ISO-DONGLE1Z Rev.B

Communications Dongle

This manual provides an overview of the ISO-DONGLE1Z Rev.B communications dongle bootloader and application software and a description of the bootloader structure, operation, and programming. The manual also contains a step-by-step guide for the application software post-processing and update in the device memory without using any external hardware tools.

Contents

1. Communications Dongle Overview

The ISO-DONGLE1Z Rev.B isolated dongle serves primarily as a communications dongle between a Battery Front End (BFE) evaluation board and a Graphical User Interface (GUI) available on a workstation (Figure 1). It supports USB and I2C communications and is compatible with multiple BFEs and GUIs. The dongle uses a powerful Renesas Advanced Family RA4 ARM-based microcontroller.

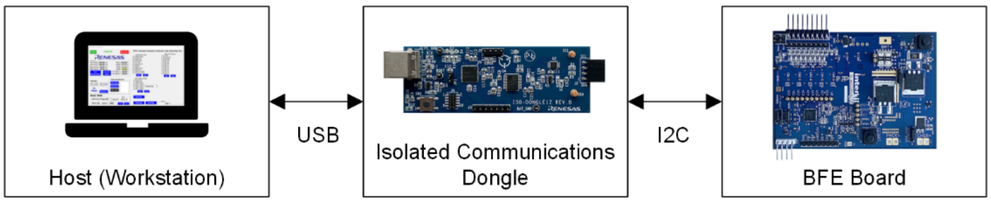


Figure 1. GUI, Communications Dongle and Battery Front End Connection

1.1 Assumptions and Advisory Notes

- It is assumed that users possess a basic understanding of microcontrollers, embedded systems hardware, battery management systems, and secondary battery cells.
- It is assumed users have prior experience working with Integrated Development Environments (IDEs) such as e2studio, Flexible Software Package (FSP), and terminal emulation programs such as Tera Term.
- Renesas recommends reviewing the ISO-DONGLE1Z Rev.B Communications Dongle Hardware Manual before changing or developing any software for the dongle.
- Renesas recommends reviewing the Renesas RA4 E1 Group 32-Bit MCU Datasheet, available on the product page, to become familiar with the MCU in use.

For more information about the bootloader operation, refer to the Renesas RA Family Secure Bootloader for RA2 MCU Series Application Note.

2. Bootloader

2.1 Structure and Operation

The bootloader of the ISO-DONGLE1Z Rev.B Isolated Communications Dongle enables upgrading or modifying the MCU firmware without the need for specialized programming hardware or tools. It features the MCUboot secure bootloader of the Renesas Flexible Software Package (FSP) to receive a binary file over the USB interface, verify its content integrity, and writes the program memory with the application program. The bootloader uses the whole code flash memory of the MCU. Figure 2 shows the flash memory map. It is separated into three major parts: Bootloader, Primary and Secondary Application Image Slots. The bootloader occupies 64-KB of the code flash memory or the first eight blocks (8-KB size), containing the MCUboot module, USB driver, HP Flash driver, and GPIO driver for the LED indication. Each image slot is 96-KB long or three blocks (32-KB size), which is the maximum memory space available for the application software.

Figure 3 shows the operation flowchart of the bootloader. When the CPU is released from reset, a low-level initialization is performed. All three user LEDs are turned simultaneously ON for 500ms and then OFF to perform an LED test and indicate the beginning of the bootloader code execution. Next, the digital input corresponding to the BOOT Test Point is tested for low level. If this condition is true, the USB interface is initialized to create a virtual COM port and activates the software image download process. The MCU waits for a connection with terminal emulator software and receives a start of download command, Carriage return (Enter). The bootloader starts to store every Byte received by the virtual serial connection in the Secondary Image Slot by portions of 32-KB (1 block) and if the transferred image size is different than 96-KB or any unexpected behavior is encountered, the operation is terminated and the LED D4 is illuminated.

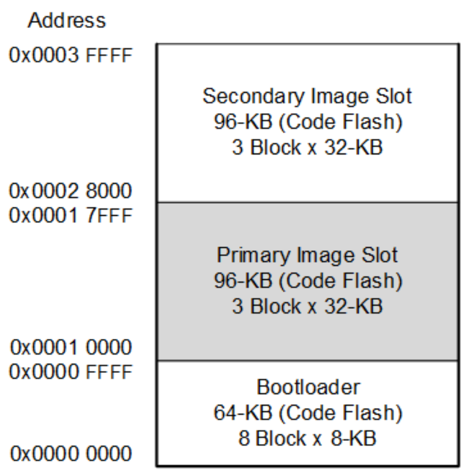


Figure 2. Bootloader Flash Memory Map

When the image download has been successfully completed, LED D3 is illuminated and the MCU halts operation and waits for manual restart. This is also the moment when the level of the digital input (the BOOT Test Point) must be changed to high (Remove jumper wire). Table 1 shows the various states indicated by the user LEDs and Table 2 shows the virtual COMM port settings.

Important: Do NOT set a higher baud rate as this can lead to buffer overrun and compromise the image transfer! After successful bootloader start-up and LED test if the level of the digital input corresponding to the BOOT Test Point is high (default), MCU proceeds to software image boot up. The bootloader checks the Secondary Image Slot memory space. If an image is available there, it is validated by checking integrity and authenticity. After successful authentication, the bootloader copies the new image using the overwrite update method where the entire content of the Primary Image Slot is overwritten with the contents of the Secondary Image Slot. The active image is always executed from the Primary Image Slot. This method is fail-safe, resistant to power-cut failures and with less memory overhead compared to the other available update methods. Conversely, it does not support image pre-testing and an automatic fallback mechanism. When the Primary Image Slot memory space is successfully overwritten, the bootloader executes the new image. If no new image is detected in the Secondary Image Slot, the bootloader directly authenticates and boots up the Primary Software Image.

Table 1. Bootloader LED States

Condition	LED States	
	D3 (Red)	D4 (Red)
LEDs blink test on start-up.	ON	ON
Proceeding to normal software image boot.	OFF	OFF
Software image download mode is entered.	Blinking slow	OFF
Waiting to receive the software image by the virtual serial connection.	Blinking fast	OFF
Receiving software image in process.	Blinking fast	Blinking fast (antiphase with D3)
Software image download is completed.	ON	OFF
An error has occurred during image download.	OFF	ON

