# RZ/V2L, RZ/V2M, RZ/V2MA AI EVALUATION SOFTWARE GUIDE

REV.7.20
SEPTEMBER 2022

本書は、RZ/V2L、RZ/V2M、RZ/V2MA向けのAI Evaluation Softwareについて説明します。
本書では、RZ/V2L、RZ/V2M、RZ/V2MAをRZ/V2xと記載しています。

R11AN0624JJ0720

# 目次


- 概要 P3
- Networkブート環境構築手順 P11
- Ubuntu PCシリアル通信実行手順 P22
- ブート手順（RZ/V2L） P24
- ブート手順（RZ/V2M, RZ/V2MA） P28
- アプリケーション実行手順 P32
- 別紙：SDカードブート方式 P43
  - SDカード作成手順 P48
  - シリアルポートドライバインストール P52
  - SDブート手順（RZ/V2L） P53
  - SDブート手順（RZ/V2M, RZ/V2MA） P56

# 概要

# AI EVALUATION SOFTWAREとは

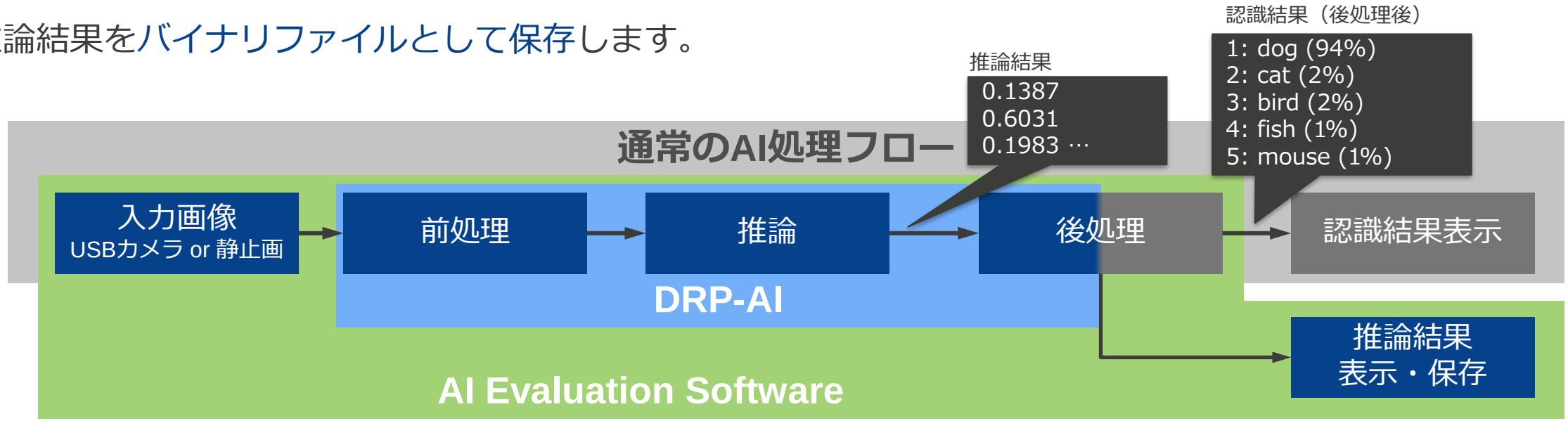
**AI Evaluation Software**とは、画像認識AIモデルのDRP-AI動作確認用環境です。

USBカメラ画像や静止画像に対して任意のAIモデル推論を実行します。

通常ではAI推論の後、認識結果を取得するための後処理が実行されますが、

AIの後処理は実行するAIモデルに依存するため、本ソフトウェアでは後処理を実行せず、

推論結果をバイナリファイルとして保存します。



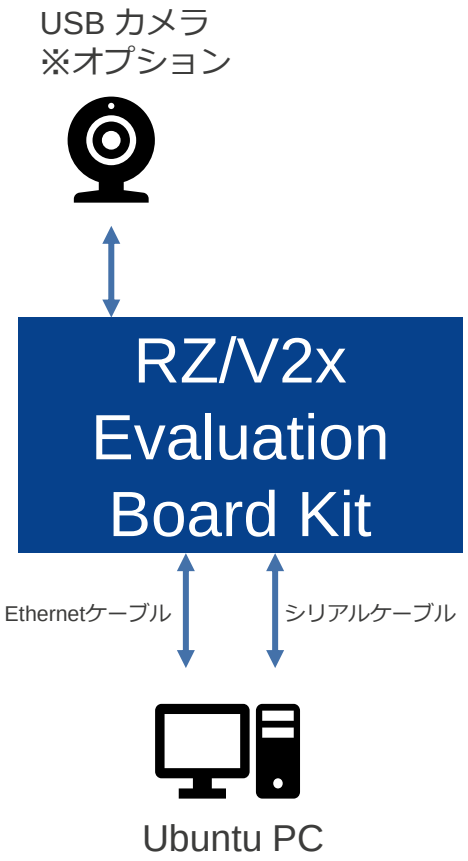
# ブート方法について

本アプリケーションは2通りのブート方法を使用することができます。
本書では①Networkブートを想定した手順をご説明します。②SDカードブート方式は別紙をご参照ください。

## ①Networkブート

Ubuntu PCのサーバ上に
ファイルシステムを構築し、
Ethernet経由で
このファイルシステムに
アクセスするブート方式です。

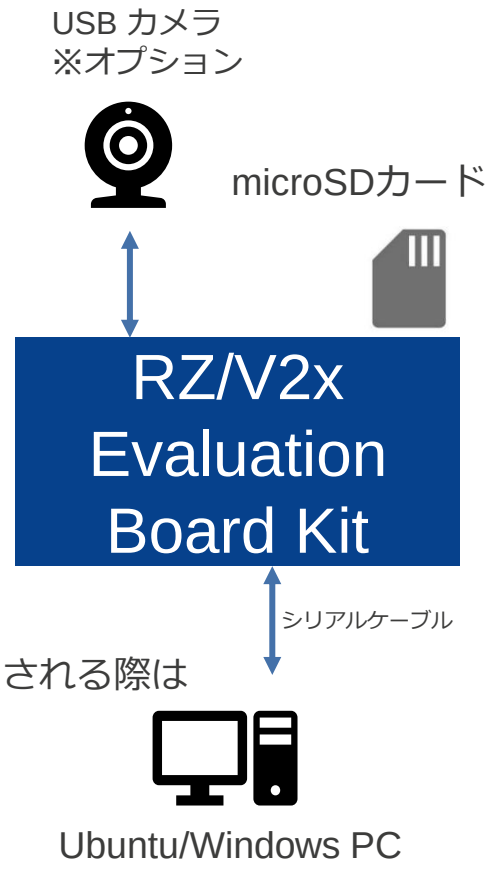
大量の画像などを扱いたい場合は
こちらの方式をお選びください。




## ② SDカードブート

SDカード上のファイルシステムを
使用するブート方式です。

本アプリケーションの手軽なお試しを希望される際は
こちらの方式をお選びください。



# 必要ファイル

本書の手順を実行する前にあらかじめ以下のファイルをご用意ください。

| 名称 | ファイル名 | | 詳細 |
|---|---|---|---|
| AI Evaluation Software | RZ/V2L | rzv2l_ai-evaluation-software_ver7.20.tar.gz | AI Evaluation Software環境一式 |
| | RZ/V2M | rzv2m_ai-evaluation-software_ver7.20.tar.gz | |
| | RZ/V2MA | rzv2ma_ai-evaluation-software_ver7.20.tar.gz | |
| AI Implementation Guide | r11an0616jj0720-rzv-ai-imp-getstarted.pdf | | AIモデルのDRP-AI実装方法ガイド |
| DRP-AI Translator | DRP-AI_Translator-v1.80-Linux-x86_64-Install | | DRP-AI Object filesを作成するツール<br>使用方法はAI Implementation Guideを参照 |

# 必要ファイル

AI Evaluation Softwareの詳細は以下の通りです。
本パッケージが提供するファイルはLinux Package及びDRP-AI Support Package v7.20のプリビルドバイナリです。

| 名称 | | ファイル名 | 詳細 |
|---|---|---|---|
| RZ/V2L Linux Package v3.0.0 | Linuxカーネルイメージ | Image-smarc-rzv2l.bin | ブート用プログラム |
| | Linuxデバイスツリーファイル | Image-r9a07g054l2-smarc.dtb | ブート用設定ファイル |
| | ルートファイルシステム一式 | core-image-weston-smarc-rzv2l.tar.bz2 | AI Evaluation Software実行環境を含むファイルシステム |

| 名称 | | ファイル名 | 詳細 |
|---|---|---|---|
| RZ/V2M Linux Package v1.3.0※ | Linuxカーネルイメージ | Image-rzv2m.bin | ブート用プログラム |
| | Linuxデバイスツリーファイル | r9a09g011gbg-evaluation-board.dtb | ブート用設定ファイル |
| | ルートファイルシステム一式 | core-image-bsp-rzv2m.tar.bz2 | AI Evaluation Software実行環境を含むファイルシステム |

| 名称 | | ファイル名 | 詳細 |
|---|---|---|---|
| RZ/V2MA Linux Package v1.0.0 | Linuxカーネルイメージ | Image-rzv2ma.bin | ブート用プログラム |
| | Linuxデバイスツリーファイル | r9a09g055ma3gbg-evaluation-board.dtb | ブート用設定ファイル |
| | ルートファイルシステム一式 | core-image-bsp-rzv2ma.tar.bz2 | AI Evaluation Software実行環境を含むファイルシステム |

