Renesas RA Family

RA4C1 RA4L1 Secure Factory Programming

Introduction

The Secure Factory Programming (SFP) feature of the Renesas RA4C1 and RA4L1 MCUs is designed to ensure the integrity and confidentiality of firmware during the manufacturing process. Additionally, the SFP process also verifies the authenticity of the MCU. This robust solution allows for secure programming without the need for special software or dedicated hardware, making it an ideal choice for factory programming in a non-secure environment.

This application note introduces the MCU boot firmware features and tool support to utilize SFP for RA4 MCUs. Guidelines and procedures for utilizing these features and tools are provided. The primary SFP operations, including firmware image encryption and device lifecycle management (DLM), are explained in detail. In addition, SFP is discussed with other security solutions, such as TrustZone, to highlight its role within the overall security architecture.

EK-RA4C1 is used to evaluate the SFP solution in this application note. The procedure steps and screenshots used EK-RA4C1 as an example. The same procedure is applicable for all the Target Devices.

Required Resources

The following resources are referenced throughout this application note.

Development Tools and Software

- Renesas Flash Programmer (RFP) v3.21
  https://www.renesas.com/us/en/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html
- Renesas Security Key Management Tool v1.11
  https://www.renesas.com/software-tool/security-key-management-tool
- Gpg4win
  http://www.gpg4win.org/
- Renesas Trusted Secure IP Key Wrap Service
- e²studio IDE v2025-12
  https://www.renesas.com/en/software-tool/e-studio

Target Devices

Below are the Renesas MCU products to which the information within this document is applicable:

- RA4C1
- RA4L1

Hardware

- Evaluation Kit for RA4C1 MCU Group (renesas.com/ra/ek-ra4c1)
- The description in the application note uses a PC running Windows® 10 OS as an example. Refer to the corresponding user manual for the Development Tools and Software for a complete list of Operating Systems supported.
- One USB-C cable to connect the EK-RA4C1 and the PC.

Prerequisites and Intended Audience

This application note assumes the user has experience with the Renesas Flash Programmer (RFP), the Renesas Security Key Management Tool (SKMT), and the e²studio IDE. In addition, background knowledge of RA4 MCU security features is a prerequisite for utilizing the SFP feature, for example, Arm®TrustZone®, Renesas Secure IP (RSIP-E31A), etc. Please refer to the chapters on *Security Features* and *Renesas Secure IP (RSIP-E31A)* in the *Renesas RA4C1 Group MCU User's Manual: Hardware* and the application notes listed in the References section to acquire the background knowledge.

The procedure described in this application note refers to key sections in the following two related application notes. It is recommended that these application notes be reviewed when performing Secure Factory Programming:

- Device Lifecycle Management for Cortex-M33 (R11AN0469).
- Injecting and Updating Secure User Keys (R11AN0496).

The intended audience includes product developers, product manufacturers, product support, or end users who are involved in Secure Factory Programming of the RA Family MCUs.

Contents

1. Introduction to Secure Factory Programming for Supported RA MCUs 4
1.1 Existing Secure Programming Solutions 4
1.2 Renesas RA4C/L1 Secure Factory Programming Features 4
1.2.1 Development Tools 5
1.2.2 High Level Operational Flow 6
1.2.3 Boot Firmware Version Requirement for SFP on RA4C1 and RA4L1 6
1.3 Final DLM State after Secure Factory Programming 6
2. Cryptographic Key Preparation for Secure Factory Programming 7
2.1 Generate the UFPK and Wrapped UFPK 8
2.2 Image Encryption Key 8
2.3 Device Lifecycle Management Keys 8
3. Firmware Encryption and Programming 9
3.1 Set up the Hardware and Initialize the MCU 9
3.2 Using the Flat Blinky Project as an Example for SFP 10
3.2.1 Create a Flat Blinky Project 11
3.2.2 Encrypt the firmware image with SKMT 13
3.2.3 Program the Encrypted Image with RFP 17
3.3 Using SFP with TrustZone Projects 19
3.3.1 Create Blinky Solution with a TrustZone Project chain 19
3.3.2 Encrypt the Secure and Non-secure Firmware Images 20
3.3.3 Program the TrustZone Encrypted Image with RFP 22
4. Appendix A - Disable Initialize command Using RFP. 23
5. References 27
6. Website and Support 28
Revision History 29

1. Introduction to Secure Factory Programming for Supported RA MCUs

1.1 Existing Secure Programming Solutions

Secure MCU programming is a critical aspect of ensuring the integrity and confidentiality of embedded systems. One of the primary goals of secure programming is to protect the binary code from being exposed during the programming process. This is essential for safeguarding intellectual property (IP) and sensitive information, including cryptographic keys. A common approach involves delivering the programming files, such as S-record files, in an encrypted format to the programming facility. The device programmer then decrypts these files just before programming the MCU.

However, the above method has vulnerabilities, particularly concerning the physical programming pins on the MCU, which are susceptible to sniffing attacks. In addition, if there is any doubt about the secure handling of data by the programming facility, this approach may not provide sufficient protection, raising concerns about the security of the entire programming process.

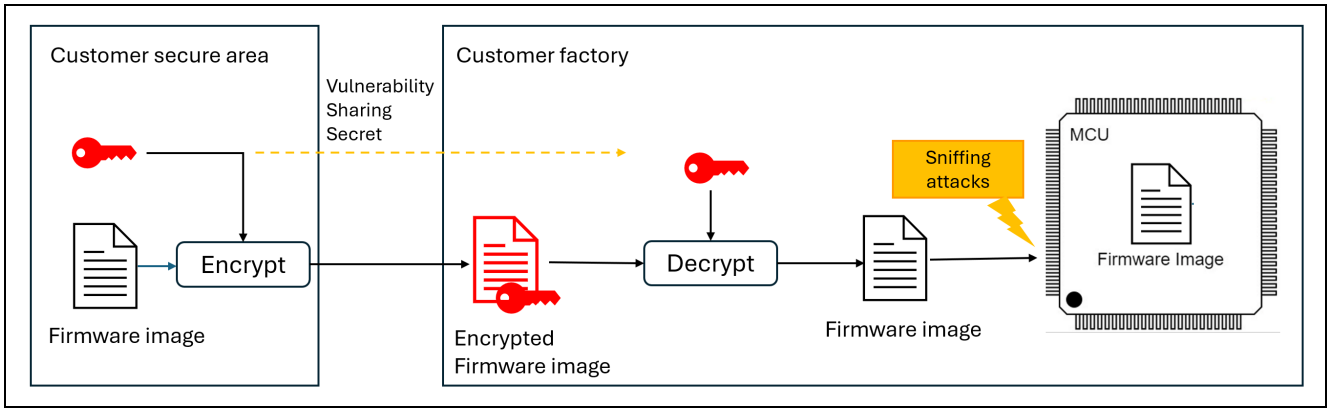

Figure 1. Existing Secure Programming Solution

To address the vulnerabilities associated with the transport of the firmware over the physical interface of the MCU during programming, more robust solutions are required. One such solution involves using an MCU that has been pre-programmed with a secure bootloader. This secure bootloader ensures that the device can verify the integrity and authenticity of the software being programmed, but the challenge is still to find a solution to securely program these pre-programmed MCUs.

Additionally, secure key management is crucial in this process, requiring the use of a Hardware Security Module (HSM). The HSM provides a tamper-resistant environment for storing and handling cryptographic keys, ensuring that the keys are managed securely.

Moreover, secure MCU programming may require special, dedicated hardware and interfaces that protect the communication between the programmer and the MCU to secure bootloaders and HSMs. These interfaces ensure that even if an attacker gains physical access to the programming setup, they cannot intercept or modify the programming data.

Implementing these security measures may incur higher costs, with multiple complex steps and procedures to protect the integrity of the programming process.

1.2 Renesas RA4C/L1 Secure Factory Programming Features

Building on the basic principles of secure programming technology for MCUs, Renesas Secure Factory Programming (SFP) offers an advanced, integrated solution specifically designed to enhance security during the programming process.

The main advantage of Renesas SFP is that its security features are built into the MCU itself, eliminating the need for external hardware or additional complex steps needed for security considerations. These security features are made available across all of the standard MCUs ' boot firmware serial interfaces, such as USB, UART, and JTAG/SWD. The integration of the security features at this level means that manufacturers can program MCUs through familiar processes while still maintaining a high level of security, which reduces both the complexity and cost of secure programming, making it easier to implement across a wide range of production environments.

Renesas SFP begins by encrypting the firmware and sensitive data at the source, usually at the developer's site. Then the protected programming package is sent to the manufacturing facility. At the factory, the encrypted data is programmed directly into the MCU. Decryption occurs only within the secure environment

of the MCU itself, ensuring that the sensitive code is never exposed during the process, even to the programming facility. This mitigates the risk of sniffing attacks or tampering at the manufacturing stage.

Figure 2 provides a conceptual presentation for the Renesas SFP process.

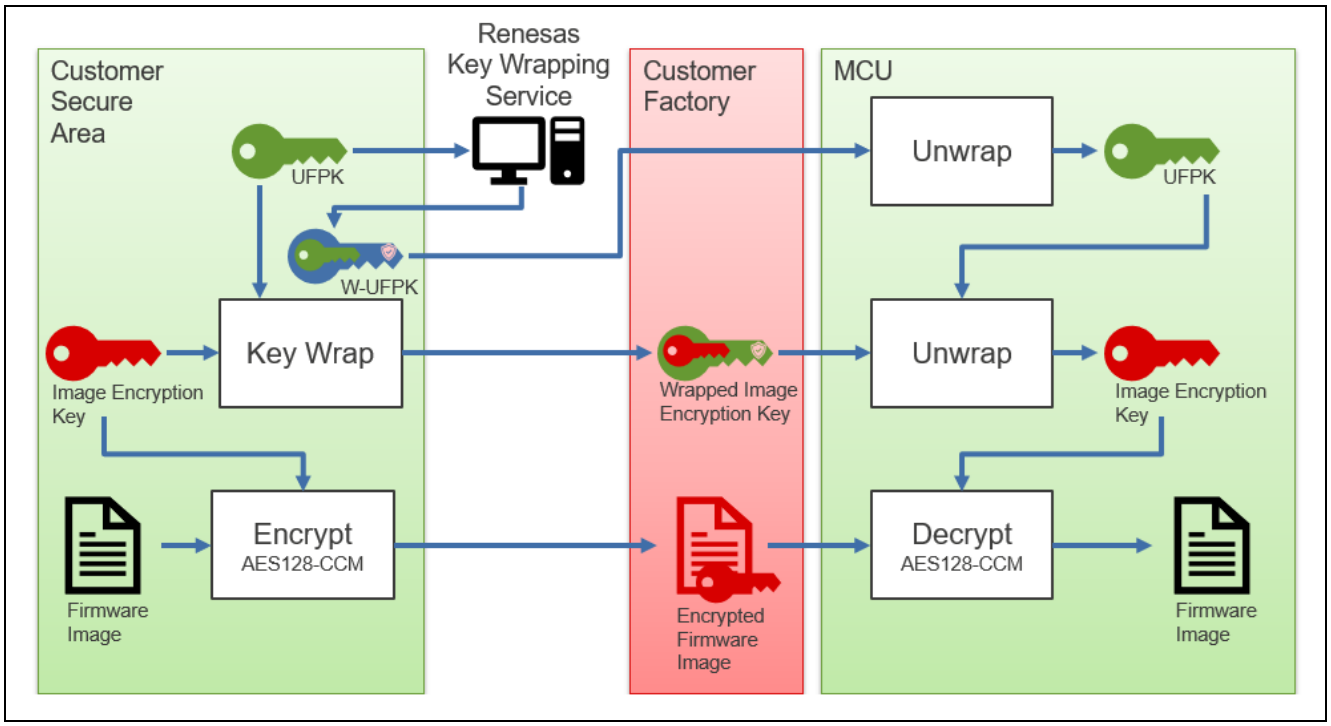

Figure 2. Secure Factory Programming Process

To enable secure factory programming in a non-secure environment, both the DLM state and the authentication keys can be configured using a single boot firmware command, the Encrypted Data Write Command (0x1Ah).

For detailed information, refer to Section “Encrypted Data Write Command” in Application Note R01AN7910, Renesas Boot Firmware for RA4C1 MCU Group.

In this application note, the programming flow described in the Encrypted Data Write Command section is simplified by using the Renesas Flash Programmer (RFP) tool to issue the required boot firmware commands.

