Alpha Board Series

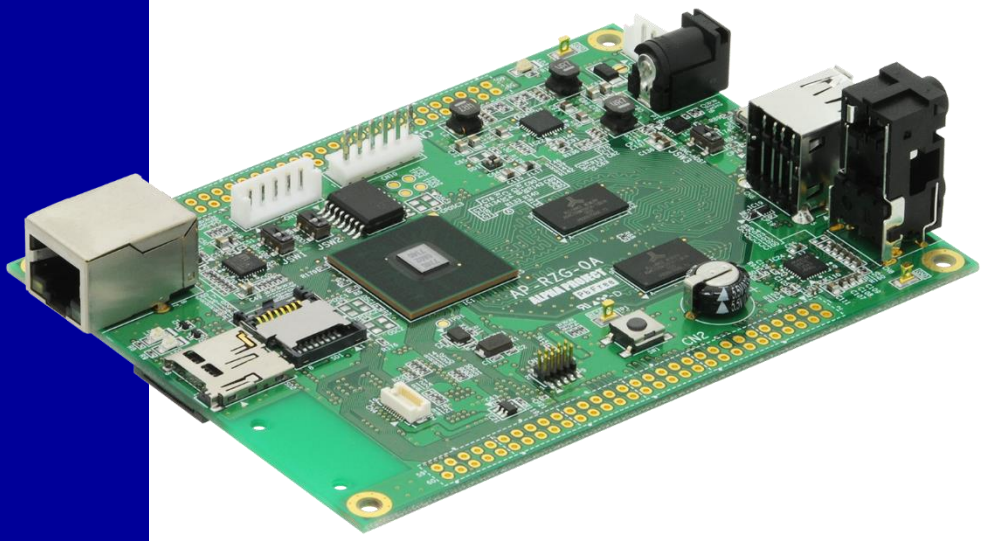
LK-RZG-A02

Cortex-A7 R8A7745 CPU BOARD

Software Manual

Rev 1.2

ダイジェスト版



ALPHA PROJECT
株式会社アルファプロジェクト

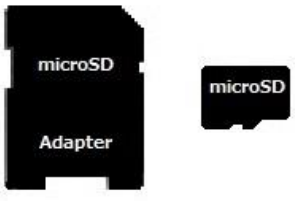
ご使用になる前に

このたびはAP-RZG-0A Linux開発キット(LK-RZG-A02)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-A02梱包内容

●LANストレートケーブル　1本 	●ACアダプタ　1本 
●PC-USB-04(ケーブル付)　1個 	●USBケーブル(A - B)　1本  コネクタの形状 
●microSDカード(4GB、アダプター付き)　1個 	●DVD-ROM　1枚 ●マニュアル・サンプルプログラムのダウンロード・保証のご案内　1枚

■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

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。

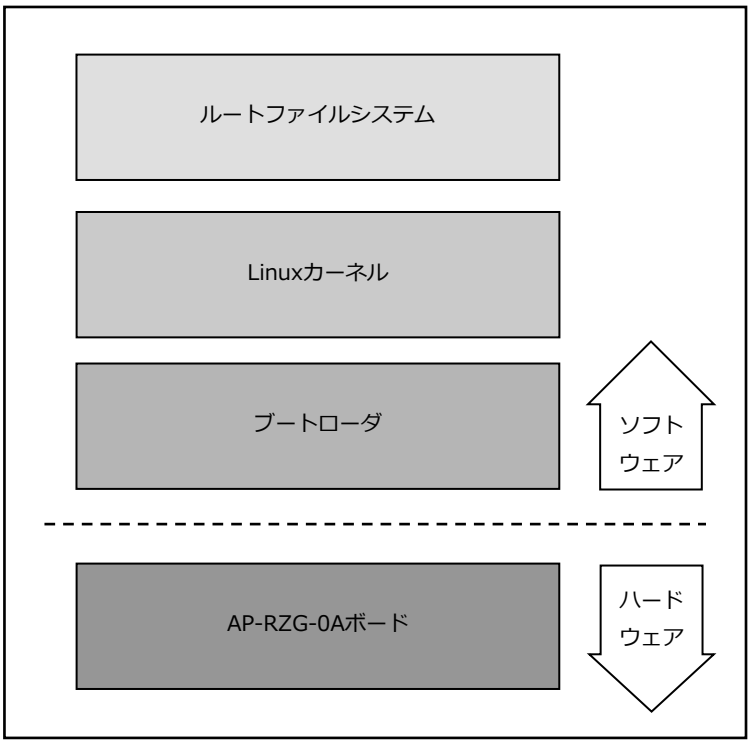


Fig 2.1-1　AP-RZG-0Aシステム概要図

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-A02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。

