JTAG Manual

This document describes how to use JTAG to flash firmware and perform debugging on the RA6W1 and the RA6W2 devices, as well as on the corresponding modules RRQ61001 and RRQ61051.

Contents

Figures

© 2026 Renesas Electronics

Tables

1. Terms and Definitions

CMD	Windows Command Line
EVK	Evaluation Kit
GDB	GNU Debugger
HOST	PC/Linux/Raspberry Pi connected to the EVK
JTAG	Joint Test Action Group
OB	On Board

2. Introduction

The J-Link GDB Server (JTAG) is used to erase and flash the firmware image to RA6W1/RA6W2/RRQ61001/RRQ61051, and to perform debugging. SEGGER tools are used for efficient and streamlined debugging and flashing processes. The JTAG interface enables real-time monitoring, diagnostics, and control over the firmware.

3. JTAG Interface

This section provides an overview of the JTAG interface options on the EK-RA6W1/RA6W2 boards, including both the on-board debugger and support for external JTAG tools.

3.1 EK-RA6W1/RA6W2 On-board JTAG

EK-RA6W1/RA6W2 includes an RA4M2 on-board to be used as a licensed JTAG programming interface.

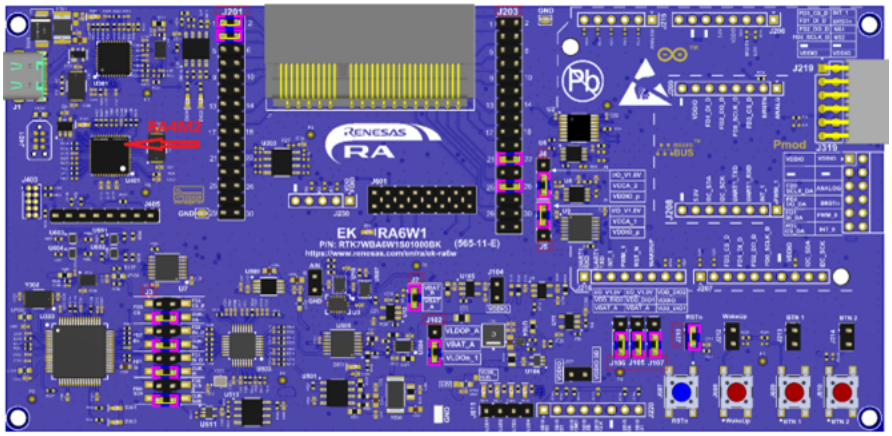

Figure 1. RA4M2 on EK-RA6W1/RA6W2 for using JTAG

3.2 External JTAG

You can also use an external SEGGER JTAG regardless of the EK-RA6W1/RA6W2 version.
The required components are:

- SEGGER: J-Link BASE Compact (recommended model)
- Cable adaptor 20 pins to 10 pins: Arm-JTAG-20-10
- Power: Micro USB cable.

Figure 2. External JTAG required components

Connect the External JTAG to EVK J403 as shown in Figure 3.

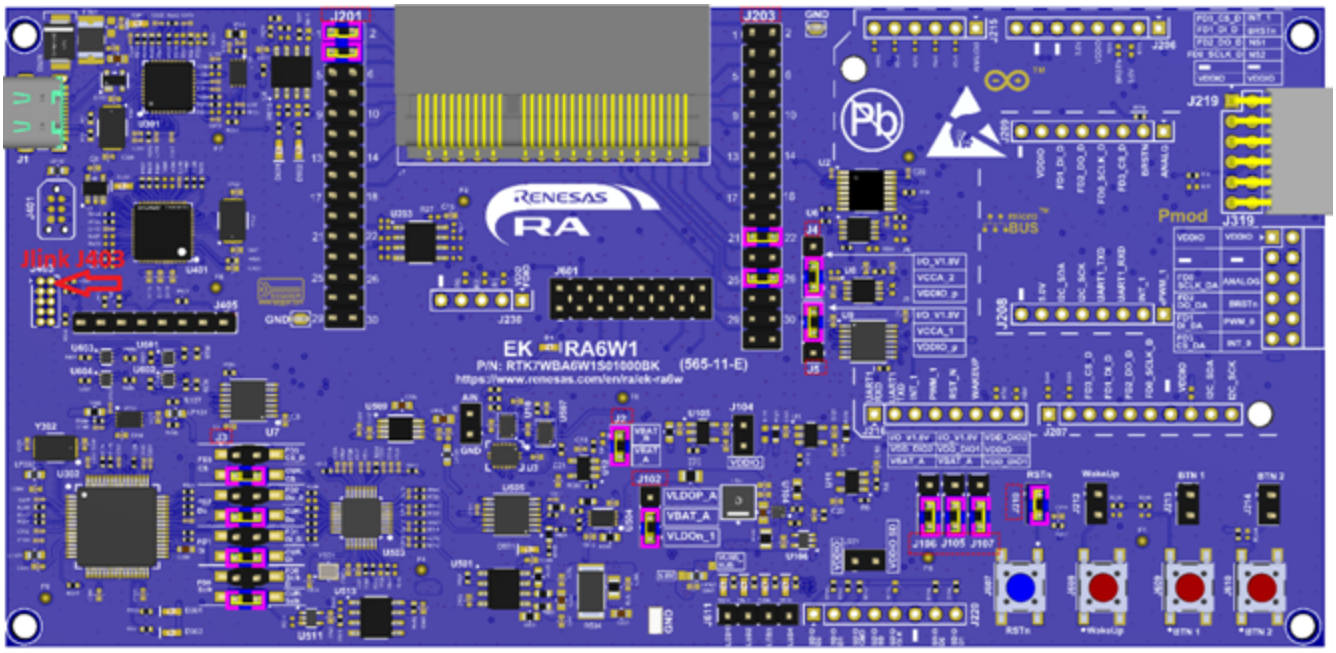

Figure 3. External JTAG J403

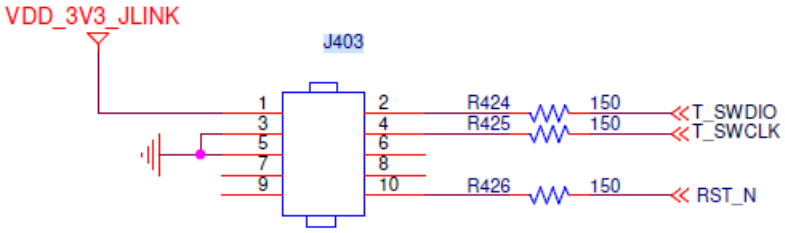

Figure 4. J403 schematic

4. GDB Server

To use the GDB server, download the latest J-Link GDB server from the SEGGER official website: https://www.segger.com/downloads/jlink/. You can start the server using Command Prompt or by running the `.exe` file.
Flashing images to the target device is supported from JLink_V878.
To start the J-Link GDB server using Command Prompt, run the following command:

```
C:\Program Files\SEGGER\JLink_V878>JLinkGDBServer.exe -if SWD -device Cortex-M33 -speed auto -
nolocalhostonly -if SWD -device Cortex-M33 -speed auto -nolocalhostonly
```