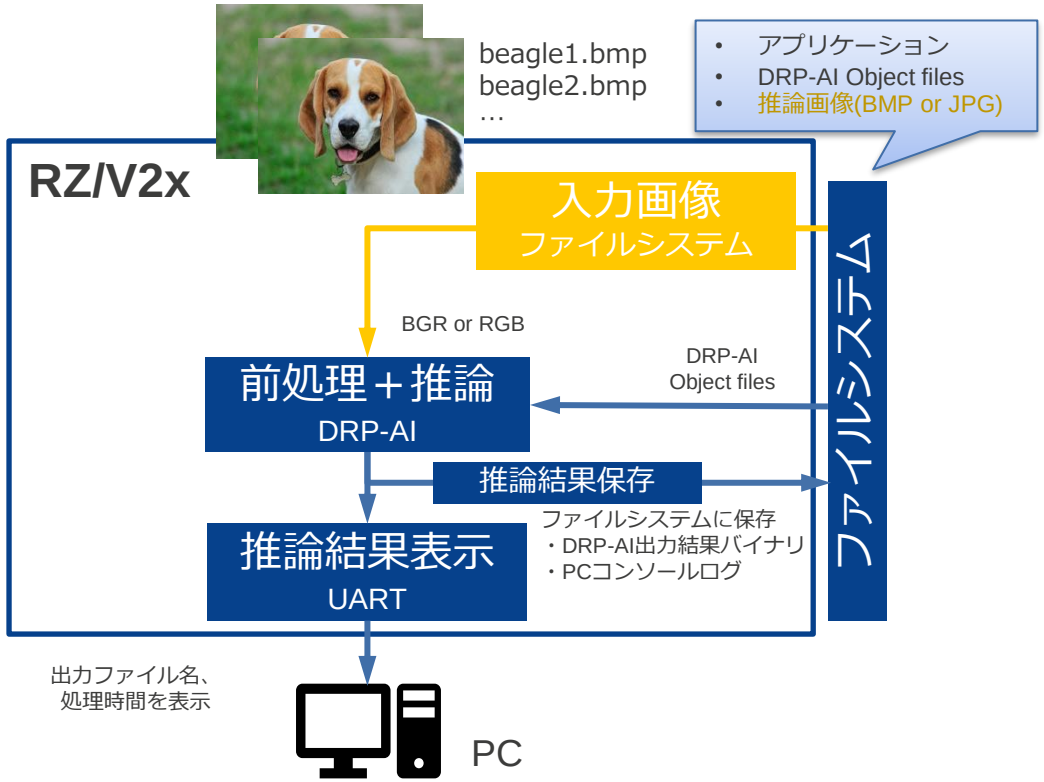
※：2022年10月提供予定

# 入力画像について

本アプリケーションには2種類の入力画像モードがあります。
入力画像はVGA サイズのみに対応しています。

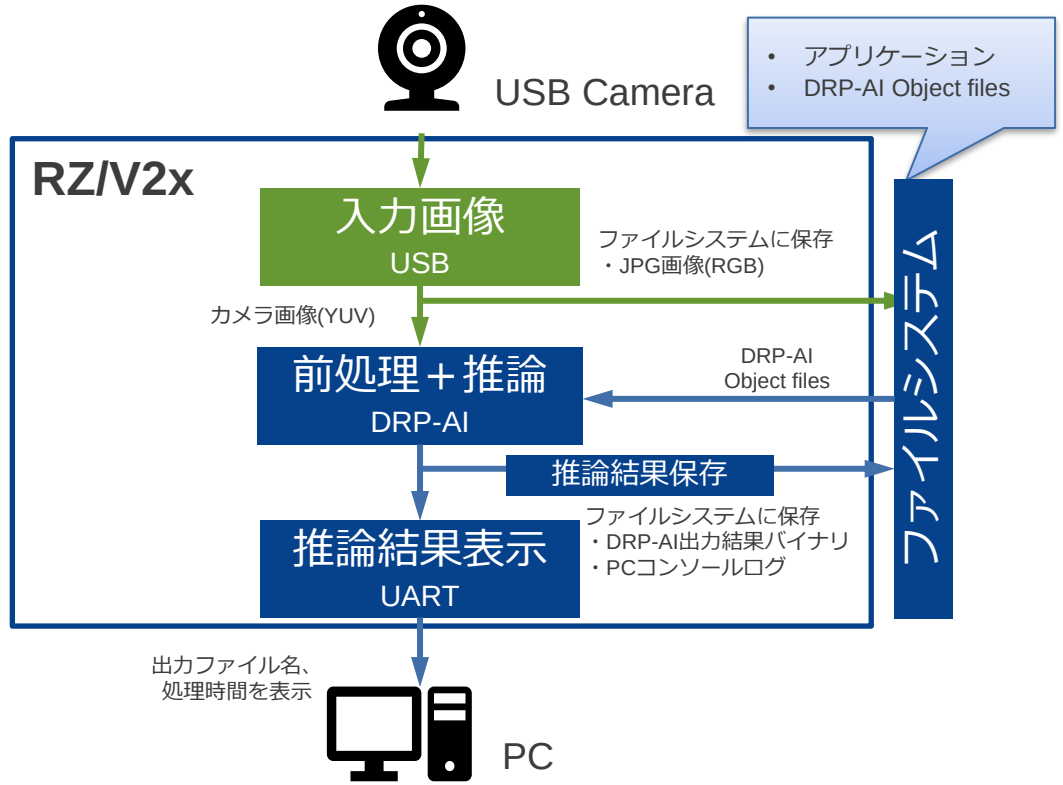
## ① Imageモード

ファイルシステム上の画像ファイルを使用します。
特定フォルダ内の複数の画像を連続で推論します。




## ② Cameraモード

USBカメラキャプチャ画像を使用します。
ユーザーの任意のタイミングで推論処理を実行します。



# 入出力データ

各モードの入出力ファイル・表示内容は以下の通りです。

| No. | 機能 | Imageモード | Cameraモード |
|---|---|---|---|
| 1 | 入力データ | **BMP画像ファイル** ：Windows Bitmap v3, v4, v5に対応。BGR<br>**JPG画像ファイル** ：jpg拡張子のみに対応。RGB<br>入力画像枚数上限 ：20000枚。 | **USBカメラキャプチャ画像** ：アプリケーションがキャプチャ。YUY2 |
| | | 入力画像は両モードとも、VGAサイズ（640x480）のみに対応。 | |
| 2 | DRP-AI処理 | ファイルシステム上のDRP-AI Object files(DRP-AI Translator出力結果の以下6ファイルを使用する)<br>① *_addr_intm.txt<br>② *_weight.dat<br>③ *_drpcfg.mem<br>④ drp_param.bin<br>⑤ aimac_desc.bin<br>⑥ drp_desc.bin | |
| 3 | 結果表示 | UARTでPCコンソール上に以下を表示<br>• DRP-AI Object files名<br>• **入力画像ファイル名**<br>• DRP-AIの処理時間<br>• DRP-AI推論処理の出力結果バイナリファイル名<br>• コンソール表示内容を保存したログファイル名 | UARTでPCコンソール上に以下を表示<br>• DRP-AI Object files名<br>• DRP-AIの処理時間<br>• DRP-AI推論処理の出力結果バイナリファイル名<br>• **カメラのキャプチャ画像JPGファイル名**<br>• コンソール表示内容を保存したログファイル名 |
| 4 | 出力データ | <TIME>.log ：No.3のコンソール出力と同内容のログファイル<br><IMAGE_NAME>.bin ：DRP-AI処理の出力結果バイナリファイル | <TIME>.log ：No.3のコンソール出力と同内容のログファイル<br>output<N>_<TIME>.bin ：DRP-AI処理の出力結果バイナリファイル<br>**output<N>_<TIME>.jpg** ：USBカメラキャプチャ画像JPGファイル |

# 【補足】コマンド実行環境について

本書ではLinuxの実行コマンドを使用して手順を説明します。

コマンドの記述は以下のコマンド表記を使用します。

```
$ コマンド
```

また、異なる実行環境でコマンドを実行するため、以下のように実行環境を区別しています。

1. Ubuntu PC

   ```
   $ printenv
   ```

2. Evaluation Board Kit 通常コンソール
   シリアル通信端末に関係なく以下のように表記します。

   ```
   # printenv
   ```

3. Evaluation Board Kit U-bootコンソール（ブート環境変数設定時のみ使用）
   シリアル通信端末に関係なく以下のように表記します。

   ```
   => printenv
   ```

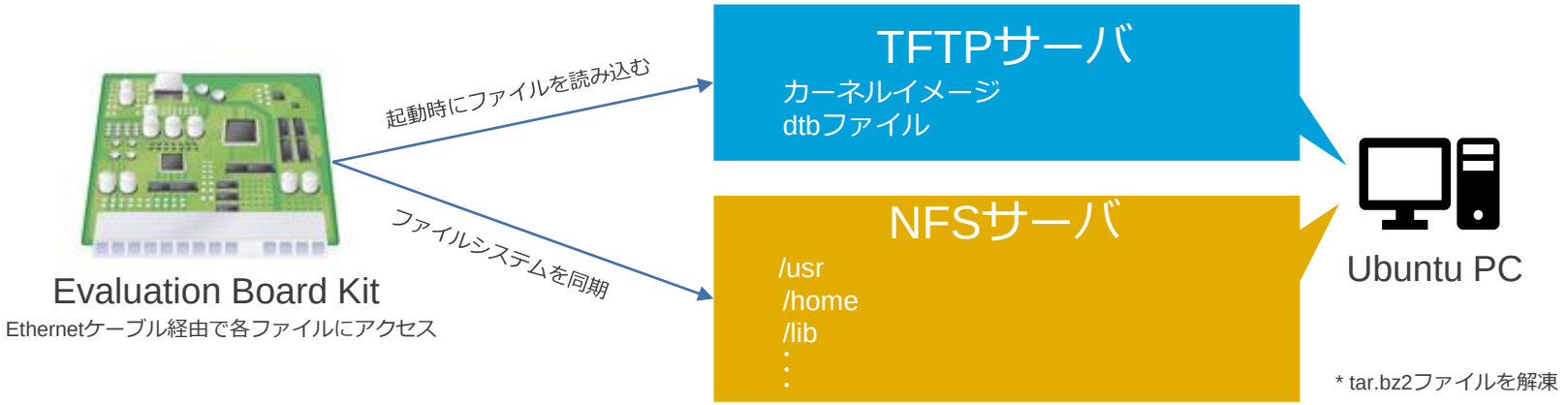
# NETWORKブート環境構築手順

本章はNetworkブート方式を想定しています。
SDカードブート方式については別紙をご参照ください。

# NETWORKブートとは

Networkブートとはボード上のメモリに展開するファイルをUbuntu PCサーバでマウントするブート方式です。
ブートに必要なデータは以下の3種類です。各マウントサーバ上に配置します。

| ファイル | 説明 | マウントサーバ |
|---|---|---|
| カーネルイメージ | Linuxカーネルイメージ（ブートプログラム） | Ubuntu上TFTPサーバ |
| dtbファイル | Linuxデバイスツリーファイル（ブート用設定ファイル） | Ubuntu上TFTPサーバ |
| tar.bz2ファイル | ルートファイルシステム一式 | Ubuntu上NFSサーバ |




- 本章では各サーバのセットアップ方法とファイルの配置方法をご説明します。

# 必要機材（RZ/V2L）

Networkブート方式で必要な機材及び動作確認済み環境を以下に示します。

| 必要機材 | 詳細 | 入手方法 |
|---|---|---|
| RZ/V2L Evaluation Board Kit | LinuxブートローダーをeMMCに書き込み済み<br>※ブートローダーの書き込み方法については、RZ/V2L Linux Package Start-up Guideを参照してください。 | ルネサスから提供 |
| シリアルケーブル | Linux PC＋RZ/V2L Evaluation Board間のシリアル通信に使用 | |
| AC adapter | 電源 | お客様にて<br>ご用意ください |
| USBケーブル Type-C | AC adapterとRZ/V2L Evaluation Boardを接続 | |
| Ethernetケーブル | Linux PC＋RZ/V2L Evaluation Board間のEthernet通信に使用 | |
| Linux PC | Ubuntu 20.04をインストール済み<br>動作確認済み環境：Intel Core i5 11th generation メモリ16GB | |
| USBカメラ | 動作確認済み環境：Logitech C930E WEBCAM | |

# 必要機材（RZ/V2M, RZ/V2MA）

Networkブート方式で必要な機材及び動作確認済み環境を以下に示します。

| 必要機材 | 詳細 | 入手方法 |
|---|---|---|
| RZ/V2x Evaluation Board Kit | Linuxブートローダーをe MMCに書き込み済み<br>※ブートローダーの書き込み方法については、RZ/V2x Linux Package Start-up Guideを参照してください。 | ルネサスから提供 |
| シリアルケーブル | Linux PC＋RZ/V2x Evaluation Board Kit間のシリアル通信に使用 | お客様にて<br>ご用意ください |
| AC adapter | 電源 | |
| Ethernetケーブル | Linux PC＋RZ/V2x Evaluation Board Kit間のEthernet通信に使用 | |
| Linux PC | Ubuntu 20.04をインストール済み<br>動作確認済み環境：Intel Core i5 11th generation メモリ16GB | |
| USBカメラ | 動作確認済み環境：Logicool C930e ウェブカム ※オプション | |
| Type-C変換アダプタ | RZ/V2x Evaluation Board KitのUSBポートがType-Cのため ※オプション<br>動作確認済み環境：SANWA SUPPLY USB A Type C変換アダプタAD-USB28CAF | |

# 必要ソフトウェア

- 本手順では以下のファイルを使用します。入手方法は必要ファイルをご参照ください。
  - カーネルイメージ
  - dtbファイル
  - tar.bz2ファイル

- 本書では以下のようにテキストエディタとしてvimを使用しています。適宜お好みのエディタをご使用ください。

```
$ vi hello.txt
```

- 事前にUbuntu PCに共通必要パッケージをインストールしてください。

```
$ sudo apt-get update
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat cpio python python3 python3-pip
$ sudo apt-get install python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm
$ sudo apt-get install python3-subunit mesa-common-dev p7zip-full
```

- Networkブート方式で必要な追加パッケージを事前に以下のコマンドでインストールしてください。

```
$ sudo apt-get install tftp tftpd-hpa nfs-common nfs-kernel-server
```