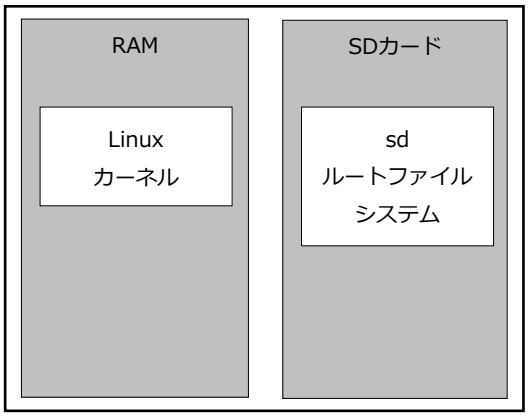


Fig 2.4-1 SD-Linuxシステム

2.6　添付DVD-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DVD-ROMから入手することができます。

```
LK_RZG_A02_VX_X
|-- an
|   |-- an1622.pdf                    : AN1622 タッチパネルLCDキットの使用方法
|   |-- an1623.pdf                    : AN1623 カメラモジュールの使用方法
|   |-- an1624.pdf                    : AN1624 無線LANモジュールの使用方法
|   `-- an1632.pdf                    : AN1632 無線LANモジュール(WM-RP-10)の使用方法
|-- binaries
|   |   `-- sd_image                  : microSDカードイメージ
|   |-- APRZG0A_MON_DRAM_SPI_VXXX_40000000.mot
|   |                                 : MiniMonitorのSレコード
|   |-- APRZG0A_SPI_LOADER_VXXX.mot   : LoaderのSレコード
|   |-- core-image-weston-aprzg0a.tar.bz2
|   |                                 : sdファイルシステムバイナリ
|   |-- helloworld                    : サンプルアプリ
|   |-- modules-aprzg0a.tgz           : ドライバモジュールバイナリ
|   |-- r8a7745-aprzg0a.dtb           : デバイスツリーバイナリ
|   |-- 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
|   |-- vscam01_test                  : カメラモジュール用サンプルアプリ(LCD-KIT-B01/C01用)
|   `-- vscam01_test_lcdkitd0x        : カメラモジュール用サンプルアプリ(LCD-KIT-D01/D02用)
|-- 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_a02_is.pdf             : Linux開発 インストールマニュアル
|   `-- lk_rzg_a02_sw.pdf             : Linux開発 ソフトウェアマニュアル
|-- 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-X.X.tar.bz2             : カメラモジュールアプリソース
|   `-- wmrp10-X.X.tar.bz2            : WM-RP-10サンプルアプリソース
`-- sources
    |-- APRZG0A_LOADER_VXXX.tar.bz2   : Loaderのソース
    |-- APRZG0A_SpiBoot_miniMonDram_VXXX.tar.bz2
    |                                 : MiniMonitorのソース
    `-- rzg_bsp_aprzg0a-X.X.tar.gz    : AP-RZG-0A用のレシピファイル
```

Table 2.6-1　DVD-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 10/11（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『**LK-RZG-A02 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。

