User's Manual

# RZ/V Verified Linux Package Start-Up Guide for RZ/V2MA

RZ/V2MA



Renesas Electronics
www.renesas.com

Rev.1.22 Jun 2025

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

# How to Use This Manual

## 1. Purpose and Target Readers

This document provides users with an understanding of the information to use the RZ/V Verified Linux Package (hereafter RZ/V VLP) on RZ/V2MA and this target board.

For the restrictions of this package, refer to the RZ/V VLP Release Note.

> Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text.

> The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.



The following documents are related to RZ/V VLP for RZ/V2MA.

| Document Type | Description | Document Title |
|---|---|---|
| Release note | Description of release information of RZ/V VLP. The restriction may be described in this document. | RZ/V Verified Linux Package Release Note |
| User's manual for Software | Description of the RZ/V2MA Linux Package instruction. | RZ/V Verified Linux Package Software Manual for RZ/V2MA |
| Usage guide | Guide how to use the RZ/V2MA Linux package. | This document |

## 2. List of Abbreviations and Acronyms

| Abbreviation | Full form |
|---|---|
| BSP | Board support package |
| eMMC | Embedded multimedia card |
| SDHC | SD high capacity |
| SDK | Software development kit |
| USB | Universal serial bus |

## 3. Conventions

Command line runs on Linux host PC will be shown as below:

```
$ echo "This is command line run on the Linux host PC."
```

Command line run on target board will be shown as below:

```
# echo "This is command line run on the target board."
```

# Table of Contents

# 1. Introduction

This start-up guide describes the procedure on how to boot RZ/V Verified Linux Package (hereinafter referred to as “VLP/V") on the RZ/V2MA Evaluation Board Kit

This guide provides the following information:

- Building procedure
- Preparation for use
- Boot loader and U-Boot
- How to run this Linux package on the target board (RZ/V2MA Evaluation Board Kit)
- How to create a software development kit (SDK)

## 1.1 RZ/V Verified Linux Package files

Refer to “RZ/V Verified Linux Package Release Note” (hereafter, release note) for the contents of files included in this package.

## 1.2 Environmental Requirement

Figure 1-1 shows the recommended environment for this package.

This environment uses the equipment and software listed in Table 1-1. Also, refer to Chapter 3 for the board, switch, cable, and SD setting.

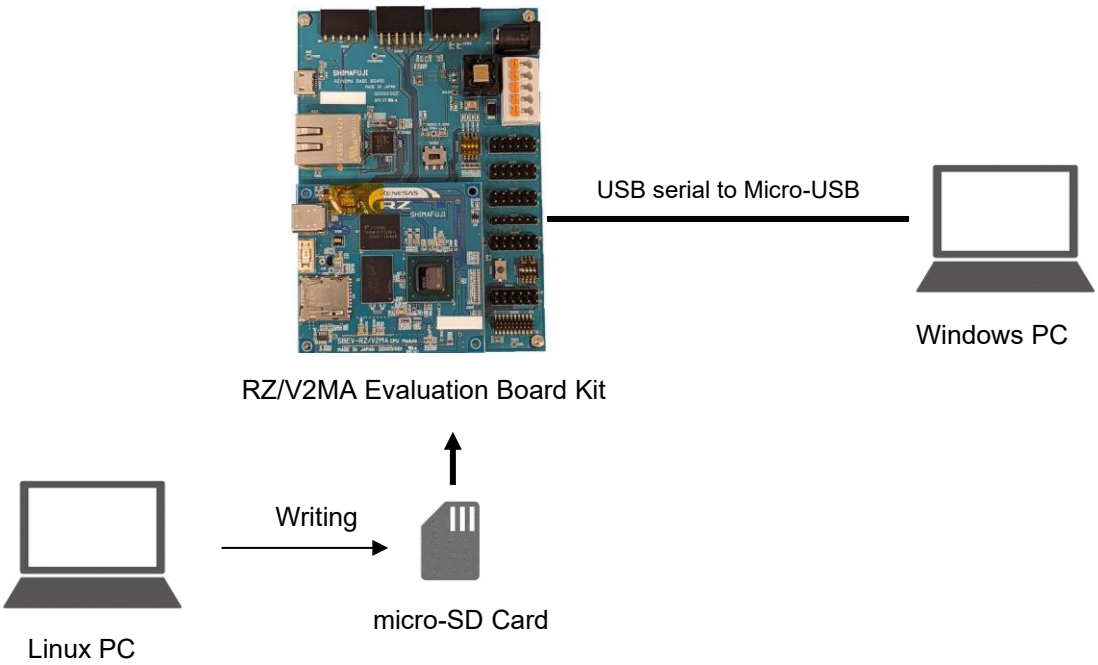



**Figure 1-1. Recommend environment**

**Table 1-1. Required equipment and software**

| Equipment | | Details |
|---|---|---|
| RZ/V2MA Evaluation Board Kit | | The evaluation board kit for RZ/V2MA. |
| | SBEV-RZ/V2MA CPU Module | Target board. The main functional components for RZ/V2MA are mounted on this board. Note that the boot loader and U-Boot images are pre-written to the eMMC (THGBMJG7C1LBAIL). |
| | RZ/V2MA BASE BOARD | Board for the generation and supply of power. |
| Linux PC | | Used as build/debug environment.<br>100GB of free space on HDD is necessary. |
| | OS | Ubuntu 22.04 LTS. Use a 64bit OS. |
| Windows PC | | Control the target board with terminal software. |
| | OS | Windows 10 is recommended. |
| | Terminal Software | Control the serial console of the target board.<br>Tera Term is recommended and available at "https://ttssh2.osdn.jp/index.html.en". |
| | VCP Driver | Virtual COM Port driver to enable the communication between Windows PC and the target board via USB. This is virtually used as a serial port and available at "https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers".<br>Install "CP210x Windows Drivers" at the above website. |
| USB serial to micro–USB Cable | | Serial communication (UART) between the RZ/V2MA Evaluation Board Kit and Windows PC. The type of USB serial connector on the RZ/V2MA Evaluation Board Kit is Micro USB type B. |
| micro–SD Card | | Use to boot the system, and store applications for the RZ/V2MA. |

# 2. Building Instructions

This section describes the instructions to build the Board Support Package (BSP) for RZ/V2MA.

Before starting the build, run the commands below on the Linux Host PC and install essential host packages used for building the BSP.

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

Refer to the URL below for detailed information.

- https://docs.yoctoproject.org/3.1.31/brief-yoctoprojectqs/brief-yoctoprojectqs.html

Note: The above URL document is for the yocto version supported in the VLP3.0.7. Please refer to the appropriate yocto version instructions.

Run the commands below and set the username and email address before starting the build procedure.
Without this setting, an error occurs when the building procedure runs git commands to apply patches.

```
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
```

Copy all required files obtained from Renesas into your home directory before the steps below. The directory which you put the files in is described as <package download directory> in the following build instructions.

**Step 1. Create a working directory and decompress the Yocto recipe and Bootloader packages**

Run the commands below. The name and the place of the working directory can be changed as necessary.

```
$ mkdir ~/rzv_vlp_<RZV_VLP_ver>
$ export WORK=/home/<user>/rzv_vlp_<RZV_VLP_ver>
$ cd $WORK
$ unzip ~/<package download directory>/RTK0EF0045Z0024AZJ-<RZV_VLP_ver>.zip
$ tar zxvf ./RTK0EF0045Z0024AZJ-<RZV_VLP_ver>/rzv_vlp_<RZV_VLP_ver>.tar.gz
$ unzip ~/<package download directory>/RTK0EF0045Z90001ZJ-<Bootloader_PKG_ver>.zip
$ tar zxvf ./RTK0EF0045Z90001ZJ-<Bootloader_PKG_ver>/meta-rz- \
features_v2ma_bl_<Bootloader_PKG_ver>.tar.gz
```

Notes:
1. The build environment must have 100GB of free hard drive space to complete the minimum build. The Yocto BSP build environment is very large. Especially in case you are using a virtual machine, please check how much disk space you have allocated for your virtual environment.
2. <RZV_VLP_ver> is the RZ/V VLP version number in the file name. (e.g. if VLP version v3.0.7, <RZV_VLP_ver> is v3.0.7.) Decompress the latest version of the RZ/V VLP. Refer to the release note for the appropriate file.
3. <Bootloader_PKG_ver> is the RZ/V2MA Bootloader Package version number in the file name. Decompress the latest version of the RZ/V VLP. Refer to the release note for the appropriate file.

**Step 2. Build Initialize**

Initialize a build using the 'oe-init-build-env' script in poky and point TEMPLATECONF to the platform conf path.

```
$ cd $WORK
$ TEMPLATECONF=$PWD/meta-renesas/meta-rzv2m/docs/template/conf/ \
 source poky/oe-init-build-env build
```

**Step 3. Add layers for the bootloaders**

Run the following command to build RZ/V2MAs bootloaders.

```
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-bootloaders
```

**Step 4. Decompress OSS files to “build” directory (Optional)**

Store the Open Source Package in your home directory and run the commands below. This step is not mandatory and able to go to the next step in case the “offline” environment is not required. All OSS packages will be decompressed with this '7z' command.

```
$ cd $WORK/build
$ 7z x ~/oss_pkg_rzv_<RZV_VLP_ver>.7z
```

Note: If this step is omitted and BB_NO_NETWORK is set to “0” in next step, all source codes will be downloaded from the repositories of each OSS via the internet when running bitbake command. Please note that if you do not use an “offline” environment, a build may fail due to the implicit changes of the repositories of OSS.

