

Alpha Board Series

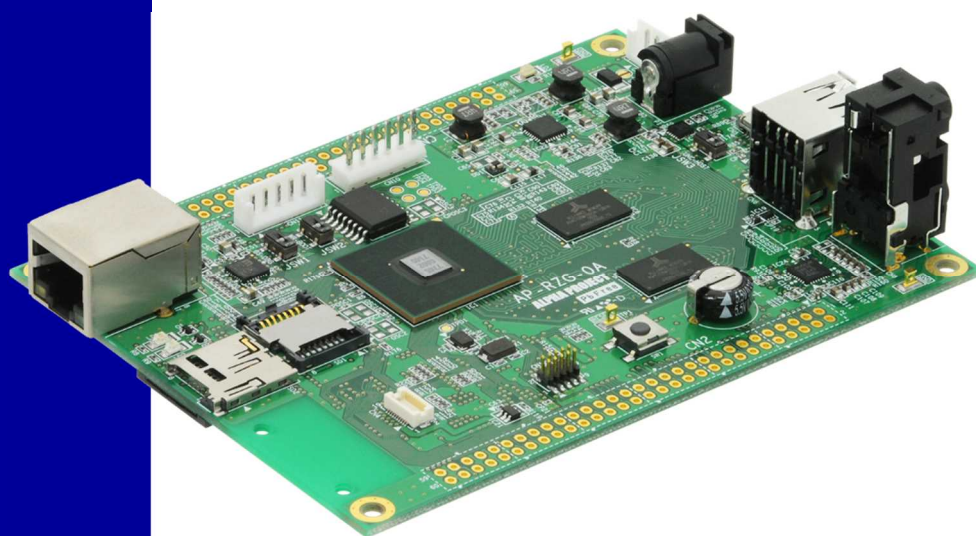
# LK-RZG-A01

Cortex-A7 R8A7745 CPU  
BOARD

## Software Manual

Rev 1.0

ダイジェスト版



**ALPHAPROJECT**

<http://www.apnet.co.jp>

## ご使用になる前に

このたびは AP-RZG-0A Linux 開発キット(LK-RZG-A01)をお買い上げいただき誠にありがとうございます。  
本製品をお役立て頂くために、このマニュアルを十分お読みいただき、正しくお使い下さい。  
今後共、弊社製品をご愛顧賜りますよう宜しくお願いいたします。

## 梱包内容

本製品は、下記の品より構成されております。梱包内容をご確認のうえ、万が一、不足しているものがあればお買い上げの販売店までご連絡ください。

### LK-RZG-A01 梱包内容

<p>●LAN ストレートケーブル 1本</p> 	<p>●AC アダプタ 1本</p> 
<p>●PC-USB-04(ケーブル付) 1個</p> 	<p>●USB ケーブル(A - B) 1本</p>  <p>コネクタの形状</p> 
<p>●microSD カード(4GB、アダプター付き) 1個</p> 	<p>●CD-ROM 1枚</p>

■本製品の内容及び仕様は予告なしに変更されることがありますのでご了承ください。

# 目次

1. 概要	1
1.1 はじめに	1
1.2 Linux について	1
1.3 U-Boot について	1
1.4 VirtualBox について	2
1.5 Ubuntu について	2
1.6 GNU と FSF について	2
1.7 Yocto Project について	2
1.8 GPL と LGPL について	3
1.9 保証とサポート	3
2. システム概要	4
2.1 システム概要	4
2.2 ブートローダ	5
2.3 Linux カーネル	5
2.4 ルートファイルシステム	6
2.5 クロス開発環境	7
2.6 添付 CD-ROM の構成	8
3. システムの動作	9
3.1 動作環境	9
3.2 シリアル初期設定値	10
3.3 ネットワーク初期設定値	10
3.4 USB ID 初期設定値	12
3.5 AP-RZG-0A ボードの接続	13
3.6 動作確認用 microSD カードの作成	14
3.7 Linux の起動	16
3.8 Linux の動作確認	18
3.9 ネットワークの設定	27
3.10 Linux の終了	30
4. Linux システムの構築	31
4.1 Linux システムの概要	31
4.2 ルートファイルシステムの概要	32
4.3 Yocto / Poky	33
4.4 Yocto のビルド環境の準備	33
4.5 Yocto のビルド	36
4.6 microSD カードの作成	38

4.7	カーネルのカスタマイズ.....	40
<b>5.</b>	<b>ブートローダ</b>	<b>43</b>
5.1	U-Boot 概要 .....	43
5.2	ブートローダの起動 .....	44
5.3	ネットワーク設定 .....	46
5.4	U-Boot の作成 .....	48
5.5	U-Boot の書込み .....	49
<b>6.</b>	<b>プログラムの作成</b>	<b>51</b>
6.1	プログラムの開発について .....	51
6.2	サンプルアプリケーションのビルド .....	52
6.3	動作確認 .....	54
<b>7.</b>	<b>デバイスドライバの作成</b>	<b>55</b>
7.1	サンプルデバイスドライバの概要 .....	55
7.2	サンプルデバイスドライバ/アプリケーションのビルド .....	57
7.3	動作確認 .....	59
<b>8.</b>	<b>製品サポートのご案内</b>	<b>60</b>
<b>9.</b>	<b>エンジニアリングサービスのご案内</b>	<b>61</b>
<b>付録 A.</b>	<b>起動ログ</b>	<b>62</b>
<b>付録 B.</b>	<b>付属品について</b>	<b>68</b>

## 2. システム概要

### 2.1 システム概要

AP-RZG-0A は、CPU コアに Arm Cortex-A7 を採用したマイクロプロセッサ「R8A7745」（RENESAS）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、Linux カーネル、ルートファイルシステムから構成されます。それぞれ、Yocto Project を利用して作成します。

