

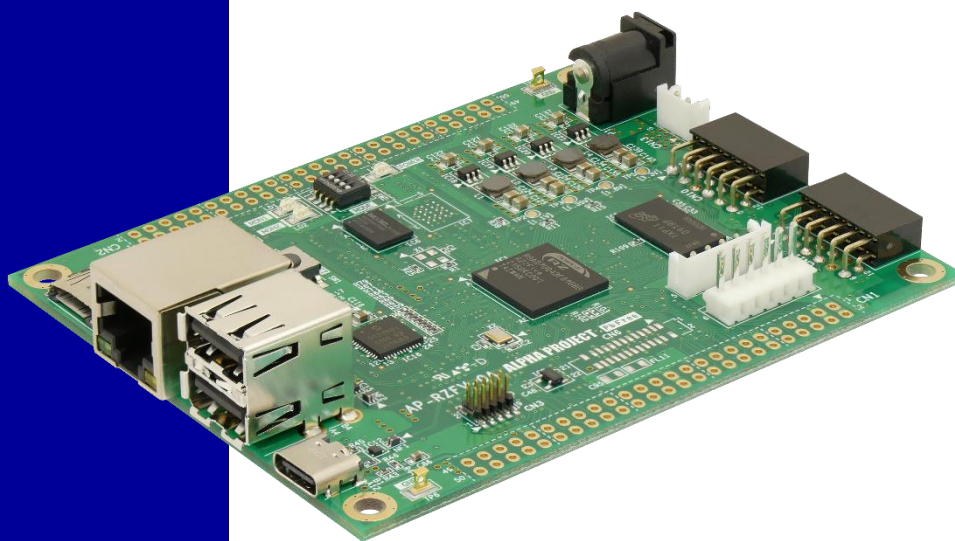
Alpha Board Series
LK-RZFV-A01

RISC-V CPU BOARD

LinuxBSP Startup Guide

Rev 1.1

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。

LK-RZFV-A01梱包内容

<ul style="list-style-type: none">●LANストレートケーブル 1本 	<ul style="list-style-type: none">●ACアダプタ 1本 
<ul style="list-style-type: none">●PC-USB-04(ケーブル付) 1個 	<ul style="list-style-type: none">●USBケーブル(A - B) 1本  <p>コネクタの形状</p>
<ul style="list-style-type: none">●microSDカード 1個 	<ul style="list-style-type: none">●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

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
2. システム概要	4
2.1 システム概要	4
2.2 U-Boot (ブートローダ)	5
2.3 Linuxカーネル	6
2.4 ルートファイルシステム	7
2.5 クロス開発環境	8
2.6 添付CD-ROMの構成	9
3. システムの動作	10
3.1 動作環境	10
3.2 microSDカードの準備	10
3.3 シリアル初期設定値	11
3.4 MACアドレス	11
3.5 ネットワーク初期設定値	12
3.6 USB ID初期設定値	13
3.7 AP-RZFB-0Aの接続	14
3.8 Linuxの起動	15
3.9 Linuxの動作確認	16
3.10 Linuxの終了/再起動	27
4. Linuxシステムの構築	28
4.1 Yoctoのビルド環境の準備	28
4.2 Yoctoのビルド	29
5. U-Boot	31
5.1 U-Bootの起動	31
5.2 U-Bootのソースの場所	33
5.3 U-Bootの作成	34
5.4 環境変数	35

5.5 ネットワーク設定.....	37
6. Linuxカーネル	38
6.1 Linuxカーネルのソースの場所.....	38
6.2 Linuxカーネルのカスタマイズ.....	38
6.3 Linuxカーネルの作成.....	42
6.4 Linuxカーネルのデバイスドライバ.....	43
7. アプリケーションの開発環境	48
7.1 アプリケーションの開発について.....	48
7.2 SDKの作成.....	49
7.3 SDKのインストール.....	50
8. サンプルアプリケーション	51
8.1 サンプルアプリケーションの作成.....	51
8.2 動作確認.....	53
9. サンプルデバイスドライバ	54
9.1 サンプルデバイスドライバの概要.....	54
9.2 サンプルデバイスドライバ/アプリケーションの作成.....	56
9.3 動作確認.....	60
10. Linuxシステムの作成	61
10.1 概要.....	61
10.2 準備.....	61
10.3 QSPI Flashへの書き込み.....	62
10.4 microSDカードへの書き込み.....	68
10.5 microSDカードのLinuxパーティション変更.....	73
11. 製品サポートのご案内	75
12. エンジニアリングサービスのご案内	76
付録A. 起動ログ	77
付録B. 終了ログ	86

