Alpha Board Series

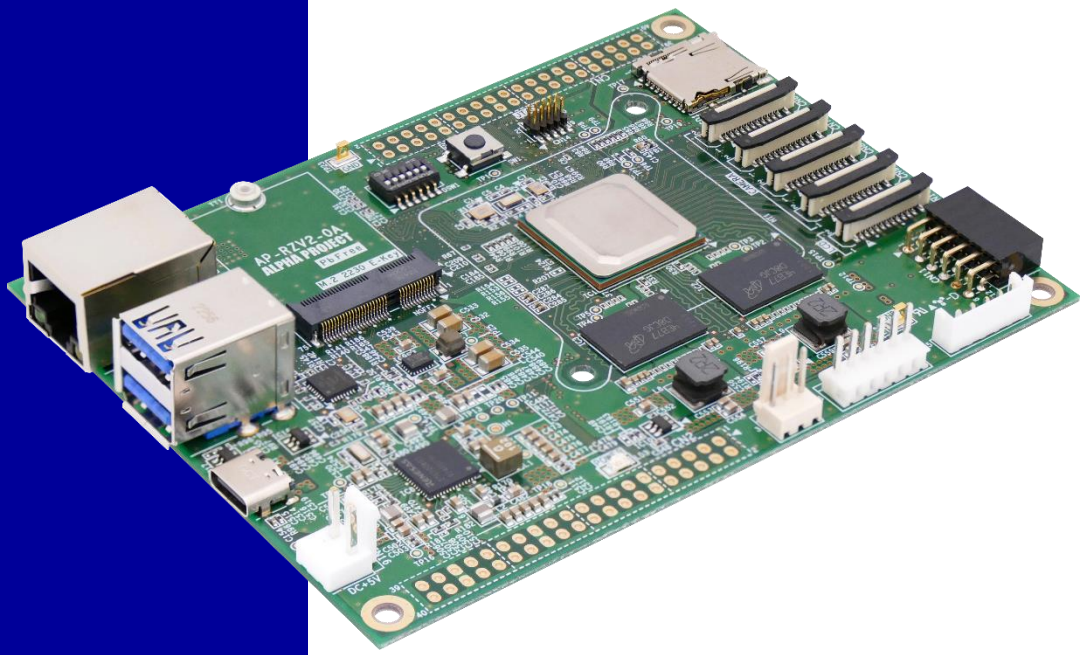
LK-RZV2-A01

RZファミリ RZ/V2H 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-RZV2-A01梱包内容

●LANストレートケーブル　1本 	●ACアダプタ(Type-C)　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

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 RZ/V2H AI Software Development Kit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 シリアル初期設定値 11
3.3 MACアドレス 11
3.4 ネットワーク初期設定値 12
3.5 USB ID初期設定値 13
3.6 AP-RZV2-0Aの接続 14
3.7 Linuxの起動 15
3.8 Linuxの動作確認 16
3.9 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 30

5. U-Boot 31

5.1 U-Bootの起動 31
5.2 BL2/U-Bootのソースの場所 33
5.3 BL2/U-Bootの作成 34
5.4 環境変数 36

5.5 ネットワーク設定(固定IP)..... 38
6. Linuxカーネル 39
6.1 Linuxカーネルのソースの場所..... 39
6.2 Linuxカーネルのカスタマイズ..... 39
6.3 Linuxカーネルの作成 43
6.4 Linuxカーネルのデバイスドライバ..... 45
7. アプリケーションの開発環境 50
7.1 アプリケーションの開発について..... 50
7.2 SDKの作成 51
7.3 SDKのインストール..... 52
8. サンプルアプリケーション 53
8.1 サンプルアプリケーションの作成..... 53
8.2 動作確認 55
9. サンプルデバイスドライバ 56
9.1 サンプルデバイスドライバの概要..... 56
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..... 58
9.3 動作確認 62
10. Linuxシステムの作成 63
10.1 概要 63
10.2 必要なファイル..... 63
10.3 QSPI Flashへの書き込み..... 64
10.4 microSDカードへの書き込み 68
10.5 eMMCへの書き込み 71
11. 製品サポートのご案内 74
12. エンジニアリングサービスのご案内 75
付録A. 起動ログ 76
付録B. 終了ログ 90