Note: Encrypted firmware programming can be performed only when the MCU is in the SSD state.

1.2.1 Development Tools

Renesas provides the following tools to significantly simplify the SFP solution operational procedure:

- **Security Key Management Tool (SKMT):** This tool implements a procedure to encrypt the firmware image to support the Secure Factory Programming concept described in Figure 2. Once the firmware image encryption is complete, the tool generates a secure programming file, which includes the encrypted firmware image, the W-UFPK, the Wrapped Image Encryption Key, and the optional Wrapped Authentication Level Keys. This file will be required for the secure programming operation.
- **Renesas Flash Programmer** (RFP): This tool interfaces with the MCU boot firmware and can be used to securely program the secure factory programming file generated by SKMT and lock the MCU.
- Additionally, a third-party PGP program should be used during the creation process of the Wrapped UFPK. It is used to establish a PGP-encrypted communication channel between the user and the Renesas Key Wrap Service server. Using this tool, the user can generate a PGP key pair, perform key exchange with the Renesas DLM server, and receive the W-UFPK.

1.2.2 High-Level Operational Flow

The following is a high-level summary of the steps using the tools to establish an SFP solution. The detailed procedure is described in Chapters 2 and 3.

- Prepare the keys: Generate **UFPK**, **W-UFPK**, **Image Encryption Key**, and Authentication Level Key (**SECDBG_KEY** and **NONSECDBG_KEY**). These procedures are described in Section 2.
- Prepare the application firmware with the normal RA application development procedure.
- Wrap the image encryption key, the authentication level key, and generate the encrypted SFP package. This is achieved using SKMT. The detailed procedure is described in Section 3.
- Program the encrypted firmware image: This can be achieved using RFP. The detailed procedure is described in Section 3. Note that in this process, secure image programming and the image encryption key injection happen in one shot without MCU reset.

Note that when Secure Factory Programming is performed through the boot firmware interface (the SCI-UART interface in SCI Boot Mode or the SWD interface in SWD Boot Mode), the entire code flash and data flash, except the option-setting memory, are erased first prior to programming the application firmware.

For detailed information on the operating modes, refer to the “Operating Modes” section in the RA4C1 Group User’s Manual: Hardware.

1.2.3 Boot Firmware Version Requirement for SFP on RA4C1 and RA4L1

The MCU boot firmware version must meet the following requirements to support SFP:

- For RA4L1, version V13.0.2 or later is required.
- For RA4C1, version V12.1.2 or later is required.

The boot firmware version can be checked using the Renesas Flash Programmer (RFP). Refer to the RFP User’s Manual for instructions on how to connect to the kit. After the connection is established, select **Target Device** and then choose **Read Device Information** to view the boot firmware version.

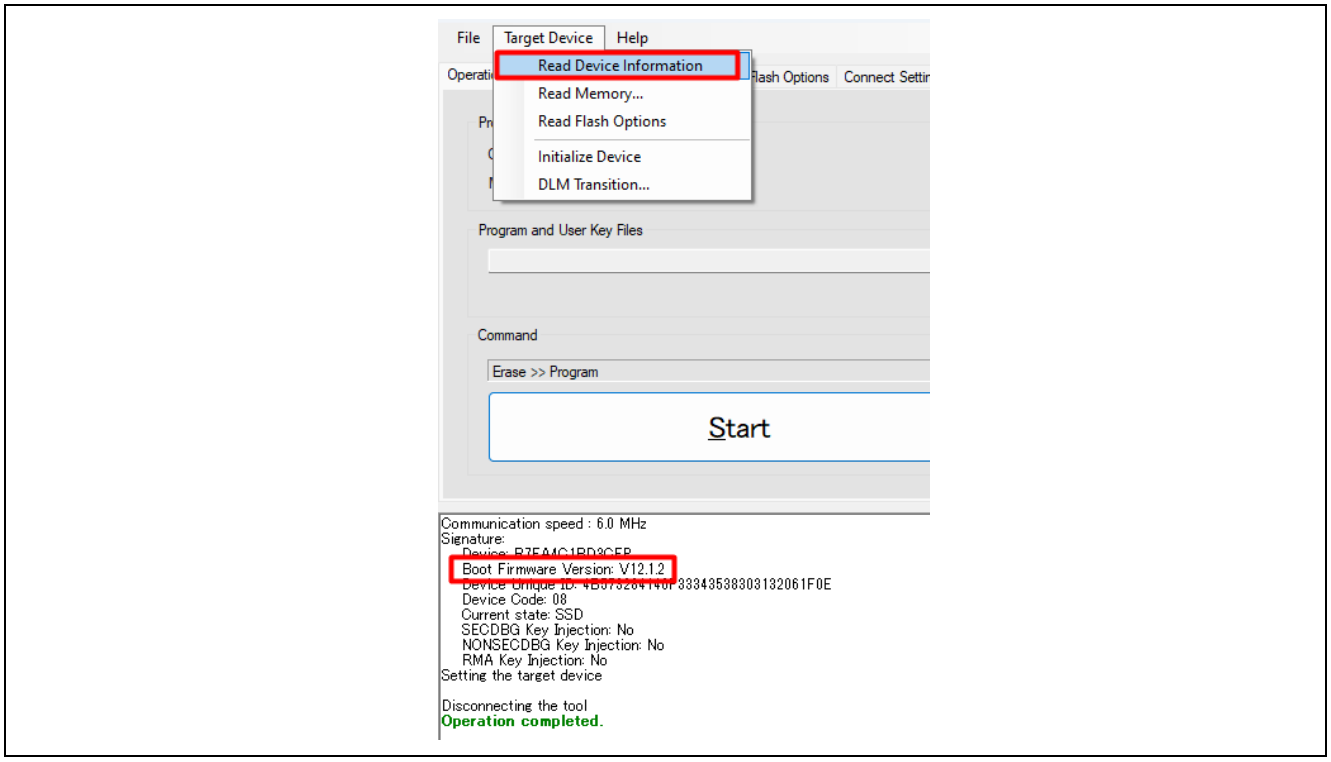

Figure 3. RA4 Boot Firmware Version

1.3 Final DLM State after Secure Factory Programming

With SFP, both DLM key injection and encrypted image programming can be performed through a single boot firmware command, as shown in the Figure 4.

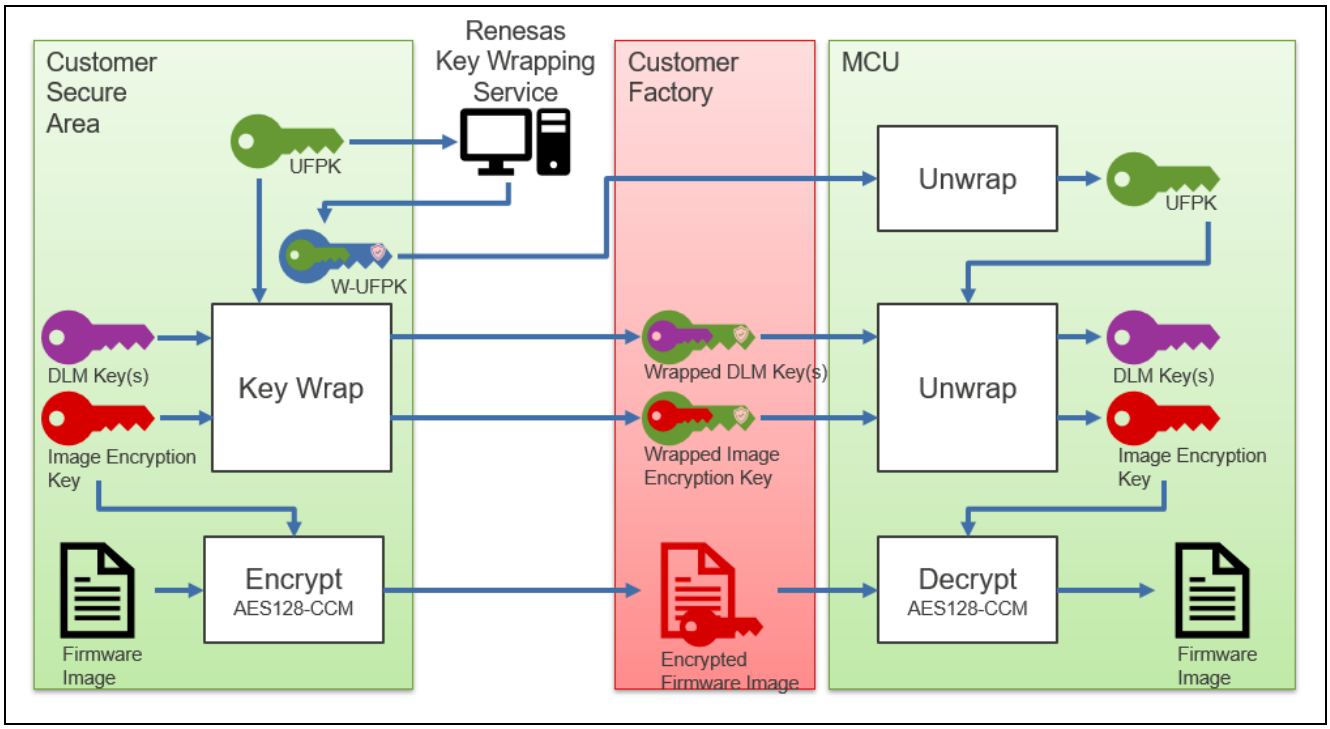

Figure 4. Secure Factory Programming Process with DLM Key Injection

The Secure Key Management Tool (SKMT) provides different options for the final DLM state depending on the RA device series that supports SFP. For the RA4C1 and RA4L1 in particular, SKMT supports two possible final DLM states:

- **DPL with SECDBG_KEY and NONSECDBG_KEY:** In this state, the device is in the field, and its lifecycle state is Deployed (DPL), with both **SECDBG_KEY** and **NONSECDBG_KEY** provisioned. Serial programming remains available. However, access to the code flash and data flash areas is blocked, and the debug interfaces are locked.

  To change the debug access level, the appropriate DLM key must be used to authenticate the corresponding lifecycle state transition. For example, the **SECDBG_KEY** is used to authenticate the transition from **NSECSD** to **SSD**, enabling the **DBG2** debug access level, while the **NONSECDBG_KEY** is used to authenticate the transition from **DPL** to **NSECSD**, enabling the **DBG1** debug access level.

  If the initialization command is permitted in this state, the device can be re-initialized to return to the **SSD** state**.** Use RFP to confirm that the initialization command is still permitted, as shown in Figure 7 and verify the successful output message indicating the transition to the SSD state, as shown in Figure 8**.**

  Disabling the initialization command should be performed with caution. In production environments, this configuration may be desirable when the MCU is intended to remain in the Deployed (DPL) state. However, it is not recommended during the development stage.

  The Initialize Command can be disabled using the Renesas Flash Programmer (RFP). For detailed instructions, refer to Appendix A, *"Disable Initialize Command Using RFP".*

- **LCK_BOOT**: Final state is LCK_BOOT. MCU is locked; the debug interface and the serial programming interface are permanently disabled. Exercise caution when transitioning the device to this final DLM state during the SFP process. It is not recommended to exercise this end state during the development stage.

2. Cryptographic Key Preparation for Secure Factory Programming

This section describes the preparation of the various cryptographic keys used during SFP operations.

Note that when Secure Factory Programming is used to program MCUs for integration into a final product, non-trivial keys that are not duplicated from Renesas examples or other publicly available cryptography references must be used for the UFPK, the Image Encryption Key, SECDBG_KEY, and NONSECDBG_KEY.

2.1 Generate the UFPK and Wrapped UFPK

A User Factory Programming Key (UFPK) and a Wrapped User Factory Programming Key (W-UFPK) are necessary for wrapping the Image Encryption Key.

Refer to section ***Establish PGP-Encrypted Communication with the Renesas DLM Server*** in Application Note R11AN0469 to establish the PGP-encrypted communication channel with the Renesas DLM server.

Refer to section ***Create the UFPK and W-UFPK*** in Application Note R11AN0469 to generate UFPK and W-UFPK. The rest of this application project assumes that the following UFPK and W-UFPK have already been created:

- `ra4c1_ufpk.key`: the UFPK.
- `ra4c1_ufpk.key_enc.key`: the W-UFPK wrapped by the DLM server.

