XG Series

XG-BBEXT

BeagleBone Black Extension Board
Cortex-A8 AM3358 CPU

Android Software Manual

Rev 1.0

ダイジェスト版

ALPHAPROJECT

http://www.apnet.co.jp

目　次

1. 概要

1.1 はじめに

XG-BBEXT は、CPU コアに ARM Cortex-A8 を採用したマイクロプロセッサ「AM3358」(TEXAS INSTRUMENTS)を搭載した汎用 CPU ボードで、標準 OS に Linux を採用しています。
Linux を採用することで、世界中のプログラマによって日々開発される膨大なオープンソースソフトウェア資産をロイヤリティフリーで利用することができます。
本ドキュメントでは、XG-BBEXT の動作方法をはじめ、SPL、U-Boot、Linux カーネル、Android 開発のための手順を説明します。

本ドキュメントでは、VirtualBox 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、『**Android 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。

1.2 Linux について

Linux とは 1991 年に Linus Torvalds 氏によって開発された、オープンソースの UNIX 互換オペレーティングシステムです。
Linux はオープンソース、ロイヤリティフリーという特性から、世界中のプログラマたちにより日々改良され、今では大手企業のサーバーや、行政機関などにも広く採用されています。
また、Linux の特長として CPU アーキテクチャに依存しないということがあげられます。これは、GNU C コンパイラの恩恵にもよるものですが、数多くのターゲット(CPU)に移植されており、デジタル家電製品を中心に非 PC 系製品にも採用されるようになりました。
Linux は、カーネルと呼ばれる OS の核となる部分とコマンドやユーティリティなど多くのソフトウェアから構成されます。これらのソフトウェアの多くは FSF の GNU プロジェクトによるフリーソフトウェアです。
本ドキュメントでは、Linux のごく一部の機能と使い方のみを説明しています。
Linux 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 U-Boot について

U-Boot は、DENX Software Engineering 社の Wolfgang Denk 氏が保守を行っているオープンソフトウェアの汎用ブートローダです。多くの開発者によって支援され、現在最も機能が豊富で柔軟性に富み、開発が活発に行われています。対応しているアーキテクチャは、SuperH、PPC、ARM、AVR32、MIPS、x86、68k、Nios、MicroBlaze などです。またプログラムのダウンロードに関しても、ネットワークを介した TFTP の他に、CF カード、SD メモリカードなどのストレージデバイスからのダウンロードにも対応しています。

2. システム概要

2.1 システム概要

XG-BBEXT は、CPU コアに ARM Cortex-A8 を採用したマイクロプロセッサ「AM3358」(TEXAS INSTRUMENTS)を搭載した汎用 CPU ボードです。
Linux システムは、ブートローダと Linux カーネル、Android から構成されます。ブートローダに SPL と U-Boot、Linux カーネルに Linux-3.2、Android にはバージョン 4.2 を使用します。

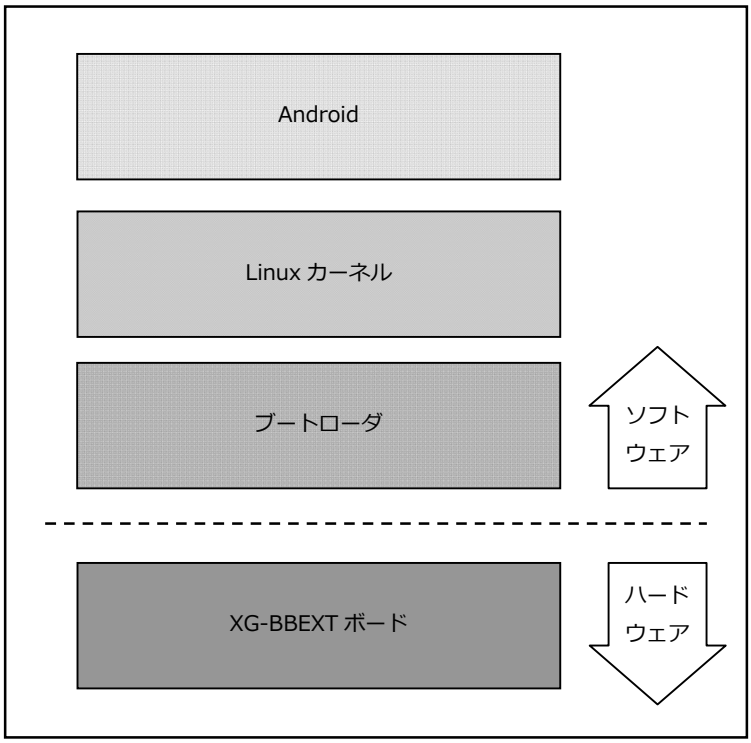


Fig 2.1-1 XG-BBEXT システム概要図

2.3 Android

Android は、カーネル、ミドルウェア、ユーザーインタフェース、Web ブラウザ等のソフトウェアを 1 つのパッケージにして提供されています。
Android ランタイムには、Dalvik VM(Dalvik 仮想マシン)が存在し、Android 用のアプリケーションは、基本的にこの DalvikVM 上で動作し、プレインストールアプリケーションと後からインストールするアプリケーションを区別なく扱います。

