XG Series

LK-1707-A01

ARM9 AM1707 CPU BOARD

Software Manual

Rev 1.4

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-1707 は、TEXAS INSTRUMENTS ARM9 マイクロプロセッサ「AM1707」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 で動作する専用パッケージを使用します。

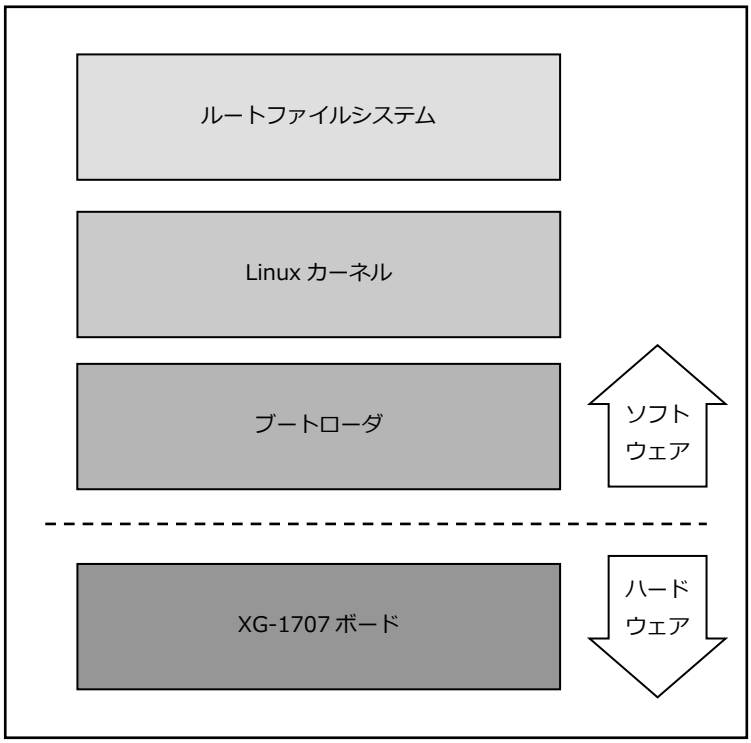


Fig 2.1-1　XG-1707 システム概要図

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-1707 のブートローダには、U-Boot を使用します。