2.2 Image Encryption Key

Specify the AES 128-bit key data to be used for encryption. This example uses a NIST test vector as provided by the Cryptographic Algorithm Verification Program (CAVP).

https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/aes/AESAVS.pdf

```
KEY = 80000000000000000000000000000000
IV = 00000000000000000000000000000000
PLAINTEXT = 00000000000000000000000000000000
CIPHERTEXT = 0edd33d3c621e546455bd8ba1418bec8
```

Figure 5. NIST AES 128 Test Vector

2.3 Device Lifecycle Management Keys

The following sections provide a brief description of the **SECDBG_KEY** and **NONSECDBG_KEY**.

For more information on Device Lifecycle Management (DLM) and authentication keys for the RA4 series, refer to Application Note R11AN0469, *Device Lifecycle Management for Cortex-M33*.

- **SECDBG_KEY** can be injected in the **SSD** state. It is used to authenticate the lifecycle transition from **NSECSD** to **SSD**.
- **NONSECDBG_KEY** can be injected in the **SSD** state or **NSECSD** state. It is used to authenticate the lifecycle transition from **DPL** to **NSECSD**.

Each device lifecycle state provides a specific level of functionality and access control. The lifecycle state of the device can be changed using the corresponding authentication keys described above.

- In the **Secure Software Development (SSD)** state, debugger connections are allowed, and there are no restrictions on access to memory and peripherals.
- In the **Non-Secure Software Development (NSECSD)** state, debugger connections are allowed. However, access is restricted to non-secure memory regions and peripherals only.
- In the **Deployed (DPL)** state, debugger connections are not permitted. In addition, serial programming is available, but access to the code flash and data flash areas is restricted.

In the SFP programming demonstration described in this application note, both the DLM state and the authentication keys are provisioned through a single boot firmware command, resulting in a final DLM state of **DPL with SECDBG_KEY and NONSECDBG_KEY.**

In this example, the following 128-bit raw key is used as the key data:

SECDBG_KEY: `000102030405060708090A0B0C0D0E0F`

NONSECDBG_KEY: `9f3c7a12b4e86d5a2c9180f7e34b6d1a`

In addition to the authentication keys described above, an **RMA_KEY** is also defined. This key is used to authenticate lifecycle transitions from **SSD** or **DPL** to **RMA_REQ**. The **RMA_KEY** is not described in detail in this document. If an **RMA_KEY** is needed, it must be injected using the process described by R11AN0469, "*Device Lifecycle Management for Cortex-M33,"* prior to using Secure Factory Programming.

3. Firmware Encryption and Programming

This section provides a demonstration of SFP using DPL with SECDBG_KEY and NONSECDBG_KEY as the final DLM state. To avoid accidentally disabling the MCU from debugging and initialization, be very cautious when setting the MCU to LCK_BOOT state during SFP operations. It is not recommended to exercise this end state during the development stage.

3.1 Set up the Hardware and Initialize the MCU

Follow the two guidelines below to connect the EK-RA4C1 to the PC.

- Use the default jumper setting on the EK-RA4C1. Refer to the EK-RA4C1 User's Manual to confirm the jumper settings.
- Connect J10 on the EK-RA4C1 to the development PC using a USB Type-C to Type-C cable to provide power, debug access, and access to the MCU boot firmware.

For a smooth evaluation of the SFP solution, it is recommended to initialize the device using the Renesas Device Partition Manager (RDPM) or Renesas Flash Programmer (RFP) prior to proceeding to the rest of the operations. The following shows the usage of RDPM. For RFP usage, please refer to the RFP User's Manual.

Launch e²studio, then launch the **RDPM.**

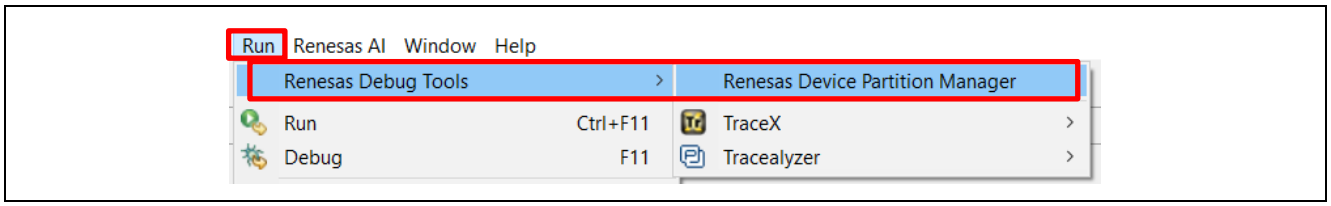

Figure 6. Open the Renesas Device Partition Manager

Next, select Initialize device to factory default settings, choose the connection method, and then click Run.

For the EK-RA4C1, select either SCI or SWD as the connection method according to the jumper configuration (refer to the EK-RA4C1 User's Manual for jumper settings). For a custom PCB, select the connection type based on the available boot mode interfaces.

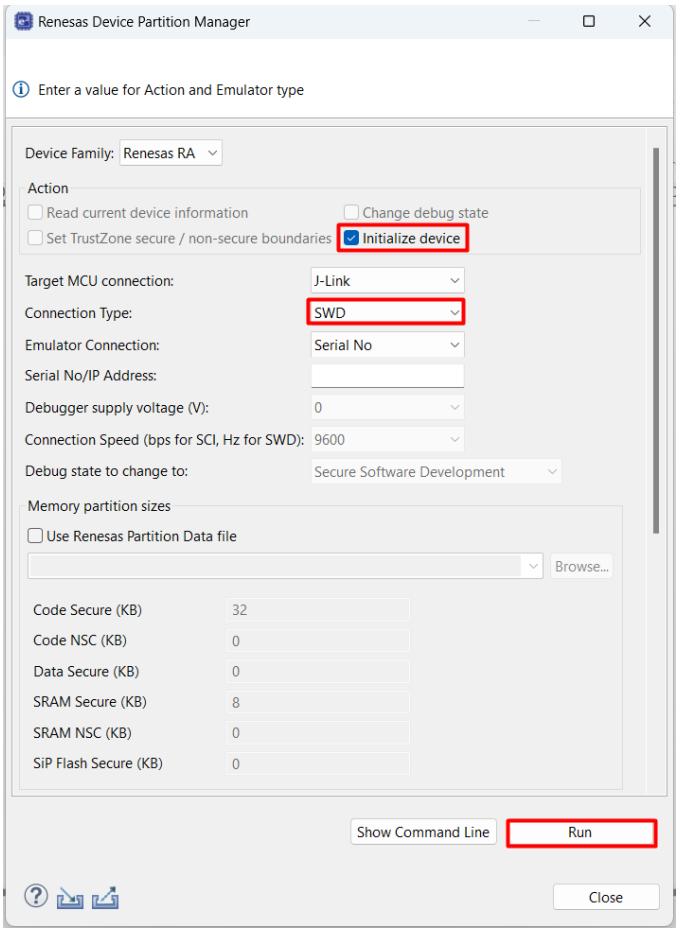

Figure 7. Initialize RA4C1 using Renesas Device Partition Manager

Ensure the following output is achieved.

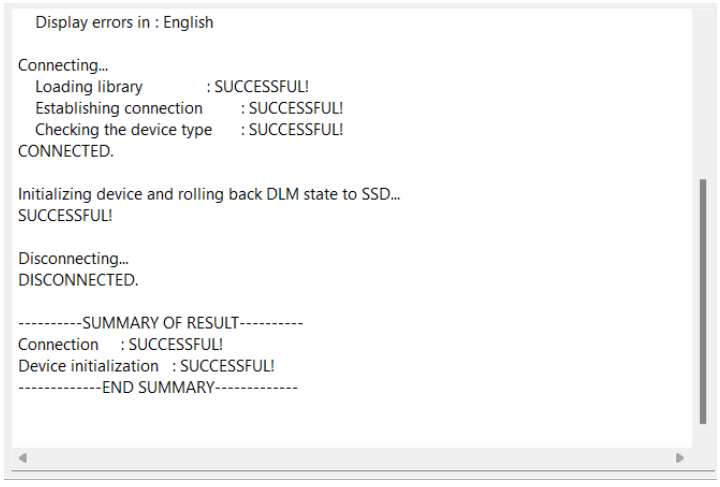

Figure 8. MCU initialization completed successfully

This will remove any TrustZone boundaries, code flash, and data flash contents that reside on the MCU unless the MCU has permanently locked code flash or data flash regions.

3.2 Using the Flat Blinky Project as an Example for SFP

This section has instructions for generating a simple blinky application to evaluate the SFP. In most real-world use cases, application development is not part of the SFP process. Any existing firmware image that targets the MCU internal code flash in .srec or .mot file format can be used in the exercises.

3.2.1 Create a Flat Blinky Project

Launch **e²studio**, click **File > New > Renesas C/C++ Project > Renesas RA,** and select **Renesas RA C/C++ Project**.

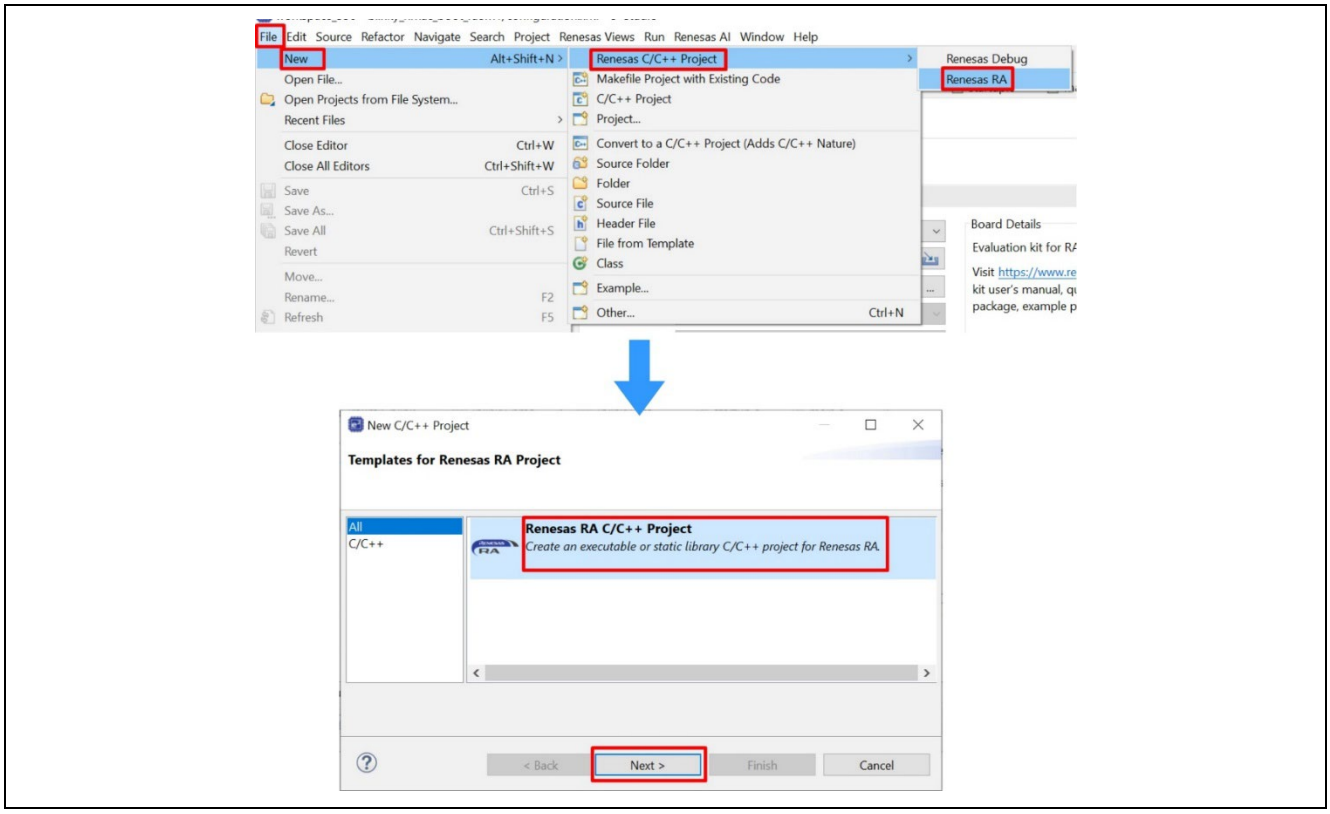

Figure 9. Select Renesas RA C/C++ Project

Assign a name for this new project, then click **Next.** Select a board from the **Device and Tools Selection** and click **Next**.