2. システム概要

2.1 システム概要

AP-RZV2-0A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。

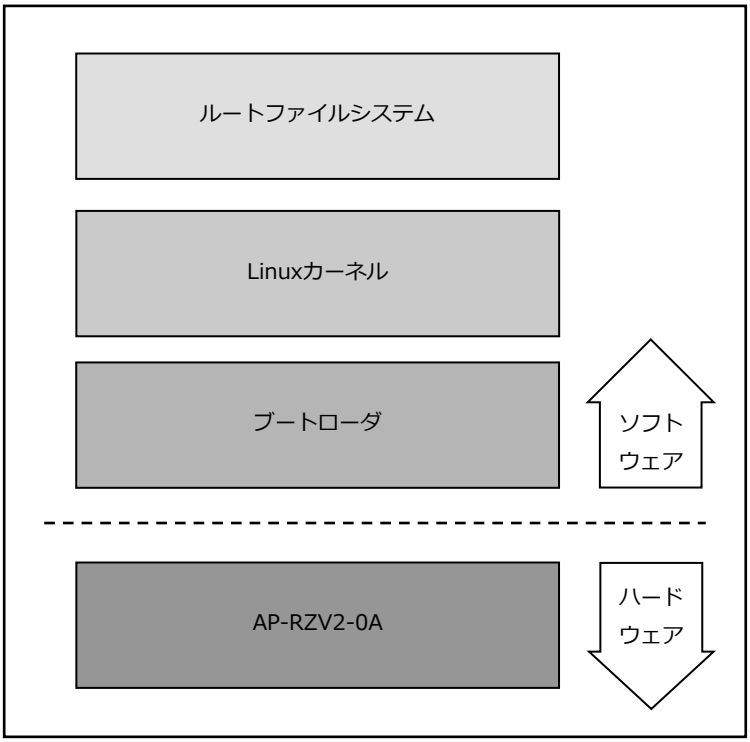


Fig 2.1-1 AP-RZV2-0Aシステム概要図

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。

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

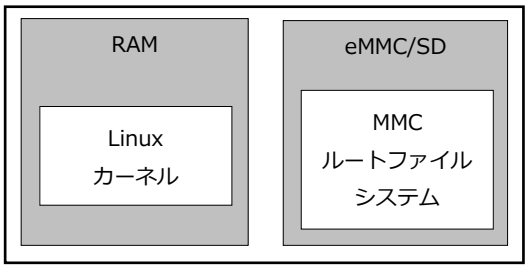


Fig 2.4-1 MMC-Linuxシステム

2.6 添付CD-ROMの構成

AP-RZV2-0A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_RZV2_A01_VX_X
|-- an
|   |-- an1664.pdf                    : AN1664 MIPI DSI Displayの使用方法
|   |-- an1665.pdf                    : AN1665 MIPIカメラの使用方法
|   |-- an1666.pdf                    : AN1666 Pmodの使用方法
|   |-- an1667.pdf                    : AN1667 DRP-AIの使用方法
|   `-- an1668.pdf                    : AN1668 DRP-AIデモの動作方法
|-- manual
|   |-- lk_rzv2_a01_inst.pdf          : 開発環境インストールガイド
|   |-- lk_rzv2_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
|   |-- mipicam-X.X.tar.bz2           : カメラ用サンプルスクリプト
|   `-- pmod-X.X.tar.bz2              : Pmodサンプル
`-- sources
    `-- bsp_lkrzv2a01_X.X.tar.gz      : AP-RZV2-0A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-RZV2-A01の製品ページ
https://www.apnet.co.jp/product/rza/lk-rzv2-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-RZV2-0A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	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-RZV2-0A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		32GByte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-RZV2-0A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-RZV2-A01 Development Environment Install Guide** 』でご確認ください。