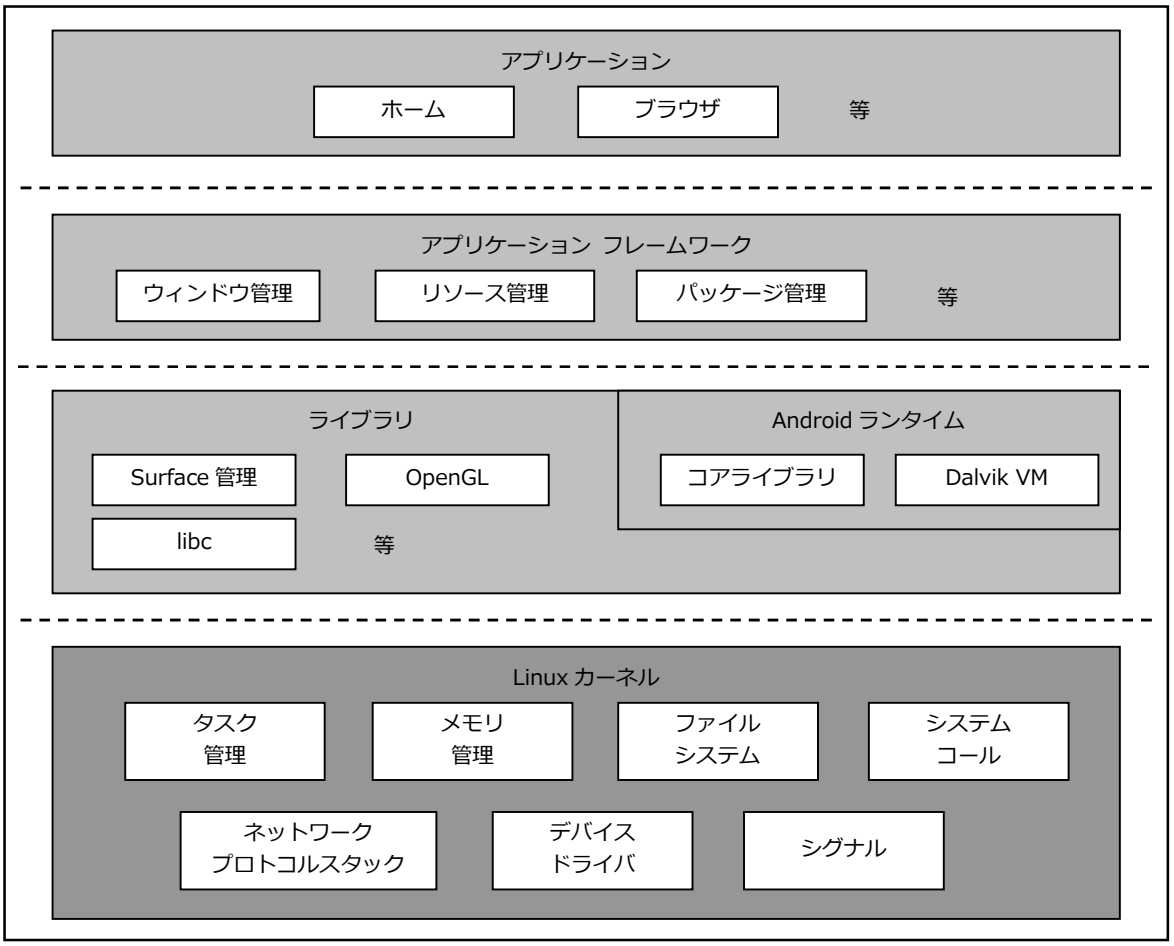


Fig 2.3-1 Android システム概要

2.5 添付 DVD-ROM の構成(Android 開発関連のみ)

XG-BBEXT の Android の開発には、Linux カーネルソース、Android ソースファイル、クロスコンパイラ等が必要です。
これらは、弊社ホームページ及び関連リンクからダウンロードするか、添付 DVD-ROM から入手することができます。

```
`-- android
    |-- binaries
    |   |-- MLO                              : SPL バイナリ
    |   |-- rootfs.tar.gz                    : Android システムバイナリ
    |   |-- u-boot.img                       : U-Boot バイナリ
    |   |-- uImage                           : Linux カーネルバイナリ
    |   `-- image
    |       `-- xgbbext_android.zip          : microSD カードイメージファイル
    |-- index.html                           : インデックス HTML
    |-- index_images                         : インデックス HTML イメージ
    |-- license
    |   |-- fdl.txt                          : GFDL 原文
    |   |-- gpl.txt                          : GPL 原文
    |   |-- lgpl.txt                         : LGPL 原文
    |   `-- LICENSE-2.0.txt                  : Apache License Version2.0 原文
    |-- manual
    |   |-- ak_install_xgbbext.pdf           : Android 開発 インストールマニュアル
    |   `-- xgbbext_android_sw.pdf           : Android ソフトウェアマニュアル
    |-- sample
    |   `-- xg_bbext_demo-X.X.tar.bz2        : サンプルアプリソース
    `-- sources
        `-- android_kit_xgbbext-X.X.tar.bz2  : XG-BBEXT Android DevKit 一式
```

Table 2.6-1 DVD-ROM 内容

※『X.X』はバージョン番号を示します。バージョン 1.0 の場合は『1.0』になります。

3. システムの動作

3.1 動作環境

Linux の起動を確認するためには、CPU ボードと以下の環境が必要です。

●ホスト PC

Linux では PC をコンソール端末として使用します。

XG-BBEXT にはコンソールポートのシリアル・USB 変換機能が内蔵されており、XG-BBEXT と PC を USB ケーブルで接続することで、PC 上では仮想シリアルポートとして認識します。

なお、仮想シリアルポートを使用した通信には、ハイパーターミナル等のターミナルソフトウェアが別途必要となります。

使用機器等	環　　境
CPU ボード	XG-BBEXT(BeagleBone Black)
HOST PC	PC/AT 互換機
OS	Windows Vista/7/8
メモリ	使用 OS による
ソフトウェア	ターミナルソフト
USB ポート	1 ポート
LAN ポート	10/100BASE-TX　1 ポート
SD カードスロット	microSD カードを読み込めるスロット(Ubuntu から認識できること)
microUSB ケーブル	ホスト PC と XG-BBEXT のシリアル接続用に使用
LAN ケーブル	ホスト PC と接続時はクロスケーブルを使用 ハブと接続時はストレートケーブルを使用
microSD カード	Android システムの作成用
WM-RP-04S もしくは WM-RP-05S	無線 LAN モジュールを用いた動作確認時に使用
電源	AC アダプタ（DC5V±5%）

Table 3.1-1　動作環境

上記の環境は、XG-BBEXT の Android の動作確認をするための環境となります。
Android 等のコンパイルに使用する開発環境に関しては、『**Android 開発 インストールマニュアル**』でご確認ください。

3.4　XG-BBEXT ボードの接続

ホスト PC と XG-BBEXT ボードの接続例を示します。
LAN をネットワークと接続する場合は、ネットワーク管理者と相談し、設定に注意して接続してください。

