XG Series

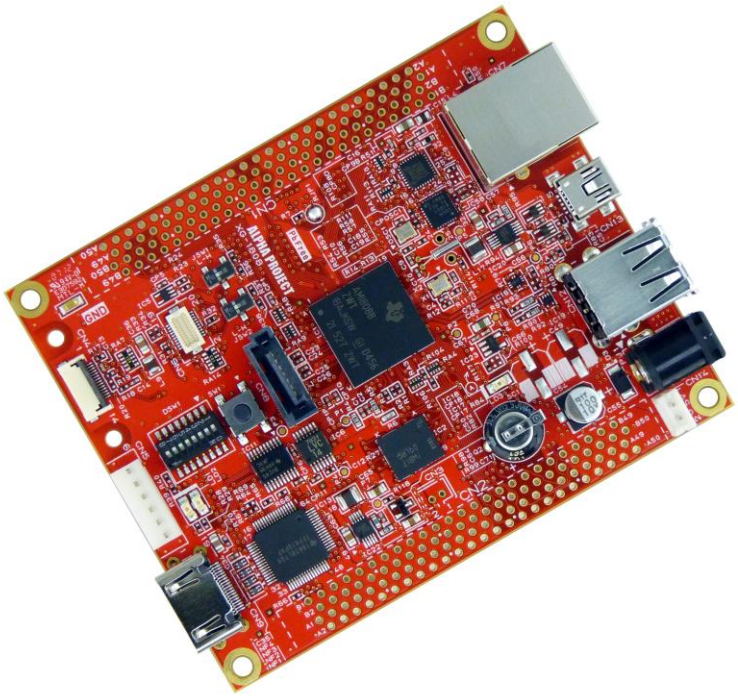
LK-1808-A01

ARM9 AM1808 CPU BOARD

Software Manual

Rev 1.7

ダイジェスト版



ALPHAPROJECT

https://www.apnet.co.jp

目　次

2. システム概要

2.1 システム概要

XG-1808 は、TEXAS INSTRUMENTS ARM9 マイクロプロセッサ「AM1808」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、ルートファイルシステムから構成されます。ブートローダに UBL と U-Boot、Linux カーネルに Linux-3.1、ルートファイルシステムには RAM か microSD カード等で動作する専用パッケージを使用します。

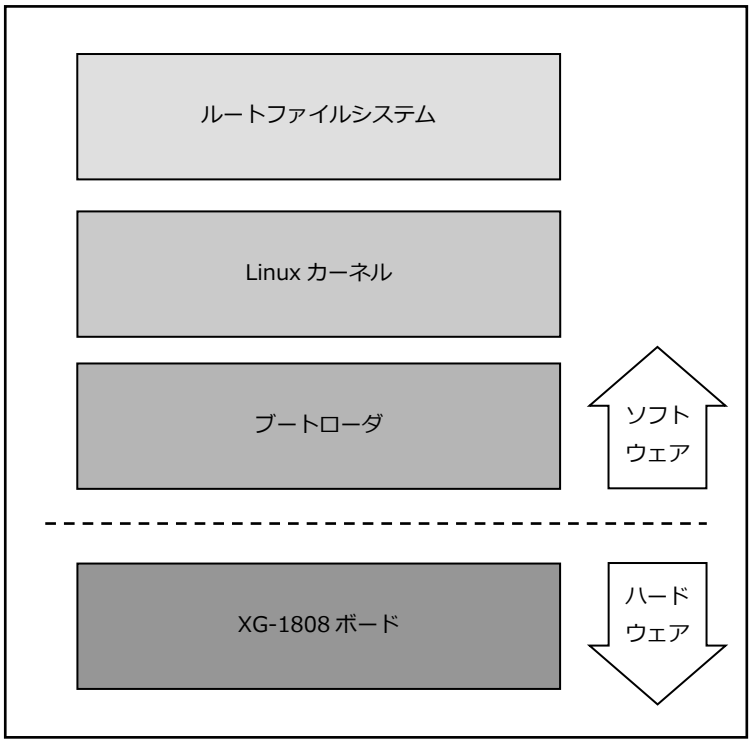


Fig 2.1-1 XG-1808 システム概要図

2.2 ブートローダ

Linux カーネルは RAM 上で動作しますが、カーネル自体は RAM 上に自身をロードする機能を有していません。そのため、Linux カーネルをロードする何らかの手段が必要となります。この手段を提供するのがブートローダです。ブートローダは CPU やメモリ、周辺ハードウェアの初期化を行い、カーネルを RAM 上に展開したあとにカーネルをブートさせます。
XG-1808 のブートローダには、U-Boot を使用します。

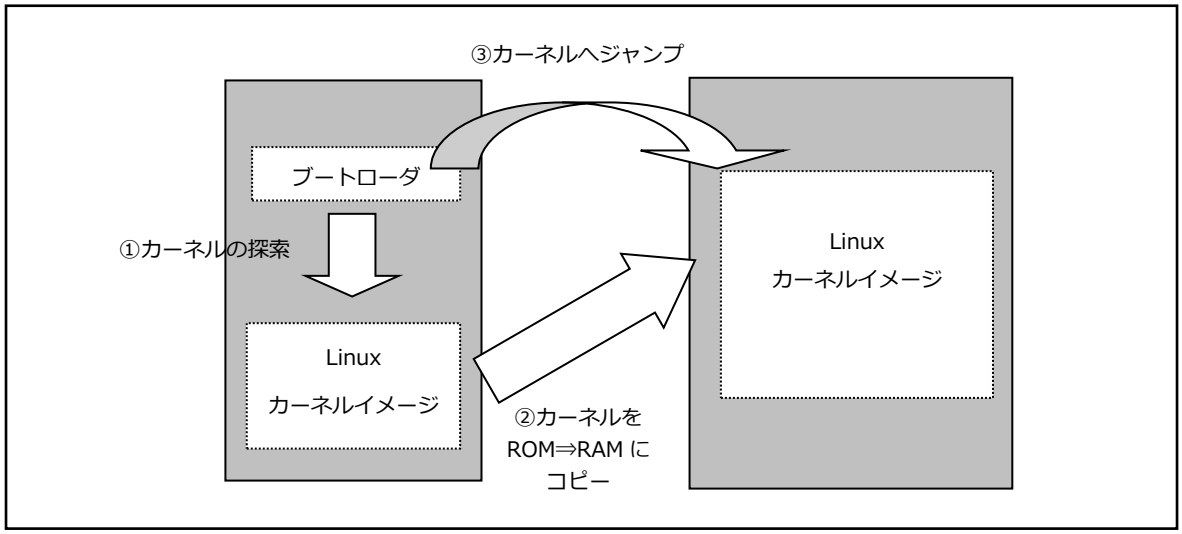


Fig 2.2-1 ブートローダ動作イメージ

2.3 Linux カーネル

Linux カーネルにはプロセス管理、メモリ管理、各種ファイルシステム、ネットワーク機能などがあり、デバイスドライバ自身もカーネルに組み込まれます。Linux カーネル上ではシェル（コマンドプロンプト）や Web サーバなどの多くのアプリケーションが動作します。
本製品では Linux カーネル 3.1 を使用しています。本製品に添付される Linux カーネルは TCP/IP によるネットワーク機能、各種ファイルシステム(ext2、ext3、NFS、FAT 等)をサポートしています。

