Renesas RZ Family

Trail Camera Demonstration Guide

Introduction

This document describes a battery-powered camera system targeted for the Trail Camera use case. This system architecture is not limited to the Trail camera use case but can support other battery-operated camera use cases, such as security cameras.

This camera system contains two processor modules. First is the Renesas RZ/V2L SoC running Yocto Linux. This is the main processor and performs AI inference using the on-chip DRP-AI module on the captured images to identify objects. The second is the RA2E1 MCU. This is an external microcontroller that is responsible for handling motion detection and waking up the main RZ/V2L processor. The RZ/VL also has an on-chip CM33, which is simultaneously powered on to quickly capture and store images from the camera sensor for post-processing by A55, ISP, and the DRP-AI cores.

One of the key highlights of this design includes optimizing the Linux boot-up time to less than 2s and capturing a valid image within 500ms from power up. This is vital for any battery-operated camera system for various use cases.

Required Resources

Development Tools and Software

The following tools are used for development:

- Renesas e^2 studio IDE (e^2 studio | Renesas) . Please ensure FSP v3.6.0 is installed during installation.
- SEGGER JLink software (SEGGER - The Embedded Experts - Downloads - J-Link / J-Trace).
- Tera Term (Download File List - Tera Term - OSDN) on Windows PC.
- Minicom on Ubuntu host machine for accessing UART on Linux.

Hardware

- Avnet RZ/V2L Board (RZBoard V2L | Avnet Boards).
- Renesas RA2E1 MCU board (https://www.renesas.com/us/en/products/microcontrollers-microprocessors/ra-cortex-m-mcus/rtk7fpa2e1s00001be-ra2e1-fast-prototyping-board).
- Windows PC with Tera Term software and admin privileges.
- Ubuntu 20.04 host environment as native install, VM or docker environment: For working on Yocto distro's.
- UART TTL cables (Raspberry Pi compatible) featuring FTDI chipset. We do not recommend PL2302-based UART TTL cables as they have shown some issues with Windows drivers.
- Micro USB cables to interface with the host machine.
- Jumper wires.
- Micro-HDMI to HDMI display interface
- Ethernet cable
- USB2.0 ethernet dongle.
- Power supply that can provide 5V at 500mA with micro-USB pin.
- Power supply that can provide 5V at 2.5 A with USB-C pin.

Contents

1. Trail Camera Subsystem Overview

The Trail Camera Subsystem consists of two major components:

1) Renesas RZBoard V2L
2) Renesas RA2E1 Fast Prototyping Board.

1.1 RZBoard V2L

RZBoard V2L is a power-efficient, vision-AI accelerated development board in a popular single-board computer format with well-supported expansion interfaces. This Renesas RZ/V2L processor-based platform is ideal for developing cost-efficient Vision-AI and a range of energy-efficient Edge AI applications. The RZ/V2L processor has two 1.2GHz Arm® Cortex®-A55 cores, a 200MHz Cortex-M33 core, a MALI 3D GPU, and an Image Scaling Unit. This processor SoC further differentiates itself with an on-chip DRP-AI accelerator plus H.264 video (1920 x 1080) encode/decode function in silicon, making it ideal for implementing cost-effective embedded-vision applications.

RZBoard V2L is engineered in a compact Raspberry Pi form factor with a versatile set of expansion interfaces, including Gigabit Ethernet, 801.11ac Wi-Fi, Bluetooth 5, two USB 2.0 hosts, and a USB 2.0 OTG interface, MIPI DSI and CSI camera interfaces, a CANFD interface, a Pi-HAT-compatible 40-pin expansion header, and a Click Shuttle expansion header.

The board supports analog audio applications via an audio codec and stereo headphone jack. It also pins out five 12-bit ADC inputs for interfacing with analog sensors. 5V input power is sourced via a USB-C connector and managed via a single-chip Renesas RAA215300 PMIC device.

Onboard memory includes 2GB DDR4, 32GB eMMC, 16MB QSPI flash memory, and a microSD slot for removable media.

Software enablement includes a CIP Kernel-based Linux BSP (maintained for 10 years+) plus reference designs that highlight efficient vision AI implementations using the DRP-AI core. An onboard 10-pin JTAG/SWD mini-header and 4-pin UART header enable the use of an external debugger and USB serial cable.

1.2 Renesas RA2E1 Fast Prototyping Board.

The RA2E1 Fast Prototyping Board comes equipped with an R7FA2E1A93CFM microcontroller and is an evaluation board specialized for prototype development for various applications. It has a built-in emulator circuit equivalent to an E2 emulator Lite so that you can write/debug programs without additional tools. In addition, Arduino Uno and Pmod™ interfaces include standard and through-hole access to all microcontroller pins, which is highly expandable.

The image below highlights the key hardware components in the Trail Camera System.

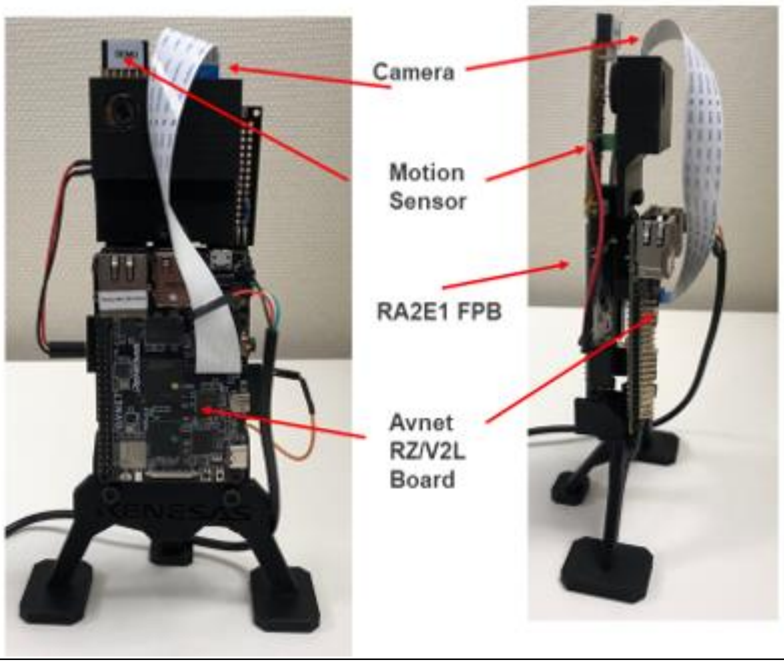


Figure 1. Trail Camera Overview

2. Key Features

The trail camera application provides the following feature set:

1. Booting Linux in less than 2 sec.
2. Capturing valid camera images in less than 500ms.
3. Low power and long operating time.
4. Demonstrates the DRP-AI NPU subsystem and its capabilities.

3. RZ/V2L MPU Subsystem

3.1 Operational Flow

The diagram below will show the operational flow of the Trail Camera system during power ON.

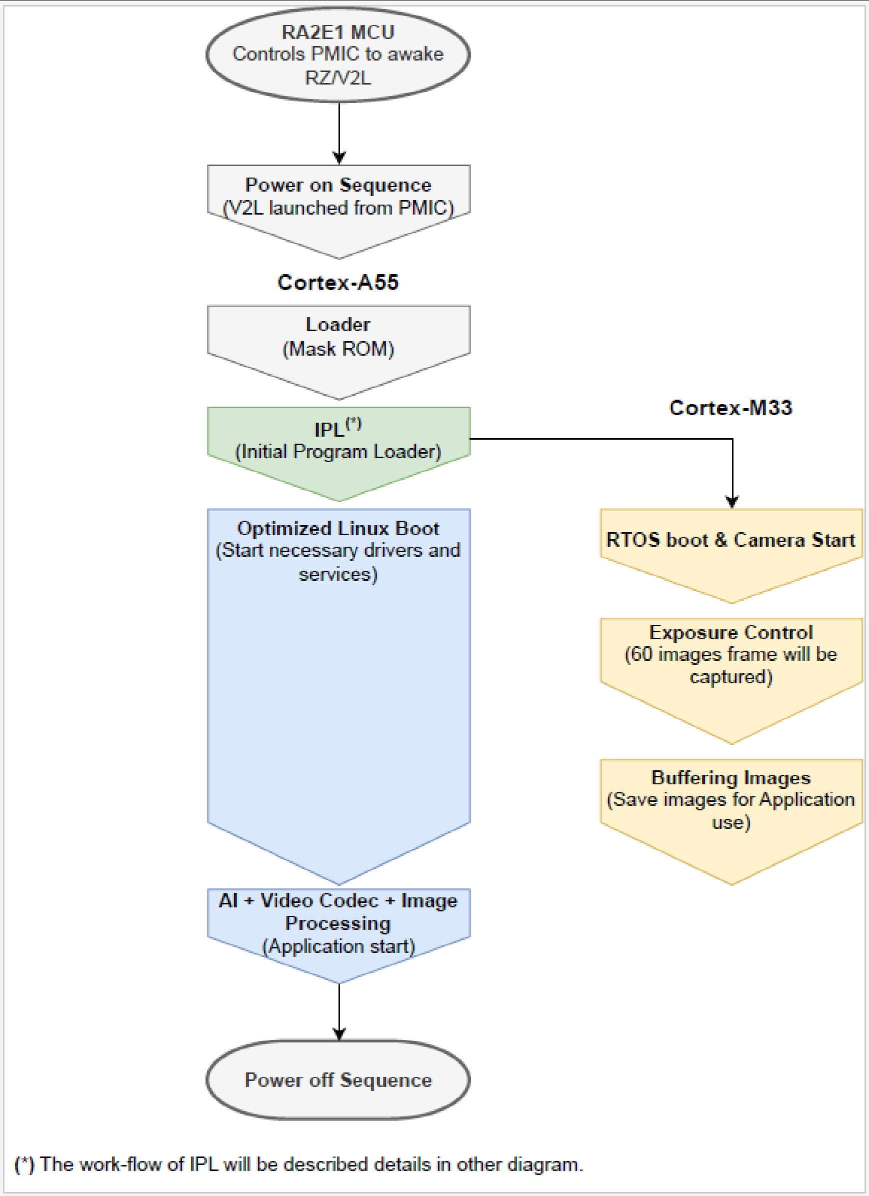

RA2E1 MCU
Controls PMIC to awake RZ/V2L
Power on Sequence (V2L launched from PMIC)
Cortex-A55
Loader (Mask ROM)
IPL(*) (Initial Program Loader)
Cortex-M33
Optimized Linux Boot (Start necessary drivers and services)
RTOS boot & Camera Start
Exposure Control (60 images frame will be captured)
Buffering Images (Save images for Application use)
AI + Video Codec + Image Processing (Application start)
Power off Sequence
(*) The work-flow of IPL will be described details in other diagram.

Figure 2. Trail Camera boot operational flow

By default, the main processor RZ/V2L is in a power-off state to conserve battery. The RA2E1 MCU is kept on power ON to detect motion using the onboard proximity sensor. Once motion is detected, RA2E1 wakes up the RZ/V2L processor, which boots up the Linux OS.

In parallel to Linux booting, the CM33 core boots up the FreeRTOS Kernel initializes the camera sensor module and starts to capture the images within 500ms from power ON for around 2s (Linux Kernel boot-up time) and stores in the shared memory between RZ/V2L processor and CM33 core.

Once the Linux boot is complete (less than 2s), the RZ/V2L processor will read the images stored in the shared memory and run the AI inference on the DRP-AI engine. This reference application project uses the Darknet YOLO v3 open-source AI model. Customers can replace this reference AI model with their custom model based on their use case.

The output of the running AI inference is routed to the HDMI port, which will show on the external HDMI monitor. Ultimately, the RZ/V2L will be shut down to conserve power, and the RA2E1 MCU will continue to monitor motion detection.

3.2 Fast Boot Linux Overview

One of the key requirements of any camera system in the market is to optimize the booting process of the Linux kernel to allow the processing of the captured images. This reference application achieves this by removing unwanted services and drivers loaded by default during the boot process. This technique is referred to as Fast Boot.

The bootup sequence has been modified to optimize the Linux kernel's boot time to meet the Trail Camera use case requirements. The picture below highlights the optimized boot flow compared to the standard boot flow of Linux OS.

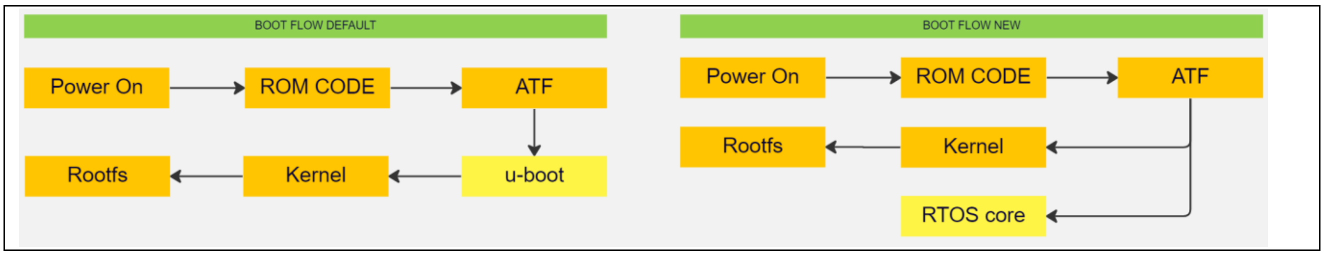


Figure 3. Comparison of normal boot flow to trail camera implementation.

U-boot is removed from the boot sequence. Instead, the ARM Trusted Firmware (ATF) loads the Linux kernel, FreeRTOS core on the M33 core, and device tree from the Flash to RAM. The CM33 core initializes the camera module and starts capturing the image from the camera sensor until the Linux kernel completes the boot-up process within 2 seconds.

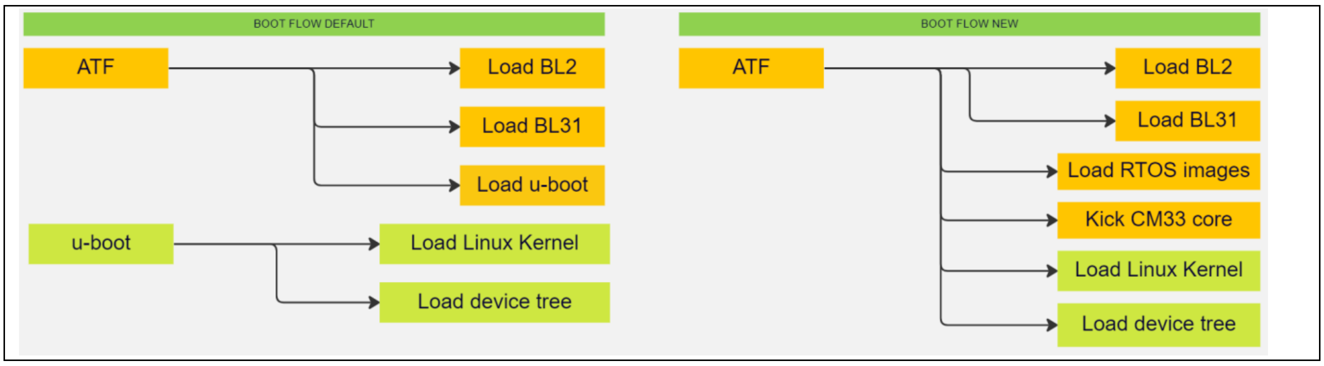


Figure 4. Comparison of ATF flow between standard and trail camera implementations

Linux Kernel, by default, loads all the built-in modules. However, the kernel is optimized to load only the necessary modules targeted for trail camera use cases. The current implementation uses "optimized systemd" for system and services management. Systemd provides an efficient and modern approach to system management, with its parallelization of service startup and on-demand starting of daemons reducing boot times and improving system responsiveness. However, systemd, by default, is a heavy mechanism and contains too many services and daemons that are started when the system comes up. Therefore, an optimized systemd is introduced for the Trail camera system with a minimal number of services to make sure the system can work efficiently and boot up as fast as possible. The necessary services are shown below:

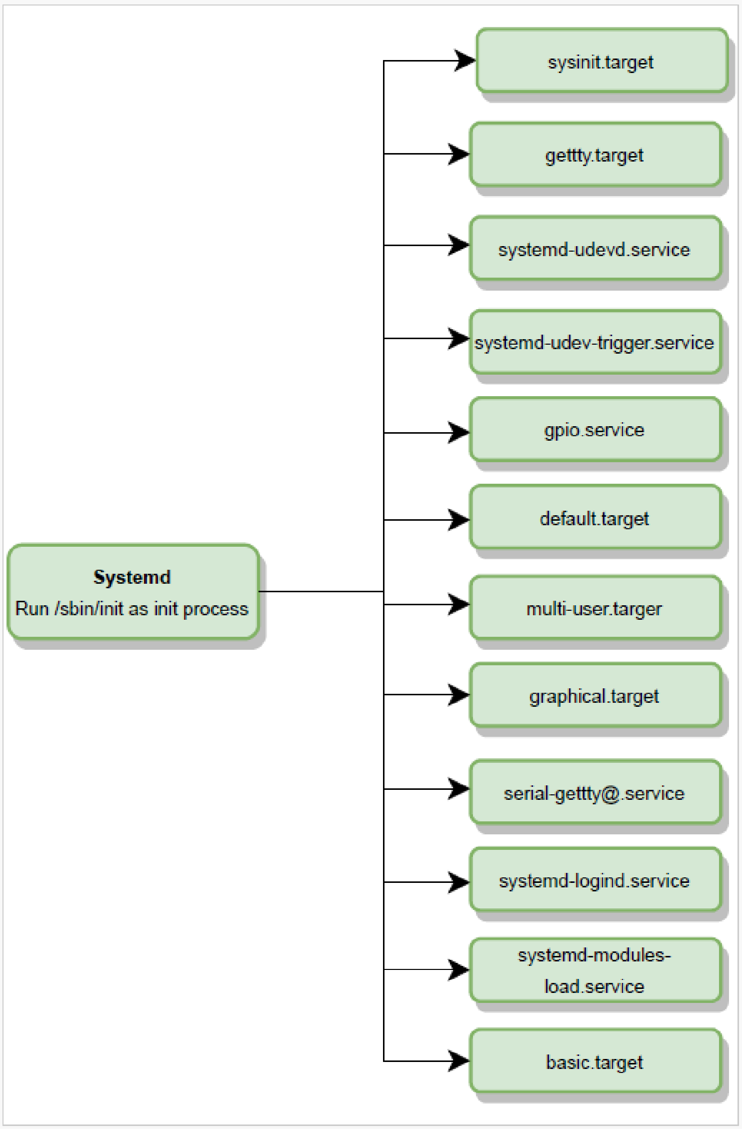


Figure 5. Necessary services of Systemd for Trail camera application

The optimized kernel only enables the necessary peripherals, as shown below:

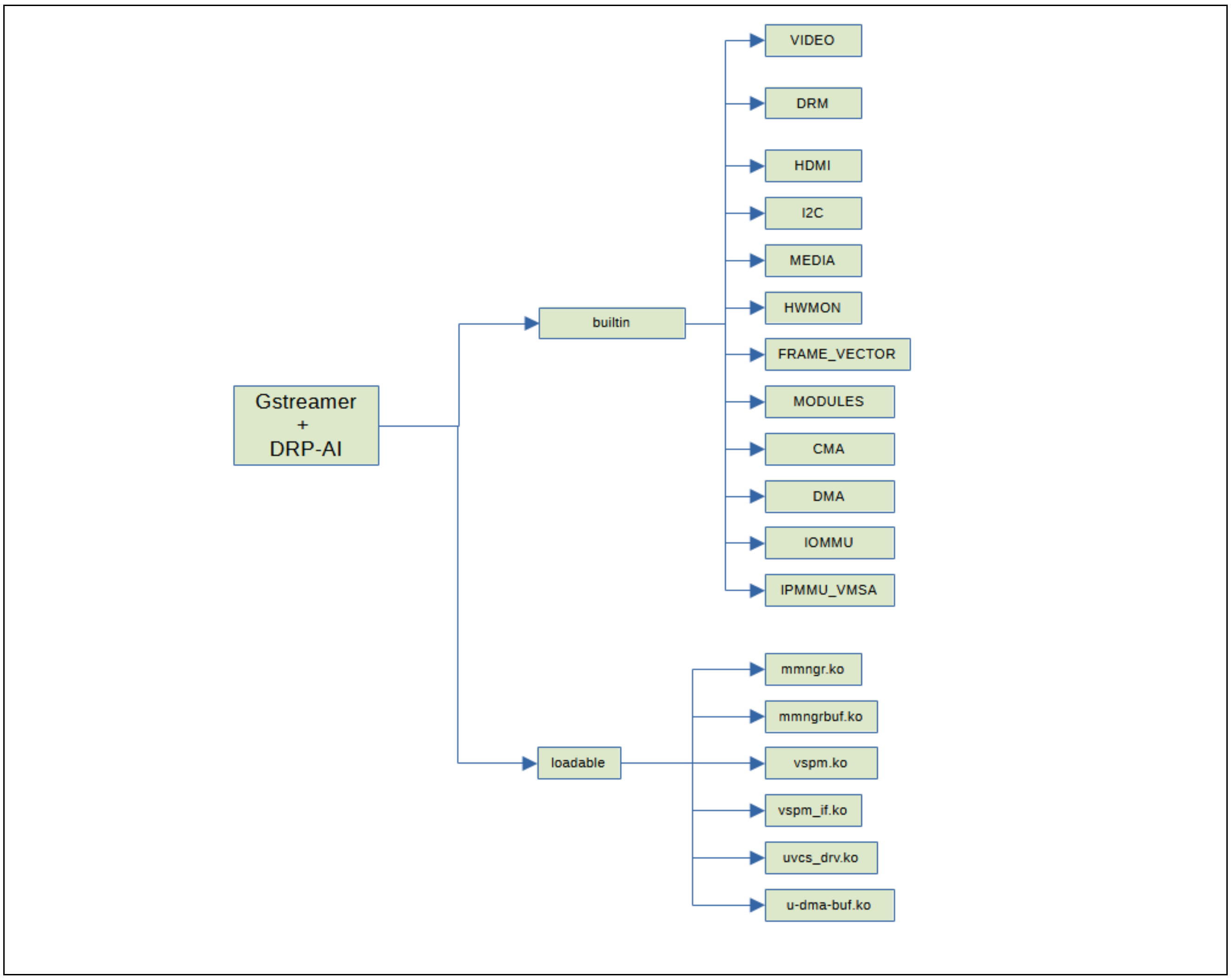


Figure 6. Post init boot of Trail Camera application

With the above optimization, the reference application achieves the following:

1) Boot up the Linux kernel in less than 2s.
2) Capture valid images using CM33 core in less than 500ms.

