Reference Boards of RZ/G2H, RZ/G2M, RZ/G2N, and RZ/G2E

R01US0555EJ0102
Rev.1.02
Oct. 31, 2023

Linux Start-up Guide

Introduction

This document provides a guide to prepare RZ/G2 reference boards to boot up with the Verified Linux Package.

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)

Target Reference Board

RZ/G2 Group reference boards

- Hoperun Technology HiHope RZ/G2H platform (hihope-rzg2h)
- Hoperun Technology HiHope RZ/G2M platform (hihope-rzg2m)
- Hoperun Technology HiHope RZ/G2N platform (hihope-rzg2n)
- Silicon Linux RZ/G2E evaluation kit (EK874)

Target Software

- RZ/G Verified Linux Package version 3.0.5 or later. (hereinafter referred to as “VLP/G”)

Contents

1. Environment Requirement

The environment for building the Board Support Package (hereinafter referred to as “BSP”) is listed in Table 1. Please refer to the below documents for details about setting up the environment:

A Linux PC is required for building the software.

A Windows PC can be used as the serial terminal interface with software such as TeraTerm.

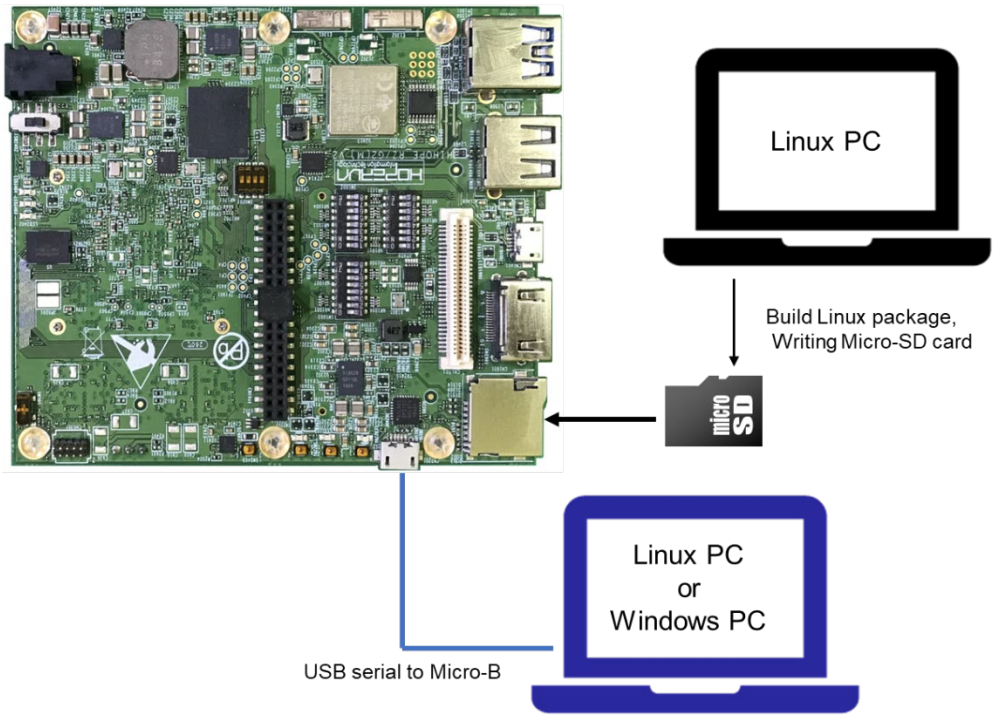


Figure 1. Recommend environment.

Note: The board shown in Figure 1 is RZ/G2N, but RZ/G2H,M,E has the same structure.

Table 1. Equipment and Software for Developing Environments of RZ/G Linux Platform

Equipment	Description
Linux Host PC	Used as build/debug environment 100GB free space on HDD or SDD is necessary
OS	Ubuntu 20.04 LTS 64 bit OS must be used. 20.04 inside a docker container also OK.
Windows Host PC	Used as debug environment, controlling with terminal software
OS	Windows 10 or Windows 11
Terminal software	Used for controling serial console of the target board Tera Term (latest version) is recommended Available at https://ttssh2.osdn.jp/index.html.en
VCP Driver	Virtual COM Port driver which enables to communicate Windows Host PC and the target board via USB which is virtually used as serial port. Available at: • http://www.ftdichip.com/Drivers/VCP.htm (for ek874) • https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers (for hihope-rzg2h, hihope-rzg2m, and hihope-rzg2n) Please install the VCP Driver corresponding to the target board.
USB serial to micro–USB Cable	Serial communication (UART) between the Evaluation Board Kit and Windows PC. The type of USB serial connector on the Evaluation Board Kit is Micro USB type B.
micro–SD Card	Use to boot the system, and store applications. Note that use a micro-SDHC card for the flash writer.

Most bootable images VLP/G supports can be built on an “offline” environment.
The word “offline” means an isolated environment which does not connect to any network. Since VLP/G includes all necessary source codes of OSS except for the Linux kernel, VLP/G can always build images in this “offline” environment without affected from changes of repositories of OSS. Also, this “offline” environment reproduces the same images as the images which were verified by Renesas.

Below images can be built “offline”.

- core-image-minimal
- core-image-bsp
- core-image-weston (including the SDK build)
- core-image-qt (including the SDK build)

2. Build Instructions

2.1 Building images

This section describes the instructions to build the Board Support Package.
Before starting the build, run the command below on the Linux Host PC to install 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 python python3 python3-pip python3-pexpect \
xz-utils debianutils iputils-ping libsdl1.2-dev xterm p7zip-full libyaml-dev \
libssl-dev bmap-tools
```

Please refer to the URL below for detailed information:

- https://docs.yoctoproject.org/3.1.26/brief-yoctoprojectqs/brief-yoctoprojectqs.html

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

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

Copy all files obtained from Renesas into your Linux Host PC prior to the steps below. The directory which you put the files in is described as <package download directory> in the build instructions.

(1) Create a working directory at your home directory, and decompress Yocto recipe package

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

```
$ mkdir ~/rzg_vlp_<package version>
$ cd ~/rzg_vlp_<package version>
$ cp ../<package download directory>/*.zip .
$ unzip ./RTK0EF0045Z0021AZJ-<package version>.zip
$ tar zxvf ./RTK0EF0045Z0021AZJ-<package version>/rzg_vlp_<package version>\
.tar.gz
```

Note) Please note that your build environment must have 100GB of free hard drive space in order 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.

Note) <package version>: e.g v3.0.5.

(2) Enable Graphics and Video Codec

If you want to enable the Graphics and the video codec on RZ/G2H, RZ/G2M, RZ/G2N, and RZ/G2E when building **core-image-weston**, please copy the Multimedia package (RTK0EF0045Z0022AZJ-<version>_EN.zip or RTK0EF0045Z0022AZJ-<version>_JP.zip) to working directory and run the commands below. If you build core-image-minimal, please ignore this step.

```
$ unzip ./RTK0EF0045Z0022AZJ-<version>_EN.zip
$ tar zxvf ./RTK0EF0045Z0022AZJ-<version>_EN/meta-rz-features.tar.gz
```

VLP/G is set video output to LVDS as default setting. In case to use HDMI as a video output, please apply the patch with these commands. This step is only for **RZ/G2H, RZ/G2M, RZ/G2N, and RZ/G2E**.

```
$ cd ./meta-renesas
$ patch -p1 < ../extra/0001-Add-HDMI-support-for-RZ-G2.patch
$ cd ..
```

After applying above patch, please note the limitations for Audio, Bluetooth and SATA in section 1.2

(3) Build Initialize

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

```
$ TEMPLATECONF=$PWD/meta-renesas/meta-rzg2h/docs/template/conf/ source \
poky/oe-init-build-env build
```

(4) Add layers

Please follow the below steps to add the layers you need. The steps add the settings to bblayers.conf.

- **Graphics**: Please run the below command if you need the Graphics library.

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

- **Video Codec**: Please run the below command if you need the video codec library.

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

- **Qt**: Please run the below commands if you want to include Qt.

```
$ bitbake-layers add-layer ../meta-qt5
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-graphics
$ bitbake-layers add-layer ../meta-rz-features/meta-rz-codecs
```

- **Docker**: Please run the below commands if you want to include Docker. This means running Docker on the RZ board, not as using Docker as part of your build environment.

```
$ bitbake-layers add-layer ../meta-openembedded/meta-filesystems
$ bitbake-layers add-layer ../meta-openembedded/meta-networking
$ bitbake-layers add-layer ../meta-virtualization
```

(5) Decompress OSS files to "build" directory (Optional)

Run the commands below. This step is not mandatory and able to go to the step (6) in case the "offline" environment is not required. All OSS packages will be decompressed with this '7z' command.

```
$ cp ../../<package download directory>/*.7z .
$ 7z x oss_pkg_rzg_<package version>.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.

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

```
BB_NO_NETWORK = "1"
```

To change BB_NO_NETWORK from "0" to "1".

(6) Start a build

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

Build the target file system image using bitbake

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

<board> can be selected by referring to the Table 2.

Table 2. List of the platforms and the boards

Renesas MPU	Board
RZ/G2H	hihope-rzg2h
RZ/G2M	hihope-rzg2m
RZ/G2N	hihope-rzg2n
RZ/G2E	ek874

<target> can be selected in below. Please refer to the Table 3 for supported image details.

- core-image-minimal
- core-image-bsp
- core-image-weston
- core-image-qt

Table 3. Supported images of VLP/G

Image name	Target devices	Purpose
core-image-minimal	RZ/G2H, M, N, E	Minimal set of components
core-image-bsp	RZ/G2H, M, N, E	Minimal set of components plus audio support and some useful tools
core-image-weston	RZ/G2H, M, N, E	Standard image with graphics support
core-image-qt	RZ/G2H, M, N, E	Enable Qt LGPL version

"RZ/G2H, M, N, E" means the devices of RZ/G2H, RZ/G2M, RZ/G2N, and RZ/G2E.

After the build is successfully completed, a similar output will be seen, and the command prompt will return.

```
NOTE: Tasks Summary: Attempted 7427 tasks of which 16 didn't need to be rerun and all succeeded.
```

All necessary files listed in the **Table 4** will be generated by the bitbake command and will be located in the **`build/tmp/deploy/images`** directory.

Table 4. Image files for RZ/G2H, RZ/G2M, RZ/G2N, and RZ/G2E

RZ/G2H	**Linux kernel**	Image-hihope-rzg2h.bin
	root filesystem	*<image name>*-hihope-rzg2h.tar.bz2
	Boot loader	u-boot-elf-hihope-rzg2h.srec bootparam_sa0.srec bl2-hihope-rzg2h.srec bl31-hihope-rzg2h.srec tee-hihope-rzg2h.srec cert_header_sa6.srec
	SD Image	core-image-*<image name>* -hihope-rzg2h.wic.gz core-image-*<image name>* -hihope-rzg2h.wic.bmap
RZ/G2M **v3.0 (*)** **v1.3 (*)**	**Linux kernel**	Image-hihope-rzg2m.bin
	root filesystem	*<image name>*-hihope-rzg2m.tar.bz2
	Boot loader	u-boot-elf-hihope-rzg2m.srec bootparam_sa0.srec bl2-hihope-rzg2m.srec bl31-hihope-rzg2m.srec tee-hihope-rzg2m.srec cert_header_sa6.srec
	SD Image	core-image-*<image name>* -hihope-rzg2m.wic.gz core-image-*<image name>* -hihope-rzg2m.wic.bmap
RZ/G2N	**Linux kernel**	Image-hihope-rzg2n.bin
	root filesystem	*<image name>*-hihope-rzg2n.tar.bz2
	Boot loader	u-boot-elf-hihope-rzg2n.srec bootparam_sa0.srec bl2-hihope-rzg2n.srec bl31-hihope-rzg2n.srec tee-hihope-rzg2n.srec cert_header_sa6.srec
	SD Image	core-image-*<image name>* -hihope-rzg2n.wic.gz core-image-*<image name>* -hihope-rzg2n.wic.bmap
RZ/G2E	**Linux kernel**	Image-ek874.bin
	root filesystem	*<image name>*-ek874.tar.bz2
	Boot loader	u-boot-elf-ek874.srec bootparam_sa0.srec bl2-ek874.srec bl31-ek874.srec tee-ek874.srec cert_header_sa6.srec
	SD Image	core-image-*<image name>* -ek874.wic.gz core-image-*<image name>* -ek874.wic.bmap

<image name> will be the name used in the step (6).