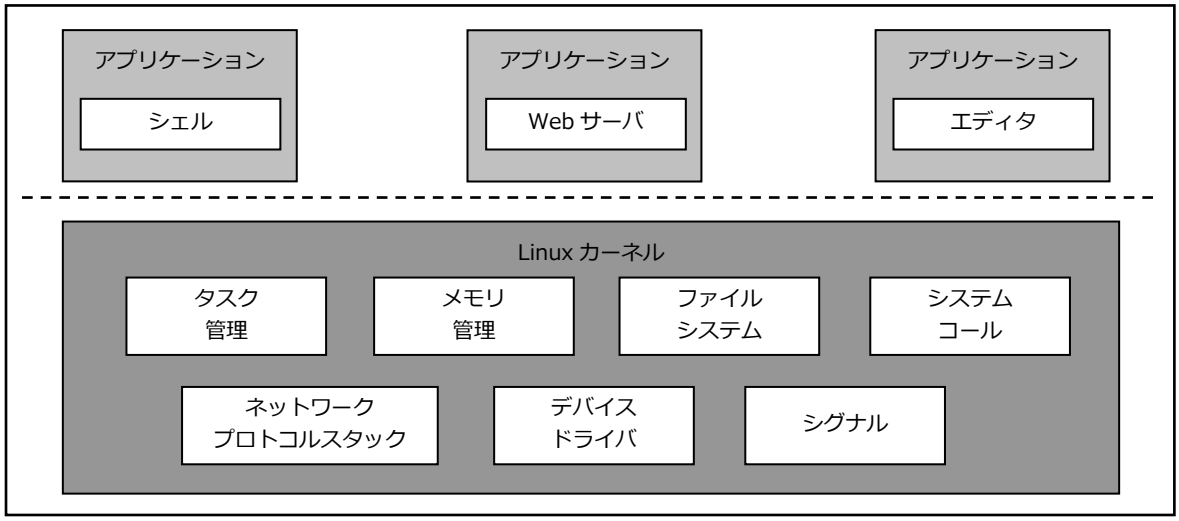


Fig 2.3-1 Linux システム概要

2.4 ルートファイルシステム

Linux は、カーネルとファイルシステムという 2 つの要素から構成されます。
Linuxでは､全てのデータがファイルという形で管理されています｡アプリケーションプログラムやデバイスドライバをはじめ、HDD や COM ポートなどの入出力デバイスもファイルとして扱われます。
Linux では全てのファイルがルートディレクトリを起点としたディレクトリ構造下に管理されており、これら全てのファイル構造のことをファイルシステムと呼びます。また、システム動作に必要なシステムファイル群のこともファイルシステムと呼びます。
本ドキュメントでは、これらの意味を明確にするため、ファイル管理構造(ext2 や ext3)のことをファイルシステム、システム動作に必要なファイル群のことをルートファイルシステムと表現しています。

Linux のルートファイルシステムは、そのシステムが必要とする機能に合わせて構築する必要があります。
XG-1808 では、以下のルートファイルシステムを用意しています。

- ●ramfs ルートファイルシステム　RAM 上で動作するように構成されたオリジナル Linux パッケージです。RAM 上に展開されるため、電源を落とすと変更した内容は破棄されます。
- ●sd ルートファイルシステム　SD カード用に構成されたオリジナル Linux パッケージです。ルートファイルシステムが SD カード上に展開されるため、電源を落としても変更した内容は破棄されませんが、電源を落とす前には適切な終了処理が必要になります。
- ●norflash ルートファイルシステム　NOR FlashROM 用に構成されたオリジナル Linux パッケージです。ルートファイルシステムが NOR FlashROM 上に展開されるため、電源を落としても変更した内容は破棄されませんが、電源を落とす前に適切な終了処理が必要になります。

本ドキュメントでは、ramfs ルートファイルシステムを利用した Linux システムを RAMFS-Linux システム、sd ルートファイルシステムを利用した Linux システムを SD-Linux システム、norflash ルートファイルシステムを利用した Linux システムを NORFLASH-Linux システムと表現します。

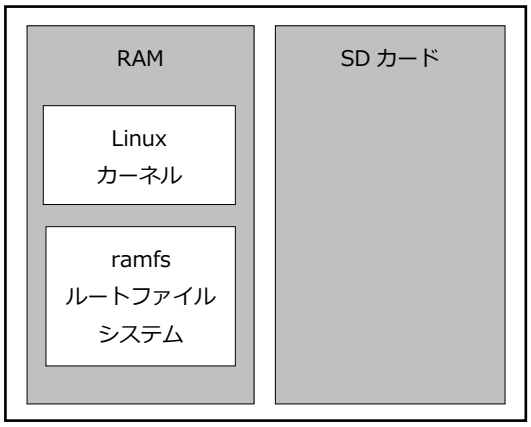


Fig 2.4-1 RAMFS-Linux システム

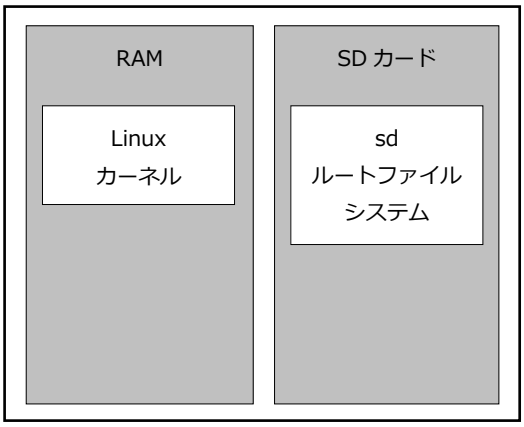


Fig 2.4-2 SD-Linux システム

2.5　クロス開発環境

XG-1808 上で動作する Linux カーネルやアプリケーションプログラムを作成するには、Linux の動作する PC/AT 互換機上でクロス開発環境を構築する必要があります。クロス開発環境を構築するには、LinuxOS 上にターゲット用の下記のパッケージをインストールする必要があります。

GNU binary utilities（アセンブラ、リンカ等）
GNU Compiler Collection（クロスコンパイラ・プリプロセッサ等）
uClibc（C 標準ライブラリ等）

上記のパッケージによりターゲット用の実行ファイルを作成することができます。実行ファイルは LinuxOS からターゲットシステムにダウンロードし、動作を確認します。

本製品では、LinuxOS として VMware Player 上で動作する『Ubuntu』を使用します。
『uClibc』は組込み用途向け C 標準ライブラリです。通常 Linux システムで使用される『Glibc』よりも容量を必要としないためメモリサイズに制限がある場合などに使用されます。