3.6 AP-RZV2-0Aの接続

ホストPCとAP-RZV2-0A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。

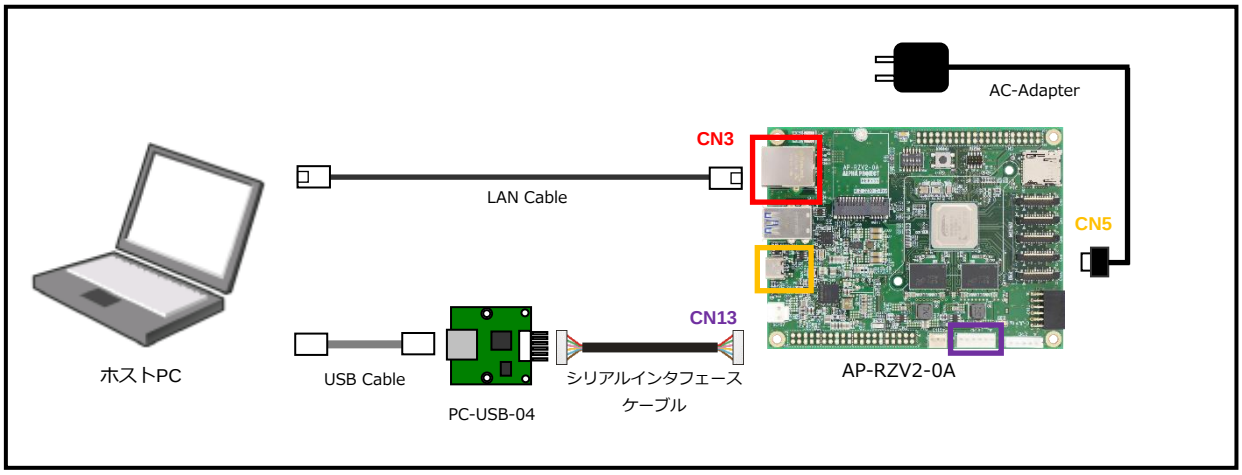


Fig 3.6-1 AP-RZV2-0Aの接続（PCに接続する場合）

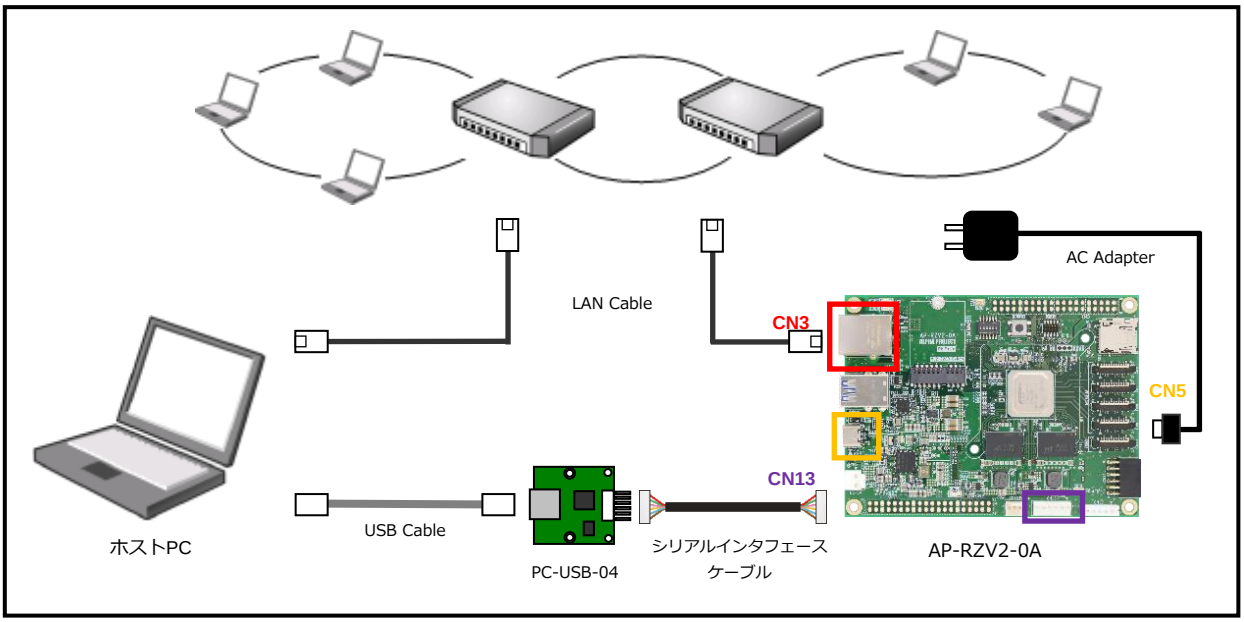
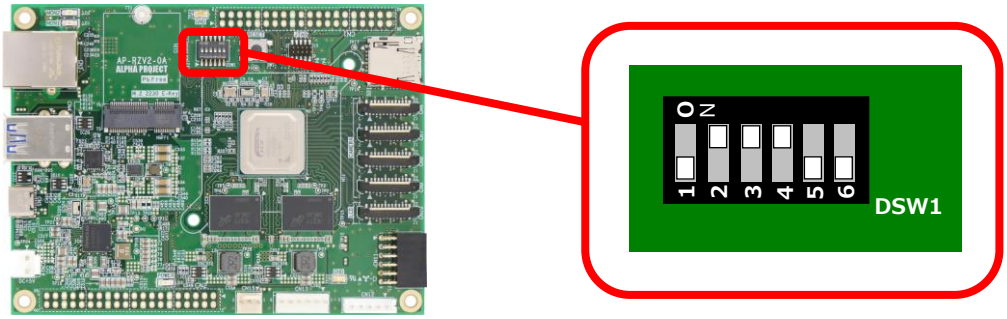


Fig 3.6-2 AP-RZV2-0Aの接続（HUBに接続する場合）

3.7 Linuxの起動

AP-RZV2-0A上でLinuxの起動を行います。

① 電源を入れる前に、AP-RZV2-0A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-RZV2-0Aのハードウェアマニュアルでご確認ください。



② SD1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-RZV2-0Aの接続**』にしたがって、ホストPCとAP-RZV2-0A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『**3.2 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-RZV2-0A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14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):2.7.0/v2h_1.0.3_rc1-dirty
NOTICE:  BL2: Built : 08:37:06, May 31 2024
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.7(release):2.7.0/v2h_1.0.3_rc1-dirty
NOTICE:  BL31: Built : 08:37:06, May 31 2024


U-Boot 2021.10 (Jun 14 2024 - 18:14:19 +0000)

          :
    <途中省略>
          :

Poky (Yocto Project Reference Distro) 3.1.31 rzv2h-aprzv20a ttySC0

BOARD: RZ/V2H AP-RZV2-0A
LSI: RZ/V2H
AI SDK V5.00 (Source Code)
rzv2h-aprzv20a login:
```

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『**rzv2h-aprzv20a 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 3.8-1 ログイン設定

```
rzv2h-aprzv20a login: root
```

時刻設定

AP-RZV2-0A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: Tue 2025-01-07 02:51:58 UTC
           Universal time: Tue 2025-01-07 02:51:58 UTC
                 RTC time: Tue 2025-01-07 02:51:59
                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, MON2）

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
```

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 [heartbea
t] cpu cpu0 cpu1 cpu2 cpu3 default-on mmc0 mmc1
```

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、CN1 3pin (P5_2, gpio-458, gpiochip0 line42)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 P5_2を制御できるようにexportする。