4. RA2E1 MCU Subsystem

4.1 Overview

The RA2E1 is a small MCU meant to be always on and run the sensor trigger to kick off the boot of the primary SoC board, the RZBoard. The RA2E1 is intended to be a low-power board and may be substituted for a board featuring the RA4 product, which supports low-power mode to optimize battery performance further. This is a simple, off-the-shelf board project with low complexity.

4.2 Operational Flow

The following is how the RA2 is expected to work.

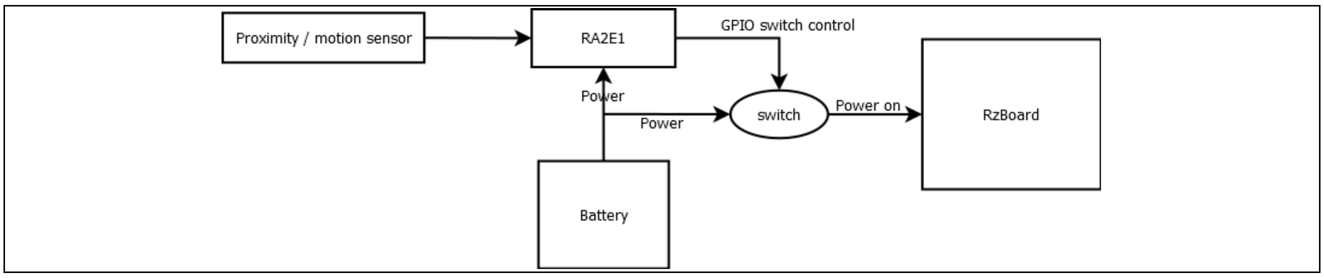


Figure 7. MCU subsystem operational flow

5. Build RZ/V2L Linux Package

5.1 Setup Build Environment

5.1.1 Requirements

- Ubuntu 20.04 LTS (64bit) is recommended as a build environment as we are using the 'Dunfell' version.
- Development packages for Yocto:
  Refer to official Yocto documentation (Yocto Project Documentation) to get started.
  Refer to the official Yocto quick build guide (Yocto Project Quick Build — The Yocto Project ® 3.1.27 documentation) for a quick start.

The files listed in the table below are part of the release package and are essential for the RZBoard Yocto build.

File	Description
rz_yocto.sh	Custom Yocto build script that unpacks VLP and other downloaded zip files arranges the layers, applies trail camera patches, sets up the environment, and initiates a build.
RZV2L-Generic-Package.zip	Required generic package for Yocto build

Table 1. ***Pre-requisite files from the release package***

5.1.2 Install packages on Ubuntu Host.

1. Update the Ubuntu package manager.

```
$ sudo apt update
```

2. Install the necessary packages and tools used by Yocto build.

```
$ sudo apt install -y 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 \
rsync curl locales bash-completion
```

3. Configure the local git account for the user.

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

4. Download the following packages provided by Renesas. Use the highlighted download links and follow the links.

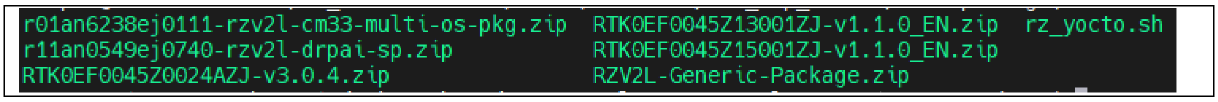
File Name	Version	Download Link	Comments
RTK0EF0045Z0024AZJ-v3.0.4.zip	v3.0.4	rzv-verified-linux-package-v304	This is the VLP release package, which contains the BSP base port and essential meta layers for the RZ family of SoCs. We are using VLP 3.0.4 for this application.
RTK0EF0045Z13001ZJ -v1.1.0_EN.zip	v1.1.0	rz-mpu-graphics-library-evaluation-version-v110	This is the Mali driver and graphics package that enables the Mali GPU in the SoC.
RTK0EF0045Z15001ZJ -v1.1.0_EN.zip	v1.1.0	rz-mpu-video-codec-library-evaluation-version-v110	H.264 video codec libraries and drivers.
r11an0549ej0740 -rzv2l-drpai-sp.zip	v7.40	rzv2l-drp-ai-support-package-version-740	This package contains the DRP-AI library and driver features.
r01an6238ej0111 -rzv2l-cm33-multi-os-pkg.zip	v1.1.1	rzv2l-multi-os-package-v111	This package provides the M33 core package containing FSP and IPC for inter-core communication.

Table 2. List of Packages to manually download for building Yocto disto

5. Copy all the downloaded zip files (shown stated above) to a folder (e.g., '~/yocto,' as shown below) in Ubuntu Host PC.

```
$ cd ~/Downloads/renesas-yocto
$ mkdir ~/yocto
$ mv *.zip ~/yocto
```

6. Copy the files **'**rz_yocto.sh**'** and 'RZV2L-Generic-Package.zip' from the release package into '~/yocto' folder as shown below.



```
r01an6238ej0111-rzv2l-cm33-multi-os-pkg.zip  RTK0EF0045Z13001ZJ-v1.1.0_EN.zip  rz_yocto.sh
r11an0549ej0740-rzv2l-drpai-sp.zip           RTK0EF0045Z15001ZJ-v1.1.0_EN.zip
RTK0EF0045Z0024AZJ-v3.0.4.zip                RZV2L-Generic-Package.zip
```

Figure 8. All the files are expected to be present in the Yocto root directory.

5.2 Initiate Yocto Build

Add execute permission to the 'rz_yocto.sh' script file.

```
$ chmod a+x rz_yocto.sh
$ ./rz_yocto.sh setup rz_avnet
```

To set up the yocto packages and prepare the build, run the following command:

The above command will:

1. Create a directory 'yocto_rzboard.'
2. Extract all the individual downloaded packages and set up the meta layers.
3. Apply changes, a.k.a patches, to build for the trail camera system.
4. Create a 'build' folder and configure it with the right 'MACHINE' and 'DISTO' build configuration.

Run the below command to build Yocto disto.

```
$ ./rz_yocto.sh build rz_avnet
```

When building is complete, run the below command to copy the Trail Camera application artifacts to the output folder:

```
$ ./rz_yocto.sh output rz_avnet
```

The output images will be located at `~/yocto/output/`

This directory will have the same contents as the release directory:
'images/RZBoard_Windows_Flashing/images/'

Make a backup copy of the release package and replace the contents of the 'images' directory with the contents of the 'output' directory.

6. Build RA2E1 MCU Package

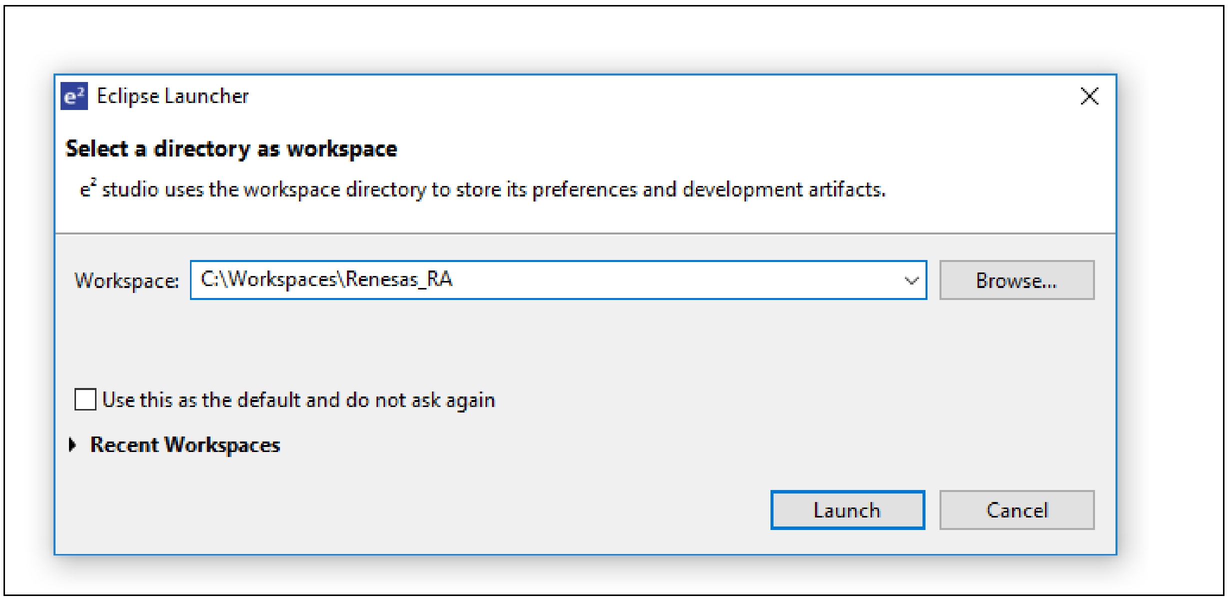
The release package contains the source code for the RA2E1 sensor trigger in the path src/RA2E1_MCU/Motion_sensor_FPB_RA2E1.zip.

Please ensure the workstation is set up correctly with all the right tools, as listed in the Software Tools section.

6.1 Importing the Project

1. Launch e^2 studio ISDE. e^2 studio can be launched from the Windows start menu.

2. In the Eclipse Launcher window, specify the destination for the new workspace. It is recommended to keep the path simple and avoid using spaces.



Eclipse Launcher

Select a directory as workspace

e² studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Workspaces\Renesas_RA

Figure 9. Setting up e^2 studio workspace for RA2E1 MCU project.

3. Click *Launch* to start e^2 studio in the specified path. When prompted, log into your My Renesas account or click *Cancel* to dismiss the popup window asking for permission to log and report usage (it will remain disabled). Select *No* when prompted to set up a password hint.
4. The welcome screen may appear inside the new workspace. You can dismiss it by clicking the Hide button at the top-right corner.

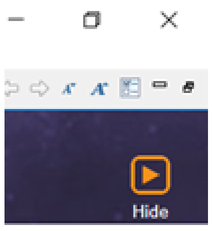


Figure 10. Hide the button on the welcome screen on the e^2 studio.

5. Go to *File -> Import*. Click import.

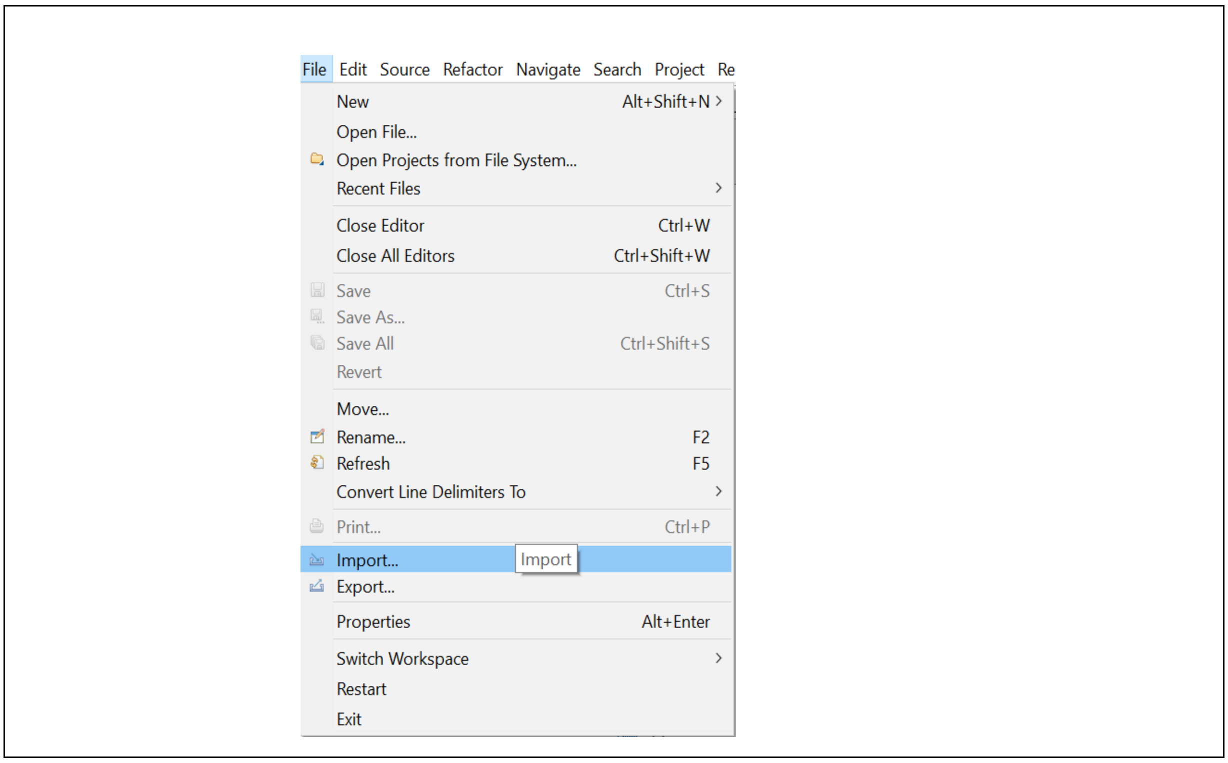
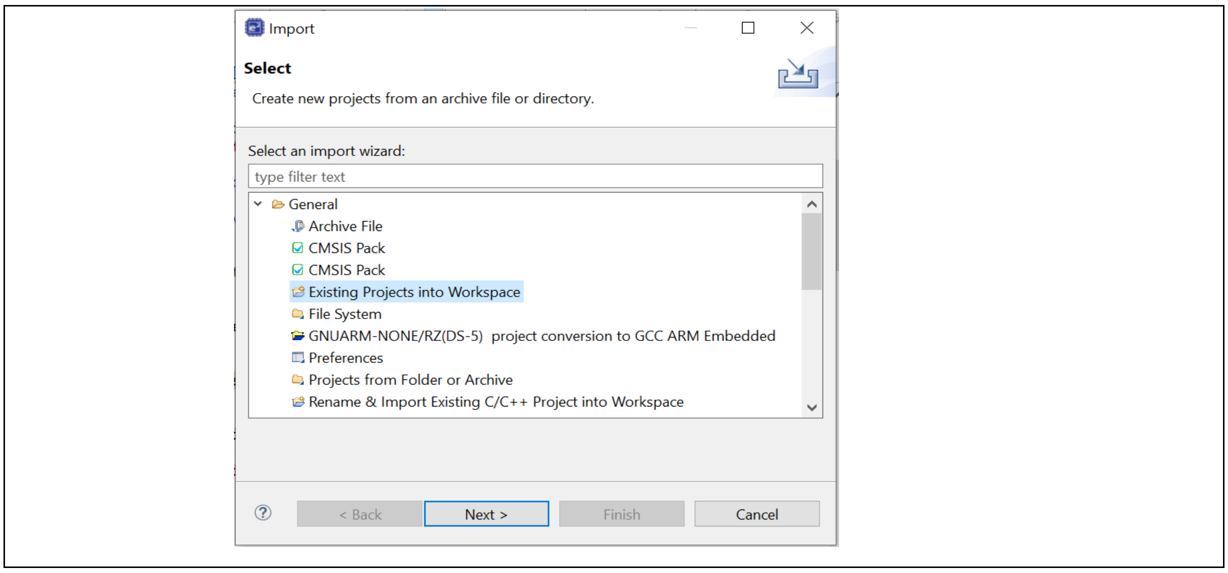


Figure 11. Import menu option.

6. Expand *General -> Existing Project Into workspace*. Click Next.


Import

Select

Create new projects from an archive file or directory.

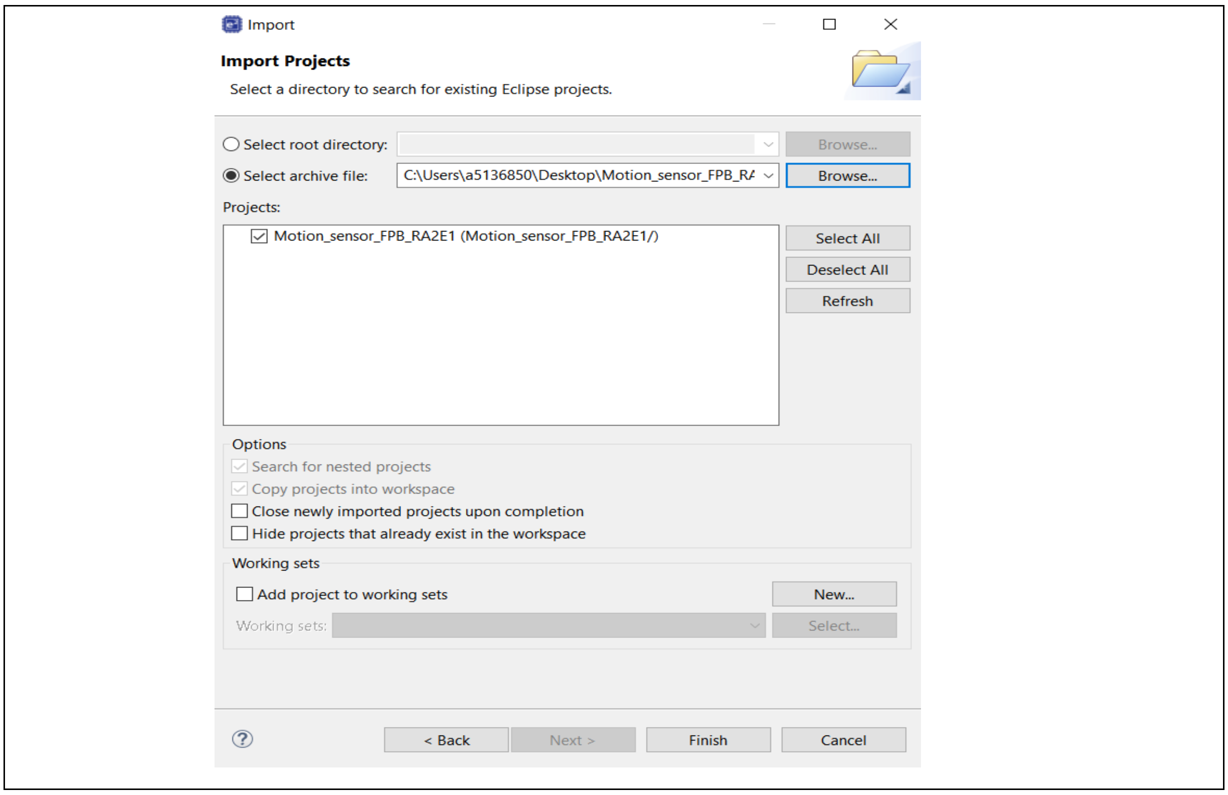
Select an import wizard:

type filter text

General
Archive File
CMSIS Pack
CMSIS Pack
Existing Projects into Workspace
File System
GNUARM-NONE/RZ(DS-5) project conversion to GCC ARM Embedded
Preferences
Projects from Folder or Archive
Rename & Import Existing C/C++ Project into Workspace
< Back
Next >
Finish
Cancel

Figure 12. Import option for existing projects.

7. On the Import dialog, Click Select archive file. Then click on Browse and navigate to the lab materials folder. Select the Motion_sensor_FPB_RA2E1.zip archive file to import from.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:
● Select archive file: C:\Users\a5136850\Desktop\Motion_sensor_FPB_RA

Projects:
☑ Motion_sensor_FPB_RA2E1 (Motion_sensor_FPB_RA2E1/)

Options
☑ Search for nested projects
☑ Copy projects into workspace
☐ Close newly imported projects upon completion
☐ Hide projects that already exist in the workspace

Working sets
☐ Add project to working sets
Working sets:

Figure 13. Selecting archive file location.

8. Click *Finish.*

At this stage, you have successfully imported the project into your workspace.

6.2 Build e^2 studio project

1. The project is now ready to be compiled. Press the “hammer” icon to start building the project.

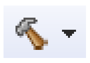
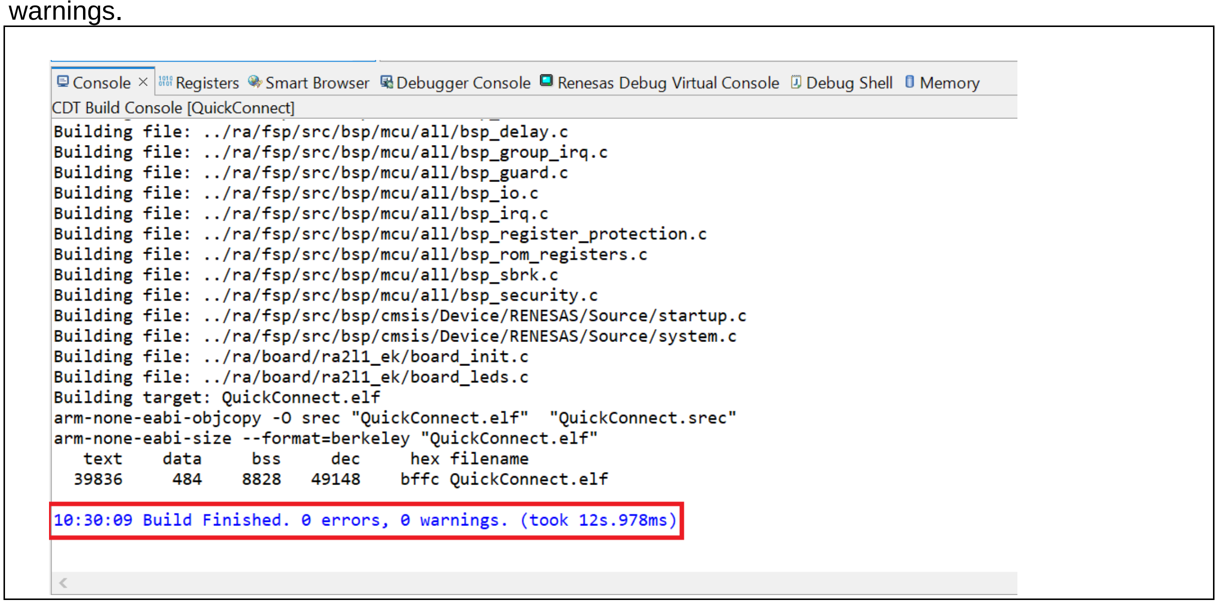


Figure 14. Build icon in e^2 studio.

