User's Manual

RZ/V Verified Linux Package Start-Up Guide for RZ/V2M

RZ/V2M

Renesas Electronics
www.renesas.com

Rev.1.40 Jul 2023

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

Trademarks

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

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/.

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/V2M 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/V2M.

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/V2M Linux Package instruction.	RZ/V Verified Linux Package Software Manual for RZ/V2M
Usage guide	Guide how to use the RZ/V2M 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 “RZ/V VLP”) on the RZ/V2M 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
- How to create a software development kit (SDK)

This RZ/V2M Yocto recipe is a basic package to operate built-in Linux and software on the RZ/V2M Evaluation Board Kit. Please contact your Renesas representatives if you have any questions about this package.

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.

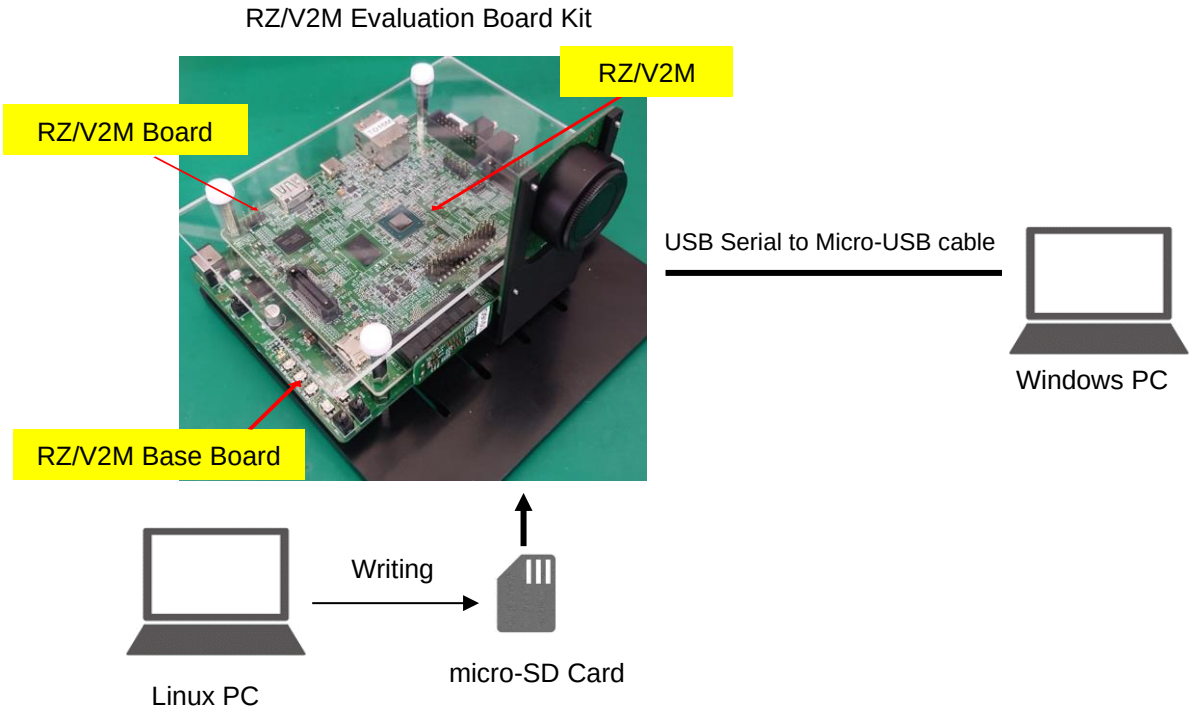


Figure 1-1. Recommend environment

Table 1-1. Required equipment and software

Equipment		Details
RZ/V2M Evaluation Board Kit		The evaluation kit for RZ/V2M.
	RZ/V2M Board (Main)	Target board. The main functional components for RZ/V2M are mounted on this board. Note that the boot loader and U-Boot images are pre-written to the eMMC (THGBMJG7C1LBAIL).
	RZ/V2M Base Board (Base)	The board for the generation and supply of power. Connected to CN12 and CN 13 on the RZ/V2M Board.
Linux PC		Used as build/debug environment. Max 100GB of free space on HDD is necessary.
	OS	Ubuntu 20.04 LTS. Use a 64bit OS.
Windows PC		Control the target board with terminal software.
	OS	Windows 10 recommended.
	Terminal Software	Control serial console of the target board. 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“. Download and install “CP210x VCP Windows” at the above web site.
USB serial to micro–USB Cable		Serial communication (UART) between the RZ/V2M Evaluation Board Kit and Windows PC. The type of USB serial connector on the RZ/V2M Evaluation Board Kit is Micro USB type B.
micro–SD Card		Use to boot the system, store applications for the RZ/V2M. **Note that use a micro-SDHC card for the flash writer.**

2. Building Instructions

This section describes the instructions to build the Board Support Package (BSP) for RZ/V2M.

Before starting the build, run the commands below on the Linux Host PC and install essential host packages 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/3.1.21/brief-yoctoprojectqs/brief-yoctoprojectqs.html (*)

Note: The above URL document is for the yocto version supported in the VLP3.0.4. 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 following commands. Here creates a working directory named "rzv2m_bsp" in the user's home directory as an example. Change the name and location of the directory as needed.

```
$ mkdir ~/rzv_vlp_<package version>
$ export WORK=/home/<user>/rzv_vlp_<package version>
$ cd $WORK
$ unzip ~/<package download directory>/RTK0EF0045Z0024AZJ-<package version>.zip
$ tar zxvf ./RTK0EF0045Z0024AZJ-<package version>/rzv_vlp_<package version>.tar.gz
$ unzip ~/<package download directory>/RTK0EF0045Z89001ZJ-<Bootloader PKG version>.zip
$ tar zxvf ./RTK0EF0045Z89001ZJ-<Bootloader PKG version>/meta-rz-
features_v2m_bl_<Bootloader PKG version>.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. <package version> is the RZ/V VLP version number in the file name. (e.g. if VLP version 3.0.4, <package version> is v3.0.4.) Decompress the latest version of the RZ/V VLP. Refer to the release note for the appropriate file.
3. <Bootloader_PKG_ver> is the RZV2M Bootloader Packages 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. Apply the additional patch (Only for RZ/V VLP3.0.4)

Run the following command to apply the patch.

Note: Skipping the step may result in
- the image_buf (udmabuf) in the RZ/V2M memory map not working properly.
- a failed build.