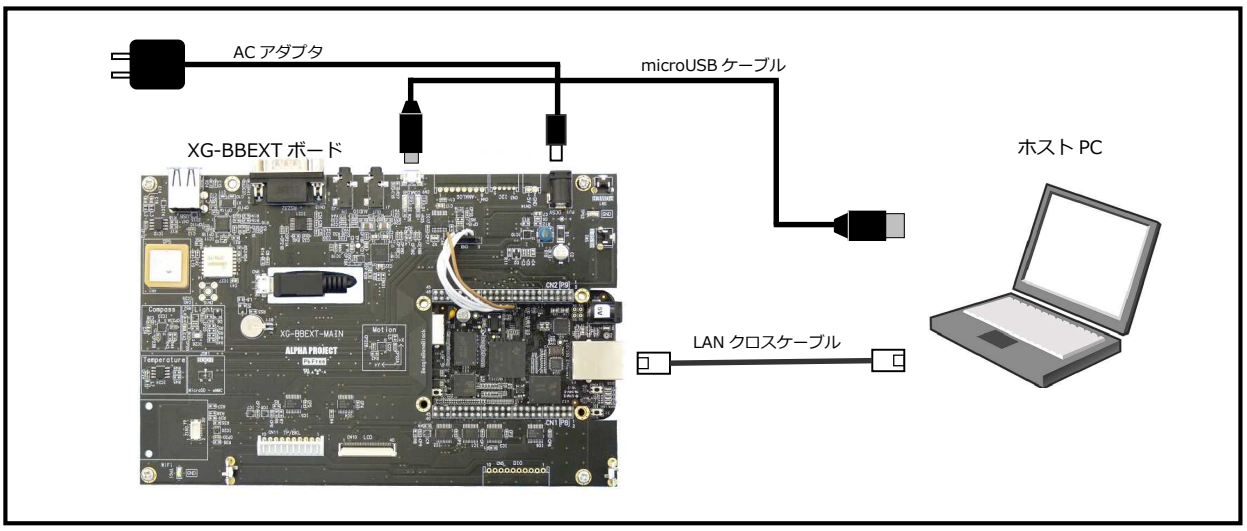


Fig 3.4-1 XG-BBEXT ボードの接続（PC に接続する場合）

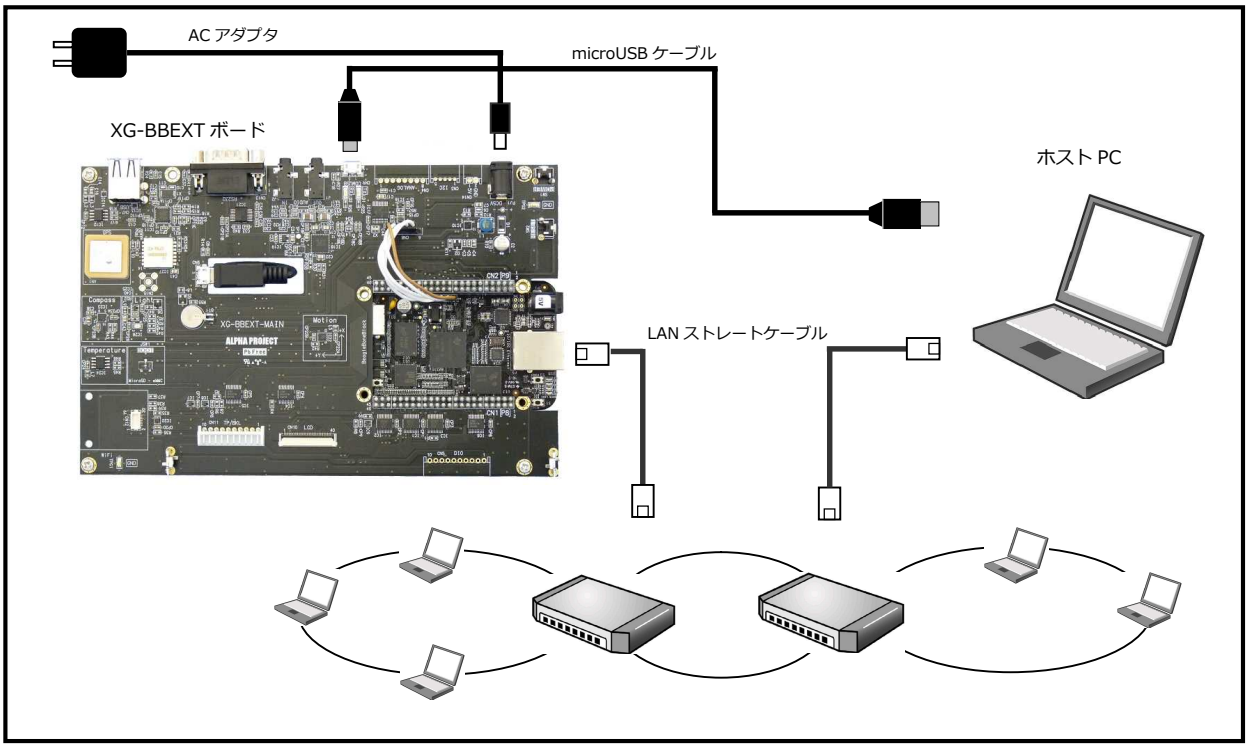


Fig 3.4-2 XG-BBEXT ボードの接続（HUB に接続する場合）

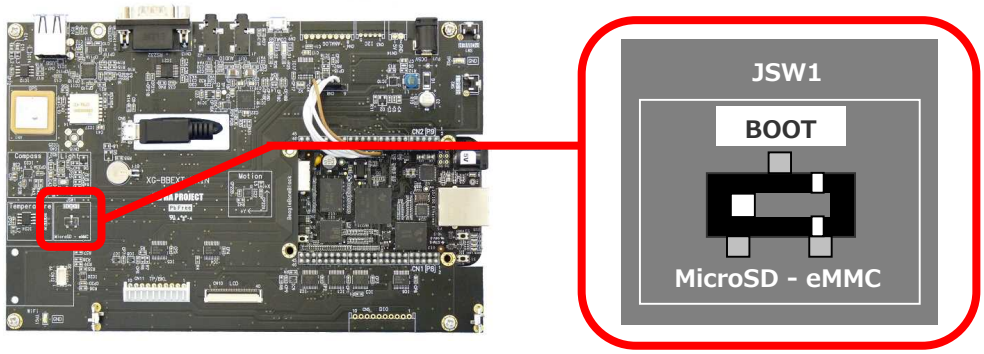
3.6 Android の起動

XG-BBEXT 上で Android の起動を行います。

Android の初回起動においては、様々な初期化処理が行われるため通常の起動に比べ遅くなります。
また、各アプリケーションの初回起動に関しても同様に遅くなります。