2. The toolchain will report compilation and build status to the console pane in the lower-right corner of e^2 studio. When the build is completed, it should confirm that there are zero errors and zero warnings.



CDT Build Console [QuickConnect]

```
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_delay.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_group_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_guard.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_io.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_irq.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_register_protection.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_rom_registers.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_sbrk.c
Building file: ../ra/fsp/src/bsp/mcu/all/bsp_security.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/startup.c
Building file: ../ra/fsp/src/bsp/cmsis/Device/RENESAS/Source/system.c
Building file: ../ra/board/ra2l1_ek/board_init.c
Building file: ../ra/board/ra2l1_ek/board_leds.c
Building target: QuickConnect.elf
arm-none-eabi-objcopy -O srec "QuickConnect.elf"  "QuickConnect.srec"
arm-none-eabi-size --format=berkeley "QuickConnect.elf"
   text    data     bss     dec     hex filename
  39836     484    8828   49148    bffc QuickConnect.elf

10:30:09 Build Finished. 0 errors, 0 warnings. (took 12s.978ms)
```

Figure 15. Build log showing successful build.

The application is now ready to be programmed and run on the FPB-RA2E1 board.

7. Programming the Platform

7.1 Powering up the system

The trail camera system has two independent boards: the RA2EK1 FPB and the RZBoard V2L. Each board needs to be powered separately. Use a USB-C power cable to power the RZBoard and a USB micro power cable for the RA2EK1.

⚠ Caution: To avoid electrical damage, both power sources must be on the same ground plane. To ensure that the two power sources are on the same ground plane, either use a single USB power source or USB power delivery hub or, at the very least, from a power strip with common grounding.

7.2 Programming RA2E1 FPB

The programming of the FPB-RA2E1 is performed from within the e^2 studio IDE as follows:

1. Once the project is imported and built, the application is now ready to be programmed and run on the FPB-RA2E1 board. Press the “bug” icon to begin the debug session.

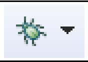
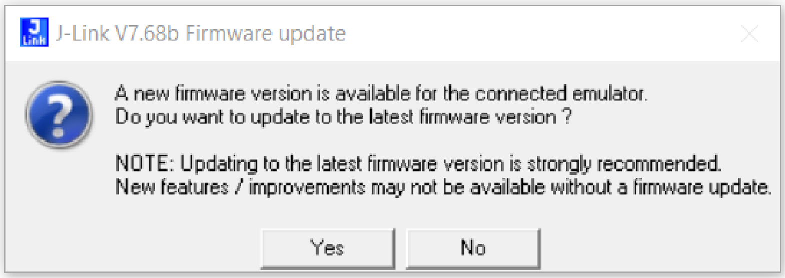


Figure 16. Debug button icon in e^2 studio

2. You may be prompted to update the J-Link debugger firmware. Click *Yes* to update. It will take a few moments to complete.



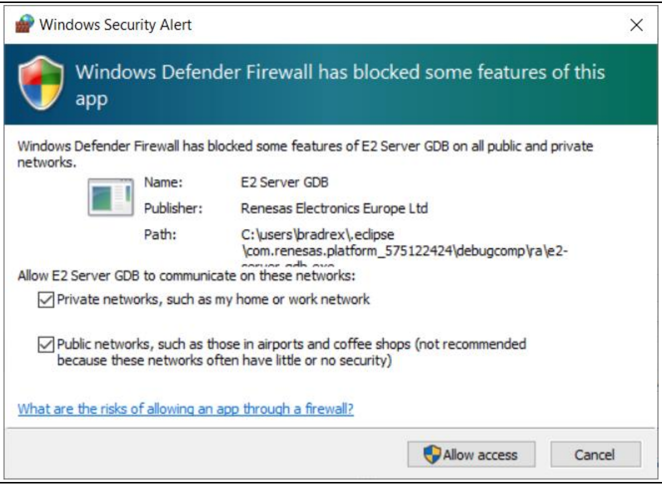
J-Link V7.68b Firmware update

A new firmware version is available for the connected emulator.
Do you want to update to the latest firmware version ?

NOTE: Updating to the latest firmware version is strongly recommended.
New features / improvements may not be available without a firmware update.

Figure 17. Firmware update notification from JLink module

3. Windows could also prompt you to allow the gdb server through your firewall. Click the checkbox to allow it through private networks, then *Allow access*.



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of E2 Server GDB on all public and private networks.

Name:	E2 Server GDB
Publisher:	Renesas Electronics Europe Ltd
Path:	C:\users\bradrex\.eclipse \com.renesas.platform_575122424\debugcomp\ra\e2-

Allow E2 Server GDB to communicate on these networks:

☑ Private networks, such as my home or work network

☑ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

Figure 18. Windows firewall permission request popup.

4. e^2 studio will perform flash programming routines, prompting you to switch to the Debug perspective. Select the Remember my decision check box and click *Switch*. LED5 near the debug USB port will blink while programming.

5. The debug session has now started, and the application has been paused at its entry function (Reset_Handler).

6. Click the *Resume* button or press *F8* on the keyboard to start the application.

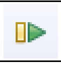


Figure 19. Resume button icon in e^2 studio.

7. The Program will stop again, this time at the start of the main function. Low-level initialization routines are now completed. Click the 'Resume' button or press the 'F8' key to resume the application and execute user code.

At this point, the programming should be successfully completed.

7.3 Programming RZBoard V2L

At this stage, it is assumed that the user followed the instructions to build the Linux package. The output images can be found under the '~/yocto/output/' folder.

Please note that this location only holds the images, not a formatted release package. The images will need to be packaged with the scripts manually. At this stage, it is recommended to follow the instructions mentioned in the "Programming RZBoard with Prebuilt Images" sections to program the built images onto the target hardware.

8. Booting Trail Camera System

At this stage, it is assumed that the trail camera system is powered on and the images are programmed to the targeted core. Please refer to the instructions in the "Powering up the system" section on how to power up the system.

Before powering ON, it is recommended that the board be set up with a test image in front of the camera mount, as shown below. This is a critical step to getting a valid output.

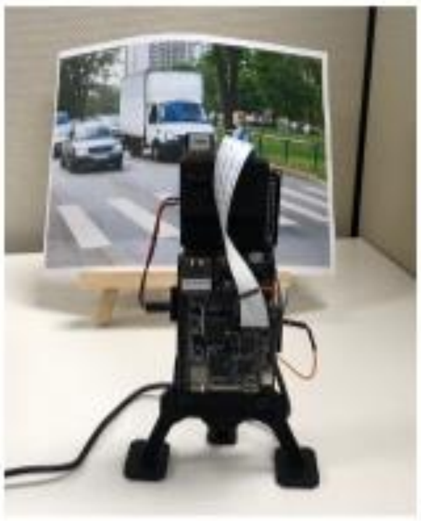


Figure 20. Trail camera minimal setup.

A complete setup would involve an HDMI display, but a minimal setup would still include the trail camera unit, two power supplies, a UART cable, and a stationary test image. You may need to ensure the image is between 2 and 3 feet from the trail camera. You will likely require the HDMI display to view and align the image.

During the bootup process, the CM33 core initializes the camera sensor, captures the images, and stores them in the shared memory between CM33 and the RZ/V2L processor.

After the system boots up, the following script, found at /home/root/app-run/capture_v1.0.sh, is executed automatically. In its default mode, this script invokes the application, which performs the inference model using DRP-AI and generates an image with markdowns.

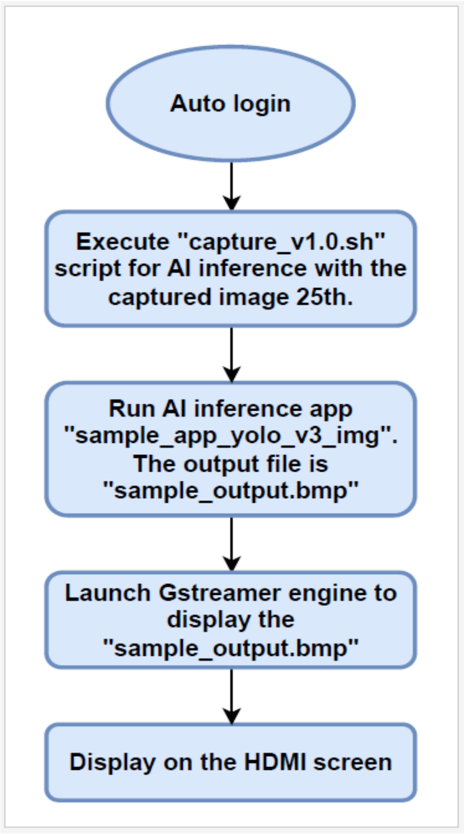

Auto login
Execute "capture_v1.0.sh" script for AI inference with the captured image 25th.
Run AI inference app "sample_app_yolo_v3_img". The output file is "sample_output.bmp"
Launch Gstreamer engine to display the "sample_output.bmp"
Display on the HDMI screen

Figure 21. Post init application flow with auto-launch.

8.1 AI Inference Application

The following diagram shows the general flow of the AI inference application.

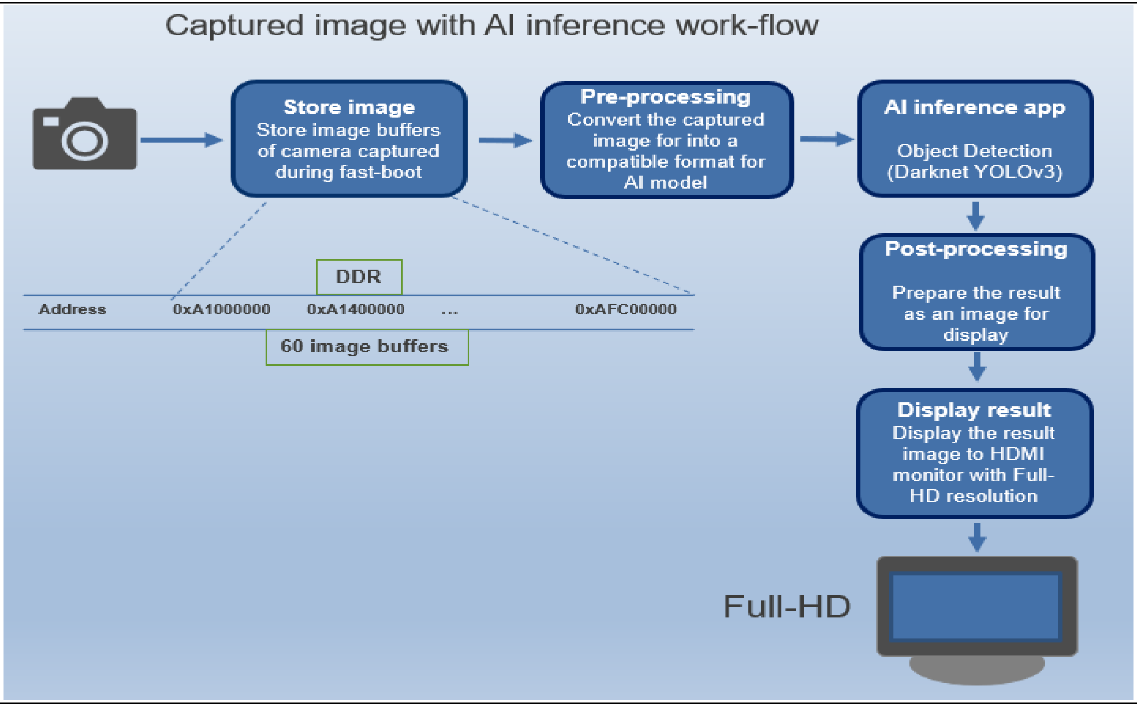

Captured image with AI inference work-flow
Store image
Store image buffers of camera captured during fast-boot
Pre-processing
Convert the captured image for into a compatible format for AI model
AI inference app
Object Detection (Darknet YOLOv3)
Post-processing
Prepare the result as an image for display
Display result
Display the result image to HDMI monitor with Full-HD resolution
DDR
Address
0xA1000000
0xA1400000
...
0xAFC00000
60 image buffers
Full-HD

Figure 22. AI application flow.

The following is the image of the RZBoard on display.

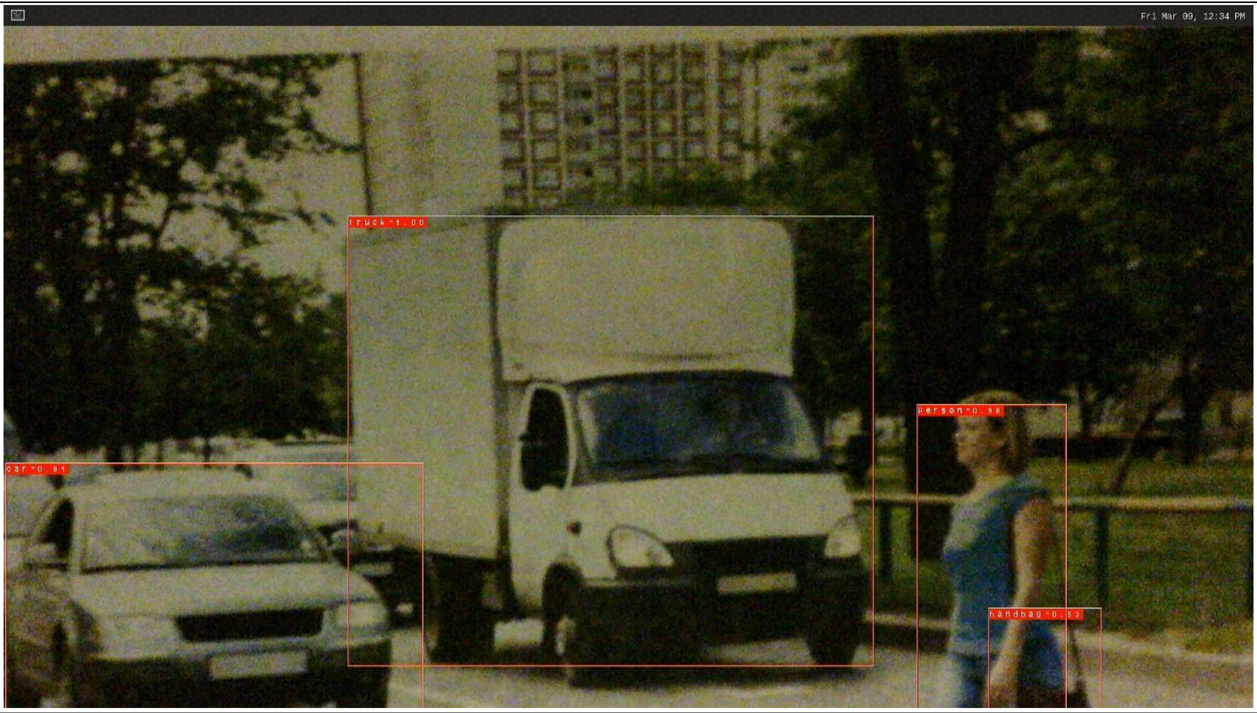
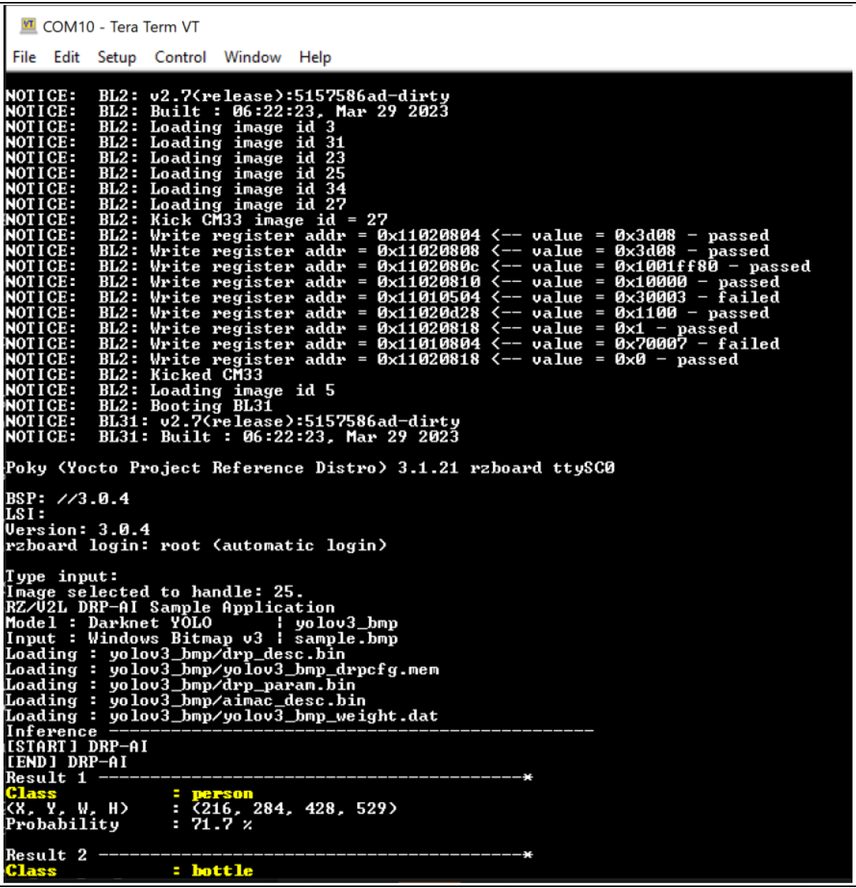


Figure 23. DRP-AI post-processed image

The console log for the image looks as follows:



```
NOTICE:  BL2: v2.7(release):5157586ad-dirty
NOTICE:  BL2: Built : 06:22:23, Mar 29 2023
NOTICE:  BL2: Loading image id 3
NOTICE:  BL2: Loading image id 31
NOTICE:  BL2: Loading image id 23
NOTICE:  BL2: Loading image id 25
NOTICE:  BL2: Loading image id 34
NOTICE:  BL2: Loading image id 27
NOTICE:  BL2: Kick CM33 image id = 27
NOTICE:  BL2: Write register addr = 0x11020804 <-- value = 0x3d08 - passed
NOTICE:  BL2: Write register addr = 0x11020808 <-- value = 0x3d08 - passed
NOTICE:  BL2: Write register addr = 0x1102080c <-- value = 0x1001ff80 - passed
NOTICE:  BL2: Write register addr = 0x11020810 <-- value = 0x10000 - passed
NOTICE:  BL2: Write register addr = 0x11010504 <-- value = 0x30003 - failed
NOTICE:  BL2: Write register addr = 0x11020d28 <-- value = 0x1100 - passed
NOTICE:  BL2: Write register addr = 0x11020818 <-- value = 0x1 - passed
NOTICE:  BL2: Write register addr = 0x11010804 <-- value = 0x70007 - failed
NOTICE:  BL2: Write register addr = 0x11020818 <-- value = 0x0 - passed
NOTICE:  BL2: Kicked CM33
NOTICE:  BL2: Loading image id 5
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.7(release):5157586ad-dirty
NOTICE:  BL31: Built : 06:22:23, Mar 29 2023

Poky (Yocto Project Reference Distro) 3.1.21 rzboard ttySC0

BSP: //3.0.4
LSI:
Version: 3.0.4
rzboard login: root (automatic login)

Type input:
Image selected to handle: 25.
RZ/V2L DRP-AI Sample Application
Model : Darknet YOLO       | yolov3_bmp
Input : Windows Bitmap v3  | sample.bmp
Loading : yolov3_bmp/drp_desc.bin
Loading : yolov3_bmp/yolov3_bmp_drpcfg.mem
Loading : yolov3_bmp/drp_param.bin
Loading : yolov3_bmp/aimac_desc.bin
Loading : yolov3_bmp/yolov3_bmp_weight.dat
Inference -----------------------------------------------
[START] DRP-AI
[END] DRP-AI
Result 1 -----------------------------------------*
Class          : person
(X, Y, W, H)   : (216, 284, 428, 529)
Probability    : 71.7 %

Result 2 -----------------------------------------*
Class          : bottle
```

Figure 24. AI inference application console logs. It will differ based on the environment and target image.

The darknet YOLOV3 model deployed in the demo application can identify multiple objects from the same image in seconds. The model can simultaneously identify faces, people, animals, computer displays, and many other objects.

9. Flashing Prebuilt images

9.1 Programming RZBoard with Prebuilt Images

9.1.1 Hardware Setup

You will need the following accessories to perform the flashing:

1. USB to micro-USB cable.
2. UART cable with individual pins. Please use an FTDI-based cable and avoid PL230x, as there are driver issues with this on Windows.
3. USB-C power supply that can deliver at least 2.5A of power.
4. Jumper wire to force the board into SCIF mode, a.k.a UART image load mode.
5. Host machine with Windows OS or Linux OS.

The Avnet RZBoard uses the USB 2.0 OTG port for flashing, which requires the cable to be a USB 2.0 Micro-B type on one end.

Attaching USB-C power will immediately power up and boot the board. Removing the USB-C power plug is the only way to remove power. Booting is indicated by changes in the LEDs and console output on the TTL UART port. Due to the electrical modifications performed, the power 'On' switch S1 on the RZBoard can no longer control power on/off.

The image below shows a disassembled trail camera RZBoard used to demonstrate the needed IO connections.