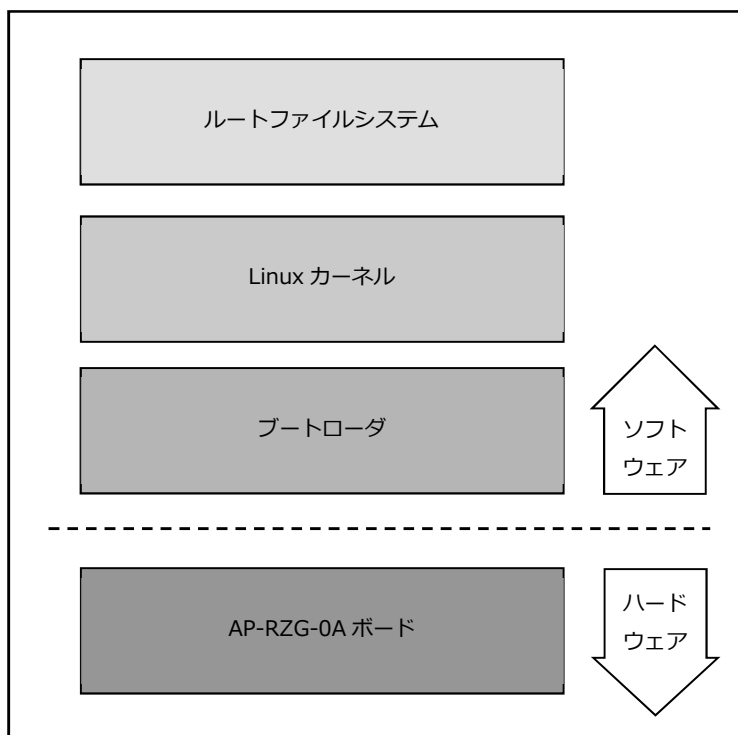


Fig 2.1-1 AP-RZG-0A システム概要図

## 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。

LK-RZG-A01 では、以下のルートファイルシステムを用意しています。

- 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。

本ドキュメントでは、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 システムと表現します。

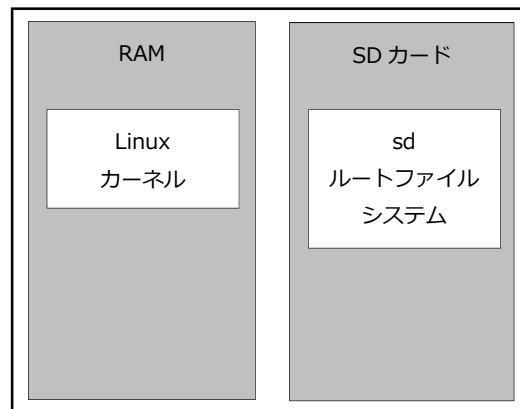


Fig 2.4-1 SD-Linux システム

## 2.6 添付 CD-ROM の構成

AP-RZG-0A の Linux の開発に必要なファイルは、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 CD-ROM から入手することができます。

LK_RZG_A01_VX_X	
-- an	
-- an1617.pdf	: AN1617 タッチパネル LCD キットの使用方法
-- an1618.pdf	: AN1618 カメラモジュールの使用方法
-- an1619.pdf	: AN1619 無線 LAN モジュールの使用方法
-- binaries	
-- sd_image	: microSD カードイメージ
-- APRZGOA_MON_DRAM_SPI_VXXX_40000000.mot	: MiniMonitor の S レコード
-- APRZGOA_SPI_LOADER_VXXX.mot	: Loader の S レコード
-- core-image-base-aprzg0a.tar.bz2	: sd ファイルシステムバイナリ
-- helloworld	: サンプルアプリ
-- sample-app	: サンプルアプリ (デバイス確認用)
-- sample-driver.ko	: サンプルデバイスドライバ
-- u-boot.srec	: U-Boot の S レコード
-- uImage	: Linux カーネルバイナリ
-- uImage-r8a7745-aprzg0a.dtb	: デバイスツリーバイナリ
-- vscam01_test	: カメラモジュール用サンプルアプリ
-- driver	: USB 仮想シリアルドライバー式
-- index.html	: インデックス HTML
-- index_images	: インデックス HTML イメージ
-- manual	
-- lk_rzg_a01_sw.pdf	: Linux 開発 ソフトウェアマニュアル
-- lk_install_rzg.pdf	: Linux 開発 インストールマニュアル
-- sample	
-- devicedriver-X.X.tar.bz2	: サンプルデバイスドライバソース
-- helloworld-X.X.tar.bz2	: サンプルアプリソース
-- vscam-1.0.tar.bz2	: カメラモジュールアプリソース
-- sources	
-- APRZGOA_LOADER_VXXX.tar.bz2	: Loader のソース
-- APRZGOA_SpiBoot_miniMonDram_VXXX.tar.bz2	: MiniMonitor のソース
-- meta-renesas-aprzg0a-sa-X.X.tar.gz	: AP-RZG-0A 用のレシピファイル
-- RZG_Series_Evaluation_Software_Package_for_Linux-20150727.tar.gz	: RZG シリーズのレシピファイル
-- RZG_Series_Evaluation_Software_Package_of_Linux_Drivers-20161207.tar.gz	: RZG シリーズのレシピファイル

Table 2.6-1 CD-ROM 内容

※ 『X\_X』、『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』、『1.0』になります。  
『VXXX』もバージョン番号を示します。バージョン V1.00 の場合は『V100』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。

本 Linux 開発キットには、PC-USB-04 が付属しており、PC-USB-04 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-04 の使用方法に関しては、PC-USB-04 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、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	AP-RZG-0A
HOST PC	PC/AT 互換機 (64bit)
OS	Windows 7/8.1/10 (64bit)
メモリ	使用 OS による
ソフトウェア	ターミナルソフトウェア
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)
PC-USB-04	ホスト PC と AP-RZG-0A のシリアル接続用に使用
USB ケーブル	PC-USB-04 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
Audio 入出力機器	Audio 入出力の動作確認時に使用 (マイク, スピーカー)
microSD カード	SD ルートファイルシステム作成に使用 SD スロット (SD2) の動作確認もする場合には、2 枚必要
microUSB ケーブル	AP-RZG-0A の USB ファンクションの動作確認時に使用
PC-CAN-02	CAN 通信の動作確認時に使用
電源	AC アダプタ (DC5V±5%)

Table 3.1-1 動作環境



上記の環境は、AP-RZG-0A 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、開発キット付属の『Linux 開発 インストールマニュアル』でご確認ください。



### 3.5 AP-RZG-0A ボードの接続

ホスト PC と AP-RZG-0A 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。

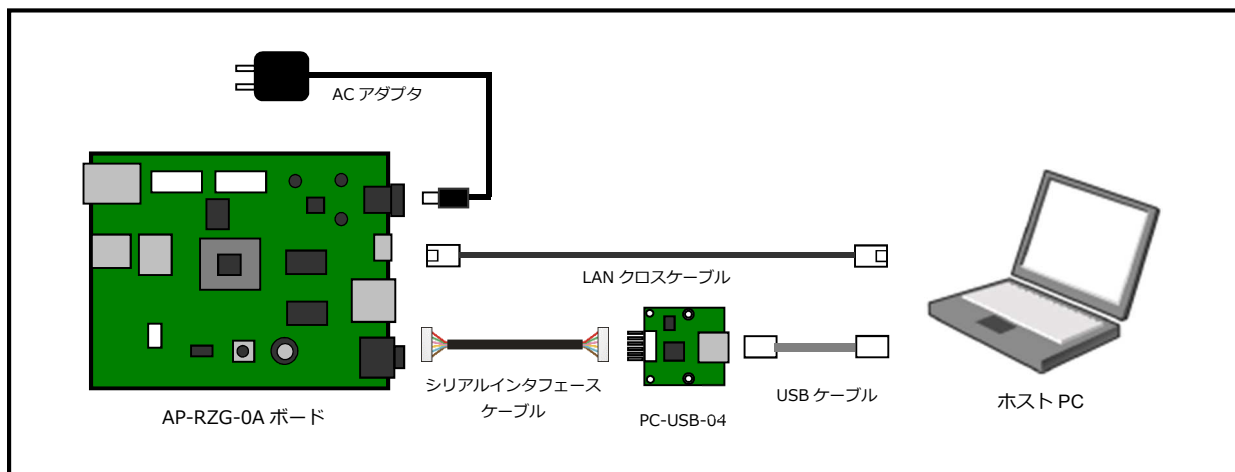


Fig 3.5-1 AP-RZG-0A ボードの接続 (PC に接続する場合)

