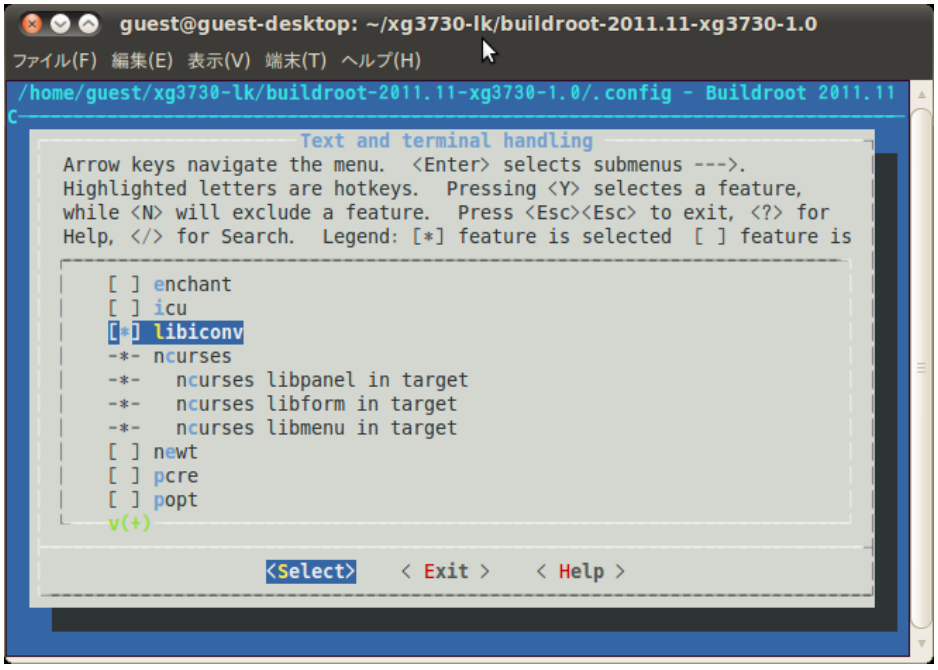


ESC-ESC、ESC-ESC にて「Package Selection for the taget」メニューまで戻ります。

⑰　Libraries を選択し「Text and terminal handling」を選択します。



⑱　文字コード変換ライブラリ「libiconv」を選択します。



⑲　一通りの設定が終了したら ESC-ESC、ESC-ESC、ESC-ESC、ESC-ESC を押して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　ルートファイルシステムのビルド

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。

```
省略 $ make　入力
```

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。（Ubuntu11.10 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である。
- menuconfig で選択したパッケージが足りない。

4.　SD ルートファイルシステム

4.1　SD ルートファイルシステムの作成

① 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    入力
output/images/rootfs.tar.gz
```

② 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。
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。

③ 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    入力
[sudo] password for guest:    入力
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.
```

④ 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    入力
[sudo] password for guest:    入力
```

⑤ 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    入力
[sudo] password for guest:    入力
```

⑥ アンマウントします。

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

4.2　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-xg3730-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開発キットによって異なります。下記の手順は LK-3730-A01 の操作例です。その他の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。

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
LK-1808-A01(XG-1808)	c0700000	uImage-xg1808-lcdkit
LK-3517-A01(XG-3517)	80200000	uImage-xg3517-lcdkit
LK-3730-A01(XG-3730)	80200000	uImage-xg3730-lcdkit

①　Linux カーネルイメージ『**uImage-xg3730-lcdkit**』を RAM 上にダウンロードします。

```
=> tftp 80200000 uImage-xg3730-lcdkit    入力
smc911x: detected LAN9221 controller
smc911x: phy initialized
smc911x: MAC 00:0c:7b:33:XX:XX

         :
     途中省略
         :

done
Bytes transferred = 3887472 (3b5170 hex)
```

②　環境変数『**bootargs**』に『**root=/dev/mmcblk0p1 rootwait**』のパラメータを追加します。

```
=> setenv bootargs $bootargs root=/dev/mmcblk0p1 rootwait    入力
```

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 80200000    入力
## Booting kernel from Legacy Image at 80200000 ...
   Image Name:   Linux-3.2.0-xg3730-X.X
   Image Type:   ARM Linux Kernel Image (uncompressed)