① Android データを入れた microSD カードを BeagleBone Black のスロットに挿入します。

② XG-BBEXT の電源をいれる前にスイッチが以下の設定になっていることを確認します。
スイッチの設定の詳細に関しては、『**XG-BBEXT ハードウェアマニュアル**』でご確認ください。



③ 『**3.4 XG-BBEXT ボードの接続**』にしたがって、ホスト PC と XG-BBEXT を接続します。
(ホスト PC と XG-BBEXT の接続は必須ではありません。必要に応じて接続してください。)

④ AC アダプタを接続して、XG-BBEXT の電源を入れます。
最初の起動ではホーム画面が出るまで数分程度かかります。

⑤ 起動が完了しますと、Android のホーム画面が表示されます。OK ボタンをタップしてください。

3.7　XG-BBEXT デバイスの Android 対応

カーネルのバージョン、デバイスドライバ、Android HAL 等により各デバイスの対応範囲があります。

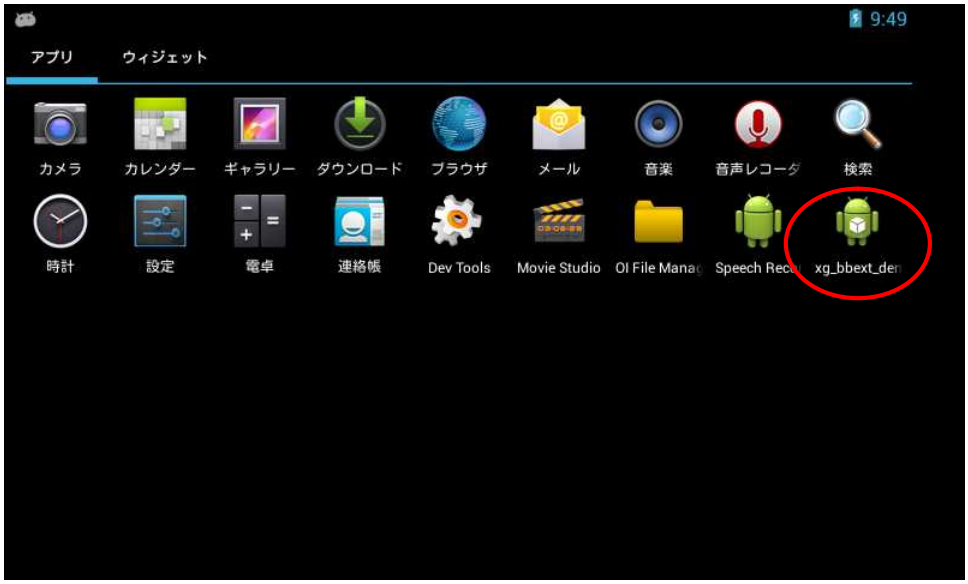
デバイスタイプ	対応範囲
GPS	緯度、経度情報のみ対応
加速度センサー	X, Y, Z 軸[m/s^2] 変化があったときのみ通知 分解能 0.459[m/s^2]
地磁気センサー	X, Y, Z 軸[μ T] 分解能 0.0917[μ T]
温度センサー	℃ 分解能 0.5[℃]
照度センサー	lx 分解能 2.44[lx]
WiFi	WiFi Direct 非対応 テザリング非対応
LCD バックライト	照度センサーとの連動は非対応

Table 3.7-1 XG-BBEXT のデバイスの Android の対応

3.8　センサーデバイスの動作確認

加速度センサー、地磁気センサー、温度センサー、照度センサー、GPS のサンプルプログラムを起動しセンサーデバイスおよび GPS の動作確認をします。

① ホーム画面からセンサーサンプルアプリ(xg_bbext_demo)をタップして起動します。



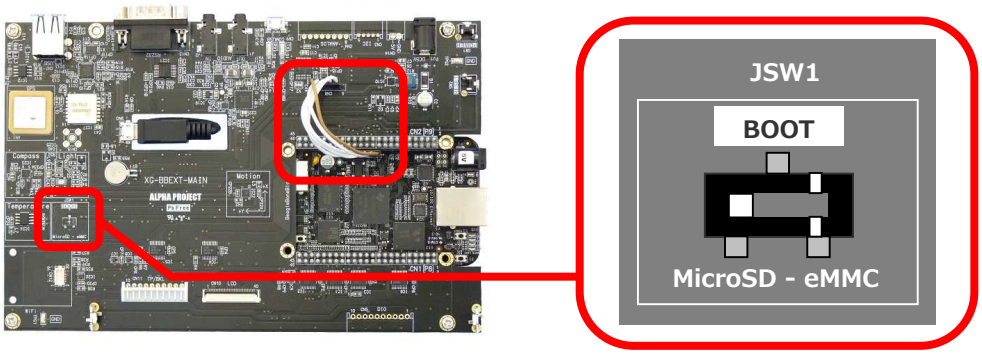
② 加速度センサー、地磁気センサーの動作確認をします。
加速度センサーはイベントが発生しないとアプリケーションに通知されません、軽く揺らしたり回転させると、加速度・地磁気の値が変化します。

4. ブートローダ

4.1 ブートローダの起動

XG-BBEXT を起動して、U-Boot のコマンドコンソールに入る方法を説明します。

① XG-BBEXT の電源を入れる前に、付属のシリアル用ハーネス及びスイッチが以下になっていることを確認します。
詳細に関しては、『**XG-BBEXT ハードウェアマニュアル**』でご確認ください。



② 『**3.5 XG-BBEXT ボードの接続**』にしたがって、ホスト PC の USB ポートと XG-BBEXT の microUSB ポート（CN9）、ホスト PC のイーサネットポートと BeagleBone Black のイーサネットポートを接続します。
ホスト PC に認識されて仮想 COM ポートが作成されます。