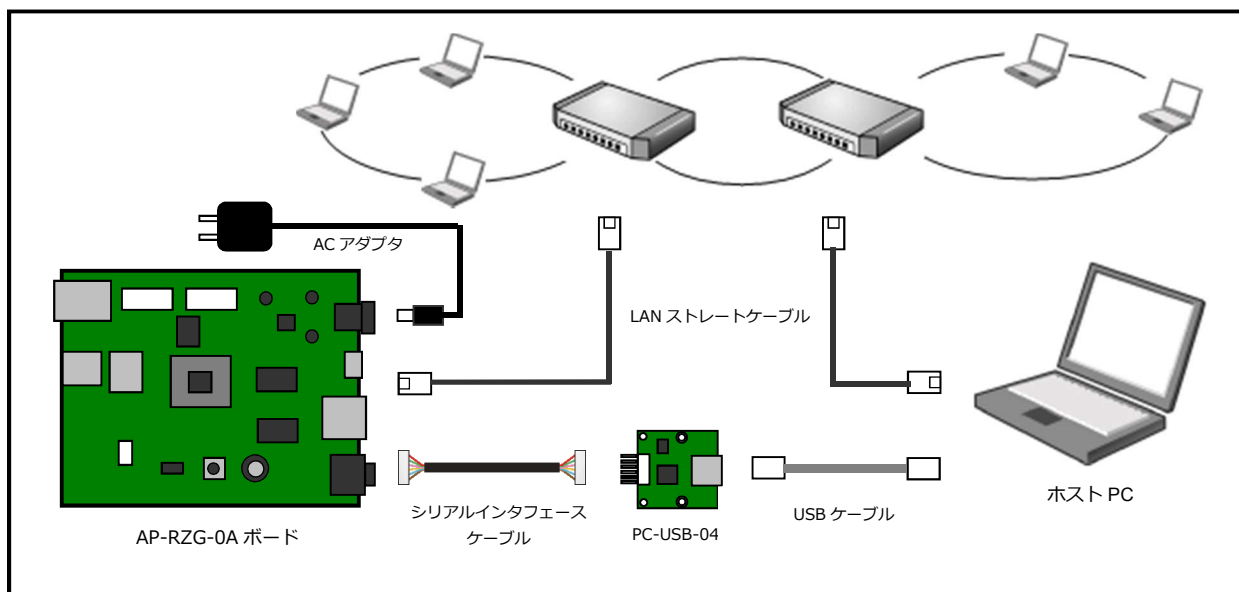


Fig 3.5-2 AP-RZG-0A ボードの接続 (HUB に接続する場合)

## 3.6 動作確認用 microSD カードの作成

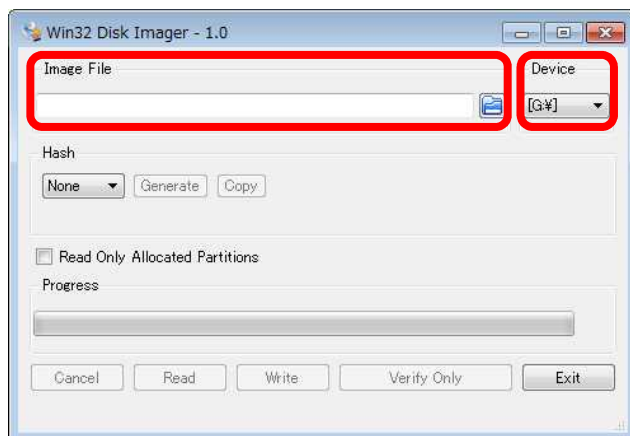
AP-RZG-0A は、動作確認用にプリビルドされた microSD 用の Linux ファイルシステムイメージファイルが、同梱の CD-ROM に用意されています。以下の手順にてそのイメージファイルを microSD カードに書き込みます。

また、作成手順では、『Win32 Disk Imager』のツールを使用する方法となりますので、事前にインストールをお願いします。

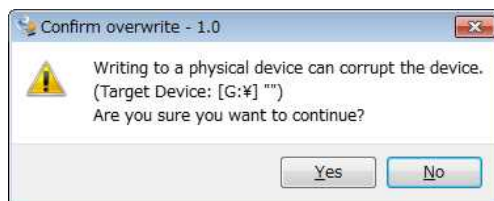
- ① 付属の CD-ROM 内に zip 圧縮ファイル『aprzg0a\_image.zip』のイメージデータがありますので、このファイルを解凍します。zip ファイルを解凍すると解凍先のフォルダに『aprzg0a\_image/aprzg0a\_sd.img』が作成されます。解凍後のファイルは、1GByte ありますので、ご注意ください。  
なお、以降の説明では、以下のフォルダに解凍されたとします。

c:/alphaproject/aprzg0a\_image/aprzg0a\_sd.img

- ② PC の microSD カードスロットに書き込む microSD カードを挿入します。
- ③ 『Win32 Disk Imager』を起動します。  
『Device』に microSD カードスロットのドライブが選択されていることを確認し、その左側のフォルダアイコンを押して、手順①で解凍したファイルを選択します。