2. システム概要

2.1 システム概要

AP-RZFV-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。

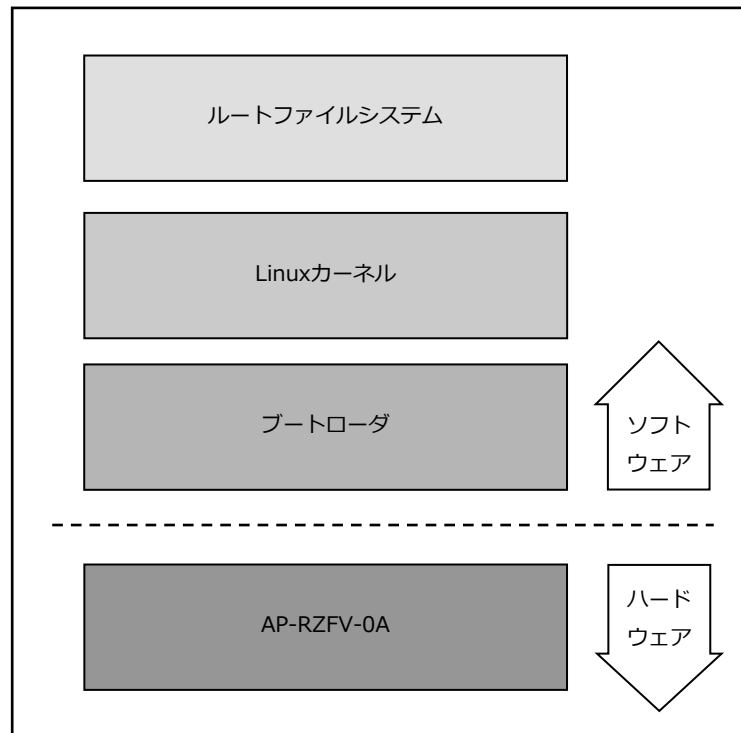


Fig 2.1-1 AP-RZFV-0Aシステム概要図

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。

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。

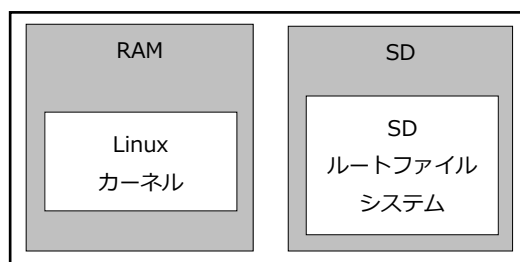


Fig 2.4-1 SD-Linuxシステム

2.6 添付CD-ROMの構成

AP-RZFV-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_RZFV_A01_VX_X	
-- an	
-- an1659.pdf	: AN1659 Pmodの使用方法
-- manual	
-- lk_rzfv_a01_inst.pdf	: 開発環境インストールガイド
-- lk_rzfv_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_lkrzfv01_X.X.tar.gz	: AP-RZFV-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-RZFV-A01の製品ページ
<https://www.apnet.co.jp/product/rza/lk-rzfv-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-RZFV-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
SDカードリーダー	microSDカードの読み書き (Ubuntuから認識できること)
PC-USB-04及びケーブル	ホストPCとAP-RZFV-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
microSDカード	4GByte以上
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-RZFV-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-RZFV-A01 Development Environment Install Guide**』でご確認ください。

3.2 microSDカードの準備

AP-RZFV-0AにてLinuxを起動させるには、Linuxシステムが書き込まれたmicroSDカードが必要です。

プリインストールされた本開発キット付属のmicroSDカード、もしくは、本ドキュメントの「[4. Linuxシステムの構築](#)」と「[10. Linuxシステムの作成](#)」で作成するmicroSDカードを用意します。

3.7 AP-RZFV-0Aの接続

ホストPCとAP-RZFV-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、設定に注意して接続してください。