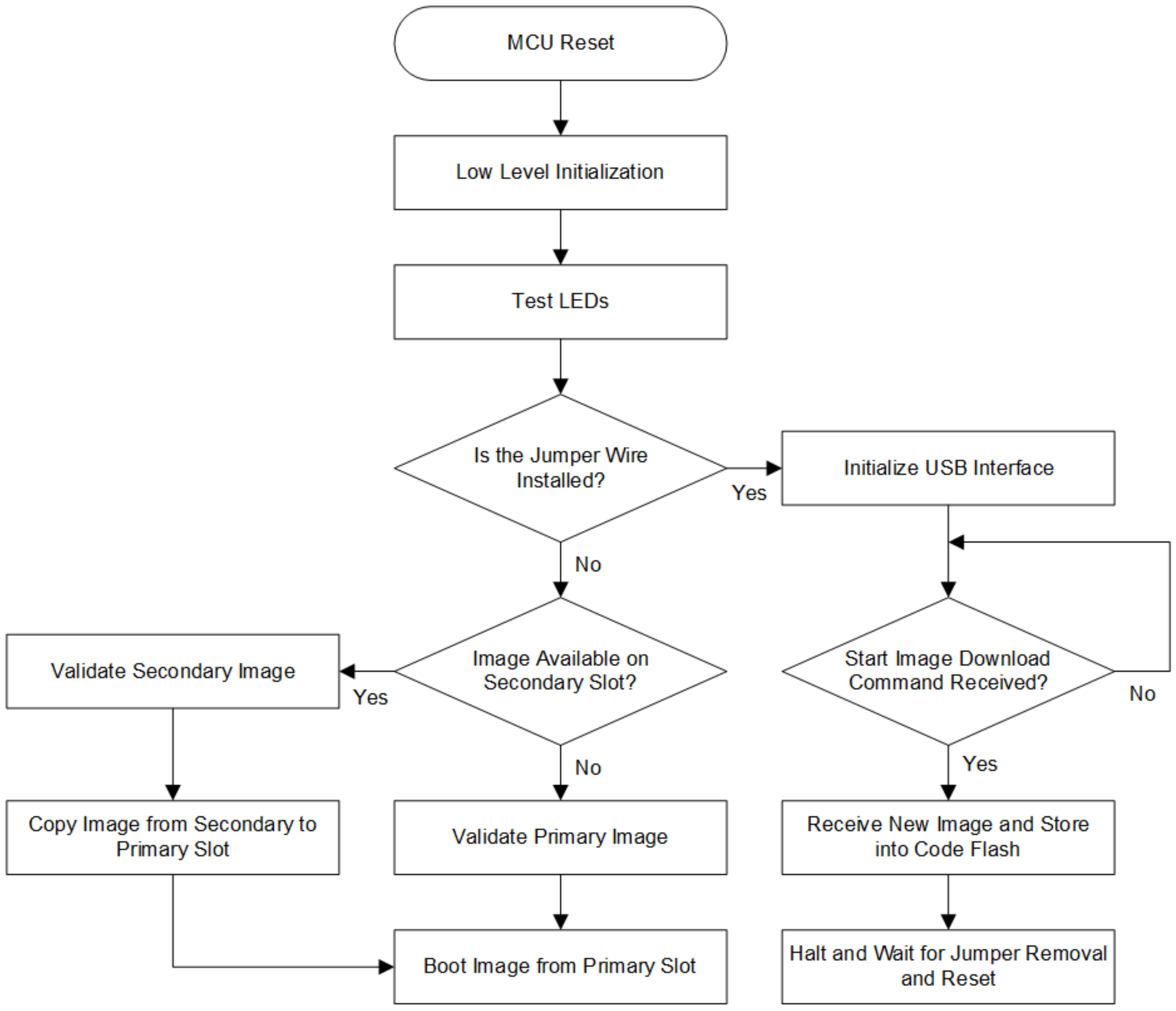

MCU Reset
Low Level Initialization
Test LEDs
Is the Jumper Wire Installed?
Yes
Initialize USB Interface
No
Image Available on Secondary Slot?
Validate Secondary Image
Yes
Start Image Download Command Received?
No
No
Yes
Copy Image from Secondary to Primary Slot
Validate Primary Image
Receive New Image and Store into Code Flash
Boot Image from Primary Slot
Halt and Wait for Jumper Removal and Reset

Figure 3. Bootloader Flowchart Diagram

Table 2. Serial Connection (Terminal) Settings

Parameter	Value
New Line (Receive)	CR
New Line (Transmit)	CR
Terminal ID	VT100
Baud Rate	9600
Data Bits	8 bits
Parity	none
Stop Bits	1 bit
Flow Control	none

2.2 Bootloader Programming

The ISO-DONGLE1Z Rev.B Isolated Communications Dongle is programmed by default with the bootloader software. However, if using external debugging tools during custom software development, it is erased or overwritten in the code flash memory. This section describes how to restore the bootloader in the MCU memory. Renesas E2 Emulator/ E2 Emulator Lite with 10-pin Cortex® Debug Connector and the following software available on a workstation running Windows® 10 are needed.

- Renesas Flash Programmer V3.11.00 or later.
- **ra4e1_iso_dongle_mcuboot_v_1_0.srec** binary file v1.0 or later

Complete the following steps for the bootloader programming process:

1. On the dongle make sure that jumper JP1 is installed.
2. Connect Renesas E2 Emulator/ E2 Emulator Lite to the JTAG connector J4 of the dongle board (Figure 4).

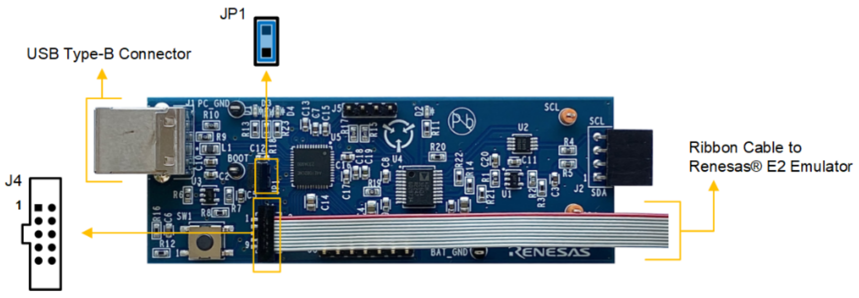
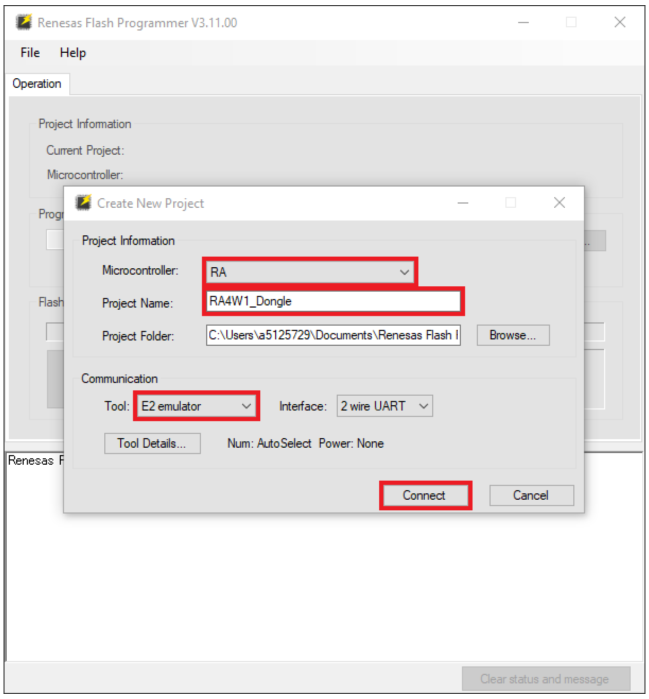


Figure 4. Communications Dongle with Attached Renesas E2 Emulator

3. Connect the USB cable between the PC and the USB Type-B connector of the board. Ensure the power ON indicator LED D1 is illuminated.
4. Run Renesas Flash Programmer and on the top menu select **File** > **New Project...** From the **Microcontroller:** drop-down menu select **RA** (Figure 5). In the **Project Name:** field enter RA4E1_Dongle. From the **Tool** drop-down menu, select **E2 emulator** or **E2 emulator Lite** and click on the **Connect** button.


Renesas Flash Programmer V3.11.00

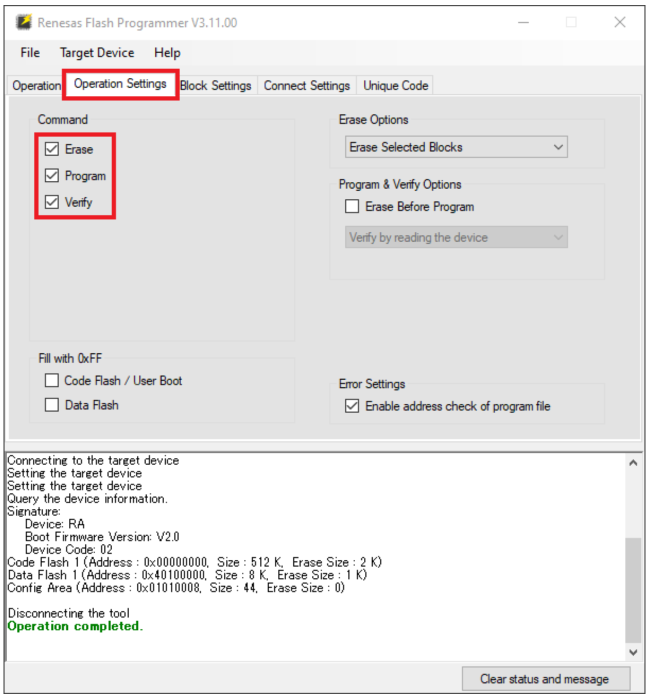
Create New Project

Project Information
Microcontroller: RA
Project Name: RA4W1_Dongle
Project Folder: C:\Users\a5125729\Documents\Renesas Flash I

Communication
Tool: E2 emulator
Interface: 2 wire UART
Num: AutoSelect Power: None

Figure 5. Creating a New Project in Renesas Flash Programmer

5. In the main window select the **Operation Settings** tab and ensure in the **Command** section **Erase**, **Program** and **Verify** are selected (Figure 6).