Where:

- `if SWD` specifies the interface (Serial Wire Debug).
- `device Cortex-M33` specifies the target device (Cortex-M33).
- `speed auto` automatically sets the speed.
- `nolocalhostonly` allows connections from any network interface, not just localhost.
- `JLinkXXXXXX` depends on J-Link version installed, in this example JLink_V878k.

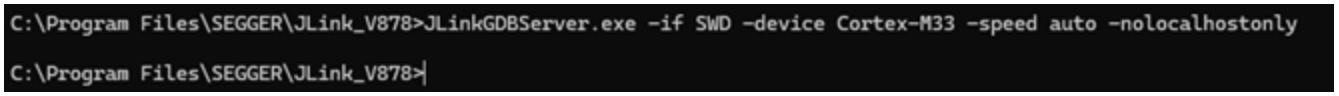

Figure 5. Run GDB server from CMD

Passing arguments to the command line automatically starts the server.
To start the J-Link GDB server by running the executable file, double-click the `JLinkGDBServer.exe` file, and in the configuration window (Figure 7), do the following, and then click **OK**:

- In **Connection to J-Link**, select **USB**.
- In **Target device**, select **Cortex-M33**.
- In **Target interface**, select **SWD** (Serial Wire Debug).

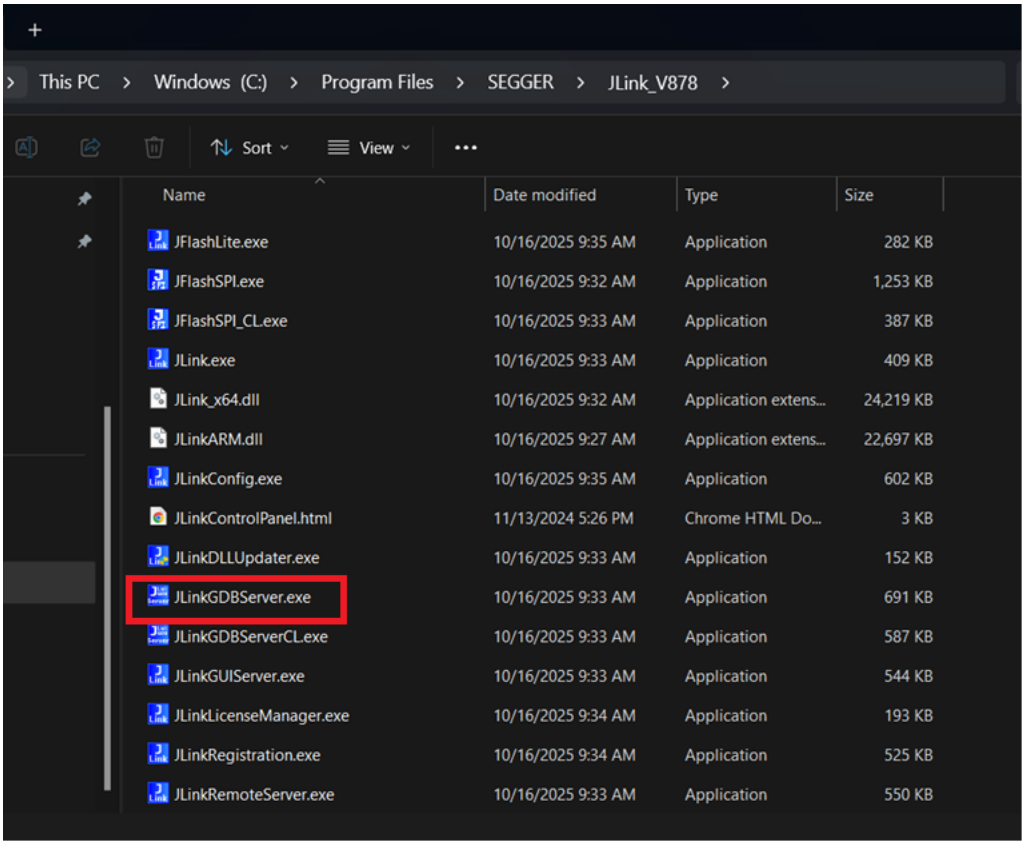

Figure 6. J-Link GDB Server

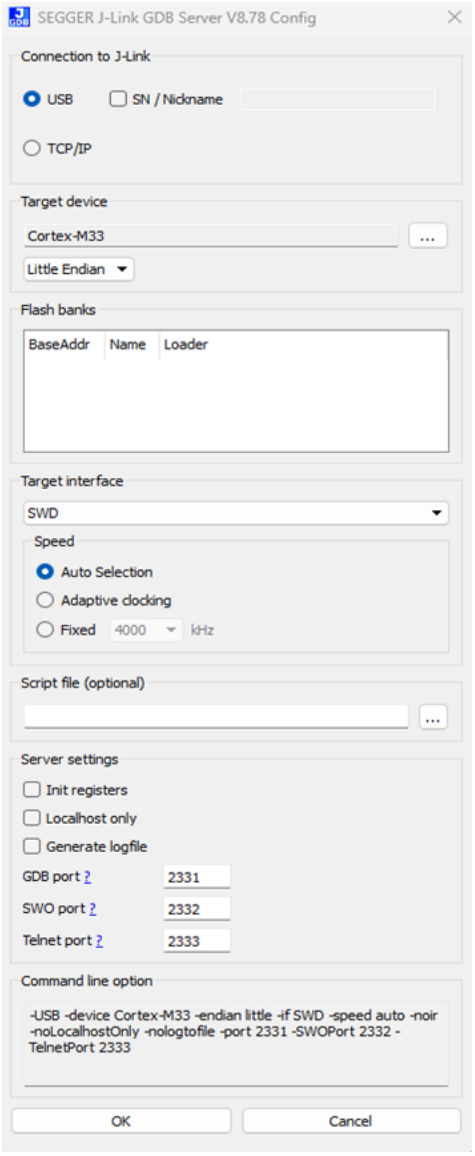

Figure 7. GDB server configuration

After the server starts, everything is configured correctly and you should see "Waiting for connection" status message in the J-Link GDB Server window on the host machine Figure 8.