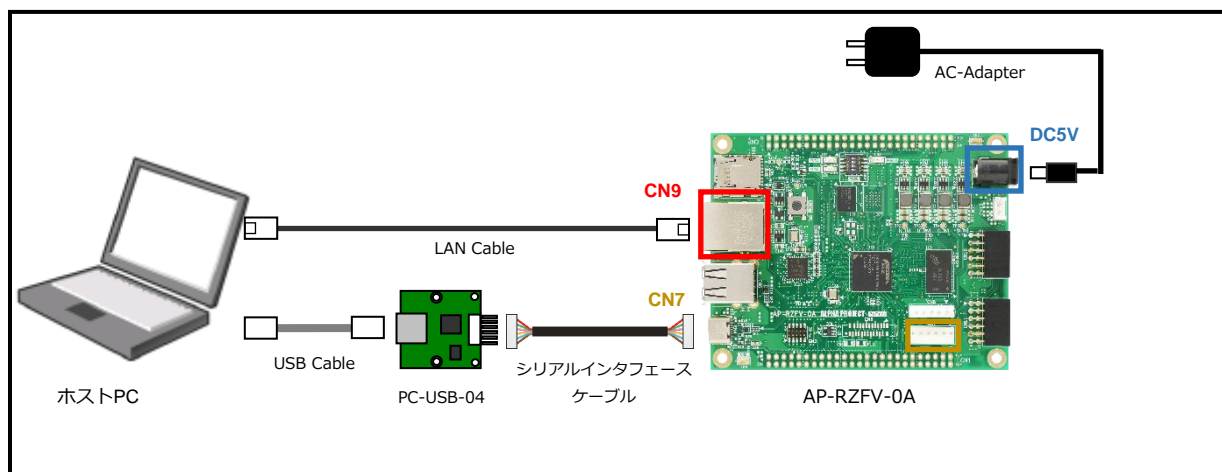


Fig 3.7-1 AP-RZFV-0Aの接続 (PCに接続する場合)

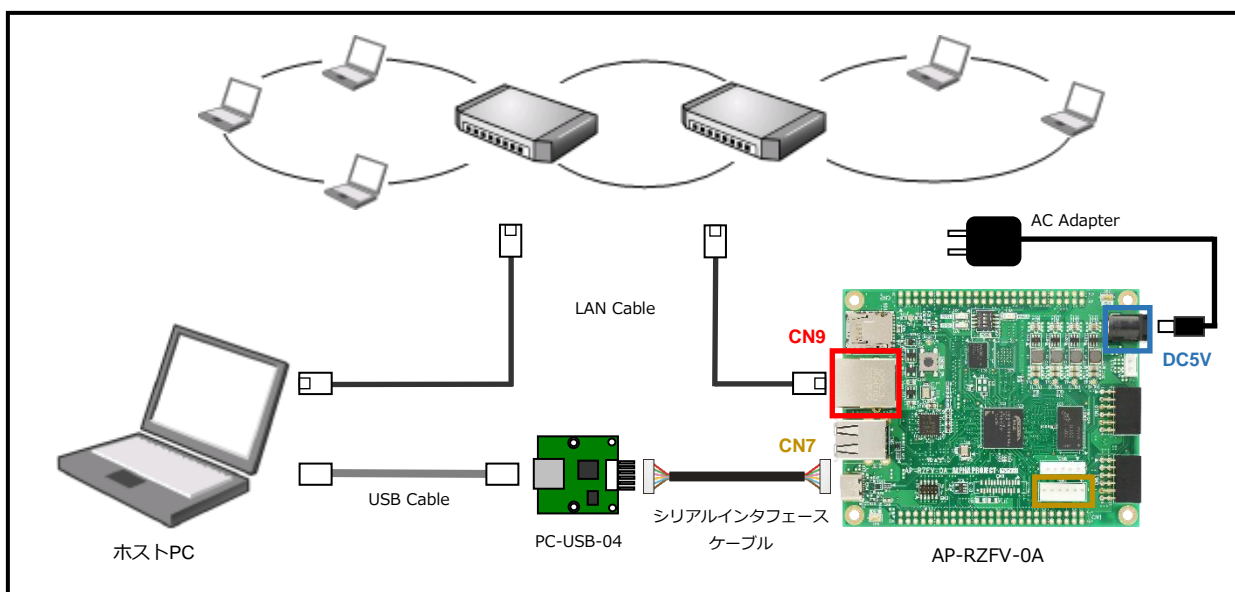
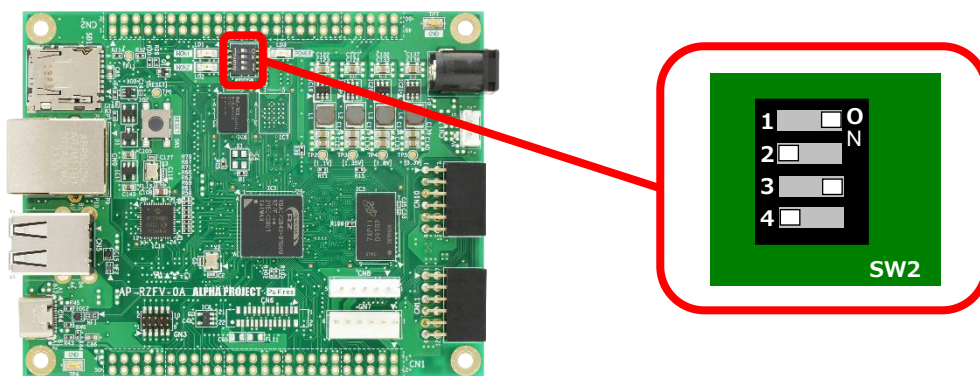