- ④ 『Write』ボタンを押すと確認ダイアログが表示されますので、問題なければ『YES』ボタンを押します。

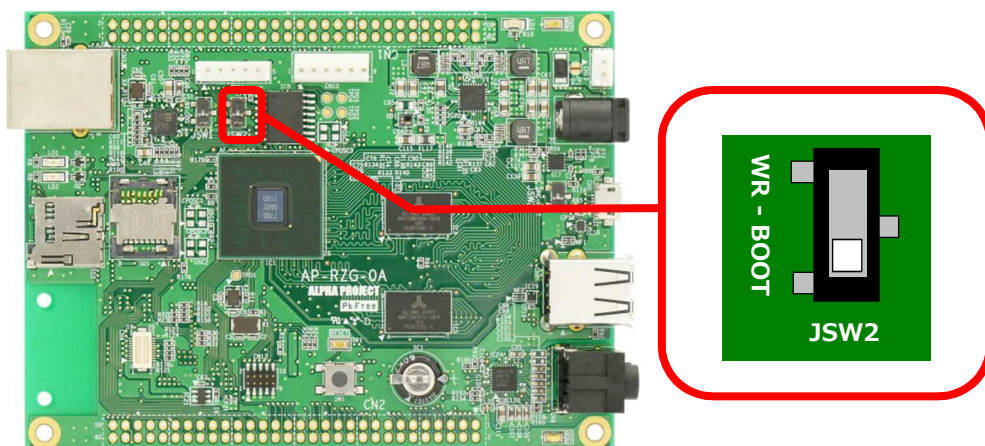


## 3.7 Linux の起動

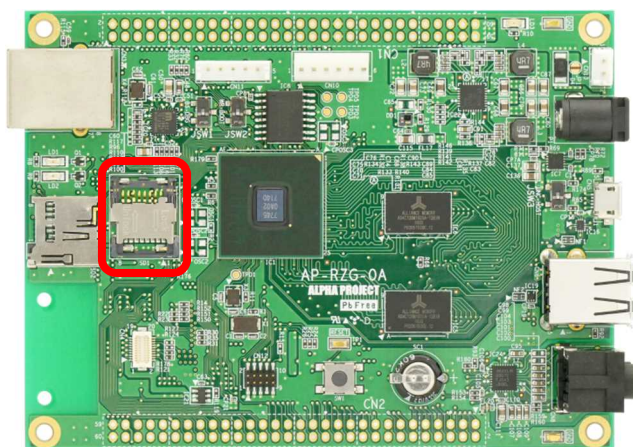
AP-RZG-0A 上で Linux の起動を行います。

『[3.6 動作確認用 microSD カードの作成](#)』にて作成した microSD カードを使用して、以下の手順にて Linux を起動します。

- ① AP-RZG-0A 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、『[AP-RZG-0A ハードウェアマニュアル](#)』でご確認ください。



- ② microSD カードを microSD カードスロット SD1 に挿入します。



- ③ 『[3.5 AP-RZG-0A ボードの接続](#)』にしたがって、ホスト PC と AP-RZG-0A を接続します。  
PC-USB-04 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 AC アダプタを接続して、AP-RZG-0A の電源を入れます。

## 3.8 Linux の動作確認

AP-RZG-0A 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『**aprzg0a 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 3.7-1 ログイン設定

```
Poky (Yocto Project Reference Distro) 1.6.1 aprzg0a /dev/ttySC10
aprzg0a login: root
```

### 時刻設定

AP-RZG-0A 上で時刻の設定をします。AP-RZG-0A 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
Mon Jul 24 06:49:09 2017 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 2017 年 10 月 11 日 12 時 13 分に設定します。

『**date -s '2017-10-11 12:13'**』実行後、『**hwclock -w**』を実行してください。

```
# date -s '2017-10-11 12:13'
Wed Oct 11 12:13:00 UTC 2017
# hwclock -w
```

## microSD カード

---

microSD カード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。

以下に、microSD カードの簡単な動作確認手順を記載します。

- ① microSD カードを SD2 コネクタに挿入する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。出力されるメッセージは環境により異なります。

```
mmc0: host does not support reading read-only switch. assuming write-enable.  
mmc0: new high speed SDHC card at address e624  
mmcblk0: mmc0:e624 SU04G 3.69 GiB  
mmcblk0: p1
```

- ② FAT ファイルシステムでフォーマットされている microSD カードを『/mnt』ディレクトリにマウントします。『mount -t vfat /dev/mmcblk0p1 /mnt』コマンドを実行してください。

```
# mount -t vfat /dev/mmcblk0p1 /mnt
```

- ③ 『ls』コマンドで内容を確認することができます。

```
# ls /mnt  
a.txt
```

- ④ 『umount』コマンドで microSD カードをアンマウント（マウント解除）することができます。microSD カードを抜く時は、必ずアンマウントを実行してください。

『umount /mnt』を実行してください。

```
# umount /mnt
```

## 3.9 ネットワークの設定

AP-RZG-0A 上での Linux のネットワーク設定を変更する方法および Web サーバへのアクセス、NFS の使用方法について説明します。

### ネットワーク設定の確認

ネットワーク設定を確認する方法について説明します。

- ① Linux の IP アドレス・サブネットマスクを確認するため、『ip addr』と入力してください。  
表示された『eth0』の項目内の『inet addr』が IP アドレス、アドレスビット数となります。

```
# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN group default qlen 10
   link/can
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
   qlen 1000
   link/ether XX:XX:XX:XX:XX:XX brd ff:ff:ff:ff:ff:ff
   inet 192.168.128.200/24 scope global eth0
       valid_lft forever preferred_lft forever
   RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

- ② 『netstat -nr』コマンドでゲートウェイの設定を確認することができます。  
下記で示している箇所が、デフォルトゲートウェイの設定です。

```
# netstat -nr
[Kernel IP routing table]
Destination    Gateway         Genmask         Flags         MSS Window  irtt Iface
0.0.0.0        192.168.128.254 0.0.0.0         UG            0 0        0 eth0
192.168.0.0    0.0.0.0        255.255.240.0   U             0 0        0 eth0
```

## ネットワーク設定の変更

ネットワーク設定を固定 IP にする方法について説明します。ネットワーク設定を変更する場合には、設定ファイル『`/var/lib/connman/wired.config`』を変更する必要があります。

- ① IP アドレス、サブネットマスク、ゲートウェイを変更するには『`/var/lib/connman/wired.config`』ファイルを編集します。

『`vi /var/lib/connman/wired.config`』を実行し、エディタを起動します。

```
# vi /var/lib/connman/wired.config
```

『`/var/lib/connman/wired.config`』ファイル

```
[global]
Name = Wired
Description = Wired network configuration

[service_ether]
Type = ethernet
IPv4 = 192.168.128.200/255.255.0/192.168.128.254
Nameservers = 192.168.128.1
```

- IPv4 の設定

IP アドレス、ネットマスク、ゲートウェイを設定します。

記述は、それぞれを『/』（スラッシュ）で区切って『`IPv4 = IP アドレス/ネットマスク/ゲートウェイ`』となります。

- Nameservers の設定

DNS サーバのアドレスを設定します。

記述は、『`Nameservers = アドレス`』となります。

複数設定する場合は、『/』（コンマ）で区切って『`Nameservers = アドレス 1,アドレス 2`』となります。

- ② ネットワーク設定を反映させます。

『`/etc/init.d/connman restart`』コマンドで再起動します。