NOTE
The display may change based on the SEGGER J-Link GDB server version.

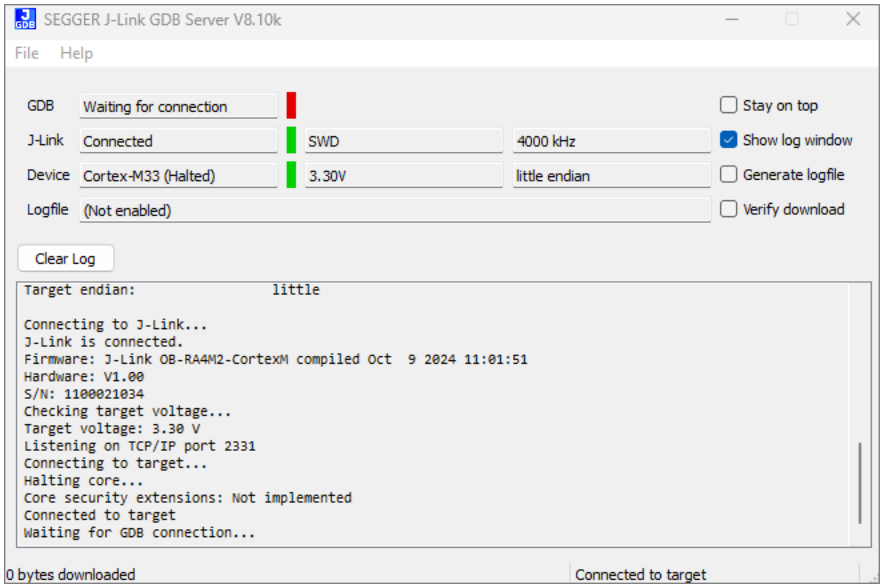

Figure 8. GDB server monitor

5. Programming Firmware Images

To program firmware image, you first need to erase the flash, and then to flash the images.
For flashing firmware images to the EVK, use the `cli_programmer.exe` tool.
You can find the executable inside the `flash_img_downloader.zip` archive at the following path: `..\flash_img_downloader\binaries\cli_programmer.exe`. For more details, see *RA6W1 Getting Started Guide* and *RA6W2 Getting Started Guide*.

5.1 Erase Flash

Address	Name	Description	Size (kB)
0x0000_0000	Product Header		4
0x0000_1000	Product Header - Backup		4
0x0000_2000	RTOS #0	(64*1024 * 40)	2560
0x0028_2000	Reserved	RTOS binary size – Expandable area	508
0x0030_0000	NVRAM	NVRAM VEE Segment #1	40
0x0030_A000	NVRAM	Backup (Multiplier 8)	280
0x0035_0000	TLS_Cert_Base		704
0x0035_0000	TLS Certificate WPA_Enterprise	CA	4 or Unused
0x0035_1000		Cert	4 or Unused
0x0035_2000		Private key	4 or Unused
0x0035_3000		Diffie-Hellmann key	4 or Unused
0x0035_4000	TLS Certificate OTA_Update	CA	4 or Unused
0x0035_5000		Cert	4 or Unused
0x0035_6000		Private key	4 or Unused
0x0035_7000		Diffie-Hellmann key	4 or Unused

Figure 9. SFlash map

1. Verify that the GDB server is started.
2. Open **Command Prompt** and enter the following command: `"C:\flash_img_downloader\binaries\cli_programmer.exe" gdbserver erase_qspi 0x0 0x282000`

   Wait for the message "done" to indicate the operation is complete.

```
C:\>"C:\flash_img_downloader\binaries\cli_programmer.exe" gdbserver erase_qspi 0x0 0x282000
cli_programmer 1.26
Copyright (C) 2023-2024 Renesas Electronics

bootloader file not specified, using internal uartboot.bin

Uploading boot loader/application executable...
chip_rev is 400AA
Executable uploaded.

done.
```

Figure 10. Erase Flash

5.2 Program Firmware Images to the EVK

In the **Command Prompt** window, type the following command: `"C:\flash_img_downloader\binaries\cli_programmer.exe" gdbserver write_qspi 0x0 C:\my_img\RRQ_vm.img`.
Where `C:\my_img\RRQ_vm.img` is an example path for the `.img` file and should be replaced with the actual path of your firmware image.
The "done" message indicates that the operation is completed.

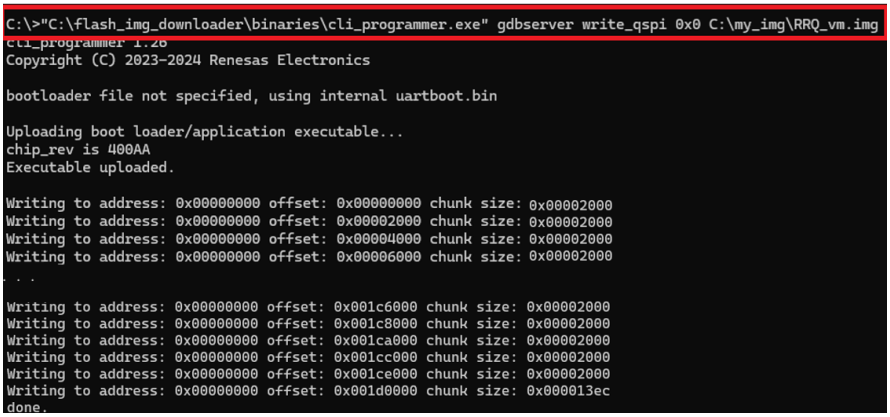

Figure 11. Programming firmware images to the EVK

To check the booting status through UART:

- After the programming firmware is completed, on the EVK, press the **RST_N (Reset)** button, and then wait for the output status.