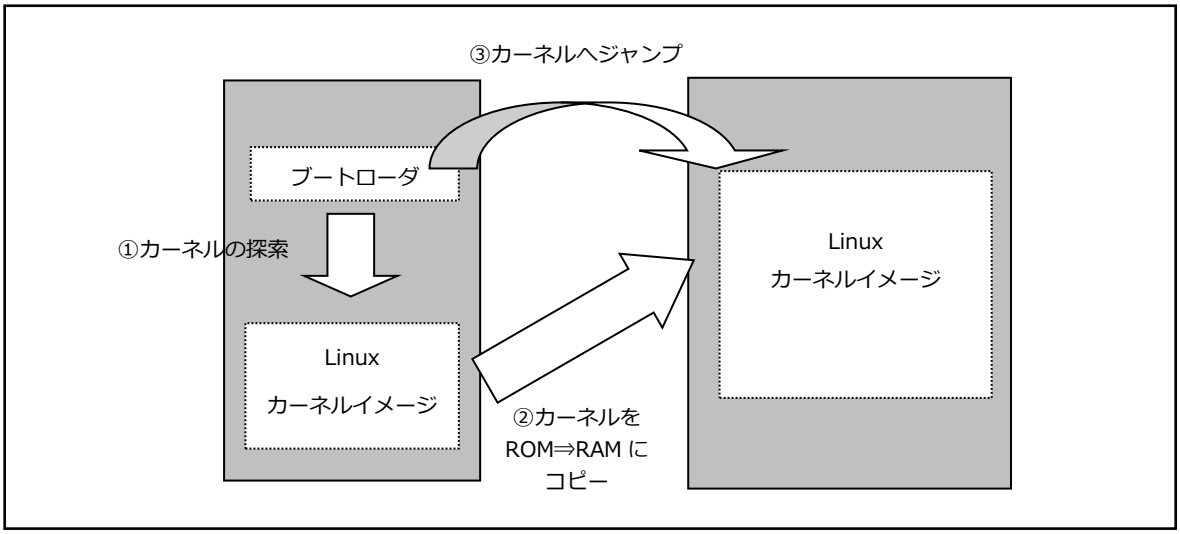


Fig 2.2-1 ブートローダ動作イメージ

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。

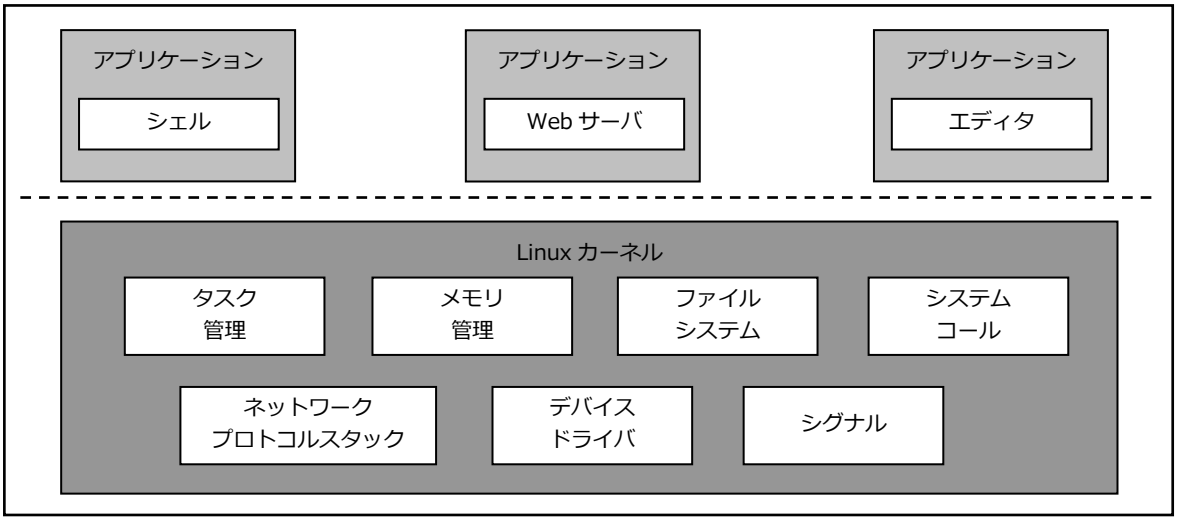


Fig 2.3-1 Linux システム概要

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-1707 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。

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 システムと表現します。

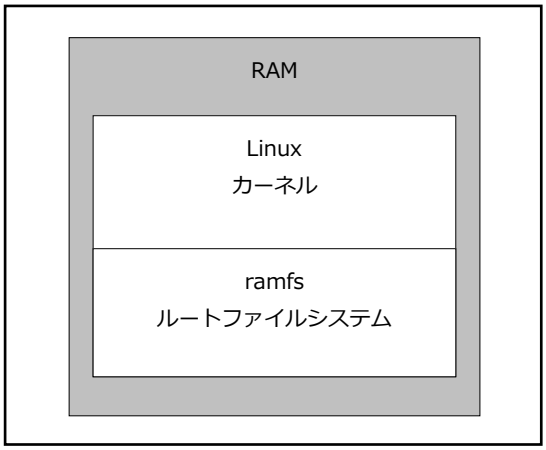


Fig 2.4-1 RAMFS-Linux システム

2.5　クロス開発環境

XG-1707 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。

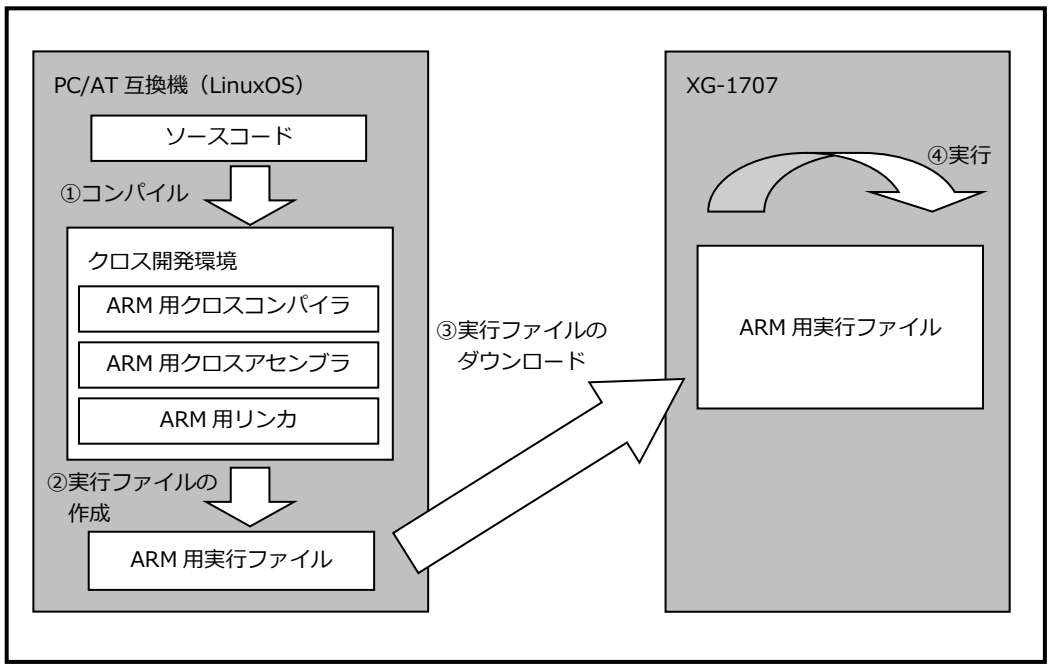


Fig 2.5-1 クロス開発環境

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。また、シリアルポートが使用可能な PC が必要になります。
PC では、ハイパーターミナル等のターミナルソフトウェアを動作させます。