```
# /etc/init.d/connman restart
Restarting Connection Manager
```



connman は、デフォルトで DHCP として動作します。

そのため、DHCP に設定する場合は、『`/var/lib/connman/wired.config`』ファイルを削除して再起動します。

## NFS

---

Linux カーネルの機能により NFS (Network File System : ネットワークを介した分散ファイルシステム) を利用することができます。NFS は利用すれば Linux 上にある共有ディレクトリ内のファイルを共有することができます。ゲスト OS (Ubuntu) 上の NFS 共有ディレクトリ 『/nfs』 を AP-RZG-0A からマウントします。



NFS を使用するには、ゲスト OS が起動し、NFS サーバを有効にする必要があります。

- ① NFS 共有ディレクトリをマウントするには『**mount -t nfs -o nolock NFS サーバ IP アドレス:共有ディレクトリ名 マウント先ディレクトリ**』と入力します。

『**mount -t nfs -o nolock 192.168.128.210:/nfs /mnt**』を実行してください。

```
# mount -t nfs -o nolock 192.168.128.210:/nfs /mnt ←入力
```

- ② 『**umount**』コマンドで NFS をアンマウント (マウント解除) することができます。

『**umount /mnt**』を実行してください。


```
# umount /mnt ←入力
```



## 3.10 Linux の終了

microSD カードのファイルを編集した等で microSD カードをアンマウントするには、以下の手順にて Linux を終了させます。

- ① 『halt』 コマンドで Linux を終了します。

```
# halt   
  
Broadcast message from root@aprzg0a (ttySC10) (Thu Oct 12 09:00:53 2017):  
  
The system is going down for system halt NOW!  
INIT: Sending processes the TERM signal  
INIT: * Stopping Avahi mDNS/DNS-SD Daemon: avahi-daemon  
... done.  
Stopping advanced power management daemon: no /usr/sbin/apmd found; none killed  
apmd.  
Stopping system message bus: dbus.  
Stopping syslogd/klogd: stopped syslogd (pid 315)  
stopped klogd (pid 318)  
done  
Stopping Telephony daemon  
ALSA: Storing mixer settings...  
Stopping rpcbind daemon...  
done.  
Stopping Linux NFC daemon  
Deconfiguring network interfaces... ifdown: interface eth0 not configured  
done.  
Sending all processes the TERM signal...  
Sending all processes the KILL signal...  
Unmounting remote filesystems...  
Deactivating swap...  
Unmounting local filesystems...  
System halted.
```



『System halted.』が表示されると正常に Linux が終了しています。

## 4. Linux システムの構築

### 4.1 Linux システムの概要

AP-RZG-0A 用 Linux システムは、Linux カーネルと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 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「busybox」が収録されています。

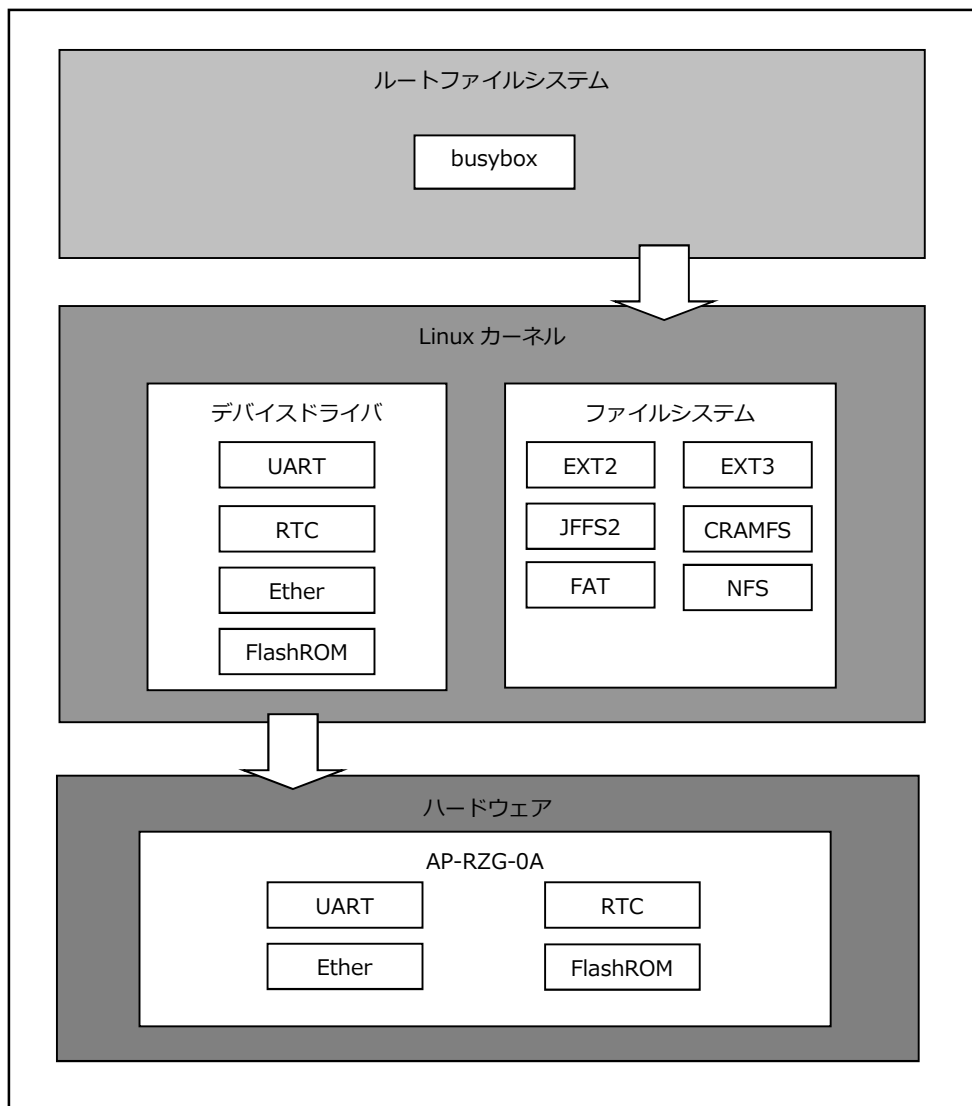


Fig 4.1-1 Linux システム

## 4.5 Yocto のビルド

Yocto Project 一式は、以下の手順でビルドします。

- ① ビルド環境の設定をします。

```
$ cd ~/aprzg0a
$ source poky/oe-init-build-env
You had no conf/local.conf file. This configuration file has therefore been
created for you with some default values. You may wish to edit it to use a
different MACHINE (target hardware) or enable parallel build options to take