③ ホスト OS（Windows）のターミナルソフトを起動します。（設定は『**3.2 シリアル初期設定値**』を参照してください）

④ AC アダプタを接続して、XG-BBEXT の電源を入れます。

⑤ ターミナルに『**Hit any key to stop autoboot**』の文字が表示され、2 秒以内にキー入力を行うと U-Boot のコマンドコンソールが起動します。
コマンドコンソールが起動すると、『**U-Boot#**』が表示されます。

```
U-Boot 2013.01.01 (Jan 22 2014 - 14:10:13) ALPHAPROJECT XG-BBEXT vX.X

I2C:   ready

       :
    途中省略
       :

Net:   cpsw
Hit any key to stop autoboot:  0     入力  2秒以内にキー入力を行います
U-Boot#
```

5. Android システムの構築

5.1 XG-BBEXT 用 Android DevKit の設定

XG-BBEXT 用 Android DevKit 環境を設定します。XG-BBEXT 用 Android DevKit は TI の Android DevKit をベースにしてあり、XG-BBEXT 専用のカスタマイズが必要な部分を含むディレクトリ以外の部分については、TI Android DevKit にシンボリックリンクしてあります。

XG-BBEXT 用 Android DevKit の準備

① 作業用ディレクトリ『**xgbbext-ak**』をホームディレクトリに作成します。
すでに作成されている場合は、手順②にお進みください。

```
省略 $ mkdir ~/xgbbext-ak   入力
```

② 手順①で作成した作業用ディレクトリに移動します。

```
省略 $ cd ~/xgbbext-ak   入力
```

③ 作業用ディレクトリに付属 DVD 内の以下の 1 つのファイルをコピーします。
手順④～⑥で例として DVD から直接コピーする方法を記述します。他の方法でコピーする場合には、コピー作業完了後に、手順⑦にお進みください。

android_kit-xgbbext-X.X.tar.bz2

※『X.X』にはバージョン番号が入ります。Ver1.0 の場合は、『1.0』

④ DVD をドライブに挿入します。
デフォルトでは、自動でマウントされますが、マウントされない場合は、以下のコマンドを実行します。

```
省略 $ gvfs-mount -d /dev/sr0   入力
```

マウントされているかどうかは、『**mount**』コマンドで確認できます。
以下のように、『**/dev/sr0**』が表示されている場合は、すでにマウントされています。
（『*******』は、DVD のボリュームラベルになります。）

```
省略 $ mount   入力
          :
      途中省略
          :
/dev/sr0 on /media/******* type udf (ro,nosuid,nodev,uhelper=udisks,uid=1000,
gid=1000,iocharset=utf8,umask=0077)
```

⑤ 1 つのファイルをコピーします。コマンド途中の『*******』は、DVD のボリュームラベルになります。
そのため、その部分は挿入した DVD に合わせて入力してください。

```
省略 $ cp /media/*******/sources/android_kit-xgbbext-X.X.tar.bz2 .   入力
```

⑥ DVD をアンマウントします。

```
省略 $ umount /dev/sr0    入力
```

⑦ Android DevKit を展開します。時間がかかりますので、しばらくお待ちください。

```
省略 $ tar -xjpf android_kit-xgbbext-X.X.tar.bz2    入力
```

XG-BBEXT Android DevKit のディレクトリ構成は次のようになっています。

```
xgbbext-ak
`-- android_kit-xgbbext-X.X
     |-- Makefile
     |-- abi -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/abi
     |-- bionic -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/bionic
     |-- bootable -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/bootable
     |-- build
     |-- cts -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/cts
     |-- dalvik -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/dalvik
     |-- development -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/development
     |-- device
     |-- external
     |-- frameworks
     |-- gdk -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/gdk
     |-- hardware
     |-- image_folder
     |-- kernel
     |-- libcore -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/libcore
     |-- libnativehelper -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/libnativehelper
     |-- ndk -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/ndk
     |-- packages -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/packages
     |-- prebuilts -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/prebuilts
     |-- sdk -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/sdk
     |-- system -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/system
     `-- u-boot -> ../../TI_Android_JB_4.2.2_DevKit_4.1.1/u-boot
```

Table 5.1-1　Android DevKit ディレクトリ構成

5.2 U-Boot の作成

XG-BBEXT Android DevKit 上で、ブートローダを作成するための手順を説明します。

U-Boot の作成

① 準備作業で展開した Android DevKit の『**u-boot**』へ移動します。

```
省略 $ cd ~/xgbbext-ak/android_kit-xgbbext-X.X/u-boot    入力
```

② Linux カーネルの設定データを呼び出します。

```
省略 $ make CROSS_COMPILE=arm-eabi- am335x_evm_config    入力
Configuring for am335x_evm - Board: am335x_evm, Options: SERIAL1,CONS_INDEX=1
```

③ make を実行します。終了までに数分かかる場合があります。

```
省略 $ make CROSS_COMPILE=arm-eabi-    入力
Generating include/autoconf.mk
Generating include/autoconf.mk.dep
arm-eabi-gcc -DDO_DEPS_ONLY ¥

        :
    途中省略
        :

make[1]: ディレクトリ `/home/guest/TI_Android_JB_4.2.2_DevKit_4.1.1/u-boot/examples/api'
に入ります
make[1]: `all' に対して行うべき事はありません.
make[1]: ディレクトリ `/home/guest/TI_Android_JB_4.2.2_DevKit_4.1.1/u-boot/examples/api'
から出ます
```

④ make が正常に終了すると U-Boot イメージ『**u-boot.img**』が作成されます。

```
省略 $ ls u-boot.img    入力
u-boot.img
```

5.3 起動用環境設定ファイル(uEnv.txt)

XG-BBEXT(BeagleBone Black)では、U-Boot にて setenv コマンドで環境変数を永続的に保存することはできません。
『**4.2 ネットワークの設定**』にて U-Boot の IP 関連設定をソースファイルにて設定する方法について解説してありますが、Linux の起動に関する環境変数を変更したい場合、毎回 U-Boot をソースの修正、ビルド、microSD へ書き込みをするのは大変です。
そのため起動に関する環境変数の設定はブートパーティション（/dev/mmcblk0p1)にある uEnv.txt というファイルにて設定します。

『uEnv.txt』の設定例としては、以下のように記述します。

```
bootargs=console=ttyO0,115200n8 androidboot.console=ttyO0 mem=512M root=/dev/mmcblk0
p2 rw rootfstype=ext4 rootwait init=/init ip=off
bootcmd=mmc rescan; fatload mmc 0 81000000 uImage; bootm 81000000
uenvcmd=boot
```