Open source software packages contain all source codes of OSSs. These are the same versions of OSSs used when VLP/V was verified. If you are just evaluating VLP/V and RZ/V2MA series, open source software packages are not mandatory to use. Usually, all the software can be built without using these files if your build machine is connected to the Internet.

Open source software packages are required for an “offline” environment. The word “offline” means an isolated environment which does not connect to any network. VLP/V can always build images in this “offline” environment by using these packages without affected from changes of original repositories of OSSs. Also, this “offline” environment always reproduces the same images as the images which were verified by Renesas. Note that if you build without using open source software packages, there are possibilities to use different source codes than Renesas used due to the implicit changes of the repositories of OSSs.

After the above procedure is finished, the “offline” environment is ready. If you want to prevent network access, please change the line in the “~/rzv_vlp_<package version>/build/conf/local.conf” as below:

```
BB_NO_NETWORK = "1"
```

To change BB_NO_NETWORK from “0” to “1”

After **Step 4. Decompress OSS files to "build" directory (Optional)**, running the following command

```
$ cd $WORK
$ unzip ~/RTK0EF0131F02000SJ-v1.0.zip
$ tar zxvf ./RTK0EF0131F02000SJ-v1.0/meta-rz-features.tar.gz
$ unzip ~/r11an0592ej0750-rzv2ma-drpai-sp.zip
$ tar zxvf ./rzv2ma_drpai-driver/meta-rz-drpai.tar.gz
$ cd build
$ bitbake-layers add-layer ../meta-rz-features/meta-rzv2ma-codec
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-drpai/
```

Note: RZ/V2MA Video Codec Package supports ORC to accelerate Encode processing. This function is no effect as a default. To decide whether to use it according to your system.

(How To enable ORC library)
Add the following setting in $(WORK)/build/conf/local.conf

```
# Configuration for ORC 0.4.33
USE_ORC_0.4.33 ?= "1"
```

To set core-image-bsp as at **Step 5. Start the build**.

### Step 5. Start the build

Run the bitbake command to start a build. Building an image can take up to a few hours depending on the user's host system performance.

Note: In this version, a fetch error will occur during the build process unless you follow the procedure outlined in the "Restrictions" section of the Release Note. Failure to comply may result in building failure. Please make sure to perform the steps before starting the build process.

```
$ MACHINE=rzv2ma bitbake core-image-<target>
```

This Linux package can build a few types of the image listed in Table 2-1.

Note: For a user of software packages for RZ/V2MA like DRP-AI Support packages and so on, apply all necessary recipes required for the build environments for the packages before bitbake. Refer to each software package documentation for information on how to apply the patches and build.

**Table 2-1. Supported images of RZ/V2MA**

| core-image-target | Detail |
|---|---|
| core-image-bsp | Basic BSP support. |
| core-image-minimal | Minimum sets of components. |

After completing the build, a similar output as below will appear, and the command prompt will return.

```
NOTE: Tasks Summary: Attempted 3894* tasks of which 8 didn't need to be rerun and all
succeeded.
```

Note: The number of tasks may change depending on your VLP version and other factors.

All necessary files listed in Table 2-2 will be generated by the bitbake command and will be in the build/tmp/deploy/images/rzv2ma directory.

**Table 2-2. Image files for RZ/V2MA**

| Files | File name | File stored path |
|---|---|---|
| Linux kernel image | Image-rzv2ma.bin | $WORK/build/tmp/deploy/images/rzv2ma |
| Device tree file | r9a09g055-v2maevk2.dtb | |
| root file system | <*image-name*>*1-rzv2ma.tar.bz2 | |
| 1st loader binary*2 | loader_1st_128kb.bin | |
| Boot parameter for 2nd loader*2 | loader_2nd_param.bin | |
| 2nd loader binary*2 | loader_2nd.bin | |
| U-Boot binary*2 | u-boot.bin | |
| Boot parameter for u-boot*2 | u-boot_param.bin | |
| Flash writer | B2_intSW.bin | |

Notes: 1. <*image-name*> is the name used in Step 5.
2. Store these loader binaries in the appropriate partition on the SD card when writing loader binaries.

# 3. Preparations

This chapter describes the preparation required before running the software on the RZ/V2MA Evaluation Board Kit. In the following procedure, the booting is from an SD card. For eMMC booting, refer to 7.2 eMMC Boot.

## 3.1 SD Card Setting

Here explains how to prepare the micro-SD card for booting the Linux or using the flash writer on the RZ/V2MA Evaluation Board.

### 3.1.1 SD card setting for using the flash writer

If you have already written this version of boot loader/U-Boot binaries on the eMMC, skip this procedure and go to 3.1.2.

#### 3.1.1.1 Files for SD card booting

The file for the flash writer is as follows. Create a partition with the specified file system format on your micro-SD card and store the flash writer binary file.

**Note: For the flash writer, use a micro-SDHC card.**

**Table 3-1. SD card boot files and partitions**

| Partition No. | Size | File system format | File name | Description |
|---|---|---|---|---|
| 1 | - | FAT32 | B2_intSW.bin | Flash writer binary. |

#### 3.1.1.2 Prepare for the flash writer

**Step 1. Create partitions on SD card**

Create one partition on Linux PC. Run the following commands in red to create SD card partitions.

```
$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): o
Created a new DOS disklabel with disk identifier 0xe68d03a6.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): <Press Enter>

Using default response p.
Partition number (1-4, default 1): <Press Enter>
```

```
First sector (2048-30425087, default 2048): <Press Enter>
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-30425087, default 30425087): <Press
Enter>

Created a new partition 1 of type 'Linux' and of size 14.5 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

$ partprobe
$ sudo mkfs.vfat -v -c -F 32 /dev/sdb1
mkfs.fat 4.1 (2017-01-24)
/dev/sdb1 has 64 heads and 32 sectors per track,
hidden sectors 0x0800;
logical sector size is 512,
using 0xf8 media descriptor, with 262144 sectors;
drive number 0x80;
filesystem has 2 32-bit FATs and 1 sector per cluster.
FAT size is 2017 sectors, and provides 258078 clusters.
There are 32 reserved sectors.
Volume ID is fb3f17b5, no volume label.
Searching for bad blocks
$
```

**Step 2. Store the flash writer binary file on the micro-SD Card**

Store the flash writer binary file (B2_intSW.bin) under FAT partition on the micro-SD card.

```
$ sudo mount /dev/sdb1 /media/
$ sudo cp <File_path_of_the_flash_writer_bin>/B2_intSW.bin /media/
$ sync
$ sudo umount /media/
```

### 3.1.2 SD card setting for booting Linux

### 3.1.3 Files for SD card booting

The required files for the SD card booting are as follows. Create partitions with the specified file system format, and store files in each area.

**Table 3-2. SD card boot files and partitions**

| Partition No. | Size | File system format | File name | Description |
|---|---|---|---|---|
| 1 | 128MB or more | FAT | Image-rzv2ma.bin* | Linux kernel image. |
| | | | r9a09g055-v2maevk2.dtb* | Device tree binary. |
| | | | Codec_Bin.bin | Codec support Binary |
| 2 | The rest | ext4 | core-image-bsp-rzv2ma.tar.bz2* | Root file system image. |

Note: The above files will be generated after building (bitbake). The rootfs here using is core-image-bsp. Refer to Table 2-2 about the directory the files are stored.

### 3.1.4 Prepare for booting from SD card

#### Step 1. Create partitions on SD card

Create two partitions for the SD card on Linux PC. The FAT area should be 128MB or more, and the ext4 area is the rest of the SD card capacity. Run the following commands in red to create SD card partitions.

Note: This description of creating partitions on the SD card is based on the following assumptions. Read them in conjunction with your environment.
· The storage that Linux uses is only “/dev/sdb”.
· SD card supports “/dev/sdb”.
Note that this operation may cause destroy your Linux environment.

```
$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): o
Created a new DOS disklabel with disk identifier 0x2c299b89.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): <Press Enter>

Using default response p.
Partition number (1-4, default 1): <Press Enter>
First sector (2048- 30199807, default 2048): <Press Enter>
Last sector, +sectors or +size{K,M,G,T,P} (2048- 30199807, default 30199807): +128M
```

```
Created a new partition 1 of type 'Linux' and of size 128 MiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): <Press Enter>

Using default response p.
Partition number (2-4, default 2): <Press Enter>
First sector (264192- 30199807, default 264192): <Press Enter>
Last sector, +sectors or +size{K,M,G,T,P} (264192- 30199807, default 30199807): <Press
Enter>

Created a new partition 2 of type 'Linux' and of size 14.3 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

$ partprobe
$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 14.41 GiB, 15462301696 bytes, 30199808 sectors
Disk model: Multi-Card
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x2c299b89

Device     Boot  Start      End  Sectors  Size Id Type
/dev/sdb1         2048   264191   262144  128M 83 Linux
/dev/sdb2       264192 30199807 29935616 14.3G 83 Linux
$ sudo mkfs.vfat -v -c -F 32 /dev/sdb1
mkfs.fat 4.1 (2017-01-24)
/dev/sdb1 has 64 heads and 32 sectors per track,
hidden sectors 0x0800;
logical sector size is 512,
using 0xf8 media descriptor, with 262144 sectors;
drive number 0x80;
filesystem has 2 32-bit FATs and 1 sector per cluster.
FAT size is 2017 sectors, and provides 258078 clusters.
There are 32 reserved sectors.
Volume ID is fb3f17b5, no volume label.
Searching for bad blocks
$ sudo mkfs.ext4 -L rootfs /dev/sdb2
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 3741952 4k blocks and 936560 inodes
Filesystem UUID: 53f29de4-1140-4917-b094-42d00b75308c
```

```
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables:   0/115       done
Writing inode tables:   0/115       done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information:   0/115       done
$
```

**Step 2. Store system files to the SD Card**

- **FAT partition**
  Store Linux image and device tree binary listed in Table 3-2 under FAT partition on SD card. Linux image and device tree binary are in $WORK/build/tmp/deploy/images/rzv2ma.