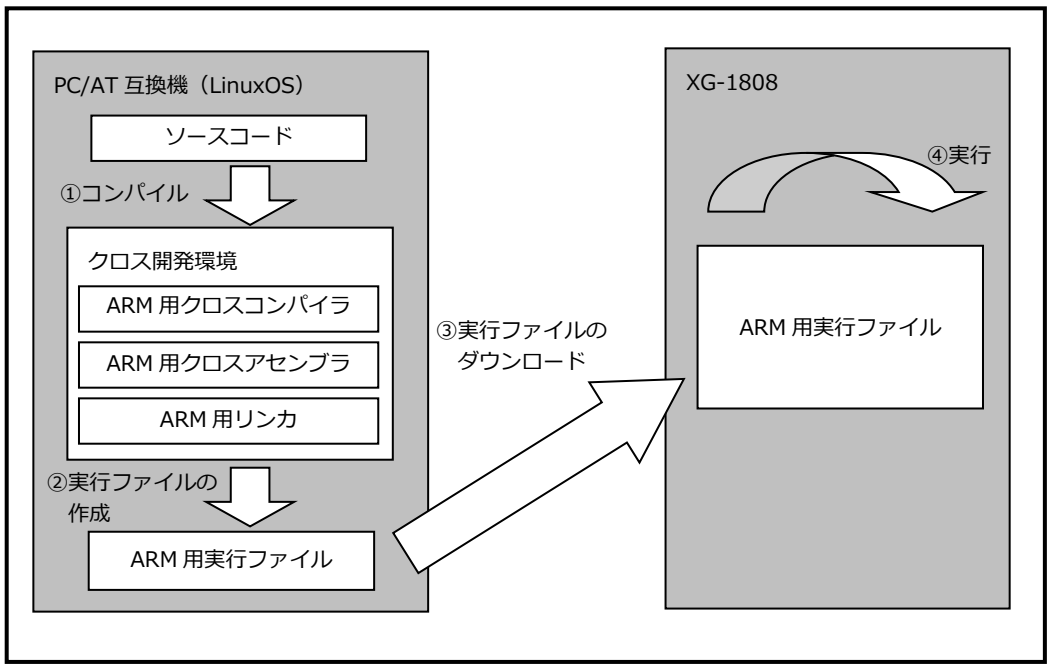


Fig 2.5-1 クロス開発環境

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。

本 Linux 開発キットには、PC-USB-03 が付属しており、PC-USB-03 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。

PC-USB-03 の使用方法に関しては、PC-USB-03 のマニュアルをご参照ください。

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-1808
HOST PC	PC/AT 互換機
OS	WindowsXP/Vista/7
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 カードが読み込めるスロット(Ubuntu から認識できること)
PC-USB-03	ホスト PC と XG-1808 のシリアル接続用に使用
USB ケーブル	PC-USB-03 で使用
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
HDMI-DVI ケーブル	DirectFB のサンプル動作を行う場合に必要
DVI ディスプレイ (800x480 が表示可能なもの)	DirectFB のサンプル動作を行う場合に必要
SATA ハードディスク	SATA の動作確認を行う場合に必要
LCD-KIT-B01/B02 もしくは LCD-KIT-C01/C02 もしくは LCD-KIT-D01/D02	タッチパネル LCD キットを用いた動作確認を行う場合に必要
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を行う場合に必要
電源	AC アダプタ（DC5V±5%） 単体動作時には 1A 以上、タッチパネル LCD キットを使用する場合は、2A 以上必要

Table 3.1-1　動作環境

上記の環境は、XG-1808 の Linux の動作確認をするための環境となります。
カーネル等のコンパイルに使用する開発環境に関しては、『**Linux 開発 インストールマニュアル**』でご確認ください。