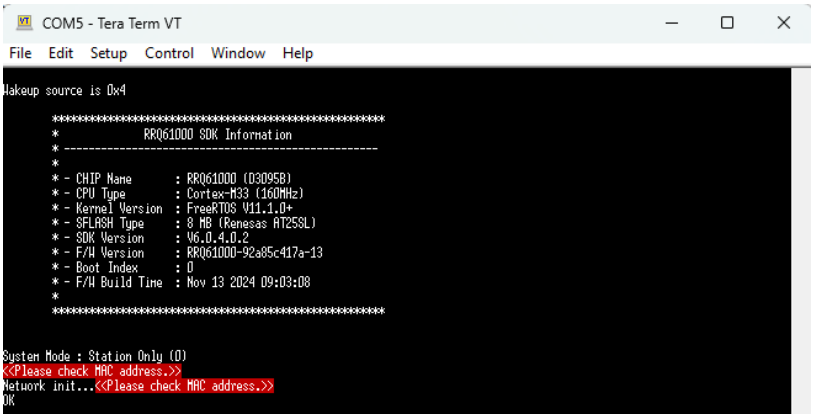

Figure 12. Flash boot with downloaded image

6. Use e^2 studio for Firmware Flashing and Debugging

For RA6W1/RA6W2 projects, you can flash images and perform debugging using Renesas GDB Hardware Debugging. In XIP mode, the system executes code directly from flash, while GDB loads the symbol file for debugging. Therefore, you must flash the image first before starting the debugging session.
The following example uses the official Wi-Fi example application `rm_wifi_test_app`:

1. To import the project:
   a. In the Workspace, go to **File** > **Import** > **General** > **Existing Projects**, and then click **Next**.
   b. In the **Import** dialog, select **Select root directory**, and then click **Browse**.
   c. Use the file manager to navigate to the `<app_root_directory>` directory, and then click **Select Folder**.
   d. Under **Projects**, select the desired application project.
   e. Under **Options**, select **Copy projects into workspace**.

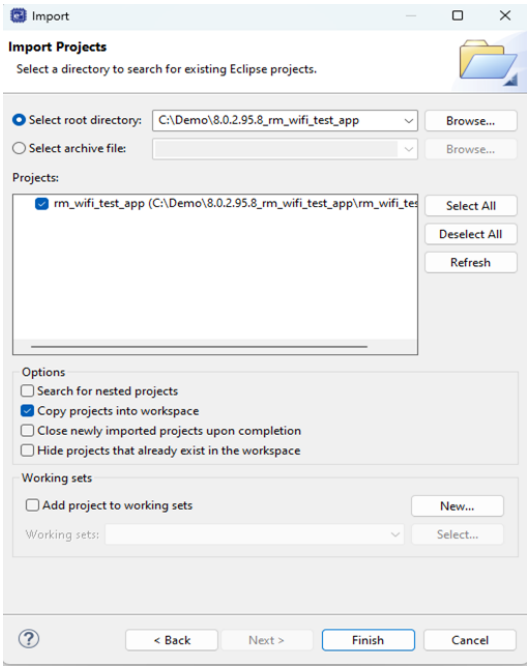

Figure 13. Import Project

2. To remove post-build old configuration, right-click the project and go to **Properties** > **C/C++ Build** > **Settings** > **Build Steps**.
3. Under **Post-build steps** > **Command(s)**, remove the existing content.

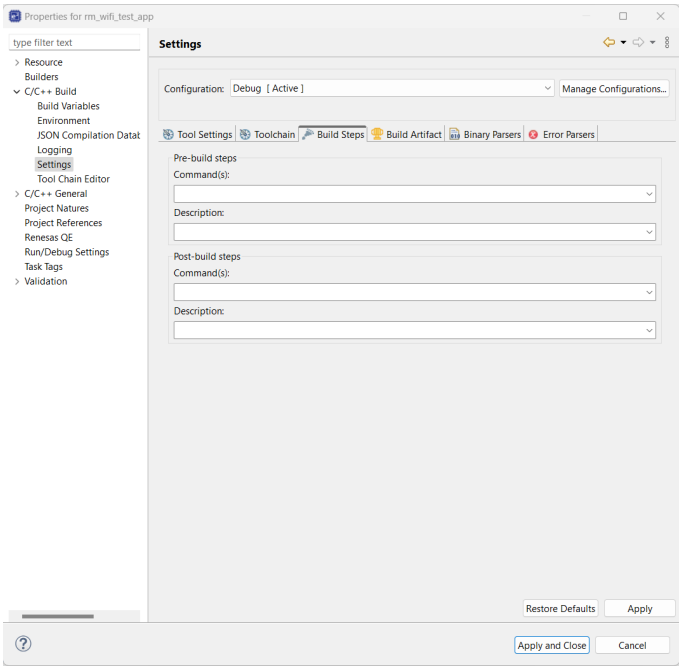

Figure 14. Remove post-build command(s)

4. Click **Build a project** (hammer icon).

NOTE
Make sure the Debug configuration is selected (in the configuration dropdown next to the hammer icon) before building.

There should not be any compilation errors.