  — Linux kernel image (Image-rzv2ma.bin) and Device tree binary (r9a09g055ma3gbg-evaluation-board.dtb)

```
$ sudo mount /dev/sdb1 /media/
$ cd /media/
$ sudo cp $WORK/build/tmp/deploy/images/rzv2ma/Image-rzv2ma.bin .
$ sudo cp $WORK/build/tmp/deploy/images/rzv2ma/r9a09g055-v2maevk2.dtb .
$ sync
$ cd $WORK
$ sudo umount /media/
```

Note: "sdb1" (above in red) may depend on the user system.

- **ext4 partition**
  Store the root file system image listed in Table 3-2 under the ext4 partition of the SD card. Refer to Table 2-2 about the directories stored in this file.

  — Root file system image (core-image-bsp-rzv2ma.tar.bz2)

```
$ sudo mount /dev/sdb2 /media/
$ cd /media/
$ sudo tar jxvf $WORK/build/tmp/deploy/images/rzv2ma/core-image-bsp-rzv2ma.tar.bz2
$ sync
$ cd $WORK
$ sudo umount /media/
```

Note: "sdb2" (above in red) may depend on the user system.

### 3.1.5 Set U-Boot environment variables

Connect between the RZ/V2MA Evaluation Board Kit and Windows PC with a USB serial to micro–USB cable. And start the terminal software (Tera Term) on Windows PC. Refer to 3.2.2 Terminal software setting and set up the terminal software.

After this setting, power on the RZ/V2MA Evaluation Board Kit and U-Boot start. When the countdown begins on the console, press Enter key to move to the U-Boot command mode. Enter “env default -a” to set the environment variables of U-Boot to boot from the SD card. After this setting, save the environment with the command “saveenv” and start the kernel.

```
=>  env default -a
=> setenv codaddr 0x7FD00000
=> setenv codbin Codec_Bin.bin
=> setenv bootsd 'run bootargs_sd;ext4load mmc 0:2 ${codaddr} ${codbin};ext4load mmc 0:2
${loadadd} boot/${kernel};ext4load mmc 0:2 ${fdt_addr} boot/${fdt_file};booti
${loadaddr} - ${fdt_addr}'
=>  setenv 'bootargs_sd' 'setenv bootargs root=/dev/mmcblk1p2 rootwait rootfstype=ext4
rw'
=>  saveenv
=>  run bootsd
```

## 3.2 Board Setting

### 3.2.1 Switch

Confirm the switch setting shown in the red frame shown below.

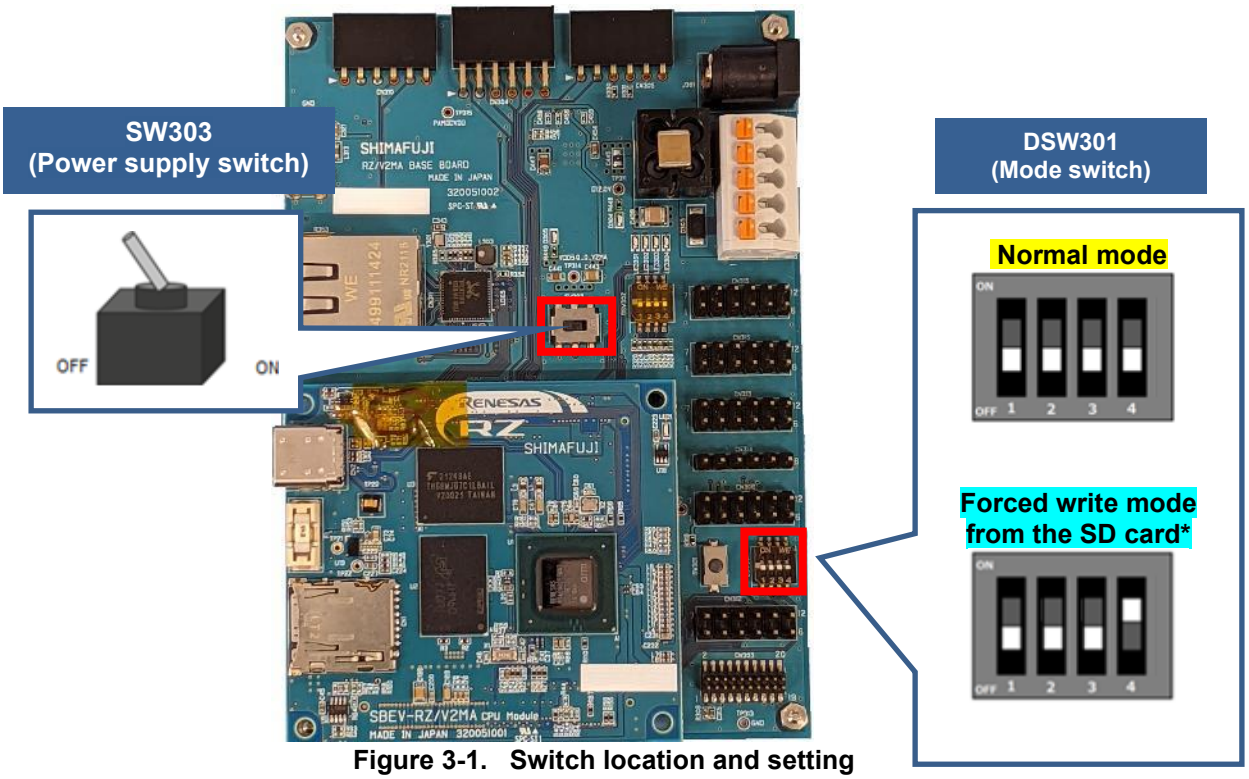



**Figure 3-1. Switch location and setting**

**Table 3-3. DSW301(Mode switch) setting**

| Mode | Switch 1 | Switch 2 | Switch 3 | Switch 4 |
|---|---|---|---|---|
| Normal | OFF | OFF | OFF | OFF |
| Forced write mode from the SD card* | OFF | OFF | OFF | ON |

Note: For the switch setting of debug mode, refer to the board manual for RZ/V2MA Evaluation Board Kit.

### 3.2.2 Terminal software setting

Connect a USB serial to micro-USB cable to Windows PC shown below.

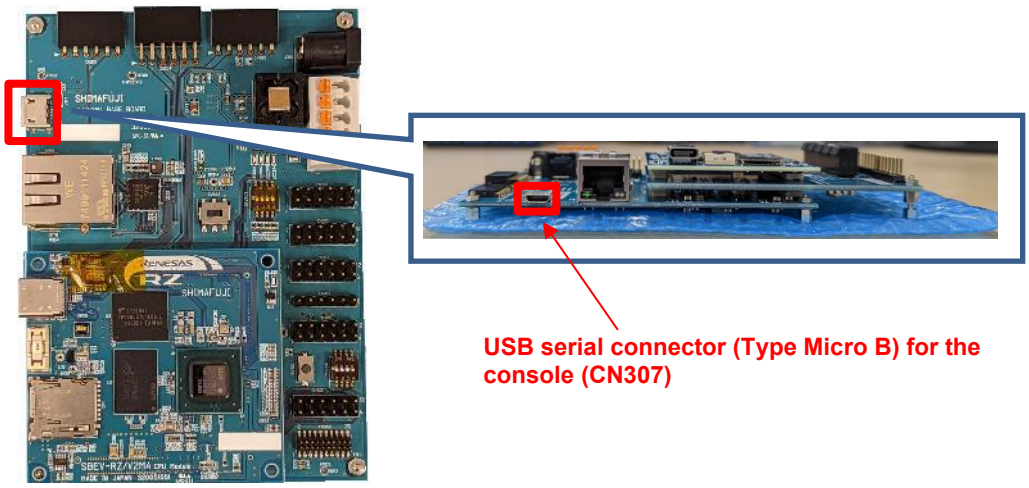



**Figure 3-2. Serial to micro-USB cable connection**

A serial port is detected on the PC after injecting the cable and choose “USB Serial Port”.