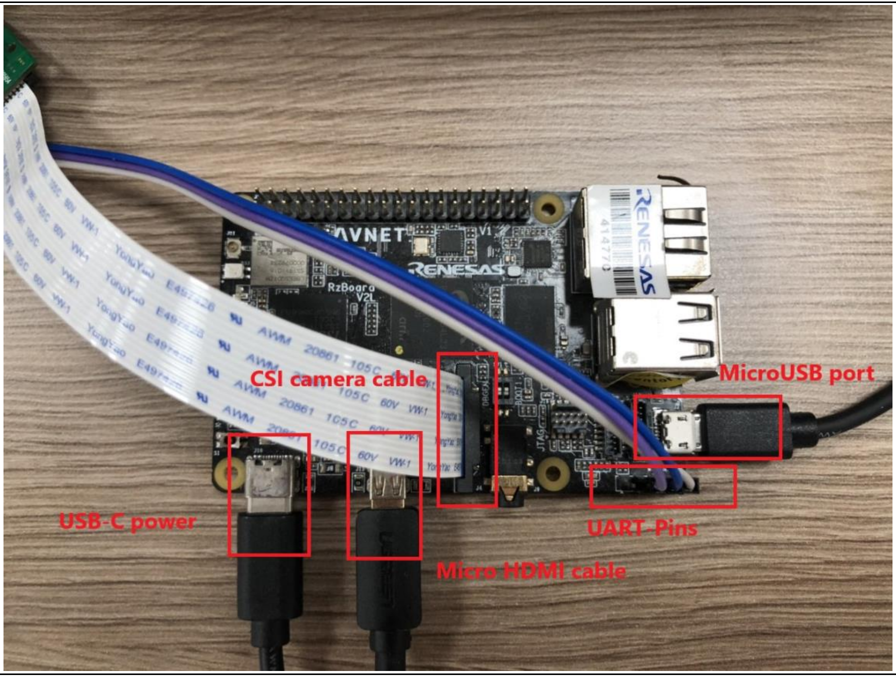
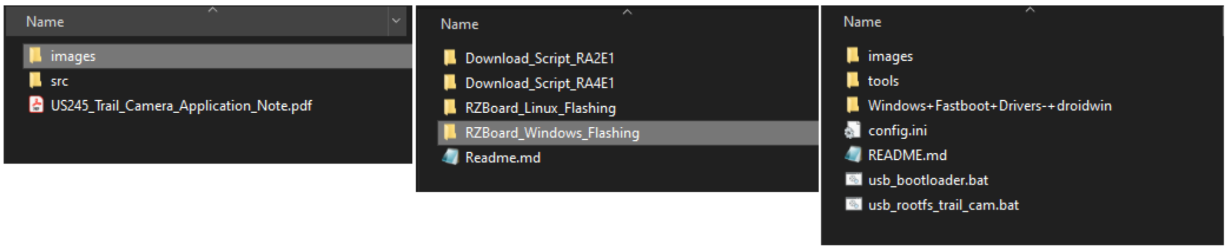


Figure 25. Trail Camera disassembled RZBoard to demonstrate IO connections for flashing.

9.1.2 Flashing procedure on Windows

The Release package contains a directory named ‘RZBoard_Windows_Flashing.’ This directory contains the images, tools, and scripts necessary to program the RZBoard on Windows OS successfully. It also includes images for bootloaders, MCU RTOS, and Linux system images.

Refer to the image below to navigate to the prebuilt folder starting from the root of the release package.



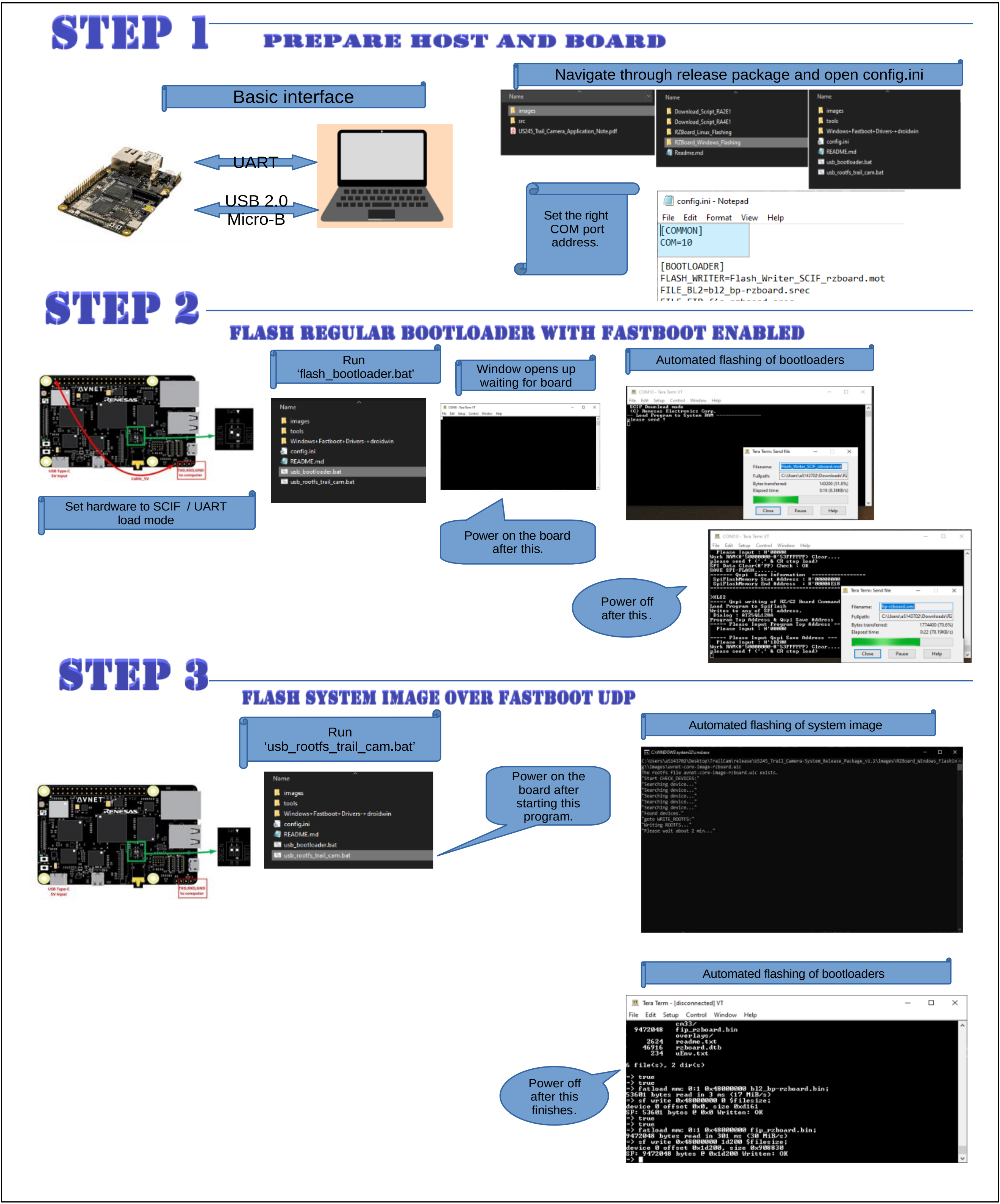
Name
images
src
US245_Trail_Camera_Application_Note.pdf

Name
Download_Script_RA2E1
Download_Script_RA4E1
RZBoard_Linux_Flashing
RZBoard_Windows_Flashing
Readme.md

Name
images
tools
Windows+Fastboot+Drivers-+droidwin
config.ini
README.md
usb_bootloader.bat
usb_rootfs_trail_cam.bat

Figure 26. Directory flow from root of the release package to Windows flashing directory.

Below is a summary of the flashing process.


STEP 1
PREPARE HOST AND BOARD
Basic interface
UART
USB 2.0 Micro-B
Navigate through release package and open config.ini
images
src
US245_Trail_Camera_Application_Note.pdf
Download_Script_RAZE1
Download_Script_RA4E1
RZBoard_Linux_Flashing
RZBoard_Windows_Flashing
Readme.md
images
tools
Windows+Fastboot+Drivers-+droidwin
config.ini
README.md
usb_bootloader.bat
usb_rootfs_trail_cam.bat
Set the right COM port address.
config.ini - Notepad
File Edit Format View Help
STEP 2
FLASH REGULAR BOOTLOADER WITH FASTBOOT ENABLED
Run 'flash_bootloader.bat'
Window opens up waiting for board
Automated flashing of bootloaders
Set hardware to SCIF / UART load mode
Power on the board after this.
Power off after this.
AVNET
RENESAS
USB Type-C 5V input
TXD,RXD,GND to computer
images
tools
Windows+Fastboot+Drivers-+droidwin
config.ini
README.md
usb_bootloader.bat
usb_rootfs_trail_cam.bat
COM10 - Tera Term VT
File Edit Setup Control Window Help
Tera Term: Send file
Close Pause Help
STEP 3
FLASH SYSTEM IMAGE OVER FASTBOOT UDP
Run 'usb_rootfs_trail_cam.bat'
Power on the board after starting this program.
Automated flashing of system image
images
tools
Windows+Fastboot+Drivers-+droidwin
config.ini
README.md
usb_bootloader.bat
usb_rootfs_trail_cam.bat
C:\WINDOWS\system32\cmd.exe
Automated flashing of bootloaders
Power off after this finishes.
Tera Term - [disconnected] VT
File Edit Setup Control Window Help

```
[COMMON]
COM=10

[BOOTLOADER]
FLASH_WRITER=Flash_Writer_SCIF_rzboard.mot
FILE_BL2=bl2_bp-rzboard.srec
```

```
SCIF Download mode
 (C) Renesas Electronics Corp.
-- Load Program to System RAM ---------------
please send !
```

```
Filename:          Flash_Writer_SCIF_rzboard.mot
Fullpath:          C:\Users\a5143702\Downloads\RZ
Bytes transferred: 143200 (51.8%)
Elapsed time:      0:16 (8.36KB/s)
```

```
  Please Input : H'000000
Work RAM(H'50000000-H'53FFFFFF) Clear....
please send ! ('.' & CR stop load)
SPI Data Clear(H'FF) Check : OK
SAVE SPI-FLASH.......
======= Qspi  Save Information  =================
 SpiFlashMemory Stat Address : H'00000000
 SpiFlashMemory End Address  : H'0000AE18
===========================================

>XLS2
===== Qspi writing of RZ/G2 Board Command
Load Program to Spiflash
Writes to any of SPI address.
 Dialog : AT25QL128A
Program Top Address & Qspi Save Address
===== Please Input Program Top Address ==
  Please Input : H'00000

===== Please Input Qspi Save Address ===
  Please Input : H'1D200
Work RAM(H'50000000-H'53FFFFFF) Clear....
please send ! ('.' & CR stop load)
```

```
Filename:          fip-rzboard.srec
Fullpath:          C:\Users\a5143702\Downloads\RZ
Bytes transferred: 1774400 (70.6%)
Elapsed time:      0:22 (78.19KB/s)
```

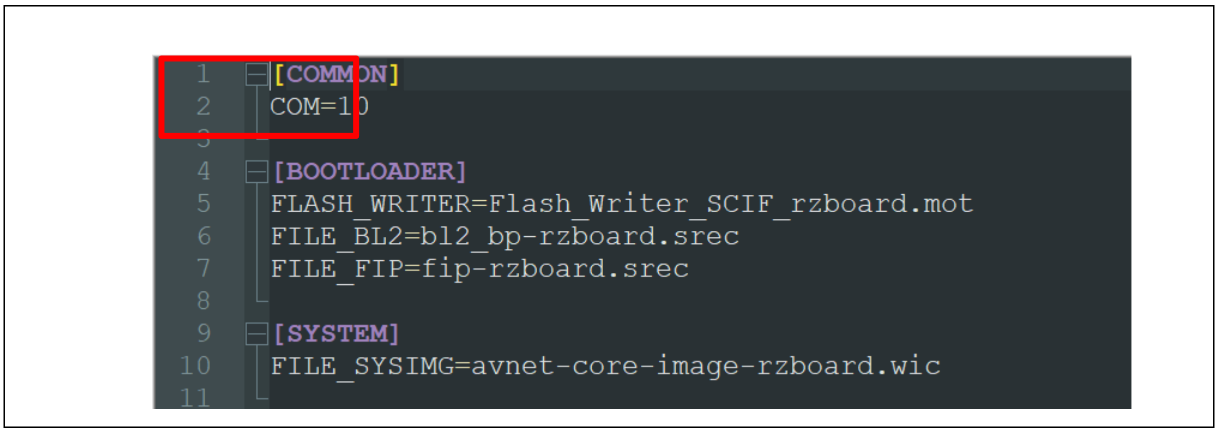
```
C:\Users\a5143702\Desktop\TrailCam\release\US245_Trail_Camera-System_Release_Package_v1.2\images\RZBoard_Windows_Flashin
g\\images\avnet-core-image-rzboard.wic
The rootfs file avnet-core-image-rzboard.wic exists.
"Start CHECK_DEVICES:"
"Searching device..."
"Searching device..."
"Searching device..."
"Searching device..."
"Searching device..."
"Found devices."
"goto WRITE_ROOTFS:"
"Writing ROOTFS..."
"Please wait about 2 min..."
```

```
            cn33/
  9472048   fip_rzboard.bin
            overlays/
     2624   readme.txt
    46916   rzboard.dtb
      234   uEnv.txt

6 file(s), 2 dir(s)

=> true
=> true
=> fatload mmc 0:1 0x48000000 bl2_bp-rzboard.bin;
53601 bytes read in 3 ms (17 MiB/s)
=> sf write 0x48000000 0 $filesize;
device 0 offset 0x0, size 0xd161
SF: 53601 bytes @ 0x0 Written: OK
=> true
=> true
=> fatload mmc 0:1 0x48000000 fip_rzboard.bin;
9472048 bytes read in 301 ms (30 MiB/s)
=> sf write 0x48000000 1d200 $filesize;
device 0 offset 0x1d200, size 0x908830
SF: 9472048 bytes @ 0x1d200 Written: OK
=>
```

Figure 27. RZBoard flashing process summary.

The following is the flashing procedure.

1. Before performing image flashing, ensure the ‘Fastboot USB driver’ is installed on the Windows host PC. Check the Appendix section for more details.
2. Connect the UART TTL cable to the Avnet board’s UART port.
3. Connect the USB to USB 2.0 micro-B cable between RZBoard and Windows host PC.
4. Identify the TTY COM port number using Windows device manager or Tera Term.
5. The USB port number will automatically be detected by the flashing scripts.
6. Update the value for ‘COM’ in the config.ini file with the number identified in step 4.



```
[COMMON]
COM=10

[BOOTLOADER]
FLASH_WRITER=Flash_Writer_SCIF_rzboard.mot
FILE_BL2=bl2_bp-rzboard.srec
FILE_FIP=fip-rzboard.srec

[SYSTEM]
FILE_SYSIMG=avnet-core-image-rzboard.wic
```

Figure 28. changes to config.ini

7. Make sure the RZBoard is powered off.
8. Set the board configuration to put the RZBoard to SCIF download mode**, initiating** an image load from the UART port**.** Use a jumper wire between the pins as indicated in the diagram below and ensure that the switch is set as indicated in the image.

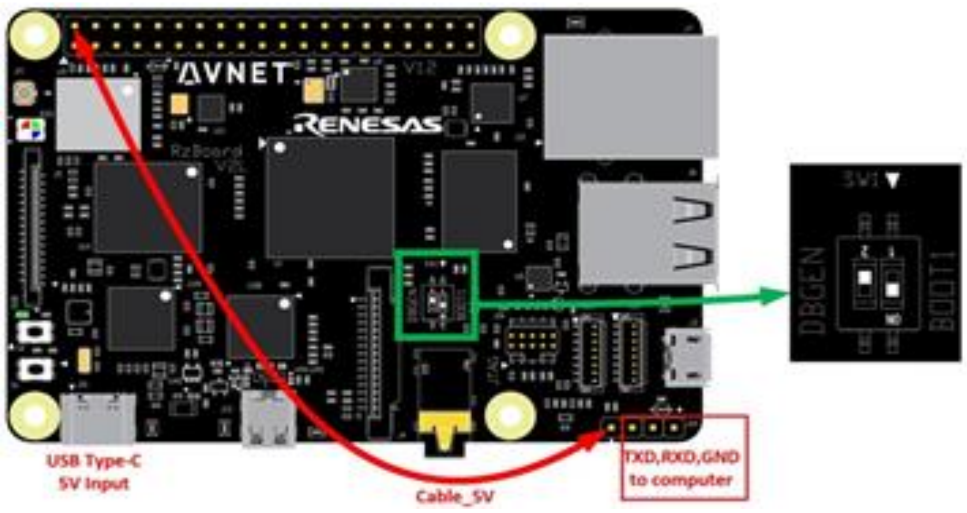
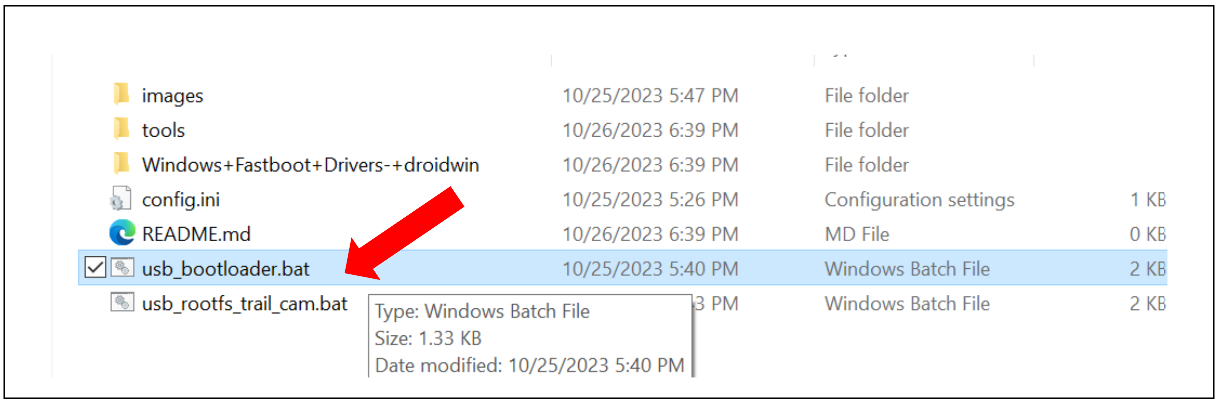


Figure 29. RZBoard scif (UART load) mode hardware configuration

9. On the host machine, execute the batch script 'usb_bootloader.bat.'

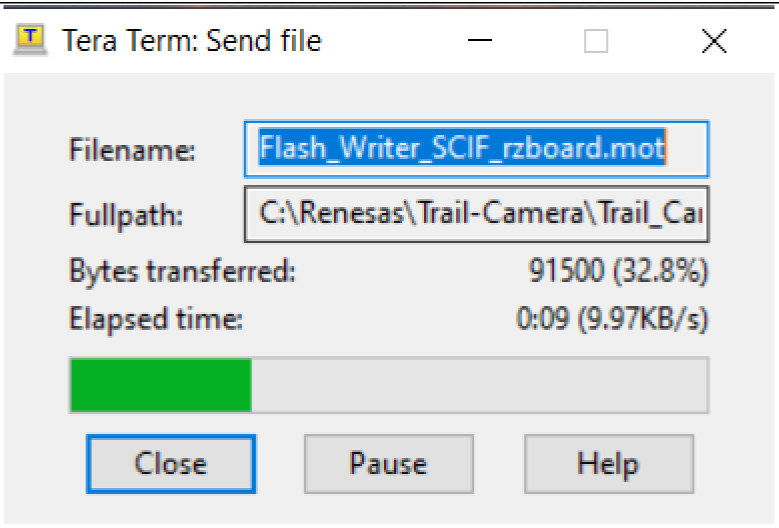


images	10/25/2023 5:47 PM	File folder	
tools	10/26/2023 6:39 PM	File folder	
Windows+Fastboot+Drivers-+droidwin	10/26/2023 6:39 PM	File folder	
config.ini	10/25/2023 5:26 PM	Configuration settings	1 KB
README.md	10/26/2023 6:39 PM	MD File	0 KB
usb_bootloader.bat	10/25/2023 5:40 PM	Windows Batch File	2 KB
usb_rootfs_trail_cam.bat	3 PM	Windows Batch File	2 KB

Type: Windows Batch File
Size: 1.33 KB
Date modified: 10/25/2023 5:40 PM

Figure 30. Select the script 'usb_bootloader.bat' to run.

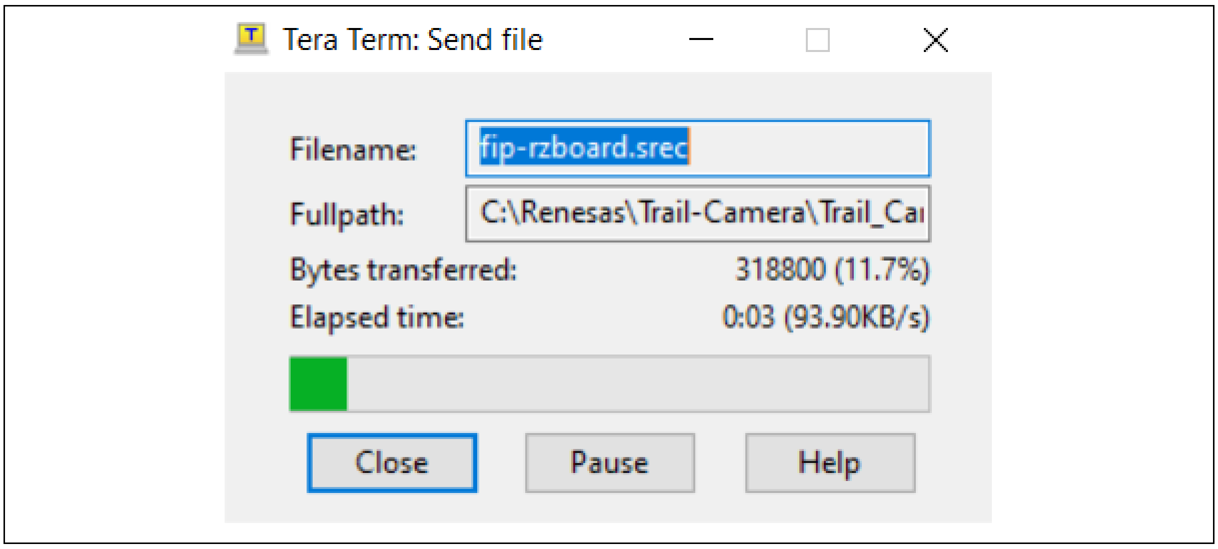
10. Power on the board and observe the host pc running the flashing commands. This stage uses UART file transfer to load bootloaders onto the QSPI or flash.
11. The Tool will automatically run and load a flash writer binary, which in turn loads and flashes three binaries: the flash writer utility, ATF, and u-boot, one after the other.