●電源
XG-1707 本体に必要な電源は DC5V±5%です。

●LAN
XG-1707 をネットワークに接続する場合は、LAN ケーブルを接続してください。直接ホスト PC と接続する際はクロスケーブル、ハブを介してネットワークに接続する際はストレートケーブルをご使用ください。
LAN ケーブルは、10/100BASE-TX 対応（UTP カテゴリ 5）ケーブルをご利用ください。

使用機器等	環　　境
CPU ボード	XG-1707
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
シリアルポート	1 ポート
LAN ポート	10/100BASE-TX　1 ポート
RS232 ケーブル	クロスケーブルを使用
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
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%　1A 以上）

Table 3.1-1　動作環境

上記の環境は、XG-1707 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-1707 ボードの接続

ホスト PC と XG-1707 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。

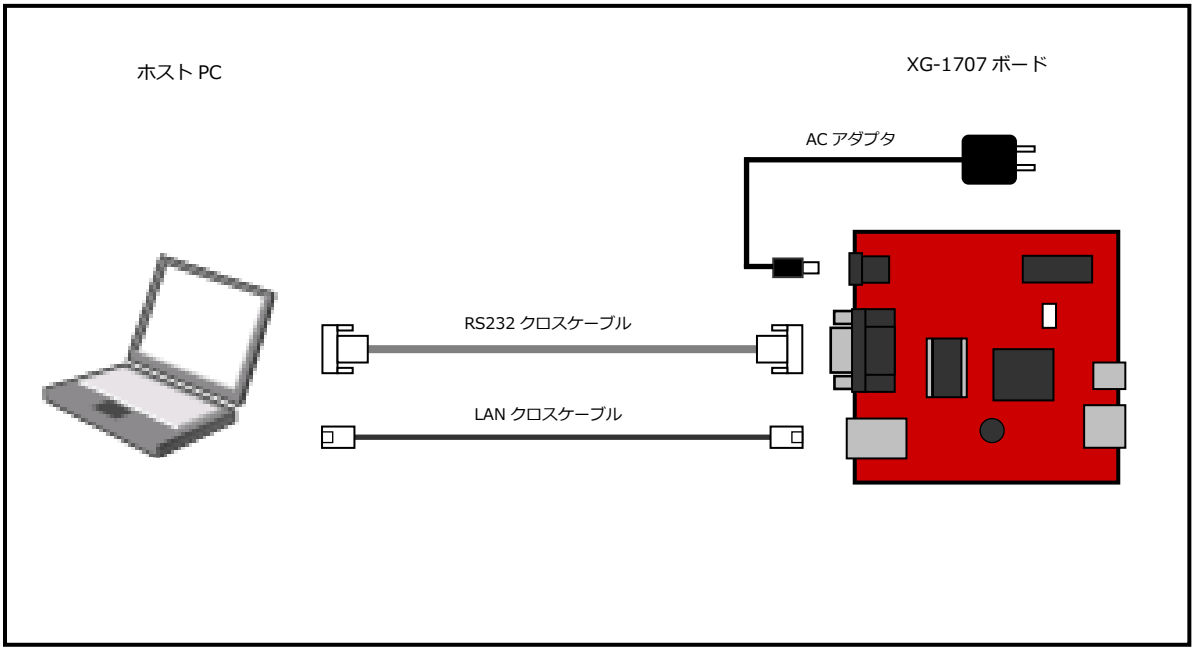


Fig 3.4-1 XG-1707 ボードの接続（PC に接続する場合）

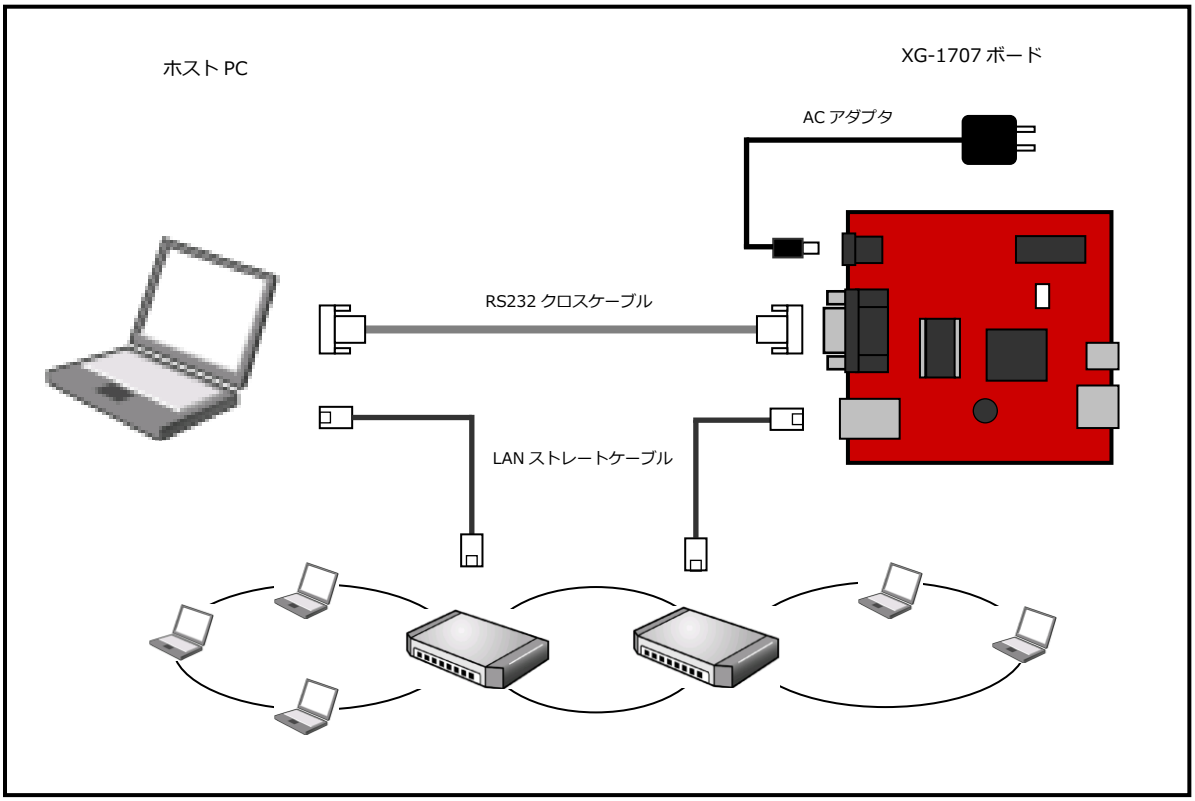


Fig 3.4-2 XG-1707 ボードの接続（HUB に接続する場合）

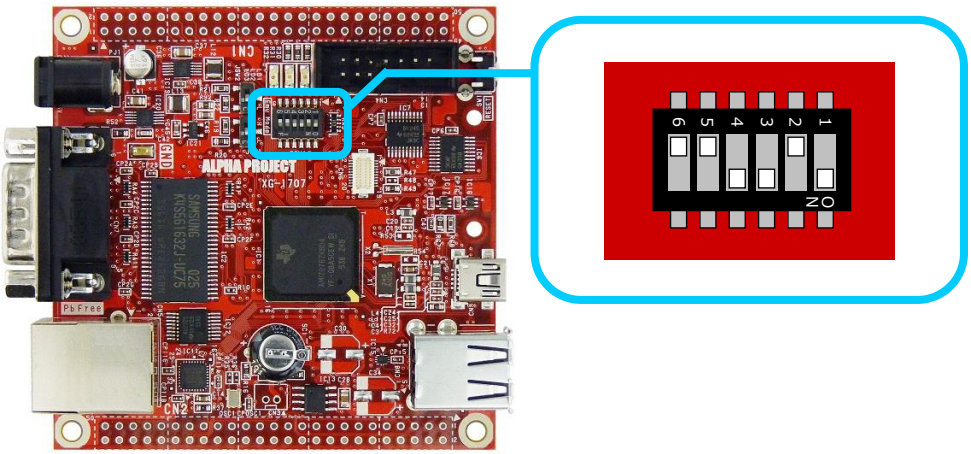
3.5 Linux の起動

XG-1707 上で Linux の起動を行います。
XG-1707 は出荷時状態で Linux が自動起動します。

① 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 シリアル設定**』を参照してください）

② 『**3.4 XG-1707 ボードの接続**』にしたがって、ホスト PC と XG-1707 のシリアルポート（CN7）とイーサネットポートを接続します。

③ XG-1707 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-1707 ハードウェアマニュアル**』でご確認ください。