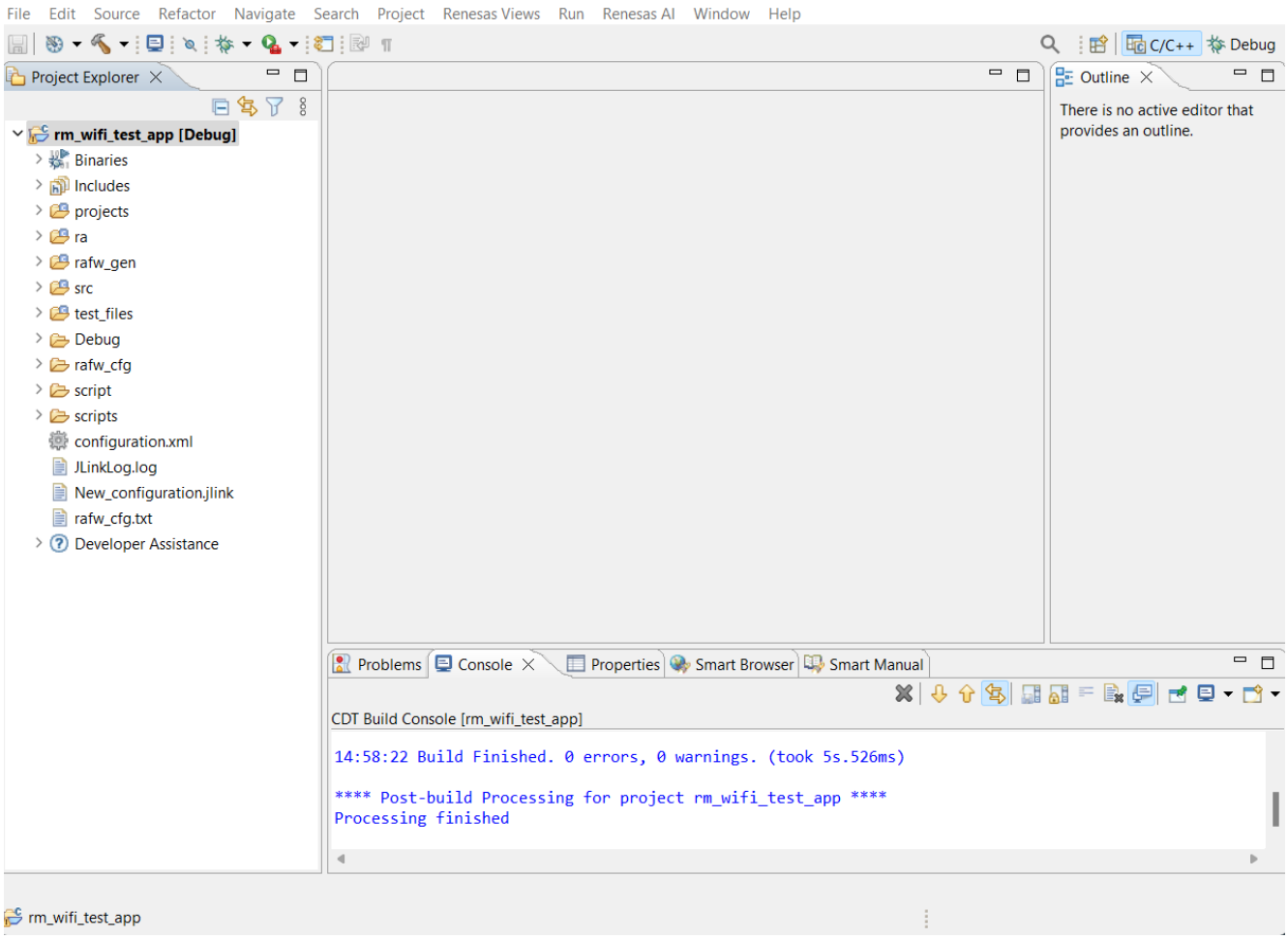

Figure 15. Build project

5. For remote debugging, on the remote device PC, open **J-Link Remote Server**, and select the needed port:

NOTE
When a **local USB connection** is used, this section can be skipped.

- Linux: JLinkRemoteServerCLExe -port 19020.

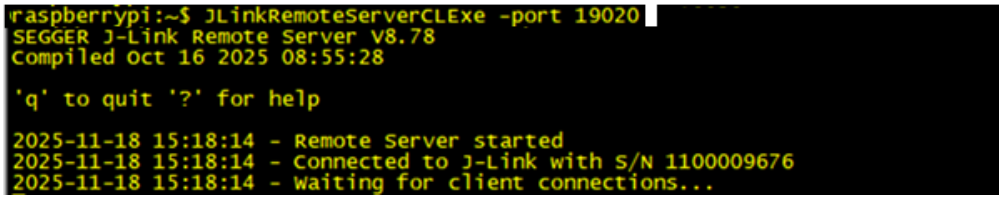

Figure 16. Linux JLinkRemoteServerCLExe

- Windows: JLinkRemoteServer.exe -port 19020.

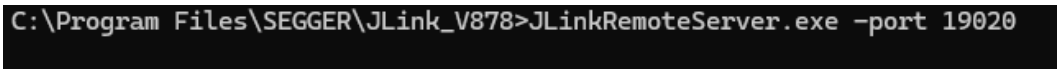

Figure 17. Windows JLinkRemoteServer.exe

6. Go to **Debug** > **Debug Configurations**.

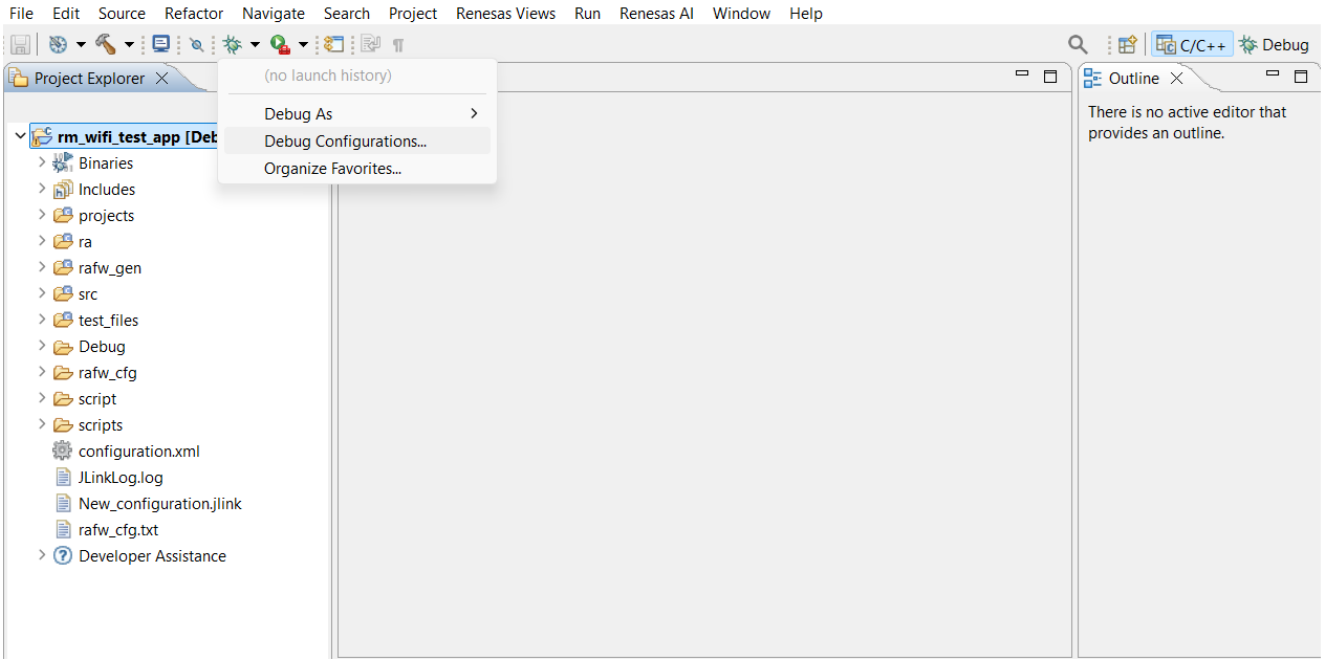

Figure 18. Debug configurations

7. In the **Debug Configurations** dialog, double-click the **Renesas GDB Hardware Debugging** option.
8. On the **Main** tab of your development environment, do the following:
   a. Click **Browse** and select the needed project.
   b. Click **Search Project** and select the `compiled .elf` file.
9. On the **Debugger** tab, under **GDB Settings**, configure the following:
   - In **Debug hardware**, select **J-Link ARM**.
   - In **Target Device**, select the appropriate device (in this example: R7A6W1CEDZDD).

