Alpha Board Series

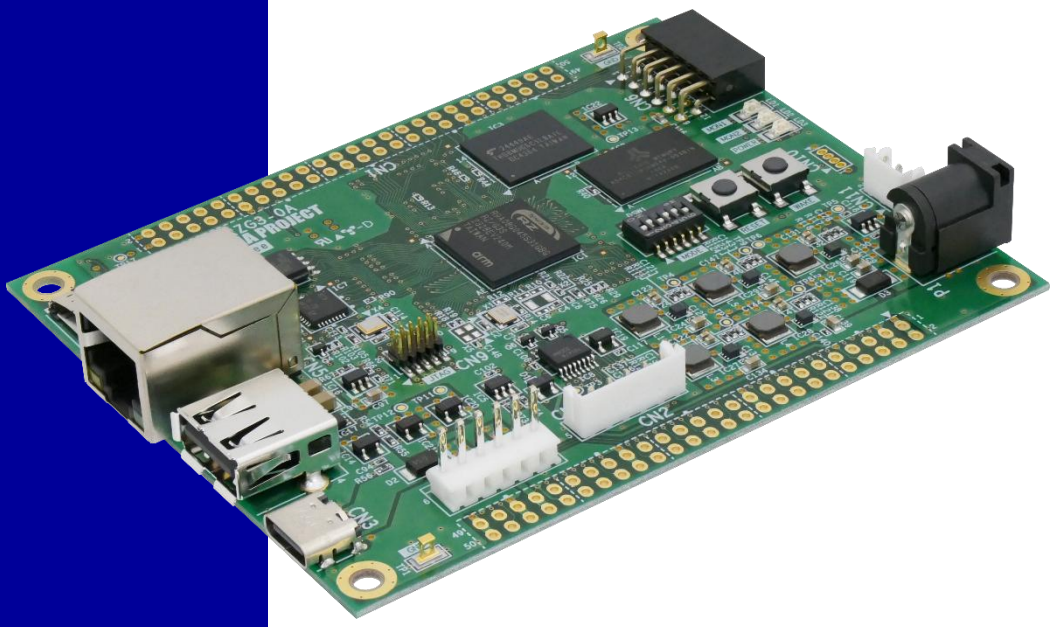
LK-RZG3-A01

RZファミリ RZ/G3S CPUボード

LinuxBSP Startup Guide

Rev 1.0

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

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、正しくお使いください。
今後とも、弊社製品をご愛顧賜りますよう宜しくお願い申し上げます。

LK-RZG3-A01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　1個 	●CD-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-RZG3-0Aの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。

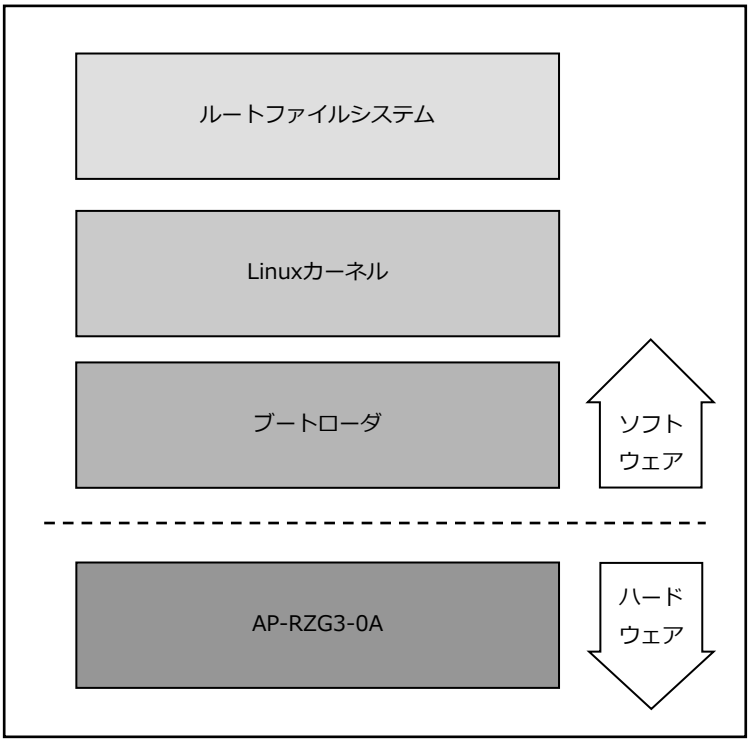


Fig 2.1-1 AP-RZG3-0Aシステム概要図

2.4 ルートファイルシステム

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。
その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（ext3やext4）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。
本製品では、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。
- eMMCルートファイルシステム　eMMC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eMMC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。
  eMMCルートファイルシステムは開発用LinuxPCにて直接書込みすることはできないため、SDルートファイルシステムにてLinuxを起動し、eMMCに書き込みます。

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、eMMCルートファイルシステムを利用したLinuxシステムをeMMC-Linuxシステムと表現します。