3.4　XG-1808 ボードの接続

ホスト PC と XG-1808 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。

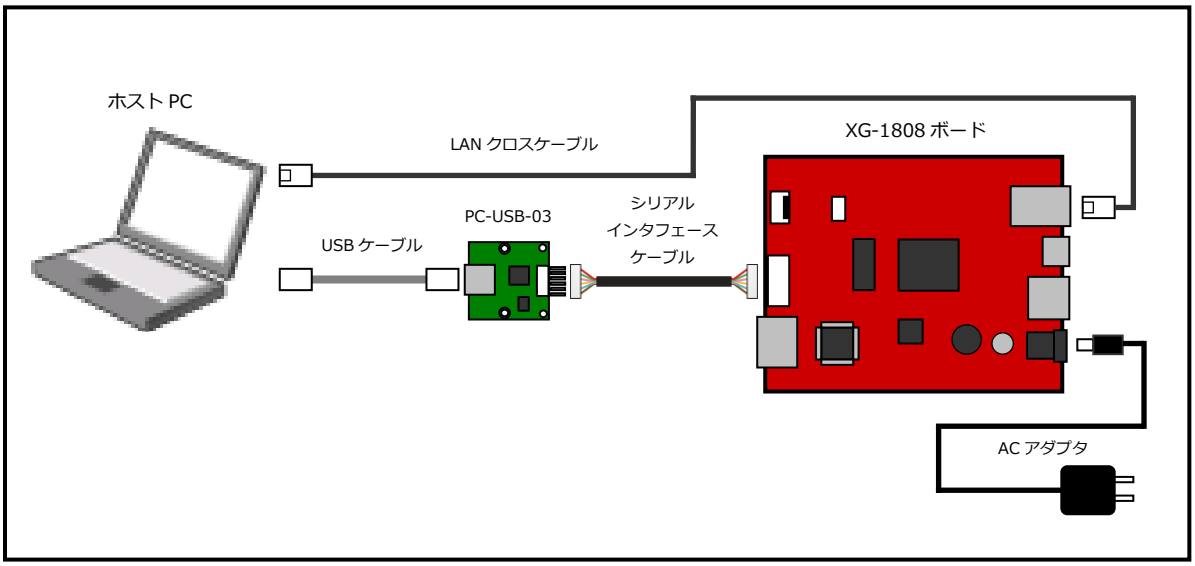


Fig 3.4-1 XG-1808 ボードの接続（PC に接続する場合）

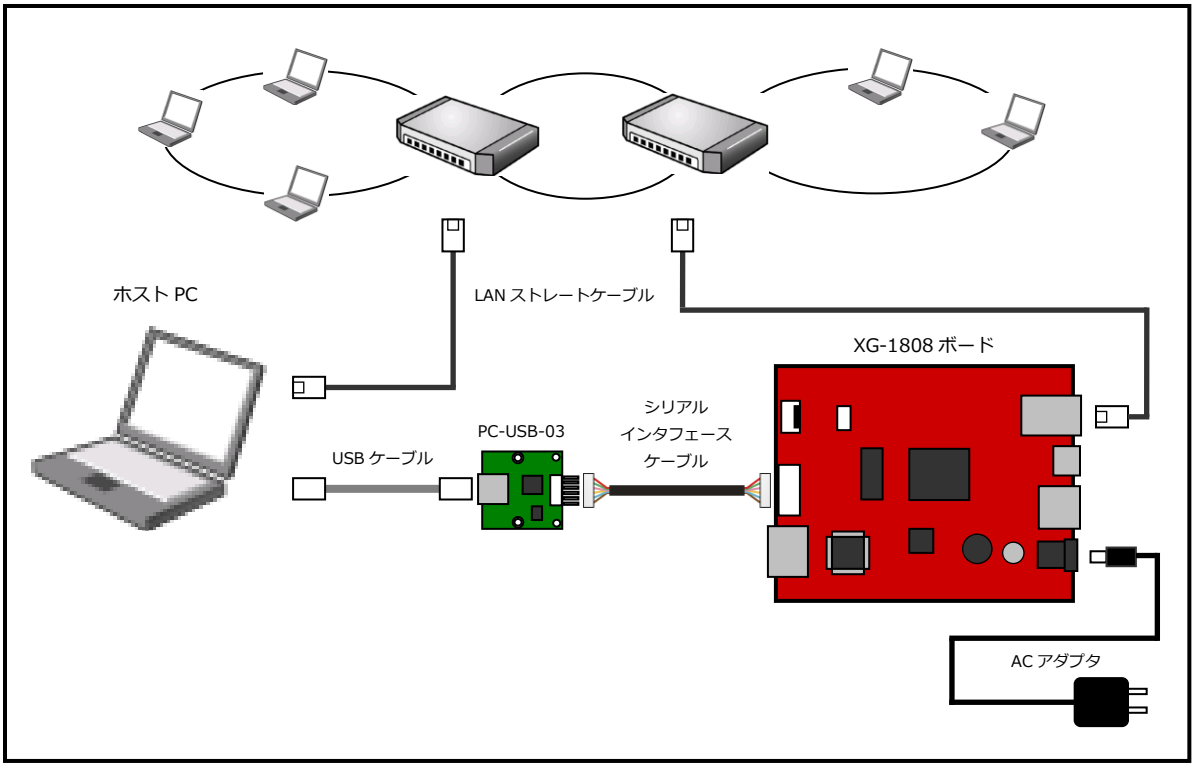


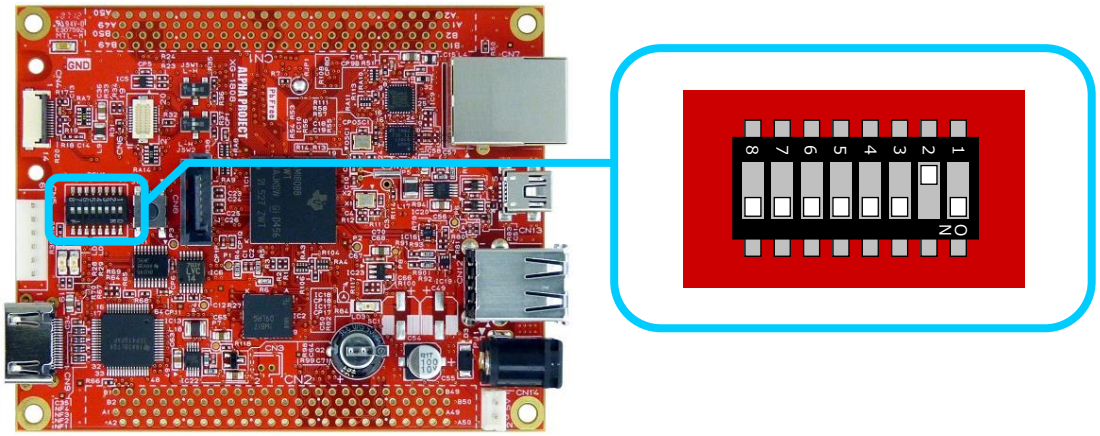
Fig 3.4-2 XG-1808 ボードの接続（HUB に接続する場合）

3.5 Linux の起動

XG-1808 上で Linux の起動を行います。
XG-1808 は出荷時状態で Linux が自動起動します。

① 『**3.4 XG-1808 ボードの接続**』にしたがって、ホスト PC と XG-1808 のシリアルポート（CN5）とイーサネットポートを接続します。
まだ、AC アダプタを接続して電源は入れないでください。

② XG-1808 のディップスイッチが以下の設定になっていることを確認します。
ディップスイッチの各設定の詳細に関しては、『**XG-1808 ハードウェアマニュアル**』でご確認ください。



③ AC アダプタを接続して、XG-1808 の電源を入れます。
PC-USB-03 がホスト PC に認識されて仮想 COM ポートが作成されます。

④ 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）

⑤ XG-1808 のリセットスイッチ(SW1)を押して、ボードをリセットします。

⑥ リセットから約 2 秒後に Linux カーネルが自動起動し、全ての起動までにはおよそ 10 秒ほどかかります。
なお、起動ログに関しては、本ドキュメントの『**付録 A. 起動ログ**』でご確認ください。

```
U-Boot 2011.09 (Mar 30 2012 - 16:05:56) ALPHAPROJECT XG-1808 vX.X

I2C:   ready
DRAM:  128 MiB
Flash: 16 MiB
MMC:   davinci: 0

       :
    途中省略
       :

Welcome to Buildroot
buildroot login:
```

3.6 Linux の動作確認

XG-1808 上での Linux の動作確認を行います。

ログイン

Linux 起動後、ログインプロンプト『**buildroot login:**』が表示されます。
ログインを実行するにはユーザ『**root**』を入力してください。

ログイン設定	
ユーザ	root
パスワード	なし

Table 4.7-1 ログイン設定

```
Welcome to Buildroot
buildroot login: root    入力
```

時刻設定

XG-1808 上で時刻の設定をします。XG-1808 には RTC(リアルタイムクロック)が搭載されており、電源を OFF にした状態でも時刻を保持することができます。Linux は起動時に RTC から時刻を読み出し、以後は RTC にアクセスすることなく、CPU 内のタイマーモジュールによって時刻を管理しています。Linux のコマンドライン上から RTC にアクセスするには『**hwclock**』コマンドを使用します。

① RTC に設定されている時刻を読み出すには『**hwclock**』コマンドを引数無しで入力します。

```
# hwclock    入力
Sat Jan  1 12:00:00 2000  0.000000 seconds
```

② RTC に設定されている時刻を変更する際には『**date**』コマンドを使用し、システムの時刻を設定し、その更新されたシステムの時刻を『**hwclock**』コマンドで RTC に書き込みます。
例として時刻を 2012 年 2 月 1 日 15 時 30 分に設定します。
『**date -s '2012-02-01 15:30'**』実行後、『**hwclock -w**』を実行してください。

```
# date -s '2012-02-01 15:30'    入力
Wed Feb  1 15:30:00 UTC 2012
# hwclock -w    入力
```

USB メモリ

USB メモリをファイルシステム上の任意のディレクトリにマウントすることにより、他のファイルと同様にアクセスすることができます。活線挿抜に対応しているため、電源を ON にした状態でも USB メモリの抜き差しが可能です。

① USB メモリを USB コネクタに差し込むと以下のようなメッセージがコマンドライン上に出力されます。Linux では、USB メモリは SCSI デバイスとして認識されます。出力されるメッセージは環境により異なります。

```
usb 2-1: new full-speed USB device number 2 using ohci
usb 2-1: not running at top speed; connect to a high speed hub
scsi1 : usb-storage 2-1:1.0

        :
    途中省略
        :

sd 1:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 1:0:0:0: [sda] No Caching mode page present
sd 1:0:0:0: [sda] Assuming drive cache: write through
sd 1:0:0:0: [sda] Attached SCSI removable disk
```