```
# echo 458 > /sys/class/gpio/export
```

② GPIO10_1を出力に設定する。

```
# echo out > /sys/class/gpio/P5_2/direction
```

③ GPIO10_1をHighに設定する。

```
# echo 1 > /sys/class/gpio/P5_2/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-weston」を例に説明を行います。

イメージ	内容
core-image-weston	ai-sdkを使用する標準的なイメージです。
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-weston, 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『**ai_sdk_work/src_setup/yocto**』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 -p ~/ai_sdk_work/src_setup/yocto
```

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_lkrzv2a01_X_X.tar.gz
```

④ 作業ディレクトリに移動します。

```
$ cd ~/ai_sdk_work/src_setup/yocto
```

⑤ AP-RZV2-0A用のBSPを展開します。

```
$ tar zxvf ./bsp_lkrzv2a01_X_X.tar.gz
```

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 ~/ai_sdk_work/src_setup/yocto
```

② ビルド環境の設定をします。

```
$ 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。
実行が完了するとカレントディレクトリは、**build**に移動します。

ビルド

① イメージを作成します。

```
$ MACHINE=rzv2h-aprzv20a bitbake core-image-weston
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 正常に完了しますと、ディレクトリ『~/ai_sdk_work/src_setup/yocto/build/tmp/deploy/images/rzv2h-aprzv20a』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
Flash_Writer_SCIF_RZV2H_UNIVERSAL_INTERNAL_MEMORY.mot	FlashWriterのバイナリファイル
bl2_bp_spi-rzv2h-aprzv20a.srec	BL2のバイナリファイル(SPI Flash用)
bl2_bp_emmc-rzv2h-aprzv20a.bin	BL2のバイナリファイル(eMMC用)
fip-rzv2h-aprzv20a.srec	U-Bootのバイナリファイル
fip-rzv2h-aprzv20a.bin	U-Bootのバイナリファイル
core-image-weston-rzv2h-aprzv20a.tar.bz2	ルートファイルシステム一式(eMMC書込み用)
core-image-weston-rzv2h-aprzv20a.wic.gz	ルートファイルシステム一式(SD書込み用)
core-image-weston-rzv2h-aprzv20a.wic.bmap	SDカードへの書き込み位置情報
Image-rzv2h-aprzv20a.bin	Linuxカーネル
r9a09g057h4-aprzv20a.dtb	デバイスツリー
modules-rzv2h-aprzv20a.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-RZV2-0A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-RZV2-0A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。

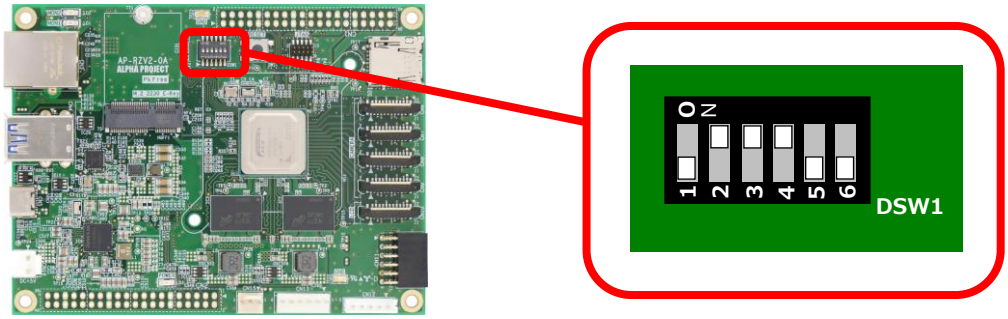


Fig 5.1-1 U-Boot起動設定

② 『**3.6 AP-RZV2-0Aの接続**』にしたがって、ホストPCとAP-RZV2-0A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-RZV2-0A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 作業用ディレクトリに移動します。

```
$ cd ~/ai_sdk_work/src_setup/yocto
```

② ビルド環境の設定をします。

```
$ 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。
実行が完了するとカレントディレクトリは、**build**に移動します。