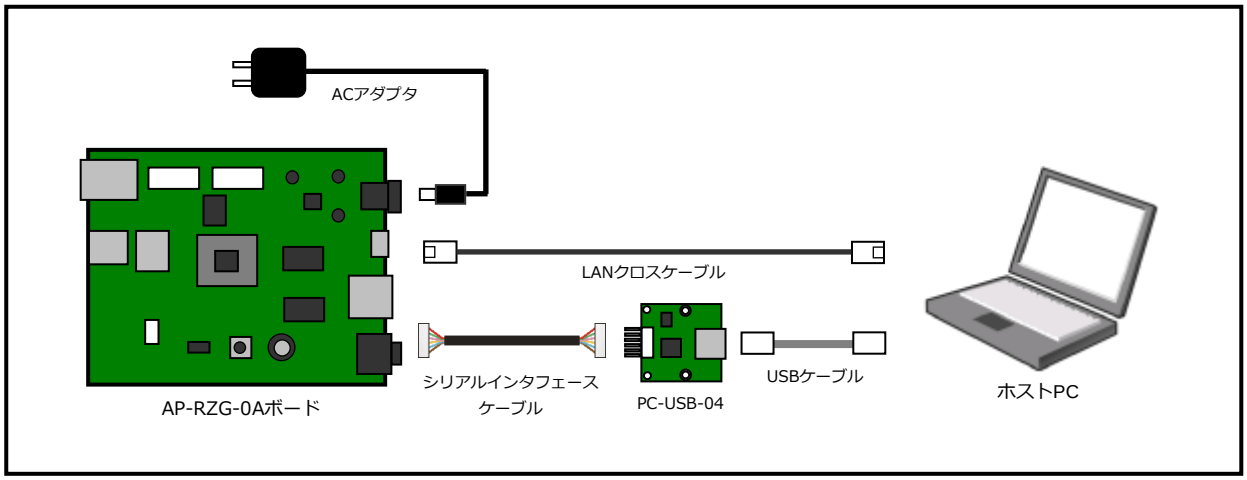


Fig 3.5-1 AP-RZG-0Aボードの接続（PCに接続する場合）

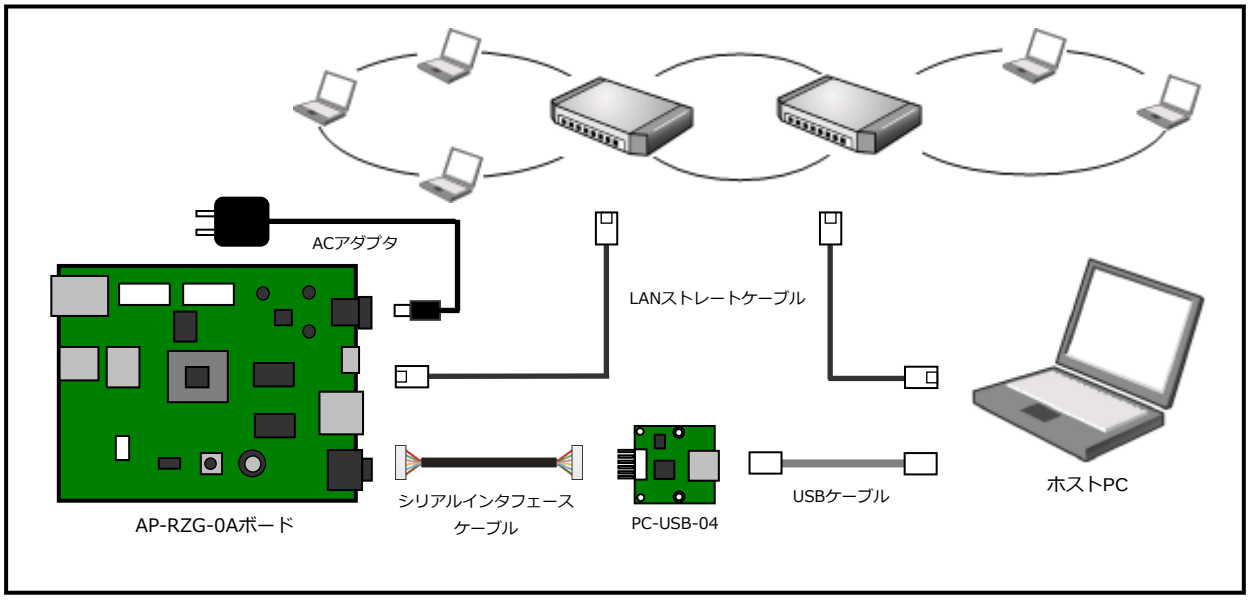


Fig 3.5-2 AP-RZG-0Aボードの接続（HUBに接続する場合）

3.6　動作確認用microSDカードの作成

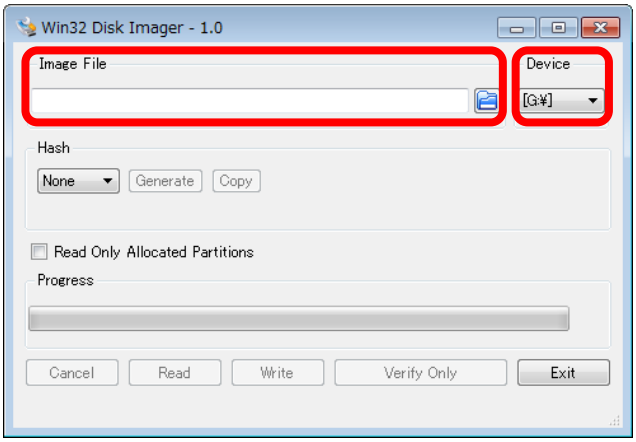
AP-RZG-0Aは、動作確認用にプリビルドされたmicroSD用のLinuxファイルシステムイメージファイルが、同梱のDVD-ROMに用意されています。以下の手順にてそのイメージファイルをmicroSDカードに書き込みます。
また、作成手順では、『**Win32 Disk Imager**』のツールを使用する方法となりますので、事前にインストールをお願いします。

① 付属のDVD-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。

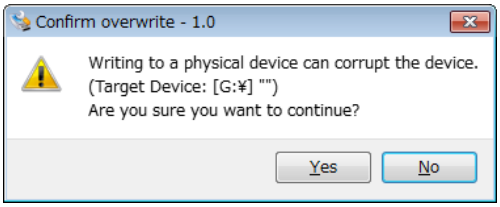
c:/alphaproject/aprzg0a_image/aprzg0a_sd.img