Command
[✓] Erase
[✓] Program
[✓] Verify

Erase Options
Erase Selected Blocks

Program & Verify Options
[ ] Erase Before Program
Verify by reading the device

Fill with 0xFF
[ ] Code Flash / User Boot
[ ] Data Flash

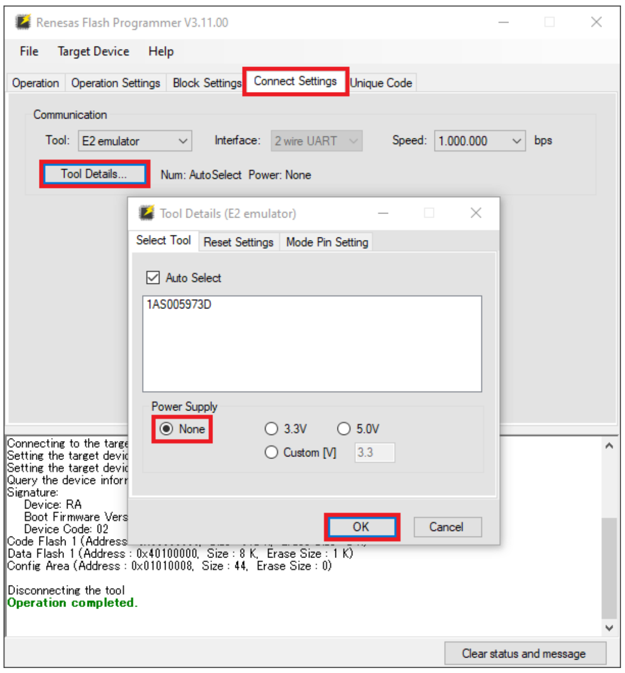
Error Settings
[✓] Enable address check of program file

```
Connecting to the target device
Setting the target device
Setting the target device
Query the device information.
Signature:
   Device: RA
   Boot Firmware Version: V2.0
   Device Code: 02
Code Flash 1 (Address : 0x00000000, Size : 512 K, Erase Size : 2 K)
Data Flash 1 (Address : 0x40100000, Size : 8 K, Erase Size : 1 K)
Config Area (Address : 0x01010008, Size : 44, Erase Size : 0)

Disconnecting the tool
Operation completed.
```

Figure 6. Operation Settings in Renesas Flash Programmer

6. Select the **Connect Settings** tab and click on the **Tool Details...** button (Figure 7). In the pop-up window make sure that option **None** is selected in the **Power Supply** section. Click on the **OK** button to close the pop-up window.



Communication
Tool: E2 emulator   Interface: 2 wire UART   Speed: 1,000,000 bps
Num: AutoSelect Power: None

[✓] Auto Select
1AS005973D

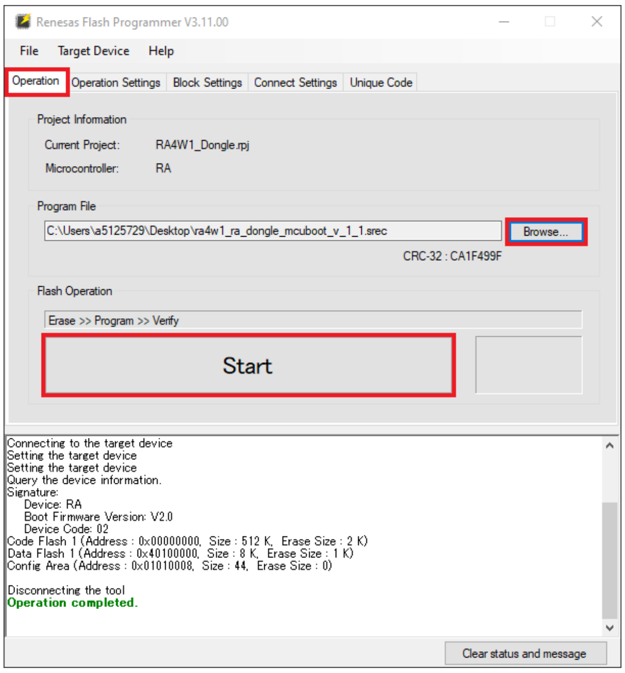
Power Supply
[✓ None] 3.3V 5.0V
Custom [V] 3.3

```
Connecting to the targe
Setting the target devic
Setting the target devic
Query the device infor
Signature:
   Device: RA
   Boot Firmware Vers
   Device Code: 02
Code Flash 1 (Address
Data Flash 1 (Address : 0x40100000, Size : 8 K, Erase Size : 1 K)
Config Area (Address : 0x01010008, Size : 44, Erase Size : 0)

Disconnecting the tool
Operation completed.
```

Figure 7. Connection Settings in Renesas Flash Programmer

7. Select the **Operation** tab and click on the **Browse…** button (Figure 8). In File Explorer navigate to the location of the bootloader binary file (.srec), select it and then click on the **Open** button. Click on the **Start** button to download the file into the code flash memory.



Project Information
Current Project: RA4W1_Dongle.rpj
Microcontroller: RA

Program File
C:\Users\a5125729\Desktop\ra4w1_ra_dongle_mcuboot_v_1_1.srec
CRC-32 : CA1F499F

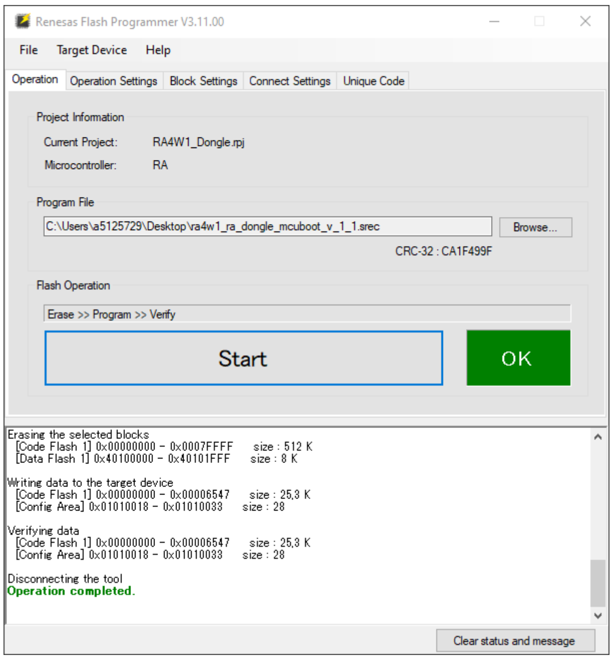
Flash Operation
Erase >> Program >> Verify

```
Connecting to the target device
Setting the target device
Setting the target device
Query the device information.
Signature:
    Device: RA
    Boot Firmware Version: V2.0
    Device Code: 02
Code Flash 1 (Address : 0x00000000,  Size : 512 K,  Erase Size : 2 K)
Data Flash 1 (Address : 0x40100000,  Size : 8 K,  Erase Size : 1 K)
Config Area (Address : 0x01010008,  Size : 44,  Erase Size : 0)

Disconnecting the tool
Operation completed.
```

Figure 8. Selection of the Program File

8. When the process completes successfully you should see a green OK message in a rectangular window next to the **Start** button (Figure 9). Disconnect the USB cable. Disconnect the Renesas E2 Emulator/ E2 Emulator Lite from the dongle and close the Renesas Flash programmer. The bootloader programming process is completed.



Project Information
Current Project: RA4W1_Dongle.rpj
Microcontroller: RA

Program File
C:\Users\a5125729\Desktop\ra4w1_ra_dongle_mcuboot_v_1_1.srec
CRC-32 : CA1F499F

Flash Operation
Erase >> Program >> Verify

```
Erasing the selected blocks
  [Code Flash 1] 0x00000000 - 0x0007FFFF    size : 512 K
  [Data Flash 1] 0x40100000 - 0x40101FFF    size : 8 K

Writing data to the target device
  [Code Flash 1] 0x00000000 - 0x00006547    size : 25,3 K
  [Config Area] 0x01010018 - 0x01010033    size : 28

Verifying data
  [Code Flash 1] 0x00000000 - 0x00006547    size : 25,3 K
  [Config Area] 0x01010018 - 0x01010033    size : 28

Disconnecting the tool
Operation completed.
```

Figure 9. Successful Programming Window

3. Software Image Update

The software image (referred to as image) in the communication dongle can be updated or modified without using any additional external software programmers. To initiate the software image update process, first ensure the required image is available on the PC and the terminal emulator program Tera Term is installed. Tera Term can be downloaded from the Tera Term Home Page.