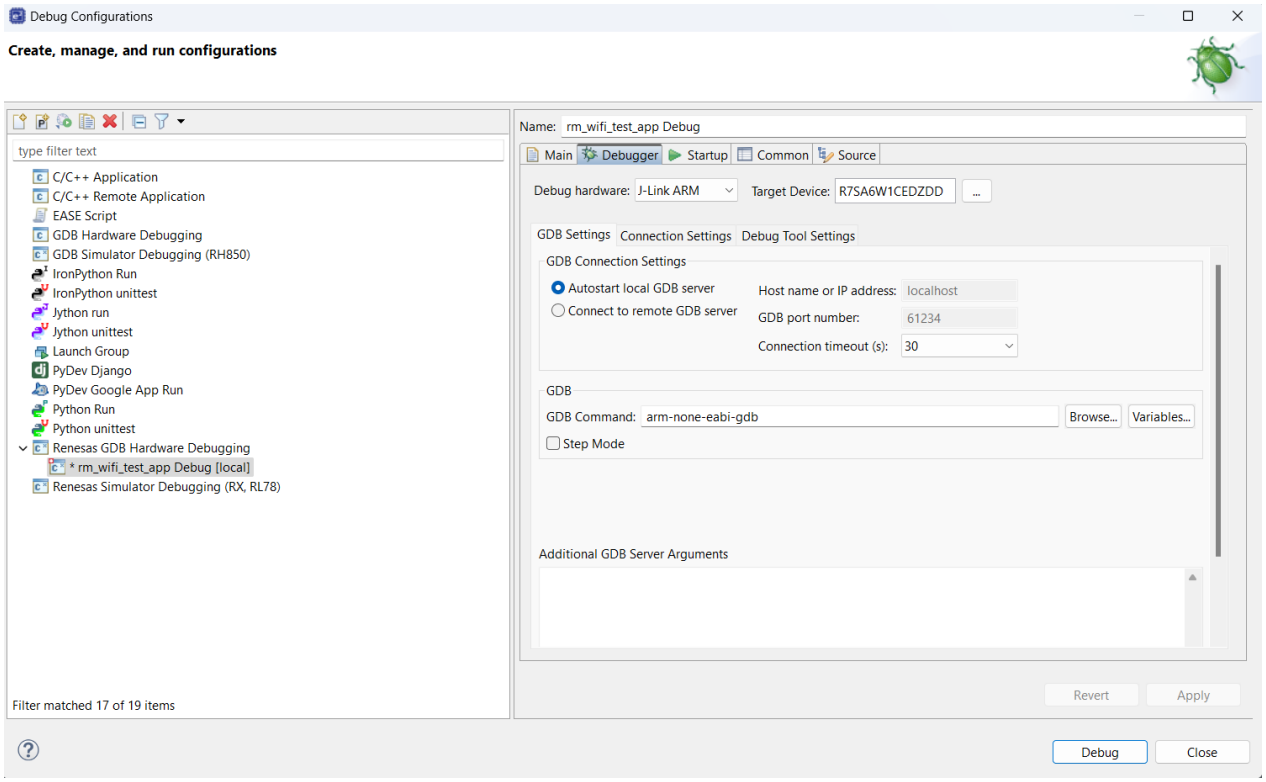

Figure 19. Debugger – GDB Setting configurations

10. On the **Debugger** tab, under **Connection Setting**, configure the following:
    - When a local USB connection is used, in **J-Link Type**, select **USB**.
    - When remote IP connection is used, select the following:
        - For **J-Link Type**, select **IP**.
        - For **IP Connection Method**, select **IP via LAN**.
        - For **Host Name/IP Address [: port number]**, enter **<IP address>: port**.
          For example: 10.14.84.57:19020.