```
$ cd $WORK/meta-renesas
$ cp ../extra/0001-apply-fix-pcie-memory-map.patch .
$ cp ../extra/0001-rzv2m-flash-writer-fix-do_compile-setting.patch .
$ patch -p1 < ./0001-apply-fix-pcie-memory-map.patch
$ patch -p1 < ./0001-rzv2m-flash-writer-fix-do_compile-setting.patch
```

Please confirm whether the patch has been applied correctly as below.

[1] For the patch "0001-apply-fix-pcie-memory-map.patch"

(1) The following patches described in the following, marked in red, have been in the directory "rzv2m_patch".

```
$ ls ./meta-rzv2m/recipes-kernel/linux/linux-renesas/patches/rzv2m_patch/
0000-arch-arm64-boot-dts-renesas-Fixed-memory-map.patch
0001-drivers-pci-Update-the-PCIE_CMA_ADDRESS.patch
```

(2) The definitions described in the following, marked in red, have been added in the "linux-renesas_5.10.bbappend" file.

```
$ cat ./meta-rzv2m/recipes-kernel/linux/linux-renesas_5.10.bbappend
COMPATIBLE_MACHINE_append = "(rzv2m|rzv2ma)"

BRANCH_rzv2m = "v2m/rz-5.10-cip29"
SRCREV_rzv2m = "1f9621c381b9b809ede3532306d9eb3acca0974c"

BRANCH_rzv2ma = "v2ma/rz-5.10-cip29"
SRCREV_rzv2ma = "6f914ec736ba254524c47ac4864e933b89ddfcd7"

FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

SRC_URI_append = " \
   file://patches/rzv2m_patch/0000-arch-arm64-boot-dts-renesas-Fixed-memory-map.patch \
   file://patches/rzv2m_patch/0001-drivers-pci-Update-the-PCIE_CMA_ADDRESS.patch \
```

[2] For the patch "0001-rzv2m-flash-writer-fix-do_compile-setting.patch"

(1) The following patches described in the below, marked in red, have been in the following directory.

```
$ ls $WORK/meta-renesas/meta-rzv2m/recipes-bsp/flash-writer/files/
0001-makefile.linaro-update-makefile-to-be-built-with-cro.patch
```

(2) The definitions described in the following, marked in red, have been updated in the " flash-writer.bb" file.

```
$ cat $WORK/meta-renesas/meta-rzv2m/recipes-bsp/flash-writer/flash-writer.bb
…
SRC_URI_append = " \
    file://0001-makefile.linaro-update-makefile-to-be-built-with-cro.patch \
"

S = "${WORKDIR}/git"

CFLAGS_prepend = " -fno-stack-protector"

do_compile () {
    cd ${S}
    oe_runmake -f makefile.linaro
}
…
```

Step 3. 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 4. Add layers for the bootloaders

Run the following command to build RZ/V2M bootloaders.

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

Step 5. Decompress OSS files to “build” directory (Optional)

*When building the bsp online, go to the next step because this step is not mandatory.

Store the Open Source Package in your home directory and run the commands below. Decompressed all OSS packages with this 7z command.

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

Note: If this step is omitted or BB_NO_NETWORK is set to “0” in the next step, all source codes will be downloaded from the repositories of each OSS via the internet when running the bitbake command. Note that if you use an “online” environment, a build may fail due to the implicit changes of the repositories of OSS.

After completing the decompression of the OSS package, the “offline” environment is ready. When preventing the network access, please change the following variables in the “$work/build/conf/local.conf”:

```
BB_NO_NETWORK = "1"
```

Note: When BB_NO_NETWORK is set to 0, the build is executed online.

Step 6. 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.

```
$ MACHINE=rzv2m 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 RZ/V2M DRP-AI Support Package and RZ/V2M ISP Support Package, apply all necessary recipes required for the build environments for these packages before executing bitbake commands. Refer to each package's document to know how to apply the patches and build.

Table 2-1. core-image-target

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 3984* tasks of which 554 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/rzv2m directory.

Table 2-2. Build image files for RZ/V2M

Generated files	File name	File stored path
Device tree file	r9a09g011gbg-evaluation-board.dtb	$WORK/build/tmp/deploy/images/rzv2m
Linux kernel image	Image-rzv2m.bin	
rootfs	<*image-name*>*1-bsp-rzv2m.tar.bz2	
1st loader binary*2	loader_1st_128kb.bin	
Boot parameter for 2nd loader*2	loader_2nd.bin	
2nd loader binary*2	loader_2nd_param.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 6.
2. Store these loader binaries in the appropriate partition on the SD card when writing loader binaries.

3. Preparations

This chapter describes the required preparation before running the software on the RZ/V2M Evaluation Board Kit. Note that the booting is from an SD card in the following procedure. For booting from the eMMC, refer to 8.2 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/V2M Evaluation Board.

3.1.1 SD Card Setting for using the flash writer

If you already write this version of bootloader/U-Boot binaries on the eMMC, skip this procedure and go to 3.1.2.

3.1.1.1 Files for the flash writer

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. The file and micro-SD card partition for the flash writer

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/
$ sudo umount /media/
```

3.1.2 SD Card Setting for booting Linux

3.1.2.1 Files for SD card booting

The files for booting from the SD card are as follows. To boot from the SD card, create partitions with the specified file system format, and store files in each partition on the SD card.

Table 3-2. SD card boot files and partitions

Partition No.	Size	File system format	File name	Description
1	128MB or more	FAT	Image-rzv2m.bin*	Linux kernel image. ** Notice for the previous version users:* *For the Linux kernel Image file, use Image-rzv2m.bin from this version.*
			r9a09g011gbg-evaluation-board.dtb*	Device tree binary.
2	The rest	ext4	core-image-bsp-rzv2m.tar.bz2*	Root file system image.

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

3.1.2.2 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/rzv2m.

  — Linux kernel image (Image-rzv2m.bin) and device tree (r9a09g011gbg-evaluation-board.dtb)

```
$ sudo mount /dev/sdb1 /media/
$ sudo cp $WORK/build/tmp/deploy/images/rzv2m/Image-rzv2m.bin /media/
$ sudo cp $WORK/build/tmp/deploy/images/rzv2m/r9a09g011gbg-evaluation-board.dtb /media/
$ 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 each file.