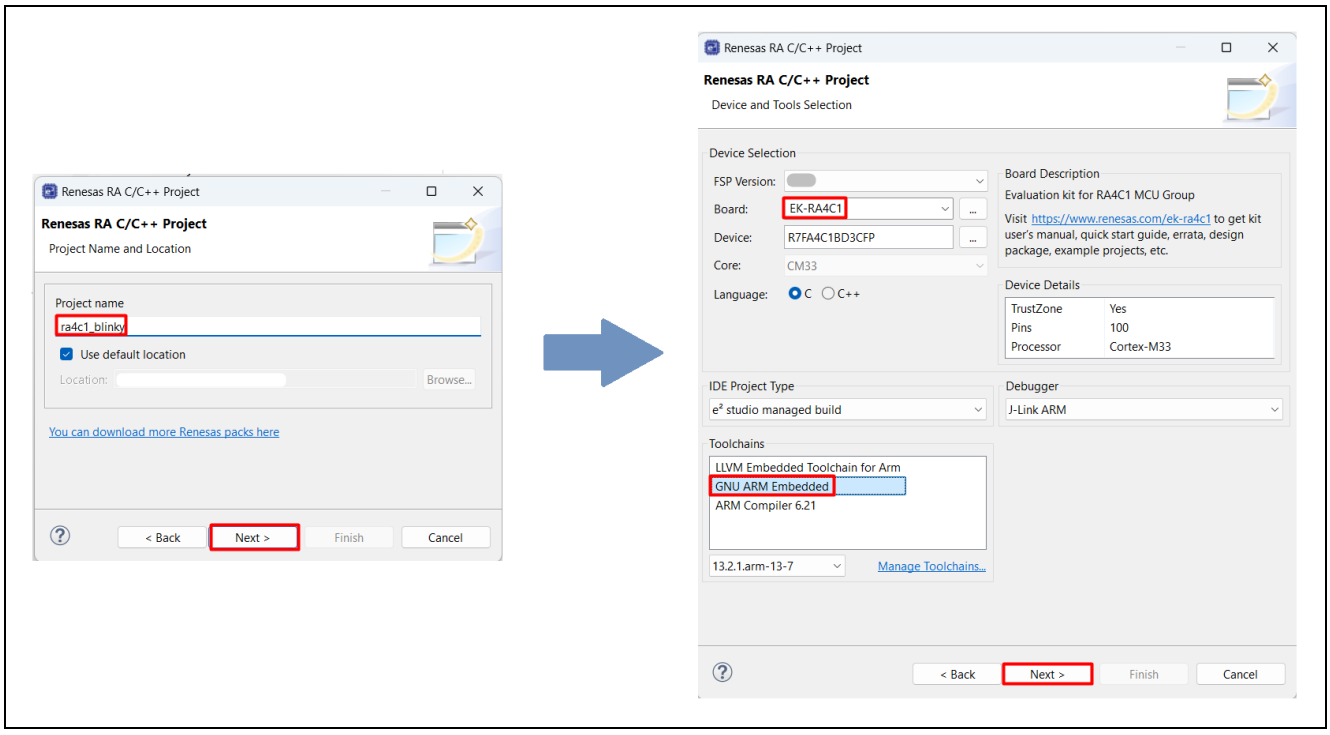

Figure 10. Assign project's name and select a board

Select **Flat (Non-TrustZone) Project** in **Project Type Selection** and click **Next**. Select **Executable** and **No RTOS** for **Build Artifact and RTOS Selection,** and click **Next.** Choose the **Bare Metal - Blinky** template and click **Finish**.

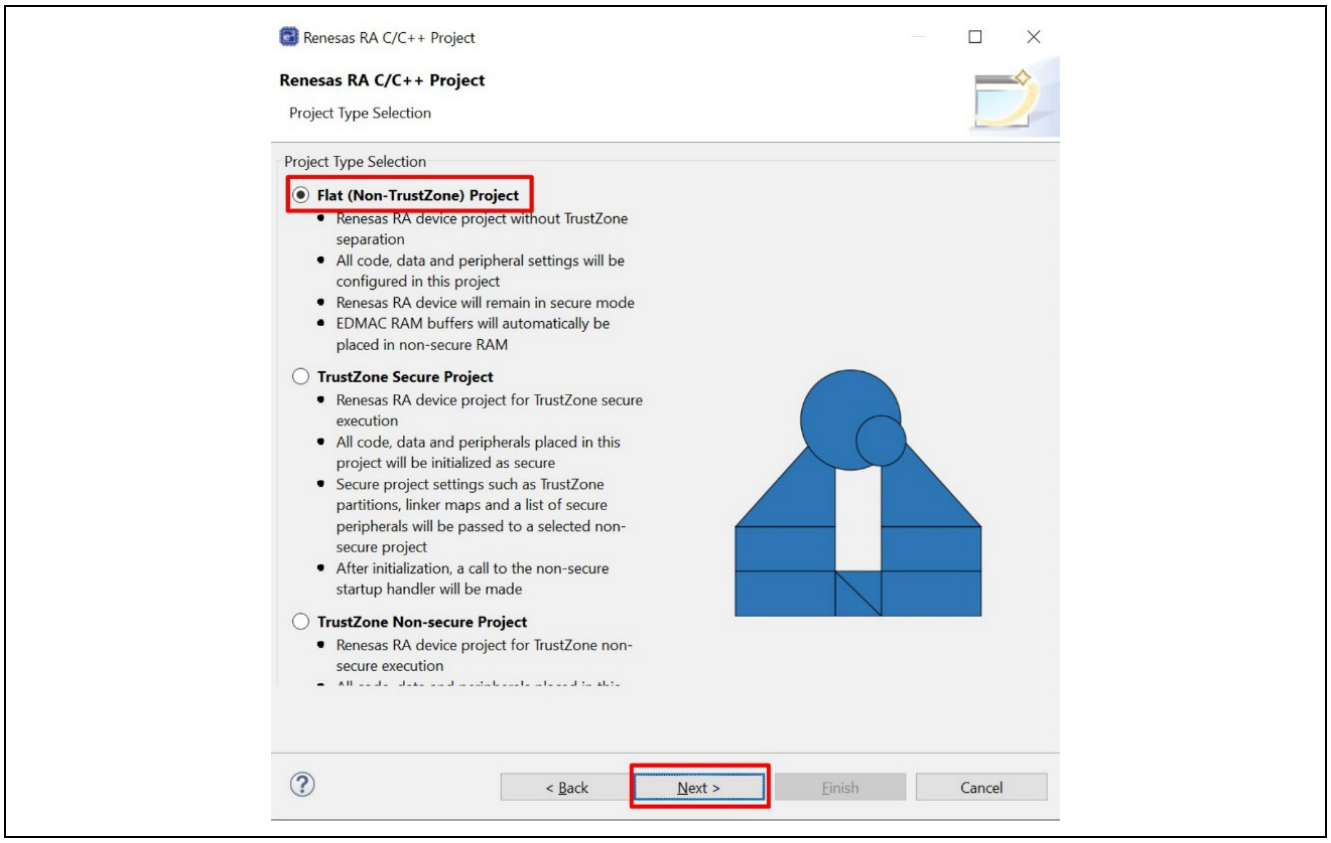

Figure 11. Blinky Project Type Selection

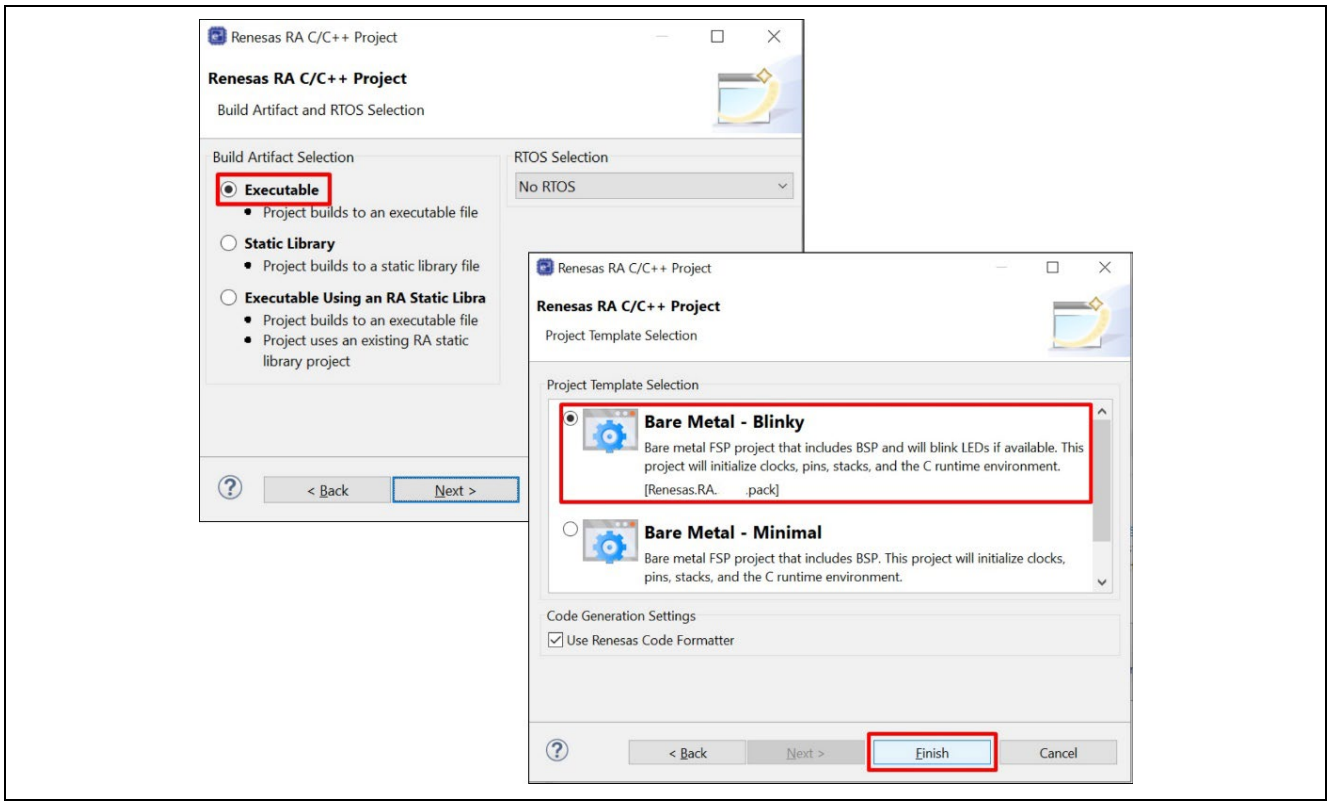

Figure 12. Blinky Project Template Selection

Once the project has been created, the name of the project will show up in the **Project Explorer** window of e²studio. Now, click the Generate Project Content button in the top right corner of the Project Configuration window to generate your board-specific files.

Click the  icon to build the project, and the firmware image (`ra4c1_blinky.srec`) is generated.