- Ubuntu PCでのシリアル通信用に、以下のコマンドで必要パッケージを事前にインストールしてください。

```
$ sudo apt-get install cu
```

# TFTPサーバの構築

本項はUbuntu PCでの作業です。

本項では以下のソフトウェアを使用します。必要ソフトウェアを参考に、事前にインストールしてください。

- tftp
- tftpd-hpa

1. TFTPサーバ用ディレクトリを作成します。

```
$ sudo mkdir /tftpboot
```

※本ディレクトリ名は別途ブート時に使用するため、必ず上記の名称をご使用ください。

2. TFTPサーバ設定を作成します。

/etc/default/tftpd-hpaを開き、右記のように修正してください。

```
$ sudo vi /etc/default/tftpd-hpa
```

3. TFTPサーバを起動します

```
$ sudo systemctl enable tftpd-hpa
$ sudo systemctl restart tftpd-hpa
```

2．TFTPサーバ設定　変更前

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

2．TFTPサーバ設定　変更後

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/tftpboot"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

# NFSサーバの構築

本項はUbuntu PCでの作業です。

本項では以下のソフトウェアを使用します。必要ソフトウェアを参考に、事前にインストールしてください。

- nfs-common
- nfs-kernel-server

1. NFSサーバ用ディレクトリを作成します。 ※本ディレクトリ名は別途ブート時に使用するため、必ず下記の名称をご使用ください。

RZ/V2Lの場合:

```
$ sudo mkdir -p /nfs/rzv2l
```

RZ/V2M、 RZ/V2MAの場合:

```
$ sudo mkdir -p /nfs/rzv2m
```

2. NFSサーバを起動します。

```
$ sudo /etc/init.d/nfs-kernel-server start
```

3. NFSサーバ設定を変更します。
/etc/exportsのファイル末尾に以下のように追記してください。

```
$ sudo vi /etc/exports
```

RZ/V2Lの場合:

```
…
/nfs/rzv2l *(rw,no_subtree_check,sync,no_root_squash)
```

RZ/V2M、RZ/V2MAの場合:

```
…
/nfs/rzv2m *(rw,no_subtree_check,sync,no_root_squash)
```

4. NFSサーバを更新します。

```
$ sudo exportfs -a
```

# サーバ起動を確認

本項はUbuntu PCでの作業です。

1. TFTPサーバが正常に起動できたかは以下のように確認することができます。

```
$ sudo chmod 777 /tftpboot
$ sudo echo "Hello World" > /tftpboot/hello.txt
$ sudo tftp localhost
> get hello.txt
```

⇒エラーなくコマンドを実行できたらTFTPサーバは正常に起動できています。

2. NFSサーバが正常に起動できたかは以下のように確認することができます。

RZ/V2Lの場合:

```
$ showmount -e localhost
Export list for localhost:
/nfs/rzv2l *
```

RZ/V2M、RZ/V2MAの場合:

```
$ showmount -e localhost
Export list for localhost:
/nfs/rzv2m *
```

⇒上記と同じように表示されたらNFSサーバは正常に起動できています。

上記の結果が確認できなかった場合、Ubuntu PCを再起動して再度ご確認ください。

# IPアドレスの固定化

本項はUbuntu PCでの作業です。

Ethernet経由でUbuntu PCとボードが通信するためにはUbuntu PCのIPアドレスを固定化する必要があります。

1. デフォルトのネットワーク設定を無効化します。（yamlファイル名は環境によって異なることがあります。）

```
$ sudo mv /etc/netplan/01-network-manager-all.yaml /etc/netplan/01-network-manager-all.yaml.disabled
```

2. 99-netcfg.yamlを新規作成します。

```
$ sudo vi /etc/netplan/99-netcfg.yaml
```

3. 右記のように記入します。

   （環境によってenp0s3の名称が異なることがあります。）

4. ネットワークを再起動します。

```
$ sudo netplan apply
```

3. 99-netfcfg.yaml記入内容

```
network:
    version: 2
    ethernets:
        enp0s3:
            addresses: [192.168.1.10/24]
            gateway4: 192.168.1.1
            nameservers:
                addresses: [192.168.1.1]
                search: []
            optional: true
```

※IPアドレスは別途ブート時に使用するため、
必ず上記を指定してください。
また、空白、インデントが異なるとエラーに
なるため、上記の通りに記入してください。

# 必要ファイルの展開

本項はUbuntu PCでの作業です。

1. TFTPサーバに以下の2ファイルを展開します。下記コマンドを実行してください。

| Board | ファイル | ファイル名 |
|---|---|---|
| RZ/V2L | カーネルイメージ | Image-smarc-rzv2l.bin |
| | dtbファイル | Image-r9a07g054l2-smarc.dtb |
| RZ/V2M | カーネルイメージ | Image-rzv2m.bin |
| | dtbファイル | r9a09g011gbg-evaluation-board.dtb |
| RZ/V2MA | カーネルイメージ | Image-rzv2ma.bin |
| | dtbファイル | r9a09g055ma3gbg-evaluation-board.dtb |

```
$ sudo cp <PATH_to_FILE>/<カーネルイメージ> /tftpboot
$ sudo cp <PATH_to_FILE>/<dtbファイル> /tftpboot
```

※1. <PATH_to_FILE> は各ファイルへのパスです。適宜変更してください。
2. <カーネルイメージ>, <dtbファイル>は上記の表を参照して変更してください。

# 必要ファイルの展開

2. NFSサーバにファイルシステムを展開します。下記コマンドを実行してください。

RZ/V2Lの場合:

```
$ sudo tar xfj <PATH_to_FILE>/core-image-weston-smarc-rzv2l.tar.bz2 -C /nfs/rzv2l
```

RZ/V2Mの場合:

```
$ sudo tar xfj <PATH_to_FILE>/core-image-bsp-rzv2m.tar.bz2 -C /nfs/rzv2m
```

RZ/V2MAの場合:

```
$ sudo tar xfj <PATH_to_FILE>/core-image-bsp-rzv2ma.tar.bz2 -C /nfs/rzv2m
```

※ <PATH_to_FILE> は各ファイルへのパスです。適宜変更してください。

# UBUNTU PCシリアル通信実行手順

# UBUNTU＋ボード間シリアル通信

本項はUbuntu PCでの作業です。

NetworkブートではUbuntu PCとボードでシリアル通信をすると想定しています。

- 本章では以下のソフトウェアを使用します。必要ソフトウェアを参考に事前にインストールしてください。
  - cu
- 実際のシリアル通信方法を説明します。

1. Ubuntu PCとボードをシリアルケーブルで接続します。
2. 以下のコマンドを実行し、シリアルポート名を取得します。RZ/V2Mは２ポートありますが、ttyUSB1を使用してください。

```
$ ls -l /dev/serial/by-id/
total 0
… -> ../../ttyUSB0
```

3. シリアルポートのアクセス権限を変更します。RZ/V2MはttyUSB1を使用してください。

```
$ sudo chmod 666 /dev/ttyUSB0
```

4. 以下のコマンドを実行してシリアル通信を開始します。RZ/V2MはttyUSB1を使用してください。

```
$ cu -s 115200 -l /dev/ttyUSB0 --parity none --nostop
```

※通信を終了するときは「~.」と入力してください。

5. Ubuntu PCで別のコンソールを起動し、"crtscts" オプションを変更します。RZ/V2MはttyUSB1を使用してください。

```
$ stty -F /dev/ttyUSB0 -crtscts
```

# ブート手順（RZ/V2L）

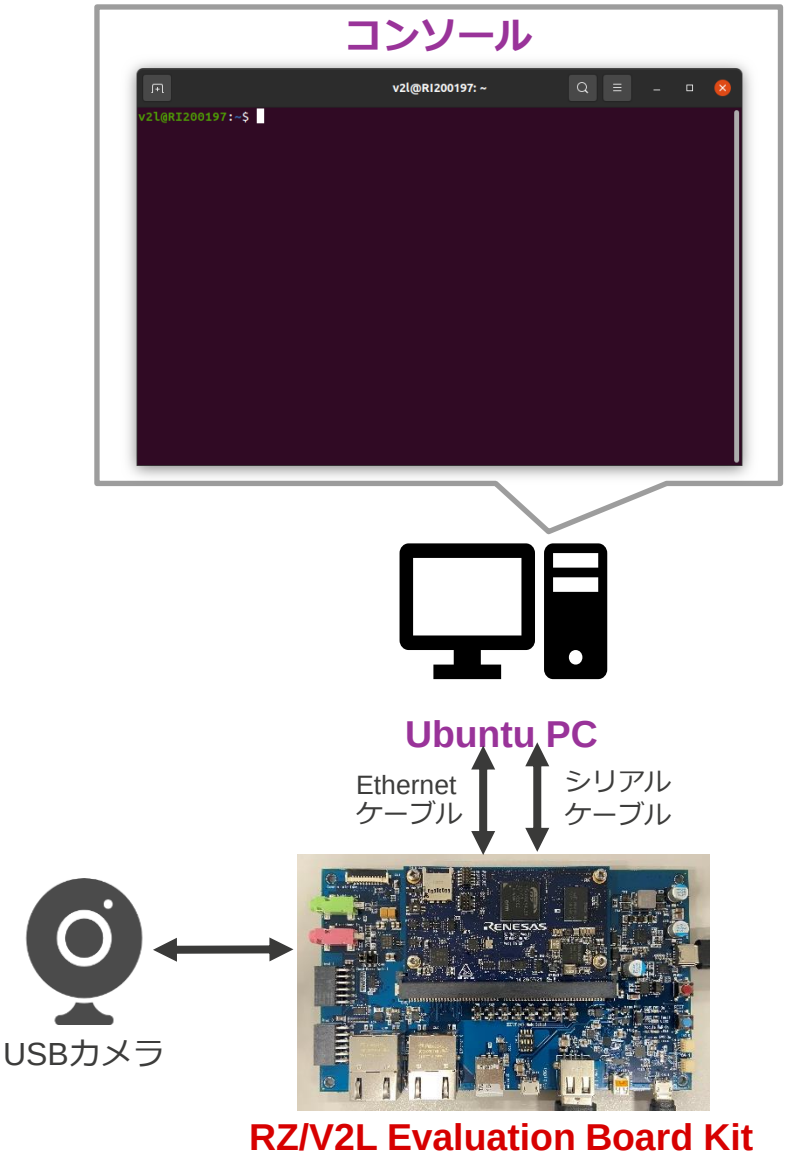
# ブート手順概要　Networkブート

■手順概要　（詳細は次ページ）

前提：以下の事前セットアップを全て実施済み

- Networkブート環境構築手順
- シリアル通信実行手順

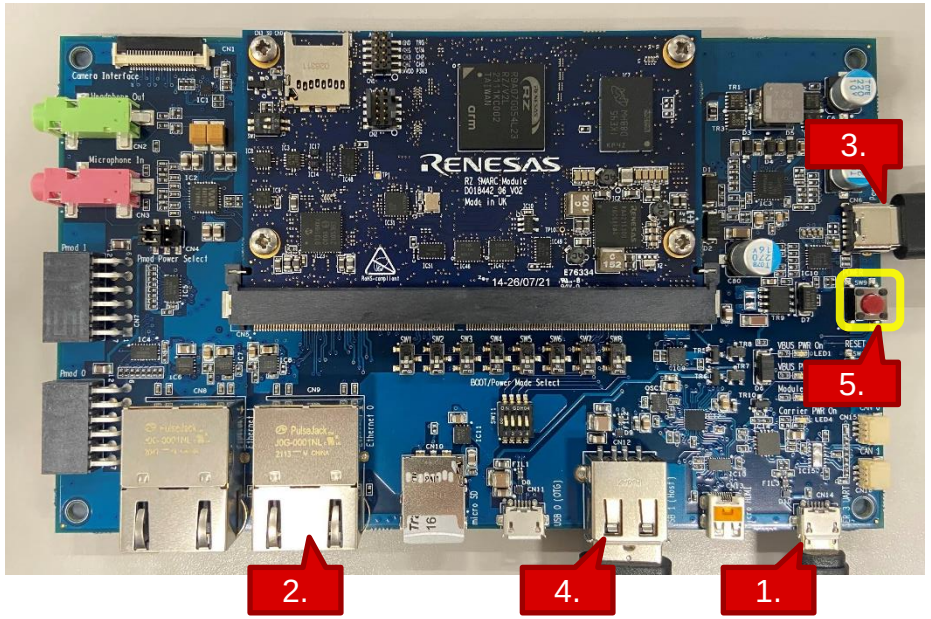
1. **ボード**と**Ubuntu PC**をシリアル-USB変換ケーブルでつなぎます。
2. **ボード**と**Ubuntu PC**をEthernetケーブルでつなぎます。
3. **ボード**の電源ケーブルをつなぎます。
4. **ボード**と**USBカメラ**をつなぎます。　※Cameraモード使用時のみ
5. **ボード**の電源を入れます。
6. **Ubuntu PC**上で**コンソール**を起動し、シリアル接続をします。
7. **コンソール**上でU-boot環境変数を変更します。 ※ブート方式変更時のみ
8. ブート後、**コンソール**上でシステムにログインします。