ビルド

① BL2をビルドします。

```
$ MACHINE=rzv2h-aprzv20a bitbake trusted-firmware-a -c compile --force
```

② BL2をデプロイ(配布)します。

```
$ MACHINE=rzv2h-aprzv20a bitbake trusted-firmware-a -c deploy
```

③ u-bootのビルドをします。

```
$ MACHINE=rzv2h-aprzv20a 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=rzv2h-aprzv20a 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=rzv2h-aprzv20a bitbake firmware-pack -c compile --force
```

⑥ firmware-packをデプロイ(配布)します。

```
$ MACHINE=rzv2h-aprzv20a 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。

```
~/
 `-- ai_sdk_work
     `-- src_setup/
         `-- yocto/
             `-- build
                 `-- tmp
                     `-- work-shared
```

6.2 Linuxカーネルのカスタマイズ

MIPI カメラ、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。

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 ~/ai_sdk_work/src_setup/yocto
```

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、『~/ai_sdk_work/src_setup/yocto/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 作業用ディレクトリに移動します。

```
$ cd ~/ai_sdk_work/src_setup/yocto
```

② ビルド環境の設定をします。

```
$ 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。
実行が完了するとカレントディレクトリは、**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=rzv2h-aprzv20a 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=rzv2h-aprzv20a bitbake linux-renesas -c deploy
```

③ 正常に完了しますと、ディレクトリ『~/ai_sdk_work/src_setup/yocto/build/tmp/deploy/images/rzv2h-aprzv20a』の 以下のファイルが更新されます。

ファイル名	内容
Image-rzv2h-aprzv20a.bin	カーネルイメージファイル
r9a09g057h4-aprzv20a.dtb	デバイスツリーファイル
modules-rzv2h-aprzv20a.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。

6.4 Linuxカーネルのデバイスドライバ

AP-RZV2-0A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	GPIOクラス
QSPI Flash	MTDクラス
Watchdog	Watchdogクラス
温度センサー	Thermal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。

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 ~/ai_sdk_work/src_setup/yocto
```

② ビルド環境の設定をします。

```
$ 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。
実行が完了するとカレントディレクトリは、**build**に移動します。

ビルド

① bitbakeコマンドでSDKを作成します。

```
$ MACHINE=rzv2h-aprzv20a bitbake core-image-weston -c populate_sdk -k
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-RZV2-0A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『**aprzv20a-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 ~/aprzv20a-app
```

② ディレクトリ『**aprzv20a-app**』に移動します。

```
$ cd ~/aprzv20a-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』

② 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/3.1.31/environment-setup-aarch64-poky-linux
```

最初の『**.**』と『**/opt/poky/3.1.31/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/3.1.31/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-RZV2-0A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-RZV2-0Aへ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
Flash_Writer_SCIF_RZV2H_UNIVERSAL_INTERNAL_MEMORY.mot	FlashWriterのバイナリファイル
bl2_bp_spi-rzv2h-aprzv20a.srec	BL2のバイナリファイル
fip-rzv2h-aprzv20a.srec	U-Bootのバイナリファイル

Table 10.2-1 必要なファイル(QSPI Flash)

ファイル名	内容
core-image-weston-rzv2h-aprzv20a.wic.gz	microSD用　ルートファイルシステム一式
core-image-weston-rzv2h-aprzv20a.wic.bmap	SDカードへの書き込み位置情報
Image-rzv2h-aprzv20a.bin	Linuxカーネルファイル
r9a09g057h4-aprzv20a.dtb	デバイスツリーファイル
modules-rzv2h-aprzv20a.tgz	モジュールファイル一式
bl2_bp_emmc-rzv2h-aprzv20a.bin	eMMC用　BL2のバイナリファイル
fip-rzv2h-aprzv20a.bin	U-Bootのバイナリファイル
core-image-weston-rzv2h-aprzv20a.tar.bz2	ルートファイルシステム一式
Image-rzv2h-aprzv20a.bin	Linuxカーネルファイル
r9a09g057h4-aprzv20a.dtb	デバイスツリーファイル
modules-rzv2h-aprzv20a.tgz	モジュールファイル一式