Device tree files

	Type 1	Type 2	Type 3	Type 4
RZ/G2H	Image-r8a774e1-hihope-rzg2h.dtb	Image-r8a774e1-hihope-rzg2h-ex.dtb	Image-r8a774e1-hihope-rzg2h-ex-idk-1110wr.dtb	Image-r8a774e1-hihope-rzg2h-ex-mipi-2.1.dtb
RZ/G2M v1.3 (*1)	Image-r8a774a1-hihope-rzg2m.dtb	Image-r8a774a1-hihope-rzg2m-ex.dtb	Image-r8a774a1-hihope-rzg2m-ex-idk-1110wr.dtb	Image-r8a774a1-hihope-rzg2m-ex-mipi-2.1.dtb
RZ/G2M v3.0 (*1)	Image-r8a774a3-hihope-rzg2m.dtb	Image-r8a774a3-hihope-rzg2m-ex.dtb	Image-r8a774a3-hihope-rzg2m-ex-idk-1110wr.dtb	Image-r8a774a3-hihope-rzg2m-ex-mipi-2.1.dtb
RZ/G2N	Image-r8a774b1-hihope-rzg2n.dtb	Image-r8a774b1-hihope-rzg2n-ex.dtb	Image-r8a774b1-hihope-rzg2n-ex-idk-1110wr.dtb	Image-r8a774b1-hihope-rzg2n-ex-mipi-2.1.dtb
RZ/G2E EK874 Rev E (*2)	Image-r8a774c0-cat874.dtb	Image-r8a774c0-ek874.dtb	Image-r8a774c0-ek874-idk-2121wr.dtb	Image-r8a774c0-ek874-mipi-2.1.dtb

There are 4 types of the device tree files. Available devices are different depending on them. Please refer to the following description:

- **Type1**: Main board only
- **Type2**: Main board + Sub board
- **Type3**: Main board + Sub board + LVDS panel
- **Type4**: Main board + Sub board + MIPI/CSI2 cameras

Please note that users who use the combination of main and sub boards need to use type2-4 as a device tree file. **If the dtb files of type1 are used, interfaces on the sub board such as Ethernet are not able to be used.**

(*1) There are 2 types of RZ/G2M LSI (“RZ/G2M v3.0” and “RZ/G2M v1.3”). In case you use the hihope-rzg2m board which has one of them, the same image files can be used, but the same device tree files cannot be used. Please refer to the above table. If the board prints the messages below when turn on the power, RZ/G2M v3.0 is used on your board. In case of RZ/G2M v1.3, “R8A774A1” will be displayed.

```
CPU: Renesas Electronics R8A774A3
```

(*2) In case you use the RZ/G2E EK874 revision C board, please see the section 2.2 (1).

2.2 Notes

(1) Device tree for RZ/G2M, RZ/G2N and RZ/G2E

The dtb files listed in the **Table 4** cannot be used for the early revision of Hoperun and Silicon Linux boards. If you are using revision 2 of Hoperun and revision C of Silicon Linux boards, please use below files. These are automatically generated at the same place as the other image files when building a BSP.

HiHope RZ/G2M board:

- Image-r8a774a1-hihope-rzg2m-**rev2**.dtb (main board only)
- Image-r8a774a1-hihope-rzg2m-**rev2**-ex.dtb (main + sub board)
- Image- r8a774a1-hihope-rzg2m-**rev2**-ex-idk-1110wr.dtb (main + sub board + LVDS-IF)
- Image-r8a774a1-hihope-rzg2m-**rev2**-ex-mipi-2.1.dtb (main + sub board + MIPI/CSI2 cameras)

HiHope RZ/G2N board:

- Image-r8a774b1-hihope-rzg2n-**rev2**.dtb (main board only)
- Image-r8a774b1-hihope-rzg2n-**rev2**-ex.dtb (main + sub board)
- Image- r8a774b1-hihope-rzg2n-**rev2**-ex-idk-1110wr.dtb (main + sub board + LVDS-IF)
- Image-r8a774b1-hihope-rzg2n-**rev2**-ex-mipi-2.1.dtb (main + sub board + MIPI/CSI2 cameras)

Silicon Linux RZ/G2E board (EK874):

- Image-r8a774c0-cat874-**revc**.dtb (main board only)
- Image-r8a774c0-ek874-**revc**.dtb (main + sub board)
- Image-r8a774c0-ek874-**revc**-idk-2121wr.dtb (main + sub board + LVDS-IF)
- Image-r8a774c0-ek874-**revc**-mipi-2.1.dtb (main + sub board + MIPI/CSI2 cameras)

Note) Board revision is printed on boards. It is printed below the Hoperun logo on boards and directly printed on the Silicon Linux boards.

(2) Video output

VLP/G is set video output to LVDS as default setting. In case to use HDMI as a video output, please apply the patch with these commands. This step is only for RZ/G2H, RZ/G2M, RZ/G2N, and RZ/G2E.

```
$ cd ~/rzg_vlp_v3.0.5/meta-renesas
$ patch -p1 < ../extra/0001-Add-HDMI-support-for-RZ-G2.patch
```

After applying above patch, please note to below points.

Audio

In case 48kHz audio, please set SW2404 to P1 side in HiHope Rev4 Boards of RZ/G2H, M, and N.

Bluetooth

Firmware of Bluetooth is integrated into the kernel. Therefore, this step is not necessary.

```
Hciattach /dev/ttySC1 texas 3000000
```

Instead, this step is required.

```
Rfkill unblock bluetooth
```

SATA

SATA interface on HiHope Rev4 Boards of RZ/G2H, N is enabled by setting switches as below. (SATA cannot be supported in the RZ/G2N Rev2 board due to HW limitation).

- SW1001-7 on main board: OFF
- SW43 on sub board: ON

(3) Video playback

Due to the specification of open source software (Gstreamer and others) and drivers, multiple Gstreamer pipelines with hardware scale cannot run.

Also, below formats of video are not supported.

- NV61
- YUV420
- YUV422
- YUV444
- H.264, 80Mbps

(4) ECC

The ECC function for DRAM has two modes: 8bit data/5bit ECC mode and 64bit data/8bit ECC mode. 8bit data/5bit ECC mode can be evaluated by the following method. When applying the ECC function to products or need other details including the method to enable 64bit data/8bit ECC mode, please contact Renesas.

Enable the function by changing the lines below in the `local.conf`.

`Disable:`

```
# MACHINE_FEATURES_append = “ ecc”
```

`# ECC_MODE = “Partial”Enable:`

```
MACHINE_FEATURES_append = “ ecc”
ECC_MODE = “Full”
```

This sets 8bit data/5bit ECC mode for all DRAM regions. After building, please replace all images including boot loaders.

(5) SDHI

The early revision of EK874 boards cannot detect the insertion of an SD card. Please plugged in a card before turning on the power.

(6) VIN

One camera input is enabled in default settings. Two camera inputs can be enabled in RZ/G2H, N and M v3.0. Please refer to the Video Capture Driver User’s Manual that is included in the RZ/G2 Group BSP Manual Set for more details.

VIN on early revision of HiHope RZ/G2M and N boards fails to work. Please use newer boards in case VIN is necessary.

(7) Wifi

Wifi is disabled in default settings but modules necessary for Wifi functions are installed into rootfs. In case Wifi is necessary, please enable it from a console as below.

```
$ rfkill list
```

If this command shows "Soft blocked: yes", run "unblock" command like this.

```
$ rfkill unblock wlan
```

Then, continue below.

```
$ connmanctl
connmanctl> scan wifi
connmanctl> services
connmanctl> agent on
connmanctl> connect <network_name>
  <imput password>
connmanctl> quit
```

You may need to retry the "connect" command few times.

Note that some settings relating about radio waves should be adjusted according to the laws of each region. Please refer to general information in books and websites about Linux networking.

(8) GPLv3 packages

In this release, the GPLv3 packages are disabled as default in *build/conf/local.conf*:

```
INCOMPATIBLE_LICENSE = "GPLv3 GPLv3+"
```

If you want to use GPLv3, just hide this line:

```
#INCOMPATIBLE_LICENSE = "GPLv3 GPLv3+"
```

(9) Disable libraries of Graphics and Video Codec

When you want to disable the functions of the libraries of the graphics and the video codec, please add the following lines in build/conf/local.conf:

- Disable OpenGL ES library in the graphics package (*1)

```
DISTRO_FEATURES_remove = " opengles"
```

- Disable OpenCL library in the graphics package (*1)

```
DISTRO_FEATURES_remove = " opencl"
```

- Disable OpenMAX library for decode in the video codec package (*2)

```
DISTRO_FEATURES_remove = " hwh264dec hwh265dec"
```

- Disable OpenMAX library for encode in the video codec package (*2)

```
DISTRO_FEATURES_remove = " hwh264enc"
```

(*1) This library is included in RTK0EF0045Z13001ZJ-v1.0.5_EN.zip and RTK0EF0045Z13001ZJ-v1.0.5_JP.zip
(*2) This library is included in RTK0EF0045Z15001ZJ-v1.1.0_EN.zip and RTK0EF0045Z15001ZJ-v1.1.0_JP.zip

(10) Real time performance

If you want to use the kernel which improves the real-time performance, please add the line below to the file "~/rzg_vlp_v3.0.x/build/conf/local.conf".

```
IS_RT_BSP="1"
```

(11) USB Video Class

USB Video Class (UVC) driver is not installed with the default settings of VLP/G due to its large size.

In case UVC devices such as USB cameras are necessary, please install the driver by adding the line below to `local.conf`.

```
IMAGE_INSTALL_append = “ kernel-module-uvcvideo “
```

(12) CIP Core Packages

VLP/G includes Debian 10 (Buster) and Debian 11 (Bullseye) based CIP Core Packages and Buster is enabled by the default settings. These packages can be replaced with other versions of packages.

Note that network access is required to start the build process when you enable these packages except for Buster (or Bullseye) which is set as the default setting.

CIP Core Packages are going to be maintained by the Civil Infrastructure Platform project. For more technical information, please contact Renesas.

1. **Buster** (default)**:**

The following lines are added as default in the `local.conf`:

```
# Select CIP Core packages by switching between Buster and Bullseye.
# - Buster (default) : build all supported Debian 10 Buster recipes
# - Bullseye : build all supported Debian 11 Bullseye recipes
# - Not set (or different with above): not use CIP Core, use default packages version
in Yocto

CIP_MODE = “Buster”
```

2. **Bullseye:**

Please change “`CIP_MODE`” in the `local.conf` to change from Buster to Bullseye:

```
# Select CIP Core packages by switching between Buster and Bullseye.
# - Buster (default) : build all supported Debian 10 Buster recipes
# - Bullseye : build all supported Debian 11 Bullseye recipes
# - Not set (or different with above): not use CIP Core, use default packages version
in Yocto

CIP_MODE = “Bullseye”
```

3. **No CIP Core Packages:**

If the CIP Core Packages are unnecessary, comment out and add the following lines to disable CIP CORE Packages in `local.conf`:

```
# Select CIP Core packages by switching between Buster and Bullseye.
# - Buster (default) : build all supported Debian 10 Buster recipes
# - Bullseye : build all supported Debian 11 Bullseye recipes
# - Not set (or different with above): not use CIP Core, use default packages version
in Yocto

#CIP_MODE = “Buster”
```

Note) The above 3 settings disable GPLv3 packages as default. In case the GPLv3 packages are required, please comment out the following line in the `local.conf`.

```
# INCOMPATIBLE_LICENSE = “GPLv3 GPLv3+”
```

By building the BSP, the packages will be replaced as below in the Table 5.

Table 5. Versions of all Buster and Bullseye Debian packages

Package	Buster Debian	Bullseye Debian
attr	2.4.48	2.4.48
busybox	1.30.1	1.30.1
coreutils	6.9	-
gcc	8.3.0	-
glib-2.0	2.58.3	-
glibc	2.28	2.31
gnupg	1.4.7	-
kbd	2.0.4	-
libassuan0	2.5.2	2.5.3
libgcrypt	1.8.4	-
libunistring	0.9.10	0.9.10
libnss	0.14.1	-
openssh	7.9p1	-
perl	5.30.1	-
pkgconfig	0.29	-
quilt	0.65	-

Note)
(-) These packages are not supported with Bullseye Debian version, so they used No CIP CORE version.

(13) WIC image

The name “WIC” is derived from OpenEmbedded Image Creator (oeic). It includes image that system can boot it in hardware device.
From VLP/G v3.0.5, WIC is supported and below guidelines are shown how to use it to boot Renesas RZ/G devices.

- Enable building WIC image in local.conf (default is enabled) by setting “**WKS_SUPPORT**” to 1:

```
WKS_SUPPORT ?= “1”
```

- Defines additional free disk space created in the image in Kbytes (keep default value if unsure):

```
IMAGE_ROOTFS_EXTRA_SPACE = "1048576"
```

- Select wks file to be built by setting “**WKS_DEFAULT_FILE**” (keep default value if unsure). Currently, there are 2 types of wks defined in “meta-renesas/meta-rz-common/wic” for uSD/eMMC (channel 0) and uSD (channel 1).