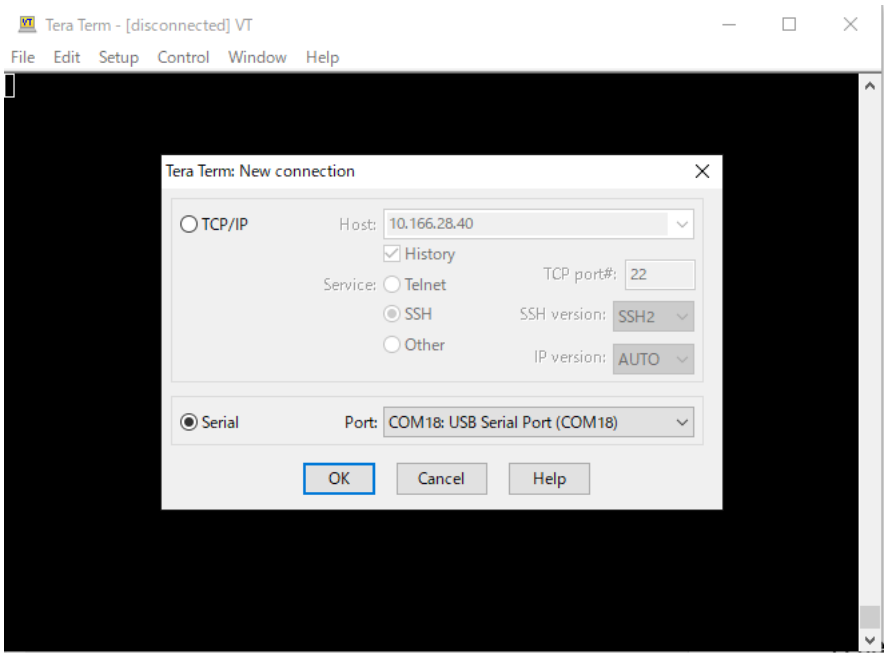



**Figure 3-3. Terminal software connection setting**

The terminal software should set the serial connection setting as follows.

(Menu > Setup > Serial port)

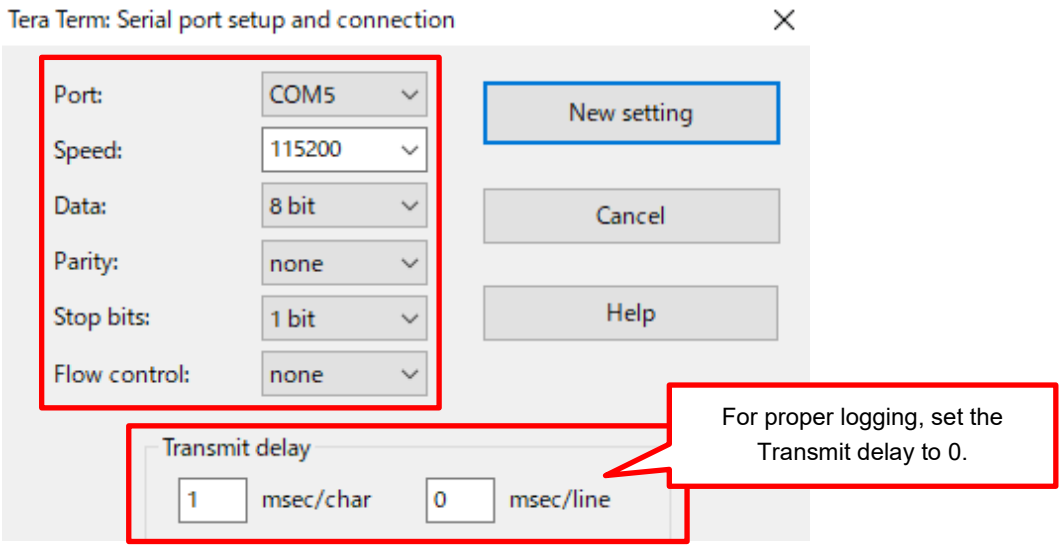



**Figure 3-4. Terminal software setting**

(Menu > Setup > Terminal)

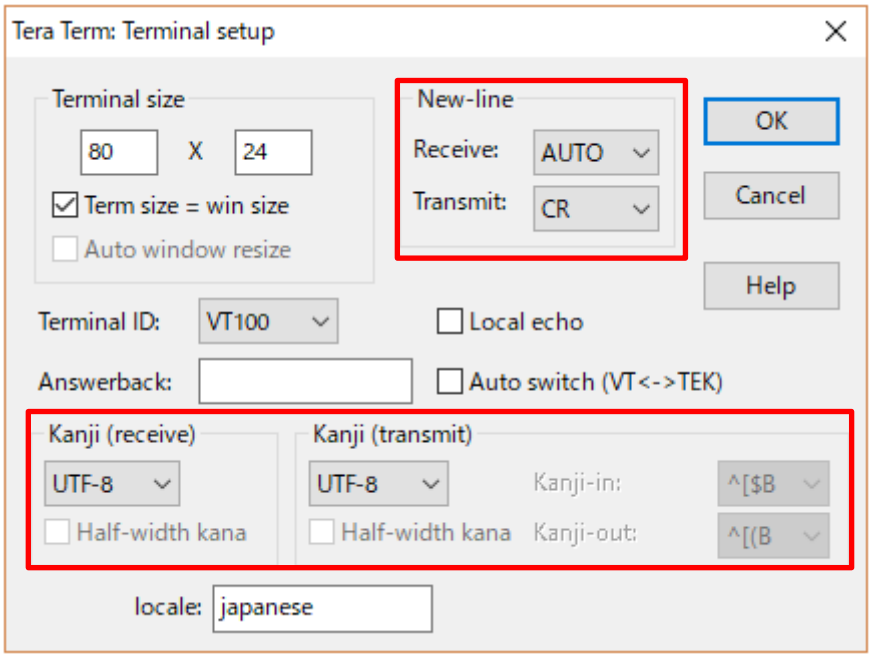



**Figure 3-5. Terminal software setting**

### 3.2.3 Insert a micro-SD

Set up a micro-SD card as described in 3.1 SD Card Setting and insert a micro-SD card into the connector shown in the following figure.

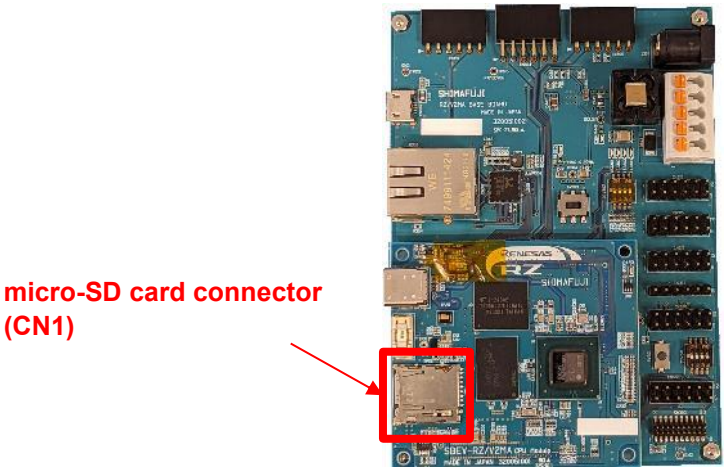



**Figure 3-6. micro-SD card connector**

***Note: The booting data should be written to the SD card beforehand to run the SD card booting on the RZ/V2MA EVK.***

### 3.2.4 LEDs

In this package, LEDs shown in the following figure are used to know the status of Linux (power on or shutdown).

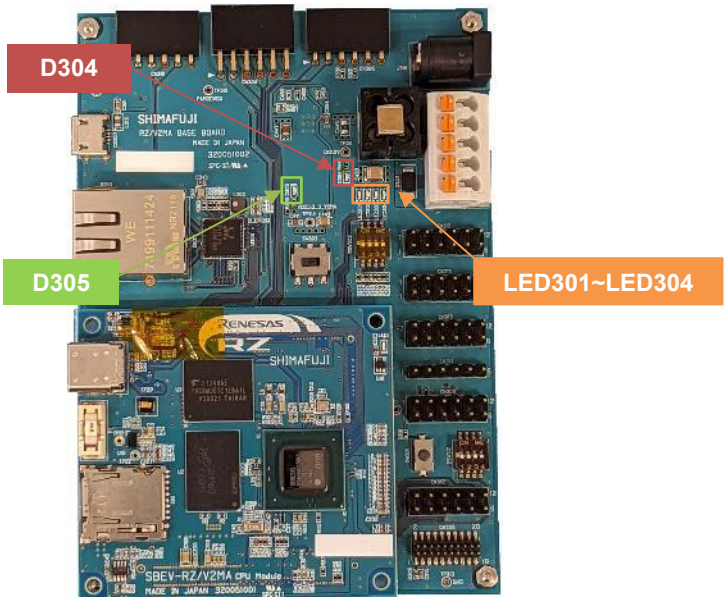



**Figure 3-7. LEDs**

# 4. Boot Loader, U-Boot (Loader Binaries)

This chapter explains the boot loader and U-Boot in this package.

For the procedure on how to build the source code of boot loader and U-Boot, refer to 7.1.1.

## 4.1 Boot loader and U-Boot Images

The boot loader and U-Boot binaries are stored in the eMMC on RZ/V2MA in advance. The bootloader and U-Boot will boot after powering on the RZ/V2MA Evaluation Board Kit.

The following table lists the address information stored in the eMMC on RZ/V2MA. These binary files of the boot loader/U-Boot are generated by the build described in 2.Building Instructions.

**Table 4-1. Boot loader data stored in the eMMC**

| File name | Program top address | eMMC save partition | eMMC save sectors*1 | File size(byte)*2 | Description |
|---|---|---|---|---|---|
| loader_1st_128kb.bin | H'80100000 | Boot partition 1 | H'000000 | H'20000 | 1st loader binary |
| loader_2nd_param.bin | On RAMA area *3 | Boot partition 1 | H'000100 | H'8 | Boot parameter for 2nd loader |
| loader_2nd.bin | H'B6000000 | Boot partition 1 | H'000101 | Variable*2 | 2nd loader binary*5 |
| u-boot_param.bin | On RAMB area *3 | Boot partition 1 | H'000901 | H'8 | Boot parameter for U-Boot |
| u-boot.bin | H'57F00000 | Boot partition 1 | H'000902 | Variable*2 | U-Boot binary |

Notes: 1. The sector size is 512bytes.
2. These file sizes are variable from the loader binary files generated by bitbake. Check the size of each file on your PC.
3. These RAM areas are not fixed because these binaries are stored in the local memory. After U-Boot boots, the boot loader and U-Boot will not use RAMA and RAMB.
4. The environment variables of U-Boot are stored in boot partition 2.
5. The maximum data size of 2nd loader is 1MB and fixed by the partition.