# ブート手順　詳細

■前提：事前セットアップを全て実施済み

1. **ボード**と**Ubuntu PC**をシリアル-USB変換ケーブルでつなぎます。
2. **ボード**と**Ubuntu PC**をEthernetケーブルでつなぎます
3. **ボード**の電源ケーブルをつなぎます。
4. **ボード**と **USB カメラ** をつなぎます。※Camera モード使用時のみ
5. **ボード**の電源を入れます。
6. **Ubuntu PC**上で**コンソール**を起動し、シリアル接続をします。
   Ubuntu＋ボード間シリアル通信を参照



# ブート手順　詳細

7. **コンソール**上でU-boot環境変数を変更します。※ブート方式変更時のみ
   1. 電源を入れた直後、**コンソール**上でENTERキーを押し続けます。
   2. U-bootコンソールが起動します。
   3. 以下を入力します。

```
=> env default -a
=> setenv ipaddr 192.168.1.11
=> setenv serverip 192.168.1.10
=> setenv netmask 255.255.255.0
=> setenv ethaddr 02:11:22:33:44:55
=> setenv boot_tftp 'tftpboot 0x48080000 Image-smarc-rzv2l.bin; tftpboot 0x48000000 Image-r9a07g054l2-smarc.dtb; booti 0x48080000 - 0x48000000'
=> setenv bootargs root=/dev/nfs rw nfsroot=${serverip}:/nfs/rzv2l,nfsvers=3 ip=${ipaddr}:${serverip}::${netmask}:rzv2l:eth0
=> setenv bootcmd run boot_tftp
=> saveenv
=> boot
```

※serverip = Ubuntu PCのIPアドレス
ipaddr = ボードのIPアドレス
Ubuntu PCのIPアドレスはIPアドレスの固定化で
設定したものを使用しています。

8. **コンソール**上にログイン画面が表示されたら、以下の通りログインしてください。
   - user: root
   - password: なし

# ブート手順（RZ/V2M, RZ/V2MA）

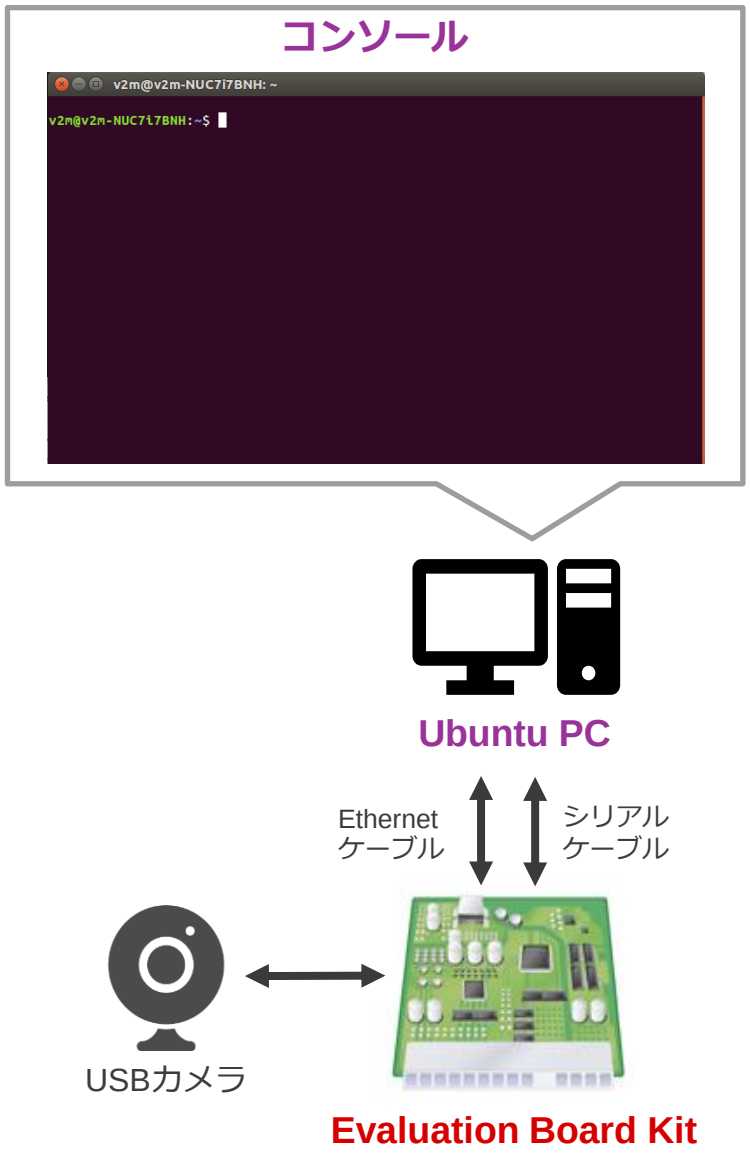
# ブート手順概要　Networkブート

■手順概要　（詳細は次ページ）

前提：以下の事前セットアップを全て実施済み

- Networkブート環境構築手順
- シリアル通信実行手順

1. **ボード**と**Ubuntu PC**をシリアル-USB変換ケーブルでつなぎます。
2. **ボード**と**Ubuntu PC**をEthernetケーブルでつなぎます
3. **ボード**の電源ケーブルをつなぎます。
4. **ボード**と**USBカメラ**をつなぎます。　※Cameraモード使用時のみ
5. **Ubuntu PC**上で**コンソール**を起動し、シリアル接続をします。
6. **ボード**の電源を入れます。
7. **コンソール**上でU-boot環境変数を変更します。 ※ブート方式変更時のみ
8. ブート後、**コンソール**上でシステムにログインします。

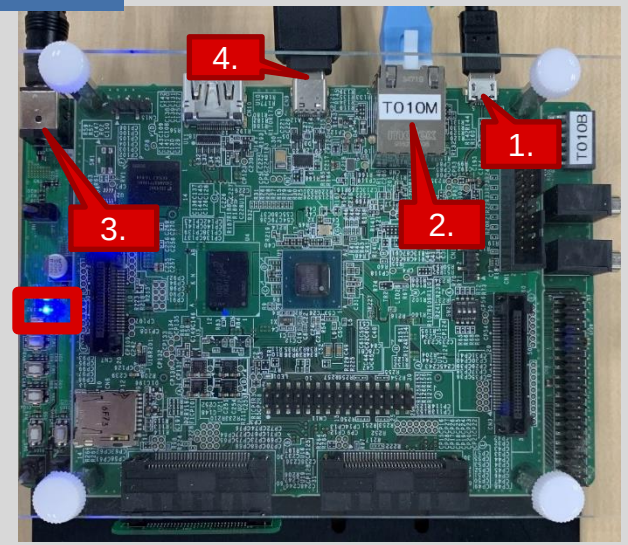


# ブート手順　詳細

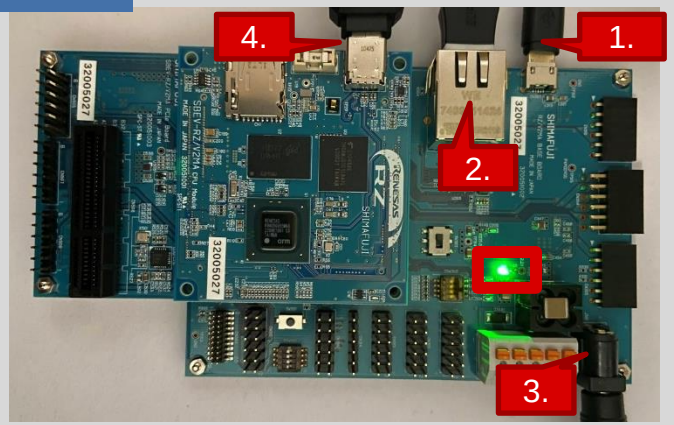
■前提：事前セットアップを全て実施済み

1. **ボード**と**Ubuntu PC**をシリアル-USB変換ケーブルでつなぎます。
2. **ボード**と**Ubuntu PC**をEthernetケーブルでつなぎます
3. **ボード**の電源ケーブルをつなぎます。※LEDが点灯する
4. **ボード**と**USBカメラ**をつなぎます。　※Cameraモード使用時のみ。
5. **Ubuntu PC**上で**コンソール**を起動し、シリアル接続をします。
   Ubuntu＋ボード間シリアル通信を参照

RZ/V2M




RZ/V2MA



# ブート手順　詳細

6. **ボード**の電源を入れます。※LEDがさらに点灯する

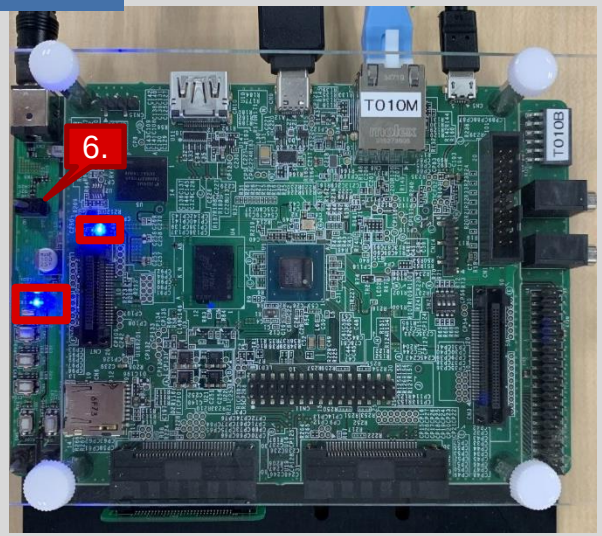
7. **コンソール**上でU-boot環境変数を変更します。※ブート方式変更時のみ
   1. 6で電源を入れた直後、**コンソール**上でENTERキーを押し続けます。
   2. U-bootコンソールが起動します。
   3. 以下を入力します。

```
=> env default -a
=> setenv ipaddr 192.168.1.11
=> setenv serverip 192.168.1.10
=> setenv bootcmd run bootnfs
=> saveenv
=> boot
```

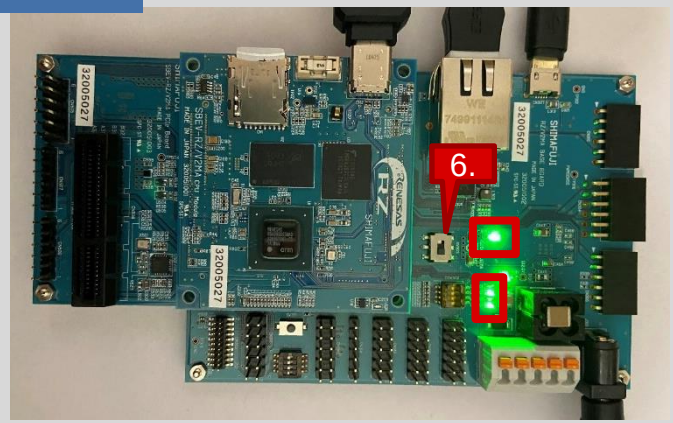
※serverip = Ubuntu PCのIPアドレス
ipaddr = ボードのIPアドレス
Ubuntu PCのIPアドレスはIPアドレスの固定化で設定したものを使用しています。