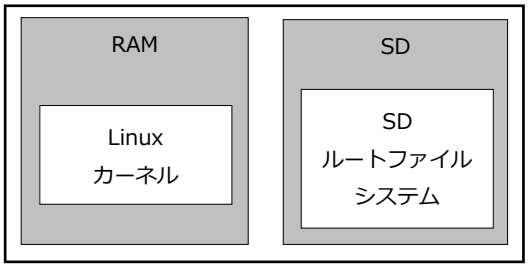


Fig 2.4-1 SD-Linuxシステム

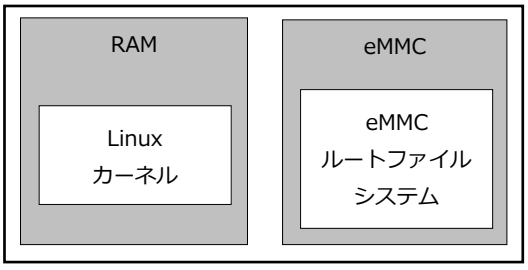


Fig 2.4-2 eMMC-Linuxシステム

2.6 添付CD-ROMの構成

AP-RZG3-0AのLinux開発に必要なファイルは、添付CD-ROMから入手することができます。

```
LK_RZG3_A01_VX_X
|-- an
|   `-- an1669.pdf                    : AN1669 Pmodの使用方法
|-- manual
|   |-- lk_rzg3_a01_inst.pdf          : 開発環境インストールガイド
|   |-- lk_rzg3_a01_startup.pdf       : LinuxBSPスタートアップガイド
|   `-- yocto_component_list_X_X.pdf  : Yoctoコンポーネントリスト
|-- sample
|   |-- devicedriver-X.X.tar.bz2      : サンプルデバイスドライバ
|   |-- helloworld-X.X.tar.bz2        : サンプルアプリケーション
|   `-- pmod-X.X.tar.bz2              : Pmodサンプル
`-- sources
    `-- bsp_aprzg30a_X.X.tar.gz       : AP-RZG3-0A用のレシピファイル
```

※『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。

また、以下の弊社ウェブサイト及び関連リンクからダウンロードにより入手することもできます。

LK-RZG3-A01の製品ページ
https://www.apnet.co.jp/product/rza/lk-rzg3-a01.html

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、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。
  PC-USB-04を使用すると、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、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
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	PC-USB-04が接続するポート (AP-RZG3-0AをUSB Peripheralとして接続する場合は、+1ポート)
	LANポート	100/1000BASE-TX　1ポート
PC-USB-04及びケーブル		ホストPCとAP-RZG3-0Aのシリアル接続用に使用
LANケーブル		ホストPCとの接続に使用
各種動作確認機器		USBメモリ等 動作確認に必要な機材
電源		ACアダプタ（DC5V±5%）

Table 3.1-1　動作環境

上記の環境は、AP-RZG3-0AのLinuxの動作確認をするためのものです。
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-RZG3-A01 Development Environment Install Guide** 』でご確認ください。

3.6 AP-RZG3-0Aの接続

ホストPCとAP-RZG3-0A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、設定に注意して行ってください。