## 4.2 Flash Writer

Flash writer is the software for writing loader binaries to the eMMC on the RZ/V2MA Evaluation Board Kit via a PC. For building the flash writer from the source codes, refer to 7.1.2.

***Note: If you used the previous version or have not written the loader/U-Boot binaries to the eMMC, update them by following this section. Otherwise, the Linux booting may cause failure. This tool only supports writing loader binaries.***

### 4.2.1 Functions

The functions provided by this tool in this package are as follows.

- Write the binary image to the eMMC.
- Erase data of eMMC on a partition-by-partition basis.

***Note: The transferable data size of the flash writer is up to 1 MB.***

### 4.2.2 Write loader binaries to eMMC

This section describes how to write loader binaries to eMMC.

**Step 1. Equipment settings**

Run the following commands and install essential host packages on your build machine.

(1) Connect your Windows PC and RZ/V2MA Evaluation Board Kit with a USB serial to micro-USB cable as shown in Figure 4-1.

(2) Start the terminal software on your PC. Figure 4-2 shows the setting of the terminal software.

(3) Start the new connection. Choose “USB Serial Port”.

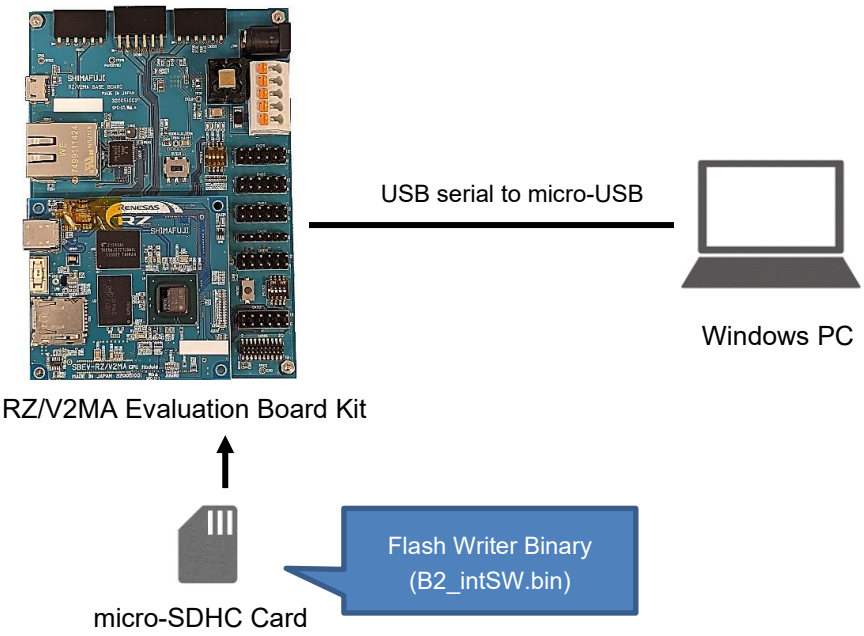



**Figure 4-1. Equipment setting**

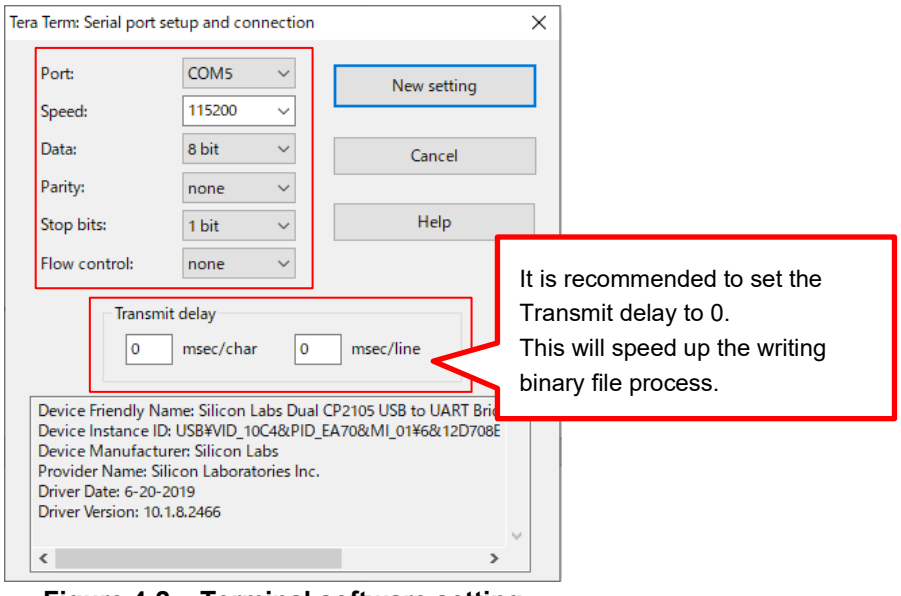



**Figure 4-2. Terminal software setting**

### Step 2. Write the flash writer to eMMC

Start RZ/V2MA with the forced write mode from the SD card and write the Flash writer binary to the eMMC in the following steps.

***Note that refer to "RZ/V2MA User's Manual: Hardware" about the forced write mode for details.***

(1) Store the Flash writer binary (B2_intSW.bin) in a micro-SD Card. Note that the micro-SD card should have only one partition formatted with FAT32.

(2) Insert the micro-SDHC card into the micro-SD card connector (CN1) on the RZ/V2MA Evaluation Board Kit.

(3) Set the DSW301 on the RZ/V2MA Base Board as shown in Table 4-2 and Figure 4-3 to change the board operation mode to the "forced write mode".

**Table 4-2. DSW301 setting for the forced write mode**

| Switch 1 | Switch 2 | Switch 3 | Switch 4 |
|---|---|---|---|
| OFF | OFF | OFF | ON |

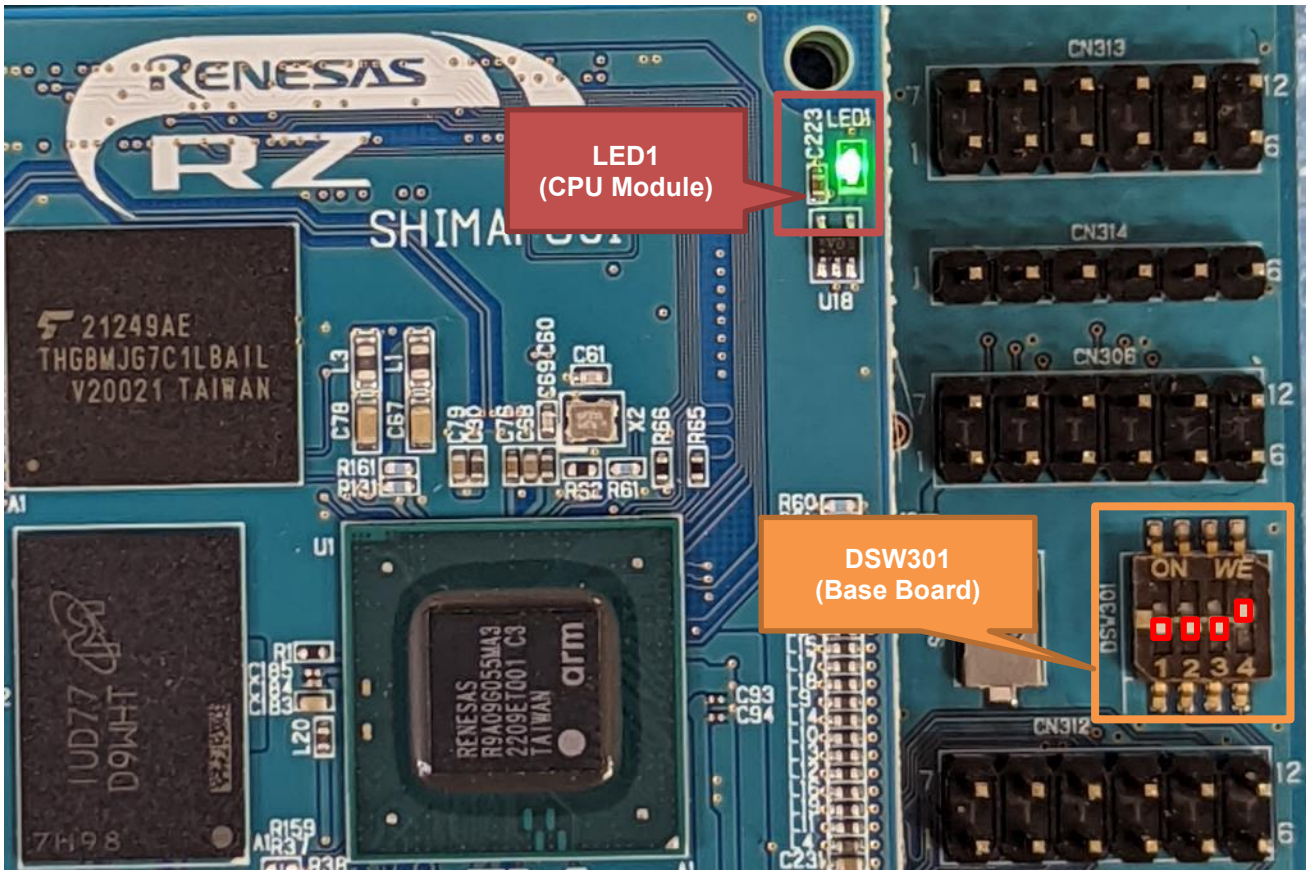



**Figure 4-3. LED lights in the forced write mode (RZ/V2MA Board)**

(4) Power on the RZ/V2MA Evaluation Board Kit. Start RZ/V2MA in forced write mode and write the Flash writer binary from the micro-SD card to eMMC.