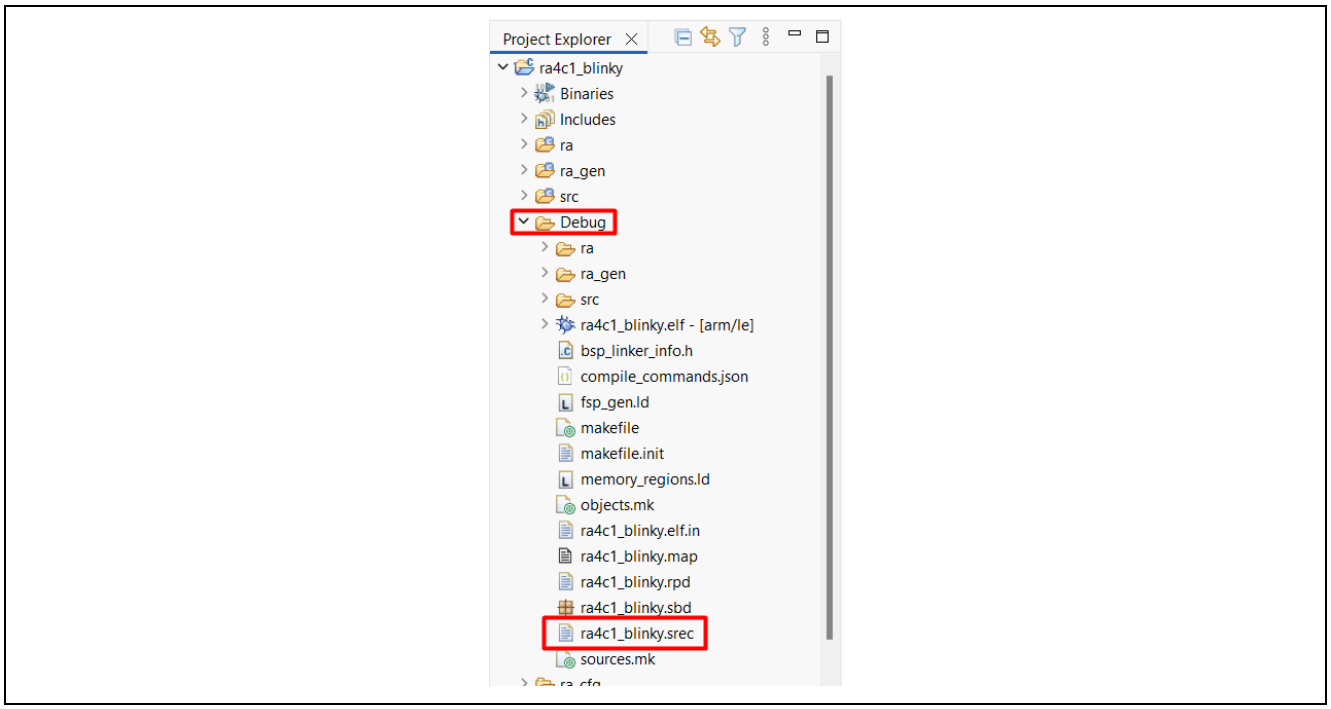

Figure 13. Build the Blinky Project

More information about the configuration of the Blinky Project on e²studio can be found at: https://renesas.github.io/fsp/_s_t_a_r_t__d_e_v.html#tutorial-your-first-ra-mcu-project-blinky

3.2.2 Encrypt the firmware image with SKMT

Launch the **SKMT** GUI and open the **Overview** tab.

Select the security engine corresponding to the target MCU used for Secure Factory Programming (SFP).

In this application note, the **RA4C1** MCU is used as an example. Therefore, **RA Family, RSIP-E31A Security Functions, and Protected Mode** are selected.

For the **RA4L1** MCU, the appropriate selection is **RA Family, RSIP-E11A Security Functions,** and **Protected Mode**.

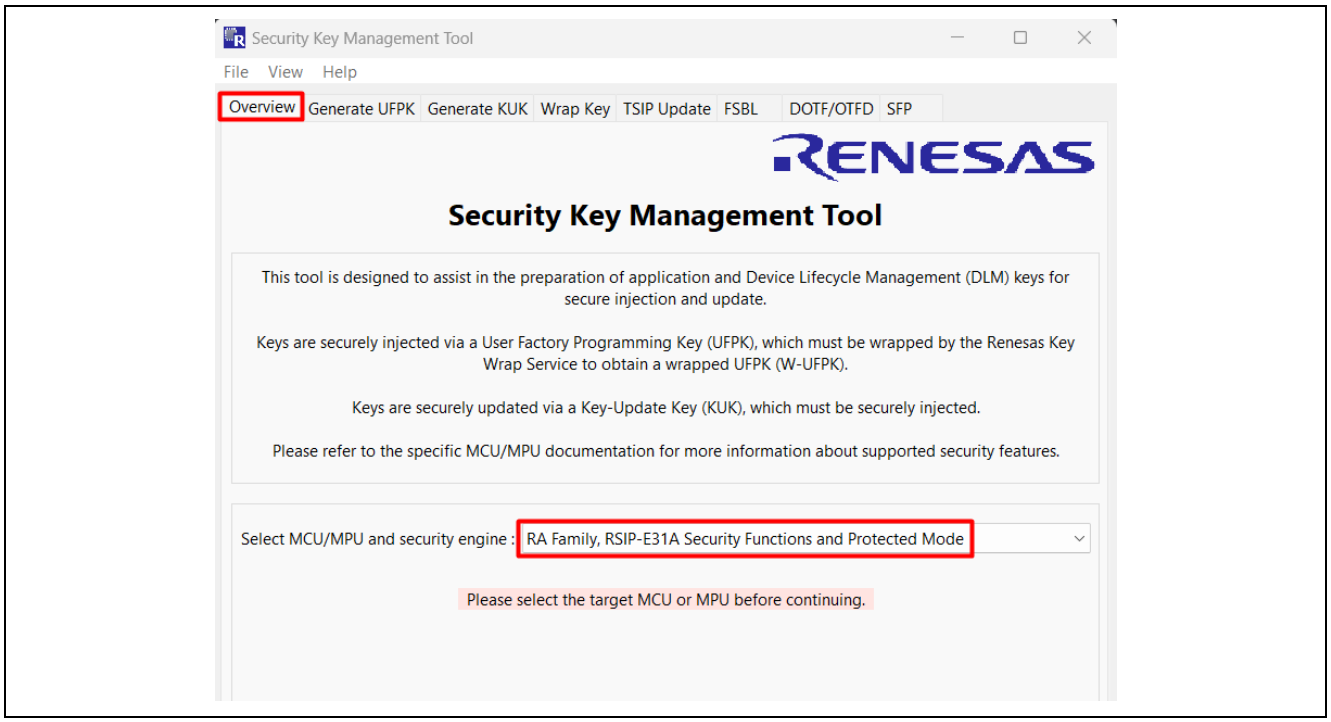

Figure 14. Select RA Family, RSIP Security Functions, and Protected Mode

In the SFP tab, the target MCU is selected from the list that corresponds to the Renesas security engine chosen in the **Overview** tab.

At this stage, the final **DLM** state, as described in Section 1.3, can be configured, and the output directory for the generated SFP file can be specified. Figure 15 shows the configuration of the final DLM state and the output SFP file location.

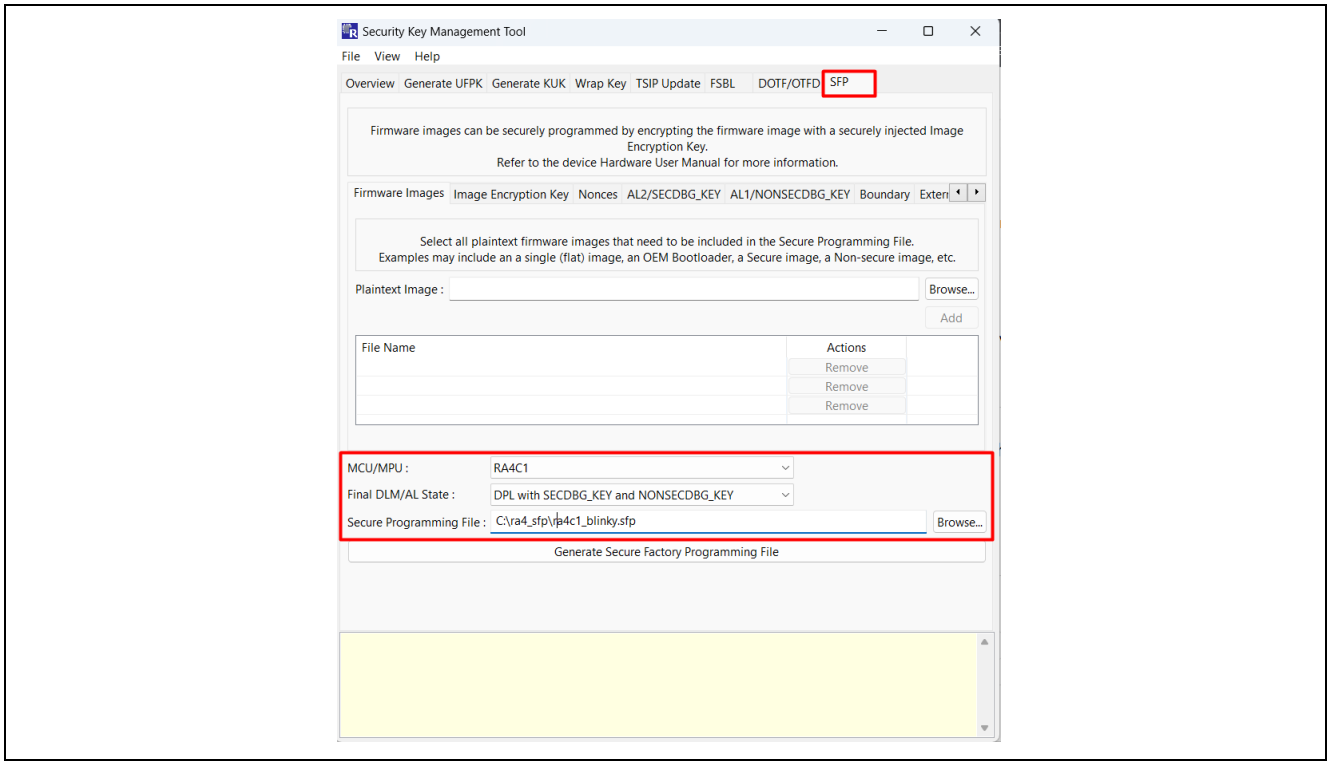

Figure 15. Overview SFP Tab

In the **Firmware Images** tab, **Browse** the **Plaintext Image** (`ra4c1_blinky.srec`) generated in Section 3.2.1 and add it by clicking the **Add** button.

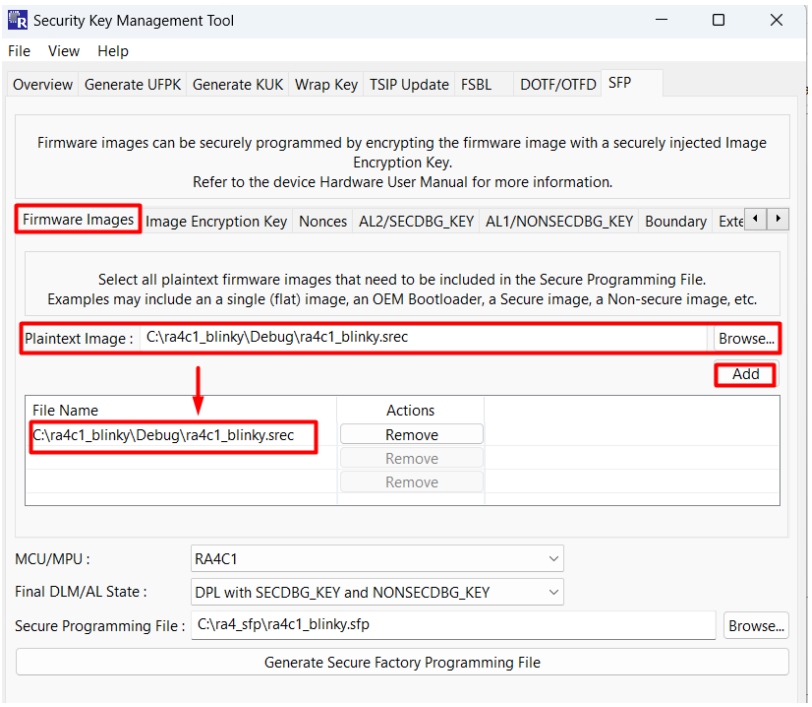

Figure 16. Setting the Firmware Image

On the **Image Encryption Key** tab, input the Image Encryption Key data mentioned in section 2.2.

The key data is duplicated here to easily copy and paste to the GUI interface:

KEY = 80000000000000000000000000000000

For simplicity, **IV** is set to **Generate random value** in this example.

Click the **Browse** buttons to select the **UFPK** and **W-UFPK** key pair generated in section 2.1.

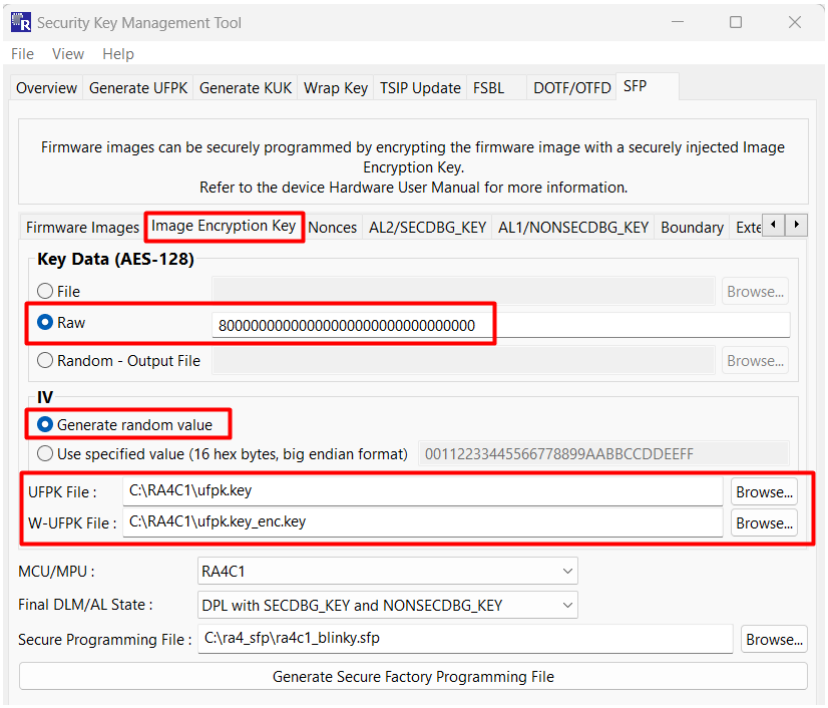

Figure 17. Setting the Image Encryption Key

For simplicity, the IV for programming parameters and firmware image is set to **Generate random value** in **Nonces** tab.