8. **コンソール**上にログイン画面が表示されたら、以下の通りログインしてください。
   - user: root
   - password: なし

RZ/V2M




RZ/V2MA



# アプリケーション実行手順

# 実行環境

アプリケーション実行環境の構成を以下に示します。

※本アプリケーションに必要なファイル
本アプリケーションが生成するファイル

```
home
  root
    RZV_AI_Eva_SW
      resnet50_bmp                          サンプルDRP-AI Object files
        resnet50_bmp_weight.dat             PyTorch ResNet-50モデルのImageモードBMP画像用DRP-AI Translator変換結果
        resnet50_bmp_drpcfg.mem             ※DRP-AI Translator変換結果のうち、本アプリケーションの実行に必要なファイルのみを記載
        aimac_desc.bin
        drp_desc.bin
        drp_param.bin
        resnet50_bmp_addrmap_intm.txt
      resnet50_jpg                          サンプルDRP-AI Object files
        resnet50_jpg_weight.dat              PyTorch ResNet-50モデルのImageモードJPG画像用DRP-AI Translator変換結果
        ・・・
      resnet50_cam                          サンプルDRP-AI Object files
        resnet50_cam_weight.dat              PyTorch ResNet-50モデルのCameraモード用DRP-AI Translator変換結果
        ・・・
      bmp_img                               サンプル入力画像
        sample.bmp                          BMP画像
      jpg_img                               サンプル入力画像
        sample.jpg                          JPG画像
      <prefix>_output                       AI Evaluation Software 出力物。<prefix>は使用したDRP-AI Object filesディレクトリ名。i.e., resnet50_bmp
        <img_dir>                           Imageモード出力物格納ディレクトリ。<img_dir>は入力データとして指定したディレクトリ名。i.e., bmp_img
          <img>.bin                         DRP-AI実行結果バイナリ。<img>は入力画像名。i.e., sample.bmp
          <TIME>.log                        AI Evaluation Software  実行結果ログ
        capture                              Cameraモード出力物格納ディレクトリ
          output<N>_<TIME>.bin              DRP-AI実行結果バイナリ
          output<N>_<TIME>.jpg              カメラキャプチャ画像（JPG)
          <TIME>.log                        AI Evaluation Software  実行結果ログ
      rzv_ai_eva_sw                         AI Evaluation Software
      start_app.sh                          AI Evaluation Software実行スクリプト
      config.ini                            設定ファイル
```

# アプリケーションの起動

- 以下のコマンドでアプリケーションを起動します。

  アプリケーションディレクトリに移動します。

```
# cd ~/RZV_AI_Eva_SW
```

  アプリケーションを実行します。

```
# ./start_app.sh <M> -d <DRP-AI_EXE> [-i <IMG_DIR>]
```

  – コマンドライン引数の詳細を以下に示します。

| 入力画像モード | <M> | <DRP-AI_EXE> | <IMG_DIR> |
|---|---|---|---|
| Imageモード | I | Imageモード用DRP-AI Object filesディレクトリ名<br>デフォルト：resnet50_bmp | 入力画像が格納されたディレクトリ名<br>デフォルト：bmp_img |
| Cameraモード | C | Cameraモード用DRP-AI Object filesディレクトリ名<br>デフォルト：resnet50_cam | カメラ画像を使用するためなし |

- アプリ起動後、DRP-AI Object filesをロードします。
  （ロード時間はモデル依存ですが、PyTorch ResNet-50で1秒程度かかります。）
- DRP-AI Object filesロード後、推論処理が可能になります。

ロード画面　(両モード共通)

```
# ./start_app.sh I
IMAGE_MODE
…
[INFO] DRP-AI Execution Binary: resnet50_bmp
[START] Loading DRP-AI Data…
  [START] Loading resnet50_bmp/resnet50_bmp_weight.dat : size …
  [END]   Loading resnet50_bmp/resnet50_bmp_weight.dat
  [START] Loading resnet50_bmp/resnet50_bmp_drpcfg.mem : size …
  [END]   Loading resnet50_bmp/resnet50_bmp_drpcfg.mem
  [START] Loading resnet50_bmp/drp_param.bin : size …
  [END]   Loading resnet50_bmp/drp_param.bin
  [START] Loading resnet50_bmp/aimac_desc.bin: size …
  [END]   Loading resnet50_bmp/aimac_desc.bin
  [START] Loading resnet50_bmp/drp_desc.bin : size …
  [END]   Loading resnet50_bmp/drp_desc.bin
[END] Loading DRP-AI Data : Total loading time  1.13 s
…
```

# 推論実行　IMAGEモード

- Imageモードでは、DRP-AI Object filesのロード後、自動で対象静止画像の推論を実行します。
- サンプルデータを使用して、推論を実行したコンソールログを右に示します。

```
# ./start_app.sh I
IMAGE MODE
[INFO] Image Directory : bmp_img
[INFO] DRP-AI Execution Binary: resnet50_bmp
[START] Loading DRP-AI Data…
…
[END] Loading DRP-AI Data : Total loading time  1.13 s
[bmp_img/sample.bmp]
1 images are loaded from img

Inference 1 ----------------------------------------------------
Input: bmp_img/sample.bmp
  DRP-AI processing time : 35.61 msec
  Output Binary          : resnet50_bmp_output/bmp_img/sample.bmp.bin
[INFO] 1 out of 1 images are processed.
[INFO] Output Log: resnet50_bmp_output/bmp_img/0722011508.log
```

- 推論対象画像について
  - Imageモードでは指定された入力データディレクトリ内の BMP画像とJPG画像のみをロードします。
  - ロードエラーが発生すると、該当画像をスキップし、次の画像に移行します。
  - 推論処理はロードされた全画像に対して実行されます。
  - 入力画像の上限枚数は20000枚です。
- アプリケーションの終了
  - 全画像の推論終了後、アプリケーションは自動で終了します。
  - アプリケーション終了後、右記の生成ファイルを確認することができます。

生成ファイル

<DRP-AI_EXE>_output/<IMG_DIR>ディレクトリに以下が保存されます。

※すでに同じ名称のファイルが存在する場合、上書きします。

- 推論結果出力バイナリファイル：　<IMG_NAME>.bin
  推論毎に1ファイル
- コンソール出力ログファイル：　<TIME>.log
  アプリ実行毎に1ファイル

# 推論実行　CAMERAモード

- Cameraモードでは、DRP-AI Object filesのロード後、ユーザーキー入力待ち状態になります。
- サンプルデータを使用して、推論を実行したコンソールログを右に示します。

```
# ./start_app.sh C
CAMERA MODE
[INFO] DRP-AI Execution Binary: resnet50_cam
[START] Loading DRP-AI Data…
…
[END] Loading DRP-AI Data : Total loading time  0.93 s

Press ENTER to capture an image or q to quit:

Inference 1 ----------------------------------------------------
DRP-AI processing time : 36.19 msec
  Output Binary          : resnet50_cam_output/capture/output1_0722011507.bin
  Output Image           : resnet50_cam_output/capture/output1_0722011507.jpg
Press ENTER to capture an image or q to quit:
[INFO] Output Log: resnet50_bmp_output/capture/0722011508.log
```

ENTERを入力

qを入力

- ユーザーキー入力について
  - ENTERキーを押すと推論が実行されます。
  - qキーを押すとアプリケーションが終了します。
  - 上記以外のキーはエラーとなり、ユーザーキー入力待ち状態が継続されます。
- 生成ファイルについて
  - アプリケーション終了後、右記の生成ファイルを確認することができます。

生成ファイル

<DRP-AI_EXE>_output/captureディレクトリに以下が保存されます。

※すでに同じ名称のファイルが存在する場合、上書きします。

- 推論結果出力バイナリファイル：　output<N>_<TIME>.bin
  推論毎に1ファイル
- カメラキャプチャJPG画像 ：　output<N>_<TIME>.jpg
  推論毎に1ファイル
- コンソール出力ログファイル：　<TIME>.log
  アプリ実行毎に1ファイル

# 推論結果について

- 本アプリケーションはDRP-AIの推論結果をそのままバイナリファイル(*.bin)として出力します。
- AI Evaluation Software の出力バイナリファイルの詳細は以下の通りです。
  - データ数：　モデルの出力サイズに依存。
    ※PyTorch ResNetの場合は[1 x 1000] = 1000
  - データ幅：　FP32データの場合、4byte
    FP16データの場合、2byte
  - バイトオーダー：　リトルエンディアン

- 上記の出力データから認識結果を取得するにはCPUによる後処理を別途実施する必要があります。
- AI Implementation Guideでは、PyTorch ResNet、PyTorch MobileNet、Darknet YOLO、MMPose HRNet、PyTorch DeepLabV3のPython3による後処理例をご紹介していますのでご参照ください。

# DRP-AI OBJECT FILESについて

- DRP-AIで推論を実行するにはONNX AIモデルをDRP-AI用に変換したDRP-AI Object filesが必要です。
- DRP-AI Object filesの作成には以下の3種類の入力ファイルが必要ですが、
  Imageモード、Cameraモードを使用するにはファイル②を変更する必要があります。

  ① *.onnx : 変換するONNXモデル。

  ② prepost_*.yaml : 前後処理設定ファイル。このファイルに変更が必要です。

  ③ addrmap_in_*.yaml : アドレスマップ設定ファイル。

- AI Implementation GuideではPyTorchによるResNet-50を例としてファイル②の変更方法をご説明しています。
  - PyTorch ResNet-50のDRP-AI Object filesを生成するために必要なファイル群（上記ファイル①②③）はAI Implementation Guideのサンプルとして提供されています。
  - pytorch_resnet_ver7.20.tar.gz内の*drpai_samples*ディレクトリ下をご参照ください。
- 入力ファイルを変更後、 AI Implementation Guideに従ってDRP-AI Translatorを実行し、
  DRP-AI Object filesを作成してください。

# アプリケーションのエラー

- 本アプリケーションはエラー発生時にエラーコードを表示します。(右記)

```
[ERROR] <エラー原因>
[ERROR] Err Code <エラーコード> : <エラー文字列>
```

