Quick Start Guide

Renesas System Release Package

RZ Common, SBC & EVK

Quick Start Guide

Renesas RZ Family
RZ G/V/H Series

Renesas Electronics
www.renesas.com

Rev.3.00 Sep.25.25

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

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

Trademarks

Trademarks (continued)

For the “Cortex” notation, it is used as follows;
— Arm® Cortex®-A55
— Arm® Cortex®-M33
Note that after this page, they may be noted as Cortex-A55 and Cortex-M33 respectively.

Examples of trademark or registered trademark used in the RZ/G2L SMARC Module Board RTK9744L23C01000BE User’s Manual: Hardware;
CoreSight™: CoreSight is a trademark of Arm Limited.
MIPI®: MIPI is a registered trademark of MIPI Alliance, Inc.
eMMC™: eMMC is a trademark of MultiMediaCard Association.

Note that in each section of the Manual, trademark notation of ® and TM may be omitted.
All other trademarks and registered trademarks are the property of their respective owners.

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

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

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

Renesas RZ Family

Renesas System Release Package

Introduction

This user manual describes the unified system release package. The system release package contains supported hardware and software.

The result is a consistent experience across the different platforms. This streamlines the development effort for user applications.

Package Contents

The system release package contains the following:

- Multiple Images that are geared to general baseline use cases.
- Yocto build scripts.
- Host side tools.
- Environmental files.
- SDKs for all images
- Documentation which includes
- User manual
- Copyright & License information

Features

The following are the general features of the system release package.

- Architected to support multiple platforms with the same image and tools over time.
- Common frameworks
- Open-source packages using GPLv2 and GPLv3 packages
- Carefully considered base images that allow for a quick starting point to build a product.
- Complete set of features working out of the box.
- Seamless out-of-box experience.
- Automated Yocto build scripts that can rebuild the entire package with only a few commands.
- Host tools to flash the firmware in multiple processes.
- Tools supporting both Linux and Windows workflows.
- Docker-friendly build scripts.
- Extensive documentation covering the hardware, software, and application development and deployment.

Contents

Glossary

Terms	Description
802.11 - Wi-Fi	The technical name of the standard specification for Wi-Fi is 802.11. This is also the working group that develops and maintains the standards for Wi-Fi that everyone conforms to.
ADC – Analog to digital converter	A hardware unit that converts an input analog signal to a digital value by measuring its immediate voltage at a fixed resolution.
BSP – Board Support Package	BSP is an essential software package that has bootloaders, Linux kernel, a minimal user space and programming tools, allowing the device to boot. This core software allows the system to boot into an operating system, enables all the features and allows application development.
CAN – Controller area network	This is a standardized communication protocol used widely on automotive and aerospace systems. It connects various ECU's known as nodes and uses two wires / lines as a pair carrying differential signals. This method of signaling allows long length cables to interface different systems on the machine with reliable signals. The CAN protocol has multiple specifications and is an ISO standard. It supports flexible data rates reaching as high as 8Mbps. Most automobiles have CAN networks in them, and it is a part of OBD-2 specification which is mandatory law in most of the world for automotive machines like cars.
DAC – Digital to analog converter	A hardware unit that takes digital value and exerts a corresponding analog voltage on an output line.
Firmware	For the scope of this document, the term 'firmware' refers to the low-level software that runs before an OS takes over. This includes arm trust zone, optee & u-boot at the very least. It also refers to the standalone binaries that run on the embedded real-time core like the CM33.
I2C - Inter Integrated circuit protocol:	This is a communication protocol used to implement digital communication between two devices (chips / board) using only two wires. It is a standardized specification and is used widely to implement low to medium data rate data transfers both among devices on the same circuit board as well as external add on peripheral boards. I2C can be implemented across a few meters in distance. I2C is half duplex meaning only one device can communicate at a time. Speeds range from 100 Kbps to 3Mbps while 100 / 400 Kbps are the typical operating mode. The other major advantage of this protocol is that it allows many devices to be on the same two lines reducing the cost of the interfacing. This is ideal when there are many devices like sensors that transfer limited amounts of data periodically. I2C can support up to 127 independent directly addressable devices on the same channel.
IEEE- Institute of Electrical and Electronics Engineers	IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity. It is a major technical organization covering vast fields of engineering and a major standards organization.
MCU – Micro controller unit	A micro controller unit is a self-contained unit that has the core processing as well as core memory within the same device. It often contains the core software programmed into the chip itself. This allows the device to start executing with minimal external devices / circuitry. Some microcontrollers can be powered on a mere breadboard.
MPU – Micro processing unit	An MPU is a processing unit: a CPU that contains only the processing core and interfaces for external peripherals. A microprocessor is usually a powerful CPU in its class. However, it requires a very large number of external circuitries to achieve its functionality like external memory, disk drives, etc.
PMIC – Power management IC	This is a specific chip on the board that manages multiple power supply lines at various levels. It manages the respective supplies along with sequences which control power on and power off cycles.
SBC – Single board computer	It is a standard term that means a tiny computer in the form factor of a single circuit board usually just inches in area. This board is self-sufficient in every way and can give you a usable computer with just a power supply, keyboard, mouse, and display.

SiP – System in Package	SiP is a device where multiple silicon IP's are combined to form a single device. It is one of the densest chips where the external devices like flash memory, DDR RAM and even Wi-Fi module are all packaged into a single chip. These are used in very niche application that require ultra small size and low thermal requirement.
SoC- System on Chip	A system on chip is a complete hardware platform packaged on to a single chip. It contains the CPU, internal fast memory, interrupt controllers, pin controllers, ROM memory, and a few other peripherals and sensors; all packaged into the same IC. An SoC despite the high level of integration does not necessarily power on and run by itself. Microcontrollers are often independent SoC's that can work on their own. However, SoC's often combine MPU's and MCU's into the same chip. This allows very powerful systems to be built in a compact form factor but requires external supporting peripherals like DDR RAM and flash memory and power management IC's.
SPI - Serial Peripheral interface	SPI is another standard interface used to interface other devices on the board or attaching peripheral boards. It specifies 3 wires / lines to achieve fast full duplex data transfer. Two devices can send / receive data at the same time in this protocol. The protocol is also a high-speed protocol where typical operating speeds start at 5Mbps and go over 50Mbps. This high speed allows interfacing high speed devices like memory, Wi-Fi, subsystems made of independent microcontrollers, etc. While only 3 lines are needed to interface two devices, a fourth line is used as a device selector allowing multiple devices to share the same interface. However, only two devices may communicate at a time.

1. Overview

The Renesas System Release Package is a unified software package that aims to provide an easy-to-use yet comprehensive software platform for the Renesas RZ series of SoC-based boards. It aims to provide fully functional base images for supported reference designs, along with easy-to-use development and programming tools that allow the user to quickly get started on their application development. This package aims to provide a standardized and familiar workflow for a similar experience across a variety of Renesas RZ SoC-based product platforms.

This package provides comprehensive documentation, Quick start guides, multiple Linux-based distribution images, automated tools and scripts, and an ongoing expansion of supported products.

1.1 Supported Distributions

The System Release package supports a set of both Yocto images and custom images to enable the user to start quickly on their embedded end application. The large collection of images in prebuilt format provides a wide set of capabilities. This release focuses on Yocto images.

1.1.1 Yocto Images

This section lists the standard Yocto images, offering a variety of configurations that cater to different embedded use cases. From a minimal bootable environment to fully graphical systems, these images provide the essential building blocks for embedded Linux development.

Table 1. Yocto images

Distribution	Image file	Version	Description
Yocto minimal	core-image-minimal	styhead-5.1.4	A basic image that contains the minimal set of components required to boot the device. It focuses on essential system functions without extra tools or features.
Yocto BSP	core-image-bsp	styhead-5.1.4	Extends core-image-minimal with additional utilities and tools, providing a lightweight environment for system validation, hardware diagnostics, and basic development.
Yocto weston	core-image-weston	styhead-5.1.4	A standard graphical image with Wayland and Weston support for embedded GUI applications.

1.1.2 Renesas Custom Images

This section presents Renesas-specific custom images, which are customized and optimized for Renesas products. These images offer specialized features, including fast booting and tailored environments for both graphical and CLI-based applications.

Table 2. Renesas custom images

Distribution	Image file	Version	Description
Renesas CLI Base	renesas-core-image-cli	styhead-5.1.4	Based on core-image-bsp, this image offers a CLI environment for Renesas hardware development without graphical interfaces. Besides the useful tools inherited from the core-image-bsp, this image also contains new packages for SBC (Single Board Computer) development. For example, package managers (apt, dpgk), network utilities for Bluetooth, Wi-Fi.
Renesas Quickboot CLI	renesas-quickboot-cli	styhead-5.1.4	This image has the same system functionality as the renesas-core-image-cli but with Quickboot enabled, allowing for faster boot times and efficient system validation on a CLI environment.
Renesas Weston (Qt6)	renesas-core-image-weston	styhead-5.1.4	Renesas customized core image based on the core-image-weston, with Qt6 framework support (no QT demo apps included). This image offers a full graphical environment for Renesas hardware development and all the useful tools from the renesas-core-image-cli.
Renesas Quickboot Wayland	renesas-quickboot-wayland	styhead-5.1.4	This image has the same system functionality as the renesas-core-image-weston but with Quickboot enabled, allowing for faster boot times and efficient system validation on a graphical environment.

Note: Quickboot is a trade term that refers to the specific optimizations that are performed to achieve ultra-low start-up times in specific images. Depending on the board architecture, the startup time can be as low as 2s. While there is no assurance of the startup time in these images for every platform, these images are the most optimized on our platforms.

1.1.3 Ubuntu Images

This section presents custom Ubuntu-based images tailored for embedded systems, offering a variety of configurations to suit both headless and graphical environments. These images are optimized for performance and ease of use, providing a solid foundation for deploying embedded applications on Renesas platforms.

Table 3. Renesas Ubuntu images

Distribution	Image file	Version	Description
Ubuntu Core	ubuntu-core-image	ubuntu-base-24.04	A minimal, headless Ubuntu image tailored for embedded systems.
Ubuntu LXDE	ubuntu-lxde-image	ubuntu-base-24.04	A lightweight Ubuntu image featuring the LXDE desktop environment, providing a graphical interface while maintaining low resource consumption. This image also includes Qt framework support for GUI development.

1.2 Supported Platforms

Table 4. Supported Platforms

Platform	SoC	OPN	Description
RZ/G2L-SBC	RZ/G2L	US157-G2LSBCPOCZ	RZ/G2L-based Pi-compatible SBC.
RZ/G2L-EVK	RZ/G2L	RTK9744L23S01000BE	Evaluation Board Kit for RZ/G2L MPU
RZ/V2L-EVK	RZ/V2L	RTK9754L23S01000BE	Evaluation Board Kit for RZ/V2L MPU
RZ/V2H-EVK	RZ/V2H	R9A09G057H48GBG#AC0 R9A09G057H48GBG#BC0	Evaluation Board Kit for RZ/V2H MPU

2. Quick Start

This section describes how to quickly get set up and start running the supported platforms with this release. The following are the essential steps for an SD-MMC card-based boot:

1. Select an image from the list of available images in Section 1.1. Supported Distributions
2. Prepare an SD MMC card that has the image programmed onto it.
3. Prepare the hardware with power and debug UART interface. Displaying the connection to one of the HDMI interfaces is highly recommended, but not essential.
4. Program the firmware using the appropriate scripts and process in the 'host/tools' directory of the package.
5. Boot normally with the SD MMC card.

2.1 SD-MMC Card Flashing

The Linux bootable SD card creation is a very simple process. The idea is to use any filesystem imaging tool (etcher) to burn the required image's '.wic' file (core-image-weston.wic for example) located in the 'target/images' directory of the release to the sd-mmc card. We recommend installing Balena etcher, which is available for Linux, MacOS, and Windows.

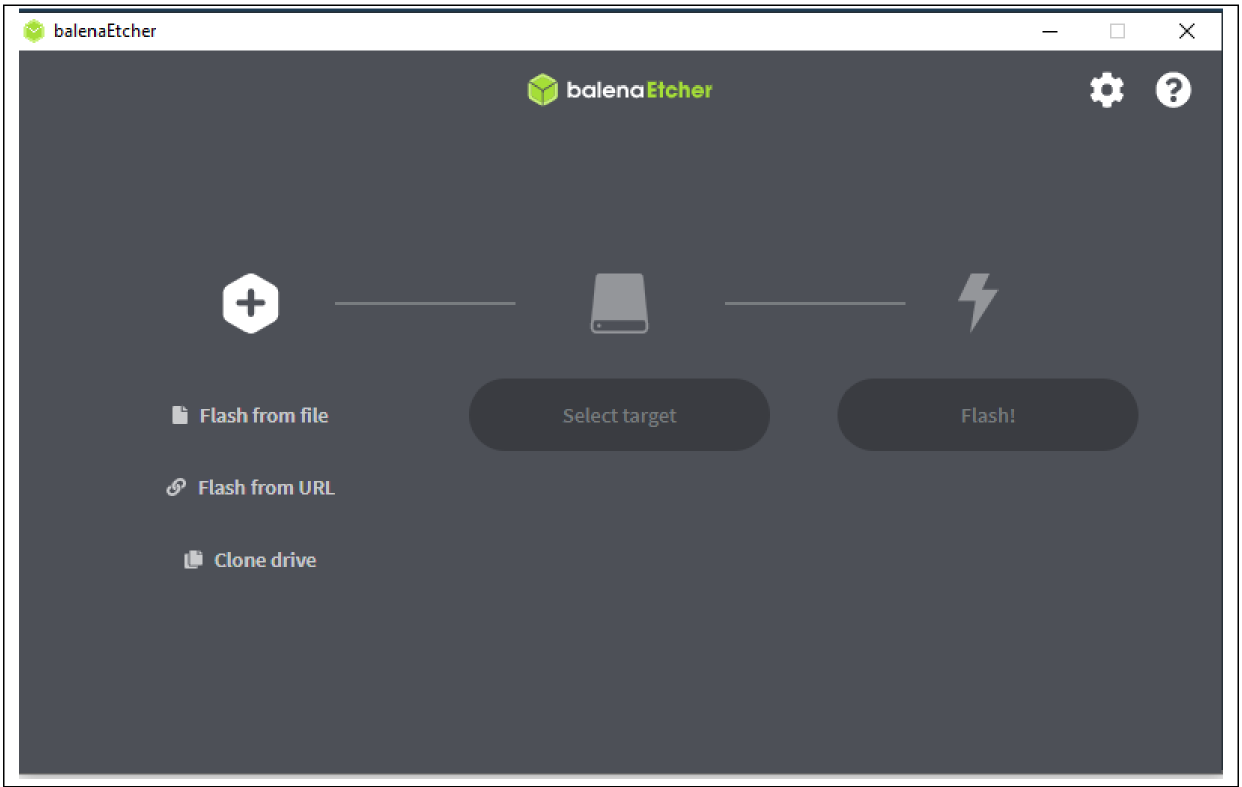

Figure 1. Balena etcher UI

Steps:

1. Select "Flash from File".
2. In the popup window, navigate to your release and select one of the chosen image files (core-image-weston.wic).
3. Then click on 'Select target,' and it will list all available devices.
4. Select your SD MMC card.
   Be mindful not to select your primary laptop/desktop hard drive.
5. Select 'Flash'.
6. When flashing is completed, it will automatically dismount the SD MMC card device.