上記のログで、『**sda: sda1**』と表示される場合は、マウント時に『**/dev/sda1**』を指定します。
『**sda**』のみ表示される場合は、『**/dev/sda**』となります。
また、動作環境によっては、『**sdb1**』や『**sdc1**』となる場合もありますので、以降の説明では、上記ログから判断してコマンドを変更ください。

② FAT ファイルシステムでフォーマットされている USB メモリを『**/mnt/usb**』ディレクトリにマウントします。
USB メモリが『**sda1**』として認識している場合は、『**mount -t vfat /dev/sda1 /mnt/usb**』となります。
『**sda1**』以外で認識した場合は、『**/dev/sda1**』の部分を適宜変更してください。

```
# mount -t vfat /dev/sda1 /mnt/usb    入力
```

③ 『**ls**』コマンドで内容を確認することができます。

```
# ls /mnt/usb    入力
a.txt
```

④ 『**umount**』コマンドで USB メモリをアンマウント(マウント解除)することができます。USB メモリをコネクタから引き抜くときは必ずアンマウントを実行してください。
『**umount /mnt/usb**』を実行してください。

```
# umount /mnt/usb    入力
```

4. ブートローダ

4.1 U-Boot 概要

XG-1808 では、ブートローダに U-Boot を使用します。
U-Boot は、CPU やメモリ、周辺デバイスの初期化を行い、Linux カーネル・ルートファイルシステムを RAM 上に展開したあとに Linux カーネルを起動します。
U-Boot は、以下の機能をサポートしています。

- ●コマンドラインインターフェース(CLI)をサポート
- ●シリアル・イーサネットポートによる通信
- ●FAT ファイルシステム対応

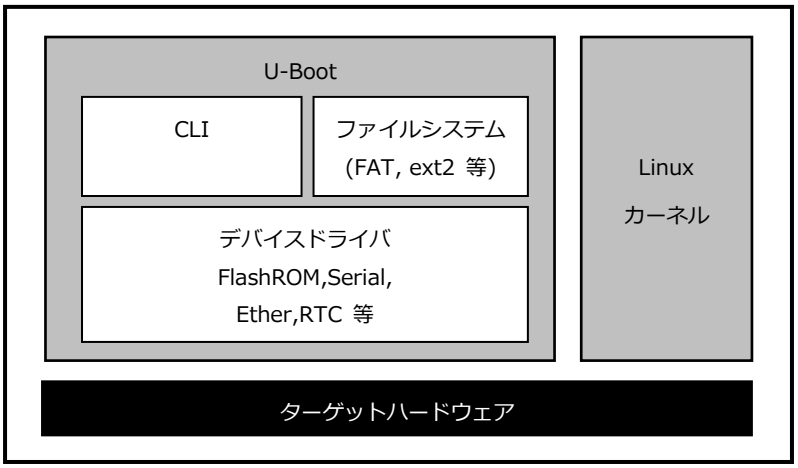
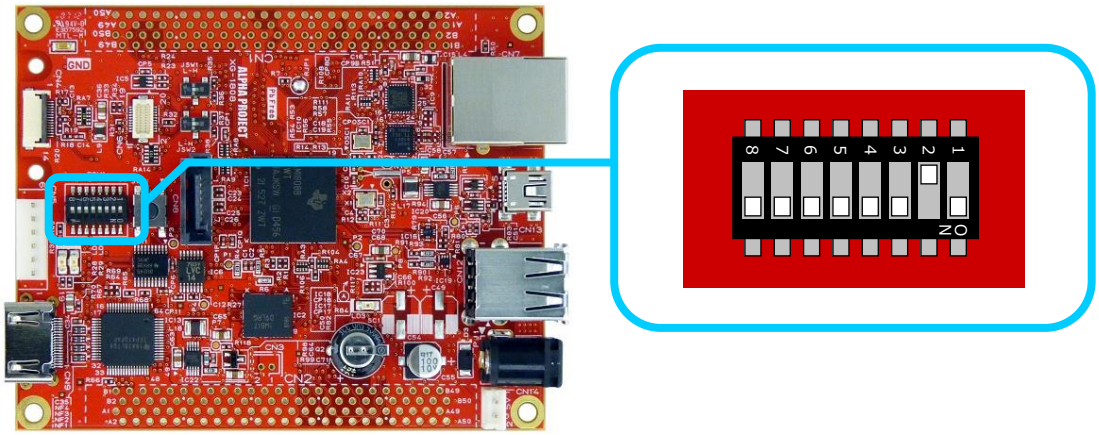


Fig 4.1-1 U-Boot アーキテクチャ

4.2 U-Boot の起動

① 『**3.4 XG-1808 ボードの接続**』にしたがって、ホスト PC と XG-1808 のシリアルポート（CN5）とイーサネットポートを接続します。
まだ、AC アダプタを接続して電源は入れないでください。

② XG-1808 のディップスイッチが以下のようになっていることを確認します。
ディップスイッチの各設定の詳細に関しては、『**XG-1808 ハードウェアマニュアル**』でご確認ください。



③ AC アダプタを接続して、XG-1808 の電源を入れます。
PC-USB-03 がホスト PC に認識されて仮想 COM ポートが作成されます。

④ 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）

⑤ XG-1808 のリセットスイッチ(SW1)を押して、ボードをリセットします。

⑥ リセット後、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、『**=>**』が表示されます。

```
U-Boot 2011.09 (Mar 30 2012 - 16:05:56) ALPHAPROJECT XG-1808 vX.X

I2C:   ready
DRAM:  128 MiB
Flash: 16 MiB
MMC:   davinci: 0
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
ARM:   456 MHz
Net:   DaVinci-EMAC
Hit any key to stop autoboot:  0   <入力>  2 秒以内にキー入力を行います
=>
```

5. Linux

5.1 Linux システムの概要

XG-1808 用 Linux システムは、Linux カーネルとルートファイルシステム(ramfs, sd, norflash)から構成されます。
Linux カーネルは、デバイスドライバとして UART、Ethernet、FlashROM 等をサポートし、ファイルシステムとして ext2、ext3、JFFS2、cramfs、FAT、NFS 等をサポートしています。
ルートファイルシステムは、基本アプリケーションとして、コマンドユーティリティ郡「busybox」が収録されています。