  — Root file system image (core-image-bsp-rzv2m.tar.bz2)

```
$ sudo mount /dev/sdb2 /media/
$ cd /media/
$ sudo tar jxvf $WORK/build/tmp/deploy/images/rzv2m/core-image-bsp-rzv2m.tar.bz2
$ sync
$ cd $WORK
$ sudo umount /media/
```

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

3.1.2.3 Set U-Boot Environment Variables

Connect the RZ/V2M 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/V2M Evaluation Board Kit and U-Boot start. When the countdown begins on the console, press Enter key to move 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
=>  saveenv
=>  run bootsd
```

3.2 Board Setting

3.2.1 Switch

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

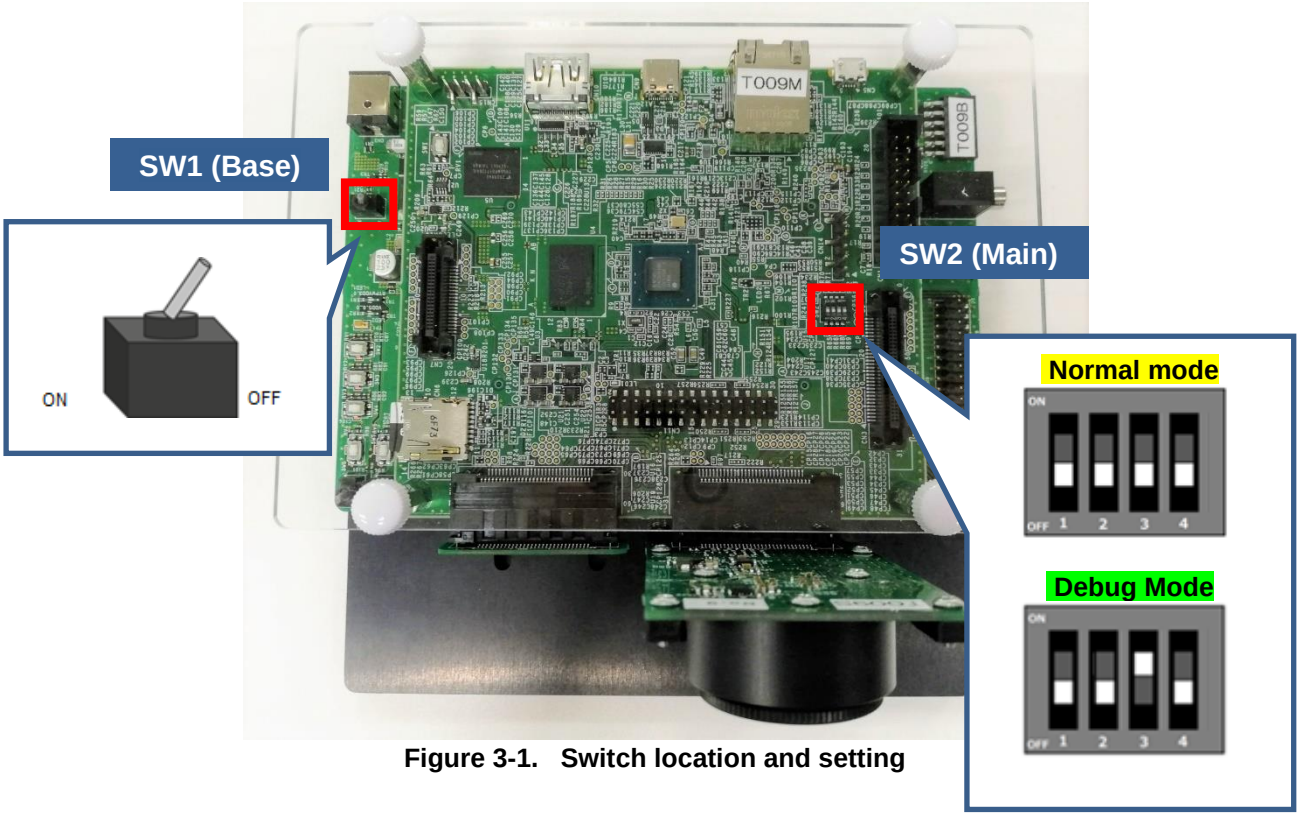


Figure 3-1. Switch location and setting

Table 3-3. SW2(Main) on RZ/V2M Board setting

Mode	Switch 1	Switch 2	Switch 3	Switch 4
Normal	OFF	OFF	OFF	OFF
Debug	OFF	OFF	ON	OFF

3.2.2 Terminal software setting

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

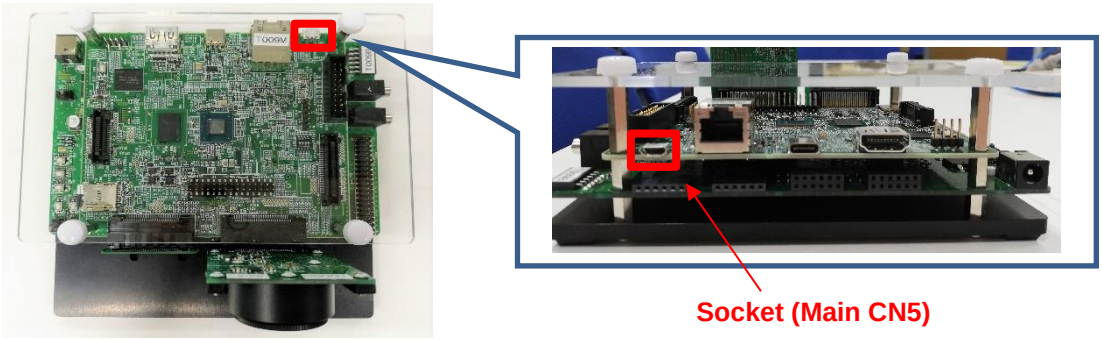


Figure 3-2. Serial to micro-USB cable connection

Two serial ports are detected on the PC after injecting the cable.

Choose “Silicon Labs Dual CP2105 USB to UART Bridge: **Standard** COM Port”.

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

(Menu > Setup > Serial port)

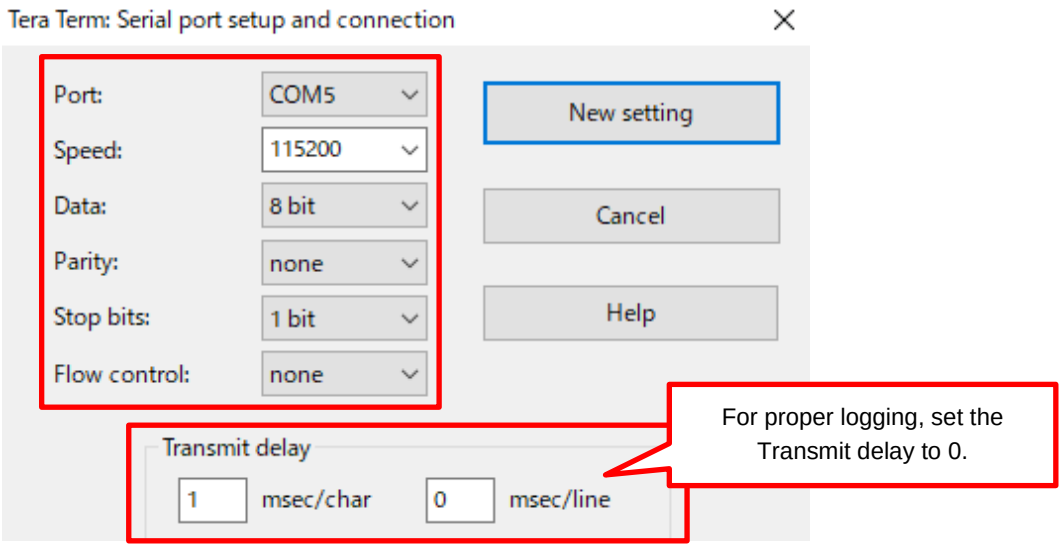


Figure 3-3. Terminal software setting

(Menu > Setup > Terminal)

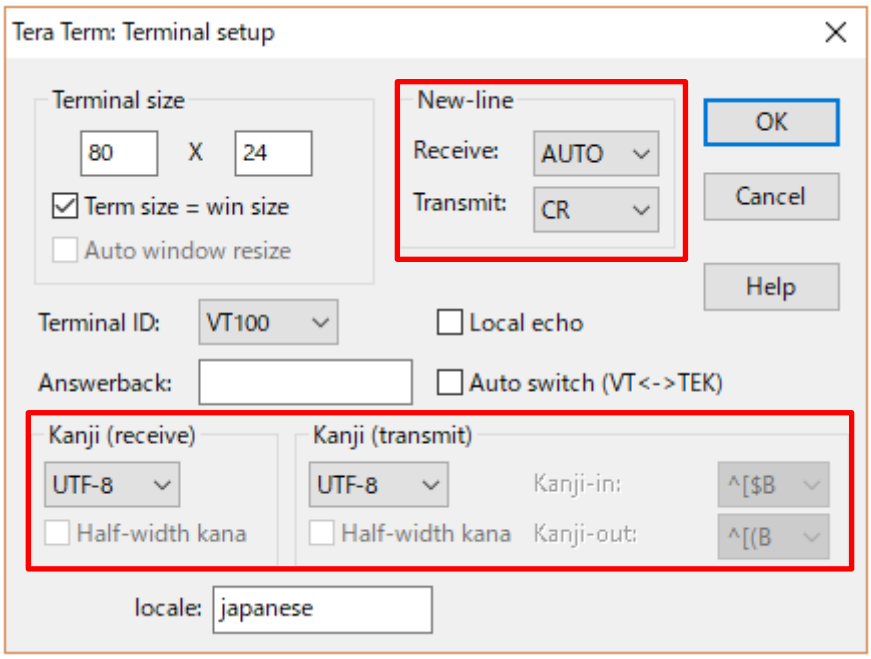


Figure 3-4. Terminal software setting

3.2.3 Insert a micro-SD

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