7. Insert the SD card into the SD-MMC card slot on the RZ board.

2.2 Universal Flash

This section explains how to program and flash various firmware onto Renesas boards. It covers firmware components, prerequisites, hardware setup for each board, and usage of the universal flashing script for seamless flashing workflows.

The universal flashing script (universal_script.py) is included in the release package under the **host/tools** folder.

This package contains only the following firmware components:

Table 5. Firmware description

Module	Binary	Stack layer	Description
ROM code	N/A	BL1	This is the internal ROM code that the Arm Cortex SoC's primary core executes at POR.
Flash writer	Flash_Writer_SCIF_ <board>.mot	BL2	This is meant for serial load in factory environments, which is directly loaded onto the SRAM by the BL1 (ROM code) through UART SCIF0. It is then executed to acquire another image on UART SCIF0 to directly flash onto qspi or emmc into the boot sector. It provides a command-based ui.
Arm trusted Firmware-A	bl2-rz-cmn.bin bl31-rz-cmn.bin <board>.dtb	BL2 and BL31	Minimal Trusted Firmware-A implementation without a device tree. The flashing script will combine bl2-rz-cmn.bin with device tree dynamically during flashing. It comes in only .bin format: • .bin – for raw flashing for native in-system flashing
U-Boot (BL33)	u-boot-nodtb-rz-cmn.bin <board>.dtb	BL33	U-Boot (nodtb) binary and matching device tree. The flashing script combines these into the FIP.
Board Identification	<board>-platform-settings.bin		This binary stores key platform settings for Renesas boards, like model IDs, revisions, memory locations, and image sizes; enabling firmware and bootloaders to identify hardware and locate boot components efficiently during startup or flashing.

Note: This release does **not** ship a prebuilt FIP. Instead, the flashing script will automatically build a valid FIP image at flash time by combining:

- bl31-rz-cmn.bin
- u-boot-nodtb-rz-cmn.bin
- <board>.dtb

It also dynamically merges bl2-rz-cmn.bin with <board>.dtb to create the BL2 binary that is flashed to the boot sector.

2.2.1 Prerequisites

Before flashing any images, ensure the following system requirements are met on the host PC and that necessary files and tools are available.

- Operating System:
  - Linux (Ubuntu 20.04 or newer recommened)
  - Windows 10 or newer
- Software Requirements:
  - Python 3.8 or later
  - GNU Binutils: Required for objcopy.
  - Firmware release package with images and tools
- Hardware requirements:
  - Required cables: USB, UART debug cables
  - SD card (8GB or larger)

2.2.1.1 Linux Setup

Follow these steps to prepare a Linux host:

1. Install Python, Binutils and build tools

```
renesas@builder-pc:~# sudo apt update
renesas@builder-pc:~# sudo apt install python3 python3-pip binutils build-essential libssl-dev
```

   - python3, python3-pip: Required to run host scripts
   - binutils: Provides objcopy for binary conversion
   - build-essential (optional): Installs gcc, g++, make for rebuilding firmware
   - libssl-dev: OpenSSL package

2. Install Python Dependencies

   It is recommended to use a virtual environment with any supported Python version (3.10, 3.11, or 3.12).

   Example for Python 3.12

```
renesas@builder-pc:~/rz-cmn-srp-3.0/host/tools# sudo apt install python3.12-venv
renesas@builder-pc:~/rz-cmn-srp-3.0/host/tools# python3 -m venv .venv
renesas@builder-pc:~/rz-cmn-srp-3.0/host/tools# source .venv/bin/activate
```

   If the distribution uses a different Python 3 version (for example, 3.10 or 3.11), replace 3.12 with the appropriate version.

   After activating the virtual environment, install the required tools using requirements.txt.

```
renesas@builder-pc:~# cd <path/to/the/package>/host/tools/
renesas@builder-pc:~/rz-cmn-srp-3.0/host/tools$ pip3 install -r requirements.txt
```

2.2.1.2 Windows Setup

Follow these steps to prepare a Windows host:

1. Install Python3
   - Download and install Python 3 from python.org
   - During installation, enable “Add Python to environment variables”
2. If pip is missing, repair your Python installation or download get-pip.py and run:

```
PS C:\Users\renesas> py get-pip.py
```

3. Install Python Dependencies: Open one of the following terminals with Administrator privileges:
   - PowerShell
   - Git Bash

   The example below uses PowerShell, but the same applies to other terminals
   - Option 1 – Use **requirements.txt** (recommended)

```
PS C:\Users\renesas> cd <path/to/the/package/host/tools>
PS C:\Users\renesas\rz-cmn-srp-3.0\host\tools> py -m pip install -r
requirements.txt
```

   - Option 2 – Install manually
     - Using the Python launcher:

```
PS C:\Users\renesas> py -m pip install pyserial
PS C:\Users\renesas> py -m pip install tomli
PS C:\Users\renesas> py -m pip install dataclasses      # Only if Python < 3.7
```

     - Or using pip directly (if already in PATH)

```
PS C:\Users\renesas> pip install pyserial
PS C:\Users\renesas> pip install tomli
PS C:\Users\renesas> pip install dataclasses # required only if using Python
versions older than 3.7
```

4. Environment and Tool Dependencies
   - GNU Binutils:
     - Download and install MinGW-w64
     - Install to default location (**C:/MinGW**)
     - Add the following path to the Windows Environment Variables → Path:

```
C:/MinGW/bin
```

   - OpenSSL (For MingW-w64):
     - Download the package from: MinGW-w64 OpenSSL
     - Extract the package into: **C:/mingw64**

Important Notice for Windows users: Executables such as fiptool.exe depend on OpenSSL runtime DLLs.

- Add this directory to your Environment Variables → Path:
  - **C:/mingw64/bin**
- Or copy the DLLs (C:\mingw64\bin\libcrypto-3-x64.dll) into:
  - **<path/to/package>/tools/bin/windows/**
- If skipped, running the tools will fail

The firmware_compile.py script also depends on objcopy (part of GNU binutils).

- Ensure **C:/MinGW/bin** is also in Windows Environment Variables Path so that objcopy.exe can be found.
- Without it, the script will fail during SREC/ELF conversions.

2.2.2 Usage and Flashing Operations

The universal_script.py serves as the main entry point for performing all flashing operations. It uses the board configuration defined in flash_images.json to select the appropriate images and procedures.

For this quick start guide, only bootloader flashing using Serial Download (SCIF) mode is required. Other operations supported by the script, such as rootfs flashing over UDP, are covered in the full user manual.

For this release, use the default image paths included in the package. Overrides in flash_images.json are only required when adding a new board or replacing images.

Table 6: Required Image Locations and Resolution Priority

Component	Expected Location	Purpose	Override (from flash_images.json)
BL2/BL31	• </path/to/yocto/package>/target/images/atf/bl2-rz-cmn.bin • </path/to/yocto/package>/target/images/atf/bl31-rz-cmn.bin	Bootloader stages are responsible for early initialization (BL2) and runtime services (BL31).	Not overridable – always taken from the default location.
BL2 FCONF Device Trees	</path/to/yocto/package>/target/images/atf/fdts/	Configuration device trees for BL2 (DDR, PFC, CPG, etc.).	**atf_fdts** can override with a per-board FCONF DTB
U-Boot (no DTB)	</path/to/yocto/package>/target/images/u-boot/u-boot-nodtb-rz-cmn.bin	Per-board U-Boot device trees. Selected automatically by the flashing script.	Not overridable – always taken from the default location.
U-Boot DTBs	</path/to/yocto/package>/target/images/u-boot/dtbs/	Per-board U-Boot device trees. Selected automatically by the flashing script.	**uboot_dtb** can override selects the exact DTB for the board.
Root filesystem	</path/to/yocto/package>/target/images/(e.g., core-image-weston.wic	Complete root file system image written to SD card or eMMC.	**rootfs** and **rootfs_flash_method** override the image and flashing method (e.g., udp, otg).
Flash writer	</path/to/yocto/package>/target/images/Flash-writer-<board>.mot	Initial loader for XSPI/eMMC flashing.	**flash_writer** must define the correct image for the board
Board identification	• </path/to/yocto/package>/target/images/<board>-platform-settings.srec • </path/to/yocto/package>/target/images/<board>-platform-settings.bin	Per-board platform/ID blob used during flashing.	**board_identification** can override selects the file.

After all, required images are placed in their expected locations (and flash_images.json is configured if needed), proceed to run the script.

- **On Linux:** Open a terminal and run the following commands

```
renesas@builder-pc:~$ cd ~/renesas/rz-cmn-srp/host/tools/
renesas@builder-pc:~/renesas/rz-cmn-srp/host/tools/$ python3 universal_flash.py
```

- **On Windows:** Open one of the following terminals with Administrator privileges:
    - PowerShell
    - Git Bash
    - MobaXterma

  Navigate to the host/tools/ directory inside the rz-cmn-srp folder. The example below uses PowerShell, but the same applies to other terminals

```
PS C:\Users\renesas> cd C:\Users\renesas\rz-cmn-srp\host\tools\
PS C:\Users\renesas\rz-cmn-srp\host\tools\> py universal_flash.py
```

Upon execution, the script will present an interactive menu to choose the desired flashing operation.

2.2.2.1 Flashing Flow

To flash the bootloader using universal_script.py, follow these steps:

1. Select the target board and press Enter.
2. Select the serial port and press Enter.
3. Select the baud rate – usually the default (115200). Press Enter to accept it.
4. When asked “Do you want to write the rootfs? (y/n)”, type n and press Enter.
5. When asked “Write IPL method:”, choose:
    - 1 - BootloaderFlash → for bootloader flashing
    - 2 - UloadFlash → not used in this quick start

   Press 1 and hit Enter.
6. Reset the board to begin flashing:
    - If the board has a reset button, press it.
    - If it does not, power cycle the board.

The diagram below shows this simplified flow.

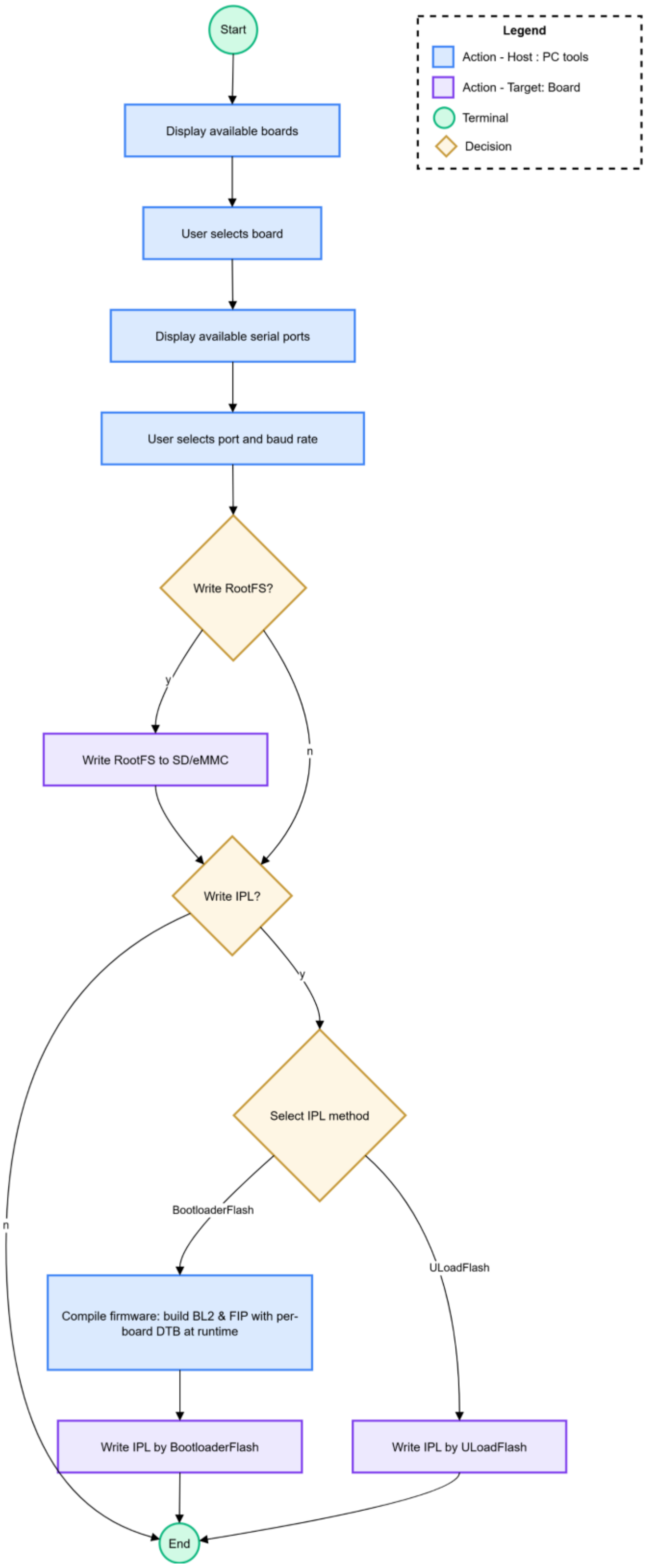

Figure 2: Universal Flashing Script — Interactive Workflow

2.3 RZ/G2L-SBC

This section describes the hardware-specific processes for the RZ/G2L-SBC (Single-board Computer).