上記内容は、bootargs、bootcmd、uenvcmd の 3 行の構成となります。
bootargs に関しては、表示上 2 行となっておりますが、1 行目から 2 行目にかけてスペース及び改行が入らない形となります。

各環境変数は以下の通りです。

環境変数名	設定内容
bootargs	コンソール出力を/dev/ttyO0 115200bps、ノーパリティ、8bit データに設定します。 メモリは 512Mbyte に設定します。 Android コンソールも/dev/ttyO0 に設定します。 ルートファイルシステムは/dev/mmcblk0p2 で読書き可能で、ファイルシステムタイプは ext4、ファイルシステムが起動するまで wait するように設定します。
bootcmd	カーネルをロードし、ロードしたアドレスから起動するように設定します。 テスト的に別カーネルを起動する場合は uImage の代わりに別カーネルのファイル名を直接指定します。
uenvcmd	uenvcmd が設定されていますと uExt.txt を解析後、そのままブートが実行されます。

Table 5.3-1 uEnv.txt の環境変数

uEnv.txt はテキストファイルですので、ゲスト OS(Ubuntu)上で microSD カード内のファイルをエディタで編集します。

5.4 Linux カーネルの作成

XG-BBEXT Android DevKit 上で、Linux カーネルを作成するための手順を説明します。

Linux カーネルの作成

Linux カーネルをコンパイルする方法を説明します。
Linux カーネルの設定データは Linux カーネルソースディレクトリ以下『**arch/arm/configs/xg_bbext_defconfig**』に保存されています。

① 準備作業で展開した作業用ディレクトリの『**kernel**』へ移動します。

```
省略 $ cd ~/xgbbext-ak/android_kit-xgbbext-X.X/kernel      入力
```

② Linux カーネルの設定データを呼び出します。

```
省略 $ make ARCH=arm CROSS_COMPILE=arm-eabi- xg_bbext_defconfig      入力
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c

        :
    途中省略
        :

#
# configuration written to .config
#
```

③ make を実行します。終了までに数分から数時間かかる場合があります。

```
省略 $ make ARCH=arm CROSS_COMPILE=arm-eabi- uImage      入力
scripts/kconfig/conf --silentoldconfig Kconfig
  WRAP    arch/arm/include/generated/asm/auxvec.h
  WRAP    arch/arm/include/generated/asm/bitsperlong.h

        :
    途中省略
        :

Load Address: 80008000
Entry Point:  80008000
  Image arch/arm/boot/uImage is ready
```

5.5 Android ファイルシステムの作成

ゲスト OS(Ubuntu)上で Android ルートファイルシステムの作成するための手順を説明します。

ルートファイルシステム作成

① Android ファイルシステムのディレクトリ『**android_kit-xgbbext-X.X**』に移動します。

```
省略 $ cd ~/xgbbext-ak/android_kit-xgbbext-X.X  入力
```

② Android ファイルシステムのビルドをします。
ビルドが終了するまでには PC の性能により数時間掛かります。

```
省略 $ make TARGET_PRODUCT=xg_bbext OMAPES=4.x  入力
```

-jn オプションを追加するとビルドが早くなります。 n はビルドする PC のプロセッサ数によって決めます。
通常は、プロセッサ数× 2 が推奨されていますが VirtualBox の Ubuntu でビルドするときは仮想空間に割り当てる CPU 数を考慮してください。
また、たまにビルドエラーが発生することがありますが、その場合は再度ビルドを継続してみてください。
(メモリ不足またはビルド順序などの問題により再度ビルドすると成功することがあります。)

③ 『**rootfs.tar.bz2**』を作成するために、以下のコマンドを実行します。

```
省略 $ make TARGET_PRODUCT=xg_bbext fs_tarball  入力
```

④ 正常に終了すると、『**out/target/product/xg_bbext**』ディレクトリにファイルが作成されます。

```
省略 $ ls out/target/product/xg_bbext/rootfs.tar.bz2  入力
rootfs.tar.bz2
```

5.6 microSD の作成

作成した U-Boot、Linux カーネル、Android ファイルシステムのファイルから microSD カードの作成方法を説明します。

microSD の作成の準備

① 作業用ディレクトリ『**xgbbext-ak**』をホームディレクトリに移動します。
すでに作成されている場合は、手順②にお進みください。

```
省略 $ mkdir ~/xgbbext-ak/android_kit-xgbbext-X.X    入力
```

② microSD に書込むファイルを image_folder にコピーします。

```
省略 $ cp kernel/arch/arm/boot/uImage image_folder    入力
省略 $ cp u-boot/u-boot.img image_folder    入力
省略 $ cp u-boot/MLO image_folder    入力
省略 $ cp out/target/product/xg_bbext/rootfs.tar.bz2 image_folder    入力
```

microSD の作成

① image_folder に移動します。

```
省略 $ cd image_folder    入力
```

② SD カードを PC の SD カードスロットに挿入し、認識されたら以下のコマンドで書き込みます。

```
省略 $ sudo ./mkmmc-android.sh /dev/sdb MLO u-boot.img uImage uEnv.txt rootfs.tar.bz2    入力
[sudo] password for guest:    入力
```

Android の初回起動においては、様々な初期化処理が行われるため通常の起動に比べ遅くなります。
また、各アプリケーションの初回起動に関しても同様に遅くなります。

6. アプリケーションプログラムの開発

本章では、XG-BBEXT 上で動作するアプリケーションの作成方法について概略を説明します。組込み Android システムの開発においては、Dalvik 上で動く Java 使った Android アプリケーション開発以外に、JNI を介して動作する C,C++などで開発した ARM バイナリレベルのプログラム、ライブラリ開発も必要となることがあります。また、センサー等をサポートするには HAL にてセンサーをサポートするプログラミングが必要です。さらに特殊なデバイスを扱うにはカーネルレベルのデバイスドライバの開発も必要となります。

6.1 Android アプリケーションの開発

Dalvik 上で動く Android アプリケーションは通常 Java で作成します。コマンドレベルで開発することも可能ですが、GUI を使った統合開発環境を使用する場合は、Android 開発用の Eclipse などを別途インストールする必要があります。アプリケーション開発のための開発環境設定は Android 開発者サイトおよび Android の書籍などを参考にしてください。