- Building your desired core-image by running “bitbake core-image-x”. “core-image-x” should be one of following options:
  - core-image-minimal
  - core-image-bsp
  - core-image-weston
  - core-image-qt

- There are 2 files *wic.bmap and *wic.gz in deploy folder after building successfully. Example:
  - core-image-minimal-hihope-rzg2h.wic.bmap
  - core-image-minimal-hihope-rzg2h.wic.gz

(14) Software bill of materials (SBoM)

Software package data exchange (SPDX) is an open standard for SBoM that identifies and catalogs components, licenses, copyrights, security references, and other metadata relating to software.

From VLP/G v3.0.5, creating SPDX is supported and below guidelines are shown how to use it:

- Enable creating SPDX in local.conf (default is disabled) by uncommenting out below line:

```
#INHERIT += "create-spdx"
```

- Select below optional features to be supported for SDPX by enable in local.conf (all is disabled by default):
  - **SPDX_PRETTY**: Make generated files more human readable (newlines, indentation)

```
SPDX_PRETTY = "1"
```

  - **SPDX_ARCHIVE_PACKAGED**: Add compressed archives of the files in the generated target packages in tar.gz files.

```
SPDX_ARCHIVE_PACKAGED = "1"
```

- SPDX is created and deployed in “tmp/deploy/spdx/$MACHINE”. All information can be checked here.s

Note: There is an issue when building SDK (example bitbake core-image-weston -c populate_sdk) with SBoM SPDX support:

```
| ERROR: core-image-weston-1.0-r0 do_populate_sdk: Error executing a python function
in exec_func_python() autogenerated:
| *** 1078:        return self._accessor.open(self, flags, mode)
|     1079:
|     1080:    def _raw_open(self, flags, mode=0o777):
|     1081:        """
|     1082:        Open the file pointed by this path and return a file descriptor,
|Exception: FileNotFoundError:
|[Errno 2] No such file or directory: 'tmp/work/smarc_rzg2l-poky-linux/core-image-
weston/1.0-r0/spdx/sdk-work/poky-glibc-x86_64-core-image-weston-aarch64-smarc-rzg2l-
target.spdx.json'
```

To fix this, please apply below change in "**poky/meta/classes/populate_sdk_base.bbclass**":

```
-do_populate_sdk[cleandirs] = "${SDKDEPLOYDIR}"
+do_populate_sdk[cleandirs] += "${SDKDEPLOYDIR}"
```

3. Preparing the SD Card

You can prepare the microSD card by the following 2 methods. Please select one of them and follow the steps.

- 3.1 Write files to the microSD card (used wic image)
- 3.2 Write files to the microSD card (not used wic image)

3.1 Write files to the microSD card (used wic image)

Set micro SD card to Linux PC. And check the mount device name with fdisk command.

```
$ sudo fdisk -l
Disk /dev/sdb: 3.74 GiB, 3997171712 bytes, 7806976 sectors
Disk model: Storage Device
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: 0xxxxxxxxxx
$ umount /dev/sdb1
$ umount /dev/sdb2
```

Expand disk image.

```
$ sudo bmaptool copy <wic image>.wic.gz /dev/sdb
```

The file names of *<wic image>* is listed in the Table 4.

When you complete the above commands, the micro SD card is prepared correctly. Please skip the section 3.2.

Table 6. File and directory in the micro SD card

Type/Number	Filesystem	Contents
Primary #1	FAT32	AArch64_Flash_writer_SCIF_DUMMY_CERT_E6300400_<board>.mot u-boot-elf-<board>.srec bootparam_sa0.srec bl2-<board>.srec bl31-<board>.srec tee-<board>.srec cert_header_sa6.srec
Primary #2	Ext4	./ ├── bin ├── boot │ ├── Image │ └── <device>-<board>.dtb ├── dev ├── etc ├── home ├── lib ├── media ├── mnt ├── proc ├── run ├── sbin ├── sys ├── tmp ├── usr └── var

Note *1) Please refer to 2.2(13) WIC image for partition size specifications.

3.2 Write files to the microSD card (not used wic image)

To boot from SD card, over 4GB capacity of blank SD card is needed. You can use Linux Host PC to expand the kernel and the rootfs using USB card reader or other equipment.

Please format the card according to the following steps before using the card:

(1) Non-connect microSD card to Linux Host PC

```
$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0  30.9G  0 disk
├─sda1   8:1    0   512M  0 part /boot/efi
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0  30.3G  0 part /
sr0     11:0    1  1024M  0 rom
```

(2) Connect microSD card to Linux Host PC with USB adapter

(3) Check the device name which is associated to the microSD card.

```
$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0  30.9G  0 disk
├─sda1   8:1    0   512M  0 part /boot/efi
├─sda2   8:2    0     1K  0 part
└─sda5   8:5    0  30.3G  0 part /
sdb      8:16   1  29.7G  0 disk
└─sdb1   8:17   1  29.7G  0 part
sr0     11:0    1  1024M  0 rom
```

The message above shows the card associated with the `/dev/sdb`. Be careful not to use the other device names in the following steps.

(4) Unmount automatically mounted microSD card partitions

If necessary, unmount all mounted microSD card partitions.

```
$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev              745652        0    745652   0% /dev
   :
   snip
   :
/dev/sdb1         511720     4904    506816   1% /media/user/A8D3-393B
$ sudo umount /media/user/A8D3-393B
```

If more than one partition has already been created on micro-SD card, unmount all partitions.

(5) Change the partition table

MicroSD card needs two partitions as listed in Table 7.

Table 7. Partitions of microSD card

Type/Number	Size	Filesystem	Contents
Primary #1	500MB (minimum 128MB)	FAT32	Linux kernel Device tree
Primary #2	All remaining	Ext4	root filesystem

Set the partition table using the fdisk command like this.

```
$ 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 0x6b6aac6e.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-62333951, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-62333951, default 6233395
1): +500M

Created a new partition 1 of type 'Linux' and of size 500 MiB.
Partition #1 contains a vfat signature.

Do you want to remove the signature? [Y]es/[N]o: Y

The signature will be removed by a write command.

Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): (Push the enter key)
First sector (1026048-62333951, default 1026048): (Push the enter key)
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1026048-62333951, default 62333
951): (Push the enter key)

Created a new partition 2 of type 'Linux' and of size 29.2 GiB.

Command (m for help): p
Disk /dev/sdb: 29.74 GiB, 31914983424 bytes, 62333952 sectors
Disk model: Transcend
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: 0x6b6aac6e

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1          2048  1026047  1024000  500M 83 Linux
/dev/sdb2       1026048 62333951 61307904 29.2G 83 Linux

Filesystem/RAID signature on partition 1 will be wiped.

Command (m for help): t
Partition number (1,2, default 2): 1
Hex code (type L to list all codes): b

Changed type of partition 'Linux' to 'W95 FAT32'.

Command (m for help): w
The partition table has been altered.
Syncing disks.
```

Then, check the partition table with the commands below:

```
$ partprobe
$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 29.74 GiB, 31914983424 bytes, 62333952 sectors
Disk model: Maker name etc.
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: 0x6b6aac6e

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sdb1          2048  1026047  1024000  500M  b W95 FAT32
/dev/sdb2       1026048 62333951 61307904 29.2G 83 Linux
```

(6) Format and mount the partitions

If the partitions were automatically mounted after the step 4, please unmount them according to the step 3.

Then format the partitions using the command below:

```
$ 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 1024000 sectors;
drive number 0x80;
filesystem has 2 32-bit FATs and 8 sectors per cluster.
FAT size is 1000 sectors, and provides 127746 clusters.
There are 32 reserved sectors.
Volume ID is a299e6a6, no volume label.
Searching for bad blocks 16848... 34256... 51152... 68304... 85072... 10209
6... 119376... 136528... 153552... 170576... 187472... 204624... 221648... 238
928... 256208... 273744... 290768... 308048... 325328... 342480... 359504... 3
76656... 393680... 410576... 427216... 444624... 462032... 479184... 495952...

$ sudo mkfs.ext4 -L rootfs /dev/sdb2
```

```
mke2fs 1.45.5 (07-Jan-2020)
Creating filesystem with 7663488 4k blocks and 1916928 inodes
Filesystem UUID: 63dddb3f-e268-4554-af51-1c6e1928d76c
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
  4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

(7) Remount microSD card

After format, remove the card reader and connect it again to mount the partitions.

(8) Write files to the microSD card

Check the mount point name with df command.

```
$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev              745652        0    745652   0% /dev
   :
   snip
   :
/dev/sdb1         510984       16    510968   1% /media/user/A299-E6A6
/dev/sdb2       30041556    45080  28447396   1% /media/user/rootfs
```

Copy kernel and device tree file to the first partition.

```
$ cp $WORK/build/tmp/deploy/images/<board>/<Linux kernel> /media/user/A299-E6A
6
$ cp $WORK/build/tmp/deploy/images/<board>/<Devise tree> /media/user/A299-E6A6
```

Expand rootfs to the second partition.