④ XG-1707 の電源を投入します。
AC アダプタを接続すると XG-1707 の電源が入ります。電源が投入されると、約 2 秒後に Linux カーネルが自動起動します。
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 (Feb 02 2012 - 12:47:52) ALPHAPROJECT XG-1707 vX.X

I2C:   ready
DRAM:  64 MiB
Flash: 16 MiB

          :
      途中省略
          :

Welcome to Buildroot
buildroot login:
```

3.6 Linux の動作確認

XG-1707 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-1707 上で時刻の設定をします。XG-1707 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 1-1: new high-speed USB device number 2 using musb-hdrc
scsi0 : usb-storage 1-1:1.0
scsi 0:0:0:0: Direct-Access  XXXXXXXXXX USB Flash Disk   BC01 PQ: 0 ANSI: 0 CCS
sd 0:0:0:0: [sda] 1982464 512-byte logical blocks: (1.01 GB/968 MiB)
sd 0:0:0:0: Attached scsi generic sg0 type 0
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] No Caching mode page present
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] No Caching mode page present
sd 0:0:0:0: [sda] Assuming drive cache: write through
 sda: sda1
sd 0:0:0:0: [sda] No Caching mode page present
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [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-1707 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

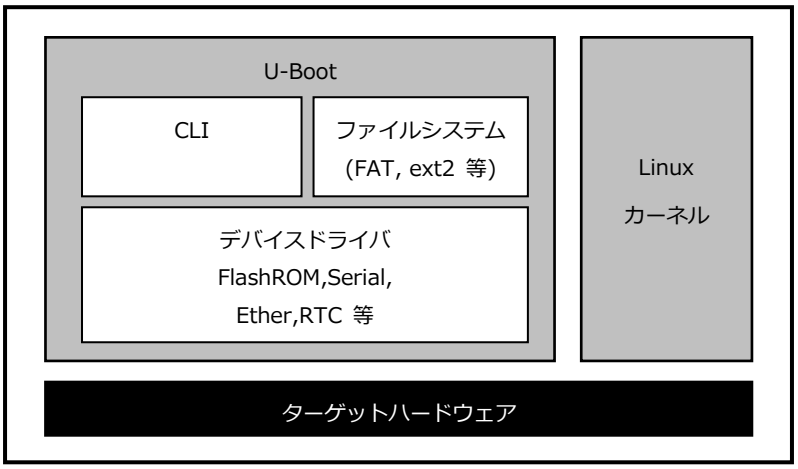


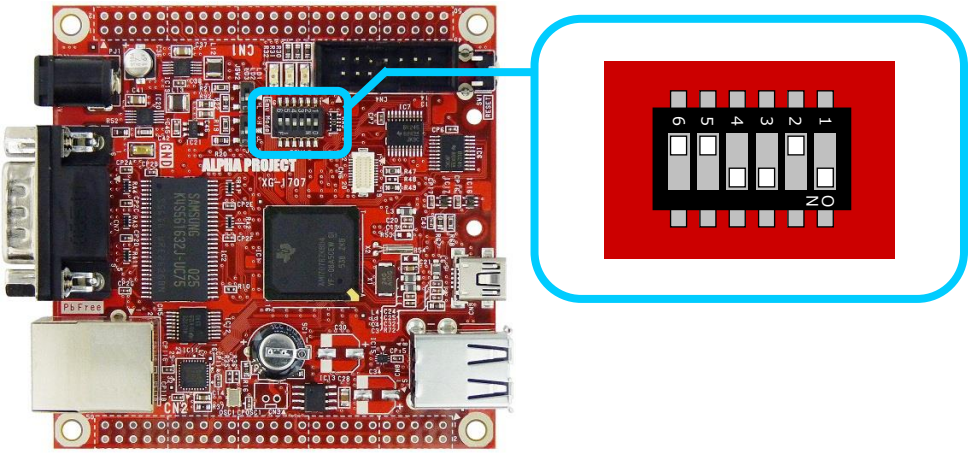
Fig 4.1-1 U-Boot アーキテクチャ

4.2 U-Boot の起動

① 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 シリアル設定**』を参照してください）

② 『**3.4 XG-1707 ボードの接続**』にしたがって、ホスト PC と XG-1707 のシリアルポート（CN7）とイーサネットポートを接続します。

③ XG-1707 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-1707 ハードウェアマニュアル**』でご確認ください。



④ 電源投入直後、『**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 (Feb 02 2012 - 12:47:52) ALPHAPROJECT XG-1707 vX.X

I2C:   ready
DRAM:  64 MiB
Flash: 16 MiB
*** 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-1707 用 Linux システムは、Linux カーネルと ramfs ルートファイルシステムから構成されます。
Linux カーネルは、デバイスドライバとして UART、Ethernet、FlashROM をサポートし、ファイルシステムとして ext2、ext3、JFFS2、cramfs、FAT、NFS をサポートしています。
ramfs ルートファイルシステムは、Linux カーネルと共に FlashROM に格納します。ルートファイルシステムに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」が収録されます。

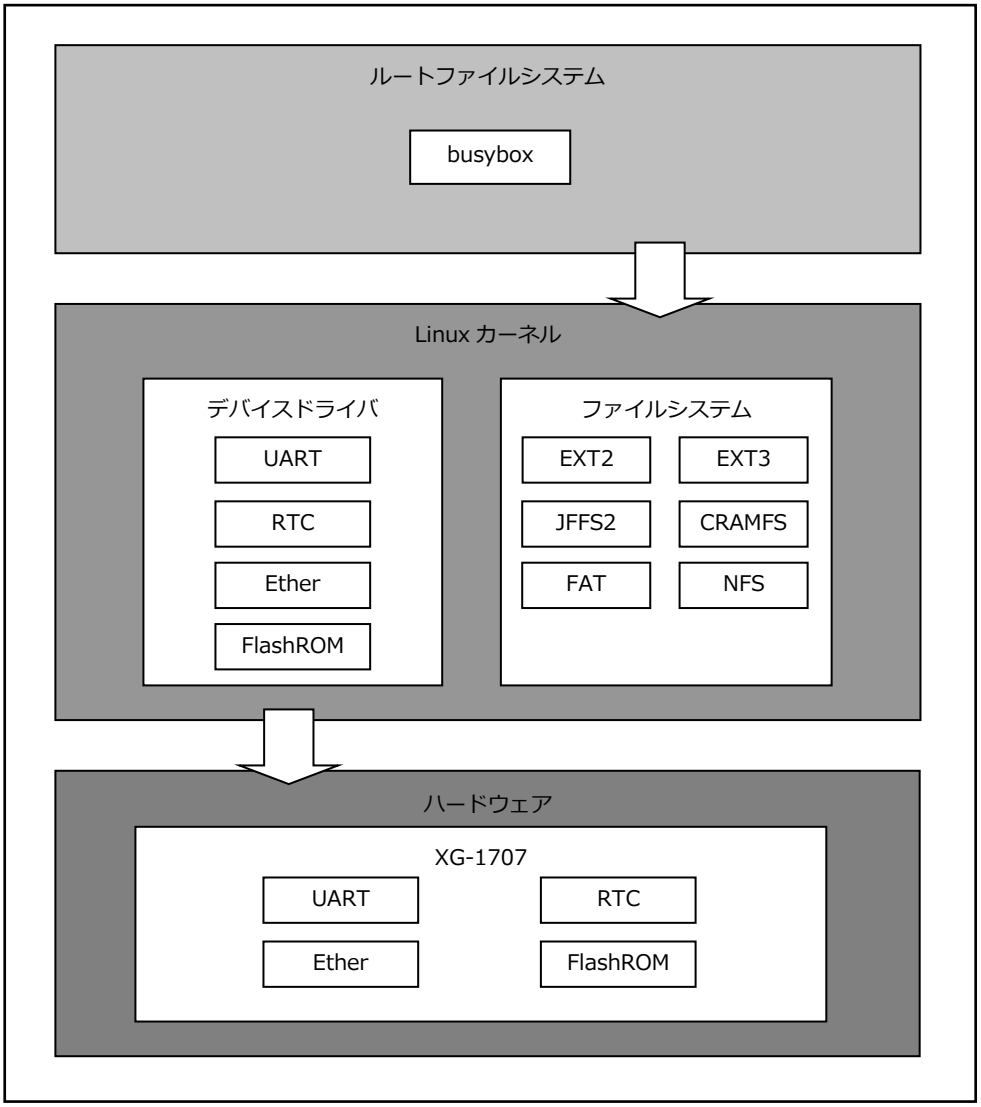


Fig 5.1-1 Linux システム

5.4 RAMFS-Linux システムの起動

U-Boot を使用し、前節で作成した Linux カーネル『**uImage-xg1707**』と ramfs ルートファイルシステム『**uInitrd-xg1707**』をネットワーク経由でダウンロードし 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-xg1707**』を RAM 上にダウンロードします。