Fig 3.7-2 AP-RZFV-0Aの接続 (HUBに接続する場合)

3.8 Linuxの起動

AP-RZFV-0A上でLinuxの起動を行います。

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



- ② SD1にLinuxルートファイルシステムの入ったmicroSDカードを挿入します。
- ③ 『[3.7 AP-RZFV-0Aの接続](#)』にしたがって、ホストPCとAP-RZFV-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。
- ④ ホストOS (Windows) のターミナルソフトを起動します。(設定は『[3.3 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-RZFV-0Aの電源を入れます。
- ⑥ 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。

```
U-Boot SPL 2021.10 (XXX XX XXXX - XX:XX:XX)
```

```
Trying to boot from NOR
```

```
U-Boot 2021.10 (XXX XX XXXX - XX:XX:XX)
```

```
CPU:   rv64imafdc
```

```
:
<途中省略>
```

```
BSP:   RZFive/AP-RZFV-0A/3.0.2-update1
```

```
LSI:   RZFive
```

```
Version: 3.0.2-update1
```

```
aprzfv0a login:
```

3.9 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『**aprzfv0a 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

```
aprzfv0a login: root ←
```

時刻設定

AP-RZFBV-0A上で時刻の設定をします。

RTCが実装されていないため、Linuxは起動時にルートファイルシステムのビルド日時、あるいは前回NTPで取得した日時などを基準としてCPU内のタイマーモジュールにて管理します。したがって、電源ONのたびに日時を設定するかNTPサーバとの時刻の同期が取れるまで正しい時刻を得ることができません。

NTPサーバとの同期が有効の場合は時刻が定期的に同期されます。

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: n/a
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てしまいますので、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ます。

LED (MON1)

LED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 ←
```

- ・ LED消灯

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 0 >/sys/class/leds/MON1/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) タイマーを使って点滅させる

```
# echo timer >/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 bluetooth-power rfkill-any rfkill-none kbd-scrolllock kbd-numlock kbd-capslock kbd-k  
analog kbd-shiftlock kbd-altgrlock kbd-ctrllock kbd-altlock kbd-shiftllock kbd-shiftrloc  
k kbd-ctrllock kbd-ctrlrlock [timer] disk-activity disk-read disk-write ide-disk heartbe  
at cpu cpu0 cpu1 mmc2 default-on panic mmc1
```



AP-RZJV-0A上のLED(MON2)は、出荷時にはSDカードのUHS対応に使用しています。
そのため、コマンドによる制御は行えません。

GPIO

各種GPIOは、GPIOクラスとして登録することで制御が行えます。

例えば、GPIO10_1にHighを出力する場合には、以下のコマンドを実行します。

① GPIO10_1を制御できるようにexportする。