```
$ cd /media/user/rootfs
$ sudo tar jxvf $WORK/build/tmp/deploy/images/<board>/<root filesystem>
```

Please replace *<board>* by the name refer to **Table** 2.

Please replace *<Linux kernel>* , *<root filesystem>* and *<Device tree>* by the name refer to **Table** 4

Table 8. File and directory in the microSD card

Type/Number	Size	Filesystem	Contents
Primary #1	Size specified when the partition was created. (minimum 128MB)	FAT32	Image-<board>.bin Image-<device>-<board type>.dtb
Primary #2	Size specified when the partition was created.	Ext4	./ ├─── bin ├─── boot ├─── dev ├─── etc ├─── home ├─── lib ├─── media ├─── mnt ├─── proc ├─── run ├─── sbin ├─── sys ├─── tmp ├─── usr └─── var

4. Reference Board Setting

4.1 Hoperun Technology HiHope RZ/G2[HMN] platform (hihope-rzg2h, hihope-rzg2m, hihope-rzg2n)

4.1.1 Preparation of Hardware and Software

The following environment of Hardware and Software is used in the evaluation.

Hardware preparation(Users should purchase the following equipment.):

- AC Adapter 12V2A with EIAJ3 connector(Any AC Adapter)
- USB Type-microB cable (Any cables)
- HDMI cable (Any cables)

PC Installed Silicon Labs VCP driver and Terminal software (Tera Term)

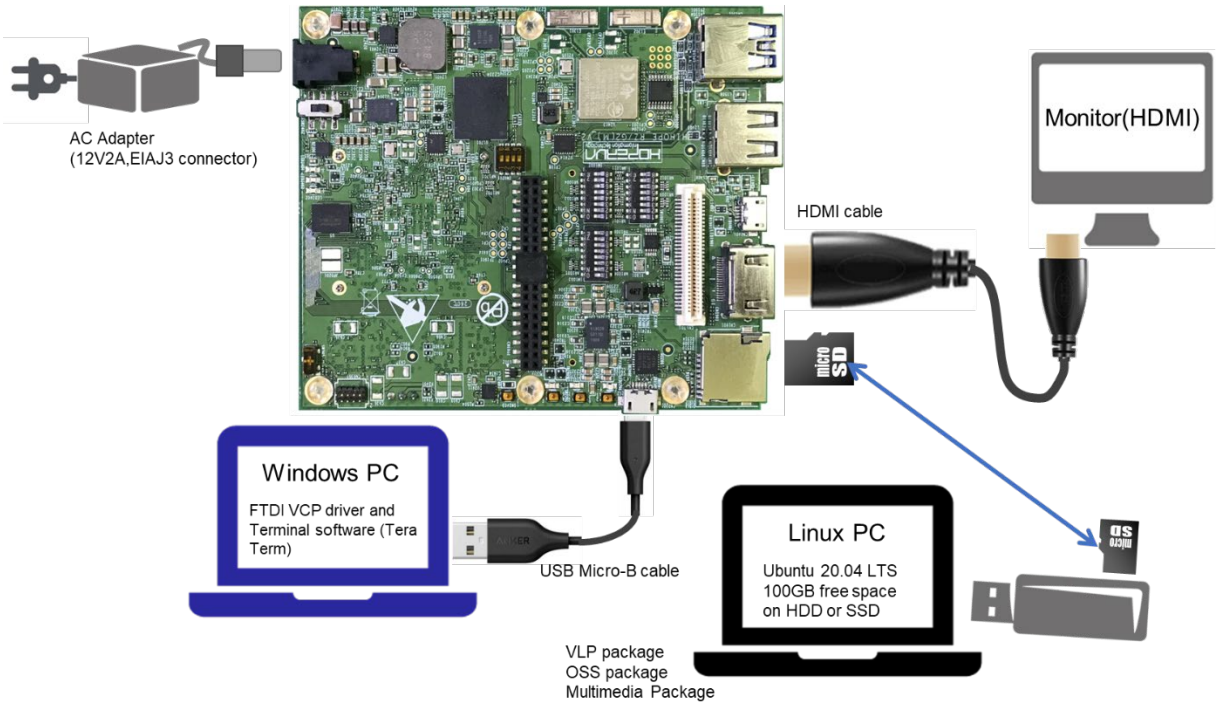


Figure 2. Operating environment

4.1.2 Building files to write

This board uses the files below as a bootloader. Please build them according to the Release Note and copy these files to the PC which runs a serial terminal software.

- bootparam_sa0.srec
- bl2-hihope-rzg2h.srec (RZ/G2H) or bl2-hihope-rzg2m.srec (RZ/G2M) or bl2-hihope-rzg2n.srec (RZ/G2N)
- cert_header_sa6.srec
- bl31-hihope-rzg2h.srec (RZ/G2H) or bl31-hihope-rzg2m.srec (RZ/G2M) or bl31-hihope-rzg2n.srec (RZ/G2N)
- u-boot-elf-hihope-rzg2h.srec (RZ/G2H) or u-boot-elf-hihope-rzg2m.srec (RZ/G2M) or u-boot-elf-hihope-rzg2n.srec (RZ/G2N)

4.1.3 Settings

Connect between the board and a control PC by USB serial cable according to the Release Note.

Set the settings about serial communication protocol on a terminal software as below:

- Speed: 115200 bps
- Data: 8bit
- Parity: None
- Stop bit: 1bit
- Flow control: None

To set the board to SCIF Download mode, set the SW1002 as below:

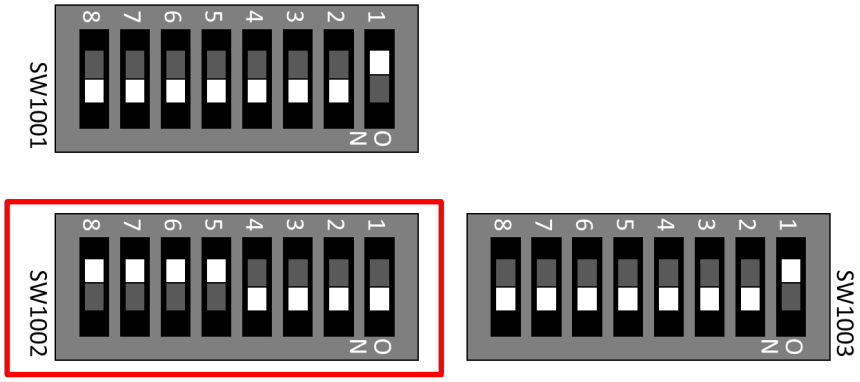


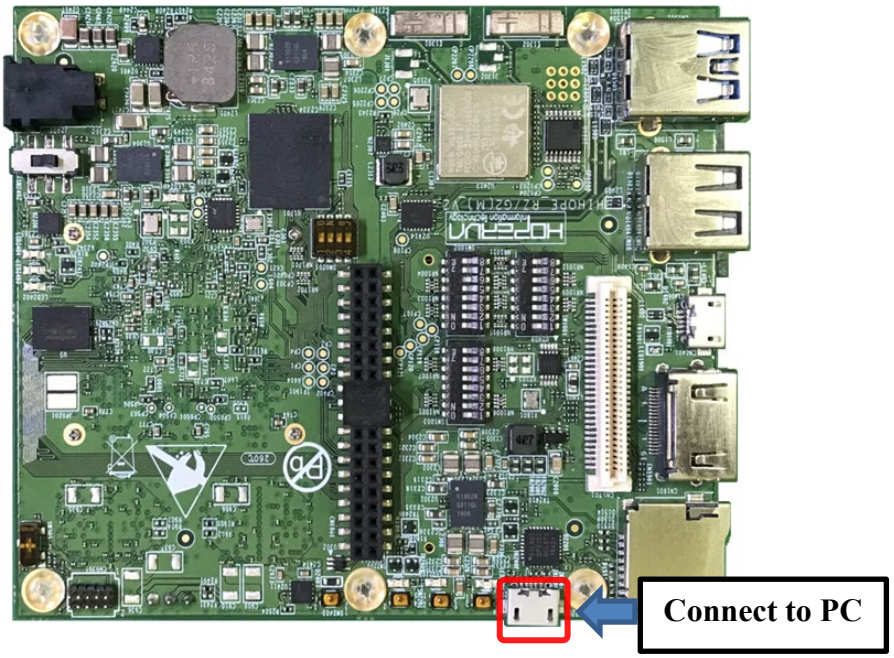
Table 9. SW1002

8	7	6	5	4	3	2	1
OFF	OFF	OFF	OFF	ON	ON	ON	ON

Note) Be careful not to change the SW1001 and SW1003

4.1.4 How to use debug serial (console ouput)

Please connect USB Type-microAB cable to CN201.



CN201:USB Type-microAB Connector

Figure 3. Connecting console for debug

4.1.5 Power supply

1. Connect AC adapter to Connector (J2401).

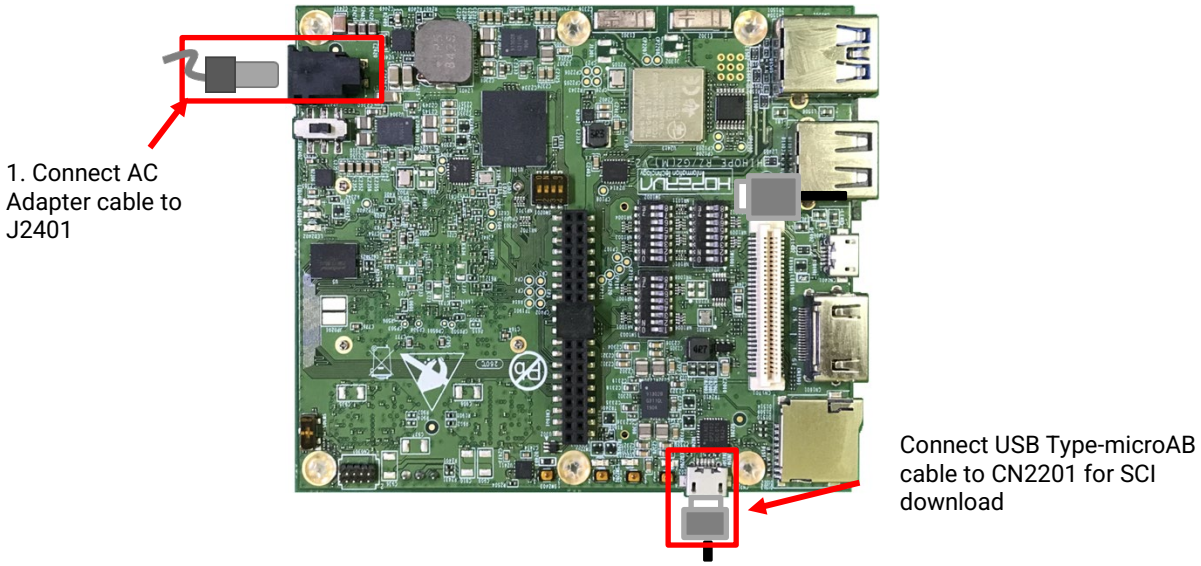


Figure 4. Connecting Power Supply

2. Slide the power switch (SW2402) to turn on the power.
3. LED2401 lighs up for a moment.
4. LED2201, LED2202, LED2203 lights up.

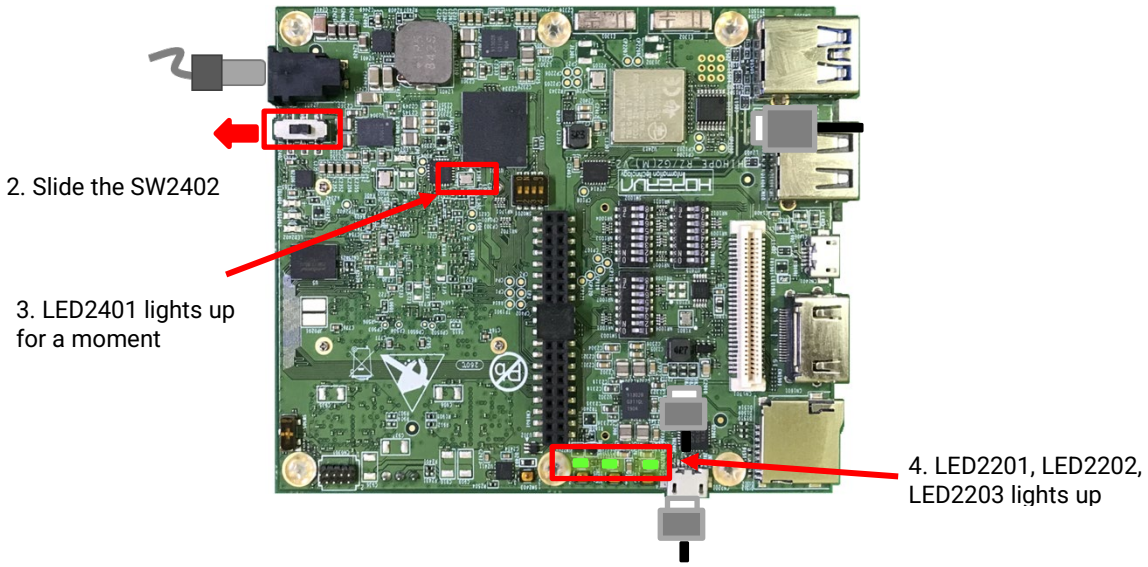


Figure 5. Power ON

4.1.6 Building files to write

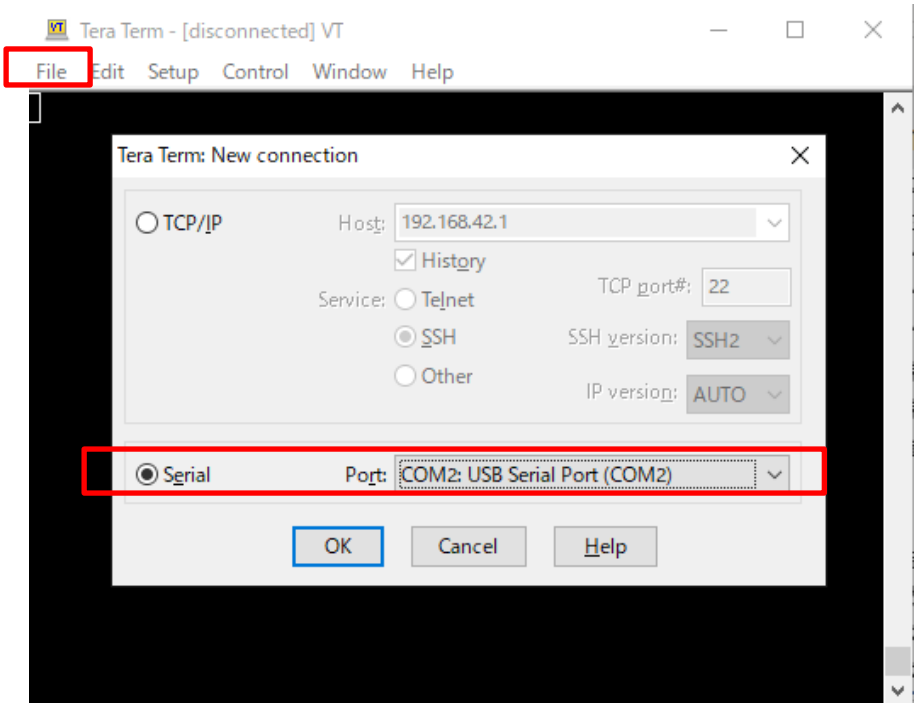
The evaluation boards use the files in the Table 10 as the boot loaders. The boot loaders files are generated by the build instructions in the section 2. Please copy these files to a PC which runs serial terminal software.

Table 10. File names of Boot loader

Board	File name of Boot loader
HiHope RZ/G2H platform	• u-boot-elf-hihope-rzg2h.srec • bootparam_sa0.srec • bl2-hihope-rzg2h.srec • bl31-hihope-rzg2h.srec • tee-hihope-rzg2h.srec • cert_header_sa6.srec
HiHope RZ/G2M platform	• u-boot-elf-hihope-rzg2m.srec • bootparam_sa0.srec • bl2-hihope-rzg2m.srec • bl31-hihope-rzg2m.srec • tee-hihope-rzg2m.srec • cert_header_sa6.srec
HiHope RZ/G2N platform	• u-boot-elf-hihope-rzg2n.srec • bootparam_sa0.srec • bl2-hihope-rzg2n.srec • bl31-hihope-rzg2n.srec • tee-hihope-rzg2n.srec • cert_header_sa6.srec

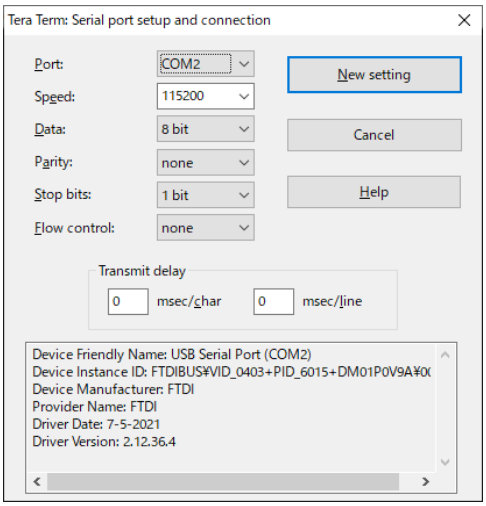
4.1.7 Settings

1. Bring up the terminal software and select the “File” > “New Connection” to set the connection on the software.



2. Select the “Setup” > “Serial port” to set the settings about serial communication protocol on the software. Set the settings about serial communication protocol on a terminal software as below:

- Speed: 115200 bps
- Data: 8bit
- Parity: None
- Stop bit: 1bit
- Flow control: None



3. To set the board to SCIF Download mode, set the SW11 as below:

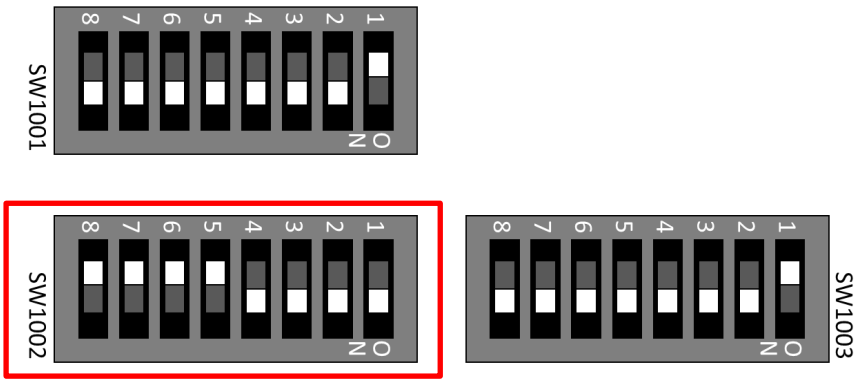


Table 11. SW1002

8	7	6	5	4	3	2	1
OFF	OFF	OFF	OFF	ON	ON	ON	ON

4. Turn on the power of the board by changing the SW2402. Messages below are shown on the terminal.

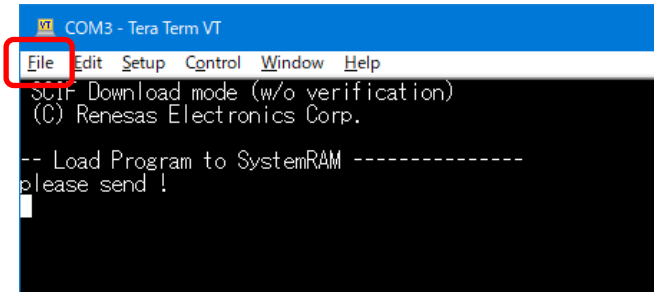
```
 SCIF Download mode (w/o verification)
 (C) Renesas Electronics Corp.