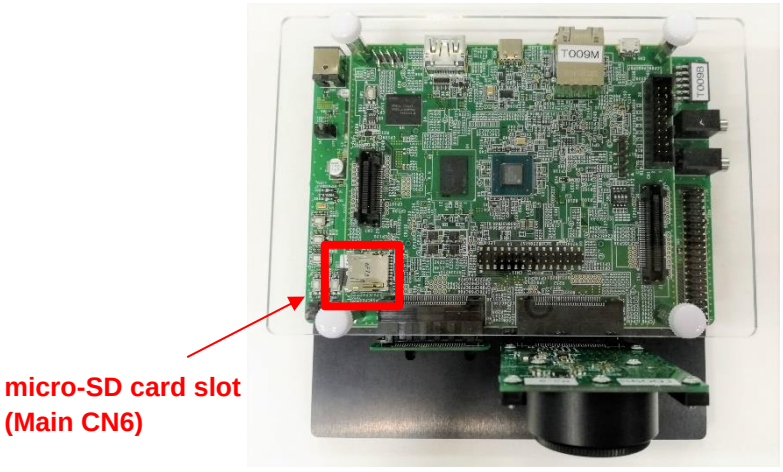


Figure 3-5. micro-SD card slot

Note: The booting data should be written to the SD card beforehand to run the SD card booting on the RZ/V2M Evaluation Board Kit.

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).

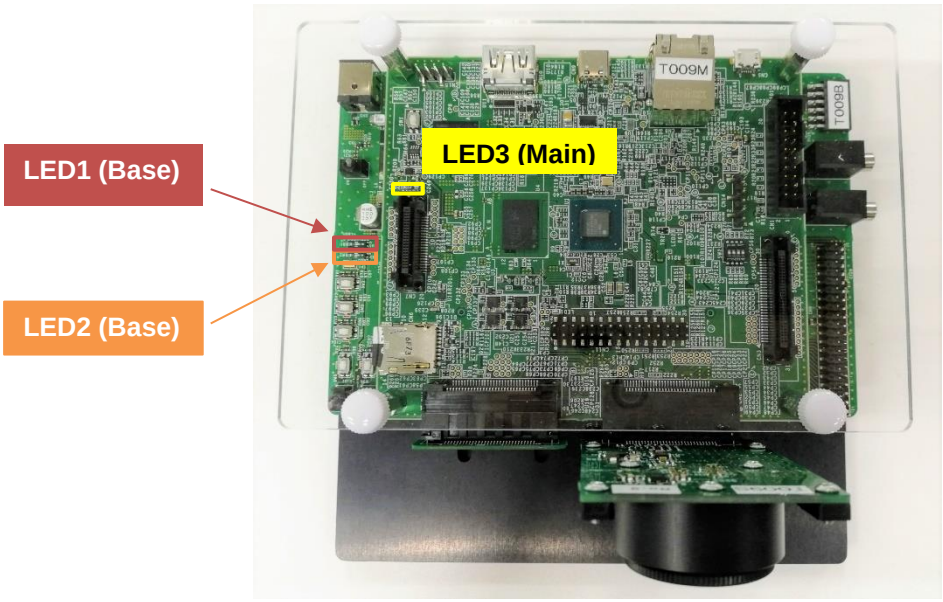


Figure 3-6. LEDs

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 8.1.1.

4.1 Boot loader and U-Boot Images

The boot loader and U-Boot binaries are stored in the eMMC on RZ/V2M in advance. The bootloader and U-Boot will be booted after powering on the RZ/V2M Evaluation Board Kit.

The following table lists the address information stored in the eMMC on RZ/V2M. The boot loader/U-Boot binary files below are generated 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
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, Boot loader and U-Boot will not use RAMA and RAMB.
4. The environment variables of U-Boot are stored in boot partition 2.

4.2 Flash Writer

Flash writer is the software for writing loader binaries to the eMMC on the RZ/V2M Evaluation Board Kit via a PC. **Here uses the micro-SDHC card the flash writer stored in 3.1.1.** For building the flash writer from the source codes, refer to 8.1.2.

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 are as follows.

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

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/V2M 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 “Standard COM Port” on the terminal software.

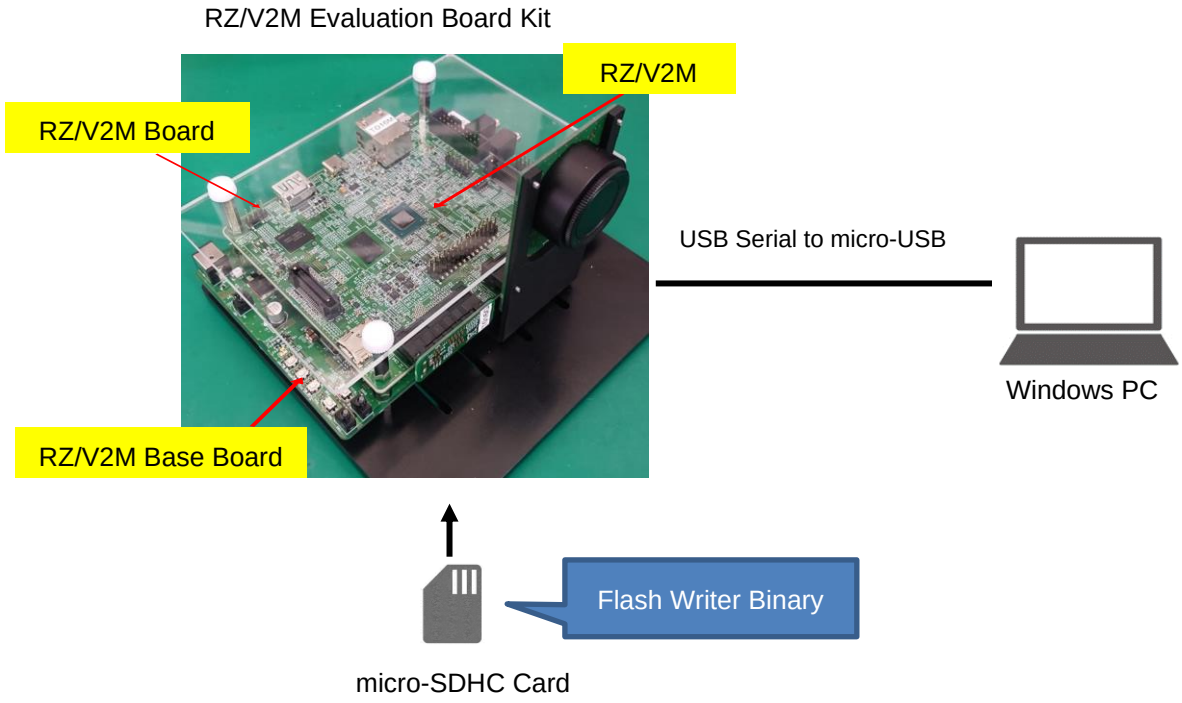