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。

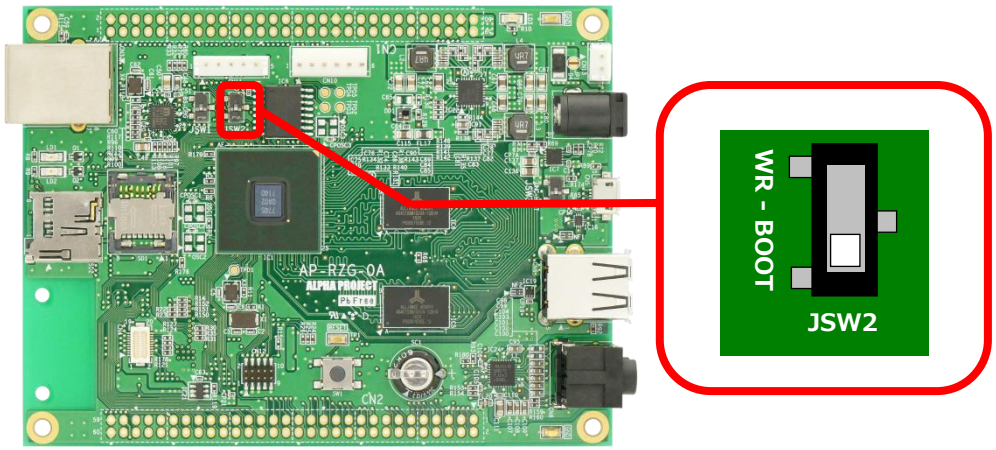


3.7 Linuxの起動

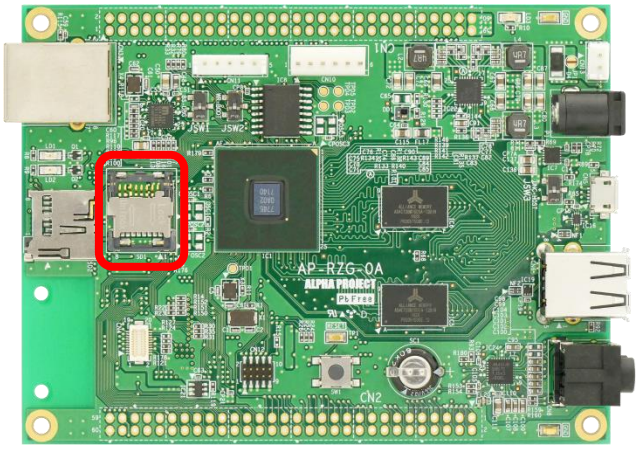
AP-RZG-0A上でLinuxの起動を行います。

『**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    入力
```

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    入力
Fri Mar 15 11:37:53 2019  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2019年03月15日11時37分に設定します。
『**date -s '2019-03-15 11:37'**』実行後、『**hwclock -w**』を実行してください。

```
# date -s '2019-03-15 11:37'    入力
Fri Mar 15 11:37:00 JST 2019
# hwclock -w    入力
```

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: new high speed SDHC card at address aaaa
mmcblk0: mmc0:XXXXXXXXXXX 14.8 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     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     入力
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     入力
```

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 address**』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 address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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
    inet6 ::1/128 scope host
       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 brd 192.168.128.255 scope global eth0
       valid_lft forever preferred_lft forever
4: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1
    link/sit 0.0.0.0 brd 0.0.0.0
```

入力
IPアドレス
アドレスビット数

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         255.255.255.0   U         0 0          0 eth0
```

入力
ゲートウェイ

③ DNSサーバのアドレスを確認するには『**/etc/resolv.conf**』ファイルで行います。
『**cat /etc/resolv.conf**』を実行して確認します。

```
# cat /etc/resolv.conf
nameserver 192.168.128.254
```

入力
DNSサーバアドレス

ネットワーク設定の変更

ネットワーク設定を変更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『**/etc/network/interfaces**』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『**/etc/network/interfaces**』ファイルを編集します。
『**vi /etc/network/interfaces**』を実行し、エディタを起動します。

```
# vi /etc/network/interfaces
```
入力

編集内容に関しては、DHCP設定か固定IP設定によって編集します。以下にそれぞれの記述例を記載します。

『**/etc/network/interfaces**』ファイル（DHCP設定）

```
# The loopback interface
auto lo
iface lo inet loopback

# Wired or wireless interfaces
auto eth0
iface eth0 inet dhcp
```

『**/etc/network/interfaces**』ファイル（固定IP設定）

```
# The loopback interface
auto lo
iface lo inet loopback

# Wired or wireless interfaces
auto eth0
iface eth0 inet static
address 192.168.128.200
netmask 255.255.255.0
gateway 192.168.128.254
```

●IPアドレスの設定
IPアドレスの設定は、『**address IPアドレス**』で設定します。

●サブネットマスクの設定
サブネットマスクの設定は、『**netmask サブネットマスク**』で設定します。

●ゲートウェイの設定
ゲートウェイの設定は、『**gateway ゲートウェイアドレス**』で設定します。

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    入力

Broadcast message from root@aprzg0a (ttySC0) (XXX XXX XX XX:XX:XX XXXX):

The system is going down for system halt NOW!
INIT: Sending processes the TERM signal
Stopping Weston
Stopping ohci-pci
rmmod: ERROR: Module ohci_pci is not currently loaded
Stopping ohci-hcd
rmmod: ERROR: Module ohci_hcd is not currently loaded
Stopping Dropbear SSH server: stopped /usr/sbin/dropbear (pid 1358)
dropbear.
 * Stopping Avahi mDNS/DNS-SD Daemon: avahi-daemon                    [ ok ]
Stopping system message bus: dbus.
Framebuffer /dev/fb0 not detected
Boot splashscreen disabled
Stoping PulseAudio
stopping statd: done
usbcore: deregistering interface driver rtl8192cu
Stopping rpcbind daemon...
done.
rmmod: ERROR: Module dc_linuxfb is not currently loaded
Warning: Could not unload dc_linuxfb
SGX quirk clk is disabled
Unloaded PowerVR consumer services.
Deconfiguring network interfaces... done.
Sending all processes the TERM signal...
Sending all processes the KILL signal...
Unmounting remote filesystems...
Deactivating swap...
Unmounting local filesystems...
EXT4-fs (mmcblk1p1): re-mounted. Opts: (null)
reboot: 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EXT3、EXT4、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。