- エラーコードの詳細を以下に示します。　※DRP-AI Driver APIについてはAI Implementation Guideをご参照ください。

| エラーコード | エラー名 | 種類 | 詳細 |
|---|---|---|---|
| 0x01 | ERR_INVALID_ARG | APP | アプリケーションへの不正コマンドライン引数。本エラー発生時はコマンドライン引数を修正してください。 |
| 0x02 | ERR_IMG_LIST_OVERFLOW | APP | Imageモードのみで発生。入力データ画像数のオーバーフローエラー。本エラー発生時は入力画像数を確認してください。<br>最大実行可能画像枚数は20000枚です。 |
| 0x10 | ERR_OPEN | FILE I/O | ファイル・ディレクトリのオープンエラー。本エラー発生時は該当ファイルを確認してください。 |
| 0x11 | ERR_FORMAT | FILE I/O | ファイル形式エラー。本エラー発生時は該当ファイルを確認してください。 |
| 0x12 | ERR_READ | FILE I/O | ファイル読み込みエラー。本エラー発生時は該当ファイルを確認してください。 |
| 0x13 | ERR_MMAP | FILE I/O | MMAP割当エラー |
| 0x14 | ERR_MALLOC | FILE I/O | MALLOCエラー |
| 0x15 | ERR_CLOSE | FILE I/O | ファイル・ディレクトリのクローズエラー。本エラー発生時は該当ファイルを確認してください。 |
| 0x20 | ERR_DRPAI_TIMEOUT | DRP-AI | DRP-AI終了割り込みのタイムアウトエラー。DRP-AIのエラーについてを参照。 |
| 0x21 | ERR_DRPAI_START | DRP-AI | DRP-AI Driver APIのioctl(DRPAI_START) システムコールのエラー。 |
| 0x22 | ERR_DRPAI_ASSIGN | DRP-AI | DRP-AI Driver APIのioctl(DRPAI_ASSIGN)システムコールのエラー。DRP-AI Object filesのアドレスエラーについてを参照。 |
| 0x23 | ERR_DRPAI_WRITE | DRP-AI | DRP-AI Driver APIのwrite関数エラー。 |
| 0x24 | ERR_DRPAI_READ | DRP-AI | DRP-AI Driver APIのread関数エラー。 |
| 0x25 | ERR_DRPAI_SELECT | DRP-AI | DRP-AI Driver APIのselect()システムコールのエラー。 |
| 0x26 | ERR_DRPAI_STATUS | DRP-AI | DRP-AI Driver APIのioctl(DRPAI_GET_STATUS)システムコールのエラー。 |
| 0x30 | ERR_V4L2_OPEN | V4L2 | Cameraモードのみで発生。V4L2のデバイスオープンに失敗 |
| 0x31 | ERR_V4L2_QUERYCAP | V4L2 | Cameraモードのみで発生。V4L2のデバイス機能のクエリに失敗 |
| 0x32 | ERR_V4L2_S_FMT | V4L2 | Cameraモードのみで発生。V4L2の設定値変更に失敗 |
| 0x33 | ERR_V4L2_REQBUFS | V4L2 | Cameraモードのみで発生。V4L2の画像データ用バッファの確保に失敗 |
| 0x34 | ERR_V4L2_QUERYBUF | V4L2 | Cameraモードのみで発生。V4L2のREQBUFSで要求したバッファの情報受け取りに失敗 |
| 0x35 | ERR_V4L2_STREAMON | V4L2 | Cameraモードのみで発生。V4L2デバイスストリーム開始に失敗 |
| 0x36 | ERR_V4L2_QBUF | V4L2 | Cameraモードのみで発生。V4L2バッファのストリーミングキューへの接続に失敗 |
| 0x37 | ERR_V4L2_DQBUF | V4L2 | Cameraモードのみで発生。V4L2バッファのストリーミングキューからの取り外しに失敗 |
| 0x38 | ERR_V4L2_STREAMOFF | V4L2 | Cameraモードのみで発生。V4L2のデバイスストリーム終了に失敗 |

# DRP-AI OBJECT FILESのアドレスエラーについて

- DRP-AIの仕様上、各DRP-AI Object filesは以下の制約を満たすアドレスに展開しなければなりません。

  1. 開始アドレスが64Byteアラインであること
  2. *_addr_intm.txtに記載されているアドレス及びサイズがDRP-AIメモリ領域内を指していること

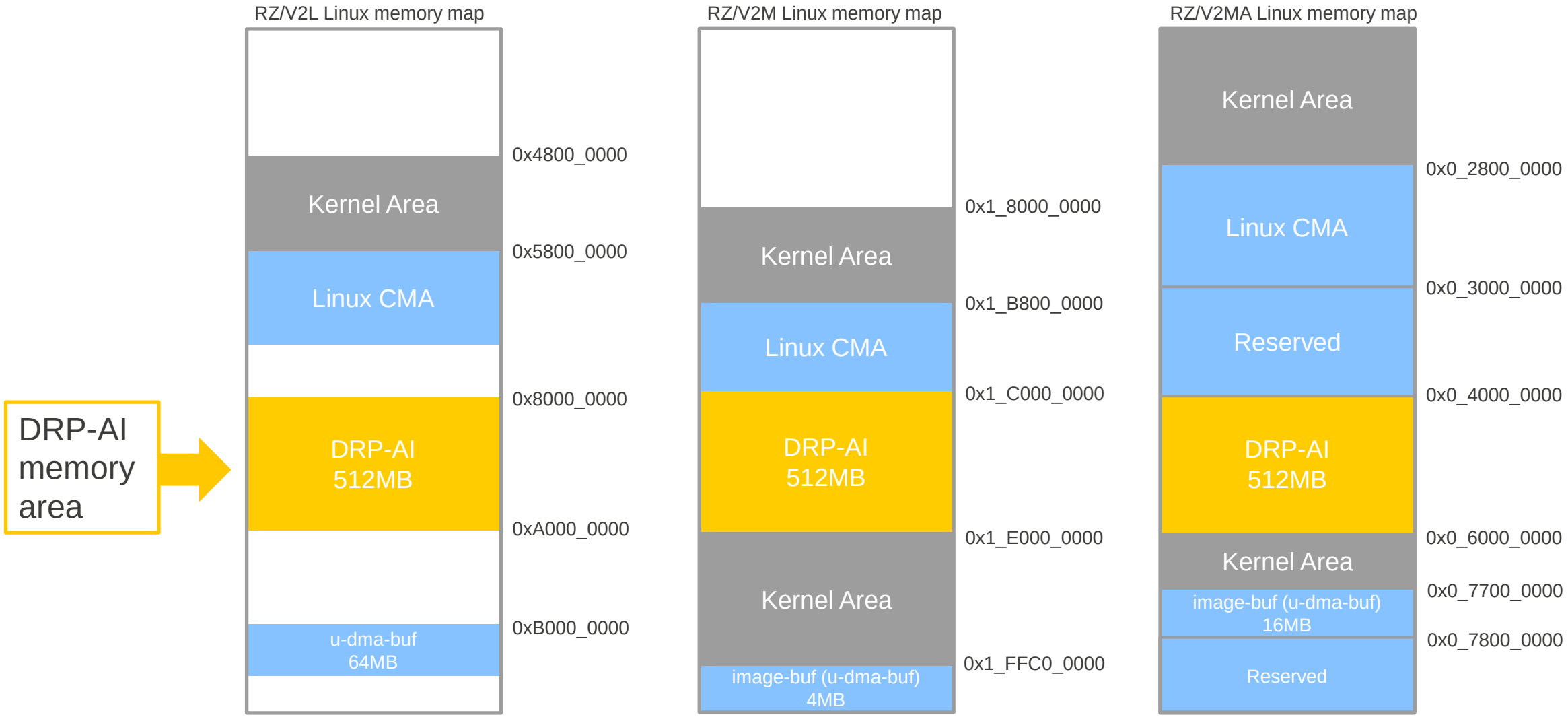
     DRP-AIメモリ領域については、次のページを参照してください。

- 上記の制約が満たされない場合、DRP-AI Driver API ioctl(DRPAI_ASSIGN)でエラーが発生します。

  ⇒ ERR_DRPAI_ASSIGN

  ※DRP-AI Translatorは上記の制約1を満たすアドレスを*_addr_intm.txtファイルに出力します。

# DRP-AI OBJECT FILESのアドレスエラーについて



# DRP-AIのエラーについて

- DRP-AIは処理をスタートさせた後、処理が終了したタイミングをアプリケーションに通知します。
- ただし、DRP-AI Object filesが不正の場合、終了通知が発生しないことがあります。
- 本アプリケーションでは上記エラー発生時に、一定時間終了通知が発生しなければ、強制終了します。

  ⇒　ERR_DRPAI_TIMEOUT
- ERR_DRPAI_TIMEOUTの閾値はデフォルトで5秒としています。
- ユーザーはコンフィグ設定ファイル(config.ini)からERR_DRPAI_TIMEOUTの閾値を変更することができます。

config.ini

```
; Comment …
[DRPAI_TIMEOUT]
5
```

※「;」で始まる行はコメントとして扱われます。

- なお、 ERR_DRPAI_TIMEOUT後は再度DRP-AIを起動することができません。
- 再度推論を実行する場合は、ボードを再起動してください。

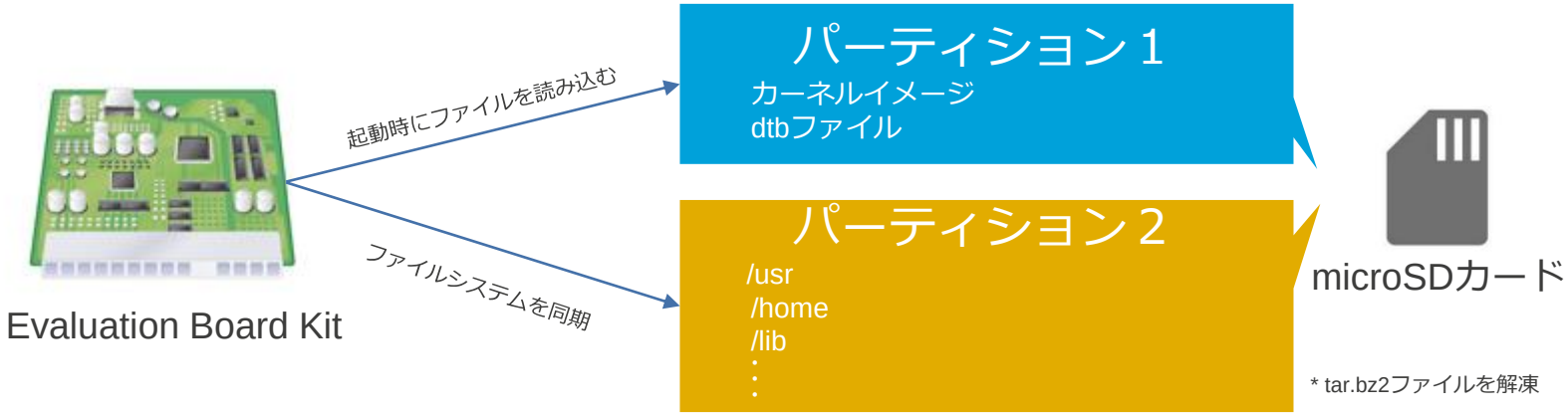
# 別紙 SDカードブート方式

本紙はSDカードブート方式について説明します。

# SDカードブートとは

SDカードブートとはボード上のメモリに展開するファイルをmicroSDカードとマウントするブート方式です。
ブートに必要なデータは以下の3種類です。microSDカードの各パーティション上に配置します。

| ファイル | 説明 | microSDカード展開先 |
|---|---|---|
| カーネルイメージ | Linuxカーネルイメージ（ブートプログラム） | パーティション１ |
| dtbファイル | Linuxデバイスツリーファイル（ブート用設定ファイル） | パーティション１ |
| tar.bz2ファイル | ルートファイルシステム一式 | パーティション２ |




- 本章ではパーティションのセットアップ方法とファイルの配置方法、及びSDカードブート方法をご説明します。

# 必要機材（RZ/V2L）

SDカードブートに必要な機材及び動作確認済み環境を以下に示します。

| 必要機材 | 詳細 | 入手方法 |
|---|---|---|
| RZ/V2L Evaluation Board Kit | Linuxブートローダーをe MMCに書き込み済み<br>※ブートローダーの書き込み方法については、RZ/V2L Linux Package Start-up Guideを参照してください。 | ルネサスから提供 |
| シリアルケーブル | PC＋RZ/V2L Evaluation Board間のシリアル通信に使用 | |
| AC adapter | 電源 | お客様にて<br>ご用意ください |
| USBケーブル Type-C | AC adapterとRZ/V2L Evaluation Boardを接続 | |
| microSDカード | 動作確認済み環境：Transcend USH-I microSD 300S 16GB | |
| PC | シリアル通信コンソール表示に使用<br>動作確認済み環境：Windows 10 | |
| USBカメラ | 動作確認済み環境：Logitech C930E WEBCAM | |
| Linux PC | microSDカードの作成に使用<br>動作確認済み環境：Ubuntu 20.04 | |
| SDカードリーダー | microSDカードの作成に使用 | |