Tera Term: Send file

Filename: Flash_Writer_SCIF_rzboard.mot
Fullpath: C:\Renesas\Trail-Camera\Trail_Cai
Bytes transferred: 91500 (32.8%)
Elapsed time: 0:09 (9.97KB/s)

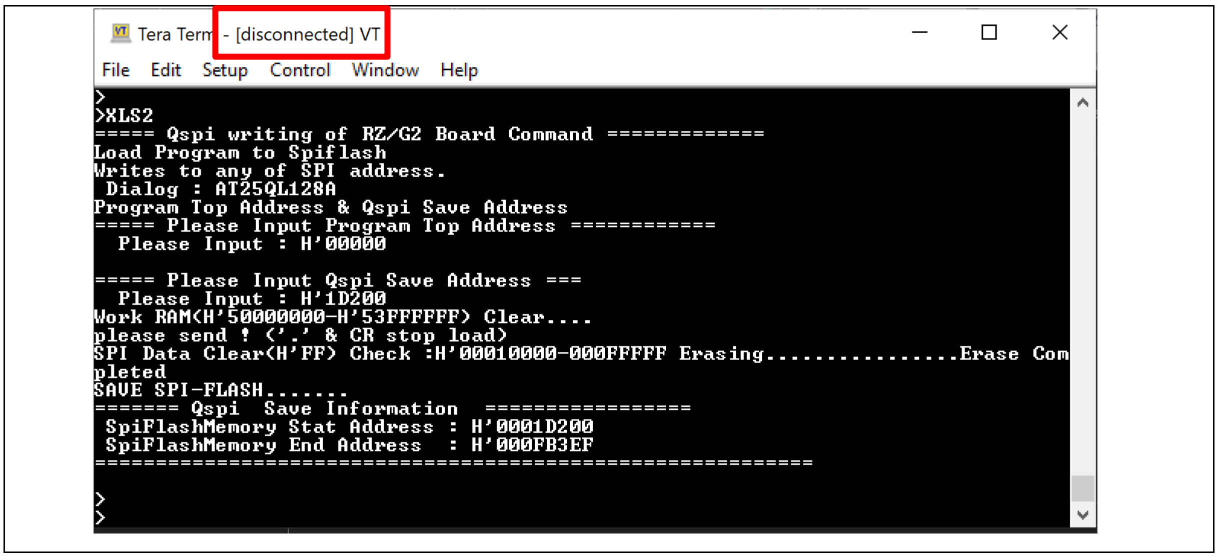
Figure 31. UART load of Flash Writer binary



Filename: fip-rzboard.srec
Fullpath: C:\Renesas\Trail-Camera\Trail_Ca
Bytes transferred: 318800 (11.7%)
Elapsed time: 0:03 (93.90KB/s)

Figure 32. UART Load of fip binary.

12. Wait for it to complete. Once it's done flashing, close the Tera Term. You should see a print like the one below.


Tera Term - [disconnected] VT
File Edit Setup Control Window Help

```
>
>XLS2
===== Qspi writing of RZ/G2 Board Command ============
Load Program to Spiflash
Writes to any of SPI address.
 Dialog : AT25QL128A
Program Top Address & Qspi Save Address
===== Please Input Program Top Address ============
  Please Input : H'00000

===== Please Input Qspi Save Address ===
  Please Input : H'1D200
Work RAM(H'50000000-H'53FFFFFF) Clear....
please send ! ('.' & CR stop load)
SPI Data Clear(H'FF) Check :H'00010000-000FFFFF Erasing................Erase Com
pleted
SAVE SPI-FLASH.......
======= Qspi  Save Information  =================
 SpiFlashMemory Stat Address : H'0001D200
 SpiFlashMemory End Address  : H'000FB3EF
===========================================================

>
>
```

Figure 33. UART flashing console debug info from the board.

13. Power off the board.

14. Remove the jumper wire you inserted in step 8. Proceed to set up the hardware connection as shown below by changing the boot switch settings to boot from QSPI.

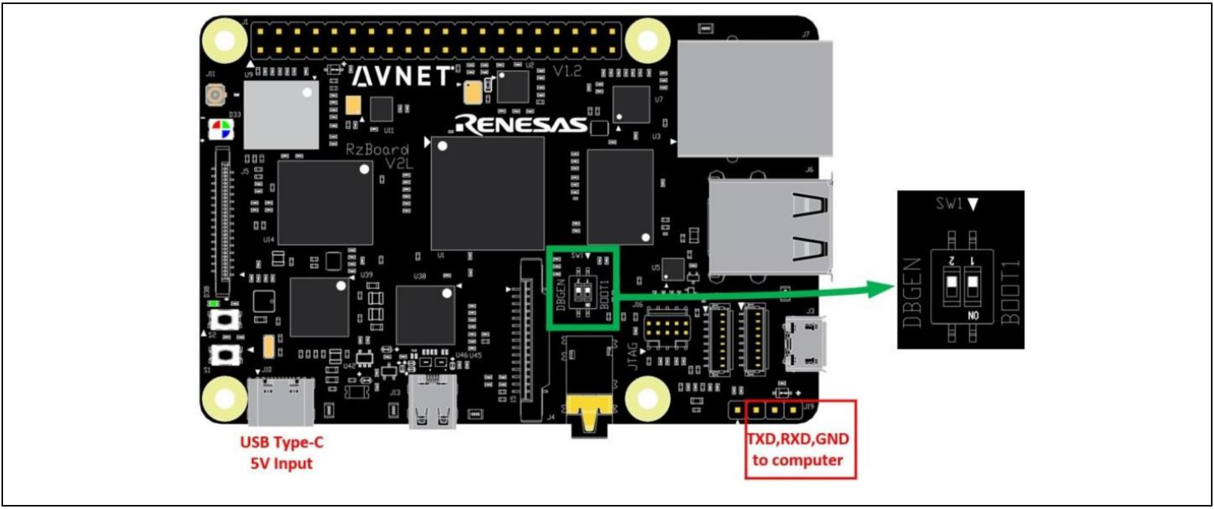
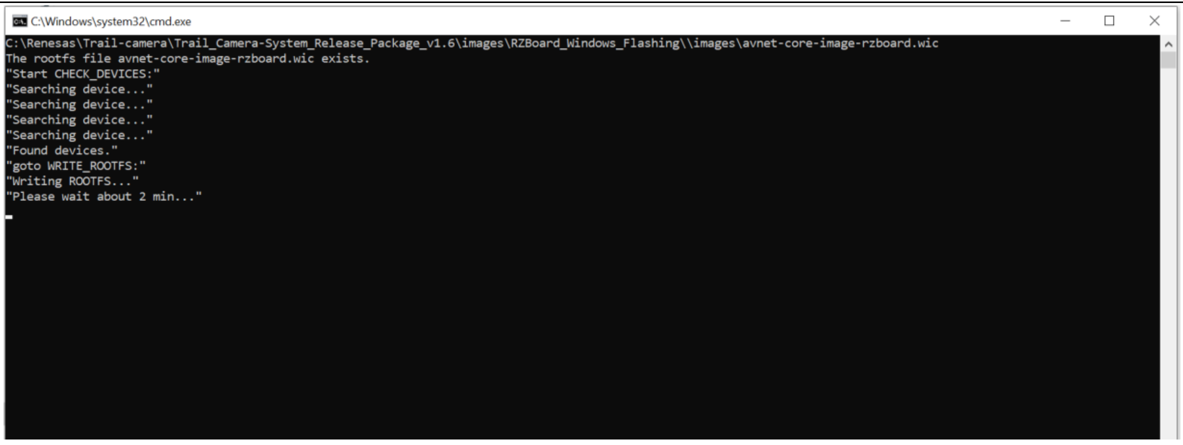


Figure 34. RZBoard normal boot hardware configuration.

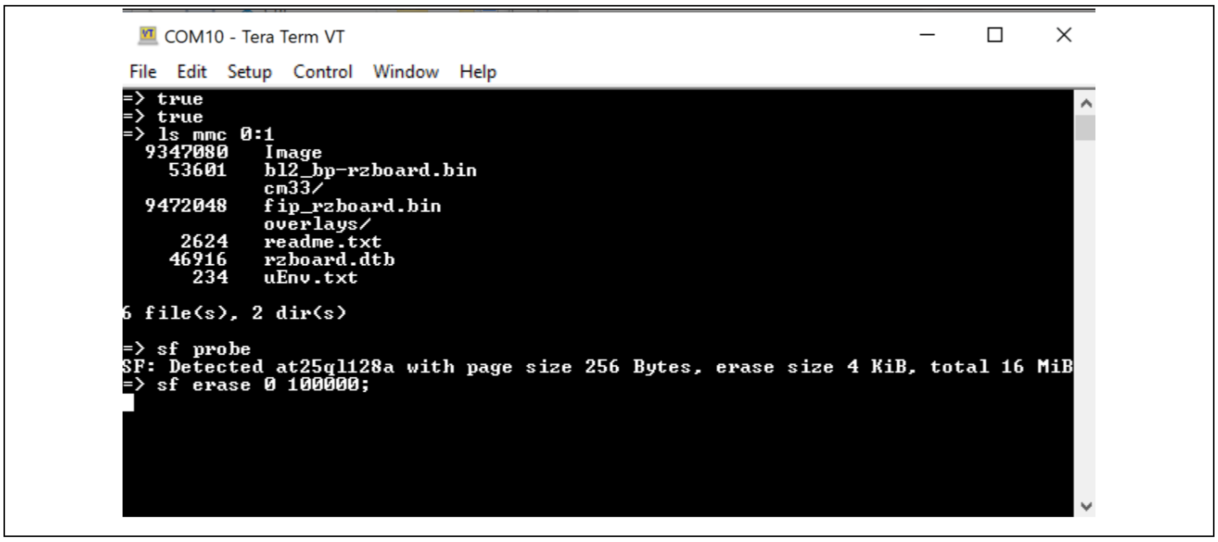
15. Next, run the batch script ‘usb_rootfs_trail_cam.bat.’
16. Power on the RZBoard.
17. It will start flashing the system image via a USB port.



```
C:\Renesas\Trail-camera\Trail_Camera-System_Release_Package_v1.6\images\RZBoard_Windows_Flashing\\images\avnet-core-image-rzboard.wic
The rootfs file avnet-core-image-rzboard.wic exists.
"Start CHECK_DEVICES:"
"Searching device..."
"Searching device..."
"Searching device..."
"Searching device..."
"Found devices."
"goto WRITE_ROOTFS:"
"Writing ROOTFS..."
"Please wait about 2 min..."
```

Figure 35. Command Windows as seen from a Windows host.

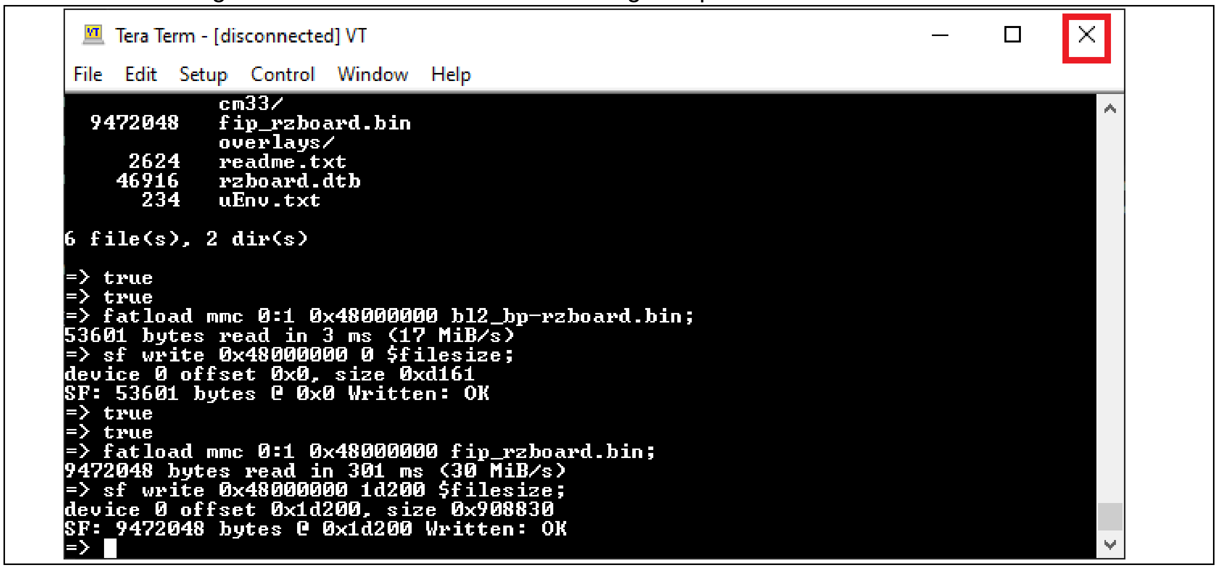
18. In a couple of minutes, it will complete the image flash onto the emmc.
19. After the system image flashing is completed, IPL and ATF will be written into QSPI flash.



```
=> true
=> true
=> ls mmc 0:1
  9347080   Image
    53601   bl2_bp-rzboard.bin
            cm33/
  9472048   fip_rzboard.bin
            overlays/
     2624   readme.txt
    46916   rzboard.dtb
      234   uEnv.txt

6 file(s), 2 dir(s)

=> sf probe
SF: Detected at25ql128a with page size 256 Bytes, erase size 4 KiB, total 16 MiB
=> sf erase 0 100000;
```

Figure 36. SQPI flash is erased to prepare for flashing optimized IPLs

20. Tera Term has logs from the board that show flashing completed.

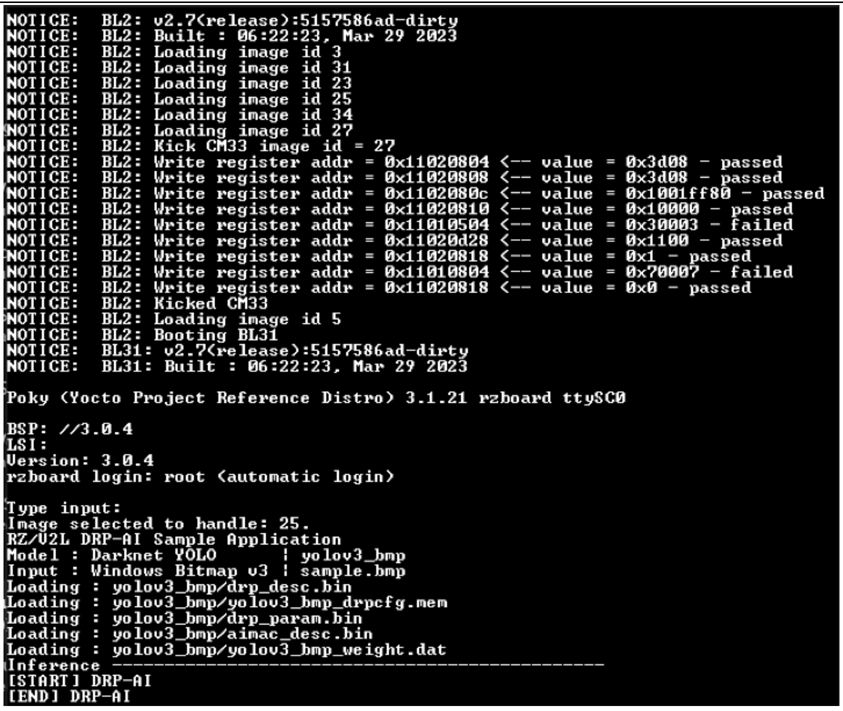

Tera Term - [disconnected] VT
File Edit Setup Control Window Help

```
            cm33/
  9472048   fip_rzboard.bin
            overlays/
     2624   readme.txt
    46916   rzboard.dtb
      234   uEnv.txt

6 file(s), 2 dir(s)

=> true
=> true
=> fatload mmc 0:1 0x48000000 bl2_bp-rzboard.bin;
53601 bytes read in 3 ms (17 MiB/s)
=> sf write 0x48000000 0 $filesize;
device 0 offset 0x0, size 0xd161
SF: 53601 bytes @ 0x0 Written: OK
=> true
=> true
=> fatload mmc 0:1 0x48000000 fip_rzboard.bin;
9472048 bytes read in 301 ms (30 MiB/s)
=> sf write 0x48000000 1d200 $filesize;
device 0 offset 0x1d200, size 0x908830
SF: 9472048 bytes @ 0x1d200 Written: OK
=>
```

Figure 37. Tera Term output showing RZBoard flashing complete.

Close the Tera Term.

21. Power off and then power on the board.
22. It will boot normally, and the TTL UART will show boot up of the log that looks like the following image:



```
NOTICE:  BL2: v2.7(release):5157586ad-dirty
NOTICE:  BL2: Built : 06:22:23, Mar 29 2023
NOTICE:  BL2: Loading image id 3
NOTICE:  BL2: Loading image id 31
NOTICE:  BL2: Loading image id 23
NOTICE:  BL2: Loading image id 25
NOTICE:  BL2: Loading image id 34
NOTICE:  BL2: Loading image id 27
NOTICE:  BL2: Kick CM33 image id = 27
NOTICE:  BL2: Write register addr = 0x11020804 <-- value = 0x3d08 - passed
NOTICE:  BL2: Write register addr = 0x11020808 <-- value = 0x3d08 - passed
NOTICE:  BL2: Write register addr = 0x1102080c <-- value = 0x1001ff80 - passed
NOTICE:  BL2: Write register addr = 0x11020810 <-- value = 0x10000 - passed
NOTICE:  BL2: Write register addr = 0x11010504 <-- value = 0x30003 - failed
NOTICE:  BL2: Write register addr = 0x11020d28 <-- value = 0x1100 - passed
NOTICE:  BL2: Write register addr = 0x11020818 <-- value = 0x1 - passed
NOTICE:  BL2: Write register addr = 0x11010804 <-- value = 0x70007 - failed
NOTICE:  BL2: Write register addr = 0x11020818 <-- value = 0x0 - passed
NOTICE:  BL2: Kicked CM33
NOTICE:  BL2: Loading image id 5
NOTICE:  BL2: Booting BL31
NOTICE:  BL31: v2.7(release):5157586ad-dirty
NOTICE:  BL31: Built : 06:22:23, Mar 29 2023

Poky (Yocto Project Reference Distro) 3.1.21 rzboard ttySC0

BSP: //3.0.4
LSI:
Version: 3.0.4
rzboard login: root (automatic login)

Type input:
Image selected to handle: 25.
RZ/V2L DRP-AI Sample Application
Model : Darknet YOLO       | yolov3_bmp
Input : Windows Bitmap v3  | sample.bmp
Loading : yolov3_bmp/drp_desc.bin
Loading : yolov3_bmp/yolov3_bmp_drpcfg.mem
Loading : yolov3_bmp/drp_param.bin
Loading : yolov3_bmp/aimac_desc.bin
Loading : yolov3_bmp/yolov3_bmp_weight.dat
Inference ----------------------------------------------------
[START] DRP-AI
[END] DRP-AI
```

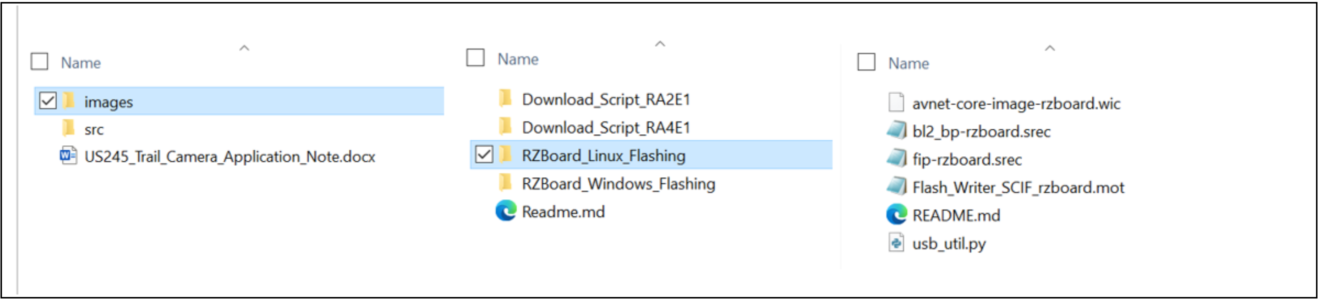
Figure 38. Tera Term console showing RZBoard boot console.

23. This completes our flashing process on Windows.

9.1.3 Flashing procedure on Linux

The Release package also contains a directory named ‘RZBoard_Linux_Flashing.’ This directory contains the images and scripts necessary to successfully program the RZBoard on Linux OS.