-- Load Program to SystemRAM ---------------
please send !
```

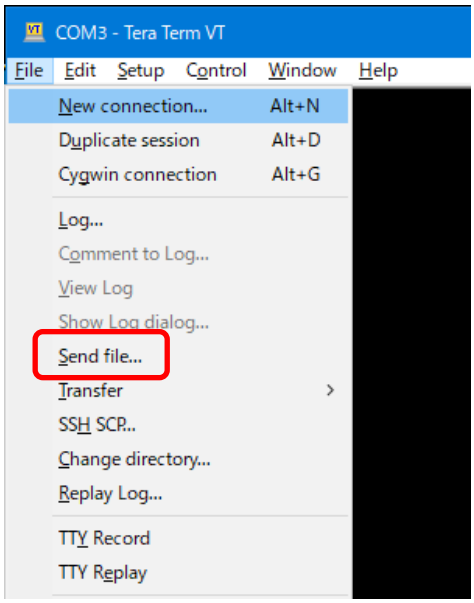
4.1.8 Download Flash Writer to RAM

Send an image of Flash Writer (AArch64_Flash_writer_SCIF_DUMMY_CERT_E6300400_hihope.mot) using terminal software after the message “please send !” is shown.

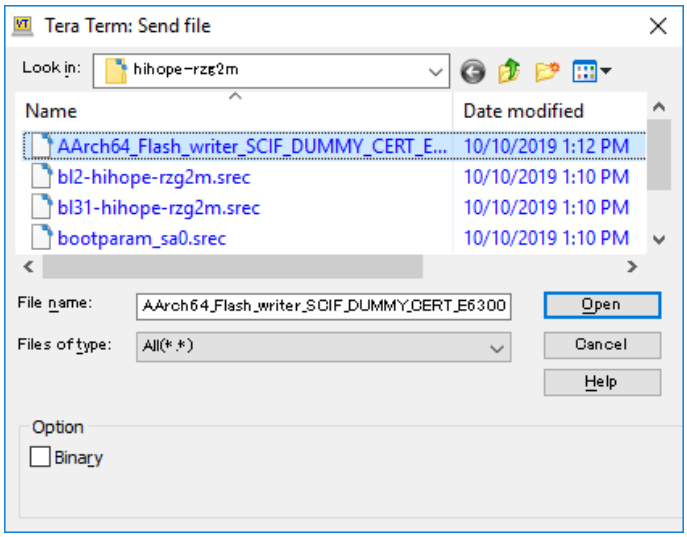
Below is a sample procedure with Tera Term.

Open a “Send file” dialog by selecting “File” → “Sendfile” menu.

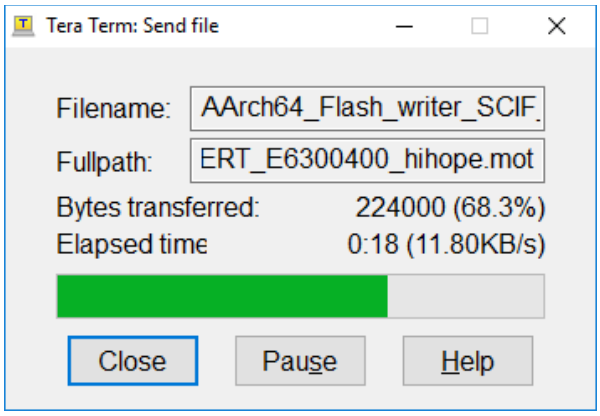




Then, select the image to be send and click “Open” button.



The image will be sent to the board via serial connection.



After successfully download the binary, Flash Writer starts automatically and show a message like below on the terminal.

```
Flash writer for RZ/G2M V1.00 Sep.24,2019
>
```

4.1.9 Writing Bootloader

"xls2" command of Flash Writer is used to write binary files. This command receives binary data from the serial port and write the data to specified address of the Flash ROM with information where the data should be loaded on the address of the main memory.

This is an example of writing "bootparam_sa0.srec" which should be placed to E6320000h of the main memory to 000000h of the Flash ROM.

```
>xls2
===== Qspi writing of RZ/G2 Board Command =============
Load Program to Spiflash
Writes to any of SPI address.
 Winbond : W25M512JW
Program Top Address & Qspi Save Address
===== Please Input Program Top Address ============
  Please Input : H'E6320000

===== Please Input Qspi Save Address ===
  Please Input : H'000000
Work RAM(H'50000000-H'53FFFFFF) Clear....
please send ! ('.' & CR stop load)
```

Send the data of "bootparam_sa0.srec" from terminal software after the message "please send !" is shown.

After successfully download the binary, messages like below are shown on the terminal.

```
SPI Data Clear(H'FF) Check :H'00000000-H'00007FFF Erasing.
........Erase Completed
SAVE SPI-FLASH.......
======= Qspi  Save Information  =================
 SpiFlashMemory Stat Address : H'00000000
 SpiFlashMemory End Address  : H'00000E67
============================================================
```

```
SPI Data Clear(H'FF) Check : H'00000000-0000FFFF,Clear OK?(y/n)
```

In case a message to prompt to clear data like above, please enter "y".

Write all necessary files using the addresses listed at the Table 12 and turn off the power of the board by changing the SW2402.

Table 12. Addresses for each file

File name	Address to load to RAM	Address to save to ROM
bootparam_sa0.srec	E6320000	000000
bl2-hihope-rzg2h.srec or bl2-hihope-rzg2m.srec or bl2-hihope-rzg2n.srec	E6304000	040000
cert_header_sa6.srec	E6320000	180000
bl31-hihope-rzg2h.srec or bl31-hihope-rzg2m.srec or bl31-hihope-rzg2n.srec	44000000	1C0000
tee-hihope-rzg2h.srec or tee-hihope-rzg2m.srec or tee-hihope-rzg2n.srec	44100000	200000
u-boot-elf-hihope-rzg2h.srec or u-boot-elf-hihope-rzg2m.srec or u-boot-elf-hihope-rzg2n.srec	50000000	300000

4.1.10 Change Back to Normal Boot Mode

To set the board to SPI Boot mode, set the SW1002 as below:

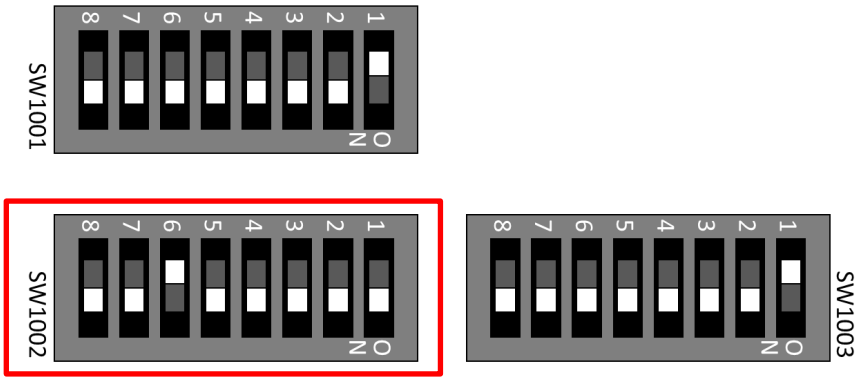


Table 13. SW1002

8	7	6	5	4	3	2	1
ON	ON	OFF	ON	ON	ON	ON	ON

Note) Be careful not to change the SW1001 and SW1003

Turn on the power of the board by changing the SW2402.

```
U-Boot 2021.10 (Sep 15 2023 - 05:00:35 +0000)

CPU:   Renesas Electronics R8A774B1 rev 1.1
Model: HopeRun HiHope RZ/G2N with sub board
DRAM:  3.9 GiB
WDT:   watchdog@00000000e6020000
WDT:   Started with servicing (60s timeout)
MMC:   mmc@ee100000: 0, mmc@ee160000: 1
```

```
Loading Environment from MMC... OK
In:    serial@e6e88000
Out:   serial@e6e88000
Err:   serial@e6e88000
U-boot WDT started!
Net:
Error: ethernet@e6800000 address not set.
No ethernet found.

Hit any key to stop autoboot:  0
=>
```

Following the messages above, many warning messages will be shown. These warnings are eliminated by setting correct environment variables. Please set default value and save them to the Flash ROM.