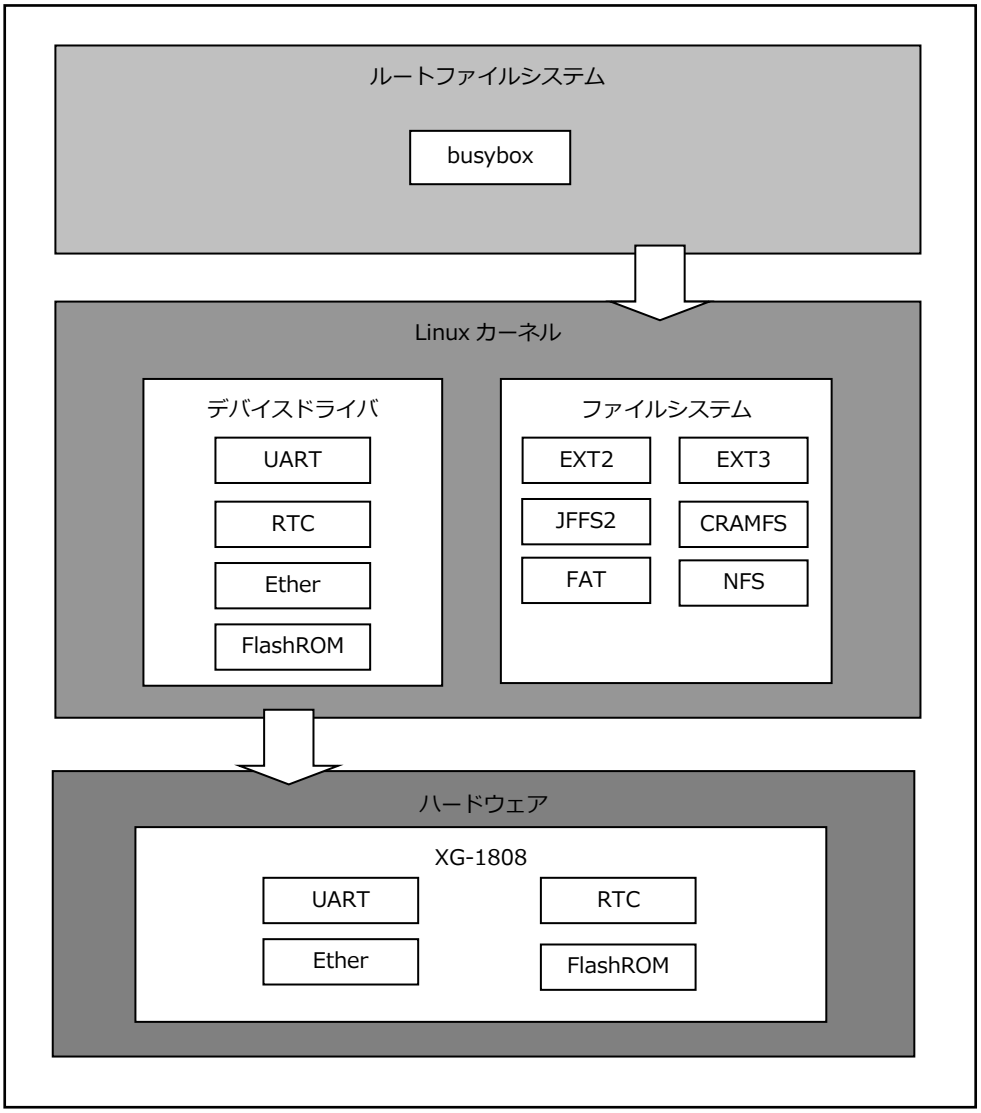


Fig 5.1-1 Linux システム

ramfs 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.cpio.gz**』ファイルが作成されています。

```
省略 $ ls output/images/rootfs.cpio.gz      入力
output/images/rootfs.cpio.gz
```

② U-Boot 用のルートファイルシステムに変換します。

```
省略 $ mkimage -A arm -O linux -T ramdisk -C gzip -d output/images/rootfs.cpio.gz output/i
mages/uInitrd-xg1808     入力
Image Name:
Created:      Fri Mar 30 15:44:38 2012
Image Type:   ARM Linux RAMDisk Image (gzip compressed)
Data Size:    4265462 Bytes = 4165.49 kB = 4.07 MB
Load Address: 0x00000000
Entry Point:  0x00000000
```

③ 起動時に TFTP を使用してロードできるように、『**/srv/tftp**』ディレクトリにルートファイルシステム『**uInitrd-xg1808**』をコピーします。

```
省略 $ cp output/images/uInitrd-xg1808 /srv/tftp      入力
```

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
```

5.4 RAMFS-Linux システムの起動

U-Boot を使用し、『**5.2 Linux カーネルの作成**』で作成した Linux カーネル『**uImage-xg1808**』と ramfs ルートファイルシステム『**uInitrd-xg1808**』をネットワーク経由(TFTP)でダウンロードし RAMFS-Linux システムを起動する方法を示します。

① Linux カーネルイメージ『**uImage-xg1808**』を RAM 上にダウンロードします。

```
=> tftp c0600000 uImage-xg1808      入力
Using DaVinci-EMAC device
TFTP from server 192.168.128.201; our IP address is 192.168.128.200
Filename 'uImage-xg1808'.
Load address: 0xc0600000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ##################################
done
Bytes transferred = 2388768 (247320 hex)
```

② ramfs ルートファイルシステム『**uInitrd-xg1808**』を RAM 上にダウンロードします。