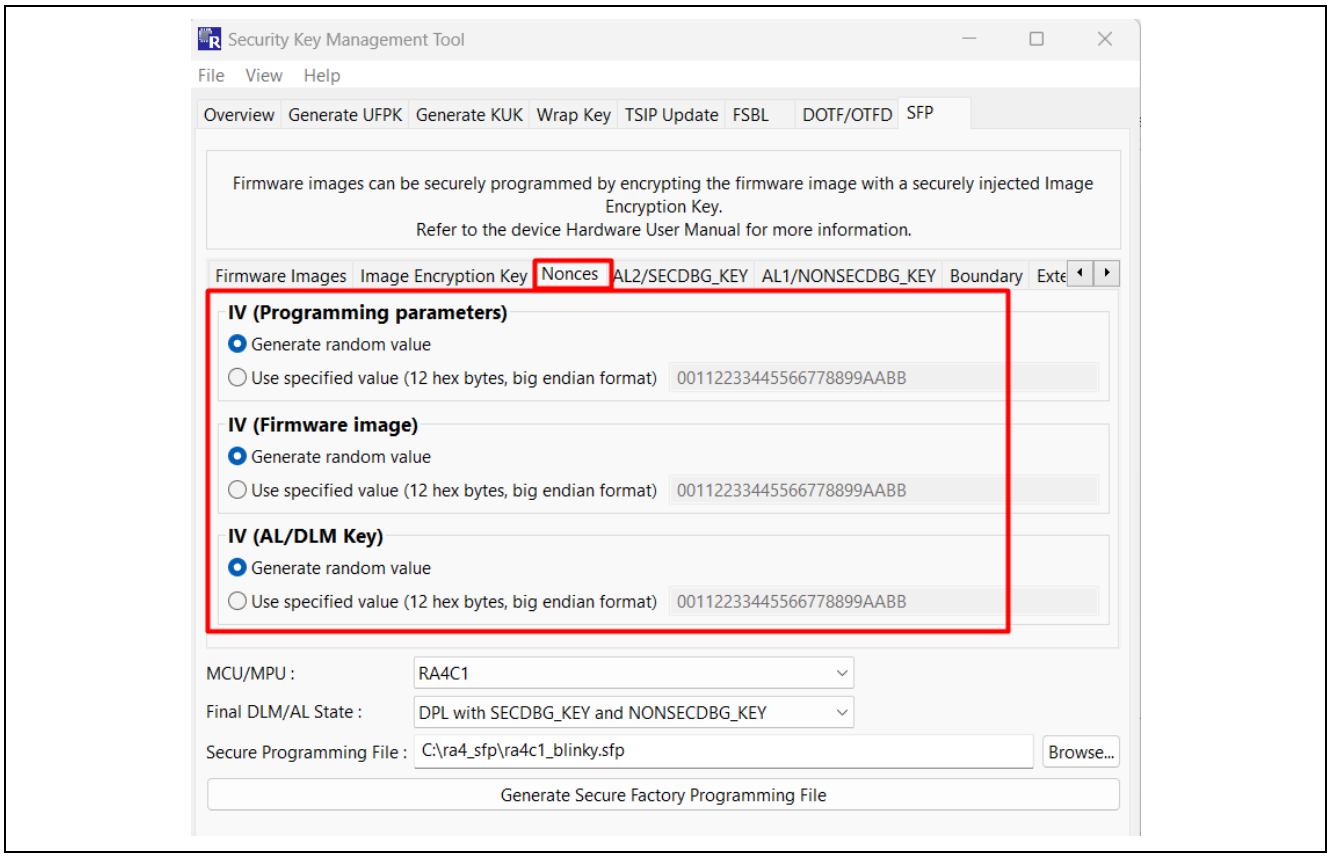

Figure 18. Setting the Nonces

In **AL2/SECDBG_KEY** and **AL1/NONSECDBG_KEY** tab, specify the AES 128-bit key data to be used.

The raw key in Section 2.3 will be used:

AL2/SECDBG_KEY	000102030405060708090A0B0C0D0E0F
AL1/NONSECDBG_KEY	9f3c7a12b4e86d5a2c9180f7e34b6d1a

Choose **Generate random value** for the **IV**. The DLM keys **(SECDBG_KEY/NONSECDBG_KEY)** are added shown in Figure 19.

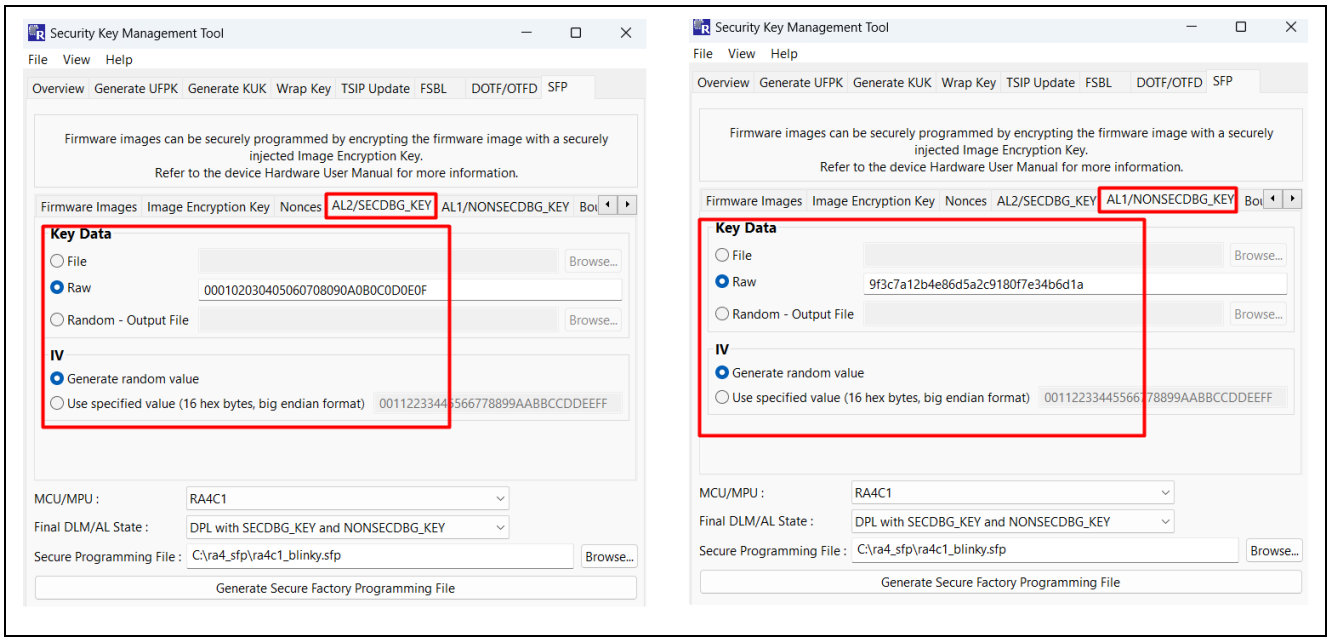

Figure 19. DLM key configuration

In addition to configuring cryptographic keys and the DLM state, the single boot firmware command for Secure Factory Programming also supports configuration of memory security attributes. This configuration is performed in the **Boundary** tab.

Users can either apply an automatically generated `.rpd` file or manually configure the security attributes for each memory region. When using manual configuration, the security attributes must match the actual memory usage of the target application. Incorrect configuration may prevent the application from running correctly or result in unpredictable behavior. In this demonstration, an automatically generated .rpd file is used as an example for boundary configuration.

Open the **Boundary** tab and specify the `.rpd` file located at `ra4c1_blinky/debug/ra4c1_blinky.rpd`, as shown in Figure 20.

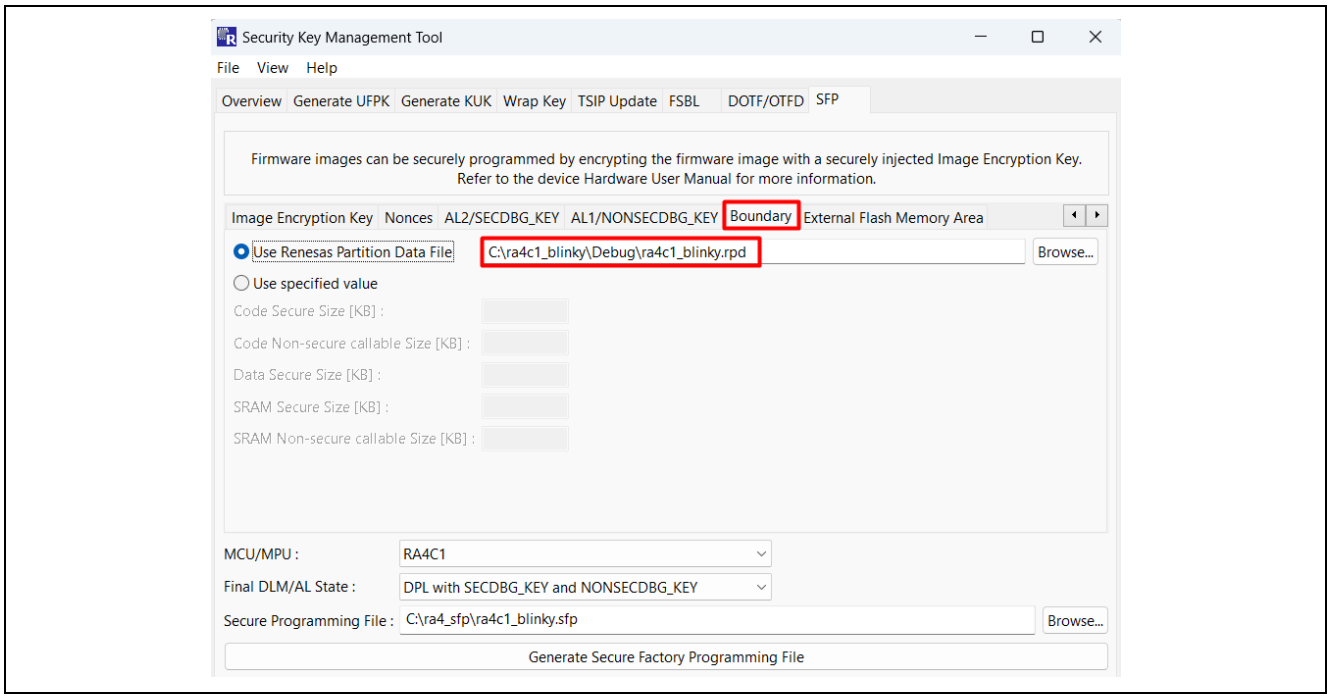

Figure 20. Security attributes configuration

After setting up the cryptographic keys, the DLM state, and the security attributes for each memory region, click the **Generate Secure Factory Programming File** button to create the output `.sfp` file. If the file is successfully generated, the “OPERATION SUCCESSFUL” will appear, and the output file `ra4c1_blinky.sfp` will be created.

Note that during generation of the `.sfp` file, the **UFPK, W-UFPK, Image Encryption Key, DLM keys (SECDBG_KEY and NONSECDBG_KEY)**, and the application image are used as input data. In this process, the DLM keys and the Image Encryption Key are wrapped using the UFPK, and the application image is encrypted using the Image Encryption Key. The generated `.sfp` file contains the encrypted application image, the wrapped DLM keys, the final DLM state, and the security attributes for each memory region. When the `.sfp` file is programmed into the MCU via the serial programming interface, the DLM keys are stored after being re-wrapped with the MCU Hardware Unique Key (HUK). The UFPK and the Image Encryption Key are discarded and are not retained in the device.