```
=> env default -a
## Resetting to default environment
=> saveenv
Saving Environment to SPI Flash... SF: Detected w25m512jv with page size 256 Bytes, er
ase size 4 KiB, total 32 MiB
Erasing SPI flash...Writing to SPI flash...done
OK
```

If you created the microSD card according to the **step 3.2**, perform the following settings. If you created the microSD card according to the **step 3.1**, this setting is not required.

Set environment variables using the commands below:

```
=> setenv bootargs 'root=/dev/mmcblk1p2 rootwait'
=> setenv bootcmd 'fatload mmc 0:1 0x48080000 Image-hihope-rzg2m.bin; fatload
mmc 0:1 0x48000000 Image-r8a774a1-hihope-rzg2m.dtb; booti 0x48080000 - 0x48000
000'
=> saveenv
Saving Environment to SPI Flash... SF: Detected w25m512jv with page size 256 B
ytes, erase size 4 KiB, total 32 MiB
Erasing SPI flash...Writing to SPI flash...done
OK
```

Note) The setting above assumes the SD card has two partitions and stores data as below:
First partition: formatted as FAT, includes Image-hihope-rzg2m.bin and Image-r8a774a1-hihope-rzg2m.dtb
Second partition: formatted as ext4, rootfs image is expanded
Please refer to chapter 3. Create a microSD card to boot Linux.

Note) Please replace the file names in “bootcmd” according to the Release Note.

Now the board can bootup normally. Please turn off and on the power again to boot up the board.

4.1.11 Note for RZ/G2H, RZ/G2M, and RZ/G2N

The dtb files listed in the Table 4 cannot be used for the early revision of Hoperun boards. If you are using revision 2 boards, please use below files. These are automatically generated at the same place as the other image files when building a BSP.

HiHope RZ/G2M board:

- Image-r8a774a1-hihope-rzg2m-**rev2**.dtb (main board only)
- Image-r8a774a1-hihope-rzg2m-**rev2**-ex.dtb (main + sub board)
- Image- r8a774a1-hihope-rzg2m-**rev2**-ex-idk-1110wr.dtb (main + sub board + LVDS-IF)
- Image-r8a774a1-hihope-rzg2m-**rev2**-ex-mipi-2.1.dtb (main + sub board + MIPI/CSI2 cameras)

HiHope RZ/G2N board:

- Image-r8a774b1-hihope-rzg2n-**rev2**.dtb (main board only)
- Image-r8a774b1-hihope-rzg2n-**rev2**-ex.dtb (main + sub board)
- Image- r8a774b1-hihope-rzg2n-**rev2**-ex-idk-1110wr.dtb (main + sub board + LVDS-IF)
- Image-r8a774b1-hihope-rzg2n-**rev2**-ex-mipi-2.1.dtb (main + sub board + MIPI/CSI2 cameras)

4.2 Silicon Linux RZ/G2E evaluation kit (EK874)

4.2.1 Preparation of Hardware and Software

The following environment of Hardware and Software is used in the evaluation.

Hardware preparation (Users should purchase the following equipment.):

- AC Adapter 12V2A with EIAJ3 connector (Any AC Adapter)
- USB Type-microB cable (Any cables)
- HDMI cable (Any cables)

PC Installed Silicon Labs VCP driver and Terminal software (Tera Term)

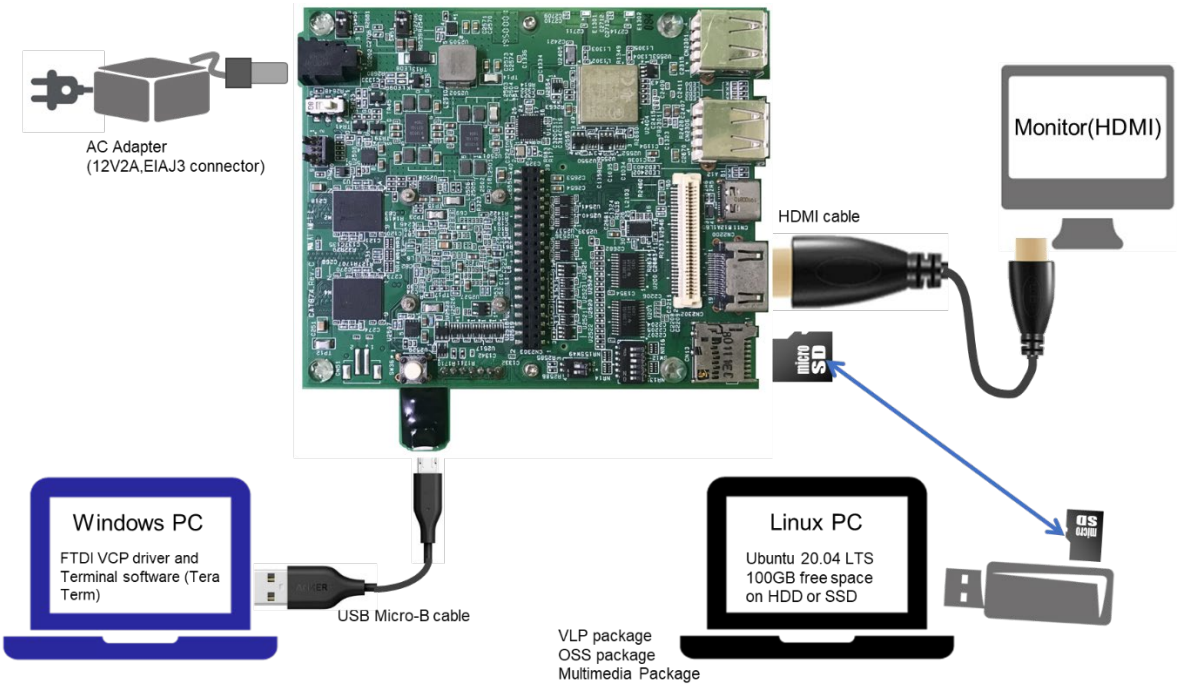


Figure 6. Operating environment

4.2.2 Building files to write

This board uses the files below as a bootloader. Please build them according to the Release Note and copy these files to the PC which runs a serial terminal software.

- bootparam_sa0.srec
- bl2-ek874.srec
- cert_header_sa6.srec
- bl31-ek874.srec
- u-boot-elf-ek874.srec

4.2.3 Settings

Connect between the board and a control PC by USB serial cable according to the Release Note.

Set the settings about serial communication protocol on a terminal software as below:

- Speed: 115200 bps
- Data: 8bit
- Parity: None
- Stop bit: 1bit
- Flow control: None

To set the board to SCIF Download mode, set the SW11 as below:

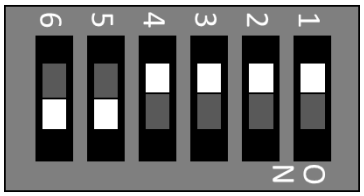


Table 14. SW11

6	5	4	3		2	1
ON	ON	OFF	OFF		OFF	OFF

4.2.4 How to use debug serial (console ouput)

Please connect USB Type-microAB cable.

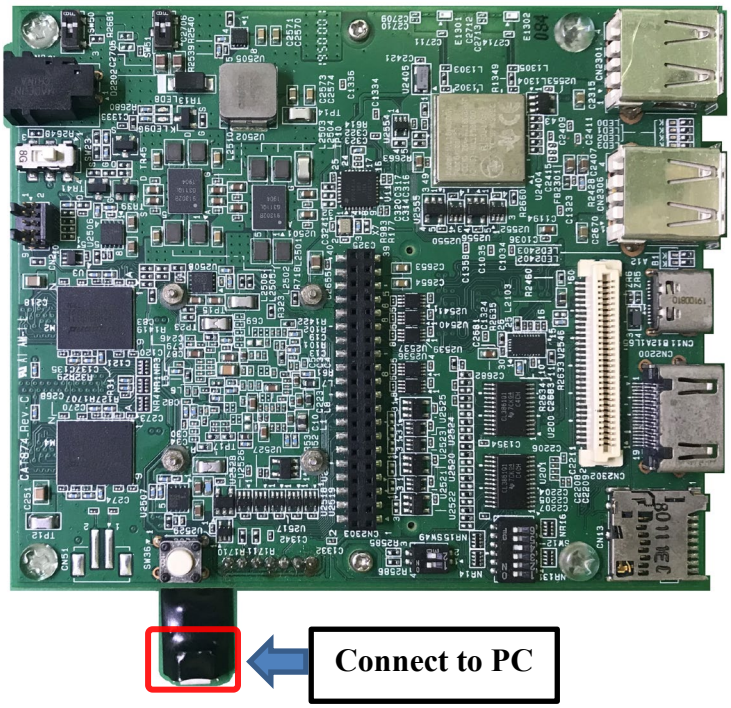


Figure 7. Connecting console for debug

4.2.5 Power supply

1. Connect AC adapter to Connector (CN36).

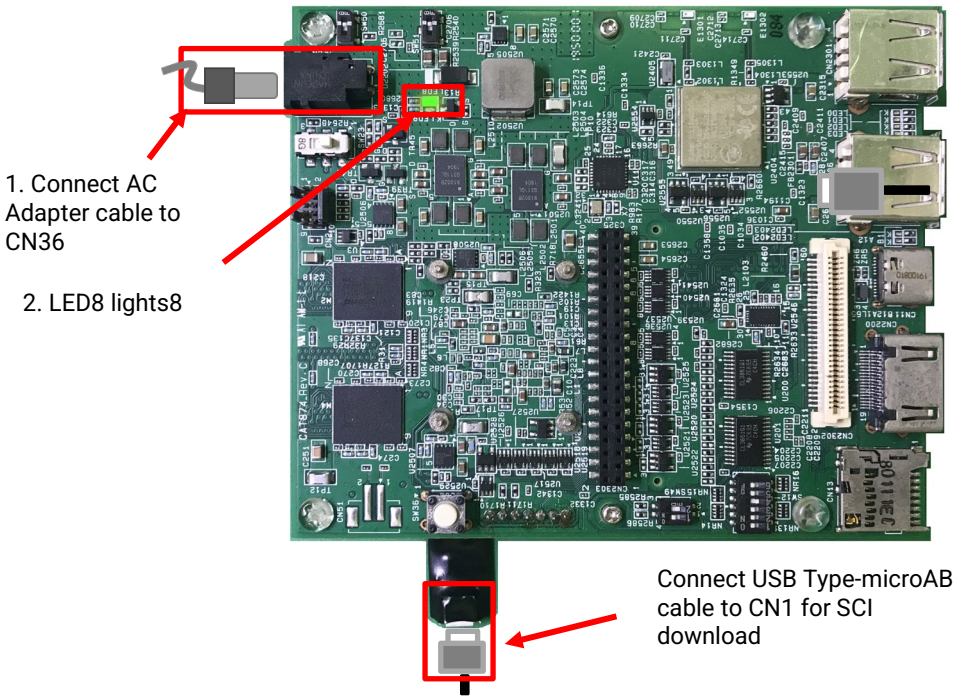


Figure 8. Connecting Power Supply

2. LED8 lights up.
3. Slide the power switch (SW23) to turn on the power.
4. LED9 lights up.

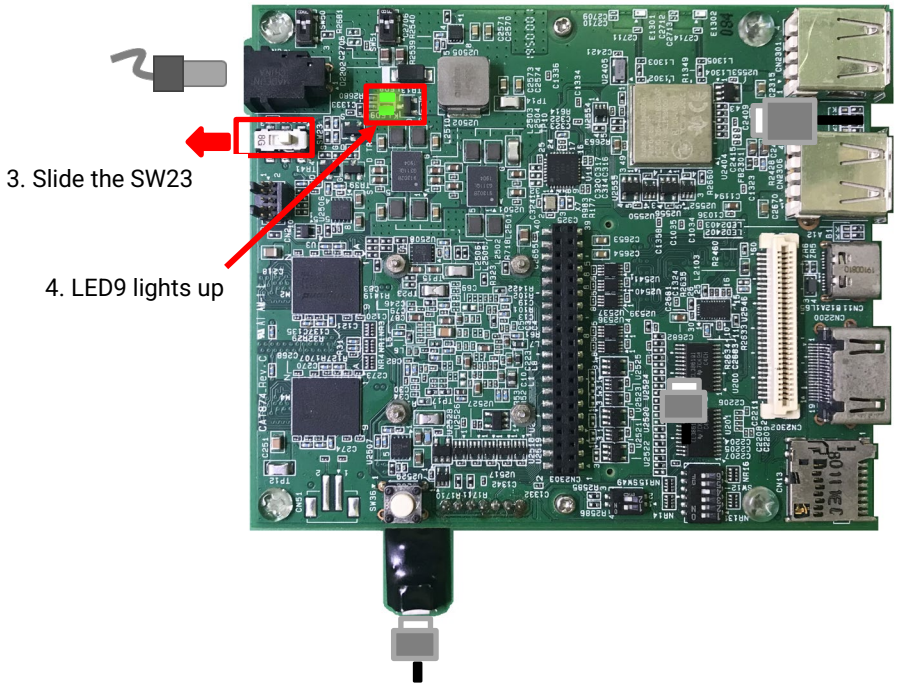


Figure 9. Power ON

4.2.6 Building files to write

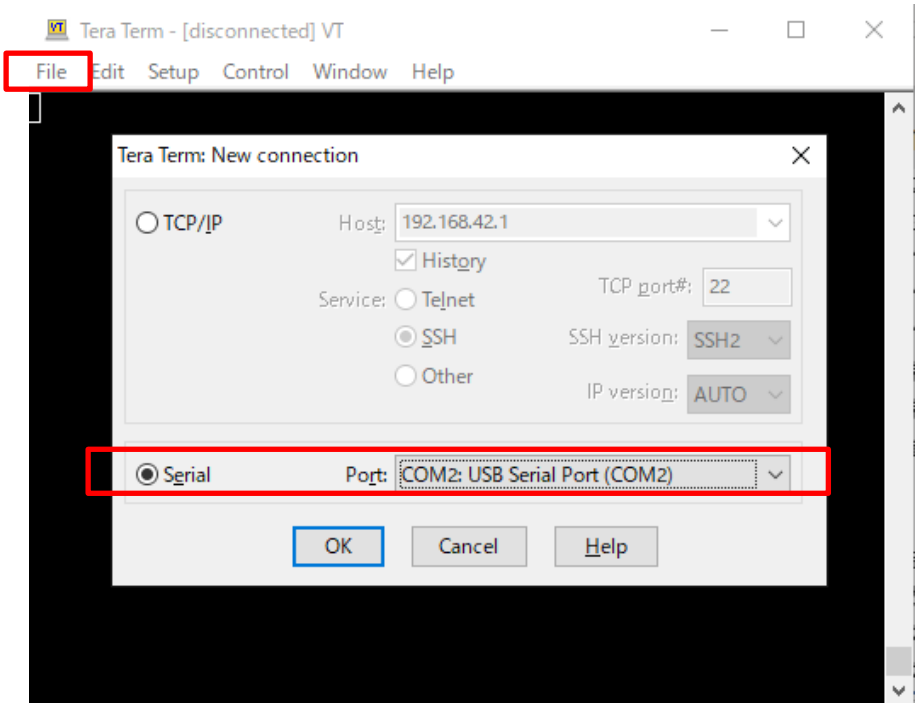
The evaluation boards use the files in the Table 15 as the boot loaders. The boot loaders files are generated by the build instructions of the section 2. Please copy these files to a PC which runs serial terminal software.

Table 15. File names of Boot loader

Board	File name of Boot loader
Silicon Linux RZ/G2E evaluation kit (EK874)	• u-boot-elf-ek874.srec • bootparam_sa0.srec • bl2-ek874.srec • bl31-ek874.srec • tee-ek874.srec • cert_header_sa6.srec

4.2.7 Settings

3. Bring up the terminal software and select the “File” > “New Connection” to set the connection on the software.



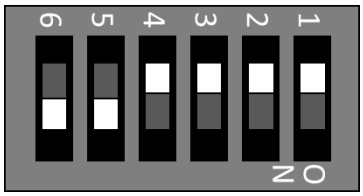
4. Select the “Setup” > “Serial port” to set the settings about serial communication protocol on the software.

Set the settings about serial communication protocol on a terminal software as below:

- Speed: 115200 bps
- Data: 8bit
- Parity: None
- Stop bit: 1bit
- Flow control: None

5. To set the board to SCIF Download mode, set the SW11 as below:

To set the board to SCIF Download mode, set the SW12 which is placed near the micro SD card slot as below:



6	5	4	3	2	1
ON	ON	OFF	OFF	OFF	OFF

4.2.8 Download Flash Writer to RAM

Turn on the power of the board by changing the SW23. Messages below are shown on the terminal.