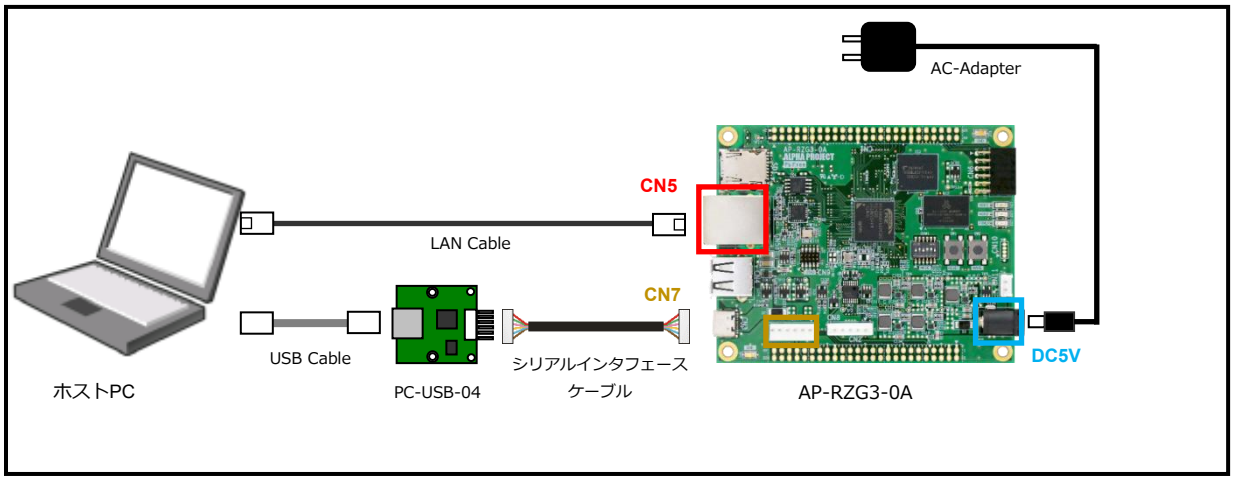


Fig 3.6-1 AP-RZG3-0Aの接続（PCに接続する場合）

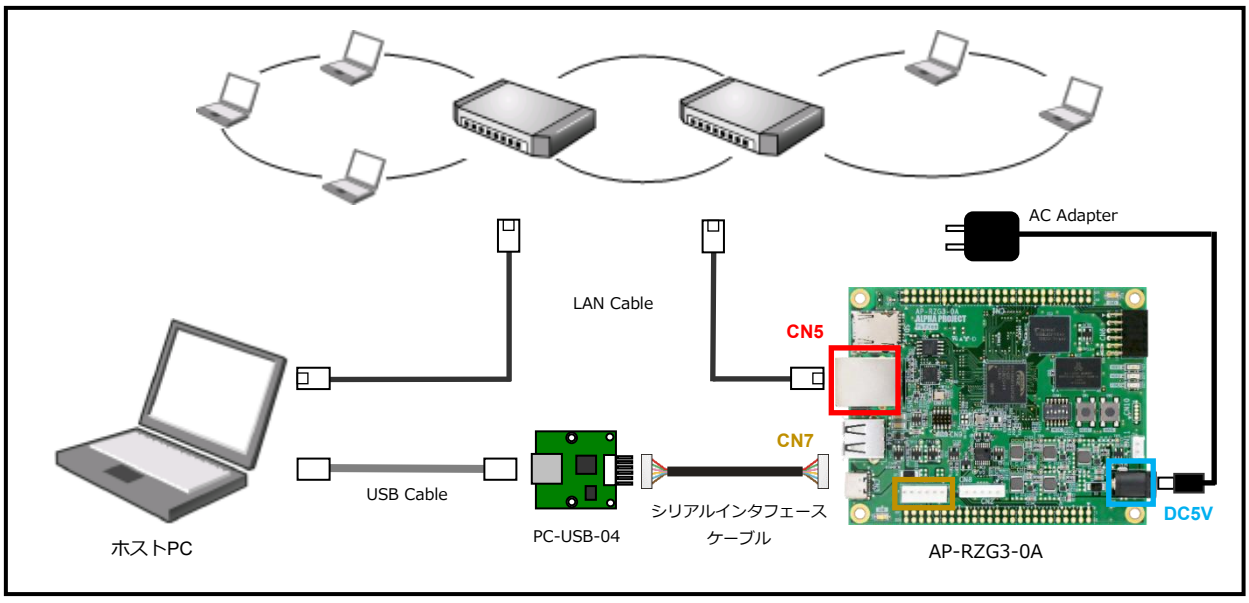
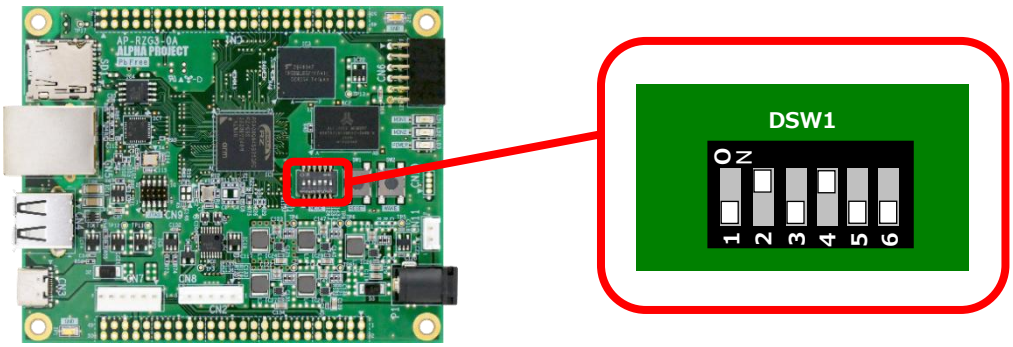


Fig 3.6-2 AP-RZG3-0Aの接続（HUBに接続する場合）

3.7 Linuxの起動

AP-RZG3-0A上でLinuxの起動を行います。

① 電源を入れる前に、AP-RZG3-0Aの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-RZG3-0Aのハードウェアマニュアルでご確認ください。



② 『**3.6 AP-RZG3-0Aの接続**』にしたがって、ホストPCとAP-RZG3-0Aを接続します。
電源はまだ入れないでください。
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-RZG3-0Aの電源を入れます。