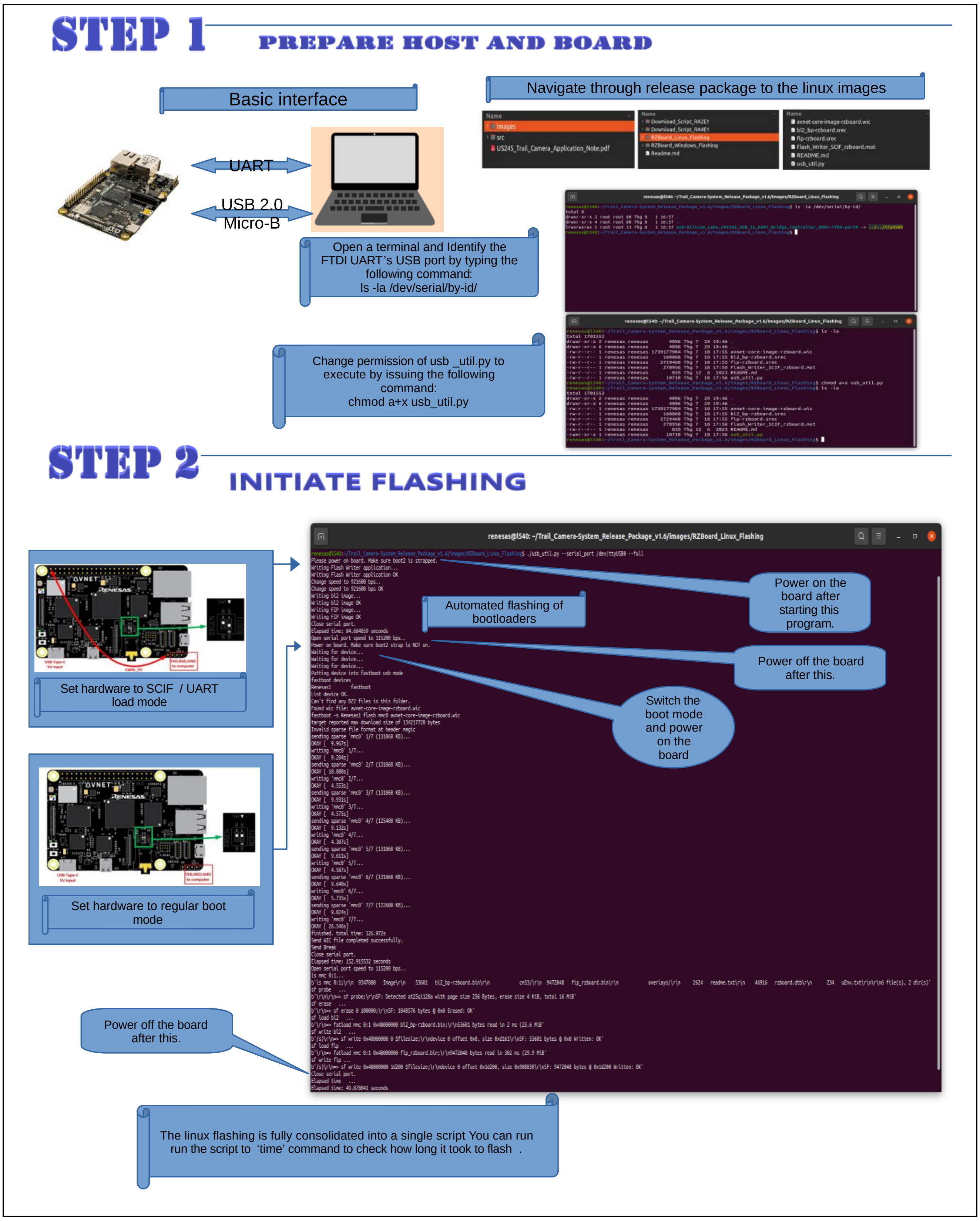
Refer to the image below to navigate to the prebuilt folder starting from the root of the release package.



```
Name
[✓] images
    src
    US245_Trail_Camera_Application_Note.docx

Name
    Download_Script_RA2E1
    Download_Script_RA4E1
[✓] RZBoard_Linux_Flashing
    RZBoard_Windows_Flashing
    Readme.md

Name
    avnet-core-image-rzboard.wic
    bl2_bp-rzboard.srec
    fip-rzboard.srec
    Flash_Writer_SCIF_rzboard.mot
    README.md
    usb_util.py
```

Figure 39. Directory flow from root of the release package to Linux flashing directory.

Below is a summary of the flashing process.


STEP 1
PREPARE HOST AND BOARD
Basic interface
UART
USB 2.0
Micro-B
Navigate through release package to the linux images
Open a terminal and Identify the FTDI UART's USB port by typing the following command:
ls -la /dev/serial/by-id/
Change permission of usb _util.py to execute by issuing the following command:
chmod a+x usb_util.py
STEP 2
INITIATE FLASHING
Set hardware to SCIF / UART load mode
Set hardware to regular boot mode
Automated flashing of bootloaders
Power on the board after starting this program.
Power off the board after this.
Switch the boot mode and power on the board
Power off the board after this.
The linux flashing is fully consolidated into a single script You can run run the script to 'time' command to check how long it took to flash .

```
Name
images
src
US245_Trail_Camera_Application_Note.pdf

Name
Download_Script_RA2E1
Download_Script_RA4E1
RZBoard_Linux_Flashing
RZBoard_Windows_Flashing
Readme.md

Name
avnet-core-image-rzboard.wic
bl2_bp-rzboard.srec
fip-rzboard.srec
Flash_Writer_SCIF_rzboard.mot
README.md
usb_util.py
```

```
renesas@l540: ~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$ ls -la /dev/serial/by-id/
total 0
drwxr-xr-x 2 root root 60 Thg 8   1 16:57 .
drwxr-xr-x 4 root root 80 Thg 8   1 16:57 ..
lrwxrwxrwx 1 root root 13 Thg 8   1 16:57 usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0 -> ../../ttyUSB0
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$
```

```
renesas@l540: ~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$ ls -la
total 1701552
drwxr-xr-x 2 renesas renesas       4096 Thg 7  29 19:46 .
drwxr-xr-x 6 renesas renesas       4096 Thg 7  29 19:46 ..
-rw-r--r-- 1 renesas renesas 1739177984 Thg 7  18 17:55 avnet-core-image-rzboard.wic
-rw-r--r-- 1 renesas renesas     160868 Thg 7  18 17:55 bl2_bp-rzboard.srec
-rw-r--r-- 1 renesas renesas    2729468 Thg 7  18 17:55 fip-rzboard.srec
-rw-r--r-- 1 renesas renesas     278956 Thg 7  18 17:56 Flash_Writer_SCIF_rzboard.mot
-rw-r--r-- 1 renesas renesas        835 Thg 12  6  2023 README.md
-rw-r--r-- 1 renesas renesas      10710 Thg 7  18 17:56 usb_util.py
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$ chmod a+x usb_util.py
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$ ls -la
total 1701552
drwxr-xr-x 2 renesas renesas       4096 Thg 7  29 19:46 .
drwxr-xr-x 6 renesas renesas       4096 Thg 7  29 19:46 ..
-rw-r--r-- 1 renesas renesas 1739177984 Thg 7  18 17:55 avnet-core-image-rzboard.wic
-rw-r--r-- 1 renesas renesas     160868 Thg 7  18 17:55 bl2_bp-rzboard.srec
-rw-r--r-- 1 renesas renesas    2729468 Thg 7  18 17:55 fip-rzboard.srec
-rw-r--r-- 1 renesas renesas     278956 Thg 7  18 17:56 Flash_Writer_SCIF_rzboard.mot
-rw-r--r-- 1 renesas renesas        835 Thg 12  6  2023 README.md
-rwxr-xr-x 1 renesas renesas      10710 Thg 7  18 17:56 usb_util.py
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$
```

```
renesas@l540: ~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$ ./usb_util.py --serial_port /dev/ttyUSB0 --full
Please power on board. Make sure boot2 is strapped.
Writing Flash Writer application...
Writing Flash Writer application OK
Change speed to 921600 bps..
Change speed to 921600 bps OK
Writing bl2 image...
Writing bl2 image OK
Writing FIP image...
Writing FIP image OK
Close serial port.
Elapsed time: 84.684859 seconds
Open serial port speed to 115200 bps..
Power on board. Make sure boot2 strap is NOT on.
Waiting for device...
Waiting for device...
Waiting for device...
Putting device into fastboot usb mode
fastboot devices
Renesas1        fastboot
List device OK.
Can't find any BZ2 files in this folder.
Found wic file: avnet-core-image-rzboard.wic
fastboot -s Renesas1 flash mmc0 avnet-core-image-rzboard.wic
target reported max download size of 134217728 bytes
Invalid sparse file format at header magic
sending sparse 'mmc0' 1/7 (131068 KB)...
OKAY [  9.967s]
writing 'mmc0' 1/7...
OKAY [  9.204s]
sending sparse 'mmc0' 2/7 (131068 KB)...
OKAY [ 10.080s]
writing 'mmc0' 2/7...
OKAY [  4.553s]
sending sparse 'mmc0' 3/7 (131068 KB)...
OKAY [  9.931s]
writing 'mmc0' 3/7...
OKAY [  4.575s]
sending sparse 'mmc0' 4/7 (125408 KB)...
OKAY [  9.132s]
writing 'mmc0' 4/7...
OKAY [  4.387s]
sending sparse 'mmc0' 5/7 (131068 KB)...
OKAY [  9.611s]
writing 'mmc0' 5/7...
OKAY [  4.587s]
sending sparse 'mmc0' 6/7 (131068 KB)...
OKAY [  9.640s]
writing 'mmc0' 6/7...
OKAY [  5.735s]
sending sparse 'mmc0' 7/7 (122600 KB)...
OKAY [  9.024s]
writing 'mmc0' 7/7...
OKAY [ 26.546s]
finished. total time: 126.972s
Send WIC file completed successfully.
Send Break
Close serial port.
Elapsed time: 152.915532 seconds
Open serial port speed to 115200 bps..
ls mmc 0:1...
b'ls mmc 0:1;\r\n  9347080   Image\r\n    53601   bl2_bp-rzboard.bin\r\n           cm33/\r\n  9472048   fip_rzboard.bin\r\n           overlays/\r\n    2624   readme.txt\r\n   46916   rzboard.dtb\r\n     234   uEnv.txt\r\n\r\n6 file(s), 2 dir(s)'
sf probe   ...
b'\r\n\r\n=> sf probe;\r\nSF: Detected at25ql128a with page size 256 Bytes, erase size 4 KiB, total 16 MiB'
sf erase   ...
b'\r\n=> sf erase 0 100000;\r\nSF: 1048576 bytes @ 0x0 Erased: OK'
sf load bl2   ...
b'\r\n=> fatload mmc 0:1 0x48000000 bl2_bp-rzboard.bin;\r\n53601 bytes read in 2 ms (25.6 MiB'
sf write bl2   ...
b'/s)\r\n=> sf write 0x48000000 0 $filesize;\r\ndevice 0 offset 0x0, size 0xd161\r\nSF: 53601 bytes @ 0x0 Written: OK'
sf load fip   ...
b'\r\n=> fatload mmc 0:1 0x48000000 fip_rzboard.bin;\r\n9472048 bytes read in 302 ms (29.9 MiB'
sf write fip ...
b'/s)\r\n=> sf write 0x48000000 1d200 $filesize;\r\ndevice 0 offset 0x1d200, size 0x908830\r\nSF: 9472048 bytes @ 0x1d200 Written: OK'
Close serial port.
Elapsed time   ...
Elapsed time: 49.878041 seconds
```

Figure 40. RZBoard Linux flashing process summary

The following is the flashing procedure.

1. Before performing image flashing, ensure all necessary packages are installed on the Linux host PC. See the Appendix section for more details.
2. Connect the UART TTL cable to the Avnet board's UART port.
3. Connect the USB to the micro-USB cable between the RZBoard and the Linux host PC.
4. Identify the TTY COM port number on the Linux host PC and check it by running the following command.

```
$ ls -la /dev/serial/by-id/*
```

5. The TTY COM port (USB-serial in Linux) is generally named **"/dev/ttyUSB*"** where * is replaced by the port's number. You use the cute-com application to test the uart.
6. The USB otg port will automatically be detected by the scripts.
7. Make sure the RZBoard is powered off.
8. Set the board configuration to put the RZBoard into SCIF download mode, initiating an image load from the UART port. Use a jumper wire between the pins as indicated in the diagram below and ensure that the switch is set as shown in the image.

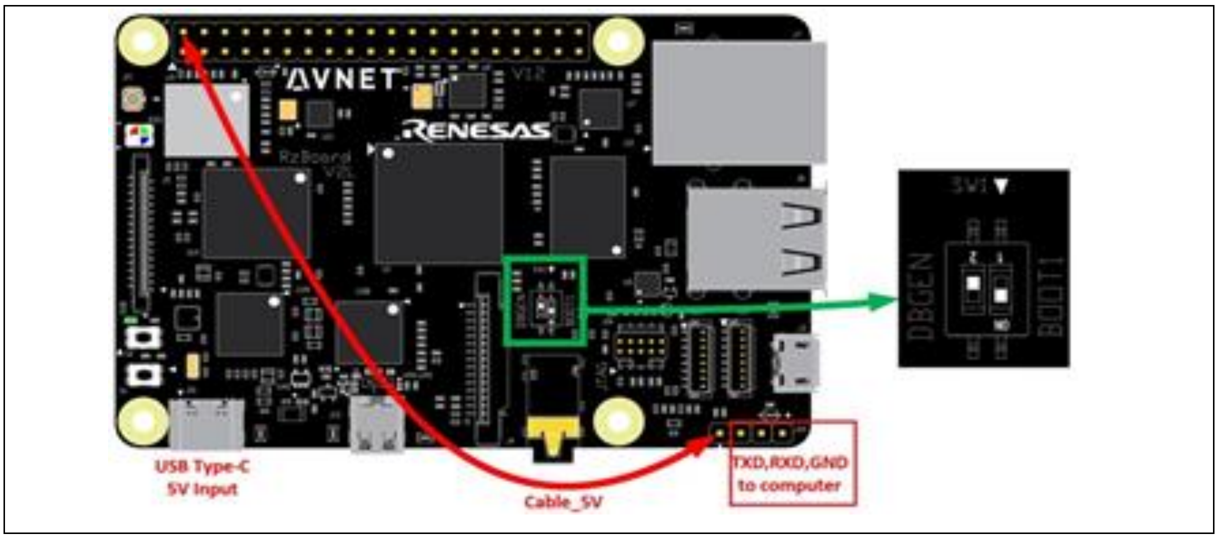


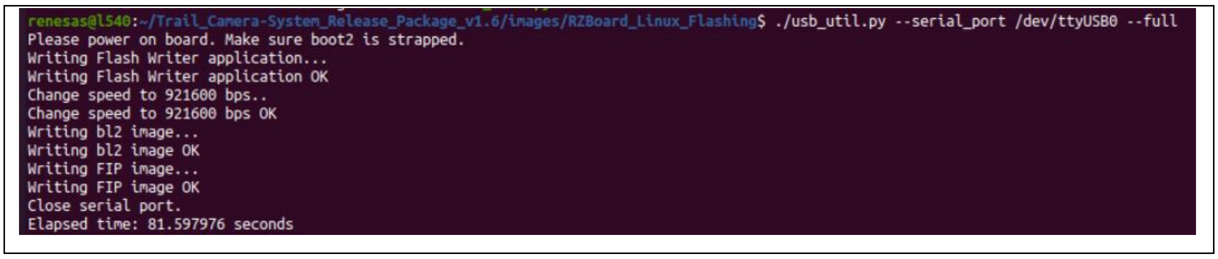
Figure 41. RZBoard scif (UART load) mode hardware configuration

9. On the Linux host machine, execute the following commands.

```
$ chmod a+x usb_util.py
$ ./usb_util.py --serial_port <serial_port> --full
```

`<serial_port>`: must be replaced by the actual serial port on board, listed in Step 4. If no `<serial_port>` is addressed, the Python script will use the default port '`/dev/ttyUSB0`'.

10. Power on the board and observe the host pc running the flashing commands. This stage uses UART file transfer to load bootloaders onto the QSPI or flash.
11. The Tool will automatically run and load a flash writer binary, which in turn loads and flashes three binaries: the flash writer utility, ATF, and u-boot, one after the other.
12. Wait for it to complete. Once it's done flashing, you should see a print like the one below.



```
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$ ./usb_util.py --serial_port /dev/ttyUSB0 --full
Please power on board. Make sure boot2 is strapped.
Writing Flash Writer application...
Writing Flash Writer application OK
Change speed to 921600 bps..
Change speed to 921600 bps OK
Writing bl2 image...
Writing bl2 image OK
Writing FIP image...
Writing FIP image OK
Close serial port.
Elapsed time: 81.597976 seconds
```

Figure 42. UART load of Flash Writer, ATF, and U-Boot binary

13. Power off the board. The script on the Linux Host PC still runs, and it shows 'Waiting for device…' on the terminal.
14. Remove the jumper wire you inserted in step 8. Set up the hardware connection as shown below by changing the boot switch settings to boot from QSPI.

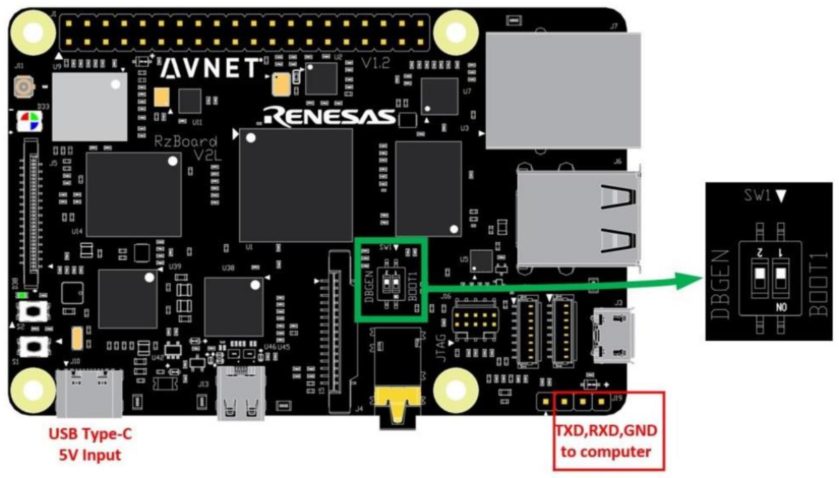
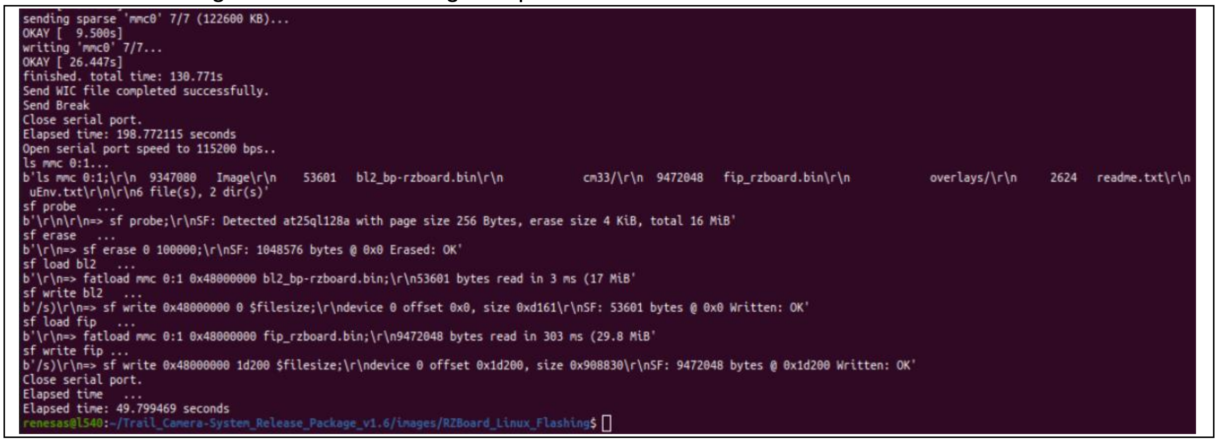


Figure 43. RZBoard normal boot hardware configuration.

15. Next, power on the RZBoard.
16. It will start flashing the system image and optimized IPLs via a USB port.
17. In a couple of minutes, it will complete the images flash onto the emmc and QSPI flash.
18. The terminal has logs that show flashing completed.



```
sending sparse 'mmc0' 7/7 (122600 KB)...
OKAY [  9.500s]
writing 'mmc0' 7/7...
OKAY [ 26.447s]
finished. total time: 130.771s
Send WIC file completed successfully.
Send Break
Close serial port.
Elapsed time: 198.772115 seconds
Open serial port speed to 115200 bps..
ls mmc 0:1...
b'ls mmc 0:1;\r\n  9347080   Image\r\n    53601   bl2_bp-rzboard.bin\r\n           cm33/\r\n  9472048   fip_rzboard.bin\r\n           overlays/\r\n     2624   readme.txt\r\n
 uEnv.txt\r\n\r\n6 file(s), 2 dir(s)'
sf probe   ...
b'\r\n\r\n=> sf probe;\r\nSF: Detected at25ql128a with page size 256 Bytes, erase size 4 KiB, total 16 MiB'
sf erase   ...
b'\r\n=> sf erase 0 100000;\r\nSF: 1048576 bytes @ 0x0 Erased: OK'
sf load bl2   ...
b'\r\n=> fatload mmc 0:1 0x48000000 bl2_bp-rzboard.bin;\r\n53601 bytes read in 3 ms (17 MiB'
sf write bl2   ...
b'/s)\r\n=> sf write 0x48000000 0 $filesize;\r\ndevice 0 offset 0x0, size 0xd161\r\nSF: 53601 bytes @ 0x0 Written: OK'
sf load fip   ...
b'\r\n=> fatload mmc 0:1 0x48000000 fip_rzboard.bin;\r\n9472048 bytes read in 303 ms (29.8 MiB'
sf write fip ...
b'/s)\r\n=> sf write 0x48000000 1d200 $filesize;\r\ndevice 0 offset 0x1d200, size 0x908830\r\nSF: 9472048 bytes @ 0x1d200 Written: OK'
Close serial port.
Elapsed time   ...
Elapsed time: 49.799469 seconds
renesas@l540:~/Trail_Camera-System_Release_Package_v1.6/images/RZBoard_Linux_Flashing$ 
```

Figure 44. Terminal output showing RZBoard flashing complete.

19. This completes our flashing process on Linux.

9.1.4 Programming the Sensing MCU with Prebuilt Images

The trail camera PoC usually uses the FPB-RA2E1. However, a different MCU board may be used in some select units. The general difference is in the flashing protocol and additional features supported on the board-SoC combination. Some boards, like the FPB-RA4E1, use JLink SWD, while the FPB-RA2E1 board uses a Renesas E2 Lite emulator circuit.

In most cases, the PoC will use FPB-RA2E1. While the prebuilt images are a great starting point, it is always recommended that the application be refactored using e^2 studio / Quick connect Studio or a combination of the two software to use the latest FSP and toolchain. The scripts provided are helpers, not the preferred method of flashing the MCU board. Using e^2 studio, as described in the section Programming RA2E1 FPB is the ideal method as it uses the project files first to build the application as described in Build RA2E1 MCU Package.

Additional tools like the Renesas Flash programmer and Segger JLinkCommander can be used to flash the board. These are beyond the scope of this document. The product links to Renesas Flash Programmer and JLink/J-Trace are as follows:

- Renesas Flash Programmer (Programming GUI) | Renesas
- SEGGER - The Embedded Experts - Downloads - J-Link / J-Trace

The PoC release package contains the images and all the flashing tools needed in the pre-built release directories under ‘images.’ While the flashing of the RA2E1 board is covered here, the alternative boards using JLink are covered in the appending section.