The image update process must follow these steps:

1. Connect the BOOT Test Point to the PC_GND Test Point using a jumper wire as demonstrated on Figure 10.
2. Connect the USB cable between the PC and the USB Type-B connector of the board. Ensure LED D3 is blinking slowly, indicating that the bootloader is in Image Update Mode.

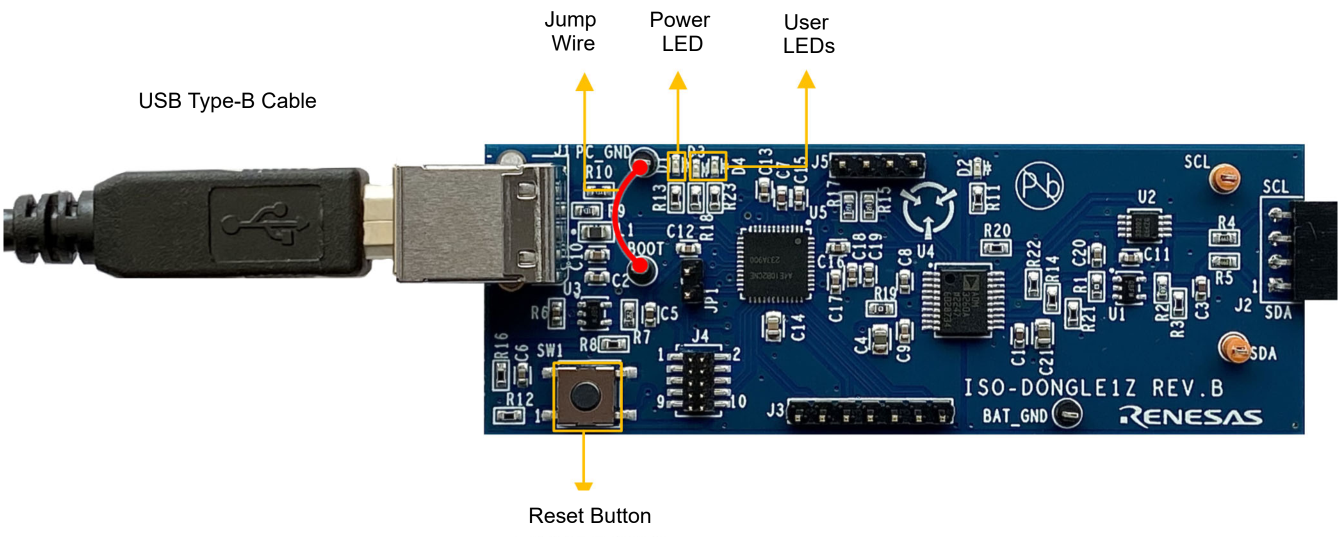
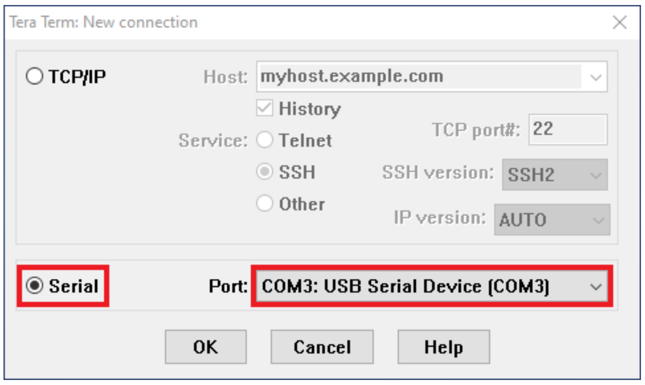


Figure 10. Connection of the ISO-DONGLE1Z Rev.B Isolated Communications Dongle for Software Image Update

3. Open Tera Term. A new connection should automatically open (Figure 11). If not, select **File** > **New Connection**. From the window, select **Serial** and ensure the **USB Serial Device COM** port is selected from the drop-down list. Click the **OK** button to open the connection.


Tera Term: New connection

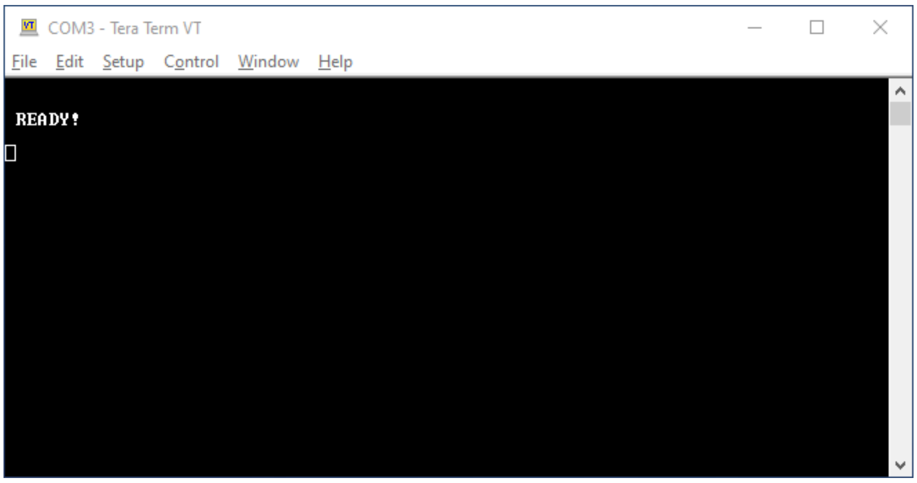
TCP/IP Host: myhost.example.com
History [✓]
Service: Telnet [● SSH] Other
TCP port#: 22
SSH version: SSH2
IP version: AUTO

[● Serial] Port: COM3: USB Serial Device (COM3)

Figure 11. Tera Term New Connection Window

4. Press **Enter** to activate Image Download Mode. LED D3 starts blinking fast and READY! message is displayed on the console screen (Figure 12). The bootloader is now waiting to receive the image file by the USB serial

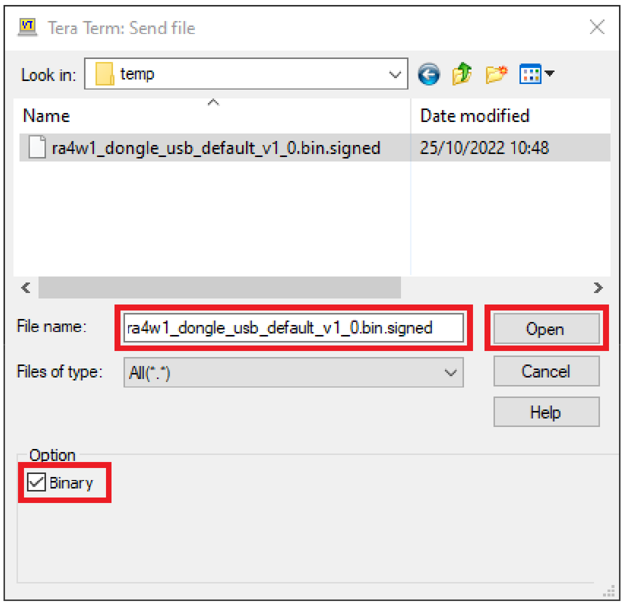
interface. Each subsequent byte is interpreted as a part of the Image. Be careful not to press any other buttons or the process will be compromised and must be restarted.



```
READY!
```

Figure 12. Console Screen – Ready for Software Image Download

5. Select **File** > **Send file...** from the menu. On the opened window (Figure 13) select **Binary**, navigate to the location of the software image file, select it, and click on the **Open** button. Now the terminal emulator program sends the whole file. LEDs D3 and D4 blink in antiphase and a pop-up window displays the transfer progress, which takes approximately 10 seconds.