```
=> tftp c0c00000 uInitrd-xg1808      入力
Using DaVinci-EMAC device
TFTP from server 192.168.128.201; our IP address is 192.168.128.200
Filename 'uInitrd-xg1808'.
Load address: 0xc0c00000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ##################################
done
Bytes transferred = 4265526 (411636 hex)
```

6. プログラムの作成

本章では、XG-1808 上で動作するアプリケーションの作成方法について説明します。

6.1 プログラムの開発について

ソースファイルのコンパイルから動作までの一連の流れを示します。

① ゲスト OS 上でソースファイルを作成。
② ゲスト OS 上でソースファイルをクロスコンパイルし、実行ファイルを作成。
③ XG-1808 ボード上でゲスト OS を nfs でマウントし、実行ファイルをダウンロード。
④ XG-1808 ボード上で動作を確認。

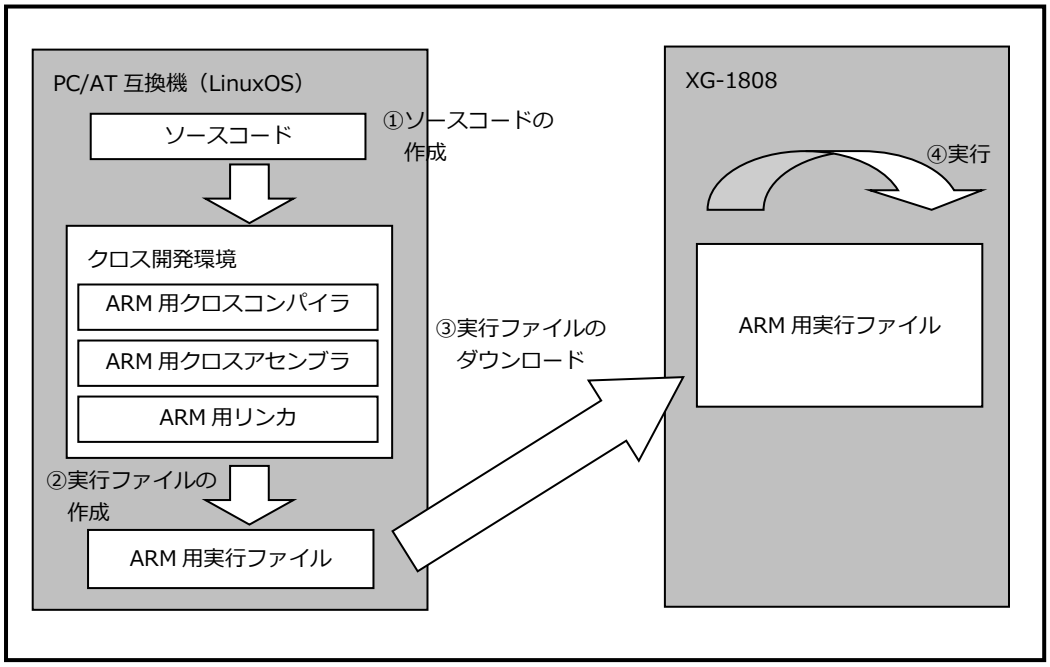


Fig 6.1-1 プログラムの開発手順

7. デバイスドライバの作成

本章では、XG-1808 上の LED にアクセス可能なサンプルデバイスドライバの作成方法とそのデバイスドライバを使用したアプリケーションの作成方法について説明します。

本章で作成するプログラムは、Linux カーネルソースが事前にコンパイル済みである必要があります。
カーネルのコンパイルについては、『**5.2 Linux カーネルの作成**』をご確認ください。

7.1 サンプルデバイスドライバの概要

サンプルデバイスドライバは LED デバイスへのアクセス関数を提供します。

デバイスドライバの概要

ユーザープログラム上からデバイスにアクセスする際、通常はデバイスファイルを通じてシステムコールを発行し、デバイスドライバに処理を依頼します。デバイスドライバはデバイスへのアクセス関数を提供することにより、ユーザープログラム上からデバイスにアクセスする手段を提供します。
サンプルデバイスドライバはキャラクタ型デバイスドライバになり、モジュールとしてコンパイルします。このデバイスドライバは、ユーザープログラム上から LED デバイスにアクセスするための関数を提供します。システムコール（API）は『**open**』、『**close**』、『**write**』になります。サンプルデバイスドライバを示すデバイスファイルは『**/dev/sample0**』になります。

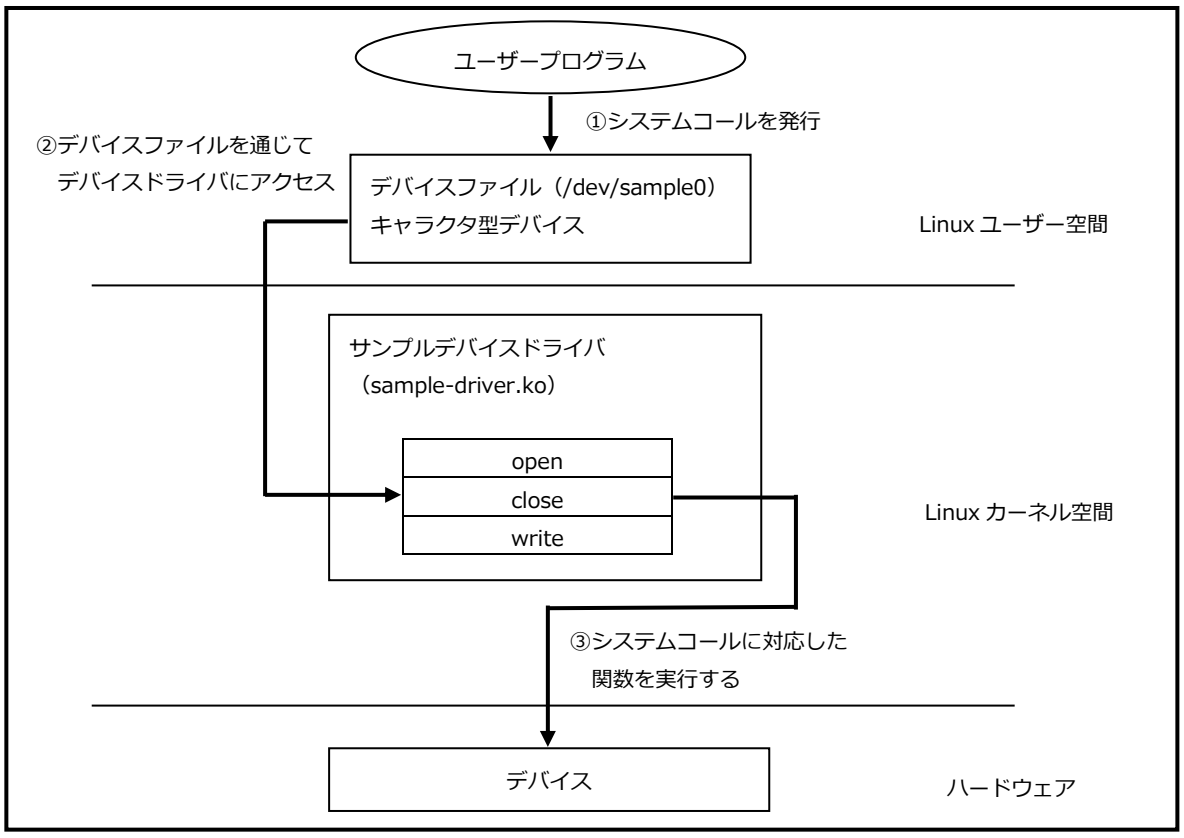


Fig 7.1-1 サンプルデバイスドライバの概要

7.3 動作確認

作成したサンプルデバイスドライバ 及び アプリケーションを XG-1808 上で動作させる手順を説明します。

① XG-1808 で Linux を起動します。起動方法に関しては『**3.5 Linux の起動**』でご確認ください。

② XG-1808 からゲスト OS の『**/nfs**』ディレクトリをマウントします。

```
# mount -t nfs -o nolock 192.168.128.201:/nfs /mnt/nfs
```
入力

③ マウントした場所にある、デバイスドライバモジュールをカーネルに組み込みます。

```
# insmod /mnt/nfs/sample-driver.ko
sample driver(major 253) installed.
```
入力

④ モジュールがカーネルに組み込まれたかを確認します。

```
# lsmod
Module                  Size  Used by    Not tainted
sample_driver           1404  0
```
入力

⑤ アプリケーションを実行します。

```
# /mnt/nfs/sample-app
```
入力

アプリケーション実行後、XG-1808 のボード上の LED が 1 秒毎に以下の順番で状態が変わります。
なお、アプリケーションを終了する場合は、『**Ctrl+c**』を入力してください。

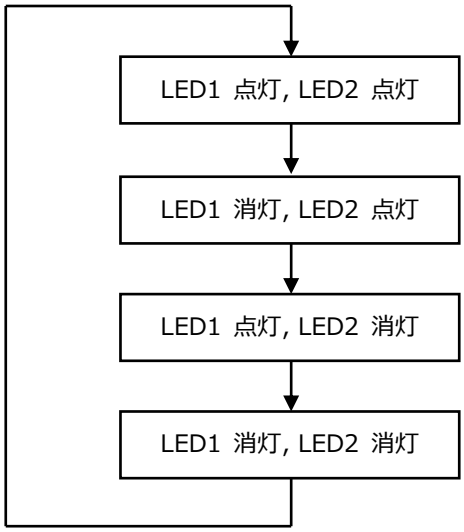


Fig 7.3-1 サンプル LED の遷移

9. タッチパネル LCD キットの使用

本章では、XG-1808 にタッチパネル LCD キットを接続して動作を行う方法を説明します。

9.1 対応しているタッチパネル LCD キット

Linux カーネルにはタッチパネル LCD キットのドライバが含まれています。

タッチパネル LCD キット	デバイス名
LCD-KIT-B01	lcdkitb01
LCD-KIT-B02	lcdkitb02
LCD-KIT-C01	lcdkitc01
LCD-KIT-C02	lcdkitc01
LCD-KIT-D01	lcdkitd01
LCD-KIT-D02	lcdkitd02

Table 9.1-1 タッチパネル LCD キット対応表

本マニュアルでは例としてLCD-KIT-B01の表記を扱います。
使用するタッチパネルLCDキットに応じて表記内のデバイス名(lcdkitb01)を適宜読み替えてください。
※LCD-KIT-C02 を使用する場合は、デバイス名を「lcdkitc01」に読み替えてください。

サンプルアプリケーションのコンパイル

サンプルアプリケーションのコンパイル手順を説明します。

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