Android_kit には XG-BBEXT に搭載されているセンサーの動作確認用のサンプルプログラムが用意されています。サンプルアプリケーションを個別にビルドするには別途 Eclipse にて取り込みビルドしてください。

6.2 カーネルレベルのドライバ開発

XG-BBEXT ボードに I2C で外部にデバイスを接続した場合などはデバイスドライバが必要となることがあります。
標準的なデバイスであればカーネルのカスタマイズにてドライバを指定することができますが、サポートされていない場合はユーザが独自にデバイスドライバを用意する必要があります。

デバイスドライバの開発については、XG-BBEXT Linux ソフトウェアマニュアルを参照してください。

6.3 HAL(Hardware Application Layer)ライブラリ開発

オーディオ、GPS、センサー、バックライト制御、各種センサーなどの HAL のライブラリは、
『~/xgbbext-ak/android_kit-xgbbext-X.X/device/ti/xg_bbext』に置かれています。
XG-BBEXT 用のライブラリやアプリケーションはこのフォルダに置かれます。センサーサンプルプログラムもここに置かれています。

I2C で独自のセンサーを追加した場合は、デバイスドライバに加えてセンサーライブラリを用意する必要があります。センサーのタイプによってはユーザが独自にライブラリを用意する必要があります。

センサライブラリをはじめとする、HAL ライブラリは主に C++/C にて開発することになります。センサー関連は『libsensors』に置かれており Android.mk にて『sensors.xg_bbext』というモジュール名で作成されます。また親ディレクトリの device.mk にて『PRODUCT_PACKAGES += lights.xg_bbext sensors.xg_bbext　gps.xg_bbext』とインストールするパッケージが指定されています。

6.4 センサー サンプルプログラム

センサーサンプルプログラムは初期状態でインストール済みになるよう設定されていますので、センサーの動作確認は Android が起動すればそのまま実行することができます。また DVD-ROM の android/sample ディレクトリにもプロジェクトが登録されています。

ここではセンサーを用いたプログラムを作成する場合のポイントを説明をします。

XG-BBEXT でサポートされているセンサーについて

XG-BBEXT でサポートされるセンサーは以下の種類があります。

種類	センサタイプ	センサデバイス
加速度センサー	Sensor.TYPE_ACCELEROMETER	MMA7660FC
地磁気センサー	Sensor.TYPE_MAGNETIC_FIELD	HMC5883L
照度センサー	Sensor.TYPE_LIGHT	BH1603FVC
温度センサー	Sensor.TYPE_AMBIENT_TEMPERATURE	LM75A

Table 8.4-1 XG-BBEXT センサー

個々の Android デバイスによって利用できるセンサーは様々です。Android ではセンサーを使用するにあたって、センサーマネージャを取得し、そこから利用できるセンサーのリストを取得します。利用できるセンサーは複数の場合もあるため、リストとして返されます。

センサーマネージャの取得

センサーマネージャの取得は以下のようにします。

```
SensorManager sensorManager
              = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

センサーのリストの取得

取得したセンサーマネージャから、センサリストを得るには、引数としてセンサータイプを渡して getSensorList メソッドを呼びます。

```
List<Sensor> list
           = sensorManager.getSensorList(センサータイプ);
```

複数のセンサータイプを指定するときは「|」を使って列挙します。すべてのセンサーを指定する場合は、Sensor.TYPE_ALL を指定します。

リスナーの定義、センサーからの値の取得

センサーからの値はリスナーの onSensorChanged(SensorEvent event)にて通知されます。
リスナーは次のように、そのままリスナーのインスタンスを作成して使用することが出来ます。

```
private SensorEventListener listener = new SensorEventListener()
{
    @Override
    public void onAccuracyChanged( Sensor sensor, int accuracy )
    {
    }

    @Override
    public void onSensorChanged( SensorEvent event )
    {
        // event.values に値が入っている
    }
};
```

もう一つの方法は、インターフェースとして SensorEventListener を使用し、onSensorchanged, onAccuracyChanged を実装（implement)する方法があります。この場合、前項目のリスナーの登録時のリスナーには「this」を指定します。

```
public class SensorEx extends Activity implements SensorEventListener
{
～　省略　～
    public void onAccuracyChanged( Sensor sensor, int accuracy )
    {
    }

    public void onSensorChanged( SensorEvent event )
    {
        // event.values に値が入っている
    }

～　省略 ～
}
```

インターフェースで SensorEventListener は実装されているので、@Override はつけません。

7. 無線 LAN モジュールの使用

本章では、XG-BBEXT に WM-RP-04S もしくは WM-RP-05S を接続して動作を行う方法を説明します。

7.1 Linux カーネルの対応方法

Linux カーネルのデフォルトでは、WM-RP-04S もしくは WM-RP-05S を使用する設定になっておりませんので、Linux カーネルを再作成する必要があります。
再作成する手順を以下に説明します。

本手順では、『**5.4 Linux カーネルの作成**』によって一度 Linux カーネルが作成されていることを前提で説明します。一度も行っていない場合は、一度作成手順を行ってください。

また、WM-RP-04S もしくは WM-RP-05S を使用するには、以下の 4 つのファイル(ファームウェア)が別途必要となります。

sbinst1, sbinst2, sbdata1, sbdata2

ファイルの入手方法に関しては、WM-RP シリーズの『**ハードウェアマニュアル**』にコンテンツのダウンロード方法が記載されておりますので、そちらでご確認ください。
なお、以下の手順では、~/wm-rp-firm にダウンロードされていることを前提で説明します。

① 『**5.4 Linux カーネルの作成**』で作成した Linux カーネルのフォルダに移動します。