# 必要機材（RZ/V2M, RZ/V2MA）

SDカードブートに必要な機材及び動作確認済み環境を以下に示します。

| 必要機材 | 詳細 | 入手方法 |
|---|---|---|
| RZ/V2x Evaluation Board Kit | Linuxブートローダーを eMMCに書き込み済み<br>※ブートローダーの書き込み方法については、RZ/V2x Linux Package Start-up Guideを参照してください。 | ルネサスから提供 |
| シリアルケーブル | PC＋RZ/V2x Evaluation Board Kit間のシリアル通信に使用 | お客様にて<br>ご用意ください |
| AC adapter | 電源 | |
| microSDカード | 動作確認済み環境：Transcend USH-I microSD 300S 16GB | |
| PC | シリアル通信コンソール表示に使用<br>動作確認済み環境：Windows 10 | |
| USBカメラ | 動作確認済み環境：Logicool C930e ウェブカム ※オプション | |
| Type-C変換アダプタ | RZ/V2x Evaluation Board KitのUSBポートがType-Cのため ※オプション<br>動作確認済み環境：SANWA SUPPLY USB A Type C変換アダプタAD-USB28CAF | |
| Linux PC | microSDカードの作成に使用<br>動作確認済み環境：Ubuntu 20.04 | |
| SDカードリーダー | microSDカードの作成に使用 | |

# 必要ソフトウェア

- 本手順では以下の同梱ファイルを使用します。入手方法は必要ファイルをご参照ください。
  - カーネルイメージ
  - dtbファイル
  - tar.bz2ファイル

- 本書では以下のようにテキストエディタとしてvimを使用しています。適宜お好みのエディタをご使用ください。

```
$ vi hello.txt
```

- 事前にUbuntu PCに共通で必要なパッケージをインストールしてください。

```
$ sudo apt-get update
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat cpio python python3 python3-pip
$ sudo apt-get install python3-pexpect xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm
$ sudo apt-get install python3-subunit mesa-common-dev p7zip-full
```

- SDカードブート方式で必要な追加パッケージを事前に以下のコマンドでインストールしてください。

```
$ sudo apt-get install gparted
```

# SDカード作成手順①

本章ではSDカードの作成手順の一例を示します。

1. microSDカードのデバイス番号を確認します。
   1. microSDカードが挿入されていない状態で、Linux PC上ターミナルで以下のコマンドを実行してください

```
$ df -h
Filesystem              Size                    Used        Avail       Use %       Mounted on
udev                    …                       …           …           …           /dev
…                       …                       …           …           …           …
/dev/sda1               …                       …           …           …           /
tmpfs                   …                       …           …           …           …
```

   2. 次にSDカードリーダーを使ってLinux PCにmicroSDカードを挿入してください。
   3. 再度以下のコマンドを実行してください

```
$ df -h
Filesystem              Size                    Used        Avail       Use %       Mounted on
udev                    …                       …           …           …           /dev
…                       …                       …           …           …           …
/dev/sda1               …                       …           …           …           /
tmpfs                   …                       …           …           …           …
/dev/sdb1               …                       …           …           …           …
```

差分

   4. 上記の例では「/dev/sdb」がmicroSDカードのデバイス番号となります。

**名称について**
/dev/sdb　：SDカード全体を指す
/dev/sdbN ：SDカードのパーティションNを指す。
　　ここではパーティション1を指している

# SDカード作成手順②

※本書では、SDカードの作り方例として
Gpartedを使用したパーティション作成手順をご紹介します。
適宜お好みの方法で2.2のパーティションを作成してください。

2. microSDカードのパーティションを作成します。

   本章では以下のソフトウェアを使用します。必要ソフトウェアを参考に、事前にインストールしてください。

   - gparted

1. パーティション作成ソフトウェアgpartedを起動します。（GUIが起動します。）
   /dev/sdbは1.4で確認したデバイス番号に置き換えてください。

```
$ sudo gparted /dev/sdb
```

2. GUI上で以下のようにmicroSDカードのパーティションを作成してください。

| パーティション番号 | サイズ | フォーマット |
|---|---|---|
| 1 | 128MB (123MiB)以上 | fat32 |
| 2 | 残り全て | ext4 |

パーティションを作成したら、内容を適用させます。
（適用させないと、変更した内容が反映されません。）

3. 正常にパーティションが作成されると「df」コマンド実行時に2つの/dev/sdbが表示されます。

```
$ df -h
Filesystem  Size                  Used        Avail       Use %       Mounted on
…                    …                      …           …           …           …
/dev/sdb1            …                      …           …           …           …
/dev/sdb2            …                      …           …           …           …
```

作成された
パーティション

# SDカード作成手順③

3. パーティション１の作成

1. パーティション１にカーネルイメージファイル（以下2ファイル）を書き込みます。

| Board | ファイル | ファイル名 |
|---|---|---|
| RZ/V2L | カーネルイメージ | Image-smarc-rzv2l.bin |
| | dtbファイル | Image-r9a07g054l2-smarc.dtb |
| RZ/V2M | カーネルイメージ | Image-rzv2m.bin |
| | dtbファイル | r9a09g011gbg-evaluation-board.dtb |
| RZ/V2MA | カーネルイメージ | Image-rzv2ma.bin |
| | dtbファイル | r9a09g055ma3gbg-evaluation-board.dtb |

2. microSDカードがUbuntu PCに挿入された状態で以下のコマンドを実行します。

```
$ sudo mkdir -p /mnt/sd
$ sudo mount /dev/sdb1 /mnt/sd
$ sudo cp <PATH_to_FILE>/<カーネルイメージ> /mnt/sd
$ sudo cp <PATH_to_FILE>/<dtbファイル> /mnt/sd
$ sync
$ sudo umount /dev/sdb1
```

※1. /dev/sdbは適宜変更してください。
2. <PATH_to_FILE> は各ファイルへのパスです。
適宜変更してください。
3. <カーネルイメージ>, <dtbファイル>は上記の表を参照して
変更してください。

# SDカード作成手順④

4. パーティション2の作成

1. パーティション2にルートファイルシステムを展開します。

| Board | tar.bz2ファイル |
|---|---|
| RZ/V2L | core-image-weston-smarc-rzv2l.tar.bz2 |
| RZ/V2M | core-image-bsp-rzv2m.tar.bz2 |
| RZ/V2MA | core-image-bsp-rzv2ma.tar.bz2 |

2. microSDカードがUbuntu PCに挿入された状態で以下のコマンドを実行します。

```
$ sudo mount /dev/sdb2 /mnt/sd
$ sudo tar xfj <PATH_to_FILE>/<tar.bz2ファイル> -C /mnt/sd
$ sync
$ sudo umount /dev/sdb2
```

※1. /dev/sdbは適宜変更してください。
2. <PATH_to_FILE> は各ファイルへのパスです。適宜変更してください。
3. <tar.bz2ファイル>は上記の表を参照して変更してください。

5. microSDカードの取り出し

1. 以下のコマンドを実行後、microSDカードをSDカードリーダーから抜きます。

```
$ sudo eject /dev/sdb
```

# シリアルポートドライバをインストール

Windows PCとボード間でシリアル通信をするには以下のドライバをインストールする必要があります。

■RZ/V2Lの場合

- https://ftdichip.com/drivers/vcp-drivers/

1. 上記リンクのダウンロードページのWindows版「setup executable」から、Virtual COM port (VCP) driverをダウンロードし、解凍します。
2. 解凍後の**exeファイル**を実行するとドライバがインストールされます。

■RZ/V2M, RZ.V2MAの場合

- https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

1. 上記リンクのダウンロードページからCP210x Windows Driverをダウンロードし、解凍します。
2. 解凍後の**exeファイル**を実行するとドライバがインストールされます。

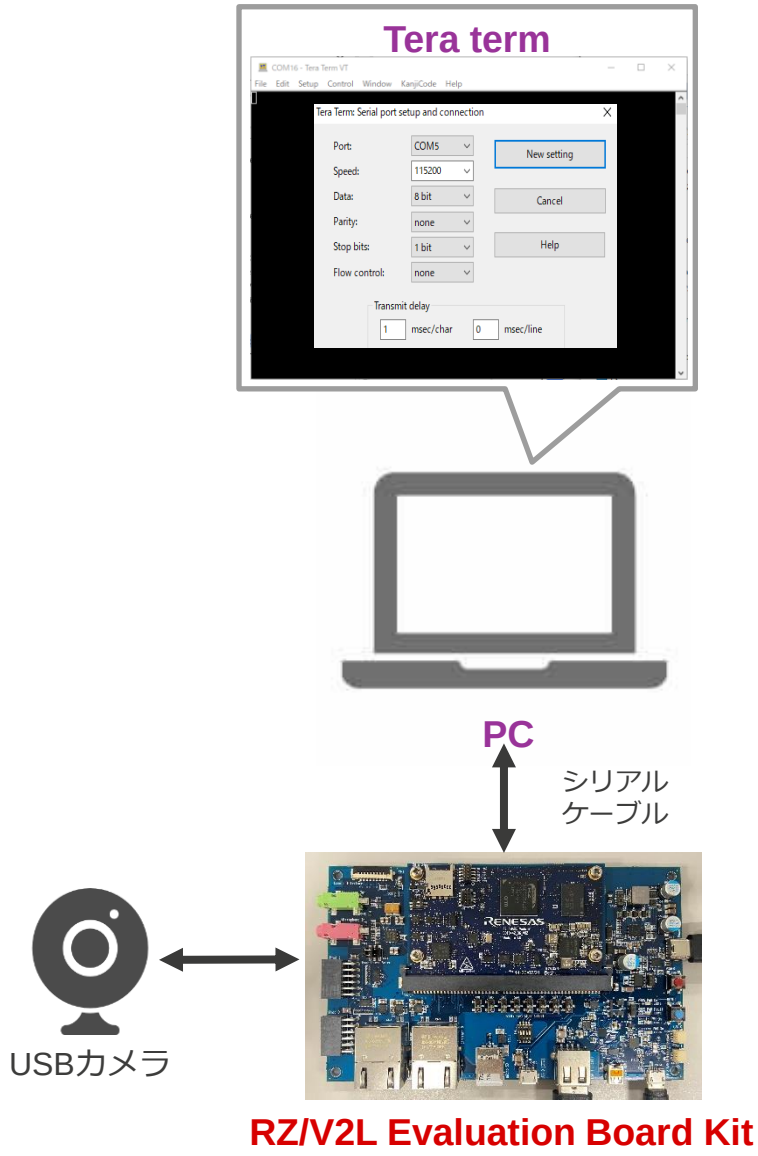
# ブート手順　SDブート（RZ/V2L）

■手順概要　（詳細は次ページ）

前提：以下の事前セットアップを実施済み

- microSDカードの作成
- シリアルポートドライバのインストール
- Windows PCにTera termをインストール

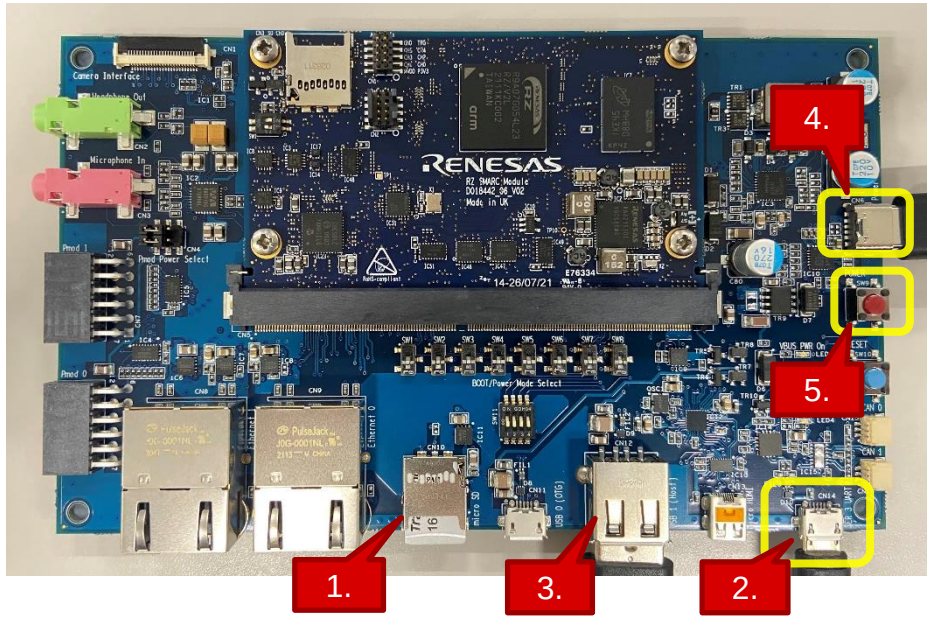
1. microSDカードを**ボード**に挿入します。
2. **ボード**と**PC**をシリアル-USB変換ケーブルでつなぎます。
3. **ボード**と**USBカメラ**をつなぎます。　※Cameraモード使用時のみ
4. **ボード**の電源ケーブルをつなぎます。
5. **ボード**の電源を入れます。
6. **Tera term**を起動し、COMSとシリアル接続をします。
7. **Tera term**でU-boot環境変数を変更します。　※ブート方式変更時のみ
8. ブート後、 **Tera term**上でシステムにログインします。