⑤ Linuxカーネルが起動します。（全ての起動までには20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。

```
NOTICE:  BL2: v2.7(release):cca4398
NOTICE:  BL2: Built : 16:49:46, Jan 30 2025
NOTICE:  BL2: Resume(vbat)
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.7(release):cca4398
NOTICE:  BL31: Built : 16:49:46, Jan 30 2025


U-Boot 2021.10 (Mar 10 2025 - 04:19:20 +0000)

CPU:   Renesas Electronics K rev 15.0
Model: rzg3s-aprzg30a
DRAM:  896 MiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from MMC... *** Warning - bad CRC, using default environment

In:    serial@1004b800
Out:   serial@1004b800
Err:   serial@1004b800
Net:   eth0: ethernet@11c30000
Hit any key to stop autoboot:  0
```

3.8 Linuxの動作確認

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『**rzg3s-aprzg30a login:**』が表示されます。
ログインを実行するには ユーザー『**root**』を入力してください。

ログイン設定	
ユーザー	root
パスワード	なし

Table 3.9-1 ログイン設定

```
rzg3s-aprzg30a login: root
```

時刻設定

AP-RZG3-0Aでは、RTC（リアルタイムクロック）が搭載されておりますので、その設定に応じて起動時の時刻が変わります。
その後、NTPサーバとの同期の有効／無効に応じて時刻が調整されます。

時刻関連設定は『**timedatectl**』コマンドにて行います。
『**timedatectl**』コマンドではNTPサーバと同期させるかどうかの設定、時刻の設定などができます。

① 現在の時刻および設定値の確認

```
# timedatectl
               Local time: Thu 2022-12-15 04:00:55 UTC
           Universal time: Thu 2022-12-15 04:00:55 UTC
                 RTC time: Thu 2022-12-15 04:00:55
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

dateコマンドも使用できますが、NTPサーバと同期している場合は、設定値は現在の時刻に上書きされます。
デバッグなどで一時的にテスト用の時刻に設定する場合は、『**timedatectl set-ntp false**』を実行して無効にしてください。

PowerLED

PowerLEDは、LEDクラスで実装されており、デフォルト状態では点灯1msec,消灯1999msecパターンにて点滅をします。

・LED点灯

PowerLEDのLEDを点灯させるには、以下のコマンドを実行します。

```
# echo none >/sys/class/leds/PowerLED/trigger
# echo 1 >/sys/class/leds/PowerLED/brightness
```

・LED消灯

PowerLEDのLEDを消灯させるには、以下のコマンドを実行します。

```
# echo none >/sys/class/leds/PowerLED/trigger
# echo 0 >/sys/class/leds/PowerLED/brightness
```

Triggerの値を変更することにより、LEDをイベントに連動させることができます。

例）ハートビートを使って点滅させる

```
# echo heartbeat >/sys/class/leds/PowerLED/trigger
```

LED（MON1, MON2）

MON1, MON2は、LEDクラスで実装されているため、コマンドにより点灯／消灯等の動作が行えます。

・LED点灯

MON1のLEDを点灯させるには、以下のコマンドを実行します。

```
# echo 1 >/sys/class/leds/MON1/brightness
```

MON2のLEDを点灯させるには、以下のコマンドを実行します。

```
# echo 1 >/sys/class/leds/MON2/brightness
```

・LED消灯

MON1のLEDを消灯させるには、以下のコマンドを実行します。

```
# echo 0 >/sys/class/leds/MON1/brightness
```

MON2のLEDを消灯させるには、以下のコマンドを実行します。

```
# echo 0 >/sys/class/leds/MON2/brightness
```

・trigger設定

triggerを指定することにより、LEDをイベントに連動させることができます。

例）ハートビートを使って点滅させる

```
# echo heartbeat >/sys/class/leds/MON1/trigger
```

例）triggerに設定可能な一覧の表示

```
# cat /sys/class/leds/MON1/trigger
none kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kbd-shiftlock kbd-altgrlock kbd
-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctrlrlock timer one
shot [heartbeat] cpu cpu0 default-on mmc0 mmc1
```

4. Linuxシステムの構築

本章では、Yocto Projectのリファレンスビルドシステムを使用して、ビルドする手順を説明します。
ビルドが行えるイメージとしては、下記のものがあります。
本章では、下記の中の「core-image-bsp」を例に説明を行います。

イメージ	内容
core-image-bsp	標準的な機能をサポートするコンソールのみのイメージです。
core-image-minimal	最小限の機能をサポートするコンソールのみのイメージです。

Table 4.1-1 ビルド可能なイメージ

core-image-bspおよびcore-image-minimalにてビルドされるコンポーネントについては、別紙「Yoctoコンポーネントリスト」を参照してください。

4.1 Yoctoのビルド環境の準備

ビルド前の環境設定方法の説明を以下に記述します。

① gitのユーザーおよびメールアドレスを登録していない場合は、最初にユーザー名とメールアドレスを登録します。
（登録されているかどうか不明な場合は「git config --list」コマンドにて確認してください）