Note:

- The release consists of images that have desktop and display support.
- At least one basic display, like a 1080p HDMI monitor, must be available for those images.
- You can also use the DSI touch panel described in the MIPI DSI Display Touch Panel.
- It is recommended to use an FTDI cable for the UART and not any other converter chip.

2.3.1 Overview of Connectors

Given below is the basic positioning of the top-level connectors.

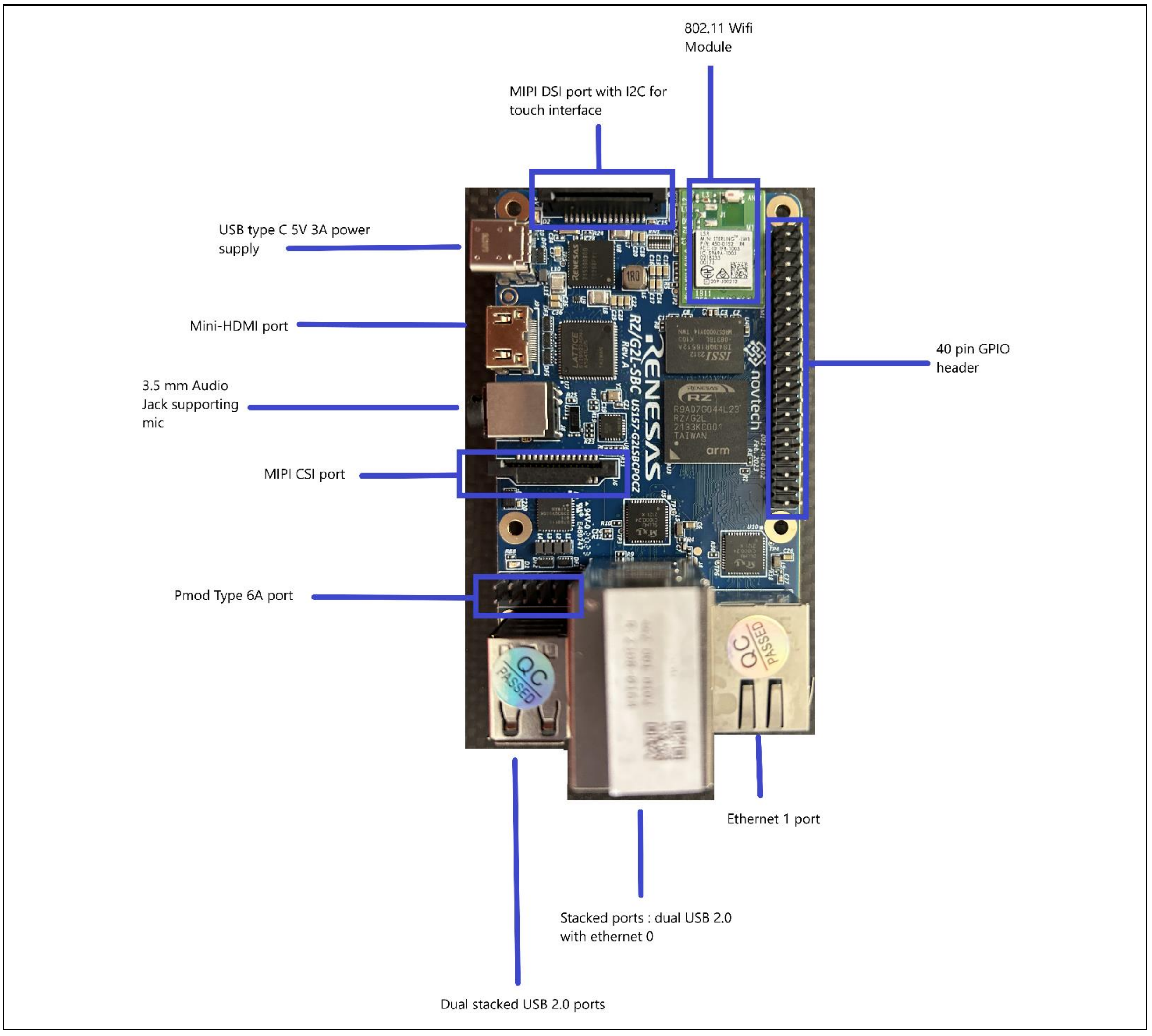

Figure 3. RZ/G2L-SBC top side connectors.

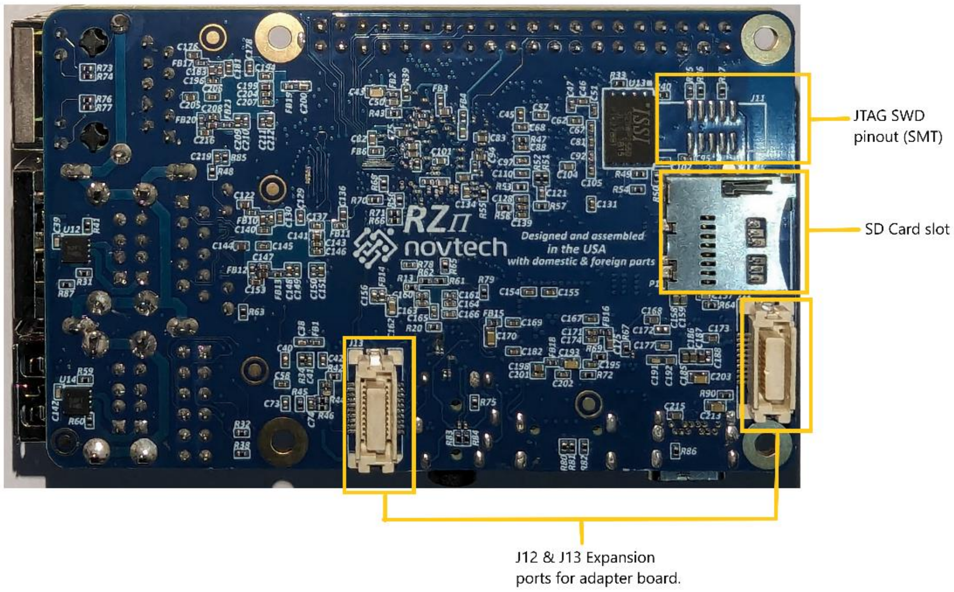

Figure 4. RZ/G2L-SBC Bottom view connectors.

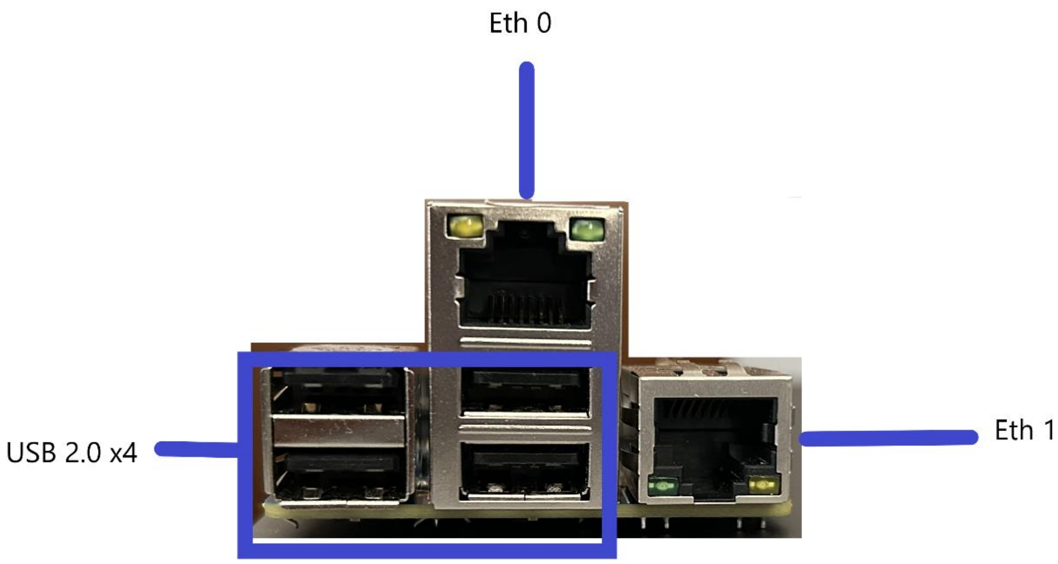

Figure 5. RZ/G2L-SBC side view I/O ports.

2.3.1.1 Power Supply

This section delves into the RZ/G2L-SBC's power supply architecture. The RZ/G2L-SBC uses a simple design with a 5V supply as the single external power source.

(1) USB TYPE-C POWER

This board has one USB Type-C receptacle for power input with USB chargers. The USB type–C power connector is meant to connect to a 5V power supply. The RZ/G2L-SBC requires a minimum of 3A power to prevent brownouts. However, we recommend a 4.5 -5A power supply as several ports support peripherals that consume substantial power.

2.3.1.2 Peripheral Interface

(1) 40-PIN I/O HEADER

The RZ/G2L-SBC comes with a 40-pin GPIO interface, which is broadly compliant with Raspberry Pi 3 40-pin GPIO interface and provides additional interfaces like two CAN ports. The diagram below shows the pin configuration along with the marking of the bottom I/O ports for reference to the orientation of the board.

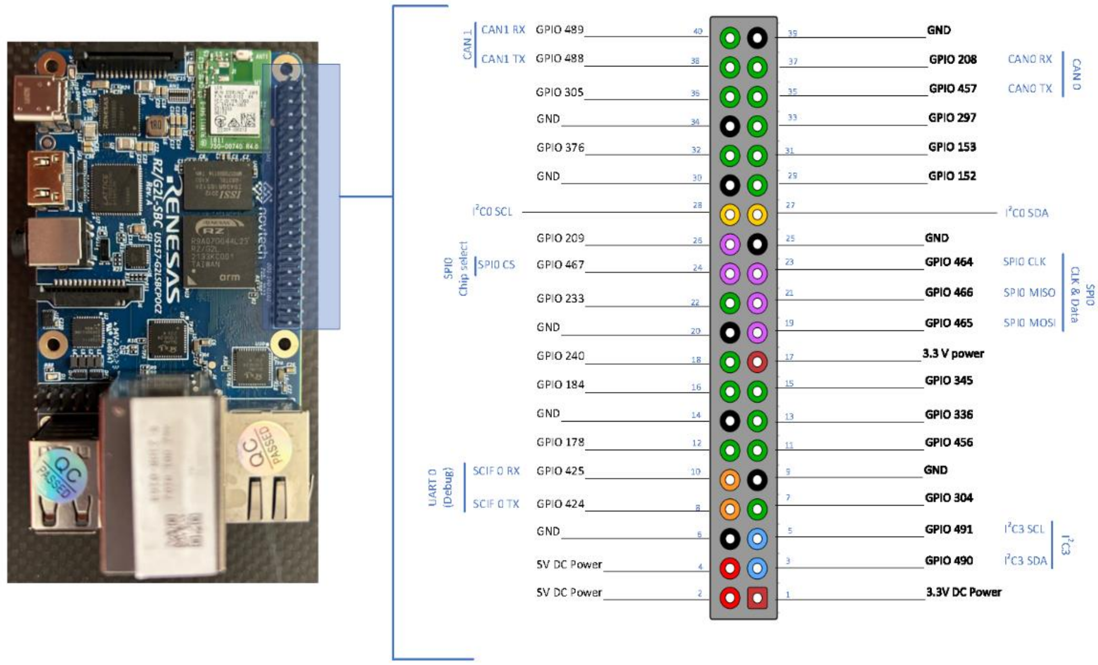

Figure 6. 40 PIN GPIO map with orientation details.

(2) PMOD TYPE 6A STANDARD INTERFACE

The RZ/G2L-SBC is equipped with a 2x6-pin header routed to the PMOD Type-6A interface, conforming to the 1.3.0 specification of PMOD. It includes the alternate pin functions from the specification.

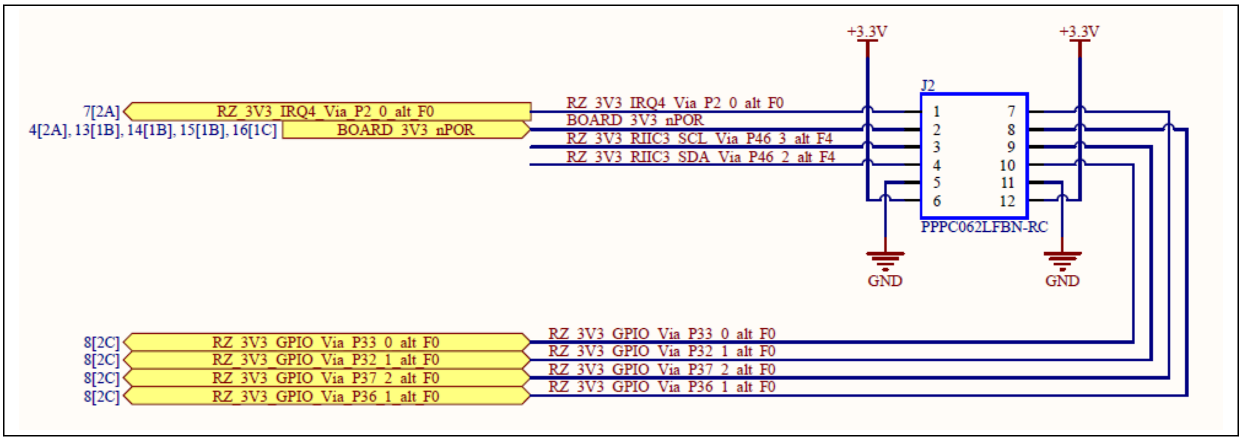

Figure 7. Schematic of PMOD Type 6 A pin header J2.