Figure 4-1. Equipment setting

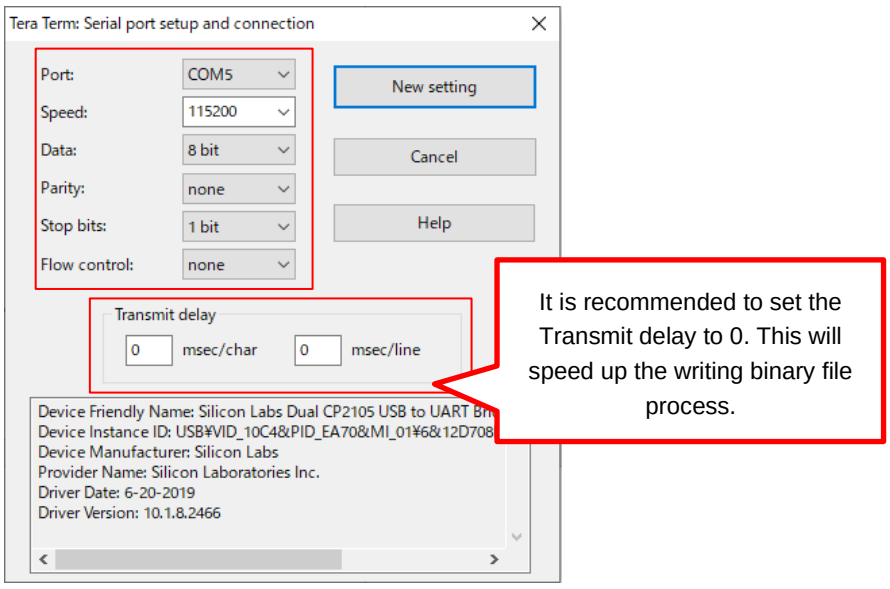


Figure 4-2. Terminal software setting

Step 2. Write the flash writer to eMMC

Start RZ/V2M with the forced write mode and write the flash writer binary to the eMMC as the following steps.

Note that refer to "RZ/V2M User's Manual: Hardware" about the forced write mode for details.

(1) Insert the micro-SDHC card prepared in 3.1.1 into the micro-SD card slot on the RZ/V2M Evaluation Board Kit.
(2) Set the SW2(Main) on the RZ/V2M Board as shown in Figure 4-2 and Figure 4-3 to change the board operation mode to "forced write mode".

Table 4-2. SW2(Main) setting for the forced write mode

Switch 1	Switch 2	Switch 3	Switch 4
OFF	OFF	OFF	ON

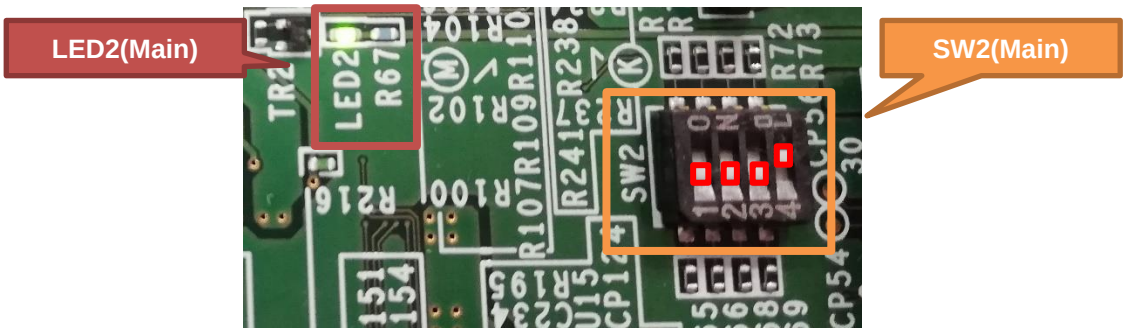


Figure 4-3. LED lights in the forced write mode (RZ/V2M Board)

(3) Power on the RZ/V2M Evaluation Board Kit. Start RZ/V2M in forced write mode and write the Flash writer binary from the micro-SD card to eMMC.
(4) Check the lighting of LED2(Main). If the LED lights as shown in the above figure, the writing to the eMMC is completed successfully. On the other hand, if the LED is blinking, writing the Flash writer binary is failed.
(5) Power off the RZ/V2M Evaluation Board Kit.

Step 3. Start Flash writer

Start RZ/V2M in normal mode and run Flash writer.

(1) Check that SW2(Main) is set to the normal mode [1:OFF, 2:OFF, 3:OFF, 4:OFF].
(2) Power on the RZ/V2M Evaluation Board Kit. The following log will appear if RZ/V2M starts in normal mode and run Flash writer successfully.