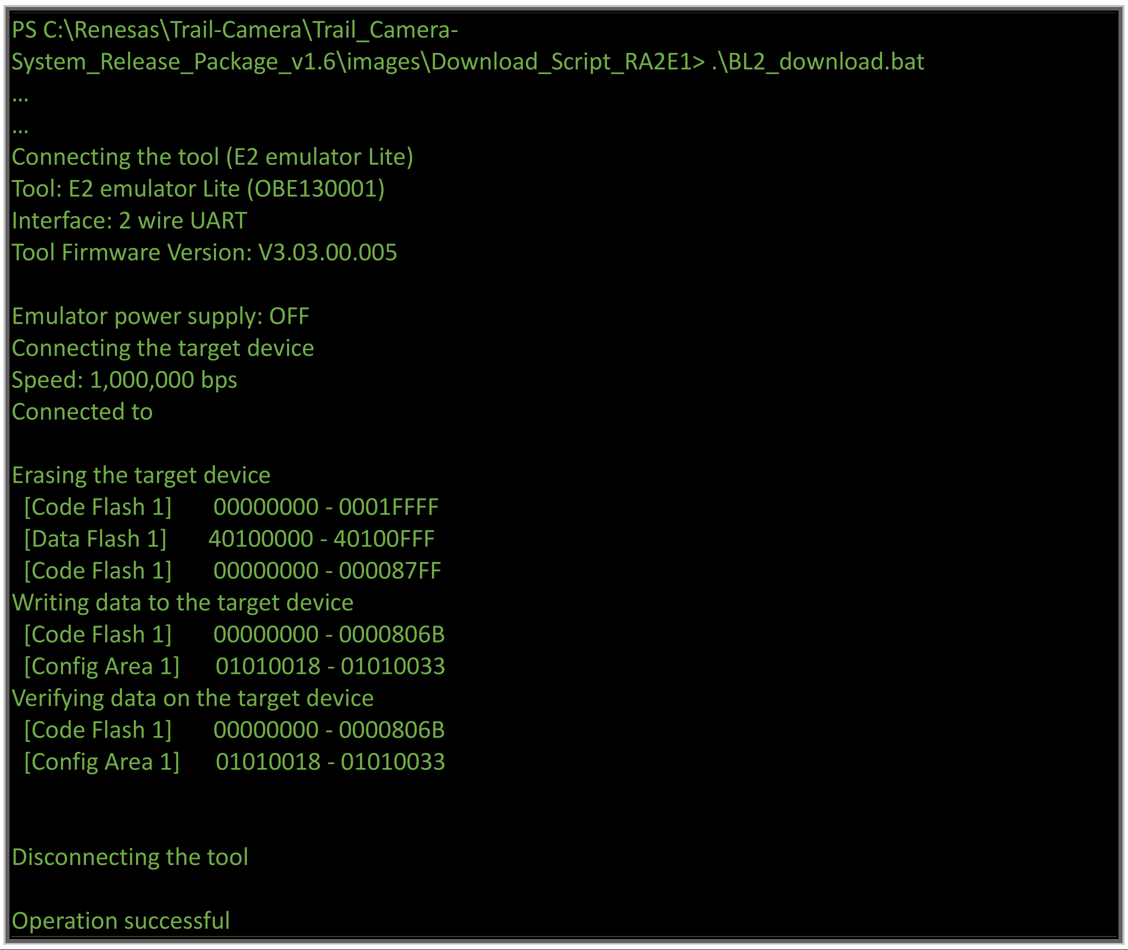
9.1.5 FPB-RA2E1 MCU board (E2 Lite boards)

The release package contains a directory named ‘Download_Script_RA2E1’. This folder contains the prebuilt image and a flashing script for the FPB-RA2E1 board. The Windows batch file ‘BL2_download.bat’ automatically flashes the connected RA2E1 FPB board with the program for motion sensor activation of the power switch. The program binary, Renesas flash programmer executable, and all dependencies are packaged in that folder.

The following is the flashing process for the MCU board using the script:

1. Change the jumper on the plain side of the FBP-RA2E1 board to flashing mode.
2. Connect the FPB-RA2E1 board to the Windows host machine using a USB 2.0 to micro-USB cable. The board will power on and be detected by the Windows device manager.
3. Run the ‘BL2_download.bat’ script in the command line or by directly double-clicking on it.

4. Check the log at the end.



```
PS C:\Renesas\Trail-Camera\Trail_Camera-
System_Release_Package_v1.6\images\Download_Script_RA2E1> .\BL2_download.bat
…
…
Connecting the tool (E2 emulator Lite)
Tool: E2 emulator Lite (OBE130001)
Interface: 2 wire UART
Tool Firmware Version: V3.03.00.005

Emulator power supply: OFF
Connecting the target device
Speed: 1,000,000 bps
Connected to

Erasing the target device
  [Code Flash 1]      00000000 - 0001FFFF
  [Data Flash 1]      40100000 - 40100FFF
  [Code Flash 1]      00000000 - 000087FF
Writing data to the target device
  [Code Flash 1]      00000000 - 0000806B
  [Config Area 1]     01010018 - 01010033
Verifying data on the target device
  [Code Flash 1]      00000000 - 0000806B
  [Config Area 1]     01010018 - 01010033


Disconnecting the tool

Operation successful
```

Figure 45. Consolidated Console log for MCU flashing script

5. Swap the jumper again to normal execution mode.

10. Appendix

10.1 Hardware Wiring Procedure

This section describes the wiring procedure for building your own PoC system from ordered parts. The PoC can be ordered premade or in parts. Premade units are usually back-ordered. Hence, it's more likely that a unit is built from parts. This is relatively simple and requires the bare minimum of skills.

For questions about ordering parts and components, please get in touch with sales@renesas.com.

10.2 Prerequisites:

The following parts are needed to build the Trail Camera PoC hardware:

1. Avnet RZBoard SBC
2. Renesas FPB-RA2E1
3. QCIOT-014 Motion Sensor Board
4. PMOD Motion on/off switchboard
5. Arducam OV5640 MIPI CSI2 camera module
6. Mechanical Mounting kit for holding the PoC together.

The following equipment is needed for this activity:

1. 6" Male to male jumper wires (4 wires).
2. Soldering station

10.3 PMOD Motion on/off-board

The motion on/off board is a simple button board that is optional but provides a helpful feature to save time in testing. The PoC takes 30 seconds to power down from boot and image processing. This button lets the user break the timer and immediately push the PoC to power down.

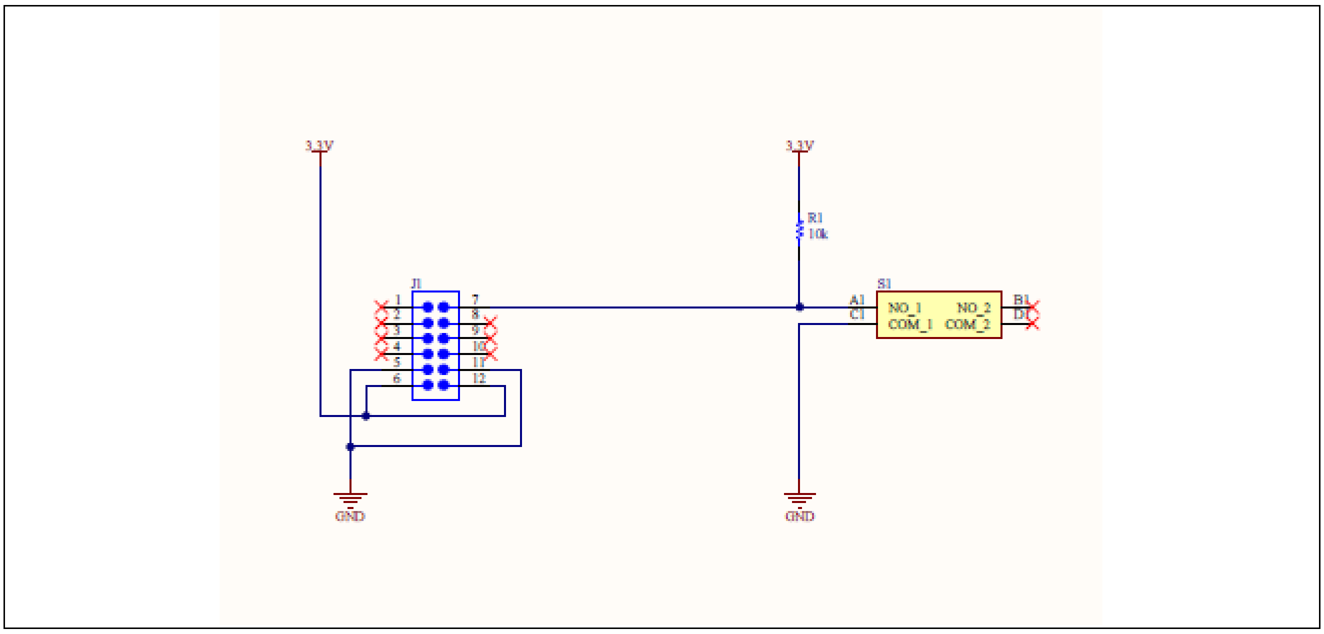


Figure 46. PMOD Button board schematic

The entire Altium design project is provided in this release of the PoC under the pcb directory. Ensure to unpackage the PMOD Motion On/Off Switch.7z before opening the Altium Design software. Otherwise, the Designer won't be able to find all the files needed.

10.4 Wiring Procedure

Step 1. Connect QCIOT-014 Motion Sensor board to PMOD1 on FPB-RA2E1 board.

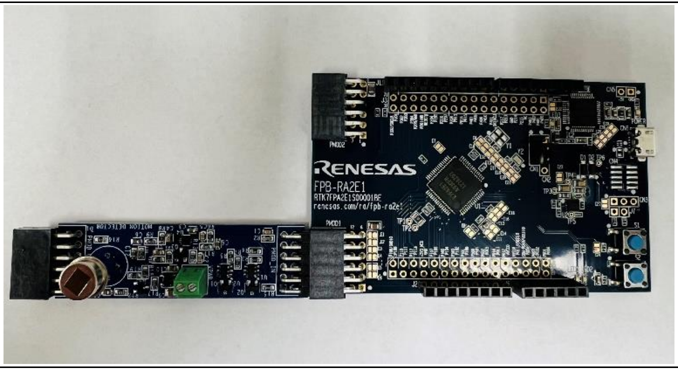


Figure 47. QCIOT-14 board connected to FPB-RA2E1 pmod1

Step 2. Connect the Switch to PMOD2 on FPB-RA2E1.

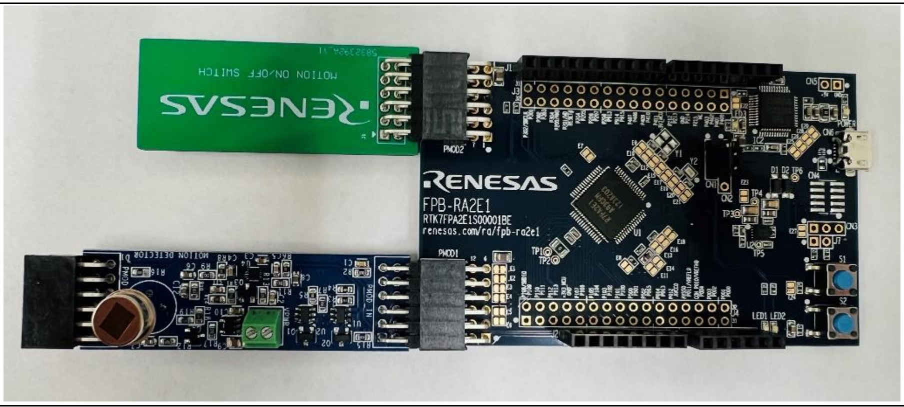


Figure 48. Button board as well as QCIOT-14 board connected to FPB-RA2E1 board

Step 3. Provide 5V-16V to Motion sensor board.

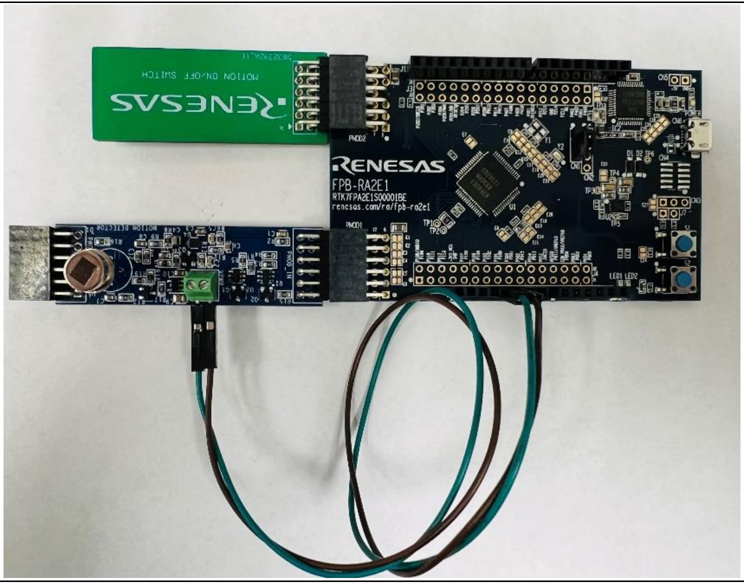


Figure 49. Wiring of power between QCIOT-14 and FPB-RA2E1 boards

To provide 5V, connect as follows:

Motion Board	FPB-RA4E1
Positive	J2-5
Negative	J2-6

Note: The dot mark near J2 indicates pin1.

Step 4. Connect a jumper wire on J5-8. Connect the other end to the CEN pin on the RZBoard.

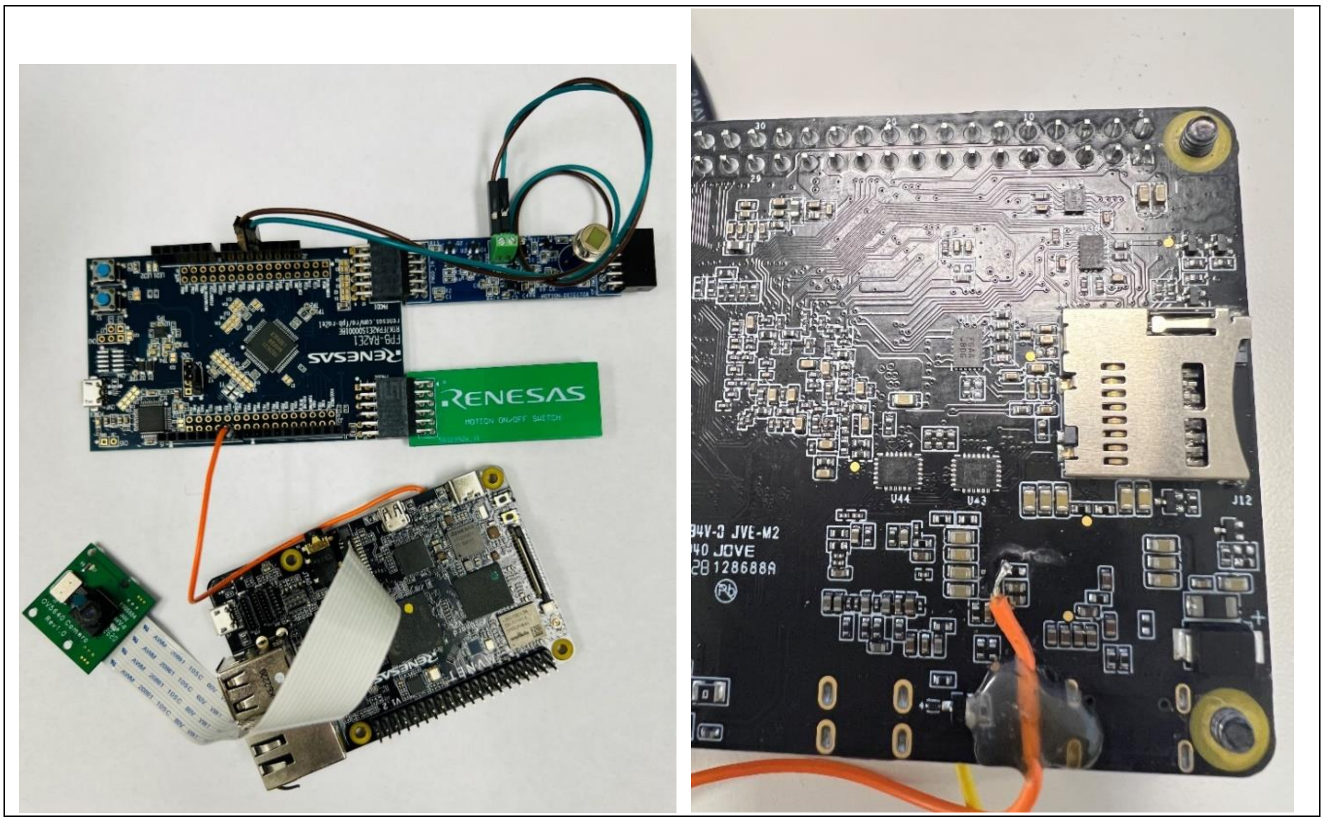


Figure 50. Complete wiring of FPB-RA2E1 to RZBoard CE-EN

Step 5. Short the SW1 switch on the RZBoard.

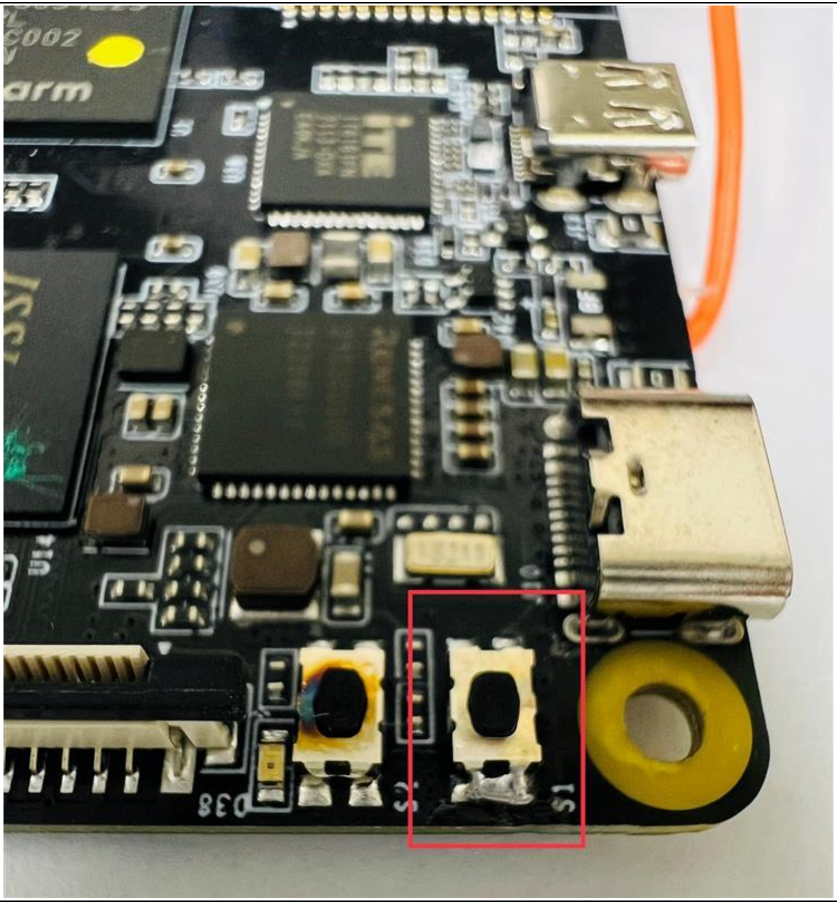


Figure 51. SW1 switch on RZBoard

Step 6. Short Pin 1-2 on CN1 connector.

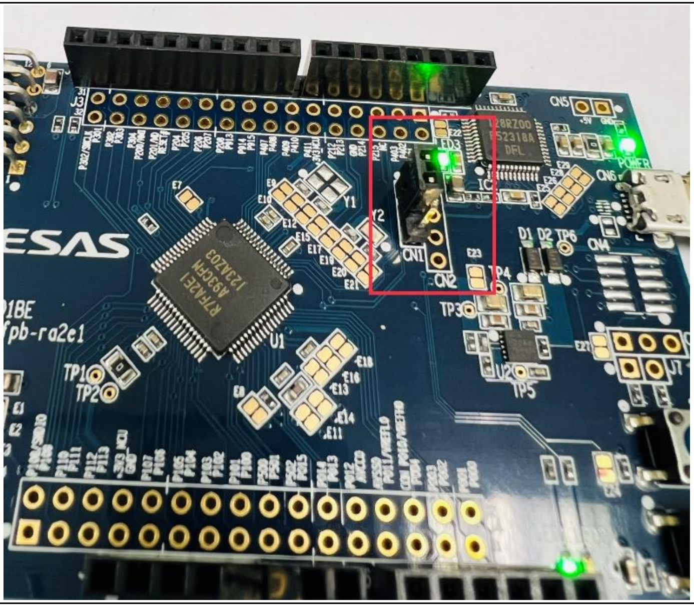


Figure 52. Pin 1-2 on CN1 connector

Step 7. Connect FPB-RA4E1 to the laptop using a USB cable.
Step 8. Flash the code on FPB-RA2E1.
Step 9. Short Pin 2-3 on CN1 connector.