<途中省略>

meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemu86'
```

環境設定が終了すると、カレントディレクトリは、『~/aprzg0a/build』に移動します。

- ② conf ファイルの準備をします。

ディレクトリ『../meta-renesas/meta-rzg1/templates/aprzg0a』には、2種類の conf ファイルが格納されたディレクトリがあります。

下記のコマンドでは、base となりますが、必要に応じて選択してコピーしてください。

(ディレクトリ内のファイル名は同じ名前になります。)

フォルダ名	内容
base	基本的なパッケージで作成する conf ファイル
x11	x11 ライブラリ等描画関連も含まれた conf ファイル

```
$ cp ../meta-renesas/meta-rzg1/templates/aprzg0a/base/bblayers.conf ../conf/
$ cp ../meta-renesas/meta-rzg1/templates/aprzg0a/base/local.conf ../conf/
```

- ③ イメージを作成します。

以下のコマンドも conf ファイル同様に必要に応じてビルドしてください。

コマンド名	内容
bitbake core-image-base	基本的なパッケージで作成するコマンド
bitbake core-image-x11	x11 ライブラリ等描画関連も含まれたコマンド

```
$ bitbake core-image-base
Parsing recipes: 100% |#####| Time: 00:02:37
Parsing of 1418 .bb files complete (0 cached, 1418 parsed). 1845 targets, 54 skipped, 17
masked, 0 errors.

<以下省略>
```

## 4.6 microSD カードの作成

ビルドして作成されたデータから microSD カードの作成方法を以下に記述します。

### microSD の作成の準備

- ① 作業用ディレクトリ『**aprzg0a\_bin**』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。

```
$ mkdir ~/aprzg0a_bin
```

- ② ディレクトリ『**aprzg0a\_bin**』に移動します。

```
$ cd ~/aprzg0a_bin
```

- ③ microSD カードに書込むファイルをコピーします。  
ここでのコピー元は『**4.5 Yocto のビルド**』で作成したファイルとします。

```
$ export OUTPUT_DIR=~/.aprzg0a/build/tmp/deploy/images/aprzg0a
$ cp $OUTPUT_DIR/uImage .
$ cp $OUTPUT_DIR/uImage-r8a7745-aprzg0a.dtb .
$ cp $OUTPUT_DIR/core-image-base-aprzg0a.tar.bz2 .
```

### microSD カードの作成

- ① 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 Host 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。)

```
$ sudo mke2fs -j /dev/sdb1
mke2fs 1.42 (29-Nov-2011)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
:
<途中省略>
:
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

## 4.7 カーネルのカスタマイズ

USB ファンクション/ホスト切替え、オプション製品（LCD キット、無線 LAN モジュール）等を使用する時は、カーネルのカスタマイズが必要となります。

ここでは、USB の設定を例に説明します。

なお、下記の手順では、本ドキュメントの『[4.5 Yocto のビルド](#)』で 1 度ビルドが終わっている環境が必要となります。

- ① ビルド環境の設定をします。

```
$ cd ~/aprzg0a
$ source poky/oe-init-build-env
```

環境設定が終了すると、カレントディレクトリは、『~/aprzg0a/build』に移動します。



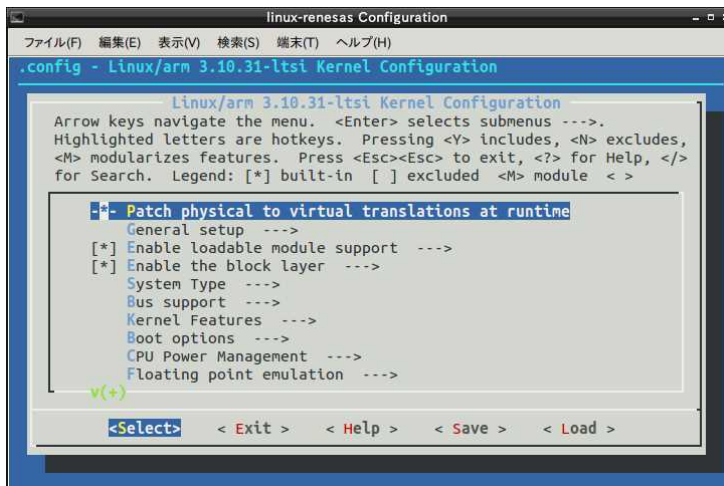
カーネルのコンフィギュレーションを初期化する場合は、以下のコマンド実行します。

```
bitbake -c configure linux-renesas --force
```

このコマンドを実行すると以前に行われた menuconfig による設定変更、およびドライバなどのソースの変更は全て初期化されます。

- ② カーネルのカスタマイズをするため、設定画面を開きます。

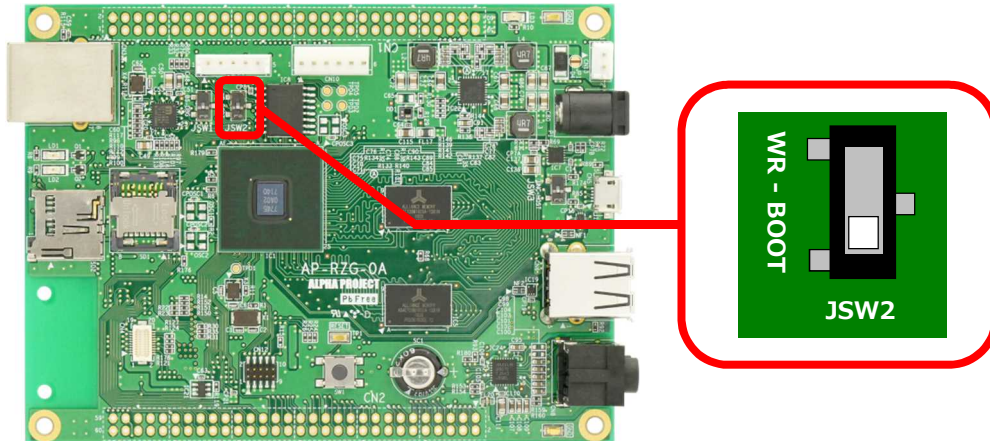
```
$ bitbake -c menuconfig linux-renesas
```



## 5.2 ブートローダの起動

AP-RZG-0A 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 AP-RZG-0A の電源を入れる前に、スイッチが以下になっていることを確認します。  
スイッチの各設定の詳細に関しては、『[AP-RZG-0A ハードウェアマニュアル](#)』でご確認ください。



- ② 『[3.5 AP-RZG-0A ボードの接続](#)』にしたがって、ホスト PC と AP-RZG-0A を接続します。  
PC-USB-04 がホスト 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、AP-RZG-0A の電源を入れます。