```
$ git config --global user.email xxxx@yyyy
$ git config --global user.name  zzzzzzzz
```

② ホームディレクトリに作業用ディレクトリ『**rzg_vlp_v3.0.7**』を作成します。
すでに作成されている場合は、手順③にお進みください。

```
$ mkdir ~/rzg_vlp_v3.0.7
```

③ 以下のファイルを作業用ディレクトリにコピーします。

```
bsp_aprzg30a_X_X.tar.gz
```

④ 作業ディレクトリに移動します。

```
$ cd ~/rzg_vlp_v3.0.7
```

⑤ AP-RZG3-0A用のBSPを展開します。

```
$ tar zxvf ./bsp_aprzg30a_X_X.tar.gz
```

⑥ OSSパッケージのインストール(任意)
Yoctoのビルドでは、オープンソースパッケージをダウンロードしてビルドを行います。
オープンソースパッケージを事前に用意しておくことで、ビルド中のネットワーク負荷の軽減やビルド時間を大幅に短縮することができます。
本開発キットで使用しているオープンソースパッケージは、添付のCD-ROMには入っておりませんが、弊社ウェブサイトにて提供しております。
オープンソースパッケージを使用する場合には、BSPと同様に作業ディレクトリにコピーし、次のコマンドにて展開します。

```
$ 7z x -obuild oss_packages_aprzg30a_X_X.tar.7z
```

4.2 Yoctoのビルド

Yocto Project一式は、以下の手順でビルドします。

ビルド環境の設定

① 作業用ディレクトリに移動します。

```
$ cd ~/rzg_vlp_v3.0.7
```

② ビルド環境の設定をします。

```
$ source poky/oe-init-build-env build
```

『**source poky/oe-init-build-env**』コマンドは、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。

ビルド

① イメージを作成します。
ビルド環境の設定コマンドの実行後は、カレントディレクトリが『~/rzg_vlp_v3.0.7/build』に変わっていますので、そのまま以下のbitbakeコマンドを実行します。

```
$ MACHINE=rzg3s-aprzg30a bitbake -k core-image-bsp
NOTE: Your conf/bblayers.conf has been automatically updated.
Loading cache: 100% |                                              | ETA:  --:--:--
Loaded 0 entries from dependency cache.
Parsing recipes:  13% |#####                                        | ETA:  0:02:27
```

② ビルドが正常に完了しますと、ディレクトリ『~/rzg_vlp_v3.0.7/build/tmp/deploy/images/rzg3s-aprzg30a』にファイルが生成されます。
なお、主な生成ファイルを以下に説明します。

ファイル名	内容
FlashWriter-rzg3s-aprzg30a.mot	FlashWriterのバイナリファイル
bl2_bp_*-rzg3s-aprzg30a.srec	書込み用BL2のバイナリファイル (*: spi, esd, emmc)
fip-rzg3s-aprzg30a.srec	書込み用BL31/U-Bootのバイナリファイル
core-image-bsp-rzg3s-aprzg30a.wic.gz	ルートファイルシステム一式(カーネル等含む)
core-image-bsp-rzg3s-aprzg30a.wic.bmap	SDカードへの書き込み位置情報
modules-rzg3s-aprzg30a.tgz	モジュールファイル一式

Table 4.2-1 生成ファイル

上記ファイルは、リンクファイルの場合もありますので、実体は、別のファイル名で作成されています。
作成したバイナリファイルの書き込みは、『**10. Linuxシステムの作成**』でご確認ください。

5. U-Boot

5.1 U-Bootの起動

AP-RZG3-0A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-RZG3-0A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、各種ハードウェアマニュアルにてご確認ください。

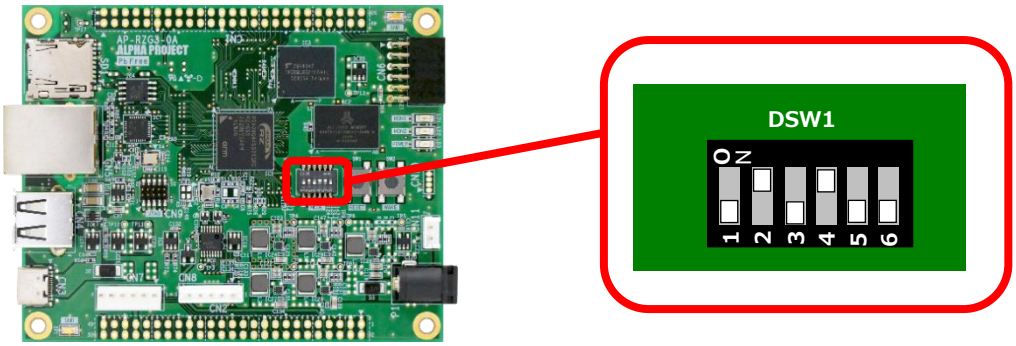


Fig 5.1-1 U-Boot起動設定

② 『**3.6 AP-RZG3-0Aの接続**』にしたがって、ホストPCとAP-RZG3-0Aを接続します。
電源はまだ入れないでください。
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-RZG3-0Aの電源を入れます。

5.3 BL2/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『4.2 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。

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。

ビルド環境の設定

① 作業ディレクトリに移動します。