```
 SCIF Download mode (w/o verification)
 (C) Renesas Electronics Corp.

-- Load Program to SystemRAM ---------------
please send !
```

Send an image of Flash Writer (AArch64_Flash_writer_SCIF_DUMMY_CERT_E6300400_ek874.mot) from terminal software after the message “please send !” is shown.

.

After successfully download the binary, Flash Writer starts automatically and shows a message like below on the terminal.

```
Flash writer for RZ/G2E V1.00 Sep.24,2019
>
```

4.2.9 Writing Bootloader

"XLS2" command of Flash Writer is used to write binary files. This command receives binary data from the serial port and write the data to specified address of the Flash ROM with information where the data should be loaded on the address of the main memory.

For detail of the procedure, please refer to the section **4.1.9 Writing Bootloader**.

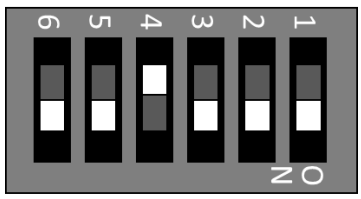
Write all necessary files using the addresses listed at the Table 16 and turn off the power of the board by changing the SW23.

Table 16. Addresses for each file

File name	Address to load to RAM	Address to save to ROM
bootparam_sa0.srec	E6320000	000000
bl2-ek874.srec	E6304000	040000
cert_header_sa6.srec	E6320000	180000
bl31-ek874.srec	44000000	1C0000
tee-ek874.srec	44100000	200000
u-boot-elf-ek874.srec	50000000	300000

4.2.10 Setting U-boot

To set the board to SPI Boot mode, set the SW12 which is placed near the micro SD card slot as below:



6	5	4	3	2	1
ON	ON	OFF	ON	ON	ON

Turn on the power of the board by changing the SW23.

```
U-Boot 2021.10 (Mar 31 2022 - 03:57:20 +0000)
U-Boot 2021.10 (Sep 15 2023 - 05:00:35 +0000)

CPU:   Renesas Electronics R8A774C0 rev 1.1
Model: Silicon Linux RZ/G2E evaluation kit EK874 (CAT874 + CAT875)
DRAM:  1.9 GiB
WDT:   watchdog@00000000e6020000
WDT:   Started with servicing (60s timeout)
MMC:   mmc@ee100000: 0, mmc@ee160000: 1
Loading Environment from SPIFlash... SF: Detected w25q512jv with page size 256 Bytes,
erase size 4 KiB, total 64 MiB
OK
In:    serial@e6e88000
Out:   serial@e6e88000
```

```
Err:   serial@e6e88000
U-boot WDT started!
Net:
Error: ethernet@e6800000 address not set.
No ethernet found.

Hit any key to stop autoboot: 0
```

Following the messages above, many warning messages will be shown. These warnings are eliminated by setting correct environment variables. Please set default value and save them to the Flash ROM.

```
=> env default -a
## Resetting to default environment
=> saveenv
Saving Environment to SPI Flash... SF: Detected w25m512jv with page size 256 Bytes, er
ase size 4 KiB, total 32 MiB
Erasing SPI flash...Writing to SPI flash...done
OK
```

If you created the microSD card according to the **step 3.2**, perform the following settings. If you created the microSD card according to the **step 3.1**, this setting is not required.

Set environment variables using the commands below:

```
=> setenv bootargs 'root=/dev/mmcblk1p2 rootwait'
=> setenv bootcmd 'fatload mmc 0:1 0x48080000 Image-ek874.bin; fatload mmc 0:1
 0x48000000 Image-r8a774c0-ek874.dtb; booti 0x48080000 - 0x48000000'
=> saveenv
Saving Environment to SPI Flash... SF: Detected w25m512jv with page size 256 B
ytes, erase size 4 KiB, total 32 MiB
Erasing SPI flash...Writing to SPI flash...done
OK
```

When you use the early version of EK874 (Revision A, B, C), please set “bootargs” as below.

```
=> setenv bootargs 'root=/dev/mmcblk0p2 rootwait'
```

Note) The setting above assumes the SD card has two partitions and stores data as below:
First partition: formatted as FAT, includes Image-ek874.bin and Image-r8a774c0-ek874.dtb
Second partition: formatted as ext4, rootfs image is expanded.
Please refer to chapter 3. Create a microSD card to boot Linux.

Now the board can bootup normally. Please turn off and on the power again to boot up the board.

4.2.11 Note for RZ/G2E

The dtb files listed in the Table 4 cannot be used for the early version (ES1.0) of RZ/G2E.

If the board prints the messages below when turn on the power, ES1.0 of RZ/G2E is implemented on the board.

```
[    0.000096] NOTICE:  BL2: RZ G2E Initial Program Loader(CA53)
[    0.004373] NOTICE:  BL2: Initial Program Loader(Rev.1.0.23)
[    0.009991] NOTICE:  BL2: PRR is RZG G2E Ver.1.0
```

```
CPU: Renesas Electronics R8A774C0 rev 1.0
```

In this case, please use below dtb files instead. These are built simultaneously when building normal dtb files.

- Image-r8a774c0-**es10**-cat874.dtb (main board only)
- Image-r8a774c0-**es10**-ek874.dtb (main + sub board)
- Image-r8a774c0-**es10**-ek874-idk-2121wr.dtb (main + sub board + LVDS panel)
- Image-r8a774c0-**es10**-ek874-mipi-2.1.dtb (main + sub board + MIPI/CSI2 cameras)

In case both old and new RZ/G2E LSIs are used in your laboratory at the same time, dtb files can automatically be selected using the environment variable `cut_ver` which is set by u-boot program according to the LSI's information. Please store multiple dtb files in an SD card and set `bootcmd` of u-boot like this.

```
setenv bootcmd 'fatload mmc 0:1 0x48080000 Image; if test "${cut_ver}" = "10"; then fa
tload mmc 0:1 0x48000000 Image-r8a774c0-es10-ek874.dtb; else fatload mmc 0:1 0x4800000
0 Image-r8a774c0-ek874.dtb ; fi ; booti 0x48080000 - 0x48000000'
```

5. Booting and Running Linux

Set microSD card to slot on carry board.

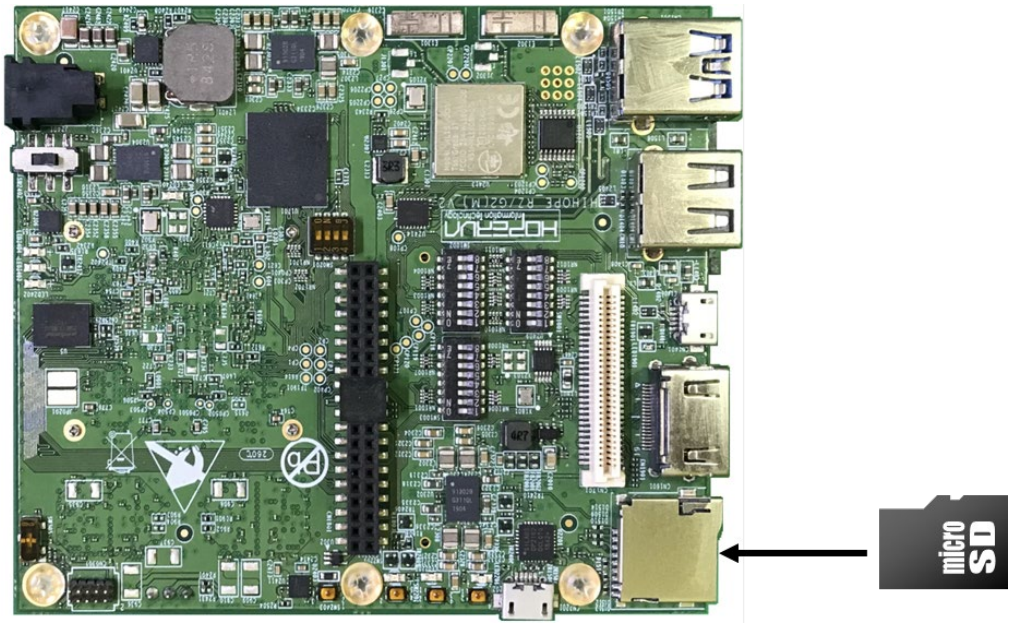


Figure 10. Set micro SD card to SMARC-EVK

Now the board can bootup normally. Please turn off and on the power again to boot up the board.

5.1 Power on the board and Startup Linux

After obtaining your reference board, please be sure to follow the document and write the bootloaders to the Flash ROM before starting the evaluation.

Before booting the board, please be sure to confirm the bootloaders which are built with your BSP/VLP are written to your board.

```
U-Boot 2021.10 (Sep 15 2023 - 05:00:35 +0000)

CPU:   Renesas Electronics R8A774C0 rev 1.1
Model: Silicon Linux RZ/G2E evaluation kit EK874 (CAT874 + CAT875)
DRAM:  1.9 GiB
WDT:   watchdog@00000000e6020000
WDT:   Started with servicing (60s timeout)
MMC:   mmc@ee100000: 0, mmc@ee160000: 1
Loading Environment from SPIFlash... SF: Detected w25q512jv with page size 256 Bytes,
erase size 4 KiB, total 64 MiB
OK
In:    serial@e6e88000
Out:   serial@e6e88000
Err:   serial@e6e88000
U-boot WDT started!
Net:
Error: ethernet@e6800000 address not set.
No ethernet found.