Look in: temp

Name	Date modified
ra4w1_dongle_usb_default_v1_0.bin.signed	25/10/2022 10:48

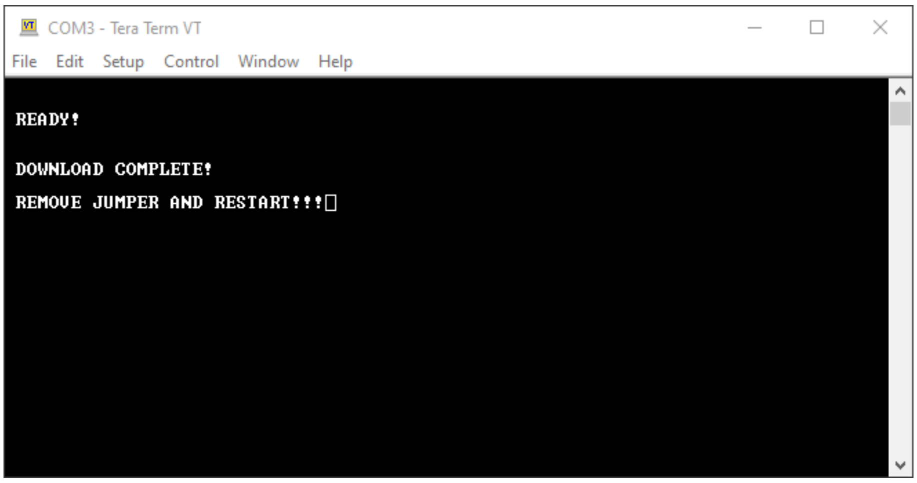
File name: ra4w1_dongle_usb_default_v1_0.bin.signed

Files of type: All(*.*)

☑ Binary

Figure 13. Selection of Software Image

6. When the software image download has finished D3 remains illuminated and DOWNLOAD COMPLETE! message is printed in the console screen (Figure 14). You must unplug the USB cable and remove the jumper wire. Next time the MCU powers up, it copies the new software image, and boots from it. This completes the software image update process. If the red LED D4 is illuminated, the bootloader becomes unresponsive or the image transfer sticks, there is something wrong with the image update process. Unplug the USB cable and restart the process from Step 1.



```
READY!

DOWNLOAD COMPLETE!
REMOVE JUMPER AND RESTART!!!
```

Figure 14. Console Screen – Software image download has finished

4. Generation of Software Image

Developing and running a customized software (application) on the dongle using the bootloader to avoid the need for any debugger tools requires the generation of a software image that meets the bootloader size and verification requirements. This section describes the process of generating a software image from a conventional project in e^2 studio. Before beginning, the following development tools and software available on a workstation running Windows® 10 are needed:

- e^2 studio Development Environment (IDE) 2022-07 or greater with GCC Arm® Embedded tool chain
- Renesas Flexible Software Package (FSP) 4.0.0 or later
- Python 3.11.0 or later
- **ra4e1_iso_dongle_mcuboot_v_1_0.rar** bootloader software project v1.0 or later

Important: Ensure that Python is installed to all users, and it is added to Windows PATH (**Control Panel** > **System** > **Advanced System Settings** > **Advanced** > **Environment Variables**). Otherwise, running python scripts through e^2 studio IDE will not work.

If this procedure has already been done and the MCU boot software project and python environment are already set up, go directly to Step 10.

The image generation process must follow these steps:

1. Import the MCUboot software project that contains the python script and is used in the image generation process. *Note*: This is the bootloader project **not** the application software that runs from the target image. Ensure the **ra4e1_iso_dongle_mcuboot_v_1_0.rar** project archive file is already available. Right-click in the e^2 studio Project Explorer view and select **Import** from the menu showing up to open the Import Wizard (Figure 15).

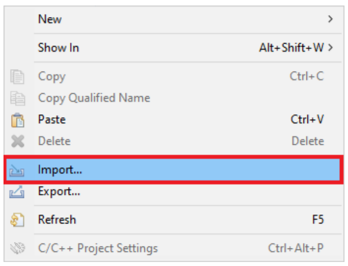


Figure 15. Right-click Menu to Import the Bootloader Project

2. From the Import Wizard window, select **Existing Project into Workspace** and then click on the **Next >** button (Figure 16).

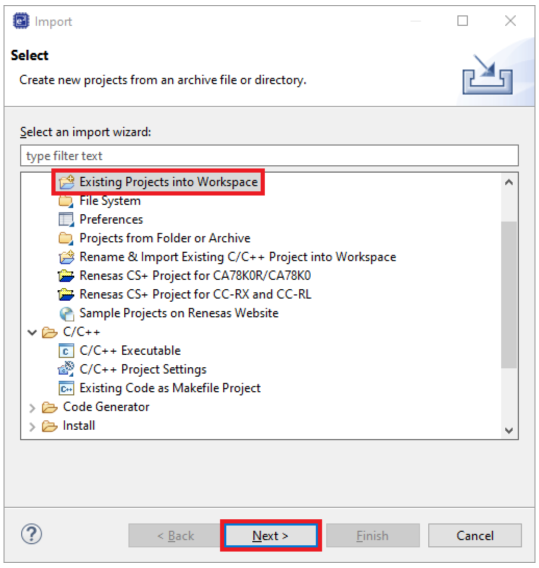

Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
Rename & Import Existing C/C++ Project into Workspace
Renesas CS+ Project for CA78K0R/CA78K0
Renesas CS+ Project for CC-RX and CC-RL
Sample Projects on Renesas Website
C/C++
C/C++ Executable
C/C++ Project Settings
Existing Code as Makefile Project
Code Generator
Install
< Back
Next >
Finish
Cancel

Figure 16. Selection of the Import Wizard Option

3. On the next window, select **Select archive file** and then click on the **Browse…** button (Figure 17). In File Explorer, navigate to the location of the bootloader project archive file (.zip), select it and then click on the **Open** button. Click on the **Finish** button to complete the import process and close the Import Wizard.

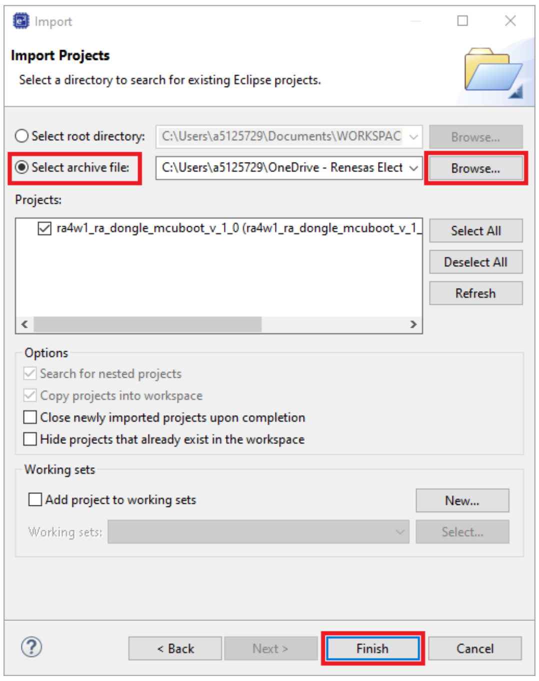

Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
C:\Users\a5125729\Documents\WORKSPAC
Browse...
Select archive file:
C:\Users\a5125729\OneDrive - Renesas Elect
Browse...
Projects:
ra4w1_ra_dongle_mcuboot_v_1_0 (ra4w1_ra_dongle_mcuboot_v_1_
Select All
Deselect All
Refresh
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
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel

Figure 17. Selection of the Bootloader Project

4. The project is now imported into the e^2 studio workspace. Double-click on **configuration.xml** to open the FSP Configurator and click on the **Generate Project Content** button to generate the hardware specific project files that are needed to run the project on the selected hardware (Figure 18).

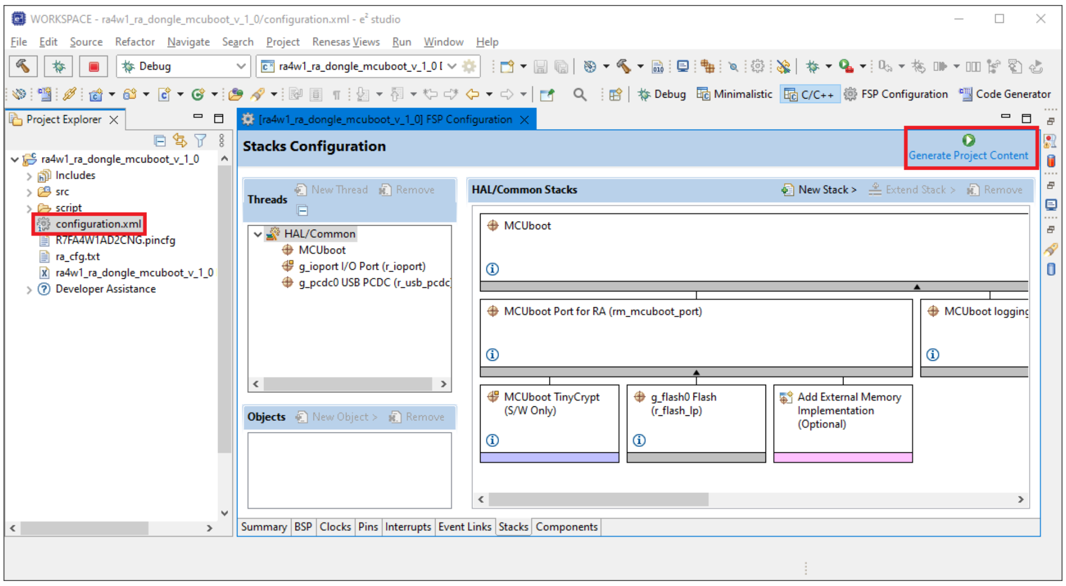

Project Explorer
ra4w1_ra_dongle_mcuboot_v_1_0
Includes
src
script
configuration.xml
R7FA4W1AD2CNG.pincfg
ra_cfg.txt
ra4w1_ra_dongle_mcuboot_v_1_0
Developer Assistance
Stacks Configuration
Generate Project Content
Threads
HAL/Common
MCUboot
g_ioport I/O Port (r_ioport)
g_pcdc0 USB PCDC (r_usb_pcdc
HAL/Common Stacks
MCUboot Port for RA (rm_mcuboot_port)
MCUboot TinyCrypt (S/W Only)
g_flash0 Flash (r_flash_lp)
Add External Memory Implementation (Optional)
Summary BSP Clocks Pins Interrupts Event Links Stacks Components