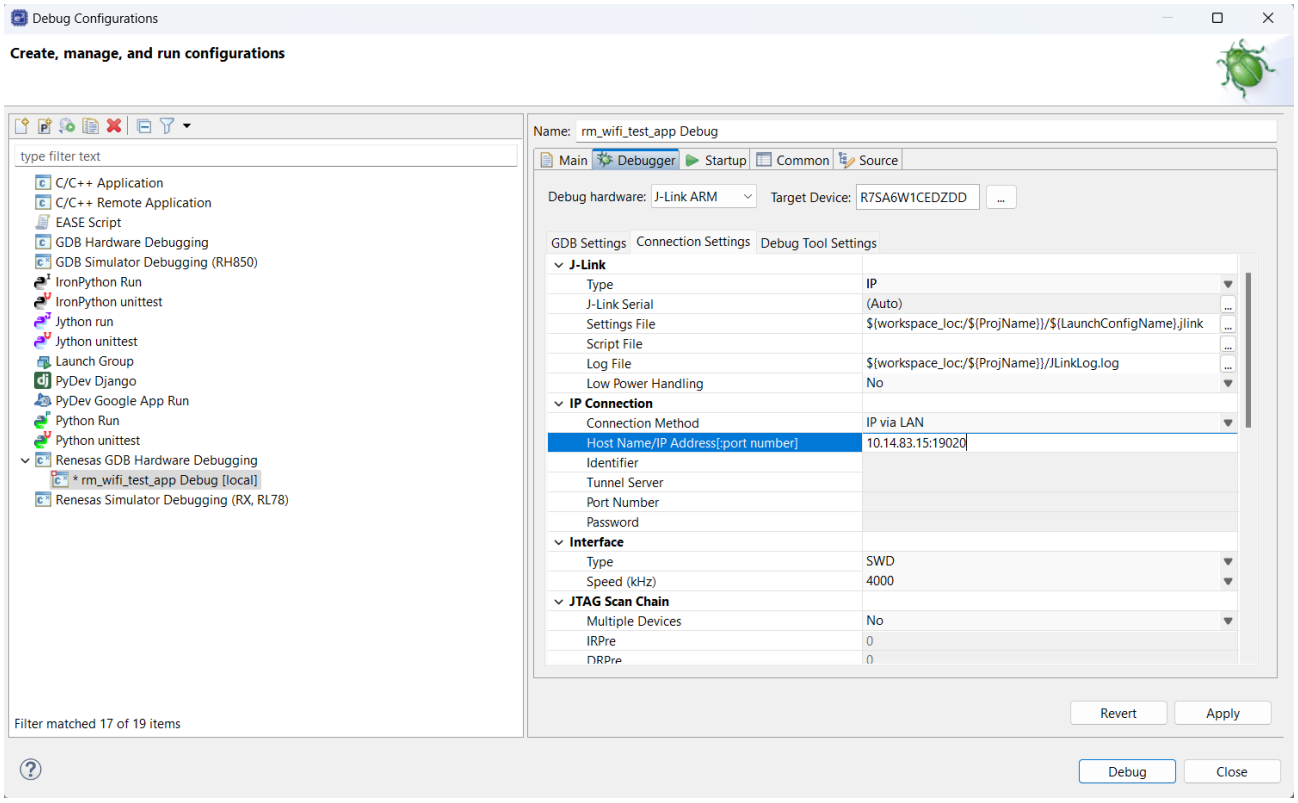

Figure 20. Debugger – Connection Setting tab

11. On the **Startup** tab, configure the following, and then click **Apply**:
    - For the **Program Binary** file (already available), in **Load type**, select **Symbols only**.
    - Add image file by clicking **Add** and selecting **Workspace**, and then choose the needed file (`*.img.bin`). For the selected file:
        - In **Load type**, select Raw Binary.
        - In **Offset**, enter 0x0A000000 (device flash offset).

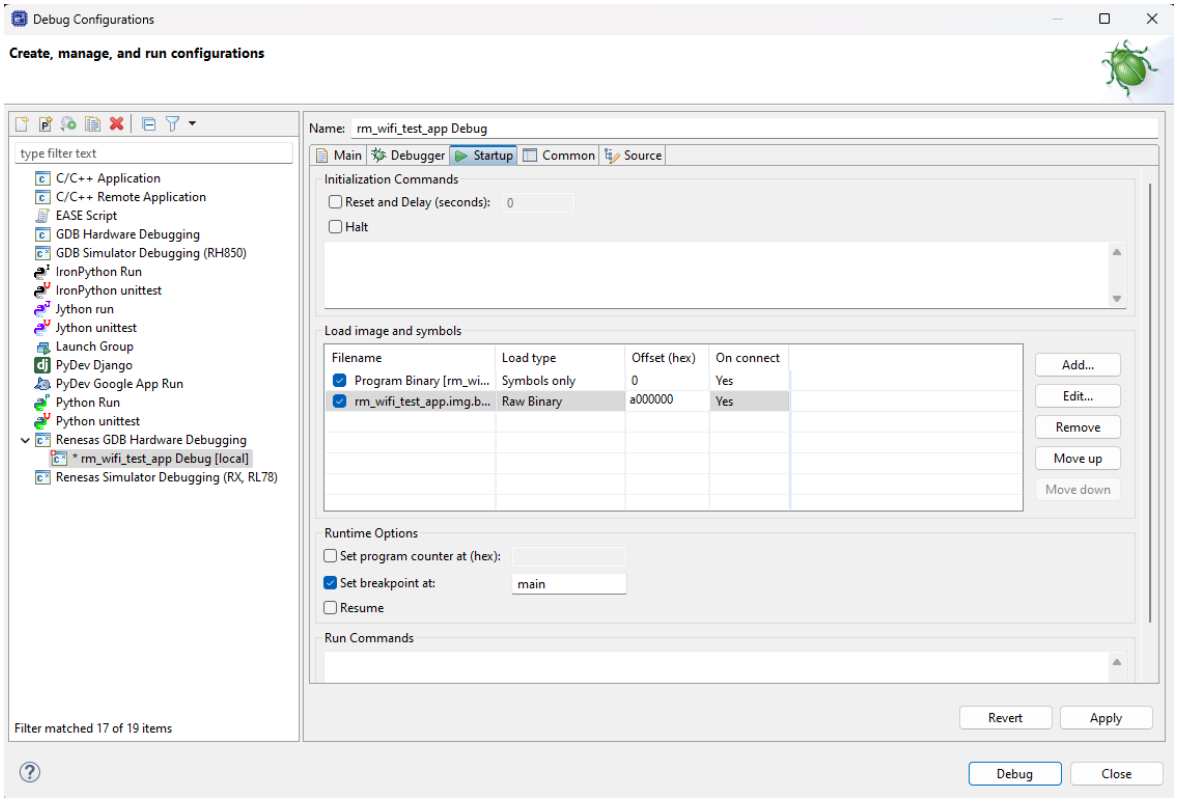

Figure 21. Startup tab

12. To execute debugging, click **Debug**.

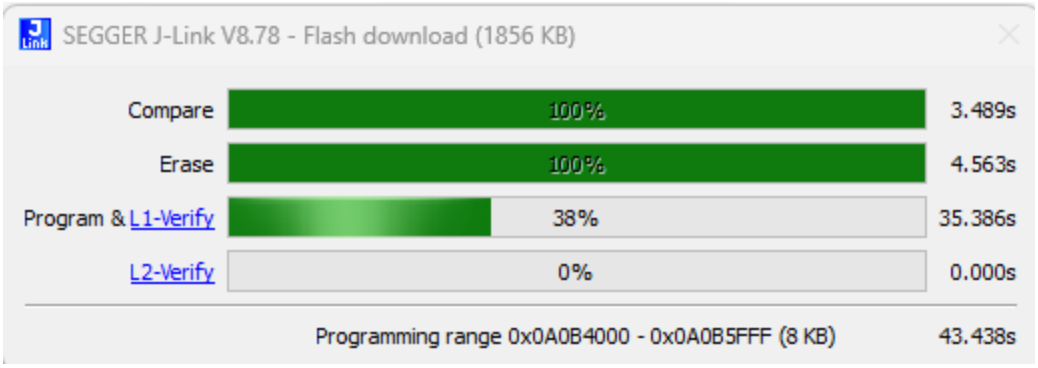

Figure 22. Flash download

6.1 Debugger Console

The **Debugger Console** is used to monitor debugging messages, interact with the target device, and execute various debugger commands. You can view real-time logs and status messages during the debugging, and directly input and execute commands such as breakpoints, stepping through code, and inspecting memory values.
Table 1 describes basic debugging, memory, variable, session management, and help commands.