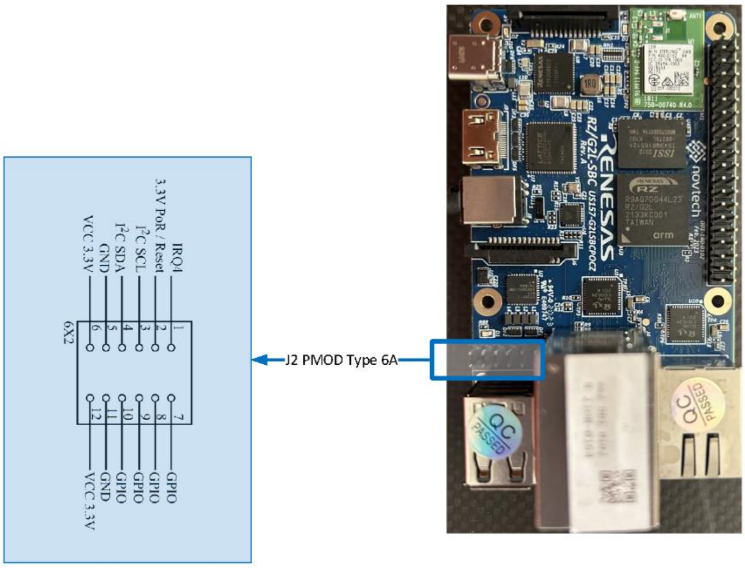

Figure 8. PMOD Type 6A 2x6 0.1mm pin out with orientation details.

(3) USD-CARD INTERFACE

The RZ/G2L-SBC comes with a spring-loaded micro-SD card slot. This is intended to be the primary storage as well as the OS boot device. The SD card is connected to channel 0 of the RZ/G2L SoC SD/MMC interface. The SoC SDIO interface is compliant with memory card standard version 3.0 and supports UHS-1 mode of 50 MB/s (SDR50) and 104 MB/s (SDR104).

2.3.2 Hardware Requirements

The basic hardware setup consists of the following:

1. RZ/G2L-SBC
2. FTDI RS232 UART cable
3. USB-C 5V 3A+ power supply
4. SD-MMC card (minimum 8 GB)
5. 1080p HDMI display/Waveshare 5” MIPI DSI display touch panel
6. Ethernet cables.
7. OV5640 MIPI CSI camera
8. USB keyboard and mouse
9. 3.5mm Headphone with microphone

2.3.3 Essential Hardware Setup

Figure 3. Essential minimum interfaces show the basic essential hardware setup. We expect a UART cable and an HDMI display to be available.

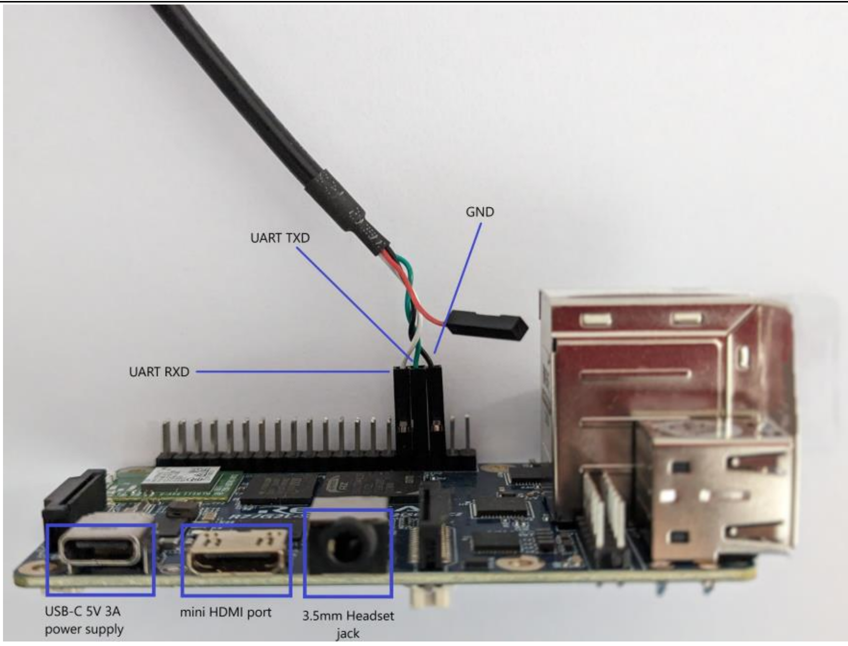

Figure 9. Essential minimum interfaces for RZ/G2L-SBC

2.3.4 Complete Hardware Setup

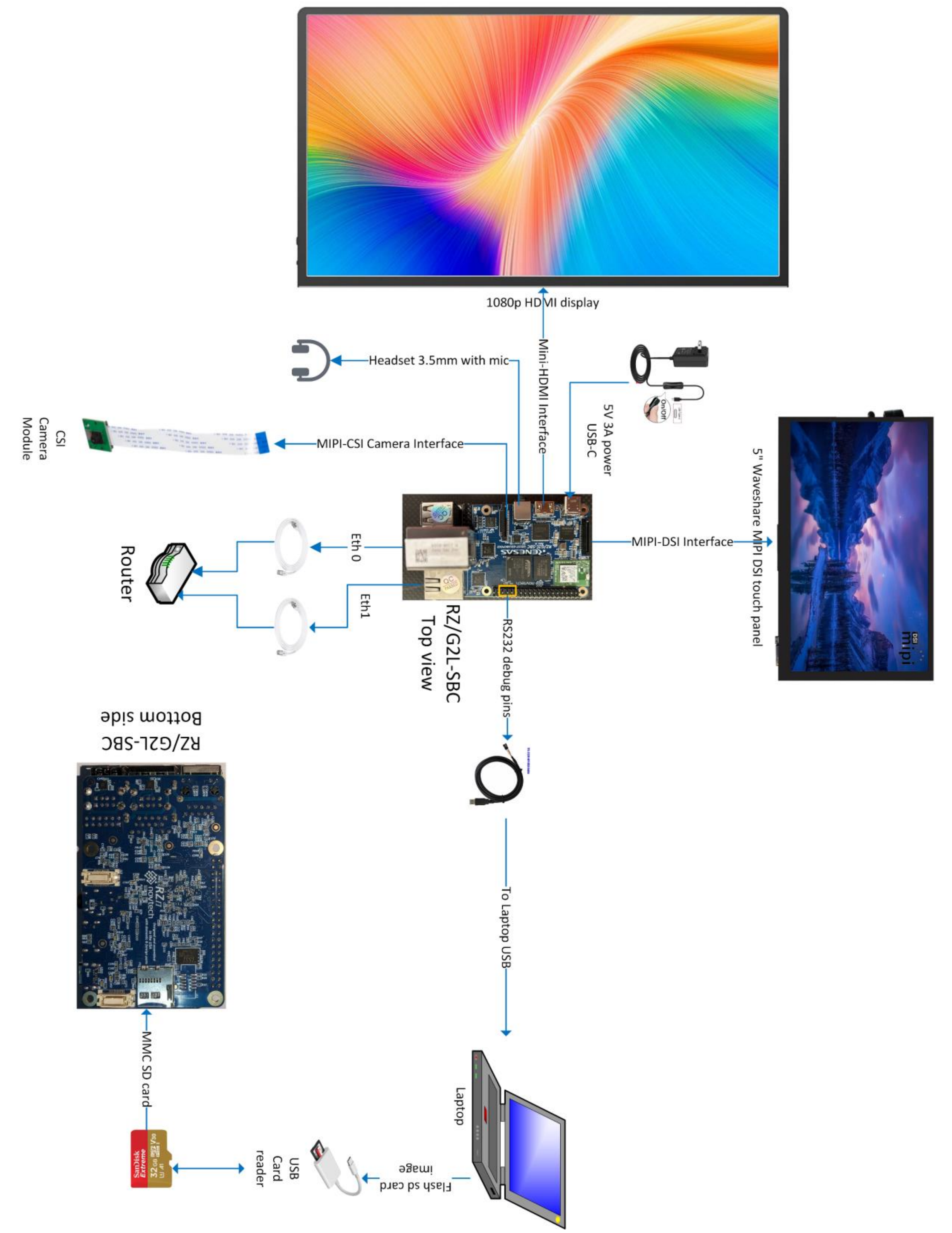

Figure 10. Complete setup for RZ/G2L-SBC board

2.3.5 Booting

The booting is straightforward.

1. Insert the MMC card into the MMC port on the bottom side of the RZ/G2L-SBC.
2. Connect the keyboard, mouse, and HDMI display; then insert the USB-C power supply and turn the power on.
3. You should see the boot log on the UART console and the Weston desktop on the HDMI screen.
4. Click on any of the applications and interact with them.

The image is fully featured and has powerful desktop-grade features. Read further to learn more about the features packed into the Linux image.

> Note: The default firmware shipped on the board may not recognize new images. If the board fails to boot, update the firmware using the Serial Download Mode (SCIF) procedure described in Section 3.1.1 RZ/G2L-SBC

2.3.6 Known Hardware and Functional Limitations on RZ/G2L-SBC

2.3.6.1 Linux (CA55) Side Known Issues

1. HDMI audio

- Status: Unverified
- Description: The functionality of the HDMI audio output has not been tested yet, and its behavior remains uncertain. Additional development and testing are required to assess its reliability and performance on the RZ/G2L-SBC.

2. Audio Sampling Rate Limitation

- Status: Currently limited to 48 kHz (validated)
- Description: The board’s clock design deviates from the standard RZ reference, which prevents the existing driver frameworks from generating the proper clocks when the SoC operates as I²S/TDM master. As a result, only 48 kHz sampling has been successfully validated so far. This is not a fundamental hardware restriction — codec-master mode has not yet been evaluated, and wider sampling-rate support may be possible but remains unsupported.

3. Onboard Bluetooth (BT) Functionality

- Status: Non-functional (onboard BT only)
- Description: The onboard Bluetooth functionality is currently non-operational due to a schematic symbol error in the Laird Wi-Fi/BT module. The Bluetooth interface is missing from the module’s schematic design, preventing Bluetooth connectivity. However, USB Bluetooth functionality remains operational. This issue requires a hardware revision to enable full Bluetooth functionality on the onboard module.

2.3.6.2 FreeRTOS/FSP (CM33) Side Known Issues

1. MIPI-CSI2 Camera and Peripherals Accessing Shared I2C1 Bus

In the RZ/G2L-SBC, the MIPI CSI Camera interface, HDMI Bridge, and MIPI DSI all share the same I2C1 channel. Due to this hardware constraint, controlling one of these devices may impact the functionality of the others.

Limitations:

- I2C1 can only be accessed by one core at a time, which can prevent both the camera and display from functioning simultaneously.
- Any device using I2C1 must be managed carefully to avoid conflicts with other peripherals.
- This limitation should be considered when designing the system to ensure both peripherals can operate as required.