# ブート手順　詳細

前提：事前セットアップを全て実施済み

1. microSDカードを**ボード**に挿入します。
2. **ボード**と**PC**をシリアル-USB変換ケーブルでつなぎます。
3. **ボード**と**USBカメラ**をつなぎます。　※Cameraモードのみ
4. **ボード**の電源ケーブルをつなぎます。
5. **ボード**の電源を入れます。
6. **PC**上で**Tera term**を起動し、COMSとシリアル接続をします。
    1. Windows Startボタンをおし、「Tera term」で検索
    2. 検索結果からTera termを起動
    3. シリアルを選択
    4. 「設定」タブから各設定画面を開き、右項目のように設定する。




Tera term：設定項目

**端末設定**

| | |
|---|---|
| 改行コード | 受信：Auto |
| | 送信：CR |

**シリアルポート設定**

| | |
|---|---|
| スピード | ：115200 |
| データ | ：8bit |
| パリティ | ：none |
| ストップビット | ：1bit |

# ブート手順　詳細

7. Tera termでU-boot環境変数を変更します。　※ブート方式変更時のみ
   1. Tera term上で　ENTERキーを押し続けます。
   2. U-bootコンソールが起動します。
   3. 以下を入力します。

```
=> env default -a
=> setenv bootargs 'root=/dev/mmcblk1p2 rootwait'
=> setenv bootcmd 'mmc dev 1;fatload mmc 1:1 0x48080000 Image-smarc-rzv2l.bin; fatload mmc 1:1 0x48000000 Image-r9a07g054l2-smarc.dtb; booti 0x48080000 - 0x48000000'
=> saveenv
=> boot
```

8. Tera term上にログイン画面が表示されたら、以下の通りログインしてください。
   - user: root
   - password: なし

アプリケーション実行手順はNetworkブートと同様です。

- アプリケーション実行手順

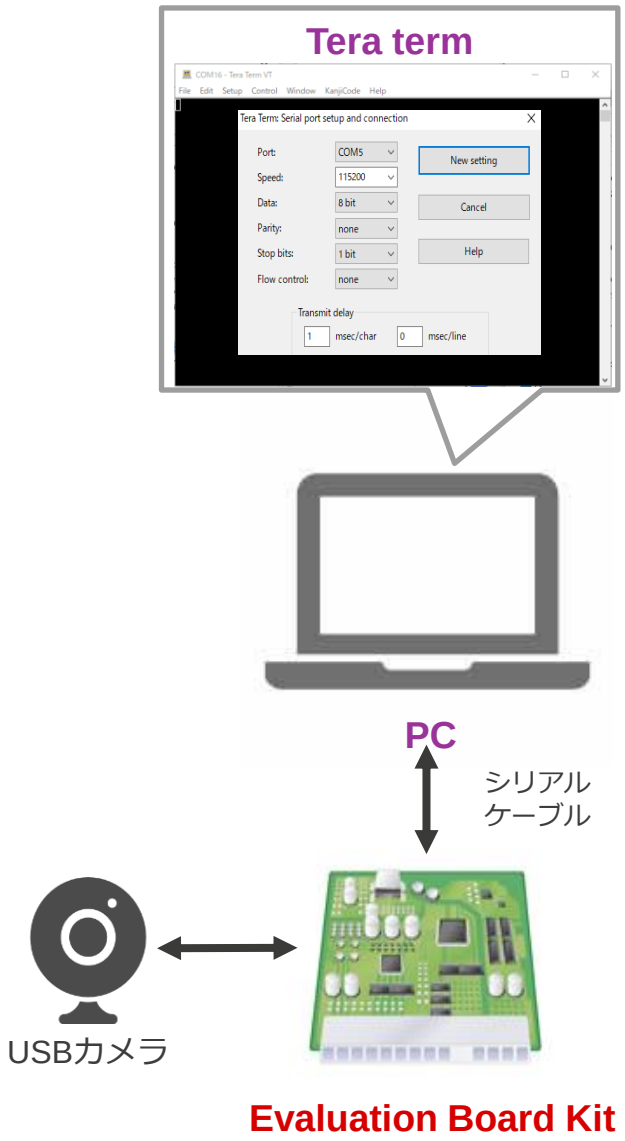
# ブート手順　SDブート（RZ/V2M, RZ/V2MA）

■手順概要　（詳細は次ページ）

前提：以下の事前セットアップを実施済み

- microSDカードの作成
- シリアルポートドライバのインストール
- Windows PCにTera termをインストール

1. microSDカードを**ボード**に挿入します。
2. **ボード**と**PC**をシリアル-USB変換ケーブルでつなぎます。
3. **ボード**の電源ケーブルをつなぎます。
4. **ボード**と**USBカメラ**をつなぎます。　※Cameraモード使用時のみ
5. **Tera term**を起動し、COMSとシリアル接続をします。
6. **ボード**の電源を入れます。
7. **Tera term**でU-boot環境変数を変更します。　※ブート方式変更時のみ
8. ブート後、**Tera term**上でシステムにログインします。

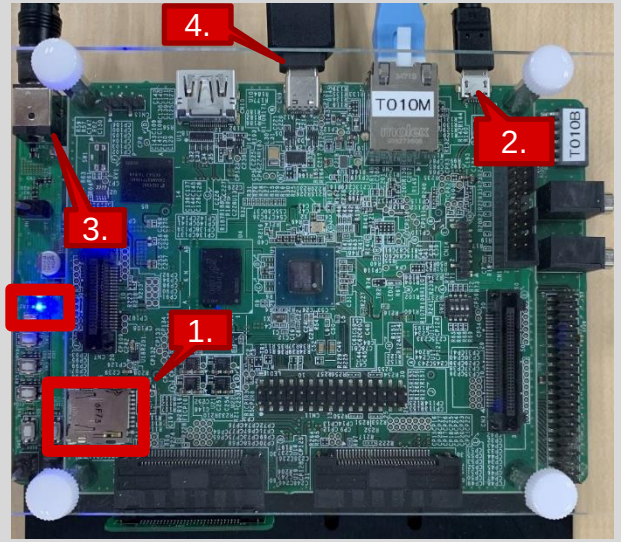


# ブート手順　詳細

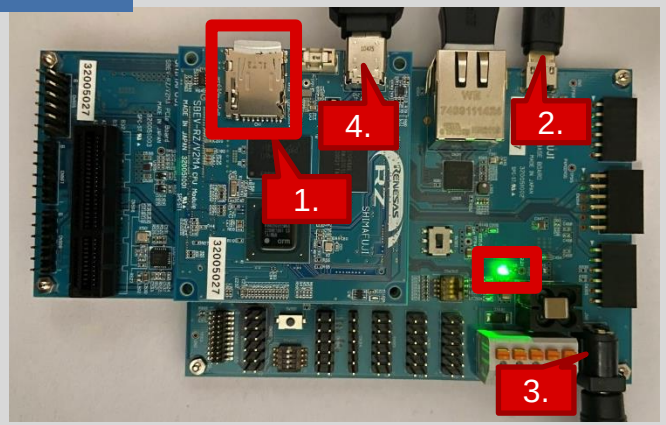
前提：事前セットアップを全て実施済み

1. microSDカードを**ボード**に挿入します。
2. **ボード**と**PC**をシリアル-USB変換ケーブルでつなぎます。
3. **ボード**の電源ケーブルをつなぎます。※LEDが点灯する
4. **ボード**と**USBカメラ**をつなぎます。　※Cameraモード使用時のみ

RZ/V2M




RZ/V2MA



# ブート手順　詳細

**5.** **PC**上で**Tera term**を起動し、COMSとシリアル接続をします。

1. Windows Startボタンをおし、「Tera term」で検索
2. 検索結果からTera termを起動
3. シリアルを選択
4. 「設定」タブから各設定画面を開き、右項目のように設定する。

Tera term：設定項目

**端末設定**

| | |
|---|---|
| 改行コード | 受信：Auto |
| | 送信：CR |

**シリアルポート設定**

| | |
|---|---|
| スピード | ：115200 |
| データ | ：8bit |
| パリティ | ：none |
| ストップビット | ：1bit |

# ブート手順　詳細

6. **ボード**の電源を入れます。 ※LEDがさらに点灯する

7. **Tera term**でU-boot環境変数を変更します。　※ブート方式変更時のみ
   1. **6**で電源を入れた直後、**Tera term**上で ENTERキーを押し続けます。
   2. U-bootコンソールが起動します。
   3. 以下を入力します。

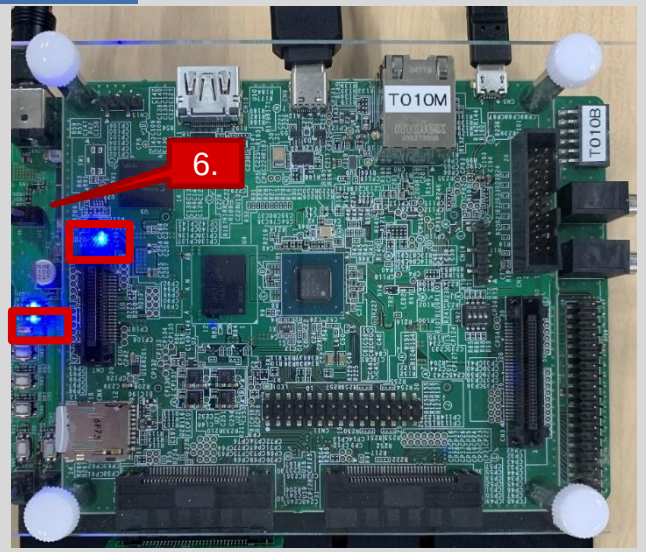
```
=> env default -a
=> setenv bootcmd run bootsd
=> saveenv
=> boot
```

8. **Tera term**上にログイン画面が表示されたら、以下の通りログインしてください。
   - user: root
   - password: なし

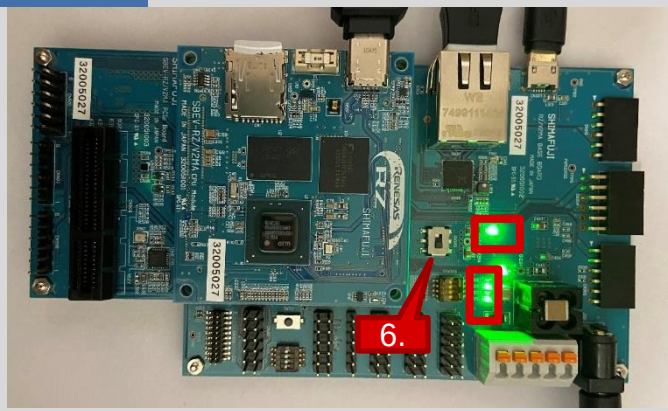
アプリケーション実行手順はNetworkブートと同様です。

- アプリケーション実行手順

RZ/V2M




RZ/V2MA



Renesas.com

# 変更履歴

| Date | Version | 変更内容 |
|---|---|---|
| Sep. 29, 2022 | 7.20 | 初版（RZ/V2L, RZ/V2M, RZ/V2MAのAI Evaluation Software Guideを統一） |