```
$ cd ~/rzg_vlp_v3.0.7
```

② ビルド環境の設定をします。

```
$ 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~/rzg_vlp_v3.0.7/buildに移動します。

ビルド

① BL2をビルドします。

```
$ MACHINE=rzg3s-aprzg30a bitbake trusted-firmware-a -c compile --force
```

② BL2をデプロイ(配布)します。

```
$ MACHINE=rzg3s-aprzg30a bitbake trusted-firmware-a -c deploy
```

③ u-bootのビルドをします。

```
$ MACHINE=rzg3s-aprzg30a bitbake u-boot -c compile --force
```

④ u-bootのビルドが成功しましたら、デプロイ(配布)します。

```
$ MACHINE=rzg3s-aprzg30a bitbake u-boot -c deploy
```

⑤ firmware-packでU-Boot関連ファイルをひとまとめにします。

```
$ MACHINE=rzg3s-aprzg30a bitbake firmware-pack -c compile --force
```

⑥ firmware-packをデプロイ(配布)します。

```
$ MACHINE=rzg3s-aprzg30a bitbake firmware-pack -c deploy
```

6. Linuxカーネル

6.1 Linuxカーネルのソースの場所

Linuxカーネルは、Yocto環境では、以下の場所に展開されてビルドされます。

```
~/
 `-- rzg_vlp_v3.0.7
     `-- build
         `-- tmp
             `-- work-shared
                 `-- rzg3s-aprzg30a
                     `-- kernel-source
```

6.2 Linuxカーネルのカスタマイズ

デフォルトのUSBファンクション／ホストモード設定、Pmod等を使用する時は、Linuxカーネルのカスタマイズが必要となります。
Linuxカーネルのカスタマイズには、主に以下の作業が必要です。

1. menuconfigによる変更
2. デバイスツリーファイルの編集

なお、以降の手順では、本ドキュメントの『**4.2 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。

6.2.1 menuconfigによる変更

デフォルトのカーネルのconfigにてサポートされていないデバイスを、カーネルレベルでサポートするときなどにも使用します。

① 作業ディレクトリに移動します。

```
$ cd ~/rzg_vlp_v3.0.7
```

② ビルド環境の設定をします。

```
$ 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、『~/rzg_vlp_v3.0.7/build』に移動します。

6.3 Linuxカーネルの作成

ゲストOS(Ubuntu)上でLinuxカーネル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『**4.2 Yoctoのビルド**』で1度ビルドが完了している環境が必要となります。

ビルド環境の設定

① 作業ディレクトリに移動します。

```
$ cd ~/rzg_vlp_v3.0.7
```

② ビルド環境の設定をします。

```
$ 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~/rzg_vlp_v3.0.7/buildに移動します。

ビルド

① Linuxカーネルのビルドをします。

```
$ MACHINE=rzg3s-aprzg30a bitbake linux-renesas -c compile --force
```

② Linuxカーネルのビルドが成功しましたら、デプロイ(配布)します。

```
$ MACHINE=rzg3s-aprzg30a bitbake linux-renesas -c deploy
```

③ 正常に完了しますと、ディレクトリ『~/rzg_vlp_v3.0.7/build/tmp/deploy/images/rzg3s-aprzg30a』の以下のファイルが更新されます。

ファイル名	内容
Image-rzg3s-aprzg30a.bin	カーネルイメージファイル
r9a08g045s33-aprzg30a.dtb	デバイスツリーファイル
modules-rzg3s-aprzg30a.tgz	モジュールファイル一式

Table 6.3-1 生成ファイル

上記ファイルは、リンクファイルの可能性がありますので、実体は、別のファイル名で作成されています。
作成したバイナリファイルの書き込みは、『**10. Linuxシステムの作成**』でご確認ください。

クリーン

変更した内容をクリーンするために、ソース等を再展開する場合には、以下の手順で行います。
変更していた内容はすべて元に戻りますので、実行する場合にはご注意ください。

① Linuxカーネルのソースをクリーンします。

```
$ MACHINE=rzg3s-aprzg30a bitbake linux-renesas -c cleansstate
```

② Linuxカーネルのソースを再展開します

```
$ MACHINE=rzg3s-aprzg30a bitbake linux-renesas -c configure
```

6.4 Linuxカーネルのデバイスドライバ

AP-RZG3-0Aシリーズの Linuxカーネルにて設定されている主なデバイスドライバは以下の通りです。

デバイス	実装内容
Ethernet	100/1000Base Ethernetインターフェース
USB ホスト	マスストレージクラス
USB ファンクション	コミュニケーションデバイスクラス
microSDカード	MMCインターフェース
LED	LEDクラス
UART	シリアルコミュニケーションインターフェース
CAN	CAN/CAN FDインターフェース
GPIO	libgpiodライブラリ
QSPI Flash	MTDクラス
Watchdog	Watchdogクラス