Figure 18. Generation of Hardware Specific Project Files

5. Configure the Python Signing Environment. Signing the application image is done using a post-build step in e^2 studio using the image signing tool **Imgtool.py**, which is included with MCUboot. In the e^2 studio Project Explorer navigate to **ra4e1_iso_dongle_mcuboot_v_1_0\ra\mcu-tools\MCUboot** folder (Figure 19). Right-click over the **MCUboot** folder and select **Command Prompt**. This opens a command window with the path set to the **\mcu-tools\MCUboot** folder.

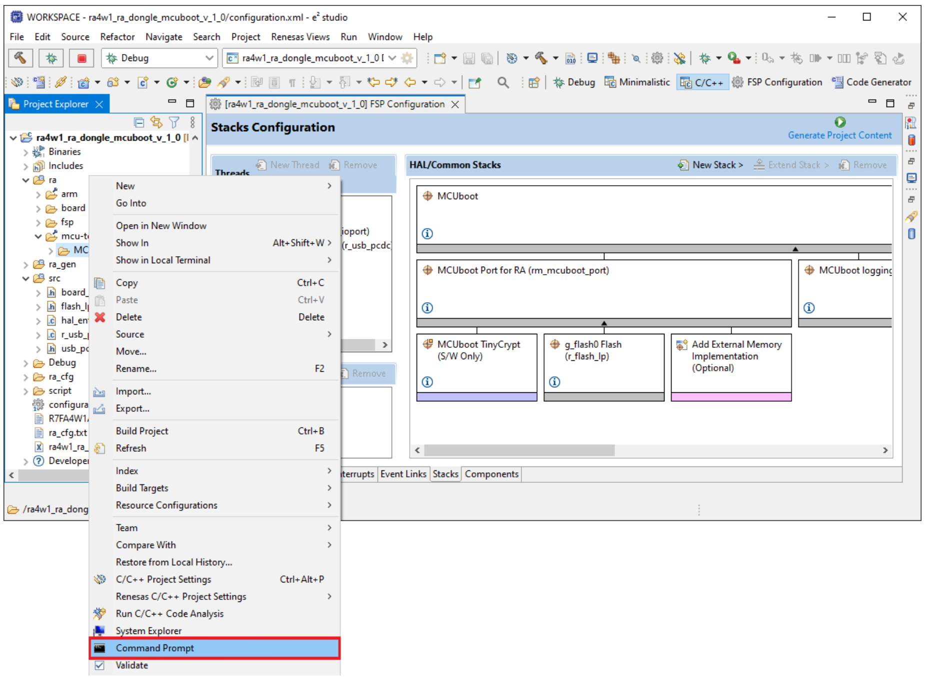


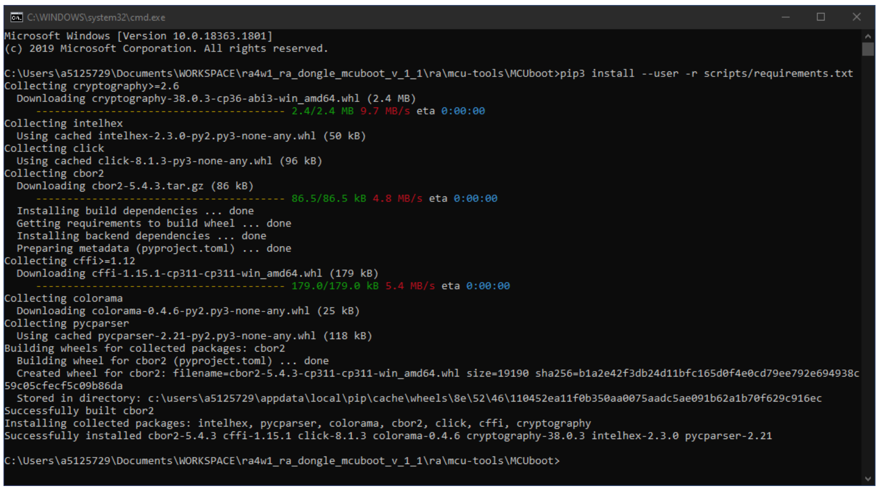
Figure 19. Opening Command Prompt in the MCUboot folder

6. Enter the following command to update pip:

   ```
   python -m pip install --upgrade pip
   ```

7. Next, in the command window, enter the following command line to verify and install all the MCUboot dependencies (Figure 20):

   ```
   pip3 install --user -r scripts/requirements.txt
   ```



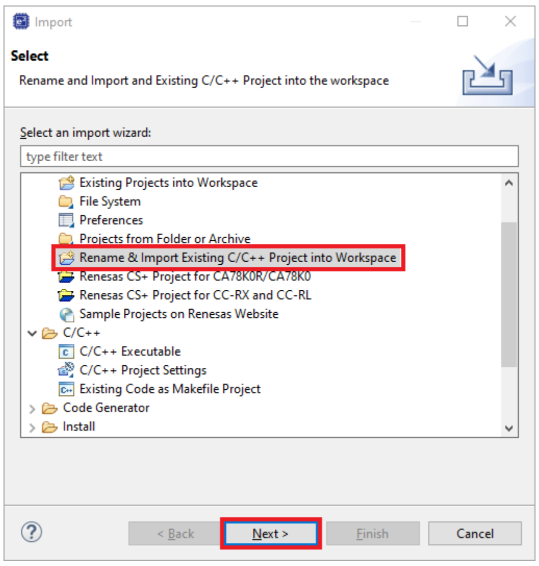
```
Microsoft Windows [Version 10.0.18363.1801]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\a5125729\Documents\WORKSPACE\ra4w1_ra_dongle_mcuboot_v_1_1\ra\mcu-tools\MCUboot>pip3 install --user -r scripts/requirements.txt
Collecting cryptography>=2.6
  Downloading cryptography-38.0.3-cp36-abi3-win_amd64.whl (2.4 MB)
     ---------------------------------------- 2.4/2.4 MB 9.7 MB/s eta 0:00:00
Collecting intelhex
  Using cached intelhex-2.3.0-py2.py3-none-any.whl (50 kB)
Collecting click
  Using cached click-8.1.3-py3-none-any.whl (96 kB)
Collecting cbor2
  Downloading cbor2-5.4.3.tar.gz (86 kB)
     ---------------------------------------- 86.5/86.5 kB 4.8 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Collecting cffi>=1.12
  Downloading cffi-1.15.1-cp311-cp311-win_amd64.whl (179 kB)
     ---------------------------------------- 179.0/179.0 kB 5.4 MB/s eta 0:00:00
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting pycparser
  Using cached pycparser-2.21-py2.py3-none-any.whl (118 kB)
Building wheels for collected packages: cbor2
  Building wheel for cbor2 (pyproject.toml) ... done
  Created wheel for cbor2: filename=cbor2-5.4.3-cp311-cp311-win_amd64.whl size=19190 sha256=b1a2e42f3db24d11bfc165d0f4e0cd79ee792e694938c
59c05cfecf5c09b86da
  Stored in directory: c:\users\a5125729\appdata\local\pip\cache\wheels\8e\52\46\110452ea11f0b350aa0075aadc5ae091b62a1b70f629c916ec
Successfully built cbor2
Installing collected packages: intelhex, pycparser, colorama, cbor2, click, cffi, cryptography
Successfully installed cbor2-5.4.3 cffi-1.15.1 click-8.1.3 colorama-0.4.6 cryptography-38.0.3 intelhex-2.3.0 pycparser-2.21

C:\Users\a5125729\Documents\WORKSPACE\ra4w1_ra_dongle_mcuboot_v_1_1\ra\mcu-tools\MCUboot>
```