```
Flash writer for RZ/V2M <Version> <Date>
>
```

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 and input the eMMC save sectors, file size, and send binary files shown in Table 4-1 as follows. In this procedure, binary files are sent by Tera Term.

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

```
>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
```

Appear this message when the writing process is successful.

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

```
 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....
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.

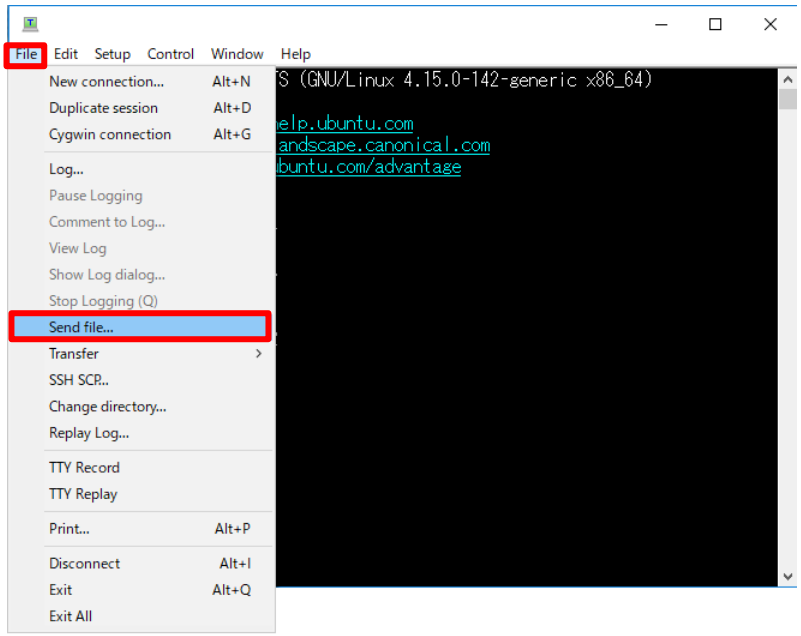


Figure 4-4. Send files by Tera Term

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

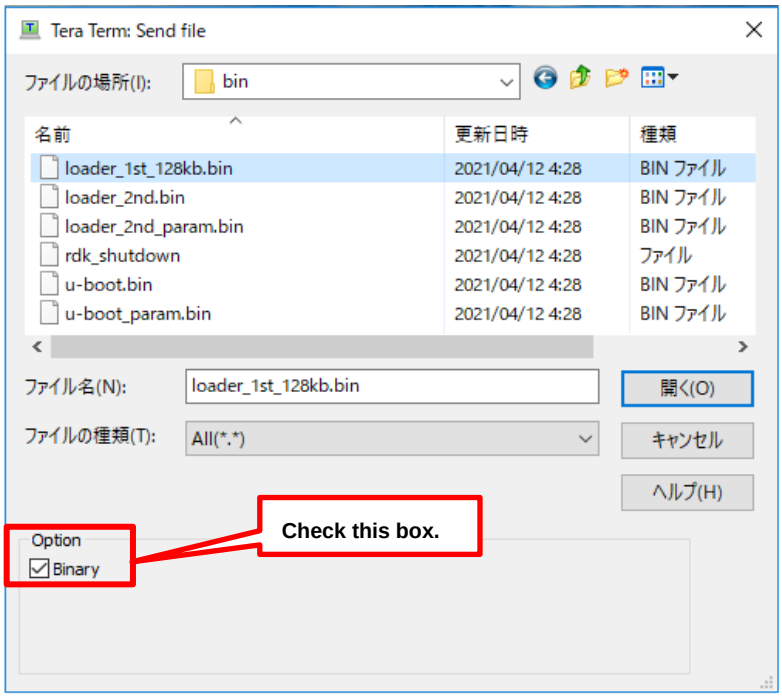


Figure 4-5. Send files by Tera Term

(3) Power off the RZ/V2M 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) Check that SW2(Main) is set 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/V2M 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/V2M Evaluation Board Kit and run the system. **Here uses the micro-SD card prepared in 3.1.2.**

5.1 Power on the board

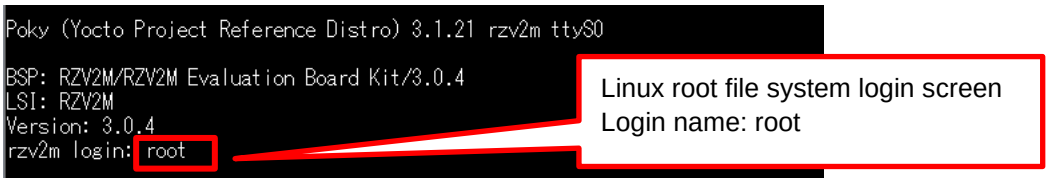
Notice: Before connecting the AC adapter (CN503) to an electrical outlet, make sure that the SW1 (Base) for the power supply is turned off.

When the AC adapter is connected, the LED1 (Base) will light up. After turning on SW1 (Base), LED3 (Main) and LED2 (Base) will light up, and RZ/V2M 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.

To power down the system, follow the step below.

Step 1. Run shutdown command

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

```
root@rzv2m:~# shutdown -h now
```

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

Step 2. Confirm the power-off

After running shutdown command, confirm LED2 (Base) and LED3 (Main) are turned off.