         :
     途中省略
         :

Welcome to Buildroot
buildroot 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    入力
The system is going down NOW!
Sent SIGTERM to all processes
Requesting system halt
musb-hdrc musb-hdrc: remove, state 4
usb usb1: USB disconnect, device number 1
musb-hdrc musb-hdrc: USB bus 1 deregistered
System halted.
```

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。

```
Welcome to Buildroot
buildroot login: root      入力
# printenv      入力
HISTFILESIZE=1000
INPUTRC=/etc/inputrc
TSLIB_TSDEVICE=/dev/input/event0
USER=root
HOSTNAME=buildroot
OLDPWD=/mnt/nfs/mplayer
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/event0           タッチパネルデバイスを指定
EDITOR=/bin/vi
```

5.2　DirectFB の動作確認

最初に DirectFB での LCD 表示ができるかどうか確認します。

```
# df_andi
   ~~~~~~~~~~~~~~~~~~~~~~~~~~| DirectFB 1.4.15 |~~~~~~~~~~~~~~~~~~~~~~~~~~
        (c) 2001-2010  The world wide DirectFB Open Source Community
        (c) 2000-2004  Convergence (integrated media) GmbH
      ------------------------------------------------------------------

(*) DirectFB/Core: Single Application Core. (2012-03-27 06:45)

          :
      途中省略
          :

(*) FBDev/Mode: Switched to 800x480 (virtual 800x480) at 16 bit (RGB16), pitch 1600
(*) Direct/Interface: Loaded 'FT2' implementation of 'IDirectFBFont'.
(*) Direct/Interface: Loaded 'PNG' implementation of 'IDirectFBImageProvider'.
(*) Direct/Interface: Loaded 'JPEG' implementation of 'IDirectFBImageProvider'.
```

デモ画面が表示されていれば OK です。

終了するには CTRL-C(Ctrl キーを押しながら C を押します)

5.3　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
```
入力

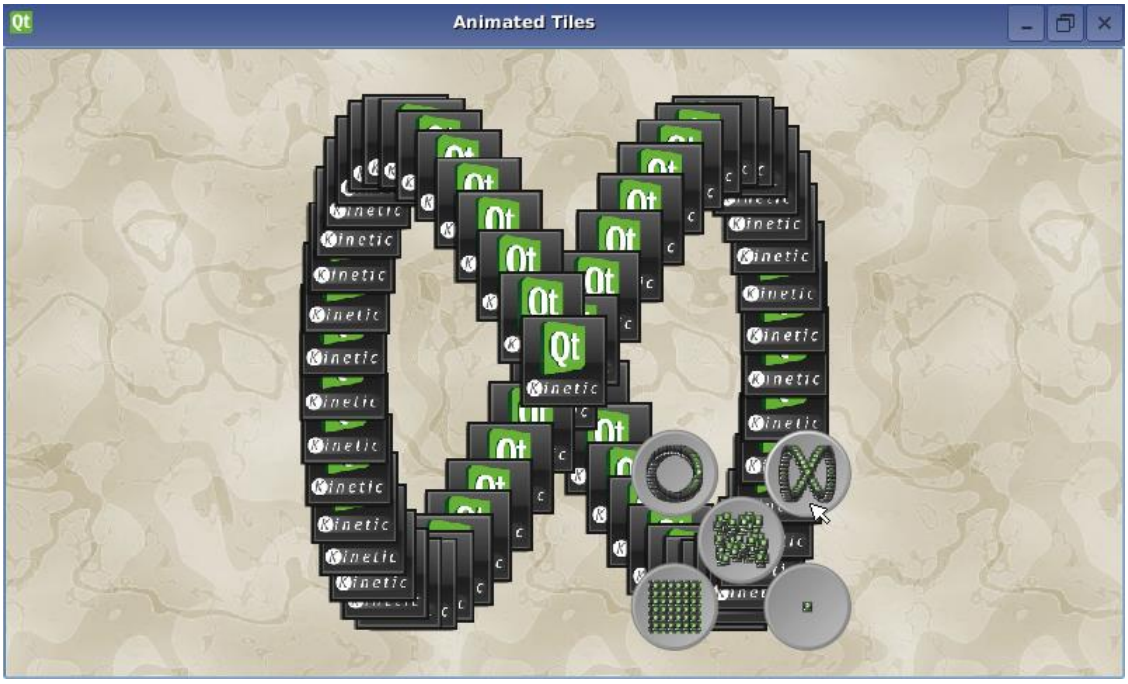
「+」が表示されますので、順にその位置をタッチしてください。タッチパネルのキャリブレーション情報は「/etc/pointercal」に格納されます。

5.4　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。

デモプログラムで画面表示、およびタッチパネル操作が効くことを確認します。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    入力
# ./animatedtiles -qws    入力
```