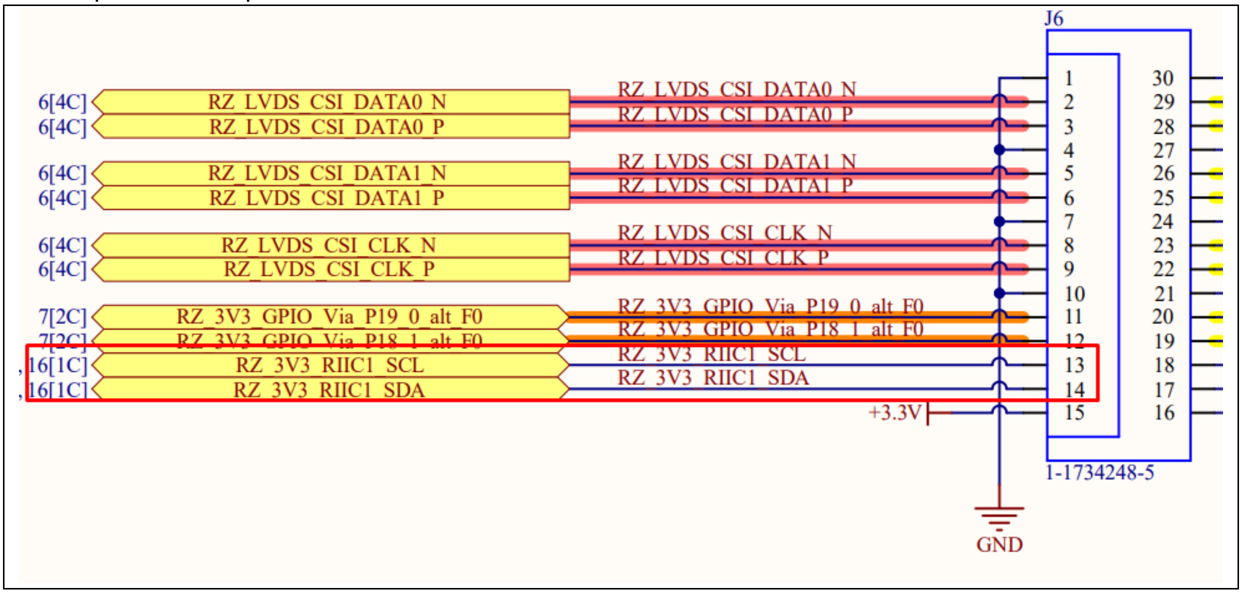

Figure 11. CSI using shared I2C1 bus

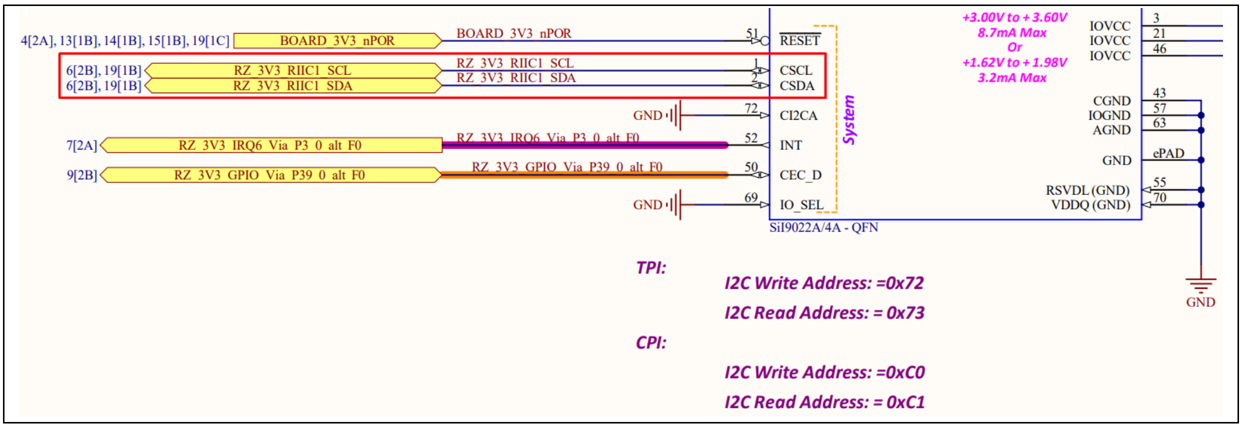

Figure 12. HDMI using shared I2C1 bus

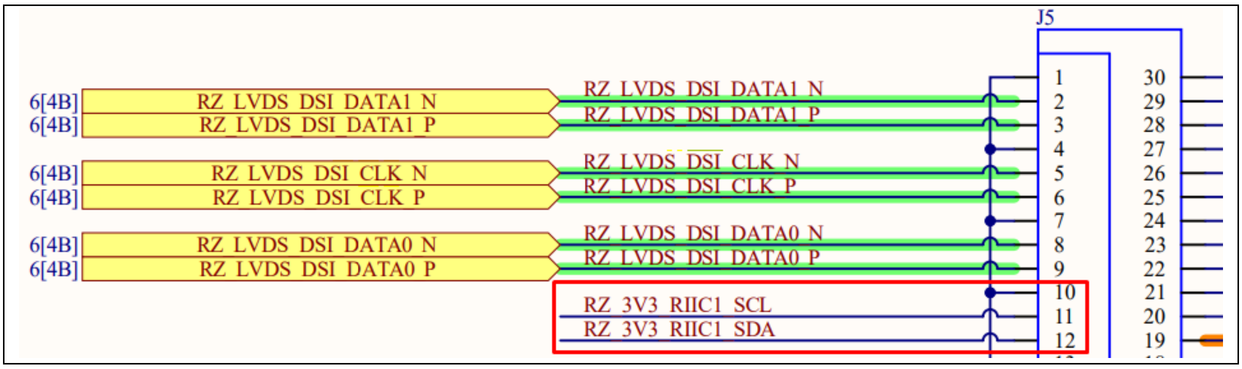

Figure 13. DSI using shared I2C1 bus

As shown in the three figures above, the shared I2C1 bus is used by multiple peripherals, which may lead to conflicts if both cores are used simultaneously. To avoid issues, users should ensure that only one core accesses I2C1 at a time or consider alternative methods for managing communication between peripherals.

2. Limited SCIF Availability for Multi-Core Development

However, a limitation exists in the number of available SCIF (Serial Communication Interface with FIFO) channels, which impacts debugging and logging functionality for multi-core development.

Limitations:

- Single SCIF Channel: Only SCIF0 is available for serial communication, and it is exclusively allocated to the CA55 core.
- Restricted logging for CM33: Since SCIF0 is dedicated to CA55, the CM33 core lacks direct access to an SCIF channel, making it challenging to perform independent serial logging or debugging.

This limitation should be considered when designing multi-core applications, especially those requiring real-time logging, debugging, or inter-core communications.

2.4 RZ/G2L-EVK and RZ/V2L-EVK

Both RZ/G2L-EVK and RZ/V2L-EVK platforms feature robust Linux images with desktop and display support. Their hardware setup and booting procedures are nearly identical, allowing a common approach for setup and evaluation.

Note:

- The release consists of images that have desktop and display support.
- At least one basic display, like a 1080p HDMI monitor, must be available for those images.

2.4.1 Hardware Requirements

The basic hardware setup consists of the following:

1. RZ/G2L-EVK or RZ/V2L-EVK
2. USB Type-A to Micro USB Type-B cable
3. USB-C 5V 3A+ power supply
4. SD-MMC card (minimum 8 GB)
5. HDMI display
6. Ethernet cables
7. OV5645 camera module (optional: not included in the hardware package)
8. USB keyboard and mouse
9. 3.5mm Audio Stereo Y Splitter extension cable (optional: not included in the hardware package)
10. 3.5mm Headphone with microphone

2.4.2 Essential Hardware Setup and Booting

The figure below shows the basic essential hardware setup for RZ/G2L-EVK & RZ/V2L-EVK. Follow these steps to prepare and power on the board:

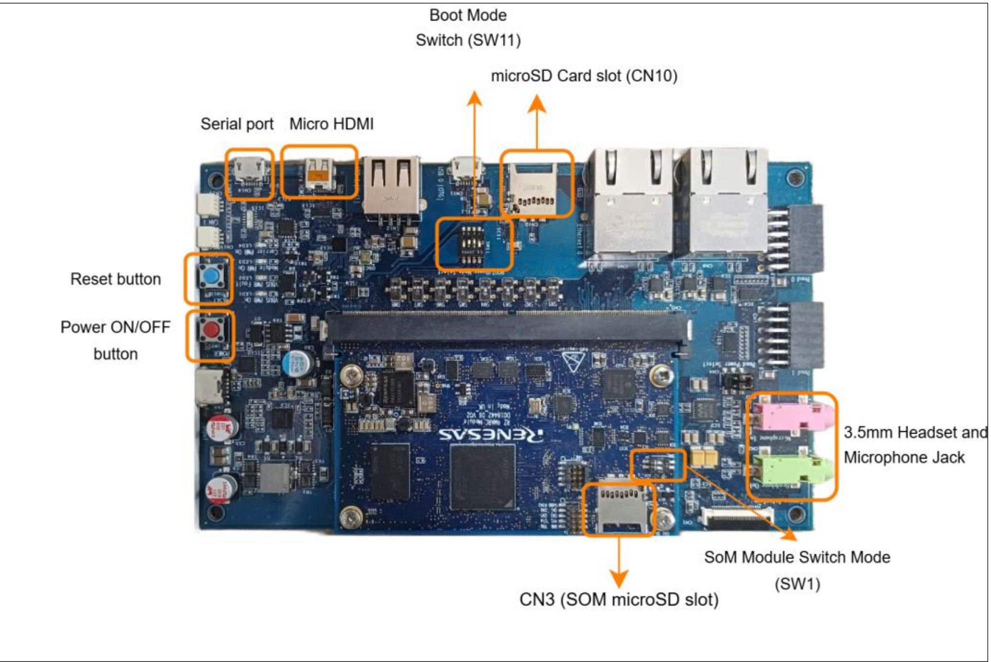

Figure 14. Essential minimum interfaces for RZ/G2L-EVK & RZ/V2L-EVK

Note:

- Boot source selection
  - In this release, all boot flows use QSPI or eMMC as selected by SW11.
  - eSD boot from CN3 is not supported.
- CN10 (Carrier microSD slot)
  - CN10 is not a boot device.
  - CN10 can be used only for rootfs or data storage after the system has booted.
  - No SW1 configuration is required for CN10.
- CN3 (SoM microSD slot)
  - BL2/FIP cannot be loaded from CN3 (eSD boot not supported).
  - When the board boots from QSPI or eMMC, CN3 is accessible in U-Boot/Linux (commonly as mmc device 0).
  - CN3 may be used as a rootfs or additional storage device if enabled via SW1-2.

1. **Prepare the microSD card**

   Flash the provided image to the microSD card using a user-friendly tool such as Balena Etcher as describe in Section 2.1. SD-MMC Card Flashing for a simplified flashing experience.

2. **Insert the microSD card**

   Insert the prepared SD card into CN10. This slot is for rootfs/data only; it cannot be used to load BL2/FIP.

3. **Configure the boot mode**

   For this release, the default boot method is QSPI boot. This configuration loads BL2 and FIP from QSPI flash, with the root filesystem provided from eMMC or the SD card in slot CN10.

   Default setup (QSPI boot):

   - SW11 (on the carrier board): Set to QSPI boot.

- SW1-2 (on the SoM): Controls access to storage devices on the SOM (Optional)
  - OFF: Enables access to the onboard eMMC
  - ON: Enables access to the microSD slot CN3
- Root filesystem: Use the SD card in slot CN10 (carrier board). CN10 remains accessible regardless of SW1-2.

Alternative setup (eMMC boot):

- SW11 (on the carrier board): Set to eMMC boot
- SW1-2 (on the SoM): Same as above for QSPI boot.

Refer to Section 3.2 – Boot Mode Reference (RZ/G2L & RZ/V2L) for the correct switch positions. In these modes, the board fetches BL2 and FIP from the chosen device.

4. **Connect peripherals**

- Attach an HDMI display

5. **Connect UART**

- Use a USB Type-A to Micro-USB Type-B cable to connect the board's UART console port to the host PC.
- Open a terminal program on the host PC to monitor the boot log.

6. **Power on the board**

- Connect the USB-C power supply (5 V, 3 A).
- Press the red power button to turn on the board.

Once powered on, the boot log should appear on the UART console, and the Weston desktop will display on the HDMI screen.

2.4.3 Complete Hardware Setup

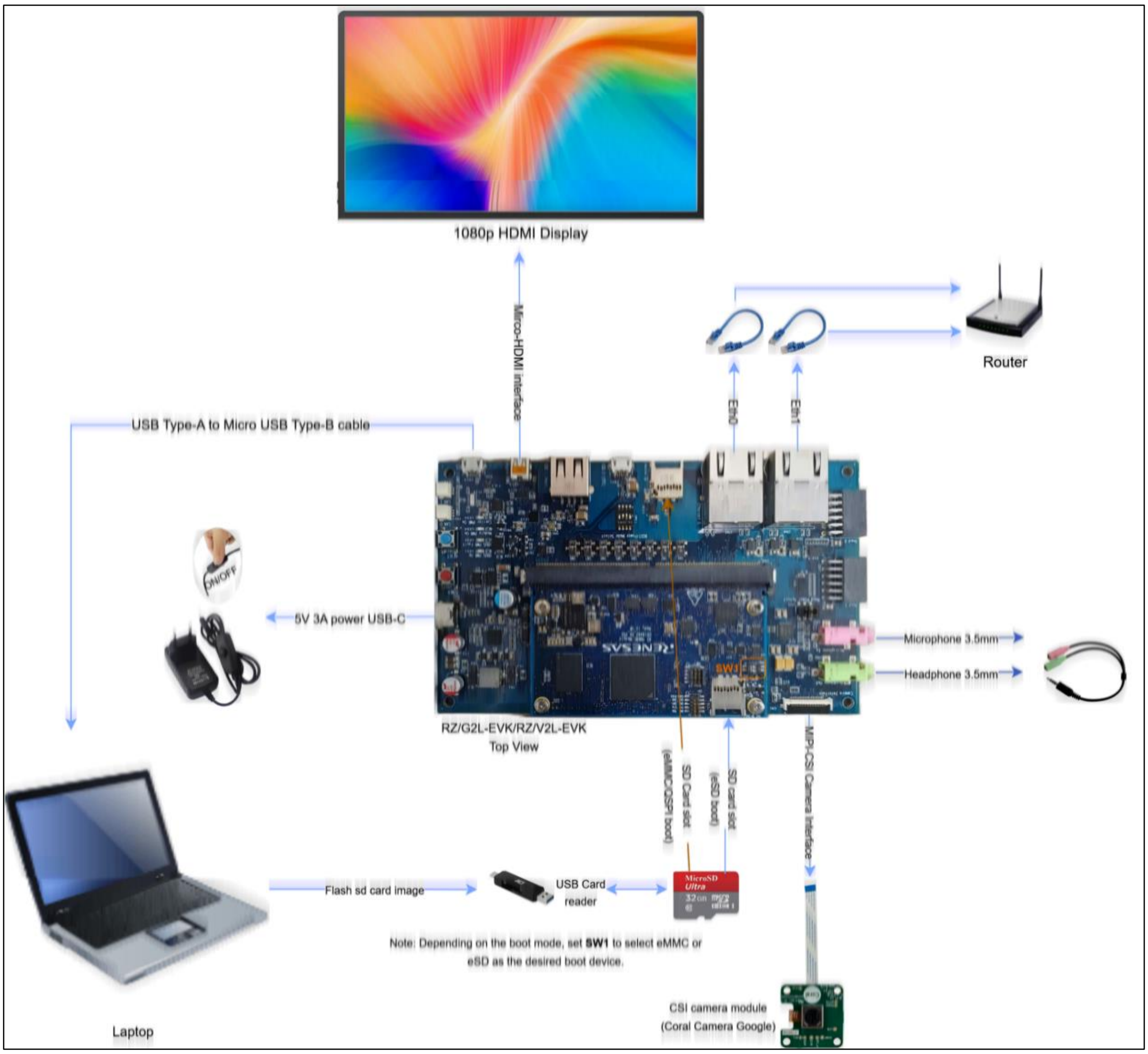

Figure 15. Complete setup for RZ/G2L-EVK & RZ/V2L-EVK

2.4.4 Booting

The boot process is straightforward:

1. Insert the prepared SD card into CN10.
2. Connect a keyboard, mouse, HDMI display, and other devices. Then connect the USB-C power supply and press the red power button to turn the board on.
3. The boot log should appear on the UART console, and the Weston desktop should display on the HDMI screen.
4. Open any of the available applications to interact with the system.

The image provided is feature-complete, offering a desktop-grade experience.

If a different boot device is required such as QSPI, eMMC, or SD card — adjust the DIP switch settings as described in Section 3.2. Boot Mode Reference (RZ/G2L-EVK & RZ/V2L-EVK) before powering on. The selected boot device determines where on-chip ROM code loads the Boot Loader stage 2 (BL2) and Firmware Image Package (FIP):

- QSPI boot loads BL2 and FIP from the onboard QSPI flash.
- eMMC boot loads BL2 and FIP from the onboard eMMC device.
- eSD boot is not supported in this release.

> Note: The default firmware shipped on the board may not recognize new images. If the board fails to boot, update the firmware using the Serial Download Mode (SCIF) procedure described in Section 3.1.2 RZ/G2L-EVK and RZ/V2L-EVK

2.4.4.1 Default Boot Behaviour

When booting from QSPI or eMMC, the U-Boot environment defaults to using the SD card (CN10) as the root filesystem.