Table 6.4-1 デバイスドライバ

以下に、上記のデバイスドライバの仕様を記載します。

Ethernet

Ethernetは、TCP/IPプロトコルスタック等で制御できるように実装しています。
各種ネットワークの設定は、Linuxカーネルの以下のmenuconfigの項目によって設定します。

[device drivers]
[Network device support]

また、インターフェース名は、以下となります。

インターフェース名
eth0

Table 6.4-2 Ethernetのインターフェース

7.2 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 ~/rzg_vlp_v3.0.7
$ source poky/oe-init-build-env
```

『**source poky/oe-init-build-env**』コマンドは、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、『~/rzg_vlp_v3.0.7/build』に移動しますので、
bitbakeコマンドでSDKを作成します。

```
$ MACHINE=rzg3s-aprzg30a bitbake -c populate_sdk core-image-bsp
Loading cache: 100% |#########################################| Time: 0:00:01
Loaded 5283 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

     <以下ログ省略>
```

8. サンプルアプリケーション

本章では、AP-RZG3-0A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。

8.1 サンプルアプリケーションの作成

ゲストOS(Ubuntu)上で、アプリケーションを作成するための手順を説明します。

作成のための準備

① 作業用ディレクトリ『**aprzg30a-app**』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 ~/aprzg30a-app
```

② ディレクトリ『**aprzg30a-app**』に移動します。

```
$ cd ~/aprzg30a-app
```

③ 作業用ディレクトリに以下のファイルをコピーします。

helloworld-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 サンプルソースを展開します。

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

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『**7.3 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/3.1.33/environment-setup-aarch64-poky-linux
```

最初の『**.**』と『**/opt/poky/3.1.33/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。

上記の環境設定コマンドは、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/3.1.33/envir...**』とYoctoビルドの環境設定『**source poky/oe-init-build-env**』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。

10. Linuxシステムの作成

本章では、AP-RZG3-0AのeMMCにLinuxシステムを書き込む方法について説明します。
eMMCのファイルシステムを入れ替えるには、microSDにより起動したLinuxより書き込む必要があります。

10.1 概要

AP-RZG3-0AへeMMC Linuxシステムを作成するには、必要に応じて以下の3つを行います。

1. QSPI FlashへU-Boot等を書き込む
2. eMMC書き変え用のmicroSDカードを用意する
3. QSPI Flash/microSDカードのSD Linuxシステムで起動して、eMMCに書き込む

10.2 必要なファイル

書き込みを行うBL2/U-BootとLinuxシステムにてそれぞれ必要なファイルが異なります。
以下にそれぞれで必要なファイルを記載します。

ファイル名	内容
FlashWriter-rzg3s-aprzg30a.mot	FlashWriterのバイナリファイル
bl2_bp_spi-rzg3s-aprzg30a.srec	BL2のバイナリファイル
fip-rzg3s-aprzg30a.srec	U-Bootのバイナリファイル

Fig 10.2-1 QSPI Flash書き込み時に必要なファイル

ファイル名	内容
core-image-bsp-rzg3s-aprzg30a.wic.gz	ルートファイルシステム一式(カーネル等含む)
core-image-bsp-rzg3s-aprzg30a.wic.bmap	SDカードへの書き込み位置情報
Image-rzg3s-aprzg30a.bin	カーネルイメージファイル
r9a08g045s33-aprzg30a.dtb	デバイスツリーファイル
modules-rzg3s-aprzg30a.tgz	モジュールファイル一式
bl2_bp_emmc-rzg3s-aprzg30a.bin	eMMC用 BL2のバイナリファイル
fip-rzg3s-aprzg30a.bin	BL31/U-Bootのバイナリファイル
core-image-bsp-rzg3s-aprzg30a.tar.bz2	ルートファイルシステム一式

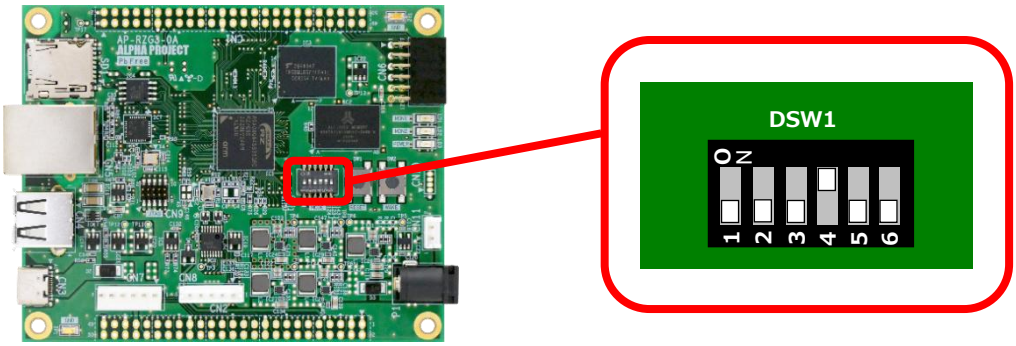
Fig 10.2-2 eMMC書き込み時に必要なファイル

10.3　QSPI Flashへの書き込み

FlashWriterを使用して、シリアル経由にてQSPI FlashにSPL, U-Bootを書き込みます。
Windowsにて実行しますので、『**10.2　必要なファイル**』にて記載したファイルをWindowsの任意の場所にコピーします。

FlashWriterの起動

① AP-RZG3-0Aのスイッチが以下の設定になっていることを確認します。



② AP-RZG3-0AにmicroSDカードが挿入されていないことを確認します。

③ 『**3.6　AP-RZG3-0Aの接続**』にしたがって、ホストPCとAP-RZG3-0Aを接続します。
本手順では、ネットワークは必要ありません。
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。
AP-RZG3-0Aの電源はまだ入れないでください。

④ ホストOS（Windows）のターミナルソフトを起動します。
シリアル設定は、以下の表の設定値で行います。

シリアルの設定	
ポート番号	PCで認識した仮想シリアルポート
通信速度	115200bps
データ長	8bit
ストップビット	1bit
パリティ	なし
フロー制御	なし

Table 10.4-1 シリアル設定(FlashWriter起動時)

⑤ ACアダプタを接続して、AP-RZG3-0Aの電源を入れます。
ターミナルソフトに以下が表示されます。