3.2.3 Program the Encrypted Image with RFP

Before programming the encrypted firmware into the MCU, ensure that all hardware configurations described in Section 3.1 are completed, and that the device has been properly initialized**.**

Launch **RFP** and click **File > New Project**. Assign the name of the project, select the tool and interface for communication, then click **Connect**.

The project will open, and the device information can be read through **Target Device > Read Device Information**.

After the Initialize command is successfully executed, the DLM state of the MCU is **SSD**.

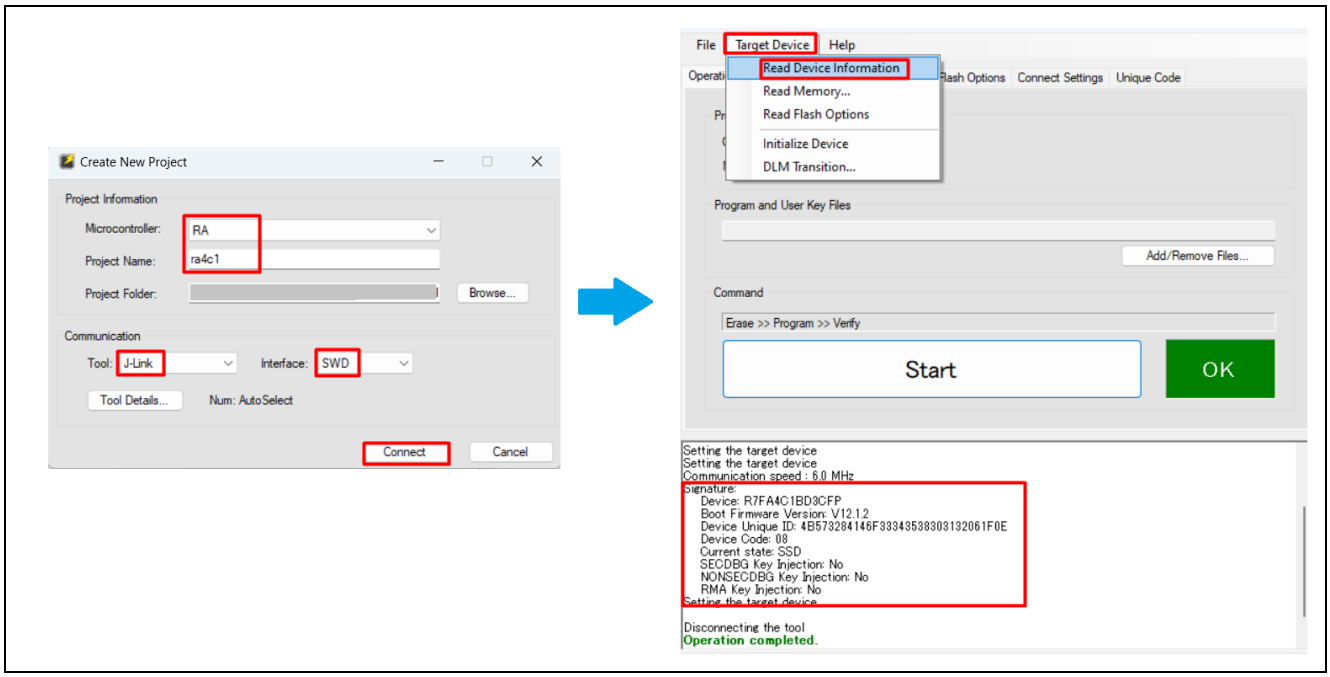

Figure 21. Create a new RFP project

Add the `ra4c1_blinky.sfp` file created in 3.2.2, with **Operation Settings** shown in Figure 22 then click **Start**. The encrypted firmware image will be programmed, and LEDs (red, green, blue) will blink on the board.

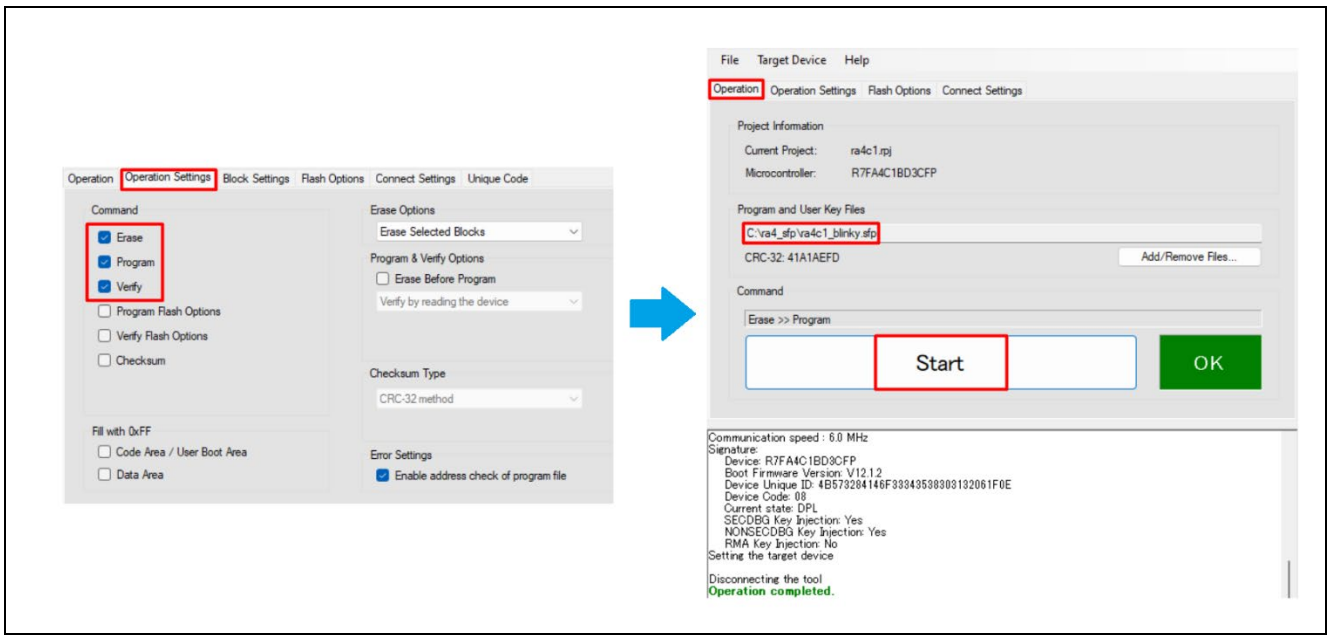

Figure 22. Operation Settings and Programming SFP file

After the `.sfp` file is programmed, the device enters the **DPL** state with both **SECDBG_KEY** and **NONSECDBG_KEY** injected. In this state, debugging is locked and serial programming cannot access the code and data flash areas.

To re-enable the debug interface, the DLM state must first be transitioned from **DPL** to **NSECSD** using the plaintext **NONSECDBG_KEY**, and then from **NSECSD** to **SSD** using the **SECDBG_KEY**, without erasing the programmed firmware.

A direct transition from the DPL state to the SSD state is not possible, except by using the **Initialize Device** command. However, this will erase the programmed firmware image.

Note: The **Disable Initialize Command** is not recommended during the development phase. Once this command is executed, a direct transition from the DPL state to the SSD state is no longer possible, and the **Initialize Command** cannot be performed again permanently.

3.3 Using SFP with TrustZone Projects

Secure Factory Programming (SFP) supports TrustZone-based projects. Detailed information on TrustZone-based application development is provided in R11AN0467 Security Design with Arm® TrustZone® using Cortex-M33. Application development itself is outside the scope of the SFP process.

Two TrustZone development models are commonly used:

- Combined Project Development:
  - The Secure and Non-secure applications are developed by a single trusted team.
- Split Project Development:
  - The Secure and Non-secure applications are developed by two separate teams.
  - The Non-secure application team does not have direct access to assets in the Secure partition. Access to Secure resources is permitted only through Non-secure Callable (NSC) APIs.

This application note demonstrates only the **Combined Project Development** model by creating a new TrustZone-based blinky solution as an example, as described in Section 3.3.1

For the **Split Project Development model**, the encrypted `.sfp` file is generated in the **SSD state** using the cryptographic keys (UFPK, W-UFPK, Image Encryption Key, and DLM keys), the secure firmware image, and a dummy non-secure firmware image. The **NONSECDBG_KEY** is then delivered to the non-secure development team for programming the actual non-secure firmware into the non-secure memory region.

3.3.1 Create Blinky Solution with a TrustZone Project chain

Launch **e²studio**, click **File > New > Renesas C/C++ Project > Renesas RA,** and select **Renesas RA FSP Solution**, click **Next**

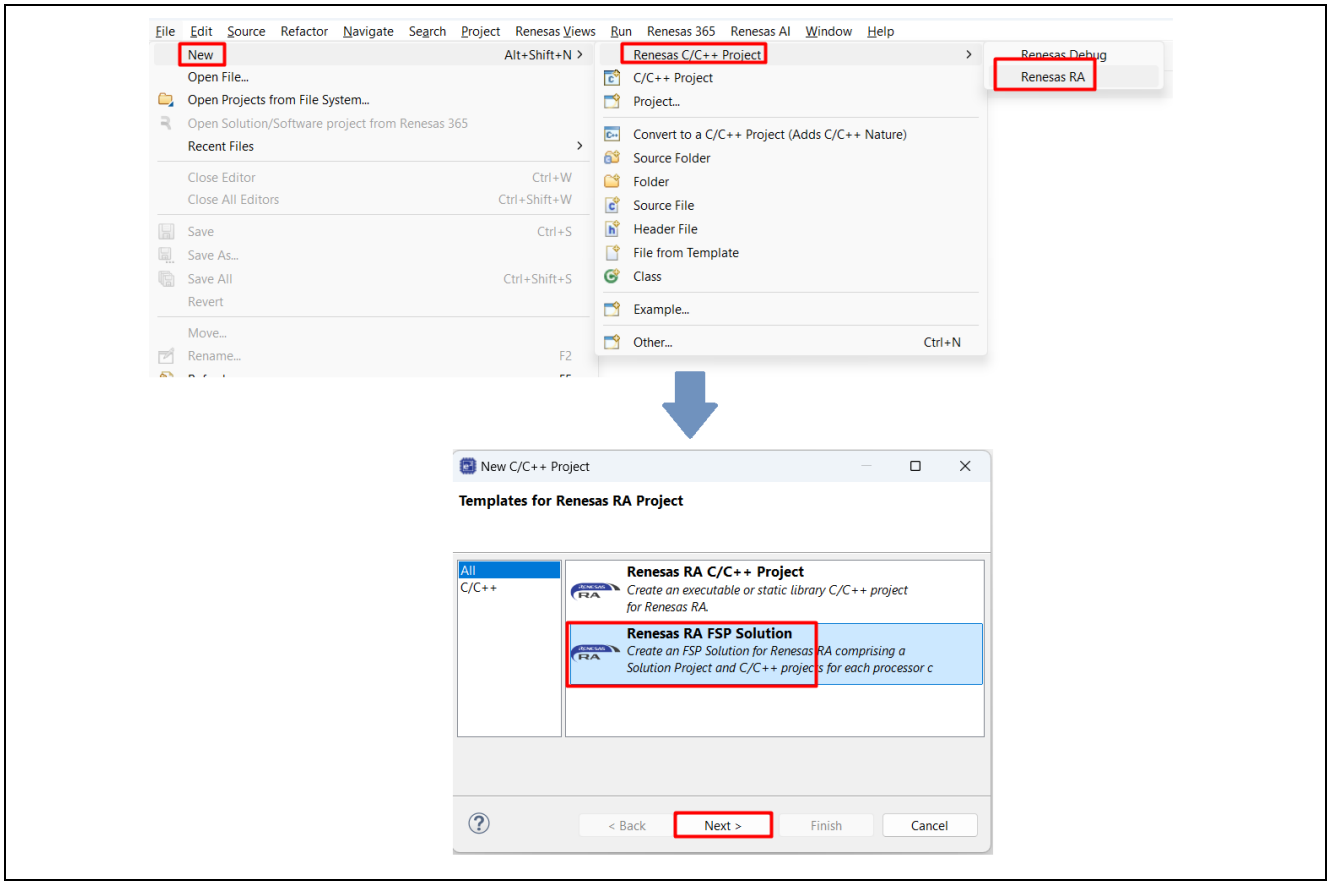

Figure 23. Operation Settings for Programming SFP file

Assign a name for this new project, then click **Next.**

In the **Device and Tools Selection** window, select the target board, choose **Single core > TrustZone > Bare Metal Blinky** as the solution template, select the toolchain from the **Toolchains** list, and click **Finish**.