```
省略 $ cd ~/xgbbext-ak/android_kit-xgbbext-X.X/kernel    入力
```

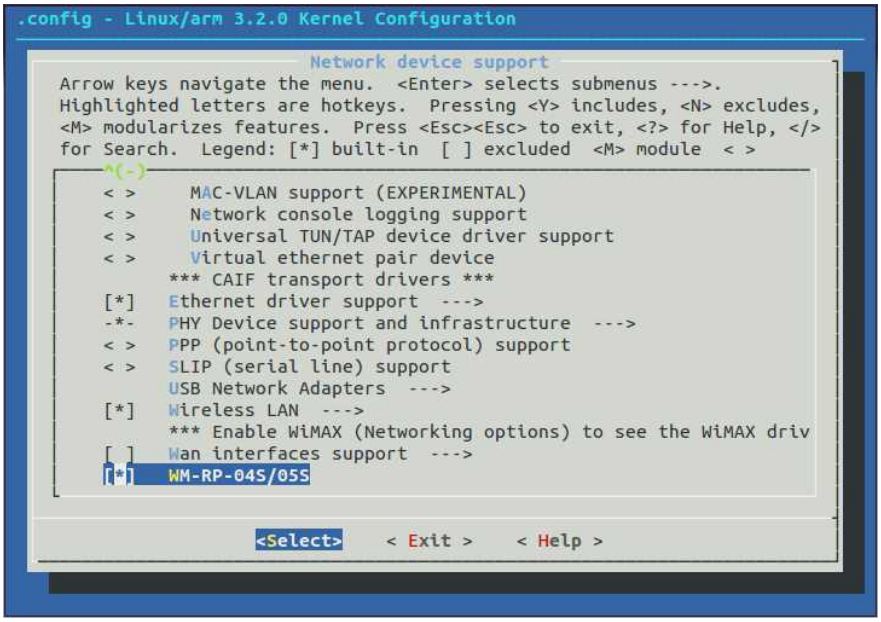
② 無線 LAN モジュールのファームウェアをコピーします。

```
省略 $ cp ~/wm-rp-firm/sbinst1 ./kernel/drivers/net/rs21/Firmware    入力
省略 $ cp ~/wm-rp-firm/sbinst2 ./kernel/drivers/net/rs21/Firmware    入力
省略 $ cp ~/wm-rp-firm/sbdata1 ./kernel/drivers/net/rs21/Firmware    入力
省略 $ cp ~/wm-rp-firm/sbdata2 ./kernel/drivers/net/rs21/Firmware    入力
```

③ 『**make ARCH=arm menuconfig**』コマンドによって、コンフィグレーションメニューを表示します。

```
省略 $ make ARCH=arm CROSS_COMPILE=arm-eabi- menuconfig    入力
scripts/kconfig/mconf Kconfig
```

④ メニューの『**Device Drivers**』－『**Network device support**』と選択して、以下の画面の『**WM-RP-04S/05S**』を選択します。



⑤ 画面下にある『<Exit>』を何度か選択して、設定を保存します。

⑥ make を実行します。終了までに数分から数時間かかる場合があります。

```
省略 $ make ARCH=arm CROSS_COMPILE=arm-eabi- uImage    入力
scripts/kconfig/conf --silentoldconfig Kconfig
  WRAP    arch/arm/include/generated/asm/auxvec.h
  WRAP    arch/arm/include/generated/asm/bitsperlong.h

        :
    途中省略
        :

Load Address: 80008000
Entry Point:  80008000
  Image arch/arm/boot/uImage is ready
```

⑦ make が正常に終了すると『**./arch/arm/boot**』ディレクトリに Linux カーネルイメージ『**uImage**』が作成されます。

```
省略 $ ls arch/arm/boot/uImage    入力
arch/arm/boot/uImage
```

⑧ Android microSD カードを PC に挿入します。
自動認識されたら次のステップに進みます。

⑨ microSD ディレクトリのブートパーティションを確認します。

```
省略 $ ls /media/boot    入力
MLO  u-boot.img  uEnv.txt  uImage
```

7.2 動作確認

『**7.1 Linux カーネルの対応方法**』で作成したカーネルで起動した XG-BBEXT 上で、無線 LAN モジュールを動作させる手順を説明します。
なお、ここでは例として以下の設定のアクセスポイントがあることを前提とした手順となります。もし、異なる環境で行う場合には、適宜読み替えて作業を進めてください。

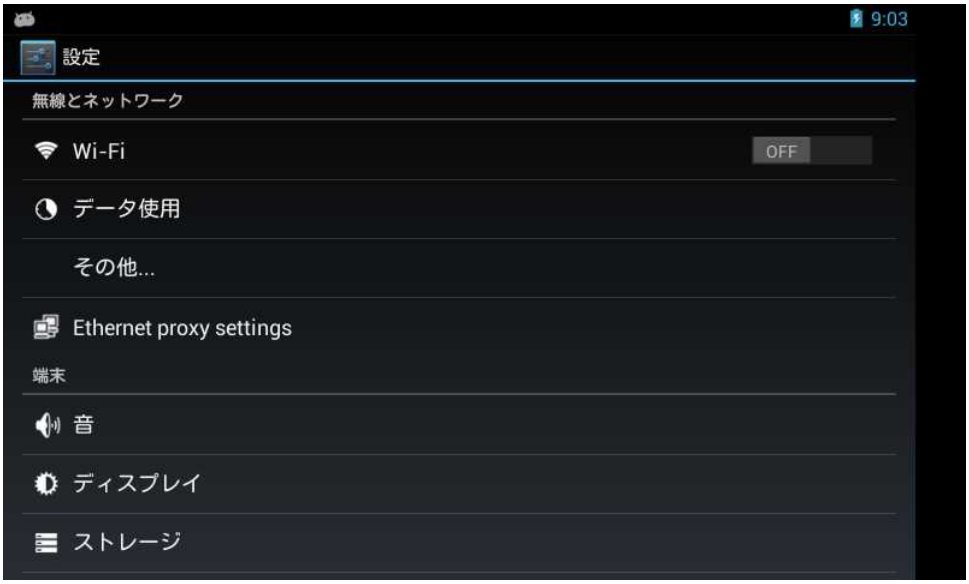
設定項目	設定値
SSID	WIFI-TEST
パスコード	wifi-pass
セキュリティモード	WPA2

Table 7.2-1 アクセスポイントの設定例

① XG-BBEXT ボードに WM-RP-04S もしくは WM-RP-05S を接続します。
接続方法に関しては、XG-BBEXT の『**ハードウェアマニュアル**』でご確認ください。

② XG-BBEXT の電源を入れて Android を起動します。

③ Android が起動したら、ホームから設定を起動します。
設定メニューが表示されましたら、『**Wi-Fi**』の項目がありますので選択します。



謝辞

Android、Linux、U-Boot の開発に関わった多くの貢献者に深い敬意と感謝の意を示します。

著作権について

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