```
=> tftp c0600000 uImage-xg1707    入力
Using DaVinci-EMAC device
TFTP from server 192.168.128.201; our IP address is 192.168.128.200
Filename 'uImage-xg1707'.
Load address: 0xc0600000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ##################################
done
Bytes transferred = 2344984 (23c818 hex)
```

② ramfs ルートファイルシステム『**uInitrd-xg1707**』を RAM 上にダウンロードします。

```
=> tftp c0c00000 uInitrd-xg1707    入力
Using DaVinci-EMAC device
TFTP from server 192.168.128.201; our IP address is 192.168.128.200
Filename 'uInitrd-xg1707'.
Load address: 0xc0c00000
Loading: #################################################################
         #################################################################
         #################################################################
         #################################################################
         #################################################################
         ##################################
done
Bytes transferred = 2102827 (20162b hex)
```

③ ダウンロードしたイメージを起動します。

```
=> bootm c0600000 c0c00000    入力
## Booting kernel from Legacy Image at c0600000 ...
   Image Name:   Linux-3.1.0-XG1707
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2181272 Bytes = 2.1 MiB
   Load Address: c0008000

         :
      途中省略
         :

Starting /usr/sbin/httpd: done

Welcome to Buildroot
buildroot login:
```

6. プログラムの作成

本章では、XG-1707 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-1707 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-1707 ボード上で動作を確認。

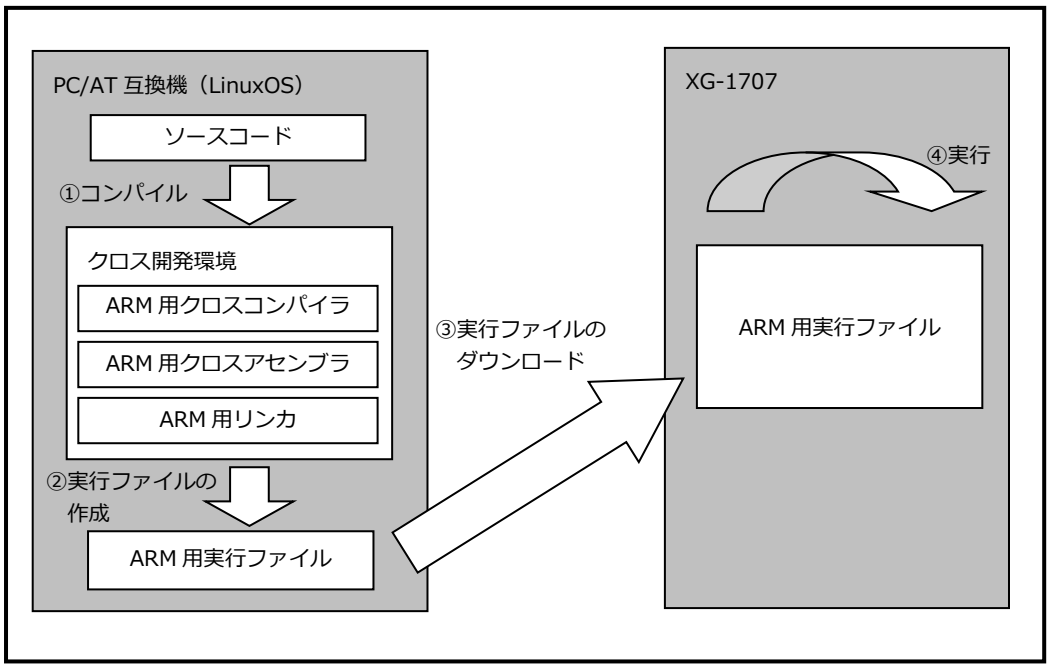


Fig 6.1-1 プログラムの開発手順

7. デバイスドライバの作成

本章では、XG-1707 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、「**5.3 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。

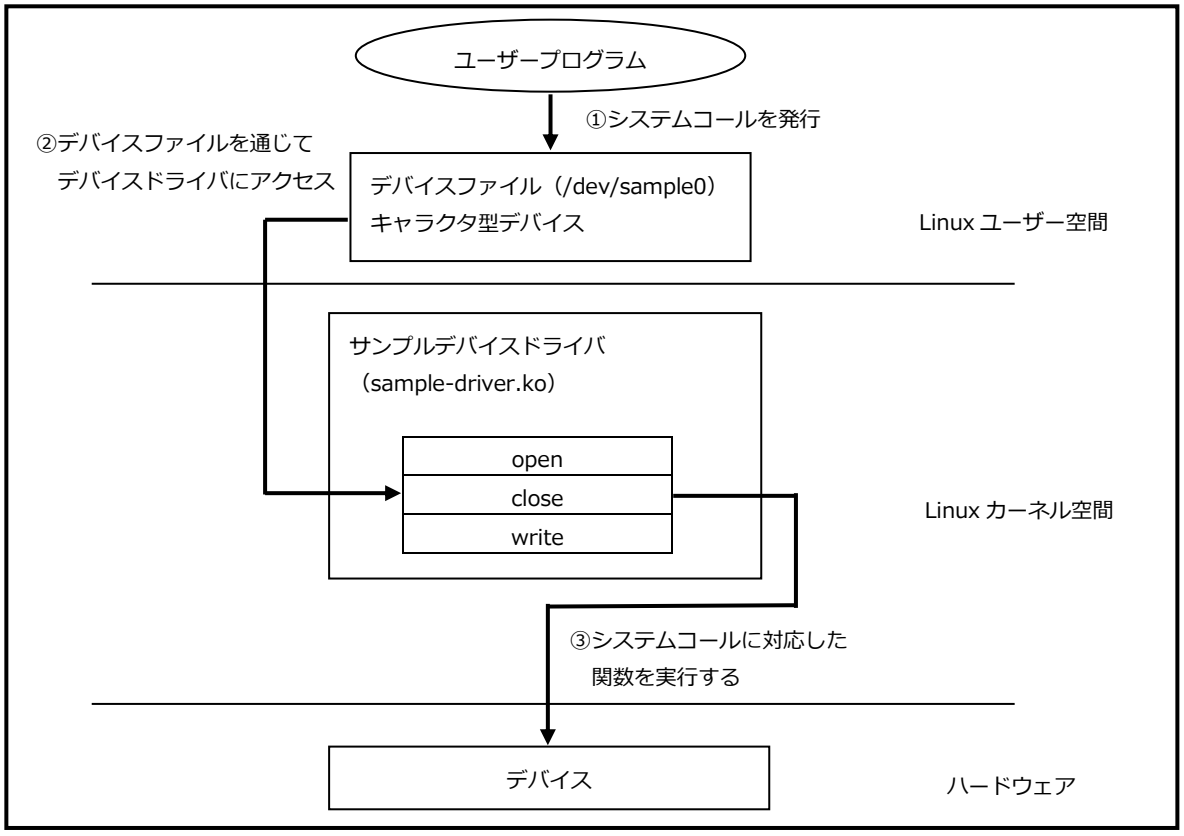