```
# echo 441 >/sys/class/gpio/export ←
```

② GPIO10_1を出力に設定する。

```
# echo out >/sys/class/gpio/gpio441/direction ←
```

③ GPIO10_1をHighに設定する。

```
# echo 1 >/sys/class/gpio/gpio441/value ←
```

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ためのイメージです。
core-image-minimal	最小限の機能をサポートするコンソールのためのイメージです。

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 zzzzzzz ←
```

- ② ホームディレクトリに作業用ディレクトリ『rzfive_vlp_v3.0.2』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 ~/rzfive_vlp_v3.0.2 ←
```

- 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_lkrzfva01_X_X.tar.gz
```

- ④ AP-RZFV-0A用のBSPを展開します。

```
$ cd ~/rzfive_vlp_v3.0.2 ←
$ tar zxvf ./bsp_lkrzfva01_X_X.tar.gz ←
```

以上の手順で、Yoctoのビルドは可能です。

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、次のコマンドにて展開します。

- ⑤ oss_packages_lkrzfva01_X_X.tar.7zファイルの展開をします。

```
$ cd ~/rzfive_vlp_v3.0.2 ←
$ 7z x oss_packages_lkrzfva01_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 ビルド環境の設定をします。

```
$ cd ~/rzfive_vlp_v3.0.2 ←
$ source poky/oe-init-build-env ←

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

Common targets are:
  core-image-minimal
  core-image-sato
  meta-toolchain
  meta-ide-support

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

Other commonly useful commands are:
- 'devtool' and 'recipetool' handle common recipe tasks
- 'bitbake-layers' handles common layer tasks
- 'oe-pkgdata-util' handles common target package tasks
```



『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。

confファイルの準備

① confファイルの準備をします。

~/rzfive_vlp_v3.0.2/build/conf/local.confは「[4.1 Yoctoのビルド環境の準備](#)」にて作成されています。

```
$ cd ~/rzfive_vlp_v3.0.2/build ←
$ cp ~/rzfive_vlp_v3.0.2/meta-renesas/docs/template/conf/aprzfv0a/*.conf ./conf/ ←
```

5. U-Boot

5.1 U-Bootの起動

AP-RZFV-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-RZFV-0Aの電源を入れる前に、スイッチが以下のように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。

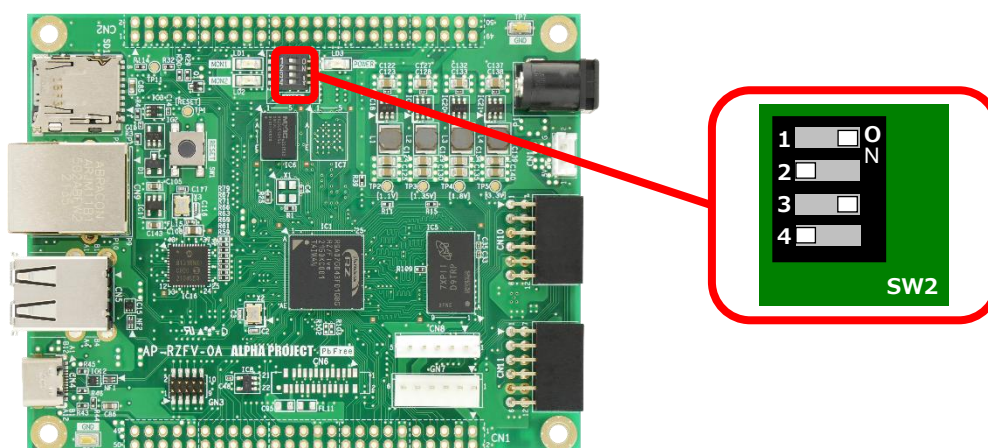


Fig 5.1-1 U-Boot起動設定

- ② 『[3.7 AP-RZFV-0Aの接続](#)』にしたがって、ホストPCとAP-RZFV-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『[3.3 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-RZFV-0Aの電源を入れます。

5.3 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。

- ① ビルド環境の設定をします。

```
$ cd ~/rzfive_vlp_v3.0.2 ←
$ 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~/rzfive_vlp_v3.0.2/buildに移動します。