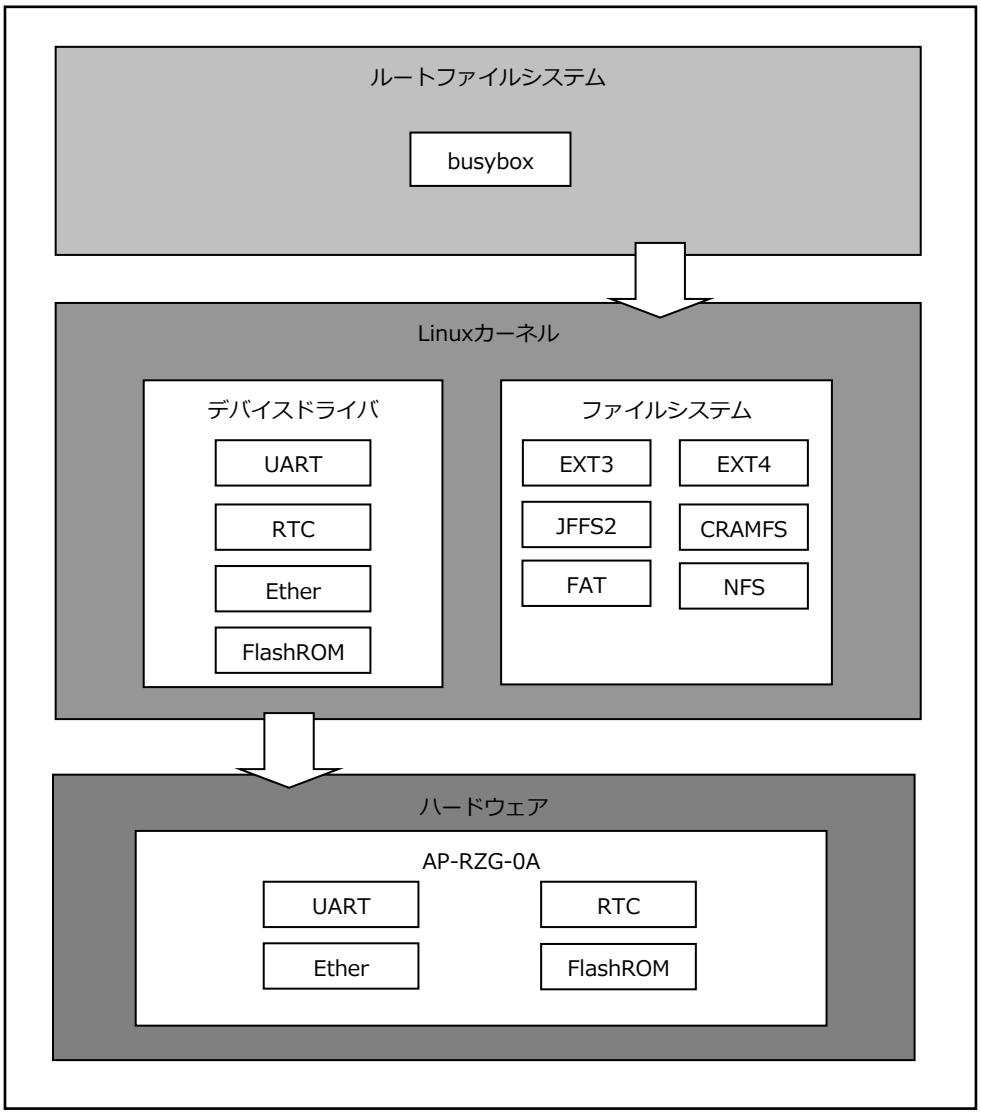


Fig 4.1-1 Linuxシステム

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 $WORK                                    入力
$ source poky/oe-init-build-env               入力
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, for
example, select a different MACHINE (target hardware). See conf/local.conf

          :
     <途中省略>
          :

    meta-toolchain
    meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemux86'
```

『**source poky/oe-init-build-env**』コマンドは、各ターミナル毎に行う必要があります。
同一のターミナルで2回実行する必要はありませんが、別のターミナルを起動した場合には、
再度実行する必要があります。

② confファイルの準備をします。

```
$ cd $WORK/build/conf                         入力
$ cp ../../meta-rzg-demos/meta-rzg1/qt-hmi-demo/template/aprzg0a/*.conf ./    入力
```

③ イメージを作成します。

```
$ cd $WORK/build                              入力
$ bitbake core-image-weston                   入力
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.

     <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。

```
$ mkdir ~/aprzg0a_bin
```
入力