## 5.4 U-Boot の作成

ゲスト OS(Ubuntu)上で U-Boot をコンパイルするための手順を説明します。

なお、下記の手順では、本ドキュメントの『[4.5 Yocto のビルド](#)』で 1 度ビルドが終わっている環境が必要となります。

- ① ビルド環境の設定をします。

```
$ cd ~/aprzg0a
$ source poky/oe-init-build-env
```

環境設定が終了すると、カレントディレクトリは~/aprzg0a/build に移動します。

- ② u-boot のビルドをします。

```
$ bitbake -c configure u-boot --force
$ bitbake -c compile u-boot --force
```

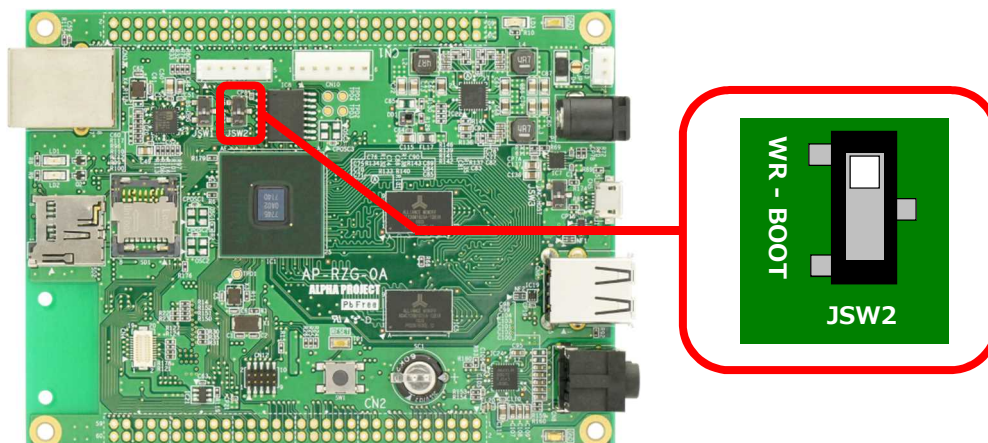
- ③ u-boot のビルドが成功しましたら、作成されたカーネルをデプロイ(配布)します。

```
$ bitbake -c deploy u-boot
```

## 5.5 U-Boot の書込み

以下に、作成した U-Boot を AP-RZG-0A 上の FlashROM に書き込む方法を説明します。

- ① AP-RZG-0A の電源を入れる前に、スイッチが以下のように JSW2 が『WR』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、『AP-RZG-0A ハードウェアマニュアル』でご確認ください。



- ② ターミナルアプリを起動します。  
ターミナル側から AP-RZG-0A にファイル転送しますので、ファイル転送の機能のあるターミナルソフトウェアを使用します。
- ③ AP-RZG-0A の電源を入れます。
- ④ ターミナルに『AP-RZG-0A MiniMonitor SPI\_BOOT』の文字が表示され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。

```
AP-RZG-0A SPI_LOADER (DDR1333) VX.XX XXXX.XX.XX
DEVICE MX25L3235E

AP-RZG-0A MiniMonitor SPI_BOOT
Work memory DRAM (H' 40200000-)
XXXX.XX.XX VerX.XX ** Program on DRAM (H' 40000000-) **
>
```

- ⑤ SPI フラッシュに書込むためのコマンドを入力します。

```
> ls 
Load Program to 64MB Spiflash memory (IC8:S25FL512S)
```

- ⑥ 画面の指示に従い JSW2 を BOOT 側に設定し『y』を入力します。

```
JSW2 BOOT-Side! Setting OK? (Push Y key) y 
```



## 6. プログラムの作成

本章では、AP-RZG-0A 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 AP-RZG-0A 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 AP-RZG-0A ボード上で動作を確認。

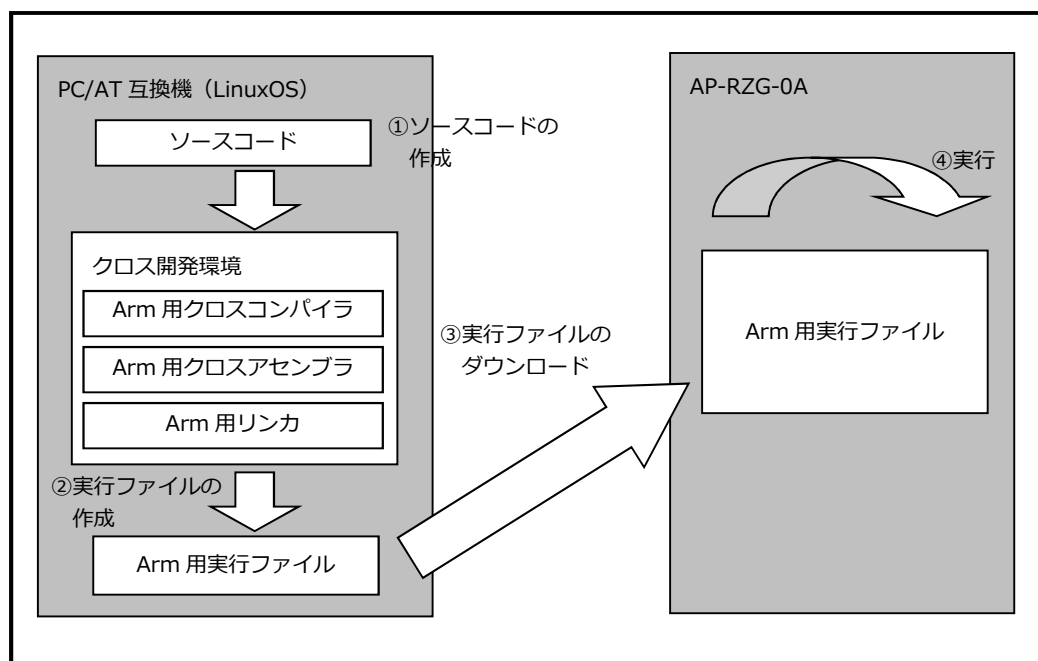


Fig 6.1-1 プログラムの開発手順

## サンプルアプリケーションのビルド

---

サンプルアプリケーションのビルド手順を説明します。

なお、下記の手順では、本ドキュメントの『[4.5 Yocto のビルド](#)』で 1 度ビルドが終わっている環境が必要となります。

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

```
$ cd ~/aprzg0a-app/helloworld
```

- ② サンプルアプリケーションをビルドします。

```
$ make  
arm-poky-linux-gnueabi-gcc -Wall helloworld.c -o helloworld
```

- 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。

```
$ cp helloworld /nfs
```