- The SD card on CN10 is always available as mmc1.
- The onboard eMMC is always available as mmc0.
- Unless reconfigured, Linux will mount the rootfs from CN10 (e.g., /dev/mmcblk1p1).

2.4.4.2 Using an eMMC Root Filesystem

To use the onboard eMMC as the root filesystem instead of CN10:

1. Enable eMMC on the SoM: Set SW1-2 = OFF (see Section 3.1 Boot Mode Reference (Non-SCIF) RZ/G2L-EVK & RZ/V2L-EVK for SW1 description).
2. Prepare the eMMC rootfs. See Section 3.3 Prepare the eMMC root filesystem for detailed information
3. Reboot the board. During startup, output similar to the following will appear:

```
NOTICE:  BL2: v2.9(release):<release-tag>
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.9(release): <release-tag>

U-Boot 2021.10 <Date>

CPU:   Renesas Electronics CPU rev 1.0
Model: <board-name>
DRAM:  1.9 GiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from SPIFlash... SF: Detected mt25qu512a ...
*** Warning - bad CRC, using default environment

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

   - When the message “Hit any key to stop autoboot” appears, press Enter (or any key) to interrupt the countdown.
   - The => prompt indicates that are in the U-Boot console.
4. Now update the environment by changing the default device from 1 (SD – CN10) to 0 (eMMC) and then boot.

```
=> setenv mmcdev 0
=> saveenv
=> boot
```

2.4.4.3 Using CN3 (eSD slot) as Root Filesystem

eSD boot from CN3 is not supported in this release, meaning BL2/FIP cannot be loaded directly from this slot. However, once the board boots from QSPI, CN3 remains accessible in U-Boot and Linux as a standard MMC device. This allows CN3 to be used for the root filesystem or as additional data storage.

Steps to use CN3 as the root filesystem:

1. Insert the SD card into the CN3 slot
2. Enable eSD on the SoM: Set SW1-2 = ON (see Section 3.1 Boot Mode Reference (Non-SCIF) RZ/G2L-EVK & RZ/V2L-EVK for SW1 description).
3. Reboot the board. During startup, output similar to the following will appear:

```
NOTICE:  BL2: v2.9(release):<release-tag>
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.9(release): <release-tag>

U-Boot 2021.10 <Date>

CPU:   Renesas Electronics CPU rev 1.0
Model: <board-name>
DRAM:  1.9 GiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from SPIFlash... SF: Detected mt25qu512a ...
*** Warning - bad CRC, using default environment

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

   - When the message “Hit any key to stop autoboot” appears, press Enter (or any key) to interrupt the countdown.
   - The => prompt indicates that the system is now in the U-Boot console.
4. Update the U-Boot environment by changing the default device from 1 (CN10) to 0 (CN3):

```
=> setenv mmcdev 0
=> saveenv
=> boot
```

2.4.5 Known Hardware and Functional Limitation

1. Unstable UDP flashing

   UDP-based flashing on RZG2L-EVK and RZV2L-EVK is not stable, likely due to limitations in the on-board Ethernet interface. Use balenaEtcher or the dd command to flash root filesystems instead of relying on UDP mode.

2. No onboard Wi-Fi/Bluetooth

   These EVKs do not include an on-chip or onboard Wi-Fi/Bluetooth module. Wireless connectivity must be provided through external modules (e.g., USB or SDIO-based Wi-Fi dongles).

3. USB OTG is not supported in this release
4. eSD boot is not supported in this release.

2.5 RZ/V2H-EVK

This section describes the hardware-specific setup and booting process for the RZ/V2H-EVK

2.5.1 Overview of Connectors

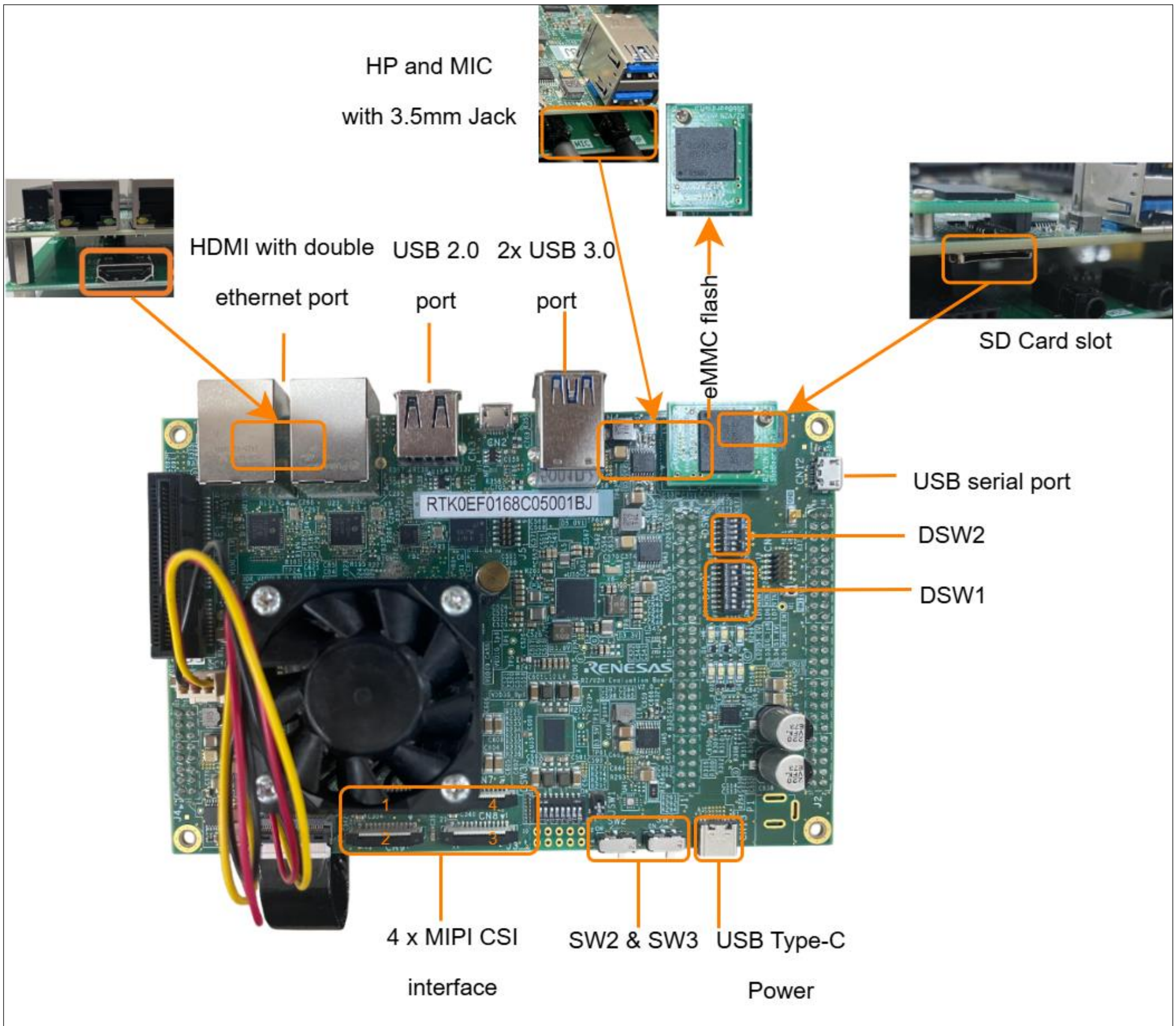

Figure 16: RZ/V2H-EVK Overview of Connectors

2.5.2 Hardware Requirements

The basic hardware setup consists of the following:

1. RZ/V2H-EVK
2. USB Type-A to Micro USB Type-B cable
3. Power supply that can provide up to 100W via USB-C PD (not included in the package).
4. SD-MMC card (minimum 8 GB)
5. 1080p HDMI display
6. Ethernet cables.
7. OV5645 camera module (optional: not included in the hardware package).
8. Camera conversion 22-pin to 25-pin FPC adapter
9. USB keyboard and mouse

10. 3.5mm Audio Stereo Y Splitter extension cable (optional: not included in the hardware package)
11. 3.5mm Headphone with microphone

2.5.3 Essential Hardware Setup and Booting

The figure below shows the basic essential hardware setup for RZ/V2H-EVK. Follow these steps to prepare and power on the board:

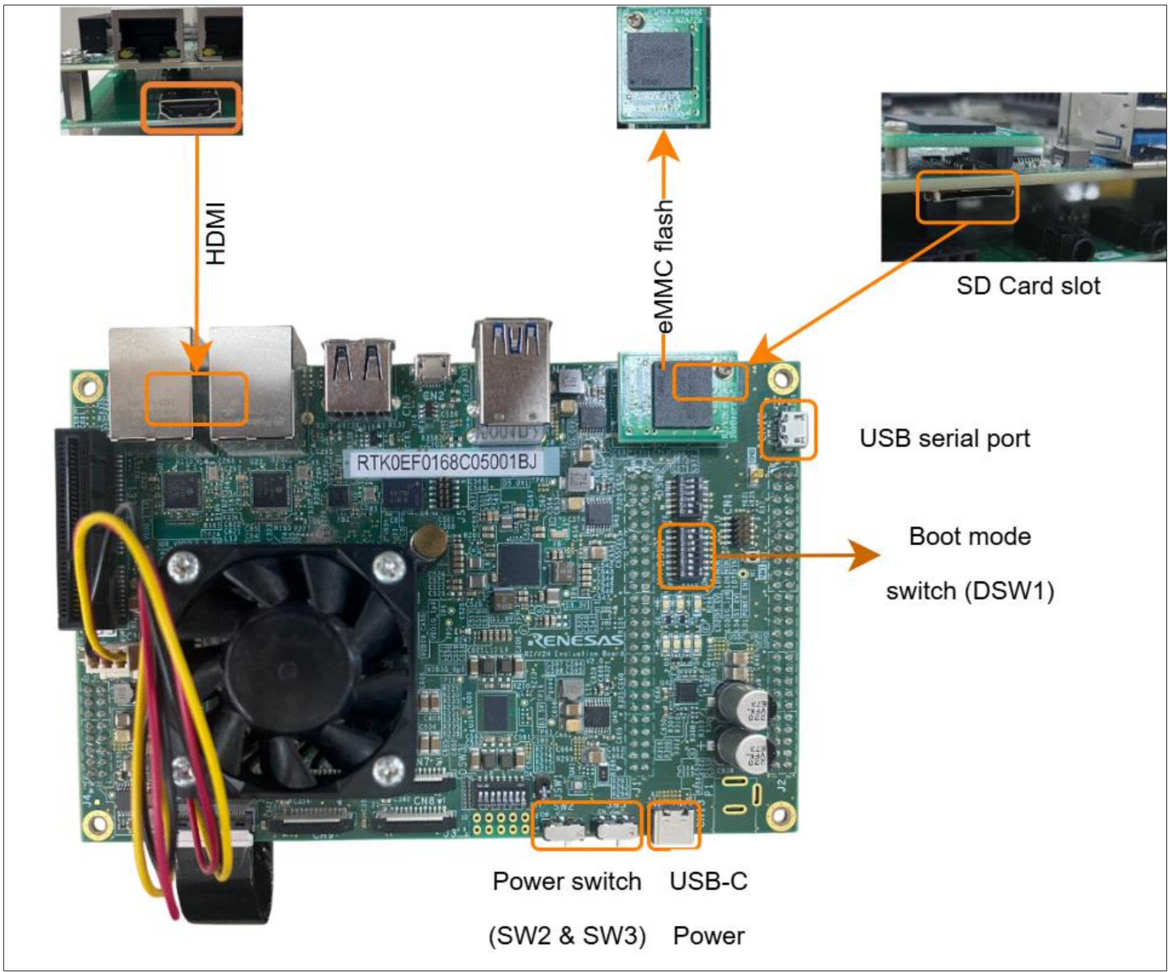

Figure 17: Essential minimum interfaces for RZ/V2H-EVK

1. **Prepare the microSD card**

   Flash the provided image to the microSD card using a user-friendly tool such as Balena Etcher as described in Section 2.1. SD-MMC Card Flashing for a simplified flashing experience.

   Insert the prepared microSD card into the SD card slot on the underside of the board.

2. **Configure the boot mode**

   Set the DIP switches (DSW1) to select the boot source. Section 3.2 – Boot Mode Reference (RZ/V2H-EVK) for the correct switch positions.

   For this release, configure DSW1 to boot from QSPI flash:

   - BL2 and FIP are loaded from QSPI.
   - Rootfs can be located on eMMC or SD card, as defined in the U-Boot environment.

Note: Booting from eMMC and eSD is currently not supported in this release. These options may be enabled in future updates.

3. **Connect peripherals**
    - Attach the HDMI display, USB keyboard, and USB mouse.
4. **Connect UART**
    - Use a USB Type-A to Micro-USB Type-B cable to connect the board's UART console port to the host PC.
    - Open a terminal program on the host PC to monitor the boot log.
5. **Power on the board**
    - Connect the USB-C power supply (100W).
    - Turn on SW2 and SW3

Once powered on, the boot log should appear on the UART console, and the Weston desktop will display on the HDMI screen.