Fig 7.1-1 サンプルデバイスドライバの概要

システムコール

サンプルデバイスドライバは『**open**』、『**close**』、『**write**』システムコールを提供します。
このシステムコールを使用することで、LED デバイスに対して制御することができます。
サンプルデバイスドライバの各システムコールについて下記に示します。各システムコールの書式は Linux の標準 API に従います。

- open システムコール

機能	デバイスをオープンする
書式	int open(char* devicename, int flags)
引数	devicename　　：論理デバイス名 flags　　：フラグ
戻り値	ファイルディスクリプタを返す エラー時は-1 を返す
備考	論理デバイス名『**/dev/sample0**』を使用 フラグは O_RDWR を使用

- close システムコール

機能	デバイスをクローズする
書式	int close(int fd)
引数	fd　　：ファイルディスクリプタ
戻り値	成功時には 0、エラー時は-1 を返す
備考	

- write システムコール

機能	32 ビットライトアクセスを実行する。
書式	ssize_t write(int fd, const void* data, size_t count)
引数	fd　　：ファイルディスクリプタ data　　：設定するデータの先頭アドレス count　　：データ数(2 固定)
戻り値	成功時には設定したサイズ、エラー時は-1 を返す
備考	データは、2Byte 構成で、1Byte 目は LED1 の情報を設定し、2Byte 目は LED2 の情報を設定します。

9. ボードの初期化

XG-1707 は、FlashROM に U-Boot、Linux カーネル等が書き込まれた状態で出荷しております。
何らかの理由により、出荷状態に戻したい場合は、本章の手順に従って行ってください。

9.1 FlashROM 構成

製品出荷状態では、以下のアドレスマップの構成で FlashROM に書き込まれております。
次節より、出荷状態とするための方法を説明します。

FlashROM 16Mbyte

開始アドレス	領域名	領域サイズ
0x60000000	UBL	128KByte
0x60020000	U-Boot	384KByte
0x60080000	U-Boot 環境変数領域	128KByte
0x600a0000	未使用	384KByte
0x60100000	Linux カーネル	3MByte
0x60400000	ramfs ルートファイルシステム	4MByte
0x60800000	未使用	8MByte

Fig 9.1-1 出荷時の FlashROM 構成

9.2 書き込み準備

書き込み手順では、USB メモリが必要になります。
USB メモリをご用意いただき、FAT でフォーマットしてください。

ホスト PC 側の特定のフォルダに、以下のファイルを入れます。

sfh_OMAP-L137.exe
ubl_AM1707_NOR.bin
u-boot-xg1707-X.X.bin

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

用意した USB メモリのルートディレクトリに以下のファイルを入れます。

uImage-xg1707-X.X
uInitrd-xg1707-X.X

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

sfh_OMAP-L137.exe を実行する環境(ホスト PC)には、『**.NET Framework 2.0**』が必要になります。
もしインストールされていない場合は、インストールしてから作業を行ってください。

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
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