Step 3. Turn off the power switch on the board

After checking the LED2 (Base) and LED3 (Main), turn SW1(Base) off.

Note: Be sure to follow the steps correctly when shut down the system.

[1] run shutdown command.

[2] Turn SW1(Base) off.

[3] Remove 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 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/V2M DRP-AI Support Package and RZ/V2M ISP 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=rzv2m 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-rzv2m-toolchain-<version>.sh
```

Set up environment variables as follows.

```
$ source /opt/poky/<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
$ MACHINE=rzv2m bitbake core-image-bsp -c cleanall
$ MACHINE=rzv2m bitbake core-image-bsp
$ MACHINE=rzv2m bitbake core-image-bsp -c populate_sdk
```

7. Application Building and Running

This chapter explains how to create and run an application on the target board using this package.

7.1 Create an application

Here is an example of how to create an “Hello, World!” application to run on RZ/V2M.
Note that you must build (bitbake) a core image for the target and prepare the SDK before building an application. Refer to 2.Building Instructions for the building an image and 6.Building SDK for building and installing SDK.

Step 1. Create a working directory for the application on the Linux host PC.

```
$ mkdir ~/hello_apl
$ cd ~/hello_apl
```

Step 2. Create the following three files (an application file, an Makefile, and a configure file) in the directory for the application. Here, the application is built using automake and autoconf.

- hello.c

```
#include <stdio.h>

int main()
{
  printf("Hello, world!\n");
  return 0;
}
```

- Makefile.am

```
bin_PROGRAMS = hello
hello_SOURCES = hello.c
```

- configure.ac

```
AC_INIT(hello.c)
AM_INIT_AUTOMAKE(hello,0.1)
AC_PROG_CC
AC_PROG_INSTALL
AC_OUTPUT(Makefile)
```

Step 3. Generate the configuration scripts, files needed by GNU coding standards, and the configure files. After that, cross-compile the application.

```
$ aclocal
$ autoconf
$ touch NEWS README AUTHORS ChangeLog
$ automake -a
$ ./configure $CONFIGURE_FLAGS
```

Step 4. Make the application by the generated makefile.

```
$ make
```

After make, confirm that the executable file (the filename is “hello”) is created in the hello_apl folder. This application must be cross-compiled for AArch64.

7.2 Store an sample application

Store the created application on the root file system (ext4 partition) as follows.

```
$ sudo mount /dev/sdb2 /media/
$ cd /media/usr/bin
$ sudo cp ~/hello_apl/hello .
$ sudo chmod +x hello
```

Note: “sdb2” (above in red) may depend on using system.

7.3 Run a sample application

Power on the RZ/V2M Evaluation Board Kit and start the system.

After booting Linux, run the sample application with the following command.

```
root@rzv2m:~# hello
Hello, world!
```

8. Appendix

8.1 Building Instructions

8.1.1 Boot loader and U-Boot

Before building loader binaries, follow the steps from Step 1 to Step 5 described in 2.Building Instructions. After that, run the following commands to build boot loader and U-Boot.

- Boot loader

```
$ MACHINE=rzv2m bitbake bootloader
```

- U-Boot

```
$ MACHINE=rzv2m bitbake u-boot
```

After running the above commands, the following files will be generated.

Table 8-1. 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/rzv2m
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/rzv2m-poky-linux/bootloader/<version>/source
U-Boot source codes	*Omitted	$WORK/build/tmp/work/rzv2m-poky-linux/u-boot/<version>/git

8.1.2 Flash writer

Run the following commands to build the flash writer. The files shown in Table 8-2 will be generated.

- Flash writer

```
$ MACHINE=rzv2m bitbake flash-writer
```

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

Table 8-2. Generated file (Flash writer)

Generated file	File name	File stored path
Flash writer	B2_intSW.bin	$WORK\build\tmp\work\rzv2m-poky-linux\flash-writer\<verseion>\git\AArch64_output

8.2 eMMC Boot

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

8.2.1 Environmental Requirement

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

Table 8-3. Required equipment and software

Equipment		Details
RZ/V2M Evaluation Board Kit		Evaluation kit for RZ/V2M.
	RZ/V2M Board (Main)	Target board. The main functional components for RZ/V2M are mounted on this board. Note that the boot loader and U-Boot images are pre-written to the eMMC (THGBMJG7C1LBAIL).
	RZ/V2M Base Board (Base)	Connected to CN12 and CN 13 on the RZ/V2M Board. Board for the generation and supply of power.
Linux PC		Build environment. Max 100GB of free space on HDD is necessary.
	OS	Ubuntu 20.04 LTS. Use a 64bit OS.
Windows PC		Control the target board with terminal software.
	OS	Windows 10 recommended.
	Terminal Software	Control serial console of the target board. 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“. Install “CP210x VCP Windows” at the above web site.
USB serial to micro–USB Cable		Serial communication (UART) between the RZ/V2M Evaluation Board Kit and Windows PC.
micro–SD Card		Use to boot the system, store applications for the RZ/V2M.

8.2.2 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 SD card.
(3) Store the following files in the “eMMC_Image” directory.

- Image-rzv2m.bin
- r9a09g011gbg-evaluation-board.dtb
- core-image-bsp-rzv2m.tar.gz ***Use .gz file generated by bitbake**

Step 3. Making a partition in eMMC

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

```
root@rzv2m:~# fdisk /dev/mmcblk1
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.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xbc51e838.

Command (m for help): o
Created a new DOS disklabel with disk identifier 0x57299e86.

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@rzv2m:~# fdisk -l /dev/mmcblk1
Disk /dev/mmcblk1: 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: 0x57299e86

Device         Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk1p1        2048   264191   262144  128M 83 Linux
/dev/mmcblk1p2      264192 30777343 30513152 14.6G 83 Linux
```

Step 5. Format eMMC partitions

Format each eMMC partition by the following commands.