Table 10.2-2 必要なファイル(eMMC)

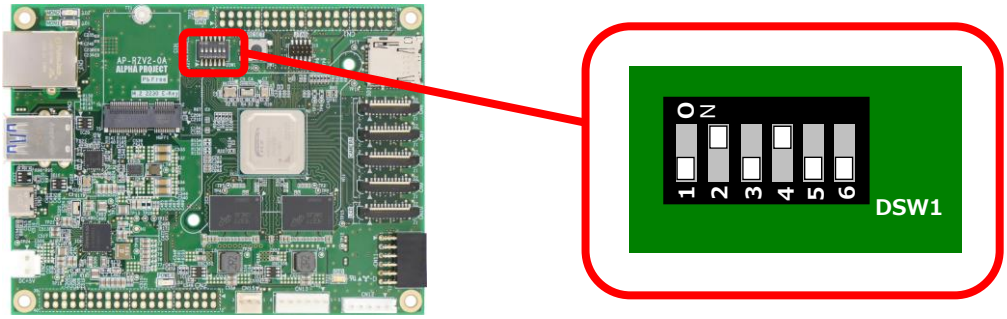
上記の表の同じ名前のファイルは、説明上分けていますが同じファイルです。
（Linuxカーネル，デバイスツリー，モジュールファイル）

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-RZV2-0A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-RZV2-0A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-RZV2-0Aの接続**』にしたがって、ホストPCとAP-RZV2-0A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-RZV2-0A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
送信改行コード	CR

Table 10.3-1 シリアル設定(FlashWriter起動時)

⑤ ACアダプタを接続して、AP-RZV2-0A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

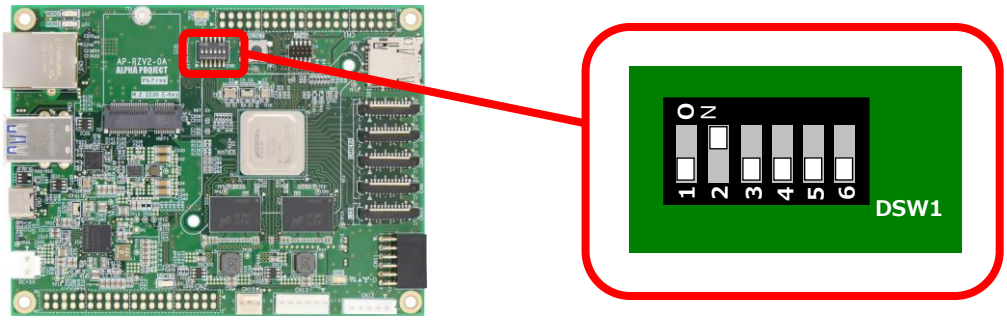
eMMCから起動するためには、以下の構成に合わせて、データが書き込まれている必要があります。

●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-rzv2h-aprzv20a.bin	書き込み開始セクタ :0x0001
	fip-rzv2h-aprzv20a.bin	書き込み開始セクタ: 0x0300
mmcblk0boot1	–	未使用
mmcblk0p1	core-image-weston-rzv2h-aprzv20a.tar.bz2	ルートディレクトリに展開
	modules-rzv2h-aprzv20a.tgz	ルートディレクトリに展開
	Image-rzv2h-aprzv20a.bin	/bootディレクトリにコピー
	r9a09g057h4-aprzv20a.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-RZV2-0AのSDカードスロット（SD1）に差し込みます。
AP-RZV2-0Aのスイッチを下記の設定にし、電源を入れます。



② ログインプロンプト『**rzv2h-aprzv20a login:**』に『**root**』と入力しログインします。

```
rzv2h-aprzv20a login:root
```

③ eMMC用データが格納されたディレクトリに移動します。

```
# cd ~/binaries
```

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