```
SCI Download mode (Normal SCI boot)
-- Load Program to SRAM ---------------
```

10.5　eMMCへの書き込み

『**10.4　microSDカードへの書き込み**』で作成したSDカードを使用し、eMMCから起動できるよう書き込みを行います。

eMMCの構成

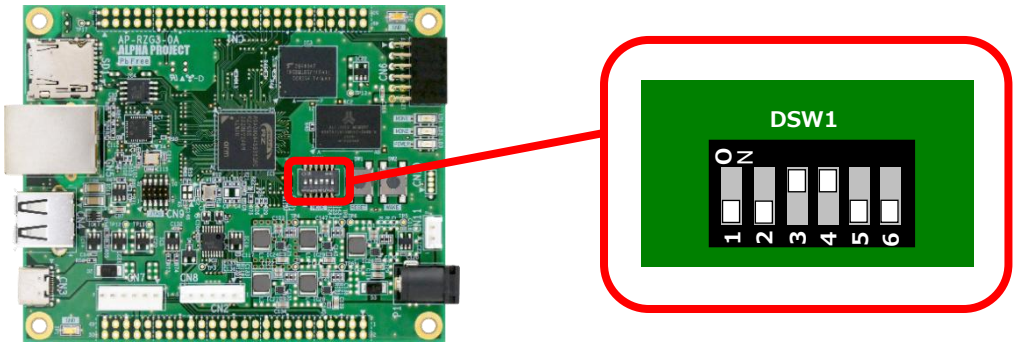
eMMCから起動するためには、以下の構成に合わせて、データが書き込まれている必要があります。

●eMMCの構成

デバイスファイル	書き込みファイル	備考
mmcblk0boot0	bl2_bp_emmc-rzg3s-aprzg30a.bin	書き込み開始セクタ :0x0001
	fip-rzg3s-aprzg30a.bin	書き込み開始セクタ: 0x0320
mmcblk0boot1	–	未使用
mmcblk0p1	–	未使用(500MB)
mmcblk0p2	core-image-weston-rzg3s-aprzg30a.tar.bz2	ルートディレクトリに展開
	modules-rzg3s-aprzg30a.tgz	ルートディレクトリに展開
	Image-rzg3s-aprzg30a.bin	/bootディレクトリにコピー
	r9a08g045s33-aprzg30a.dtb	/bootディレクトリにコピー

eMMCへの書き込み

① 『**10.4　microSDカードへの書き込み**』にて作成したSDカードをAP-RZG3-0AのSDカードスロット（SD1）に差し込みます。
AP-RZG3-0Aのスイッチを下記の設定にし、電源を入れます。



② U-Bootが起動時にキー入力をし、U-Bootのコマンドコンソールを起動します。
コマンドコンソールの起動方法の詳細は、『5.1 U-Bootの起動』を参照してください。

③ SD1からLinuxを起動するコマンドをコマンドコンソールから入力します。

```
=> env default -a ; run sd1load ; run bootimage
```

④ ログインプロンプト『**rzg3s-aprzg30a login:**』に『**root**』と入力しログインします。

```
rzvg3s-aprzg30a login:root
```

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。

本文書について

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