2.5.4 Complete Hardware Setup

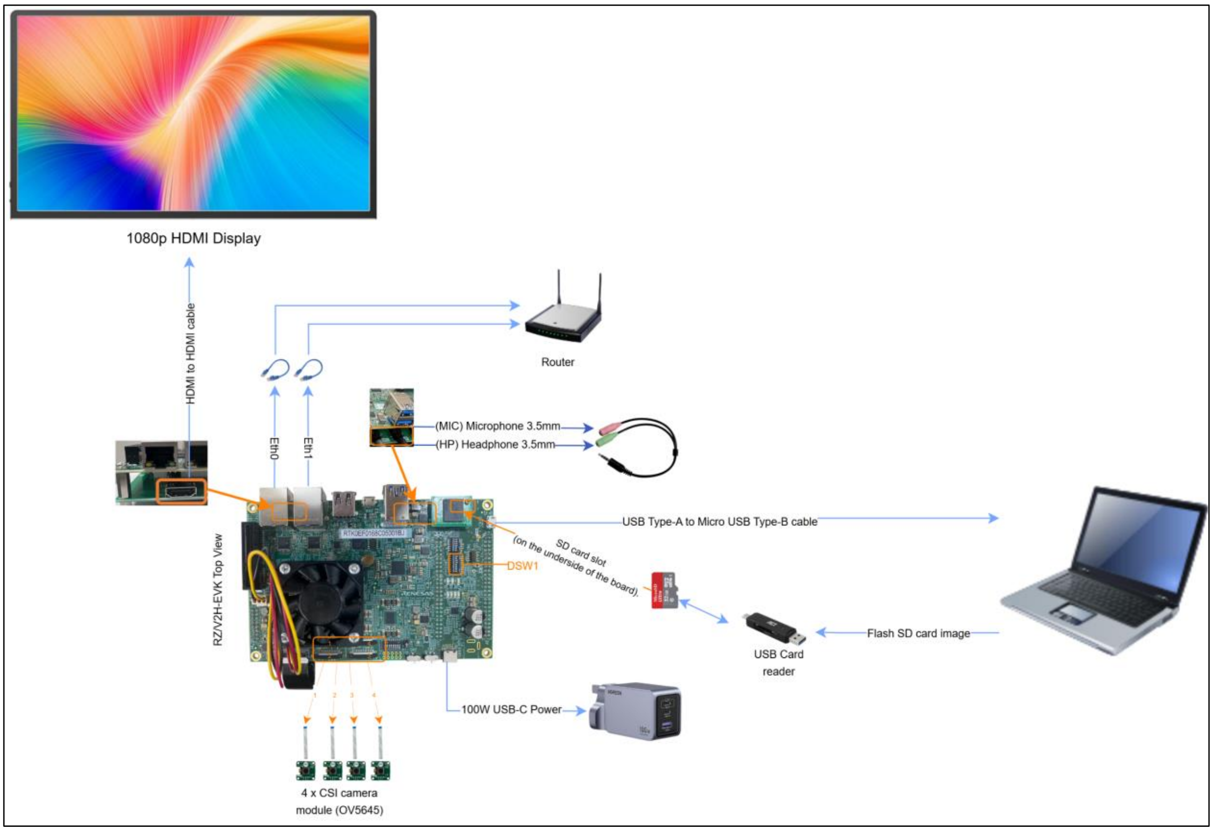

Figure 18: Complete setup for RZ/V2H-EVK board

Note: The RZ/V2H-EVK is available in two hardware versions, which differ in their storage options:

- **Version 1**: Equipped with two SD card slots and no onboard eMMC.
- **Version 2**: Equipped with one onboard eMMC (default boot device) and one SD card slot for alternate booting or data storage.

The complete setup shown above applies to Version 2. For Version 1, the setup procedure is the same as Version 2, with the only difference being the use of two SD card slots instead of one (and no onboard eMMC).

2.5.5 Booting

The boot process for RZ/V2H-EVK is very similar to RZ/G2L-EVK and RZ/V2L-EVK, as the boot mode must be selected using the DIP switches before power-on. Follow the steps below to boot the board:

1. Insert the SD card into the SD card slot (located on the underside of the carrier board; see the complete setup picture in the previous section).
2. Connect the keyboard, mouse, and HDMI display. Then connect the 100 W USB-C power supply and toggle two power switches next to the USB-C port (SW2 and SW3)
3. The boot log will appear on the UART console, and the Weston desktop will display on the HDMI screen.
4. Launch any of the available applications to interact with the system.

The provided image is feature-complete and offers desktop-grade user experience.

As with other RZ/G2L-EVK & RZ/V2L-EVK, the RZ/V2H-EVK requires selecting a boot device — such as xSPI, eMMC, or SD Card before powering on. Adjust the DIP switch settings as described in Section 3.2. Boot Mode Reference (RZ/V2H-EVK).

Note: The default firmware shipped on the board may not recognize new images. If the board fails to boot, update the firmware using the Serial Download Mode (SCIF) procedure described in Section 3.1.3 RZ/V2H-EVK

2.5.5.1 Default Boot Behaviour

When booting from QSPI or eMMC, the U-Boot environment defaults to using the SD card as the root filesystem.

- The SD card is always available as mmc0.
- The onboard eMMC is always available as mmc1.
- Unless reconfigured, Linux will mount the rootfs from SD card slot (e.g., /dev/mmcblk0p1).

2.5.5.2 Using an eMMC Root Filesystem

To use the onboard eMMC as the root filesystem instead of CN10:

1. Enable eMMC by setting DSW1 to eMMC boot mode (see Section 3.1 Boot Mode Reference (Non-SCIF) RZ/V2H-EVK for details)
2. Prepare the eMMC rootfs. Refer to Section 3.3 Prepare the eMMC root filesystem
3. Reboot the board. During startup, output similar to the following will appear:

```
NOTICE:  BL2: v2.9(release):<release-tag>
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.9(release): <release-tag>

U-Boot 2021.10 <Date>

CPU:   Renesas Electronics CPU rev 1.0
Model: <board-name>
DRAM:  1.9 GiB
MMC:   sd@11c00000: 0, sd@11c10000: 1
Loading Environment from SPIFlash... SF: Detected mt25qu512a ...
*** Warning - bad CRC, using default environment

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

- When the message “Hit any key to stop autoboot” appears, press Enter (or any key) to interrupt the countdown.
- The => prompt indicates that are in the U-Boot console.

4. Now update the environment by changing the default device from 0 (SD) to 1 (eMMC) and then boot.

```
=> setenv mmcdev 0
=> saveenv
=> boot
```

2.5.6 Known Hardware and Functional Limitations

1. No on-board Wi-Fi/Bluetooth:

   These EVKs do not include an on-chip or onboard Wi-Fi/Bluetooth module. Wireless connectivity must be provided through external modules (e.g., USB or SDIO-based Wi-Fi dongles).
2. USB OTG is not supported in this release
3. eSD boot is not supported in this release.
4. eMMC boot is not supported in this release.

   The RZ/V2H board exists in two hardware versions, one of which does not include eMMC. As a result, eMMC flash support was not provided

3. Appendix

3.1 Factory Firmware Flashing Using Serial Downloader (SCIF) Mode

In most cases, the RZ boards come preloaded with the latest firmware. The preferred method of updating the firmware is through the SD card flashing method, as described in section 2.1. SD-MMC Card Flashing.

However, there are cases where you might require the use of a serial downloader. This is more common in a factory environment where the boards are being programmed for the first time or in cases where the board is bricked.

This is considered hardware flashing because it requires the board to be put into the serial download mode (called SCIF mode), by altering the bootstrapping pins.

3.1.1 RZ/G2L-SBC

This section describes the firmware flashing process for the RZ/G2L-SBC board.

Note: The RZ/G2L-SBC does not have any interfaces on the main board to alter the boot mode. The bootstrapping pins are routed through the bottom connectors J12 & J13. Hence, the process requires the use of an adapter board, which is not included in the package.

3.1.1.1 Required Hardware

This flashing process requires the use of boot mode change, which is achieved using an adapter board which is not included in the package.

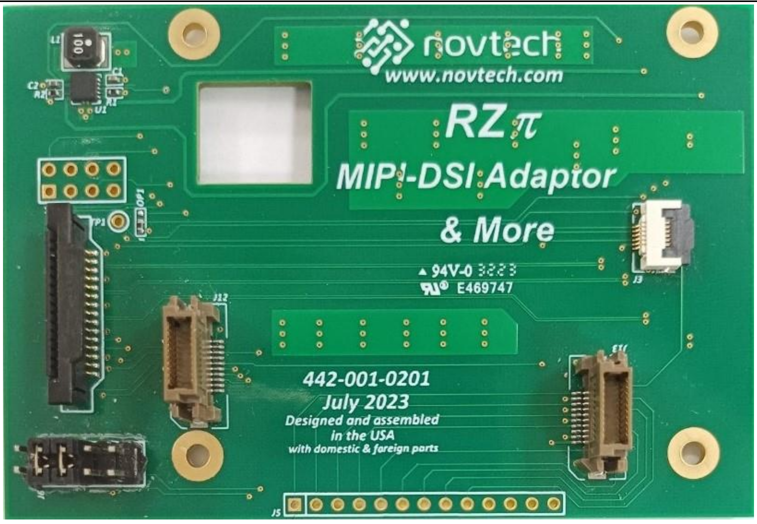

Figure 19: Adaptor board

After setting up the required hardware and configuring the boot mode through connectors J12 and J13 on the RZ/G2L-SBC, refer to Section 2.2.2. Usage and Flashing Operations for instructions on using the universal script to flash the firmware

3.1.2 RZ/G2L-EVK and RZ/V2L-EVK

This section describes the firmware flashing process and boot mode configuration for each supported board. Firmware flashing is required to write bootloaders (BL2 and FIP) to the onboard flash memory. The process uses Renesas' Flash Writer tool and requires setting the board into SCIF Download Mode.

Unlike the RZ/G2L-SBC, which requires external hardware to configure boot mode, the RZ/G2L-EVK and RZ/V2L-EVK feature onboard DIP switches that allow direct selection of boot modes. This simplifies the flashing process and eliminates the need for manual signal strapping.

3.1.2.1 DIP Switch Settings

Use the DIP switch SW11 to configure the boot mode:

Table 7: SCIF Download Mode - RZ/G2L-EVK & RZ/V2L-EVK

Switch	SCIF Download Mode
SW11-1	OFF
SW11-2	ON
SW11-3	OFF
SW11-4	ON

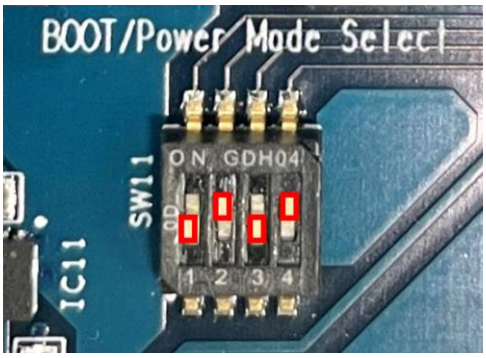

Figure 20: SW11 SCIF Download Mode - RZ/G2L-EVK & RZ/V2L-EVK

Use the DIP switch1 to select eMMC as boot device:

Table 8: Select eMMC as boot device

Switch	Select eMMC
SW1-1	ON
SW1-2	OFF

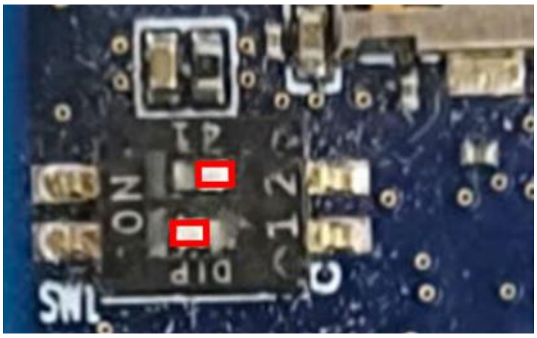

Figure 21: SW1 Settings for eMMC Boot - RZ/G2L-EVK & RZ/V2L-EVK

After setting up the required hardware and configuring the SCIF download mode for the RZ/G2L-EVK or RZ/V2L-EVK, refer to Section 2.2.2. Usage and Flashing Operations for instructions on using the universal script to flash the firmware

3.1.3 RZ/V2H-EVK

Use the DIP switch DSW1 to configure the boot mode:

Table 9: SCIF Download Mode - RZ/V2H-EVK

Switch	Status	Function
DSW1-1	ON	Select the cold boot CPU - OFF: CM33 - ON: CA55 (default)
DSW1-2	OFF	Input the CA55 frequency at the CA55 cold boot - [OFF: OFF]: 1.6GHz - [OFF: ON]: 1.7GHz (default) - [ON: OFF]: 1.1GHz - [ON: ON]: 1.5GHz
DSW1-3	ON	
DSW1-4	OFF	- [OFF: OFF]: xSPI - [OFF: ON]: SCIF - [ON: OFF]: SD (default) - [ON: ON]: eMMC
DSW1-5	ON	
DSW1-6	OFF	OFF: SSCG ON (default), ON: SSCG OFF
DSW1-6	OFF	OFF: Normal mode, ON: Debug mode
DSW1-7	OFF	Fixed to OFF

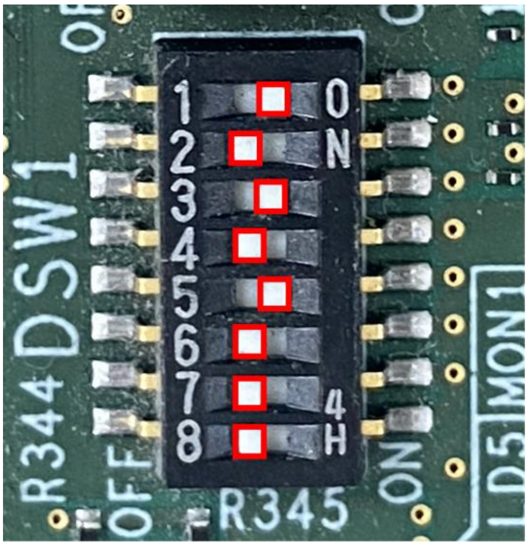

Figure 22. DSW1 - SCIF Boot Mode

To enable SCIF Download Mode, set DSW1-4 and DSW1-5 according to the SCIF configuration in the Table 9: SCIF Download Mode - RZ/V2H-EVK

Other switches (DSW1-1, DSW1-2, DSW1-3, DSW1-6, and DSW1-7) should remain in their default positions unless you need to change CPU selection, boot frequency, SSCG, or debug mode as described in the table.

After setting up the required hardware and configuring the SCIF download mode for the RZ/V2H-EVK, refer to Section 2.2.2. Usage and Flashing Operations for instructions on using the universal script to flash the firmware

3.2 Boot Mode Reference (Non-SCIF)

This section summarizes the switch/strap settings required for normal boot and boot-device selection across supported boards. Use these settings after completing factory flashing or when switching the boot device during bring-up.

3.2.1 RZ/G2L-EVK & RZ/V2L-EVK

These EVKs provide on-board DIP switches for boot mode and boot device selection.

The settings below tell the BootROM which device to read the initial firmware from, i.e., where to fetch BL2 (and Boot Parameter, if used) and subsequently the FIP.

Use these settings after factory flashing to boot from the programmed device.

Table 10: SW11 – Boot Device Selection (Normal Boot)

Boot device	SW11-1	SW11-2	SW11-3	SW11-4	Description
eMMC	ON	OFF	OFF	ON	Boot from on-board eMMC (BootROM loads BL2/BL2+BP from eMMC, then FIP).
QSPI	OFF	OFF	OFF	ON	Boot from QSPI NOR flash
SD / eSD	ON	ON	OFF	ON	Boot from SD/eSD card (slot media)

Figure 23: SW11 - eSD Boot Mode

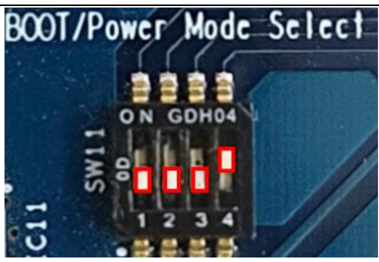

Figure 24. SW11 – QSPI Boot Mode

Figure 25. SW11 – eMMC Boot Mode

On the SOM module, SW1 selects eMMC or microSD boot mode. Please refer to the table below for the boot-mode options for each switch setting.

Table 11: SW1 – SOM module Switch mode

Boot device	ON	OFF
SW1-1	Normal Operation	JTAG debug mode
SW1-2	Select microSD slot on RTK9744L23C01000BE	OFF Select eMMC on RTK9744L23C01000BE

3.2.2 RZ/V2H-EVK

The RZ/V2H-EVK also provide on-board DIP switches for boot mode and boot device selection.

Use these settings after factory flashing to boot from the programmed device.

Table 12: DSW1 – Boot Device Selection (Normal Boot)

Boot device	DSW1-1	DSW1-2	DSW1-3	DSW1-4	DSW1-5	DSW1-6	DSW1-7	DSW1-8
eMMC	ON	OFF	ON	ON	ON	OFF	OFF	OFF
xSPI	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF
SD / eSD	ON	OFF	ON	ON	OFF	OFF	OFF	OFF

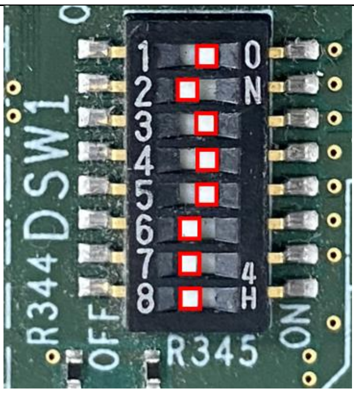

Figure 26. DSW1 - eMMC Boot Mode

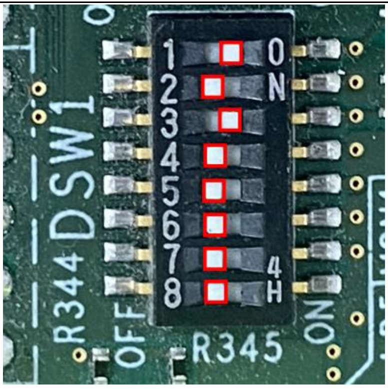

Figure 27. DSW1 – xSPI Boot Mode

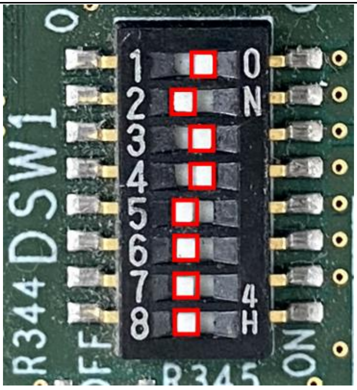

Figure 28. DSW1 – eSD Boot Mode

DSW2 controls output from the on-board clock generator (5P35023B) and one protected utility signal.

Table 13: DSW2 - Audio Clock / Utility DIP

Switch	Signal	OFF (default)	ON
1	Audio_CLKB_OE	Disables 5P35023B Audio_CLKB output	Enables Audio_CLKB output
2	Audio_CLKB	Audio_CLKB not supplied	Audio_CLKB is driven
3	Audio_CLKC_OE	Disables 5P35023B Audio_CLKC output	Enables Audio_CLKC output
4	Audio_CLKC	Audio_CLKC not supplied	Audio_CLKC is driven
5	NEN_VPROG	Must remain OFF	Prohibited — do not set ON
6	—	—	—

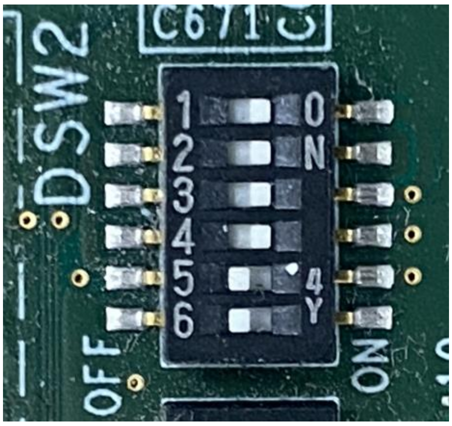

Figure 29. DSW2 - Audio Clock / Utility DIP

The table below lists the settings of the DIP switch (JSW1 on the RZ/V2H Secure Evaluation Board) and its functions.

Table 14: JSW1 Functions

Switch	Function
1-2	MIPI CSI-2 camera interface voltage: 1.8 V
2-3	MIPI CSI-2 camera interface voltage: 3.3 V (default)

Note: Set this switch according to the interface voltage of the camera module to be connected.

3.3 Prepare the eMMC root filesystem

The onboard eMMC can be used as the main root filesystem, but it must first be initialized with a valid Linux rootfs. Since the eMMC is initially empty or unformatted, it cannot be used directly.

During preparation, the system must boot Linux using an SD card rootfs, while BL2/FIP are loaded from either QSPI or eMMC depending on the boot mode. Once Linux is running, the onboard eMMC (e.g. /dev/mmcblk0) becomes accessible and can be partitioned, formatted, and written with the rootfs.

Step 1. Prepare the rootfs archive

- Obtain the provided root filesystem archive (e.g core-image-weston.tar.bz2). Obtain the provided root filesystem archive, which is included in the release images under:

```
rz-cmn-srp/target/images/rootfs
```

- Copy the archive to an SD card. This SD card will later be used as the source for writing the rootfs into the onboard eMMC.

Step2. Boot the board in QSPI/eMMC mode

- Set DIP switch to select the boot source: QSPI or eMMC. Refer to Section 3.2. Boot Mode Reference (Non-SCIF) for the correct switch positions for the target boards.
- Insert the prepared SD card (from Step 1) into the correct slot.

Table 15. SD card slot used for rootfs preparation

Board	SD card slot used for rootfs preparation	Notes
RZ/G2L-EVK	CN10 (Carrier microSD slot)	SW1-2 must be set to **ON (eMMC)**
RZ/V2L-EVK	CN10 (Carrier microSD slot)	SW1-2 must be set to **ON (eMMC)**
RZ/V2H-EVK	SoM microSD slot (on CPU board, underneath the module)	--

Note: Some early RZ/V2H-EVK revisions included two SD slots, and either slot could be used for rootfs preparation. The production version provides only the SoM slot underneath the CPU board. Always check the hardware manual for the board revision in use.

- Power on the board

Step 3. Identify the eMMC device

Determine which MMC index corresponds to the onboard eMMC. This can be checked in either U-Boot or Linux.

- In U-boot: Run the following commands:
  1. Reboot or power on the board

2. When the message “Hit any key to stop autoboot” appears, press a key to interrupt boot and enter the U-Boot console (=>).
3. Run the following commands:

```
=> mmc rescan
=> mmc list
```

After rescanning, the list will show which device is the eMMC. The example below is for the RZ/G2L-EVK, where **mmc0** is identified as the eMMC device and **mmc1** as the SD card (CN10).

```
=> mmc list
sd@11c00000: 0
sd@11c10000: 1

=> mmc rescan
=> mmc list
sd@11c00000: 0 (eMMC)
sd@11c10000: 1
```

- Run *lsblk* to list all block devices. The numbering in Linux matches the U-Boot mapping: for example, if U-Boot shows mmc0 = eMMC, then in Linux the eMMC will appear as /dev/mmcblk0.

```
root@rz-cmn:~# lsblk
```

Example output:

```
mmcblk0      179:0    0 59.3G  0 disk
|-mmcblk0p1  179:1    0  100M  0 part
`-mmcblk0p2  179:2    0  2.5G  0 part
```

Step 4. Create the partition table

```
root@rz-cmn:~# fdisk /dev/mmcblk0
```

Inside fdisk, create the partition table as follows:

1. Press o → create a new DOS disklabel.
2. Press n, then p, then ENTER for defaults, then type +500M → creates Partition 1 (boot).
3. Press n, then p, then ENTER twice for defaults → creates Partition 2 (rootfs).
4. Press t, then select partition 1, then type b → set Partition 1 type to W95 FAT32.
5. Press w → write and exit.

Afterward, check the partition table with the command below:

```
root@rz-cmn:~# fdisk -l /dev/mmcblk0
```

Expected table:

```
root@rz-cmn:~# fdisk -l  /dev/mmcblk0
Disk /dev/mmcblk0: 59.28 GiB, 63652757504 bytes, 124321792 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: 0x076c4a2a

Device         Boot  Start     End Sectors  Size Id Type
/dev/mmcblk0p1 *        32  204831  204800  100M  c W95 FAT32 (LBA)
/dev/mmcblk0p2      204832 5376159 5171328  2.5G 83 Linux
```

Step 5. Format the partitions

Format Partition 1 as FAT32 (boot):

```
root@rz-cmn:~# mkfs.vfat -F 32 -n boot /dev/mmcblk0p1
```

Format Partition 2 as ext4 (rootfs)

```
root@rz-cmn:~# mkfs.ext4 -L rootfs /dev/mmcblk0p2
```

Step 6. Populate the rootfs and kernel Image

Once the partitions are formatted, the eMMC must be populated with the Linux root filesystem and the kernel image.

4. Mount the partitions
```
root@rz-cmn:~# mkdir -p /mnt/boot
root@rz-cmn:~# mkdir -p /mnt/rootfs

root@rz-cmn:~# mount /dev/mmcblk0p1 /mnt/boot
root@rz-cmn:~# mount /dev/mmcblk0p2 /mnt/rootfs
```
5. Extract the rootfs archive into the rootfs partition
```
root@rz-cmn:~# cd /mnt/rootfs
root@rz-cmn:~# tar xpf /home/root/core-image-weston.tar.bz2
root@rz-cmn:~# sync
```
6. Copy the kernel Image, DTBs and user environment (uEnv.txt) to the boot partition
```
root@rz-cmn:~# cp -rf /boot/* /mnt/boot/
```
7. Unmount the partitions
```
root@rz-cmn:~# umount /mnt/boot
root@rz-cmn:~# umount /mnt/rootfs
```

At this point, the eMMC contains:

- Partition 1 (boot): Kernel Image + DTBs + user environment (uEnv.txt)
- Partition 2 (rootfs): Full Linux root filesystem

The board can now boot using eMMC as the root filesystem.

3.4 How To Get the Console After Bootup

Once the RZ boards has booted, on the UART terminal, you will be able to login using the default user ‘root’. There is no password. Leave the password field empty and just hit the return / enter key.

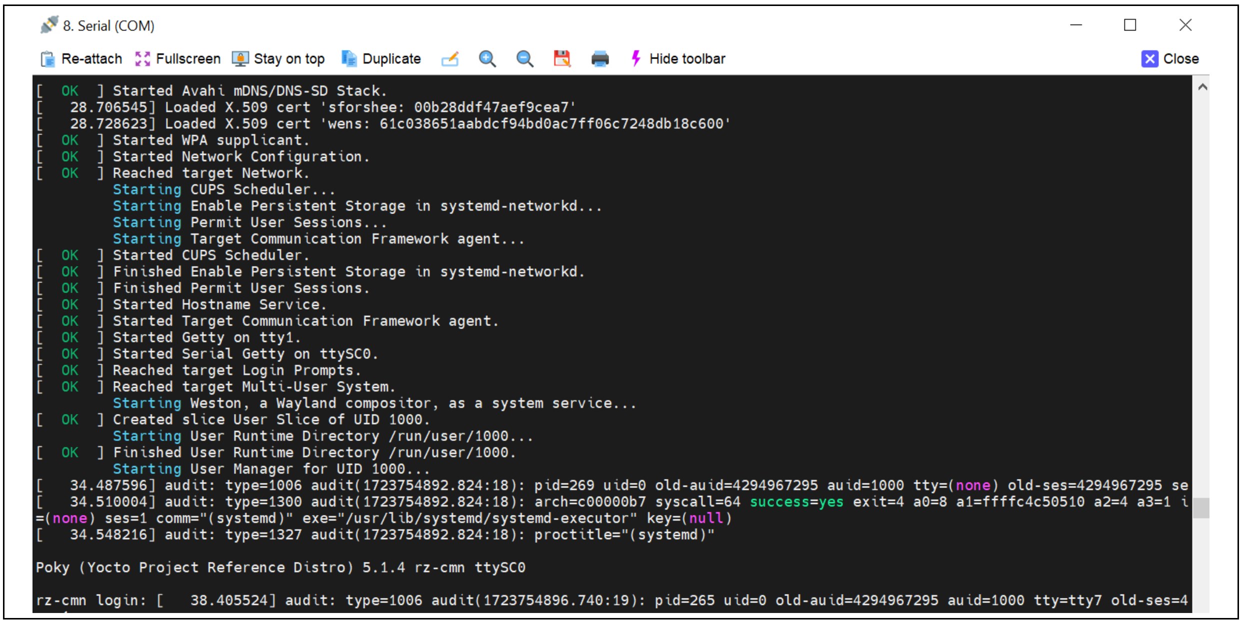

Figure 30. Root login of Linux console over UART 0.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.03.25	—	Initial release
1.10	Jun.04.25	—	Ubuntu release
2.00	Jul.02.25	—	Yocto Styhead release
3.00	Sep.25.25	—	Expanded release with support for multiple boards

System Release Package, RZ Series – User Manual

Publication Date: Sep.25.25

Published by: Renesas Electronics Corporation

RZ Family/ RZ/G Series

Renesas Electronics Corporation