```
root@rzv2m:~# mkfs.vfat -F 32 /dev/mmcblk1p1
mkfs.vfat 2.11 (12 Mar 2005)
root@rzv2m:~# mkfs.ext4 -L rootfs /dev/mmcblk1p2
mke2fs 1.45.7 (28-Jan-2021)
Discarding device blocks:    4096/3814144/3814144                done
Creating filesystem with 3814144 4k blocks and 954720 inodes
Filesystem UUID: a61ad36c-a0c7-4a50-8033-906e7f78f762
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@rzv2m:~# mkdir /mnt/image
root@rzv2m:~# mount -t vfat /dev/mmcblk1p1 /mnt/image/
```

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

```
root@rzv2m:~# dd if=eMMC_Image/Image-rzv2m.bin of=/mnt/image/Image-rzv2m.bin
root@rzv2m:~# dd if=eMMC_Image/r9a09g011gbg-evaluation-board.dtb
of=/mnt/image/r9a09g011gbg-evaluation-board.dtb
```

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

```
root@rzv2m:~# mkdir /mnt/rootfs
root@rzv2m:~# mount -t ext4 /dev/mmcblk1p2 /mnt/rootfs/
[  438.161754] EXT4-fs (mmcblk1p2): mounted filesystem with ordered data mode.

root@rzv2m:~# dd if=eMMC_Image/core-image-bsp-rzv2m.tar.gz of=/mnt/rootfs/core-image-
bsp-rzv2m.tar.gz
```

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

```
root@rzv2m:~# cd /mnt/rootfs/
root@rzv2m:/mnt/rootfs# tar -zxvf core-image-bsp-rzv2m.tar.gz
```

(5) Run the shutdown command as follows.

```
root@rzv2m:/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/mmcblk1p2 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/V2M

Rev.	Date	Description	
		Page	Summary
1.00	Jun.30.2021	—	First edition issued.
1.10	Oct.8.2021	6	3. Preparations Added the notice about eMMC boot in the head of this page.
		14	Table 4-1. Boot loader data stored in the eMMC Updated the file sizes of loader_2nd.bin and u-boot.bin for Version 1.1.0.
		15	4.2, Step 3. Start U-Boot Added the notice when powering on.
		15	Figure 4-1. Text box to select the target version Updated the figure for Version 1.1.0.
		17-20	4.2, (1) Case of eMMC Writes Succeed Updated the log of eMMC writer script for Version 1.1.0.
		22	4.2, Step 5. Shutdown Added the notice when powering off.
		23	Figure 5-1. Root file system login screen Updated the figure for Version 1.1.0.
		24	5.3, Step 3. Turn off the power switch on the board Added the notice when powering off.
		29	Table 7-4. Boot loader data stored in the eMMC Updated the file sizes of loader_2nd.bin and u-boot.bin for Version 1.1.0.
		30-32	Step 4. Write loader binaries to eMMC with Flash writer, (2) Updated the log of Flash writer for Version 1.1.0.
		36-39	8.2 eMMC Boot Added the description of how to boot from eMMC.
1.20	Jan.28.2022	-	Changed the name of target board from “RZ/V2M Evaluation Kit” to “RZ/V2M Evaluation Board Kit”.
		2, 36	Table 1-1. Required equipment and software Table 8-2. Required equipment and software Updated the supported Linux OS version due to the kernel updating in V1.2.0.
		4	Step 6. Start the build Added the explanation for the warning related to the distribution on Linux Host PC when building.
		7-8	3.1.2 Step 1. Create partitions on SD card Updated the log of creating partitions due to the change of Linux OS (Ubuntu 18.04).
		11	Figure 3-3. Terminal software setting Added the note on setting the Transmit delay.
		14, 30	Table 4-1. Boot loader data stored in the eMMC Table 7-4. Boot loader data stored in the eMMC Updated the file sizes of loader_2nd.bin and u-boot.bin for Version 1.2.0.
		15	Figure 4-1. Text box to select the target version Updated the figure for Version 1.2.0.
		17-20	4.2, (1) Case of eMMC Writes Succeed Updated the log of eMMC writer script for Version 1.2.0.

		23	Figure 5-1. Root file system login screen Updated the figure for Version 1.2.0.
		25	Updated the URL of the reference.
		28	Figure 7-2. Terminal software setting Added this figure to explain the setting for Transmit delay.
		29	Figure 7-4. Start the flash writer Updated the figure.
		30	Table 7-3. Boot loader data stored in the eMMC Updated this figure for Version1.2.0.
		31-33	Step 4. Write loader binaries to eMMC with Flash writer, (2) Updated the log of Flash writer for Version 1.2.0.
		34	Table 7-4. Error messages (Flash writer) Added a list of the flash writer error messages.
		35	8.1.1 Offline build, Step 1. Extract OSS packages Changed the name of the OSS package file.
		36	Table 8-1. Generated files (Boot loader and U-Boot) Fixed the file stored path of the boot loader source codes.
		41	8.3 rdk_shutdown Created a new section for describing how to build rdk_shutdown.
1.30	14.Oct.2022	-	Changed the structure of this document due to the Linux kernel update, and changing the contents of the RZ/V2M Linux package.
		-	Updated the log and log images for v1.3.0 due to the version up.
		2, 28	Table 1-1. Required equipment and software, Table 7-4. Required equipment and software Updated the supported Ubuntu version for building the bsp from 18.04 to 20.04.
		3-5	2. Building Instructions Correct some descriptions in the build procedure. Here des
		5,8, 10 28, 30	Table 2-2. Build image files for RZ/V2M, Table 3-2. SD card boot files and partitions, Step 2. Store system files to the SD Card, 7.2.2 eMMC booting procedure, Step 6. Write required files to each eMMC partition Changed the name of the kernel image file from “Image” to “Image-rzv2m.bin”.
		6-7	3.1.1 SD Card Setting for using the flash writer Added this section to explain how to prepare the micro-SD card for the flash writer.
		8-10, 28, 30-31	3.1.2 SD Card Setting for booting Linux, 7.2.2 eMMC booting procedure Changed the file format in the 2nd partition (rootfs area) from ext3 to ext4.
		14-22	4. Boot Loader, U-Boot (Loader Binaries) Changed the method of writing Bootloader/U-Boot to eMMC to Flash writer (removed the method of writing from a ttl file.)
		24, 31	5.3 Shutdown the Board, 7.2.2 eMMC booting procedure (Previous) run “rdk_shutdown”. (Now) run the Linux “shutdown” command.
		25	6. Building SDK Changed the title of this section and updated the explanation due to updating the supported kernel version.
1.40	Jul.31.2023	—	Changed the document title.
		—	Changed the package name from “RZ/V2M Linux Package” to “RZ/V Verified Linux Package”.

		3-6	2. Building Instructions Changed the build steps.
		25	Figure 5-1. Root file system login screen Changed the figure and added a note.
		27	6. Building SDK Changed the build command.
		28	7. Application Building and Running New added.
		30	7.1.1 Boot loader and U-Boot Changed the build command.
			7.1.2 Flash writer Updated the contents.

RZ/V Verified Linux Package Start-Up Guide for RZ/V2M

Publication Date: Rev.1.40 Jul.31.2023

Published by: Renesas Electronics Corporation

RZ/V Verified Linux Package Start-Up Guide for RZ/V2M

Renesas Electronics Corporation

R01US0528EJ0140