(5) Check the lighting of LED1 on the CPU Module Board. If the LED1 lights up as shown in the above figure, writing to the eMMC has been completed successfully. On the other hand, if the LED1 is blinking, writing the Flash writer binary fails.

(6) Power off the RZ/V2MA Evaluation Board Kit.

### Step 3. Start Flash writer

Start RZ/V2MA in normal mode and run Flash writer.

(1) Set DSW301 to the normal mode [1:OFF, 2:OFF, 3:OFF, 4:OFF].

(2) Power on the RZ/V2MA Evaluation Board Kit. The following log will appear if the RZ/V2MA starts in normal mode and run Flash writer successfully.

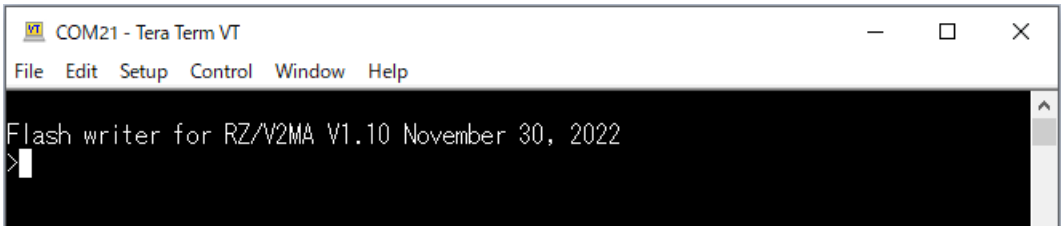



**Figure 4-4. Start Flash writer**

### Step 4. Write loader binaries to eMMC with Flash writer

Enter each command in red for Flash writer on the terminal software (Tera Term) on Windows PC.

(1) Erase the data of boot partition 1. Run the command EM_E as follows.

```
>EM_E
EM_E Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area    : 15388672 KBytes
  eMMC Sector Cnt : H'0 - H'01D59FFF
 1:Boot Partition 1       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
 2:Boot Partition 2       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
-----------------------------------------------------------
  Select area(0-2)>1
-- Boot Partition 1 Program ----------------------------
EM_E Complete!
>
```

(2) Write loader binaries to the boot partition 1. Run the command EM_WB, enter the eMMC partition area, file size, and send binary files shown in Table 4-1. In this procedure, the binary files are sent by Tera Term.

```
>EM_WB

EM_WB Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area    : 15388672 KBytes
  eMMC Sector Cnt : H'0 - H'01D59FFF
 1:Boot Partition 1       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
 2:Boot Partition 2       : 4096 KBytes
```

```
  eMMC Sector Cnt : H'0 - H'00001FFF
-----------------------------------------------------------
  Select area(0-2)>1

-- Boot Partition 1 Program -----------------------------
Please Input Start Address in sector : 000 *Input in hexadecimal

Work RAM(H'B6000000-H'B60FFFFF) Clear....
Please Input File size(byte) : 20000 *Input in hexadecimal

please send binary file! <Send "loader_1st_128kb.bin">
SAVE -FLASH.......
EM_WB Complete!

>EM_WB
EM_WB Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area    : 15388672 KBytes
  eMMC Sector Cnt : H'0 - H'01D59FFF
 1:Boot Partition 1       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
 2:Boot Partition 2       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
-----------------------------------------------------------
  Select area(0-2)>1

-- Boot Partition 1 Program -----------------------------
Please Input Start Address in sector : 100 *Input in hexadecimal

Work RAM(H'B6000000-H'B60FFFFF) Clear....
Please Input File size(byte) : 8 *Input in hexadecimal

please send binary file! <Send "loader_2nd_param.bin">
SAVE -FLASH.......
EM_WB Complete!

>EM_WB
EM_WB Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area    : 15388672 KBytes
  eMMC Sector Cnt : H'0 - H'01D59FFF
 1:Boot Partition 1       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
 2:Boot Partition 2       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
-----------------------------------------------------------
  Select area(0-2)>1

-- Boot Partition 1 Program -----------------------------
Please Input Start Address in sector :101 *Input in hexadecimal

Work RAM(H'B6000000-H'B60FFFFF) Clear....
```

Appear this message when the writing process is successful.

The method of sending files will be described after the following log.

```
Please Input File size(byte) : <Enter the file size of “loader_2nd.bin”> *Input in hexadecimal

please send binary file! <Send “loader_2nd.bin”>
SAVE -FLASH.......
EM_WB Complete!

>EM_WB
EM_WB Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area    : 15388672 KBytes
  eMMC Sector Cnt : H'0 - H'01D59FFF
 1:Boot Partition 1       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
 2:Boot Partition 2       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
-----------------------------------------------------------
  Select area(0-2)>1

-- Boot Partition 1 Program -----------------------------
Please Input Start Address in sector :901

Work RAM(H'B6000000-H'B60FFFFF) Clear....
Please Input File size(byte) : 8

please send binary file! <Send “u-boot_param.bin”>
SAVE -FLASH.......
EM_WB Complete!

>EM_WB

EM_WB Start --------------
-----------------------------------------------------------
Please select,eMMC Partition Area.
 0:User Partition Area    : 15388672 KBytes
  eMMC Sector Cnt : H'0 - H'01D59FFF
 1:Boot Partition 1       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
 2:Boot Partition 2       : 4096 KBytes
  eMMC Sector Cnt : H'0 - H'00001FFF
-----------------------------------------------------------
  Select area(0-2)>1

-- Boot Partition 1 Program -----------------------------
Please Input Start Address in sector :902

Work RAM(H'B6000000-H'B60FFFFF) Clear....
Please Input File size(byte) : <Enter the file size of “u-boot.bin”> *Input in hexadecimal

please send binary file! <Send “u-boot.bin”>
SAVE -FLASH.......
EM_WB Complete!
```

**[How to send files by Tera term]**

1. Select "File" -> "Send File" as shown in the figure below.

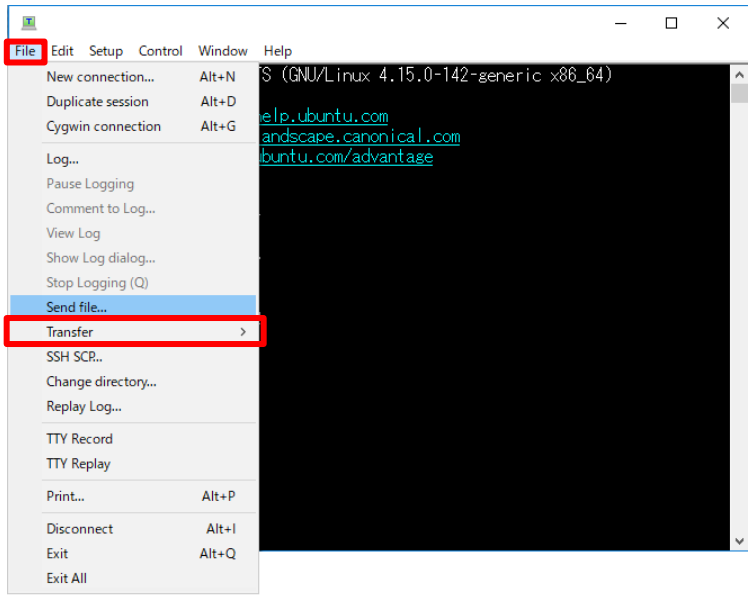



**Figure 4-5. Send files by Tera Term**

2. Select the file to send. Note that check "Binary" in the option.

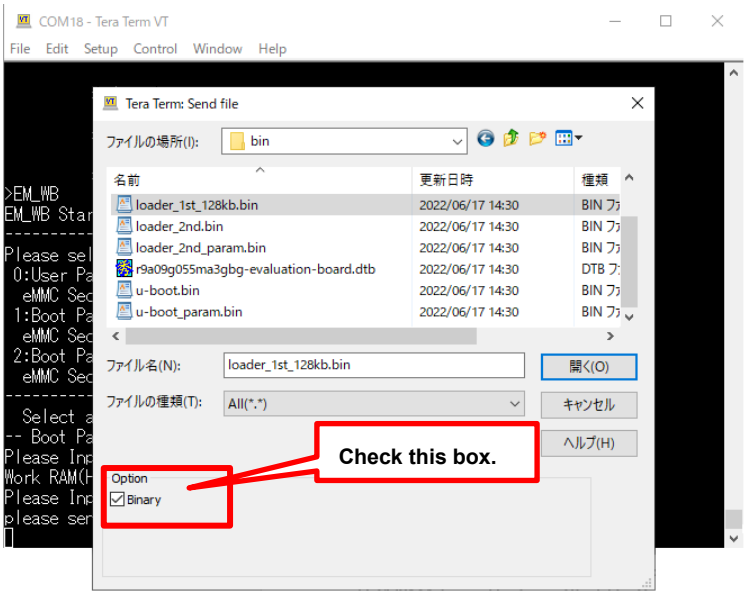



**Figure 4-6. Send files by Tera Term**

(3) Power off the RZ/V2MA Evaluation Board Kit.