## 7. デバイスドライバの作成

本章では、AP-RZG-0A 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。

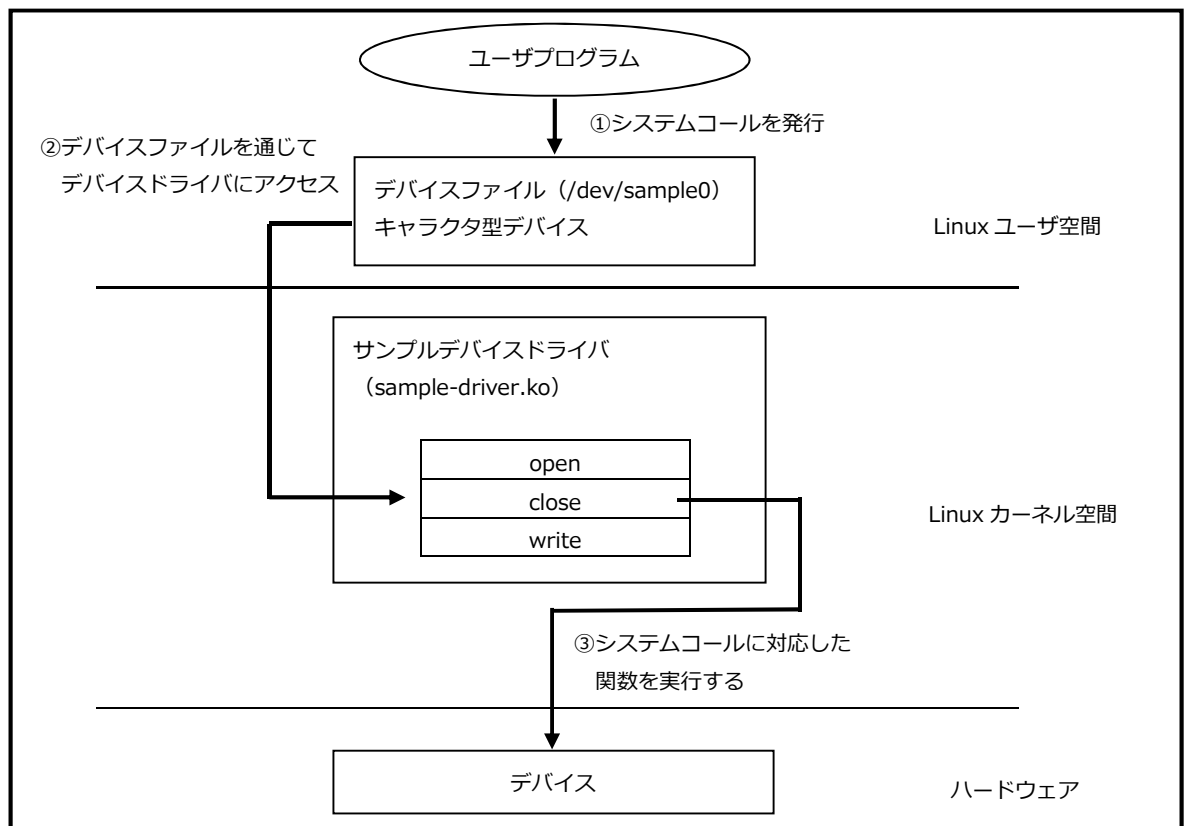


Fig 7.1-1 サンプルデバイスドライバの概要

## サンプルデバイスドライバのビルド

サンプルデバイスドライバのビルド手順を説明します。

なお、下記の手順では、本ドキュメントの『[4.5 Yocto のビルド](#)』で 1 度ビルドが終わっている環境が必要となります。

- ① 準備作業で展開した作業用ディレクトリの『`devicedriver/driver`』へ移動します。

```
$ cd ~/aprzg0a-app/devicedriver/driver
```

- ② 汎用デバイスドライバをビルドします。

```
$ make
make ARCH=arm CROSS_COMPILE=/home/guest/aprzg0a/build/tmp/sysroots/x86_64-linux/usr/bin/cortexa7hf-vfp-neon-poky-linux-gnueabi/arm-poky-linux-gnueabi- -C /home/guest/aprzg0a/build/tmp/work/aprzg0a-poky-linux-gnueabi/linux-renesas/3.10+git34547b2a5032ce6dca24b745d608d2f3baac187f-r0/git SUBDIRS=/home/guest/aprzg0a-app/devicedriver/driver modules
make[1]: ディレクトリ `/home/guest/aprzg0a/build/tmp/work/aprzg0a-poky-linux-gnueabi/linux-renesas/3.10+git34547b2a5032ce6dca24b745d608d2f3baac187f-r0/git' に入ります
CC [M] /home/guest/aprzg0a-app/devicedriver/driver/sample-driver.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/guest/aprzg0a-app/devicedriver/driver/sample-driver.mod.o
LD [M] /home/guest/aprzg0a-app/devicedriver/driver/sample-driver.ko
make[1]: ディレクトリ `/home/guest/aprzg0a/build/tmp/work/aprzg0a-poky-linux-gnueabi/linux-renesas/3.10+git34547b2a5032ce6dca24b745d608d2f3baac187f-r0/git' から出ます
```



KBUILD は、Linux カーネルのソースディレクトリを指定します。  
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。

- ③ 作成した汎用デバイスドライバを NFS の共有ディレクトリにコピーします。

```
$ cp sample-driver.ko /nfs
```

## サンプルアプリケーションのビルド

サンプルアプリケーションのビルド手順を説明します。

なお、下記の手順では、本ドキュメントの『[4.5 Yocto のビルド](#)』で 1 度ビルドが終わっている環境が必要となります。

- ① 準備作業で展開した作業用ディレクトリの『`devicedriver/application`』へ移動します。

```
$ cd ~/aprzg0a-app/devicedriver/application
```

- ② サンプルアプリケーションをビルドします。

```
$ make
/home/guest/aprzg0a/build/tmp/sysroots/x86_64-linux/usr/bin/cortexa7hf-vfp-neon-poky-linux-gnueabi/arm-poky-linux-gnueabi-gcc -Wall sample-app.c -o sample-app
```

- 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。

```
$ cp sample-app /nfs
```

## 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

- ・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。
- ・本文書の内容は、将来予告なしに変更されることがあります。
- ・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。万が一不審な点、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。

## 商標について

- ・R8A7745 は、ルネサスエレクトロニクス株式会社の登録商標、商標または商品名称です。
  - ・Linux は、Linus Torvalds の米国およびその他の国における登録商標または商標です。
  - ・Yocto Project は、Linux Foundation の登録商標です。
  - ・U-Boot は、DENX Software Engineering の登録商標、商標または商品名称です。
  - ・Windows®の正式名称は、Microsoft®Windows®Operating System です。
  - ・Microsoft、Windows 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®10、Windows®7 は、米国 Microsoft Corporation.の商品名称です。
  - ・VirtualBox は、OracleCorporation 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。ご了承下さい。
- Windows®10 は、Windows 10 もしくは Win10  
Windows®7 は、Windows 7 もしくは Win7
- ・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。