- ② u-bootのソースを再展開するには、以下のコマンドを実行します。

既存のソースを使用する場合には、この手順のコマンドは行いません。

```
$ bitbake -c clean u-boot ←
$ bitbake -c configure u-boot ←
```

- ③ u-bootのビルドをします。

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

- ④ u-bootのビルドが成功しましたら、デプロイ(配布)を実行します。

```
$ bitbake -c deploy u-boot ←
```

- ⑤ 正常に完了しますと、ディレクトリ『~/rzfive_vlp_v3.0.2/build/tmp/deploy/images/aprzfv0a』に以下のファイルが更新されます。

ファイル名	内容
spl-aprzfv0a.srec	SPLのバイナリファイル
fit-aprzfv0a.srec	U-Bootのバイナリファイル

Table 5.4-1 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るので、コピー時にはご注意ください。

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。

```
~/
├── rzfive_vlp_v3.0.2
│   ├── build
│   │   ├── tmp
│   │   │   ├── work-shared
│   │   │   │   ├── aprzfv0a
│   │   │   │   │   └── kernel-source
```

6.2 Linux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、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

以下では、USBの設定を例にして、カスタマイズ方法を説明します。

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 ビルド環境の設定をします。

```
$ cd ~/rzfive_vlp_v3.0.2 ←
$ 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、『~/rzfive_vlp_v3.0.2/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ので、ご注意ください。

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 ~/rzfive_vlp_v3.0.2 ←
$ 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~/rzfive_vlp_v3.0.2/buildに移動します。

- ② Linuxカーネルのソースを再展開するには、以下のコマンドを実行します。

既存のソースを使用する場合には、この手順のコマンドは行いません。

```
$ bitbake -c configure linux-renesas ←
```

- ③ Linuxカーネルのビルドをします。

```
$ bitbake -c compile --force linux-renesas ←
```

- ④ 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。

```
$ bitbake -c deploy linux-renesas ←
```

- ⑤ 正常に完了しますと、ディレクトリ『~/rzfive_vlp_v3.0.2/build/tmp/deploy/images/aprzfv0a』の以下のファイルが更新されます。

なお、変更内容によっては変わらないファイルもあります。

ファイル名	内容
Image-aprzfv0a.bin	カーネルイメージファイル
r9a07g043f01-aprzfv0a.dtb	デバイスツリーファイル
modules-aprzfv0a.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、リンクファイルとなります。実体は、別のファイル名で作成されています。

6.4 Linuxカーネルのデバイスドライバ

AP-RZFV-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	マスタストレージクラス
USB ファンクション	コミュニケーションデバイスクラス
SD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	GPIOクラス
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。

[Networking support]
 [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 ~/rzfive_vlp_v3.0.2 ←  
$ 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、『~/rzfive_vlp_v3.0.2/build』に移動しますので、bitbakeコマンドでSDKを作成します。

```
$ bitbake -k -c populate_sdk core-image-bsp ←
```

<ログ省略>

8. サンプルアプリケーション

本章では、AP-RZFV-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『**aprzfv0a-app**』または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 ~/aprzfv0a-app ←
```

- ② ディレクトリ『**aprzfv0a-app**』に移動します。

```
$ cd ~/aprzfv0a-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-1.0.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/3.1.17/environment-setup-riscv64-poky-linux ←
```



最初の『.』と『/opt/poky/3.1.17/environment-setup-riscv64-poky-linux』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/3.1.17/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、QSPI FlashにSPL/U-Bootを、microSDカードにLinuxシステム（カーネル・デバイスツリー、ルートファイルシステム）を書き込む方法について説明します。

10.1 概要

AP-RZFV-0AへLinuxシステムを作成するには、以下のファイルが必要となります。
必要なファイルは、「[4. Linuxシステムの構築](#)」の手順にて作成することができます。

ファイルタイプ	ファイル名	備考
FlashWriter	Flash_Writer_SCIF_RZFIVE_APRZV0A.mot	QSPI Flashへ書き込みする時に使用
SPL	spl-aprzfv0a.srec	
U-Boot	fit-aprzfv0a.srec	
Linux Kernel	Image-aprzfv0a.bin	
Linux Device Tree	r9a07g043f01-aprzfv0a.dtb	
Root File System	core-image-bsp-aprzfv0a.tar.bz2 または core-image-minimal-aprzfv0a.tar.bz2	