```
省略 $ cd ~/xg1808-lk/lcdkit      入力
```

② サンプルアプリケーションをコンパイルします。

『**BR_DIR**』は、buildroot のコンパイル出力先を指定します。

```
省略 $ BR_DIR=~/xg1808-lk/buildroot-2011.11-xg1808/output make      入力
arm-linux-gcc -Wall --sysroot /home/guest/xg1808-lk/buildroot-2011.11-xg1808/output/staging`PKG_CONFIG_SYSROOT_DIR=/home/guest/xg1808-lk/buildroot-2011.11-xg1808/output/staging /home/guest/xg1808-lk/buildroot-2011.11-xg1808/output/host/usr/bin/pkg-config --cflags directfb`  --sysroot /home/guest/xg1808-lk/buildroot-2011.11-xg1808/output/staging  lcdkit.c `PKG_CONFIG_SYSROOT_DIR=/home/guest/xg1808-lk/buildroot-2011.11-xg1808/output/staging/home/guest/xg1808-lk/buildroot-2011.11-xg1808/output/host/usr/bin/pkg-config --libs directfb` -o lcdkit
```

③ アプリケーションプログラムを NFS の共有ディレクトリにコピーします。

```
省略 $ cp lcdkit /nfs      入力
```

9.4 動作確認

『**9.2 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-1808 上で、タッチパネル LCD キット用のサンプルアプリケーションを動作させる手順を説明します。
なお、ルートファイルシステムは、『**5.3 ルートファイルシステムの作成**』で作成した ramfs ルートファイルシステムを使用した方法で説明します。

① 『**4.2 U-Boot の起動**』の手順に従って、U-Boot を起動します。
なお、XG-1808 ボードには、以下のようにタッチパネル LCD キットを接続します。
詳しい接続方法に関しては、使用するタッチパネル LCD キットの『**ハードウェアマニュアル**』でご確認ください。

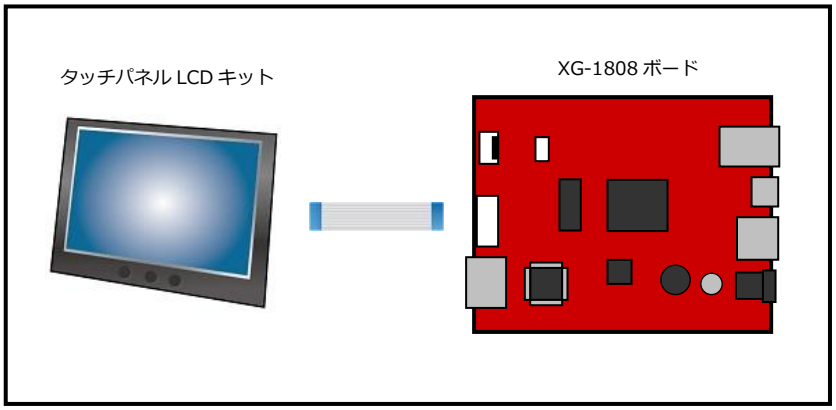


Fig 9.4-1 XG-1808 とタッチパネル LCD キットの接続

⑪ XG-1808 からゲスト OS の『**/nfs**』ディレクトリをマウントします。

```
# mount -t nfs -o nolock 192.168.128.201:/nfs /mnt/nfs
```

⑫ アプリケーションを実行します。
タッチパネルをタッチするとその時のタッチパネル LCD キットから取得した値を画面上に表示します。
終了する場合は、『**Ctrl+c**』を入力してください。

```
# /mnt/nfs/lcdkit

   ~~~~~~~~~~~~~~~~~~~~~~~~~~| 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. (2012-03-29 04:16)
(*) Direct/Memcpy: Using libc memcpy()

          :
       以降省略
          :
```

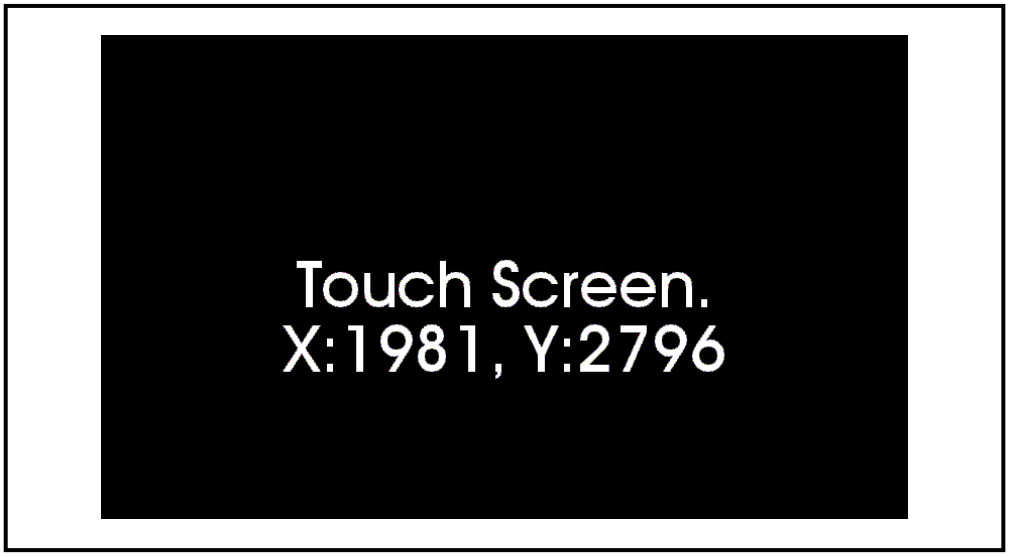


Fig 9.4-2　タッチパネル LCD キットのサンプル動作画面

表示している座標は、タッチパネル LCD キットのコマンドで取得した値そのままとなります。
値の詳細に関しては、各タッチパネル LCD キットの『**ハードウェアマニュアル**』でご確認ください。
なお、表示座標は、マルチタッチ対応のタッチパネル LCD の場合は、『**1st Finger touch**』のみとなります。

参考文献

VMware Player については以下の URL を参考にしてください。

・VMware 社ホームページ

http://www.vmware.com/jp/

・VMware Player 製品ホームページ

http://www.vmware.com/jp/products/player/

謝辞

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
https://www.apnet.co.jp
E-MAIL：query@apnet.co.jp