Hit any key to stop autoboot: 0
```

```
## Resetting to default environment
switch to partitions #0, OK
mmc0 is current device
19778048 bytes read in 788 ms (23.9 MiB/s)
47293 bytes read in 5 ms (9 MiB/s)
Error: Bad gzipped data
Moving Image from 0x48080000 to 0x48200000, end=49550000
## Flattened Device Tree blob at 48000000
   Booting using the fdt blob at 0x48000000
   Loading Device Tree to 0000000057ff1000, end 0000000057fff8bc ... OK

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd034]
[    0.000000] Linux version 5.10.184-cip36-yocto-standard (oe-user@oe-host) (aarch64-
poky-linux-gcc (GCC) 8.3.0, GNU ld (GNU Binutils) 2.31.1) #1 SMP PREEMPT Sat Feb 27 0
2:21:18 UTC 2021
[    0.000000] Machine model: Silicon Linux RZ/G2E evaluation kit EK874 (CAT874 + CAT8
75)
[    0.000000] earlycon: scif0 at MMIO 0x00000000e6e88000 (options '115200n8')
[    0.000000] printk: bootconsole [scif0] enabled
[    0.000000] efi: UEFI not found.
[    0.000000] Reserved memory: created CMA memory pool at 0x0000000058000000, size 25
6 MiB
 :
 :
 [  OK  ] Started Weston Wayland Compositor.
 [  OK  ] Started Update UTMP about System Runlevel Changes.
 [   12.655601] audit: type=1006 audit(1600598647.428:3): pid=425 uid=0 old-auid=42949
67295 auid=0 tty=(none) old-ses=4294967295 ses=1 res=1

 Poky (Yocto Project Reference Distro) 3.1.26 ek874 ttySC0

 BSP: RZG2E/EK874/3.0.5
 LSI: RZG2E
Version: 3.0.5
```

5.2 Shutdown the Board

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@ek874:~# 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, you can see " reboot: Power down" message.

Step 3. Turn off the power switch on the board

After checking the above LEDs, turn SW23 off.

6. Building the SDK

To build Software Development Kit (SDK), run the commands below after the steps (1) – (6) of section 2.1 are finished.

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.

For building general applications:

```
$ cd ~/rzg_vlp_<package version>/build
$ MACHINE=<board> bitbake core-image-weston -c populate_sdk
```

For building Qt applications:

```
$ cd ~/rzg_vlp_<package version>/build
$ MACHINE=<board> bitbake core-image-qt -c populate_sdk
```

The resulting SDK installer will be located in **build/tmp/deploy/sdk/**

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

```
$ sudo sh poky-glibc-x86_64-core-image-weston-aarch64-<board>-toolchain-3.1.26.sh
```

Or

```
$ sudo sh poky-glibc-x86_64-core-image-qt-aarch64-<board>-toolchain-3.1.26.sh
```

- Hoperun Technology HiHope RZ/G2H platform hihope-rzg2h
- Hoperun Technology HiHope RZ/G2M platform hihope-rzg2m
- Hoperun Technology HiHope RZ/G2N platform hihope-rzg2n
- Silicon Linux RZ/G2E evaluation kit ek874

Note) The SDK build may fail depending on the build environment. At that time, please run the build again after a period of time. Or build it again from scratch with the below commands.

```
$ cd ~/rzg_vlp_<package version>/build
$ MACHINE=<board> bitbake core-image-weston -c cleanall
$ MACHINE=<board> bitbake core-image-weston
```

For building general applications:

```
$ MACHINE=<board> bitbake core-image-weston -c populate_sdk
```

For building Qt applications:

```
$ MACHINE=<board> bitbake core-image-qt -c populate_sdk
```

7. Application Building and Running

This chapter explains how to make and run an application for RZ/G2H,M,N,E with this package.

7.1 Make an application

Here is an example of how to make an application running on VLP. The following steps will generate the “Hello World” sample application.

Note that you must build (bitbake) a core image for the target and prepare SDK before making an application. Refer to the start-up guide on how to make SDK.

7.1.1 How to extract SDK

Step 1. Install toolchain on a Host PC:

```
$ sudo sh ./poky-glibc-x86_64-core-image-weston-sdk-aarch64-toolchain-<version>.sh
```

Note:

sudo is optional in case user wants to extract SDK into a restricted directory (such as: /opt/).

If the installation is successful, the following messages will appear:

```
renesas@49d1d5882435:/mnt/host$ sudo sh ./rzg_vlp_<package version>/build/tmp/deploy/s
dk/poky-glibc-x86_64-core-image-weston-aarch64-hihope -rzg2n-toolchain-x.x.xx.sh
Poky (Yocto Project Reference Distro) SDK installer version x.x.xx
==========================================================================
Enter target directory for SDK (default: /opt/poky/x.x.xx):
You are about to install the SDK to "/opt/poky/x.x.xx". Proceed [Y/n]? Y
Extracting SDK.....................................................................
.....................................................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
Each time you wish to use the SDK in a new shell session, you need to source the envir
onment setup script e.g.
 $ . /opt/poky/x.x.xx/environment-setup-aarch64-poky-linux
 $ . /opt/poky/x.x.xx/environment-setup-armv7vet2hf-neon-vfpv4-pokymllib32-linux-gnuea
bi
```

Step 2. Set up cross-compile environment:

```
$ source /<Location in which SDK is extracted>/environment-setup-aarch64-poky-linux
```

Note:

User needs to run the above command once for each login session.

```
$ source /opt/poky/x.x.xx/environment-setup-aarch64-poky-linux
```

7.1.2 How to build Linux application

Step 1. Go to linux-helloworld directory:

```
$ cd $WORK/linux-helloworld
```

Step 2. Cross-compile:

```
$ make
```

Step 3. Copy all files inside this directory to /usr/share directory on the target board:

```
$ scp -r $WORK/linux-helloworld/ <username>@<board IP>:/usr/share/
```

Step 4. Run the application:

```
/usr/share/linux-helloworld/linux-helloworld
```

How to extract SDK

Step 1. Make a work directory for the application on the Linux host PC.

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

Step 2. Make the following three files (an application file, Makefile, and configure file) in the directory for the application.

Here, the application is made by automake and autoconf.

• main.c

```
#include <stdio.h>
/* Display "Hello World" text on terminal software */
int main(int argc, char** argv)
{
  printf("\nHello World\n");
  return 0;
}
```

• Makefile.am

```
APP = linux-helloworld
SRC = main.c

all: $(APP)

CC ?= gcc

# Options for development
CFLAGS = -g -O0 -Wall -DDEBUG_LOG

$(APP):
        $(CC) -o $(APP) $(SRC) $(CFLAGS)

install:
        install -D -m755 $(APP) $(DESTDIR)/home/root/$(APP)

clean:
        rm -rf $(APP)
```

Step 3. Make the application by the generated makefile.

```
$ make
```

```
renesas@49d1d5882435:/mnt/host/linux-helloworld$ make
aarch64-poky-linux-gcc  -mtune=cortex-a55 -fstack-protector-strong  -D_FORTIFY_SOURCE=
2 -Wformat -Wformat-security -Werror=format-security --sysroot=/opt/poky/3.1.26/sysroo
ts/aarch64-poky-linux -o linux-helloworld main.c -g -O0 -Wall -DDEBUG_LOG
renesas@49d1d5882435:/mnt/host/linux-helloworld$ ls -l
total 24
-rwxr-xr-x 1 renesas renesas 15320 May 10 23:32 linux-helloworld
-rw-r--r-- 1 renesas renesas   148 Sep 24  2020 main.c
-rw-r--r-- 1 renesas renesas   250 Sep 24  2020 Makefile
```

After making, confirm that the execute application (the sample file name is “hello”) is generated in the hello_apl folder. Also, this application must be cross-compiled for AArch64. Mar.17.2023 4.2

Store a sample application

The sample application could be written by the following procedure. The application should be stored in the ext3 partition.

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

Notes: 1. “sdb2” (above in red) may depend on using system.

2. is an optional directory name to store the application.

7.2 Run a sample application

Power on the RZ/G2H,M,N,E Evaluation Board Kit and start the system. After booting, run the sample application with the following command.

```
BSP: RZG2N/HIHOPE-RZG2N/3.0.5
LSI: RZG2N
Version: 3.0.5
hihope-rzg2n login: root
root@hihope-rzg2n:~# ls
linux-helloworld  v4l2-init.sh
root@ hihope-rzg2n:~# ./linux-helloworld

Hello World
root@ hihope-rzg2n:~#
```

Note: Refer to the start-up guide for the method of how to boot the board and system.

8. Appendix

8.1 Preparing Flash Writer

Flash Writer is built automatically when building BSP by bitbake command. Please refer to the Release Note of the Verified Linux Package to obtain a binary file of Flash Writer.

If you need latest one, please get source code from the GitHub repository and build it according to the following instructions. In general, new revision of reference boards requires latest Flash Writer.

8.1.1 Preparing cross compiler

FlashWriter runs on target boards. Please get cross compiler built by Linaro or setup a Yocto SDK.

Linaro toolchain:

```
$ cd ~/
$ wget https://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-e
lf/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-elf.tar.xz
$ tar xvf gcc-linaro-7.3.1-2018.05-x86_64_aarch64-elf.tar.xz
```

Yocto SDK:

Build an SDK according to Release Notes and install it to a Linux Host PC. Then, enable the SDK as below.

```
$ source /opt/poky/3.1.26/environment-setup-aarch64-poky-linux
```

8.1.2 Building Flash Writer

Get source codes of Flash Writer from the GitHub repository and checkout according to the version of VLP you use.

```
$ cd ~/
$ git clone https://github.com/renesas-rz/rzg2_flash_writer.git
```

For VLP/G v3.0.xx

```
$ cd rzg2_flash_writer
$ git checkout -b tmp 6606c4f4351f56fb3bd84d7836c01c8932a4f884
```

For VLP64 v1.0.xx

```
$ cd rzg2_flash_writer
$ git checkout -b v1.05 v1.05
```

Build Flash Writer as an s-record file by the following commands. Please specify a target board by "BOARD" option.

Linaro toolchain:

```
$ make -f makefile.linaro clean
$ CROSS_COMPILE=~/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-elf/bin/aarch64-elf- make -f
 makefile.linaro BOARD=<board>
```

Yocto SDK:

```
$ make clean
$ make BOARD=<board>
```

Please replace <*board*> to a proper option according to this table.

Target board	**BOARD option** *<board>*	**Image to be generated**
HiHope RZ/G2H	HIHOPE	AArch64_Flash_writer_SCIF_DUMMY_CERT_E6300400_hihope.mot
HiHope RZ/G2M		
HiHope RZ/G2N		
EK874	EK874	AArch64_Flash_writer_SCIF_DUMMY_CERT_E6300400_ek874.mot

8.2 Device drivers

The following drivers are supported: For detail information on how to use, please refer to these documents in the BSP manual set (RTK0EF0045Z9002AZJ-v3.0.x.zip).

Table 17. Support device drivers

Device Driver	Documents
Kernel Core	R01US0408EJxxxx_KernelCore_UME_vxxx.pdf
GPIO	R01US0405EJxxxx_GPIO_UME_vxxx.pdf
Power Management	R01US0420EJxxxx_PM_UME_vxxx.pdf
Thermal Sensor Unit (TSU)	R01US0421EJxxxx_Thermal_UME_vxxx.pdf
IPMMU	R01US0423EJxxxx_IPMMU_UME_vxxx.pdf
Direct Memory Access Controller (DMAC)	R01US0403EJxxxx_DMAE_UME_vxxx.pdf
Initial Program Loader	R01US0495EJxxxx_IPL_UME_v1.02.pdf
Boot loader for RZ/G2 Group System Evaluation Board	R01US0414EJxxxx_UBOOT_UME_v1.12.pdf
Video Input Interface	R01US0424EJxxxx_VideoCapture_UME_vxxx.pdf
Gstreamer Startup Guide	R01US0400EJxxxx_GStreamer_UME_vxxx.pdf
Wayland Solution Guide	R01US0399EJxxxx_Wayland_UME_vxxx.pdf
Video for Linux 2 (VSP-du Unit or VSPS Unit)	R01US0426EJxxxx_V4L2_UME_vxxx.pdf
VSP Manager for Linux	R01US0427EJxxxx_VSPM_UME_vxxx.pdf
Display Unit	R01US0402EJxxxx_Display_UME_vxxx.pdf
Audio Interface (Audio-DMAC, Audio-DMACpp, SCU, SSIU, and ADG)	R01US0401EJxxxx_AUDIO_UME_vxxx.pdf
gPTP Timer in Ethernet AVB Unit	R01US0412EJxxxx_PTP_UME_vxxx.pdf
Gigabit Ethernet Interface	R01US0404EJxxxx_GEther_UME_vxxx.pdf
PCI Express Controller	R01US0411EJxxxx_PCIEC_UME_vxxx.pdf
Serial communication Interface with FIFO (SCIF) and High Speed Serial Communication Interface with FIFO (HSCI-F)	R01US0406EJxxxx_SCIF_HSCIF_UME_vxxx.pdf
I2C Bus Interface	R01US0407EJxxxx_I2C_UME_vxxx.pdf
PWM output	R01US0422EJxxxx_PWM_UME_vxxx.pdf
Clock-Synchronized Serial Interface with FIFO (MSIOF)	R01US0410EJxxxx_MSIOF_UME_vxxx.pdf
Serial-ATA Interface	R01US0413EJxxxx_SATA_UME_vxxx.pdf
USB 2.0 Host	R01US0417EJxxxx_USB2.0H_UME_vxxx.pdf
USB 2.0 Function	R01US0415EJxxxx_USB2.0F_UME_vxxx.pdf
USB 3.0 Host	R01US0416EJxxxx_USB3.0F_UME_vxxx.pdf
USB 3.0 Function	R01US0418EJxxxx_USB3.0H_UME_vxxx.pdf
Watchdog Timer (WDT)	R01US0425EJxxxx_Watchdog_UME_vxxx.pdf

Note) “xxx” is the revision of the file. Please refer to the latest one.

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Mar. 31, 2022	–	First edition issued.
1.01	Jun. 6, 2022	11	Add the instructions to format SD cards.
		17	Add the Notes section.
		18	Move the section “Preparing Flash Writer” to the chapter 4 “Appendix”.
1.02	Mar. 31, 2023	-	Add build instruction.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/