Note: When getting an error message listed in the following table, retry the writing process.

**Table 4-3. Error messages (Flash writer)**

| Error message | Error content |
|---|---|
| Time out! Unable to receive data for the specified size! | Timeout error. |
| Received a break signal | Received a break signal from the host. |
| Framing Error! Failed to receive data! | Framing error. |
| Parity Error! Failed to receive data! | Parity error. |
| Overrun Error! Failed to receive data! | Overrun error. |
| FIFO Error! Failed to receive data! | FIFO error. |
| System Error! Failed to receive data! | System error. |

#### Step 5. Confirm booting by the boot loader and U-Boot

Confirm that the loader binaries are written to eMMC normally by checking the operation of the boot loader and U-Boot.

(1) Set DSW301 to the normal mode [1:OFF, 2:OFF, 3:OFF, 4:OFF].
Confirm that the startup operation mode is "normal mode".
(2) Power on the RZ/V2MA Evaluation Board Kit.
(3) Confirm that the boot loader and U-Boot boots successfully.
(4) When starting the countdown, press Enter to move to the U-Boot command mode.

#### Step 6. Load the U-Boot environment variables

Execute the following commands to load the u-boot environment variables for this version to their default values.

```
=> env default -a
=> saveenv
```

After the above, boot the Linux kernel and confirm the Linux kernel booting successfully.

Note: The shutdown command is different when running from the Linux kernel or U-Boot command mode. On U-Boot mode, run the “evk_shutdown” as follows.

```
=> evk_shutdown
```

# 5. Run on the Board

This chapter explains how to set up the RZ/V2MA Evaluation Board Kit and run the system.

## 5.1 Power on the board

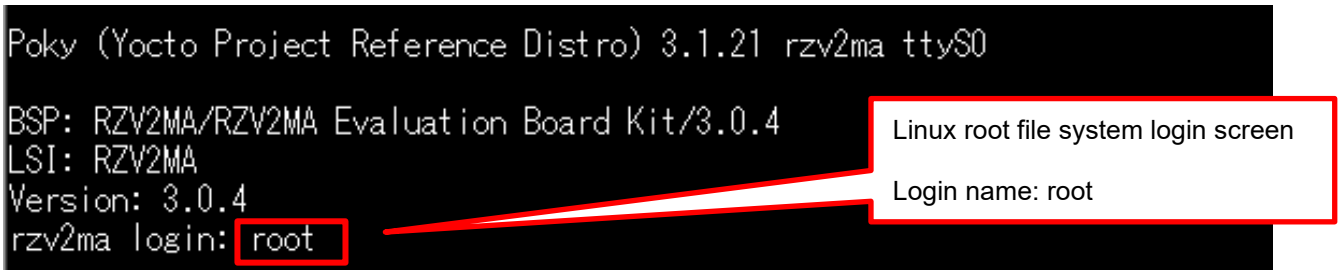
Notice: Before connecting the AC adapter (J301) to an electrical outlet, make sure that the SW303 on the RZ/V2MA Base Board for the power supply is turned off.

When the AC adapter is connected, the D304 light up. After turning on SW303, LED302, LED304, and D305 light up, and RZ/V2MA boots.

## 5.2 Startup Linux

Turn on the power switch, and the U-Boot and Linux start. After booting, check the serial console on Windows PC.

After the initialization of the Linux kernel, the root file system starts. The red line in the following figure appears after the initialization of the root file system. Enter “root” on the login screen.




Note: This login screen is for VLP3.0.4 and the yocto version is 3.1.21.The VLP and yocto version will change when the VLP is updated.

**Figure 5-1. Root file system login screen**

## 5.3 Shutdown the Board

Note: The shutdown command is different when running from the Linux kernel or U-Boot command mode. On the Linux kernel, run the “shutdown” as follows.

Follow the steps below to turn off the system.

**Step 1. Run shutdown command**

Run shutdown command on the console as below. After that, the shutdown sequence will start.

```
root@rzv2ma:~# shutdown -h now
```

Note: Run this command during the power-off sequence on rootfs.

**Step 2. Confirm the power-off**

After executing the shutdown command, confirm that LED302, LED304, and D305 are off.

**Step 3. Turn off the power switch on the board**

After checking the above LEDs, turn SW303 off.

***Note: Be sure to follow the steps correctly when shutting down the system in the kernel.***

***[1] run shutdown command.***

***[2] Turn SW303 off after the LED302, LED304, and D305 are off.***

***[3] Remove the AC adapter from the outlet. *When finished using the system completely.***

***If the shutdown process does not follow the above steps, it may lead to the cause of destroying the device.***

# 6. Building SDK

This section describes how to build the Software Development Kit (SDK). To build the SDK, run the commands below after the steps described in 2. Building Instructions

The SDK allows you to build custom applications outside of the Yocto environment, even on a completely different PC. The results of the commands below are 'installer' that you will use to install the SDK on the same PC or a completely different PC.

Note: If you use RZ/V2MA DRP-AI Support Package, deploy them before building the SDK. Refer to each package's document for the detail to apply each application to the SDK.

For building the SDK, run the following commands.

```
$ cd $WORK/build
$ MACHINE=rzv2ma bitbake core-image-bsp -c populate_sdk
```

The resulting SDK installer will be stored in $WORK/build/tmp/deploy/sdk/.

The SDK installer will have the extension .sh. To run the installer, execute the following command:

```
$ cd $WORK/build/tmp/deploy/sdk/
$ sudo sh poky-glibc-x86_64-core-image-bsp-aarch64-rzv2ma-toolchain-<Yocto_version>.sh
```

Note: The <Yocto_version> is the number of the Yocto version supported in the RZ/V VLP.
(e.g.) RZ/V VLP v3.0.4 supports Dunfell(3.1.21), so the <Yocto_version> is 3.1.21.

Set up environment variables as follows.

```
$ source /opt/poky/<Yocto_version>/environment-setup-aarch64-poky-linux
```

Note: The SDK build may fail depending on the building environment. At that time, build again after a while. Or rebuild it from scratch with the below commands.

```
$ cd $WORK/build
$ bitbake core-image-bsp -c cleanall
$ bitbake core-image-bsp
$ bitbake core-image-bsp -c populate_sdk
```

# 7. Appendix

## 7.1 Building Instructions

Before building the RZ/V2MA boot loader, U-Boot, and flash writer binaries, follow the steps from Step 1 to Step 5 described in 2.Building Instructions.

### 7.1.1 Boot loader and U-Boot

Run the following commands to build boot loader and U-Boot. The files shown in Table 7-1 will be generated.

- Boot loader

```
$ MACHINE=rzv2ma bitbake bootloader
```

- U-Boot

```
$ MACHINE=rzv2ma bitbake u-boot
```

**Table 7-1. Generated files (Boot loader and U-Boot)**

| Generated files | File name | File stored path |
|---|---|---|
| 1st loader binary | loader_1st_128kb.bin | $WORK/build/tmp/deploy/images/rzv2ma |
| Boot parameter for 2nd loader | loader_2nd.bin | |
| 2nd loader binary | loader_2nd_param.bin | |
| U-Boot binary | u-boot.bin | |
| Boot parameter for u-boot | u-boot_param.bin | |
| Boot loader source codes | *Omitted | $WORK/build/tmp/work/rzv2ma-poky-linux/bootloader/<version>/source |
| U-Boot source codes | *Omitted | $WORK/build/tmp/work/rzv2ma-poky-linux/u-boot/<version>/git/board/renesas/rzv2ma-dev |

### 7.1.2 Flash writer

Run the following commands to build the flash writer. The files shown in Table 7-2 will be generated.

- Flash writer

```
$ MACHINE=rzv2ma bitbake flash-writer
```

After running the above command, the following binary file will be generated.

**Table 7-2. Generated file (Flash writer)**

| Generated file | File name | File stored path |
|---|---|---|
| Flash writer | B2_intSW.bin | $WORK\build\tmp\work\rzv2ma-poky-linux\flash-writer\<verseion>\git\AArch64_output |

## 7.2 eMMC Boot

Here explains how to change the boot procedure from SD boot to eMMC boot.

### 7.2.1 Environmental requirement

The following table shows the required equipment and software to configure the environment for eMMC boot.

**Table 7-3. Required equipment and software**

| Equipment | | Details |
|---|---|---|
| RZ/V2MA Evaluation Board Kit | | Evaluation board kit for RZ/V2MA. |
| | SBEV-RZ/V2MA CPU Module | Target board. The main functional components for RZ/V2MA are mounted on this board. Note that the boot loader and U-Boot images are pre-written to the eMMC (THGBMJG7C1LBAIL). |
| | RZ/V2MA BASE BOARD | Board for the generation and supply of power. |
| Linux PC | | Build environment. Max 100GB of free space on HDD is necessary. |
| | OS | Ubuntu 22.04 LTS. Use a 64bit OS. |
| Windows PC | | Control the target board with terminal software. |
| | OS | Windows 10 is recommended. |
| | Terminal Software | Control the serial console of the target board.<br>Tera Term is recommended and available at "https://ttssh2.osdn.jp/index.html.en". |
| | VCP Driver | Virtual COM Port driver to enable the communication between Windows PC and the target board via USB. This is virtually used as a serial port and available at "https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers".<br>Install "CP210x Windows Drivers" at the above website. |
| USB serial to micro–USB Cable | | Serial communication (UART) between the RZ/V2MA Evaluation Board Kit and Windows PC. The type of USB serial connector on the RZ/V2MA Evaluation Board Kit is Micro USB type B. |
| micro–SDHC Card | | Use to boot the system and store applications for the RZ/V2MA. |

### 7.2.2 eMMC booting procedure

**Step 1. Run bitbake**

Run bitbake following the description in 2. Building Instructions.