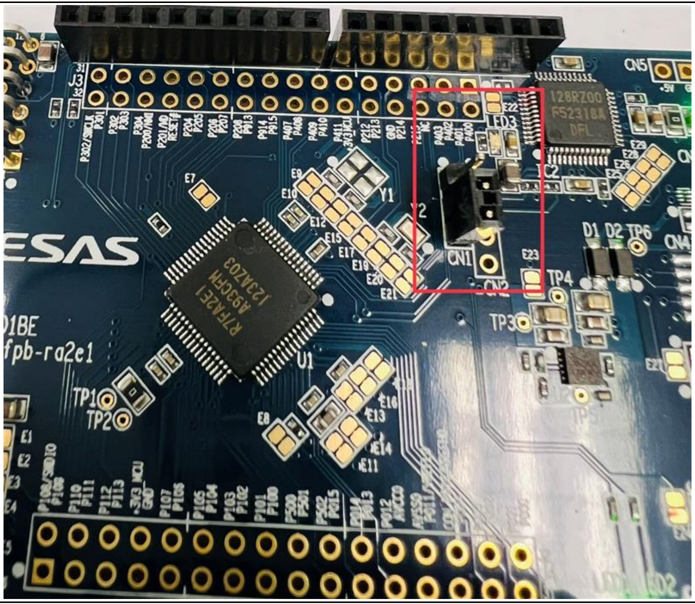
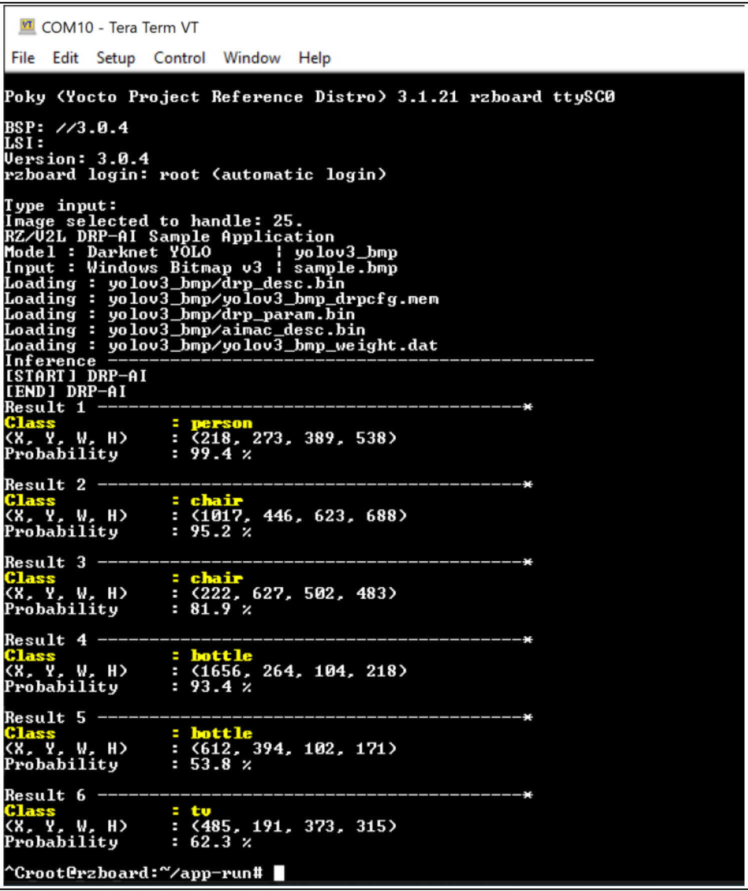


Figure 53. The FPB-RA2E1 and RZBoard

Step 10. Turn ON the FPB-RA2E1 and RZBoard.

10.5 How to get the console after bootup

Once the trail camera application has booted, pressing ‘ctrl+c’ on the UART terminal will break the script and give you the console.



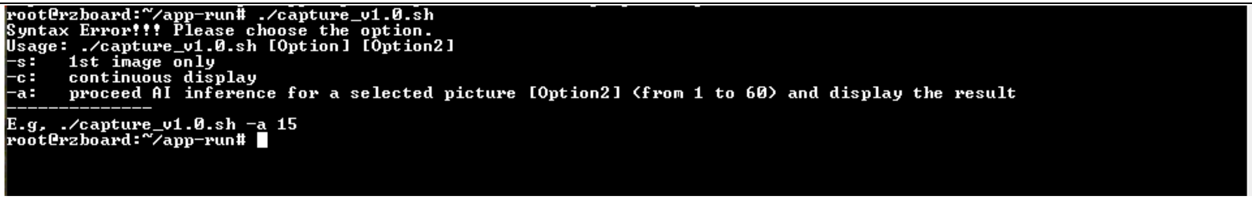
```
Poky (Yocto Project Reference Distro) 3.1.21 rzboard ttySC0

BSP: //3.0.4
LSI:
Version: 3.0.4
rzboard login: root (automatic login)

Type input:
Image selected to handle: 25.
RZ/V2L DRP-AI Sample Application
Model : Darknet YOLO       | yolov3_bmp
Input : Windows Bitmap v3  | sample.bmp
Loading : yolov3_bmp/drp_desc.bin
Loading : yolov3_bmp/yolov3_bmp_drpcfg.mem
Loading : yolov3_bmp/drp_param.bin
Loading : yolov3_bmp/aimac_desc.bin
Loading : yolov3_bmp/yolov3_bmp_weight.dat
Inference ----------------------------------------------
[START] DRP-AI
[END] DRP-AI
Result 1 -----------------------------------------*
Class          : person
(X, Y, W, H)   : (218, 273, 389, 538)
Probability    : 99.4 %

Result 2 -----------------------------------------*
Class          : chair
(X, Y, W, H)   : (1017, 446, 623, 688)
Probability    : 95.2 %

Result 3 -----------------------------------------*
Class          : chair
(X, Y, W, H)   : (222, 627, 502, 483)
Probability    : 81.9 %

Result 4 -----------------------------------------*
Class          : bottle
(X, Y, W, H)   : (1656, 264, 104, 218)
Probability    : 93.4 %

Result 5 -----------------------------------------*
Class          : bottle
(X, Y, W, H)   : (612, 394, 102, 171)
Probability    : 53.8 %

Result 6 -----------------------------------------*
Class          : tv
(X, Y, W, H)   : (485, 191, 373, 315)
Probability    : 62.3 %

^Croot@rzboard:~/app-run#
```

Figure 54. Drop into console post-bootup.

10.6 Application flow in details

The RZBoard image contains a script in /home/root/app-run/capture_v1.0.sh which does the following:

1. Checks for command line options and executes appropriate applications. It has two arguments:
    a. The first option can be:
        i. '-s': execute 'gst_app_wayland' using the same argument we got. This program shows what is captured in a slideshow.
        ii. '-c': execute 'gst_app_wayland' using the same argument we got. This program shows only the very first captured image.
        iii. '-a': execute 'sample_app_yolo_v3_img' which is the AI inference model. The second argument is passed on to this application, which tells the image to be used for inferencing. This is the 25th image by default.
2. Launches a 'gstreamer' command that takes the bitmap file from the previous step and displays it on the HDMI display.



```
root@rzboard:~/app-run# ./capture_v1.0.sh
Syntax Error!!! Please choose the option.
Usage: ./capture_v1.0.sh [Option] [Option2]
-s:    1st image only
-c:    continuous display
-a:    proceed AI inference for a selected picture [Option2] (from 1 to 60) and display the result
---------------
E.g. ./capture_v1.0.sh -a 15
root@rzboard:~/app-run#
```

Figure 55. capture_v1.sh command options

10.7 Demo Inference App

The sample application at /home/root/app-run/exe is called ‘sample_app_yolo_v3_img’. This application performs the following:

1. Get the selected image from the second argument of the ‘capture_v1.0.sh’ script.
2. Copies the appropriate DDR buffer of the selected image. It then creates a bitmap file called ‘sample.bmp’ from the acquired buffer.
3. Runs the AI inference by leveraging the DRP-AI module and its interface. The result of this inference is a bitmap file at the exact location called ‘sample_output.bmp.’ The bitmap file is located at the same location as the application: ' home/root/app-run/exe.’

10.8 Camera Operation

The camera module used in the trail camera application is the OV5640 mipi CSI camera module. One must remember that the camera is mounted to the chassis in reverse. This is to provide adequate safe routing to the delicate flex cable of the CSI interface. Hence, the images captured by the CM33 are inverted. The images captured are inverted by the inference application before inferencing. One must be mindful of this while writing their own application and/or modifying the base design. If you remove the OV5640 camera module from the housing and hold it straight, the AI model inference will not identify any objects. In this case, inverting the test image is the first quick fix to try.

You can access the camera through gstreamer commands. Refer to the RZBoard documentation for more info.

10.9 Flashing prebuilt images on FPB-RA4E1 MCU board (Segger JLink boards)

The release package contains a directory named ‘Download_Script_RA4E1’. This folder contains the prebuilt image and a flashing script for the FPB-RA4E1 board. The Windows batch file ‘BL2_download.bat’ automatically flashes the connected RA4E1 QCIOT board with the program for motion sensor activation of the power switch. The program binary, JLink executable, and all dependencies are packaged in that folder.

The actual JLink commands are in the file ‘s1.Jlink’. The first line in this file tells JLink to expect the R7FA4E10D device identifier. If the RA2E1 is replaced by any other pin-compatible MCU, modifying the first line should allow the same images and tools to work. However, it is recommended that the application be refactored using E2 Studio / Quick Connect Studio or a combination of the two software.

The following is the flashing process for the MCU board using a script:

1. Connect the FPB-RA4E1 board to the Windows host machine using a USB2.0 to micro-USB cable. The board will power on and get detected by the Windows device manager.
2. Run the ‘BL2_download.bat’ script in the command line or by directly double-clicking on it.

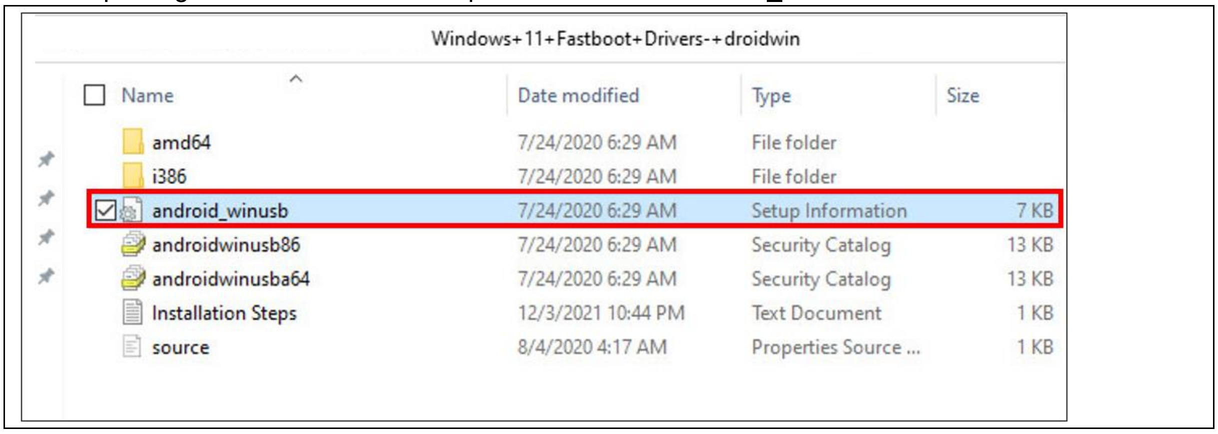
Check the log at the end.

```
SEGGER J-Link Commander V7.80 (Compiled Sep  9 2022 11:04:40)
DLL version V7.92a, compiled Aug 16 2023 15:34:08


J-Link Command File read successfully.
Processing script file...
J-Link>device R7FA4E10D
…

…
Hardware version: V1.00
J-Link uptime (since boot): 0d 00h 03m 28s
S/N: 831195433
USB speed mode: Full speed (12 MBit/s)
…
…
Target connection not established yet but required for command.
Device "R7FA4E10D" selected.
…
J-Link: Flash download: Total: 0.902s (Prepare: 0.183s, Compare: 0.019s, Erase: 0.157s, Program & Verify:
0.482s, Restore: 0.058s)
J-Link: Flash download: Program & Verify speed: 65 KB/s
O.K.
…
…
J-Link>g

Script processing completed.



Type "connect" to establish a target connection, '?' for help
J-Link>
```

10.10 Installing Fastboot USB driver on Windows

To program RZBoard images on Windows, ‘Android Fastboot USB driver’ is required to be installed on your Windows machine. The following are the step-by-step instructions to install the driver:

1. Navigate to ‘images\RZBoard_Windows_Flashing\Windows+Fastboot+Drivers-+droidwin\’ inside the release package. You will see the Setup Information file ‘android_winusb.inf.’



Windows+11+Fastboot+Drivers-+droidwin

Name	Date modified	Type	Size
amd64	7/24/2020 6:29 AM	File folder	
i386	7/24/2020 6:29 AM	File folder	
android_winusb	7/24/2020 6:29 AM	Setup Information	7 KB
androidwinusb86	7/24/2020 6:29 AM	Security Catalog	13 KB
androidwinusba64	7/24/2020 6:29 AM	Security Catalog	13 KB
Installation Steps	12/3/2021 10:44 PM	Text Document	1 KB
source	8/4/2020 4:17 AM	Properties Source ...	1 KB

Figure 56. Android Fastboot USB driver setup file

2. Right-click on it and select ‘Install’

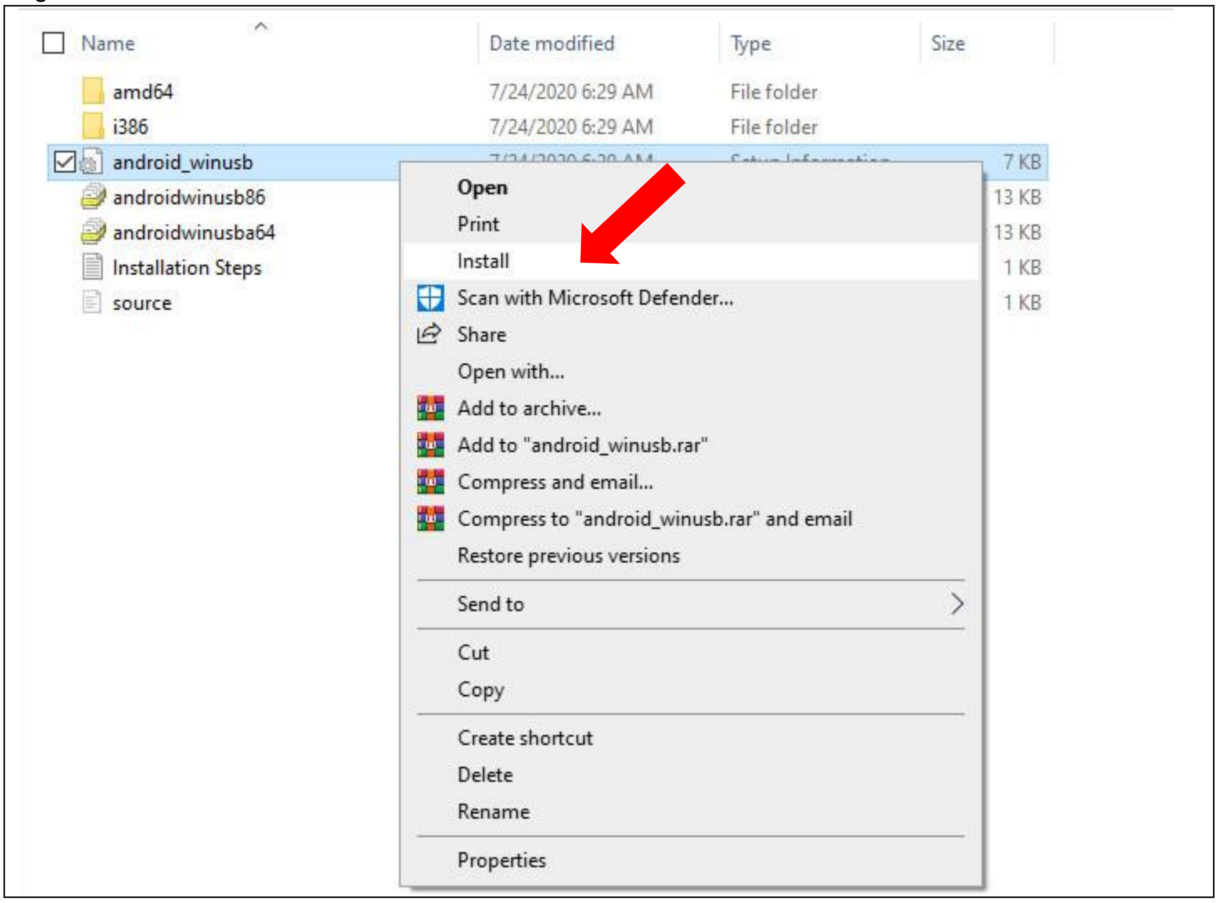
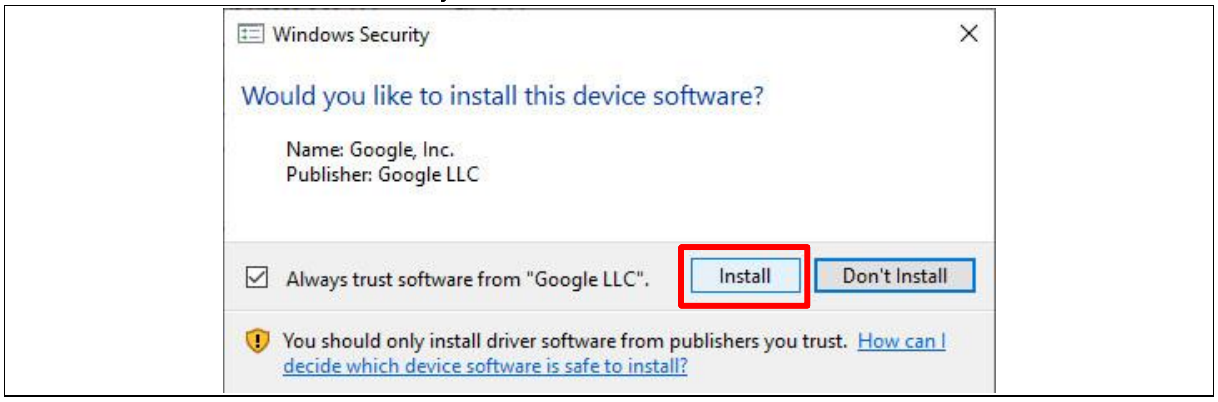


Figure 57. Android Fastboot USB driver installation window

3. If a ‘User Account Control’ window appears, click ‘Yes.’
4. Click ‘Install’ in the ‘Windows Security’ window.


Windows Security

Would you like to install this device software?

Name: Google, Inc.
Publisher: Google LLC

☑ Always trust software from "Google LLC".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 58. Confirm installing the Android Fastboot USB driver with Windows Security

5. Wait for the installation to complete.
6. Successful installation will end with the following notification.

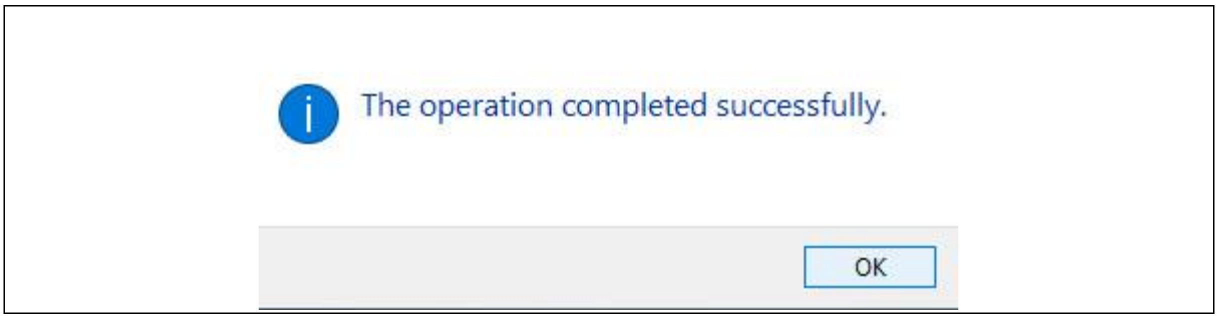

The operation completed successfully.

Figure 59. Successfully install the Android Fastboot USB driver

10.11 Installing necessary packages to run the support script for programming RZBoard images on Linux

Installation of some packages is required on your Linux machine when using the support script 'usb_util.py' for programming RZBoard images. If Ubuntu is used as a Linux OS, please install the following packages:

```
$ sudo apt update
$ sudo apt install -y python3 python3-pip
$ sudo apt install -y bzip2
$ sudo apt install -y fastboot
```

For Python3, we need to install some modules used in the script.

```
$ pip3 install pyserial==3.5 argparse==1.4.0 pytest==7.4.2 tqdm==4.66.1
```

11. References

RZBoard

Product page: RZBoard V2L | Avnet Boards

Quick start manual : RZBoard+V2L+Quick+StartGuide+v1.1.pdf (avnet.com)

Block Diagram: Block Diagram of RZBoard

Yocto User Manual : RZBoard-Linux-Yocto-UserManual-v2.2.pdf (avnet.com)

RZ/V2L SoC

Product page: RZ/V2L - General-Purpose Microprocessor Equipped with "DRP-AI" and 3D GPU

Wiki: RZ/V Series 64-bit MPU - Renesas.info

DRP-AI tools: RZ/V Software Tools and Scripts – Renesas.info

11.1 IP Configuration

IP address is a bit tricky to get right. It often won't show up unless the port is powered up, and it gets complicated to identify the interface name and ensure there is an address on it. There is some trial and error involved in this step for flashing the system image. You can manually assign the IP address to your host if necessary. Refer to the following for more info on Windows IP settings:

a. How to configure a static IP on Windows 10 or 11 | Windows Central
b. Change TCP/IP settings - Microsoft Support

12. Website and Support

Visit the following URLs to learn about key elements of the RA family, download components and related documentation, and get support:

RA Product Information	http://www.renesas.com/rz
RA Product Support Forum	https://community.renesas.com/rz
RA Flexible Software Package	www.renesas.com/FSP
Renesas Support	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.13.23	—	First release featuring: • Sys-V init optimized for 1.8-sec boot. • Renesas Fast-boot. • Yolov3 application using drp-ai. • VLP 3.0.0 and dependencies. • Automated fast flashing process
1.10	Nov.09.23	—	Updated: • Systemd init optimized for 1.5-sec boot. • VLP updated to 3.0.4 along with dependencies. • Correction to MCU flash procedure and hardware variation.
1.20	Dec.15.23	—	Updated: • Add adb fastboot-based flashing process. • Change the RZBoard flash procedure on Windows. • Add support for RZBoard flashing on Linux pc
1.30	Jun.08.24	—	Add: • Evaluation License to release. • Hardware wiring information to recreate the PoC. • Altium Design of the on/off button board and schematics.
1.40	Jun.27.24	—	Update: • Application note format. • Application note document number.
1.50	Jul.17.24	—	Update: • Standardize title and references. • Format as per web kit upload.
1.60	Jul.29.24	—	Update: • Title of document. • Minor corrections to document. • License.

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.

Corporate Headquarters

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

Trademarks

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

Contact information

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