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。

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 ~/xg3730/buildroot-2011.11-xg3730-X.X/fs/skeleton-xg3730  入力
```

上記は、LK-3730-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、各開発キットのソフトウェアマニュアルを参照ください。

開発キット	ディレクトリ
LK-1808-A01(XG-1808)	~/xg1808/buildroot-2011.11-xg1808-X.X/fs/skeleton-xg1808
LK-3517-A01(XG-3517)	~/xg3517-lk/buildroot-2011.11-xg3517-X.X/fs/skeleton-xg3517
LK-3730-A01(XG-3730)	~/xg3730/buildroot-2011.11-xg3730-X.X/fs/skeleton-xg3730

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ので、此処でも環境変数も指定します。

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

以下のテキストを入力します。

```
# cat /etc/init.d/S97appstart
#! /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'

#  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リプトを記述します。
```

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、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、システムのリブート、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、システムリブート、システム停止をするようにします。

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。

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)

```
# cat /etc/init.d/K95debug
#! /bin/sh

case "$1" in
  start)
# FTP など、inet デーモンを使用する場合
        inetd
# Telnet を使用する場合
        telnetd
# NFS で開発 LinuxPC の NFS サーバをアクセスする場合
        sleep 5
        mount -t nfs -o nolock 192.168.128.201:/nfs /mnt/nfs
        ;;
  stop)
        echo -n "Stopping $NAME: "
        killall -9 inetd
        killall -9 telnetd
        umount /mnt/nfs
        echo "done"
        ;;
  *)
        echo $"Usage: $0 {start|stop }"
        exit 1
esac

exit $?
```

K95debug を S95debug にリネイムすることにより起動時に自動実行されます。ftp,telnetd などを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すが
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 ~/xg3730/buildroot-2011.11-xg3730-X.X/target/generic　入力
```

上記は、LK-3730-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、
各開発キットのソフトウェアマニュアルを参照ください。

開発キット	ディレクトリ
LK-1808-A01(XG-1808)	~/xg1808/buildroot-2011.11-xg1808-X.X/target/generic
LK-3517-A01(XG-3517)	~/xg3517-lk/buildroot-2011.11-xg3517-X.X/target/generic
LK-3730-A01(XG-3730)	~/xg3730/buildroot-2011.11-xg3730-X.X/target/generic

```
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  -  -  -  -  -
```

⑦　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。

ビルドルートのディレクトリに移ります。

```
省略 $ cd ~/xg3730/buildroot-2011.11-xg3730-X.X　入力
```

上記は、LK-3730-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、
各開発キットのソフトウェアマニュアルを参照ください。

開発キット	ディレクトリ
LK-1808-A01(XG-1808)	~/xg1808/buildroot-2011.11-xg1808-X.X
LK-3517-A01(XG-3517)	~/xg3517-lk/buildroot-2011.11-xg3517-X.X
LK-3730-A01(XG-3730)	~/xg3730/buildroot-2011.11-xg3730-X.X

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 4 プログラミング	Mark Summerfield	オライリージャパン
Qt プログラミング入門 – 使いやすいフレームワークを基礎から解説	津田 信秀	工学社

Table 7-1 参考書籍

サイト名	URL
digia	http://www.digia.com
Qt	https://www.qt.io
Qt 開発ツール	https://www.qt.io/product/development-tools
Qt wiki	https://wiki.qt.io/Main
Qt Creator 日本語化プロジェクト	http://qt-creator-jp.sourceforge.jp/

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。

・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
静岡県浜松市中央区積志町 834
https://www.apnet.co.jp
E-Mail: query@apnet.co.jp