Table 1. Debugger commands

Command	Description
`monitor reset`	Resets the target system (hardware reset).
`b <function | line>`	Sets a breakpoint at a function or line number.
`info b`	Lists all breakpoints.
`delete <num>`	Deletes a specific breakpoint (use delete to remove all).
`c`	Continues execution until the next breakpoint.
`s`	Steps into the next function call (line-by-line execution).
`n`	Steps over a function (executes it but does not enter it).
`finish`	Runs until the current function returns.
`p <var>`	Prints the value of a variable or memory address.
`x/<format> <address>`	Displays memory at a given address (for example, x/4xw 0x20000000).
`kill`	Stops the debugging session.
`help`	Displays general help.
`help breakpoints`	Shows help related to breakpoints.
`help running`	Shows help for execution-related commands.
`help memory`	Shows help for memory-related commands.

6.2 Example: Set a Breakpoint and Run the Debugger

1. Reset the target system by entering command: `monitor reset`.
2. Set a breakpoint at the `wifi_init` function by entering command: `b wifi_init`.

   GDB Output Confirms that the breakpoint is set:
   Breakpoint 1 at 0x2cee0: file `../rrq/fsp/src/r_wifi_b/r_wifi_b_init.c`, line 1136.
3. Start or continue execution by entering command: `c`.

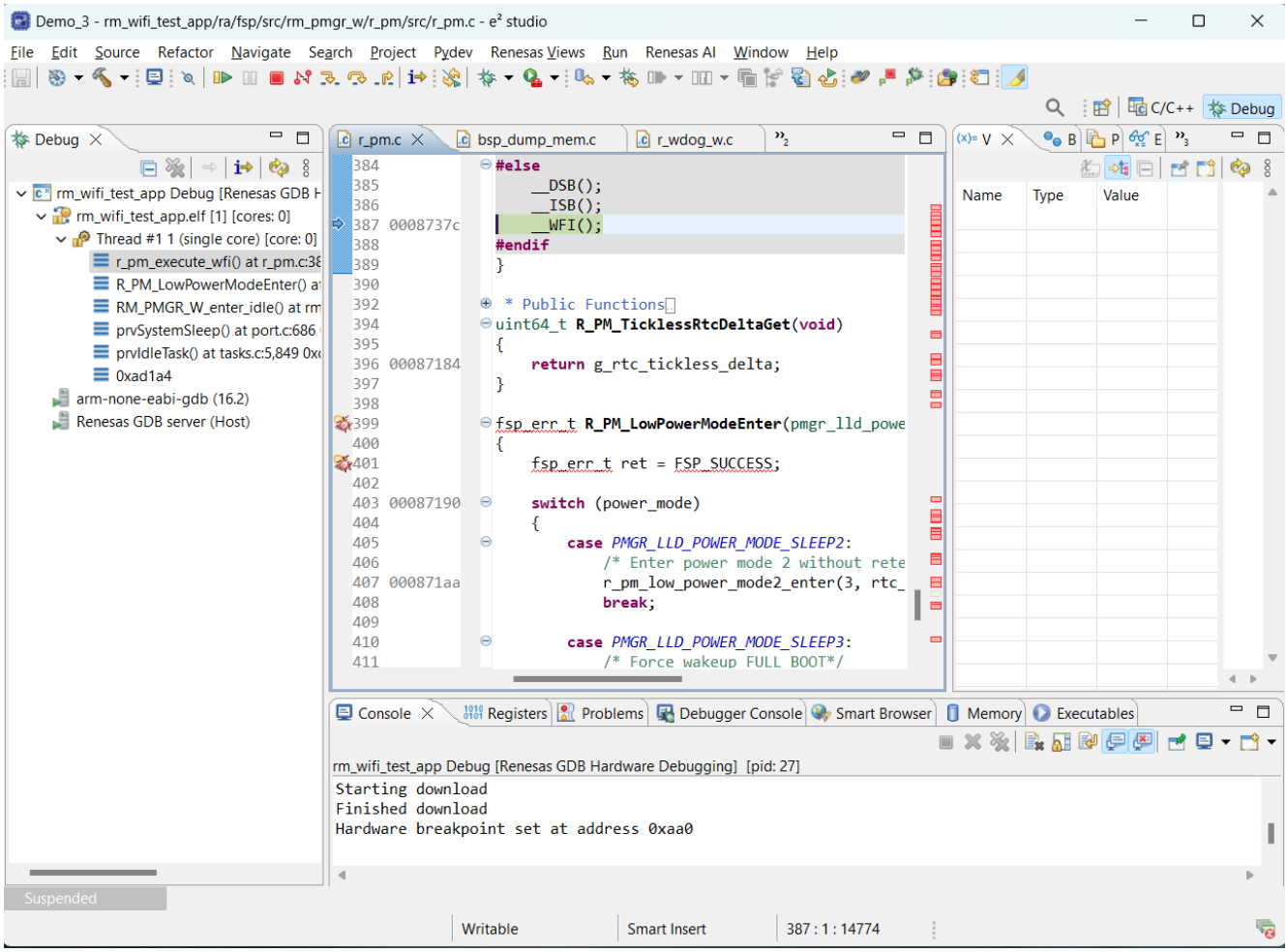

Figure 23. e² studio debugger

7. Revision History

Revision	Date	Description
1.02	May 7, 2026	Updated image SFlash map for Erase Flash. Added RA6W2 as the supported device.
1.01	Dec 11, 2025	Updated sections about erasing flash and using e2 studio for firmware flashing and debugging. Updated images RA4M2 on EK-RA6W1 for using JTAG and External JTAG J403. Added table about Debugger commands.
1.00	Mar 13, 2025	First release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

CorporateHeadquarters

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-us/

© 2026 Renesas Electronics Corporation. All rights reserved.