② ディレクトリ『**aprzg0a_bin**』に移動します。

```
$ cd ~/aprzg0a_bin
```
入力

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=~/user_work/build/tmp/deploy/images/aprzg0a
$ cp $OUTPUT_DIR/uImage .
$ cp $OUTPUT_DIR/r8a7745-aprzg0a.dtb .
$ cp $OUTPUT_DIR/modules-aprzg0a.tgz .
$ cp $OUTPUT_DIR/core-image-weston-aprzg0a.tar.bz2 .
```
入力

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。

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 1.42.13 (17-May-2015)
/dev/sdb1 contains a vfat file system
Proceed anyway? (y,n) y
Creating filesystem with 3888512 4k blocks and 972944 inodes

        :
    ＜途中省略＞
        :

Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```
入力

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。

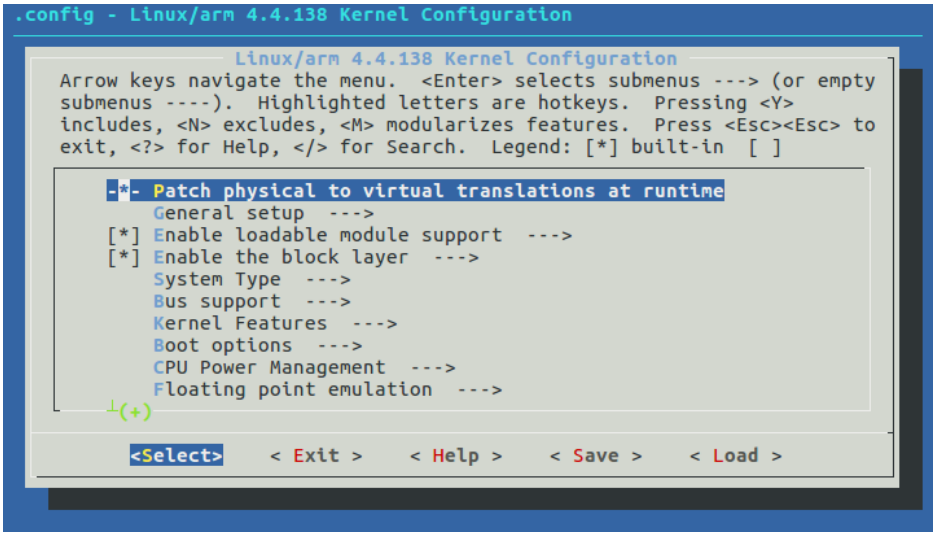
```
$ cd ~/user_work
$ source poky/oe-init-build-env
```

環境設定が終了すると、カレントディレクトリは、『~/user_work/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。

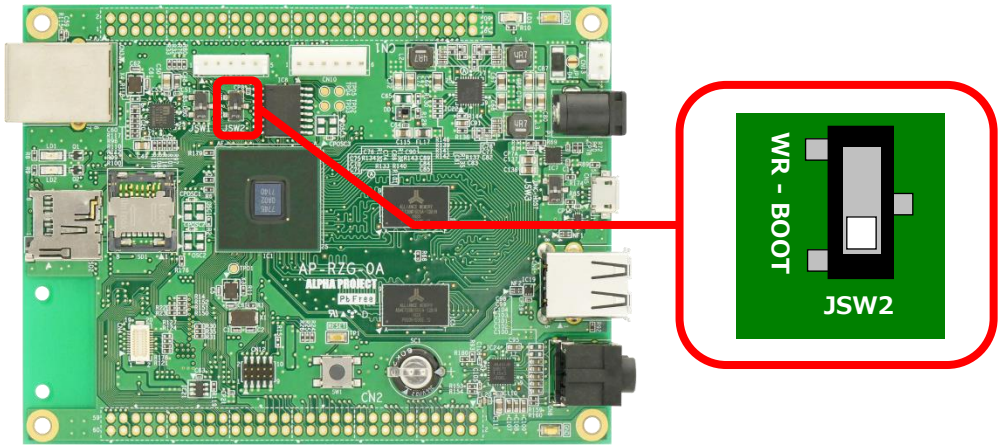
```
$ bitbake -c menuconfig linux-renesas
```



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、スイッチが以下のよう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 ~/user_work          入力
$ source poky/oe-init-build-env     入力
```

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

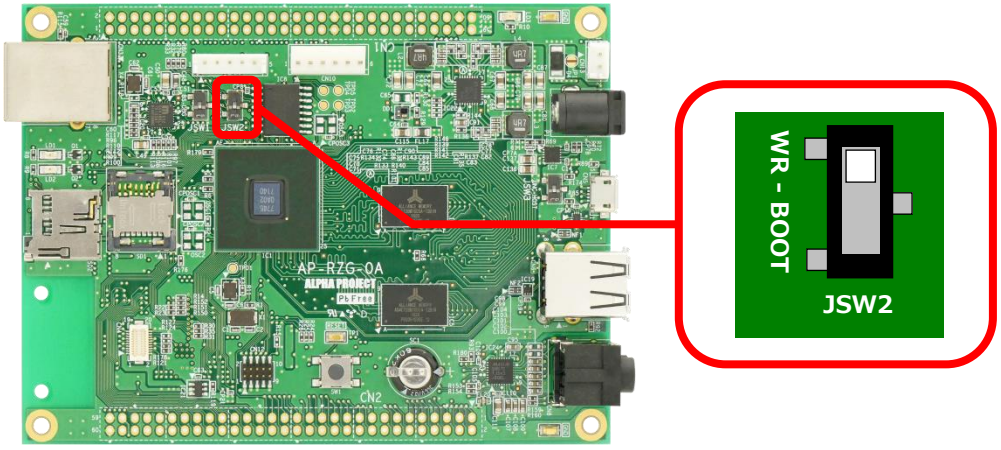
③ u-bootのビルドが成功しましたら、作成されたカーネルをデプロイ(配布)します。