Fig 10.1-1 必要なファイル

10.2 準備

書き込み手順に必要なmicroSDカード(4GByte以上)を準備します。

SPL,U-Bootの書き込みは、Windowsのターミナルソフト（TeraTerm等）で行いますので、事前に以下のファイルをWindows側にコピーしておきます。

ファイル名
Flash_Writer_SCIF_RZFIVE_APRZV0A.mot
spl-aprzfv0a.srec
fit-aprzfv0a.srec

Fig 10.2-1 Windows側で使用するファイル

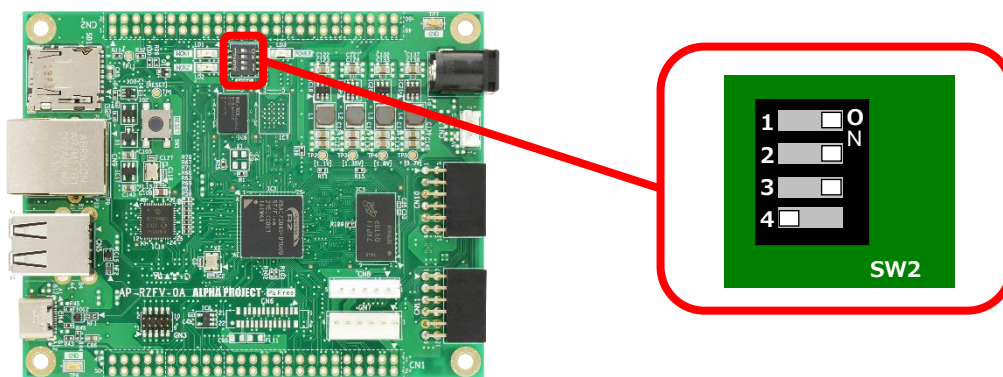
10.3 QSPI Flashへの書き込み

FlashWriterを使用して、QSPI FlashにSPL, U-Bootを書き込みます。

シリアル経由にて起動したFlashWriteを使用してSPL, U-Bootを書き込みます。

FlashWriterの起動

- ① シリアルコンソールを接続し、TeraTermを起動します。
- ② ボードのMODE SW2-2をONにします。



- ③ 電源を入れると以下のメッセージが表示されます。

```
SCIF Download mode due to error
-- Load Program to System RAM -----
please send !
```

10.4 microSDカードへの書き込み

microSDカードへLinuxシステムを書き込みます。

本手順では、Ubuntuを使用して書き込みます。

なお、必要なファイルを以下に記載しますので、事前にUbuntu内に用意します。

(本ドキュメントでは、~/aprzfv0a_binに用意した手順で説明します。)

項目	ファイル名
Linux kernel	Image-aprzfv0a.bin
Device tree file	r9a07g043f01-aprzfv0a.dtb
root filesystem	core-image-bsp-aprzfv0a.tar.bz2 または core-image-minimal-aprzfv0a.tar.bz2

Fig 10.4-1 microSDカードに書き込むファイル

① microSDカードのデバイス名の確認

以下のコマンドでmicroSDカードのデバイスファイル名を確認します。

```
$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0         7:0    0     4K  1 loop /snap/bare/5
loop1         7:1    0  55.6M  1 loop /snap/core18/2620
:
<途中省略>
:
sdb          8:16    1   3.7G  0 disk
├─sdb1        8:17    1   500M  0 part /media/guest/boot
└─sdb2        8:18    1   3.1G  0 part /media/guest/rootfs
```



上記では、microSDカードデバイス名は、/dev/sdbとなっています。
接続機器や環境によっては、/dev/mmcblkや/dev/sdc等となる場合もありますので、環境に応じて以下の手順では、読み替えてください。



VirtualBoxのゲストOSとしてUbuntuにてUSB接続のカードリーダーを使用している場合は、VirtualBoxのメニュー[デバイス]-[USB]にてお使いのカードリーダーを選択しておく必要があります。

以降の手順にて他のデバイス名を使わないように注意してください。

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

- ・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が、万一ご不審な点、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。



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