Figure 20. Installing the MCUboot dependencies

8. Close the Command Prompt.
9. Make sure the **ra4e1_iso_dongle_mcuboot_v_1_0** project is active (selected in Project Explorer) and on the top menu select **Project** > **Build Project** and wait until the process completes.
10. The next step is to configure the existing application to use the bootloader project. The software project that has to run on the dongle is duplicated and renamed so that the original can be still debugged using external debugging tool during further code development. Keep in mind that the binary size must be smaller or equal to 96-KB. This is the fixed image size and dedicated memory space in the code flash. It cannot be exceeded! Right-click in the e^2 studio Project Explorer view and select **Import** from the menu showing up to open the Import Wizard (Figure 15).
11. From the Import Wizard window select **Rename & Import Existing C/C++ Project into Workspace** and then click on the **Next >** button (Figure 21).


Import

Select

Rename and Import and Existing C/C++ Project into the workspace

Select an import wizard:

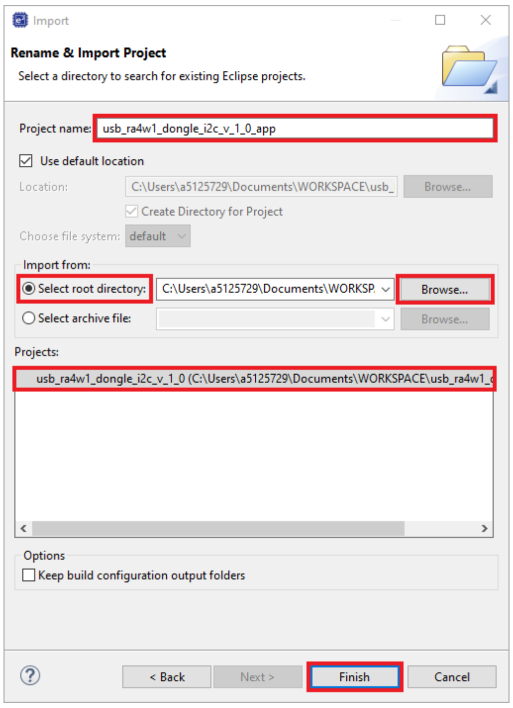
type filter text

- Existing Projects into Workspace
- File System
- Preferences
- Projects from Folder or Archive
- Rename & Import Existing C/C++ Project into Workspace
- Renesas CS+ Project for CA78K0R/CA78K0
- Renesas CS+ Project for CC-RX and CC-RL
- Sample Projects on Renesas Website
- C/C++
  - C/C++ Executable
  - C/C++ Project Settings
  - Existing Code as Makefile Project
- Code Generator
- Install

< Back | Next > | Finish | Cancel

Figure 21. Selection of the Import Wizard Option

12. On the next window enter a project name (Figure 22). Renesas recommends using the same name as the original project with a suffix ***_app**. Select the **Select root directory** option and then click on the **Browse…** button. In File Explorer, navigate to the location of the existing application project, select it, click on the **Select Folder** button, and select the required project in the **Projects:** section. Click on the **Finish** button to complete the import process and close the Import Wizard.


Import

Rename & Import Project

Select a directory to search for existing Eclipse projects.

Project name: usb_ra4w1_dongle_i2c_v_1_0_app

[✓] Use default location

Location: C:\Users\a5125729\Documents\WORKSPACE\usb_ Browse...

[✓] Create Directory for Project

Choose file system: default

Import from:

[●] Select root directory: C:\Users\a5125729\Documents\WORKSP Browse...

○ Select archive file: Browse...

Projects:

usb_ra4w1_dongle_i2c_v_1_0 (C:\Users\a5125729\Documents\WORKSPACE\usb_ra4w1_

Options

☐ Keep build configuration output folders

< Back | Next > | Finish | Cancel

Figure 22. Selection of the Sample Project

13. The project is now duplicated into the e^2 studio workspace and must be configured. In the **Project Explorer** right-click on the application project folder and select **Properties** to open the project properties window (Figure 23).

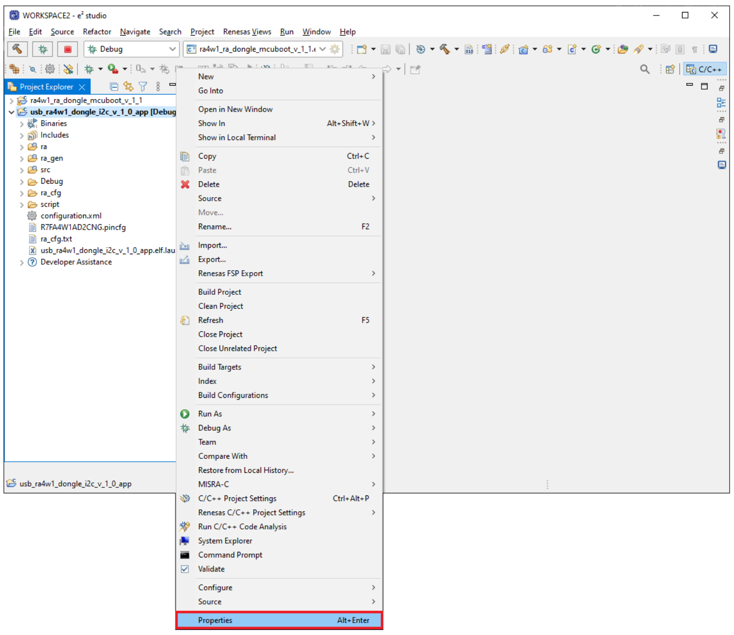
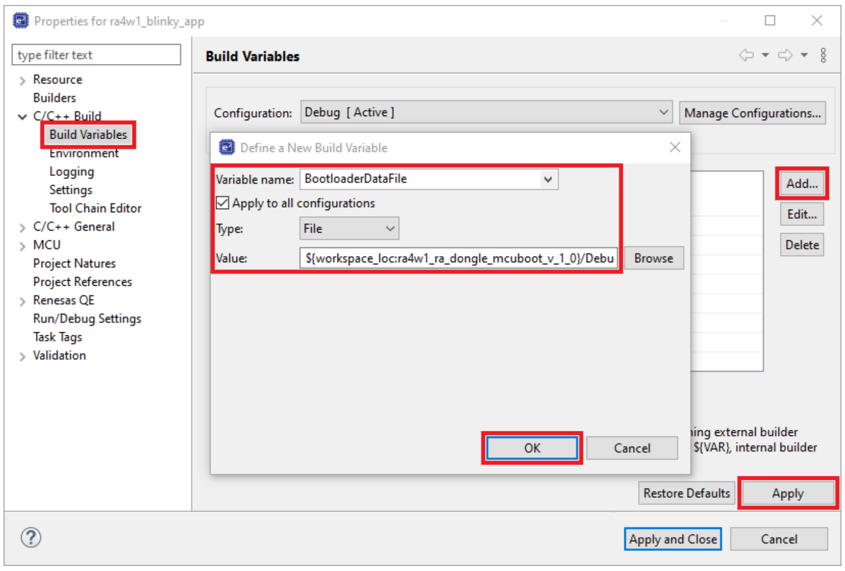


Figure 23. Opening Project Properties

14. Select **C/C++ Build** > **Build Variables**, click on the **Add...** button. In the **Variable name:** field enter **BootloaderDataFile**. Check the **Apply to all configurations** box (Figure 24). Change the **Type:** to **File** and in the bootloader project in **Value:** enter the following relative path to the *.bld file: **${workspace_loc:ra4e1_iso_dongle_mcuboot_v_1_0}/Debug/ra4e1_iso_dongle_mcuboot_v_1_0.bld** *Note*: In case the bootloader project name or location are different, the value format is: **${workspace_loc:<boot_project_name>}/Debug/<boot_project_name>.bld**. Click on the **OK** button and then on the **Apply** button to save the change of the properties.