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

5.5 U-Bootの書込み

U-Bootの書き込みは、U-Bootが書き込まれているのとは別のFlashROMに書き込み用プログラム(MiniMonitor)が書き込まれており、そのプログラムを使用して書き込みます。
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    入力
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    入力
```

6. アプリケーションの開発環境

本章では、AP-RZG-0A上で動作するアプリケーションの開発環境について説明します。

6.1 アプリケーションの開発について

Yocto Projectでは、アプリケーションを開発する場合に、まず、SDK（Software Development Kit）を作成します。
そのSDK内にクロス開発環境があり、その開発環境を使用して、以下のような一連の流れで開発します。

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。

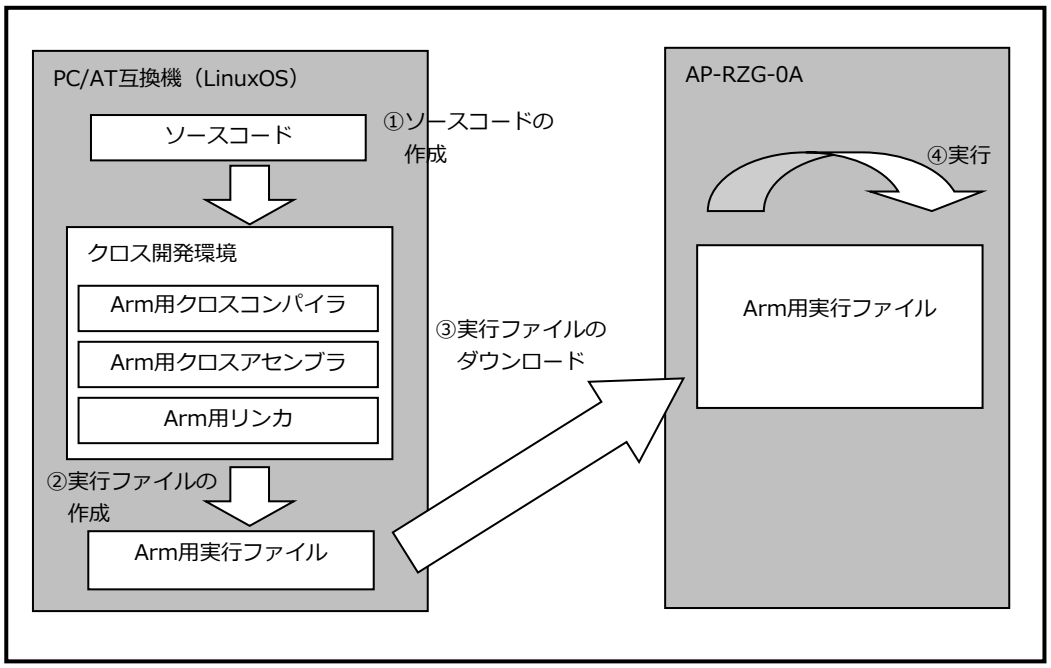


Fig 6.1-1 アプリケーションの開発手順

以降では、SDKの作成とインストールに関して説明します。

6.2 SDKの作成

SDKの作成方法を説明します。

① ビルド環境の設定をします。

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

『source poky/oe-init-build-env』コマンドは、各ターミナル毎に行う必要があります。
同一のターミナルで2回実行する必要はありませんが、別のターミナルを起動した場合には、
再度実行する必要があります。

② 環境設定が終了すると、カレントディレクトリは、『~/user_work/build』に移動しますので、bitbakeコマンドでSDKを作成します。

```
$ bitbake core-image-weston-sdk -c populate_sdk
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-renesas/meta-rzg1/recipes-qt/packagegroups/pack
agegroup-qt5-examples.bb: base_contains is deprecated, please use bb.utils.contains inste
ad.
NOTE: /home/guest/user_work/build/../meta-renesas/meta-rzg1/recipes-qt/packagegroups/pack
agegroup-qt5-examples.bb: base_contains is deprecated, please use bb.utils.contains inste
ad.
NOTE: /home/guest/user_work/build/../meta-renesas/meta-rzg1/recipes-bsp/u-boot/u-boot_201
3.01.01.bb: base_contains is deprecated, please use bb.utils.contains instead.
Parsing recipes: 100% |##########################################| Time: 0:01:47
Parsing of 2181 .bb files complete (0 cached, 2181 parsed). 3036 targets, 268 skipped, 7
masked, 0 errors.

     <以下省略>
```

⑥ DVD-ROMをアンマウントします。

```
$ umount /dev/sr0
```
入力

⑦ サンプルソースを展開します。

```
$ tar -xjpf helloworld-X.X.tar.bz2
```
入力

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、アプリケーションのビルドには、SDKが必要となります。そのため、本ドキュメントの『**6.3 SDKのインストール**』が行われている環境が必要となります。

① SDKの環境を設定します。

```
$ . /opt/poky/2.4.2/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```
入力

最初の『**.**』と『**/opt/poky/2.4.2/envir...**』の間には、半角スペースが必要ですので、ご注意ください。

『**. /opt/poky/2.4.2/envir...**』の環境設定コマンドは、各ターミナル毎に行う必要があります。
同一のターミナルで2回実行する必要はありませんが、別のターミナルを起動した場合には、再度実行する必要があります。

SDKの環境設定『**. /opt/poky/2.4.2/envir...**』とYoctoビルドの環境設定『**source poky/oe...**』は、同じターミナルで行うことができません。もし、Yoctoビルドの環境設定がターミナルで実行されている場合には、別のターミナルを起動して行う必要があります。

② 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
```
入力

③ 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  -march=armv7ve -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=/opt/poky/2.4.2/sysroots/cortexa7hf-neon-poky-linux-gnueabi  -O2 -pipe -g -feliminate-unused-debug-types  -Wall  -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed  helloworld.c -o helloworld
```
入力