**Step 2. Store required files for eMMC boot to an SD card.**

(1) Prepare an SD card for SD card boot. *Refer to 3.1 SD Card Setting for this procedure.
(2) Make an "eMMC_Image" directory in /home/root of rootfs extracted to the ext4 area in the made SD card.
(3) Store the following files in the "eMMC_Image" directory.

- Image-rzv2ma.bin
- r9a09g055-v2maevk2.dtb
- core-image-bsp-rzv2ma.tar.gz ***Use .gz file generated by bitbake**

**Step 3. Making a partition in eMMC**

(1) Boot Linux from an SD card.
(2) After booting, make eMMC partitions by the following commands in red.

```
root@rzv2ma:~# fdisk /dev/mmcblk0
Welcome to fdisk (util-linux 2.35.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): o
Created a new DOS disklabel with disk identifier 0xce03fc10.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): <Press Enter>

Using default response p.
Partition number (1-4, default 1): <Press Enter>
First sector (2048-30375935, default 2048): <Press Enter>
Last sector, +sectors or +size{K,M,G,T,P} (2048-30777343, default 30777343): +128M

Created a new partition 1 of type 'Linux' and of size 128 MiB.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): <Press Enter>

Using default response p.
Partition number (2-4, default 2): <Press Enter>
First sector (264192-30777343, default 264192): <Press Enter>
Last sector, +sectors or +size{K,M,G,T,P} (264192-30777343, default 30777343):<Press
Enter>
Created a new partition 2 of type 'Linux' and of size 14.6 GiB.

Command (m for help): w

The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

#### Step 4. Confirm eMMC partitions

Confirm eMMC partition information by the following commands. Check each size partition is the value set by the above step.

```
root@rzv2ma:~# fdisk -l /dev/mmcblk0
Disk /dev/mmcblk0: 14.69 GiB, 15758000128 bytes, 30777344 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xce03fc10

Device         Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk0p1        2048   264191   262144  128M 83 Linux
/dev/mmcblk0p2      264192 30777343 30513152 14.6G 83 Linux
```

#### Step 5. Format eMMC partitions

Format each eMMC partition by the following commands.

```
root@rzv2ma:~# mkfs.vfat -F 32 /dev/mmcblk0p1
mkfs.vfat 2.11 (12 Mar 2005)
root@rzv2ma:~# mkfs.ext4 -L rootfs /dev/mmcblk0p2
mke2fs 1.45.7 (28-Jan-2021)
Discarding device blocks:    4096/3814144               done
Creating filesystem with 3814144 4k blocks and 954720 inodes
Filesystem UUID: 64973183-cf3f-4032-9a5f-518415b9b6fd
Superblock backups stored on blocks:
   32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables:   0/117       done
Writing inode tables:   0/117       done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information:   0/117       done
```

#### Step 6. Write required files to each eMMC partition

(1) Mount the 1st partition (FAT32 area) of eMMC as follows.

```
root@rzv2ma:~# mkdir /mnt/image
root@rzv2ma:~# mount -t vfat /dev/mmcblk0p1 /mnt/image/
```

(2) Write the kernel image, and device tree to the 1st partition of eMMC as follows.

```
root@rzv2ma:~# dd if=eMMC_Image/Image-rzv2ma.bin of=/mnt/image/Image-rzv2ma.bin
36785+0 records in
36785+0 records out
18833920 bytes (19 MB) copied, 0.876929 s, 21.5 MB/s
root@rzv2ma:~# dd if=eMMC_Image/r9a09g055-v2maevk2.dtb of=/mnt/image/r9a09g055-
v2maevk2.dtb
```

(3) Mount the 2nd partition (ext4 area) of eMMC as follows.

```
root@rzv2ma:~# mkdir /mnt/rootfs
root@rzv2ma:~# mount -t ext4 /dev/mmcblk0p2 /mnt/rootfs/
[  438.161754] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode.
root@rzv2ma:~# dd if=eMMC_Image/core-image-bsp-rzv2ma.tar.gz of=/mnt/rootfs/core-image-
bsp-rzv2ma.tar.gz
```

(4) Write the rootfs to the 2nd partition of eMMC as follows.

```
root@rzv2ma:~# cd /mnt/rootfs/
root@rzv2ma:/mnt/rootfs# tar -zxvf ./core-image-bsp-rzv2ma.tar.gz
```

(5) Run the shutdown command as follows.

```
root@rzv2ma:/mnt/rootfs/# shutdown -h now
```

(6) Turn off the target board.

**Step 7. Configure U-Boot environmental variables**

Power on the target board and press Enter to move to the U-Boot command mode. Configure the following environmental variables on U-Boot command mode.

```
=> env default -a
## Resetting to default environment
=> setenv bootargs_emm 'setenv bootargs root=/dev/mmcblk0p2 rootwait rootfstype=ext4 rw'
=> setenv bootemm 'run bootargs_emm;fatload mmc 1:1 ${loadaddr} ${kernel};fatload mmc 1:1
${fdt_addr} ${fdt_file};booti ${loadaddr} - ${fdt_addr}'
=> setenv bootcmd 'run bootemm'
=> saveenv
Saving Environment to MMC... Writing to MMC(1)... OK
```

**Step 8. Boot by eMMC**

Run the following command on U-Boot command mode to boot by eMMC.

```
=> boot
```

| Revision History | RZ/V Verified Linux Package Start-Up Guide for RZ/V2MA |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Sep.16.2022 | — | First edition issued. |
| 1.10 | Jan.13.2023 | 5 | 2. Building Instructions<br>Deleted the notice for the RZ/V2MA Linux PKG V0.9.0 users. |
| | | 6-7 | 3.1.1 SD card setting for using the flash writer<br>Added the explanation on how to format an SD card for the flash writer. |
| | | 8 | Table 3-2. SD card boot files and partitions<br>Deleted the notice for the RZ/V2MA Linux PKG V0.9.0 users. |
| | | 16 | Table 4-1. Boot loader data stored in the eMMC<br>Added the note *5. |
| | | 16 | 4.2.1 Functions<br>Added the notice for the maximum transfer size the flash writer support. |
| | | 19 | Figure 4-4. Start Flash writer<br>Updated the image to V1.1.0. |
| | | 23 | Step 2. Load the U-Boot environment variables<br>Added the explanation on how to shut down in the U-Boot mode. |
| | | 24 | Figure 5-1. Root file system login screen<br>Updated the image to V1.1.0. |
| 1.20 | Jul.31.2023 | — | Changed the document title. |
| | | — | Changed the package name from “RZ/V2MA Linux Package” to “RZ/V Verified Linux Package”. |
| | | 3-6 | 2. Building Instructions<br>Changed the build steps. |
| | | 25 | Figure 5-1. Root file system login screen<br>Changed the figure and added a note. |
| | | 27 | 6. Building SDK<br>Changed the build command. |
| | | 28 | 7.1.1 Boot loader and U-Boot<br>Changed the build command. |
| | | | 7.1.2 Flash writer<br>Updated the contents. |
| 1.21 | Jul.31.2024 | 1 | 1. Introduction<br>Changed “RZ/V VLP” to “VLP/V” |
| | | 3 | 2. Building Instructions<br>Updated the command to install essential host packages<br>Updated the version of Yocto to 3.1.31<br>Updated version to 3.0.6<br>Step 1. Create a working directory and decompress the Yocto recipe and Bootloader packages<br>Updated the content |
| | | 4 | Removed Step 2: Apply the additional patch<br>Step 4. Decompress OSS file to “build” directory (Optional)<br>Updated the content |

| | | | |
|---|---|---|---|
| | | 5 | Table 2-2. Image files for RZ/V2MA<br>Changed device tree file names |
| | | 8 | Table 3-2. SD card boot files and partitions<br>3.1.4. Prepare for booting from SD card -Step2<br>Changed device tree file name |
| | | 11 | Updated environment settings to boot from the SD card |
| | | 28 | 7.2.2. eMMC booting procedure -Step2<br>Changed device tree file name |
| | | 28-31 | 7.2.2 eMMC booting procedure<br>Changed eMMC partitions from “mmcblk1” to “mmcblk0” |
| | | 31 | 7.2.2. eMMC booting procedure -Step6<br>Changed device tree file name |
| | | — | Updated document reversion |
| 1.22 | Jun.15.2025 | 2 | Updated the OS information |
| | | 5 | Updated the Codec build instruction |
| | | 12 | Add Codec setting in Sd bootargs |
| | | 28 | Table 7-3. Updated the OS information |
| | | — | Updated document reversion |

RZ/V Verified Linux Package Start-Up Guide for RZ/V2MA

Publication Date: Rev.1.22 Jun.15.2025

Published by: Renesas Electronics Corporation

# RZ/V Verified Linux Package Start-Up Guide for RZ/V2MA

Renesas Electronics Corporation

R01US0578EJ0122