Properties for ra4w1_blinky_app

Build Variables

Configuration: Debug [Active]

Define a New Build Variable

Variable name: BootloaderDataFile

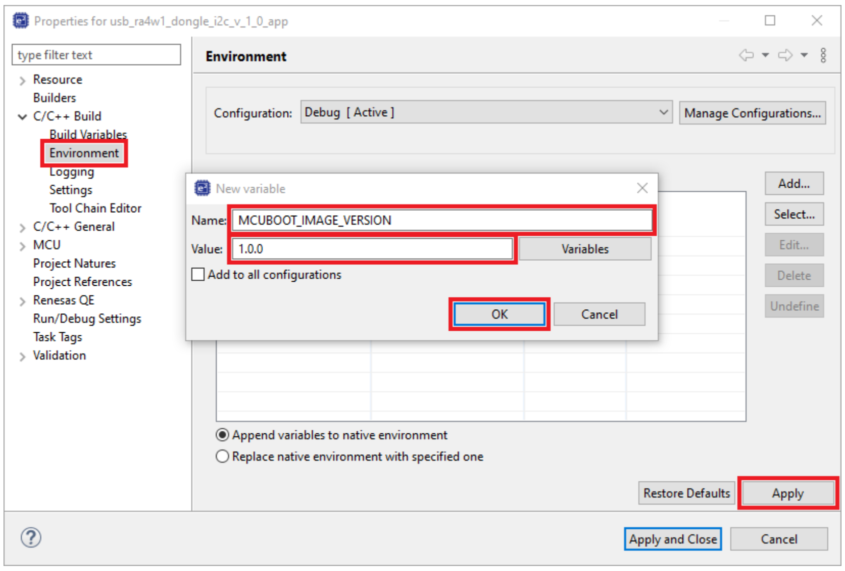
☑ Apply to all configurations

Type: File

Value: ${workspace_loc:ra4w1_ra_dongle_mcuboot_v_1_0}/Debu

Figure 24. Adding a Build Variable to Use the Bootloader

15. Select **C/C++ Build > Environment**, click on the **Add...** button. In the **Name:** field enter MCUBOOT_IMAGE_VERSION and set the **Value:** to the actual application project version number (Figure 25). Click on the **OK** button and then on the **Apply** button to save the change of the properties.



Configuration: Debug [Active]

Name: MCUBOOT_IMAGE_VERSION

Value: 1.0.0

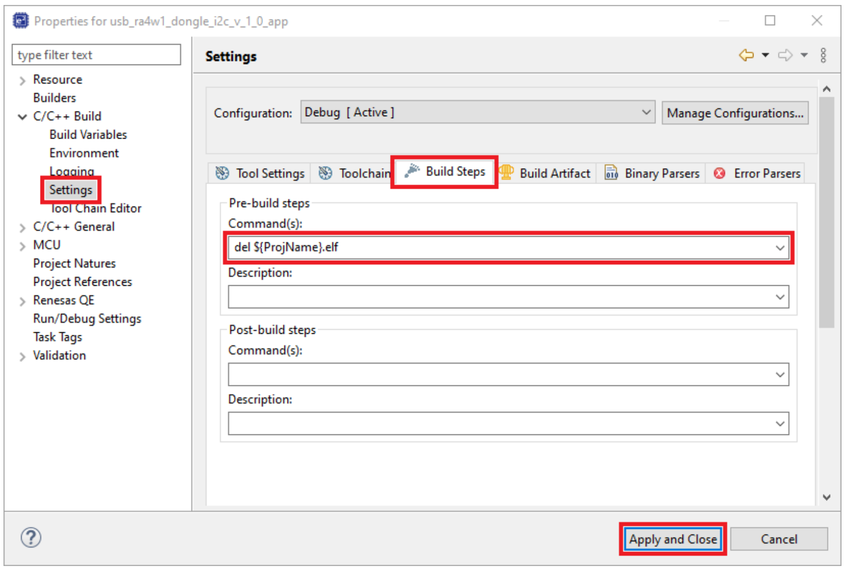
Add to all configurations

Append variables to native environment

Replace native environment with specified one

Figure 25. Adding a Project Version Number

16. Select **C/C++ Build** > **Settings** and then select the **Build Steps** tab (Figure 26). In the **Command(s):** field enter the following pre-build command: **del ${ProjName}.elf** to always delete the *.elf file. Click on the **Apply and Close** button to save the change of the properties and close the window.



Configuration: Debug [Active]

Pre-build steps

Command(s): del ${ProjName}.elf

Description:

Post-build steps

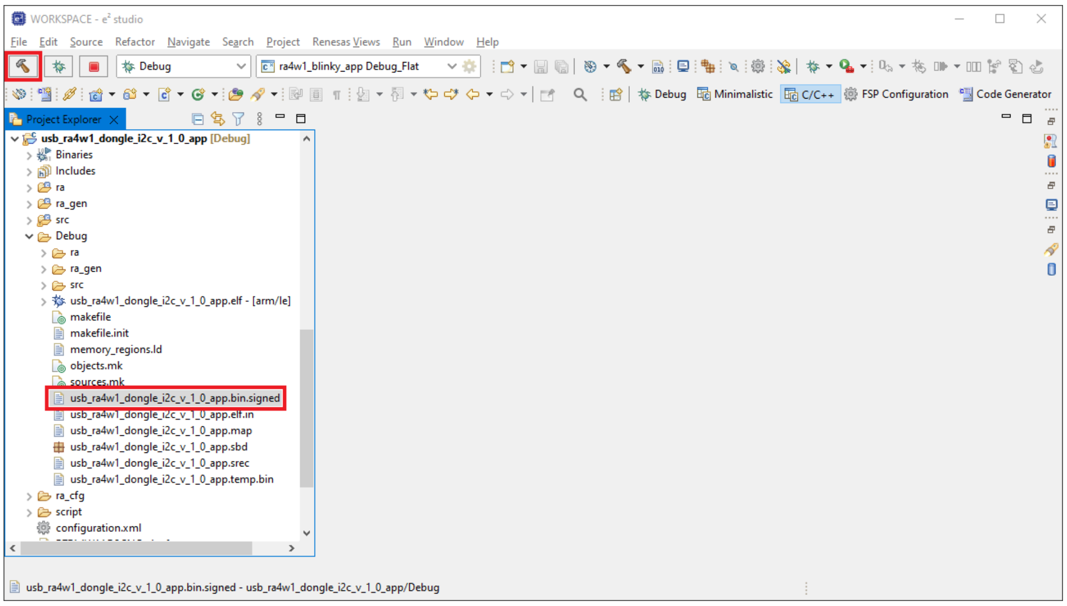
Command(s):

Description:

Figure 26. Adding a Pre-Build Command

17. Ensure the project is active (selected in Project Explorer), on the top menu select **Project** > **Build Project** and wait until the process completes (Figure 27). Then, in **Project Explorer** expand the **Debug** folder. The

generated application image binary file is under the name format **<app_project_name>.bin.signed**. From here it can be directly copied to a known location.



WORKSPACE - e² studio
File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
ra4w1_blinky_app Debug_Flat
Debug Minimalistic C/C++ FSP Configuration Code Generator
Project Explorer

```
usb_ra4w1_dongle_i2c_v_1_0_app [Debug]
  Binaries
  Includes
  ra
  ra_gen
  src
  Debug
    ra
    ra_gen
    src
    usb_ra4w1_dongle_i2c_v_1_0_app.elf - [arm/le]
    makefile
    makefile.init
    memory_regions.ld
    objects.mk
    sources.mk
    usb_ra4w1_dongle_i2c_v_1_0_app.bin.signed
    usb_ra4w1_dongle_i2c_v_1_0_app.elf.in
    usb_ra4w1_dongle_i2c_v_1_0_app.map
    usb_ra4w1_dongle_i2c_v_1_0_app.sbd
    usb_ra4w1_dongle_i2c_v_1_0_app.srec
    usb_ra4w1_dongle_i2c_v_1_0_app.temp.bin
  ra_cfg
  script
  configuration.xml
```

usb_ra4w1_dongle_i2c_v_1_0_app.bin.signed - usb_ra4w1_dongle_i2c_v_1_0_app/Debug

Figure 27. Generating the Application Image

For more information about the image generation process, refer to Renesas RA Family Secure Bootloader for RA2 MCU Series Application Note.

5. Revision History

Revision	Date	Description
1.01	Jul 24, 2024	Updated Figure 10. Updated J3 to the BOOT Test Point in three places.
1.00	Dec 12, 2023	Initial release.

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-us/.

Trademarks

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