④ 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
```
入力

8. サンプルデバイスドライバ

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。

本章の作業を行うには、ドライバおよびアプリケーションの開発環境が必要となります。
それぞれの開発環境は、『**4.5 Yoctoのビルド**』および『**6.3 SDKのインストール**』にてご確認ください。

8.1 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。

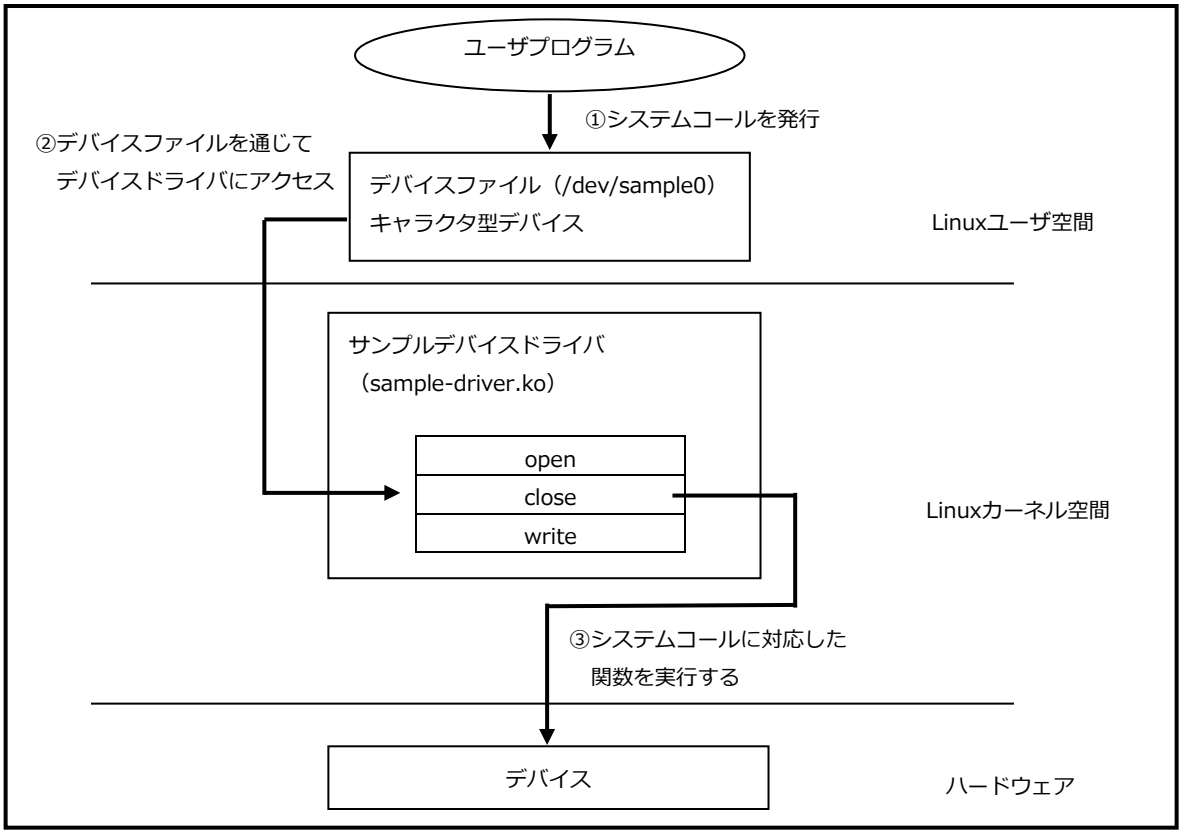


Fig 8.1-1 サンプルデバイスドライバの概要

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 ビルド環境の設定をします。

```
$ cd $WORK                                    入力
$ source poky/oe-init-build-env               入力
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, for
example, select a different MACHINE (target hardware). See conf/local.conf

          :
    ＜途中省略＞
          :

   meta-toolchain
   meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemux86'
```

『**source poky/oe-init-build-env**』コマンドは、各ターミナル毎に行う必要があります。
同一のターミナルで2回実行する必要はありませんが、別のターミナルを起動した場合には、再度実行する必要があります。

Yoctoビルドの環境設定『**source poky/oe...**』とSDKの環境設定『**. /opt/poky/2.4.2/envir...**』は、同じターミナルで行うことができません。もし、すでにSDKの環境設定がターミナルで実行されている場合には、別のターミナルを起動して行う必要があります。

② デバイスドライバ作成用のターミナルを開きます。

```
$ bitbake -c devshell virtual/kernel          入力
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.
NOTE: /home/guest/user_work/build/../meta-qt5/recipes-qt/qt5/qtbase_git.bb: base_contains
 is deprecated, please use bb.utils.contains instead.

    ＜以下省略＞
```

デバイスドライバ作成用のターミナルのコマンドプロンプトは、『**#**』となりますが、マニュアル表記上は、ゲストOS(Ubuntu)での操作となりますので、『**$**』にて表記します。

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、アプリケーションのビルドには、SDKが必要となります。そのため、本ドキュメントの『**6.3　SDKのインストール**』が行われている環境が必要となります。

① SDKの環境を設定します。

```
$ . /opt/poky/2.4.2/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```
入力

最初の『**.**』と『**/opt/poky/2.4.2/envir...**』の間には、半角スペースが必要ですので、ご注意ください。

『**. /opt/poky/2.4.2/envir...**』の環境設定コマンドは、各ターミナル毎に行う必要があります。
同一のターミナルで2回実行する必要はありませんが、別のターミナルを起動した場合には、再度実行する必要があります。

SDKの環境設定『**. /opt/poky/2.4.2/envir...**』とYoctoビルドの環境設定『**source poky/oe...**』は、同じターミナルで行うことができません。もし、Yoctoビルドの環境設定がターミナルで実行されている場合には、別のターミナルを起動して行う必要があります。

② 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
```
入力

③ 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  -march=armv7ve -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a7 --s
ysroot=/opt/poky/2.4.2/sysroots/cortexa7hf-neon-poky-linux-gnueabi  -O2 -pipe -g -felimina
te-unused-debug-types  -Wall  -Wl,-O1 -Wl,--hash-style=gnu -Wl,--as-needed  sample-app.c
 -o sample-app
```
入力

④ 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
```
入力

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