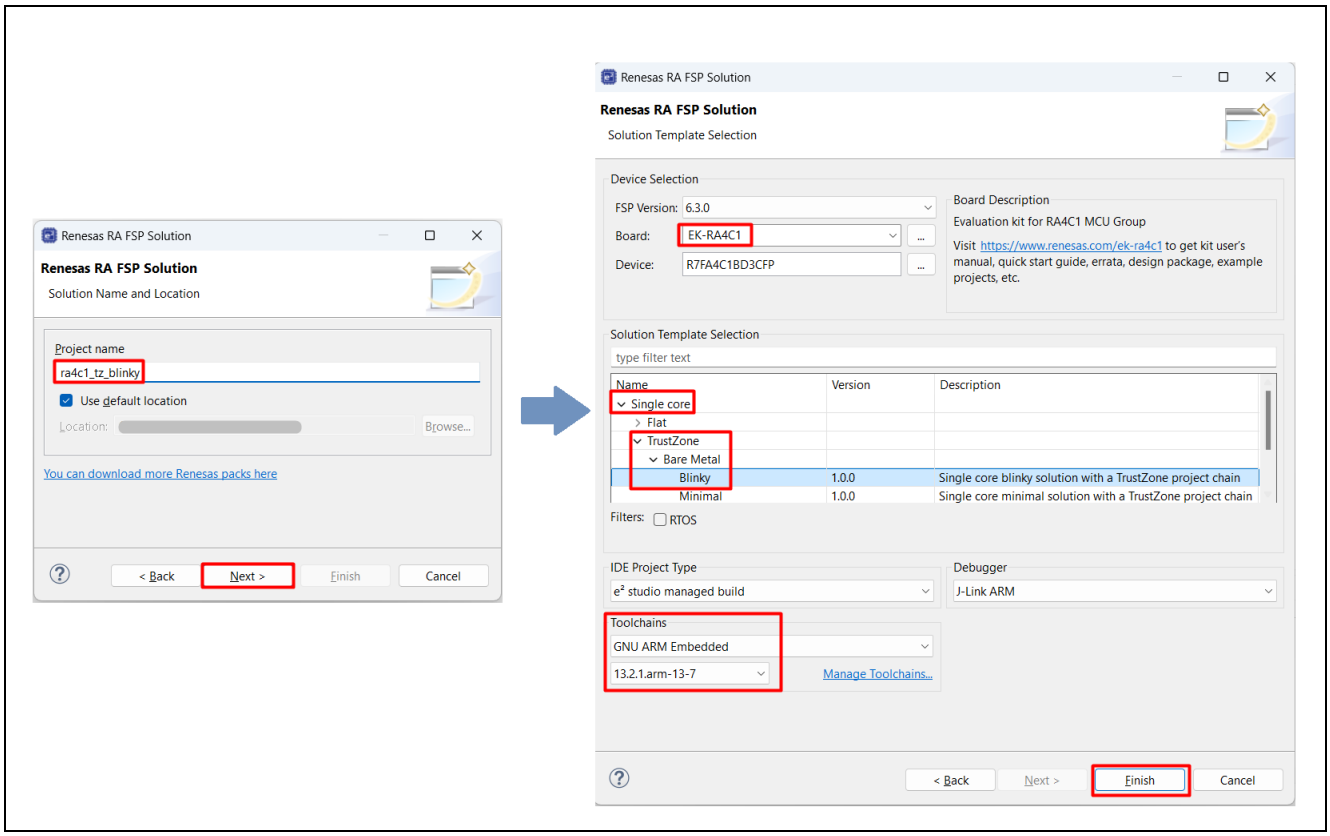

Figure 24. Project Name Assignment and TrustZone Template Selection

Verify that a TrustZone chain project is created and automatically built. The build process generates the following output files:

- Non-Secure project ra4c1_tz_blinky_nonsecure: `/debug/ra4c1_tz_blinky_nonsecure.srec`
- Secure Project ra4c1_tz_blinky_secure: `/debug/ra4c1_tz_blinky_secure.srec`

3.3.2 Encrypt the Secure and Non-Secure Firmware Images

All cryptographic keys created in Section 2 are used in this section to encrypt both the Secure and Non-secure firmware images. Table 1 summarizes the key information created in Section 2.

Table 1. Cryptographic Keys used for Firmware Image Encryption

Cryptographic Keys	Key Data	Description
UFPK	`ra4c1_ufpk.key`	User Factory Programming Key
W-UFPK	`ra4c1_ufpk.key_enc.key`	Wrapped-User Factory Programming Key
Image Encryption Key	`80000000000000000000000000000000`	Key for Image Encryption
SECDBG_KEY	`000102030405060708090A0B0C0D0E0F`	Lifecycle authentication key for transitioning from the NSECSD state to the SSD state.
NONSECDBG_KEY	`9f3c7a12b4e86d5a2c9180f7e34b6d1a`	Lifecycle authentication key for transitioning from the DPL state to the NSECSD state.

Note: The cryptographic keys used in this demonstration are for example purposes only. In a real development environment, all cryptographic keys must be newly generated in accordance with the procedure described in Section 2.

The image encryption procedure using the SKMT tool follows the same process described in Section 3.2.2. The only difference is that, in the Firmware Images tab, the Non-secure and Secure firmware image must be added, as shown in Figure 25.

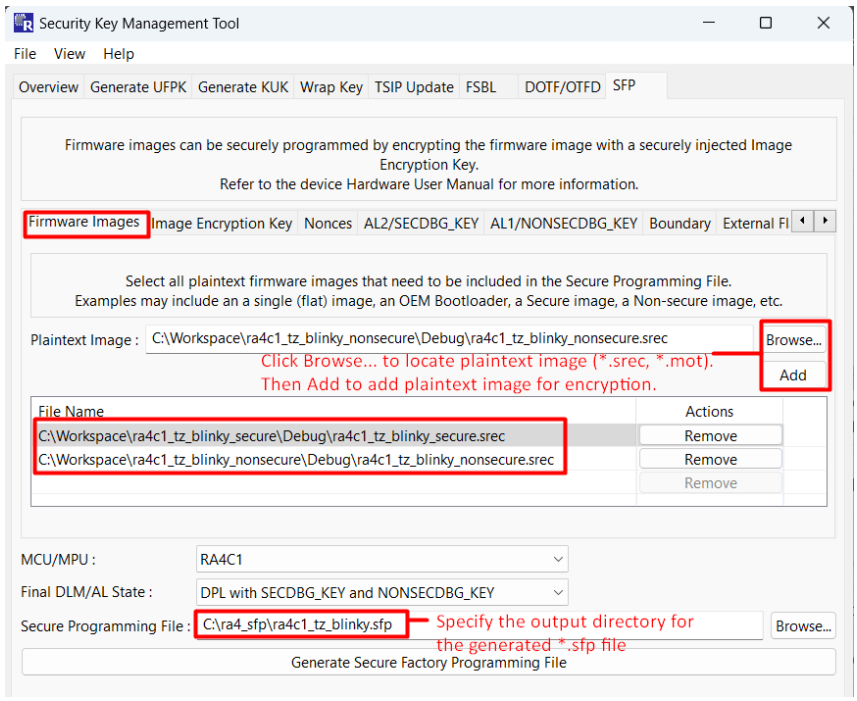

Figure 25. Select the Secure and Non-Secure Application to Generate the SFP File

To configure the security attributes for each memory region, specify the .rpd file automatically generated by the Secure project, such as `ra4c1_tz_blinky_secure/Debug/ra4c1_tz_blinky_secure.rpd`, as shown in Figure 26.

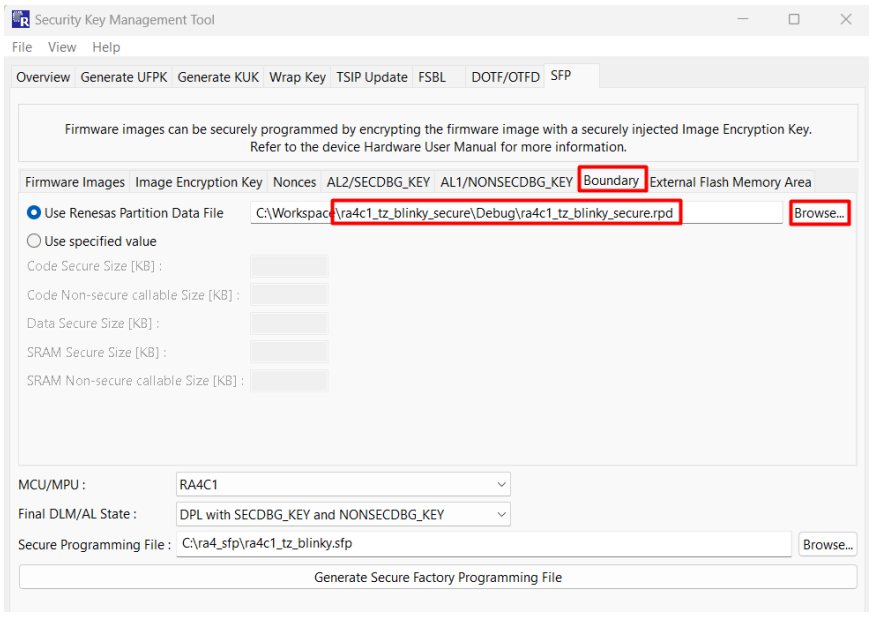

Figure 26. Using the .rpd file to Configure Security Attributes

After all cryptographic keys, firmware images, and memory security attribute settings have been specified, click **Generate Secure Factory Programming File**. As a result, the encrypted ".sfp" output file is generated. In this example, the output file is named `ra4c1_tz_blinky.sfp`.

3.3.3 Program the TrustZone Encrypted Image with RFP

The output file `ra4c1_tz_blinky.sfp` generated in Section 3.3.2 is used to program the MCU using the RFP tool.

Follow Section 3.2.2 to set up the hardware and create an RFP project. Then verify the **Operation Settings** and program the `ra4c1_tz_blinky.sfp` file, as shown in Figure 27.

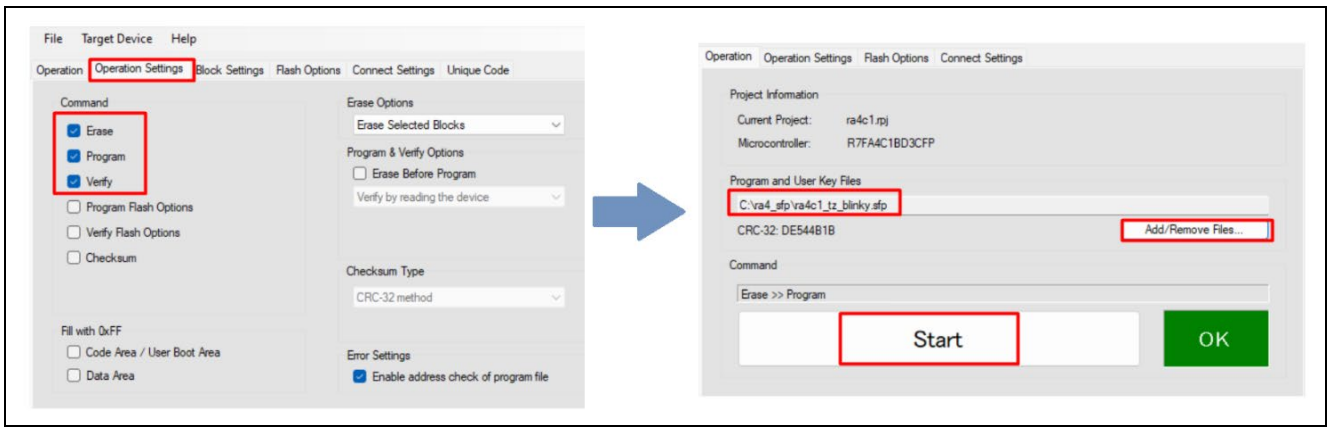

Figure 27. Operation Settings and Programming SFP file

A successful status is shown in the log window with the message "Operation completed.", and the three LEDs on the board start blinking.

To verify the final DLM state, use the serial programming command to read back the device information through the RFP tool. Select **Target Device > Read Device Information**.

The displayed information indicates that the final DLM state is DPL with both the Secure and Non-secure keys injected, as shown in Figure 28.

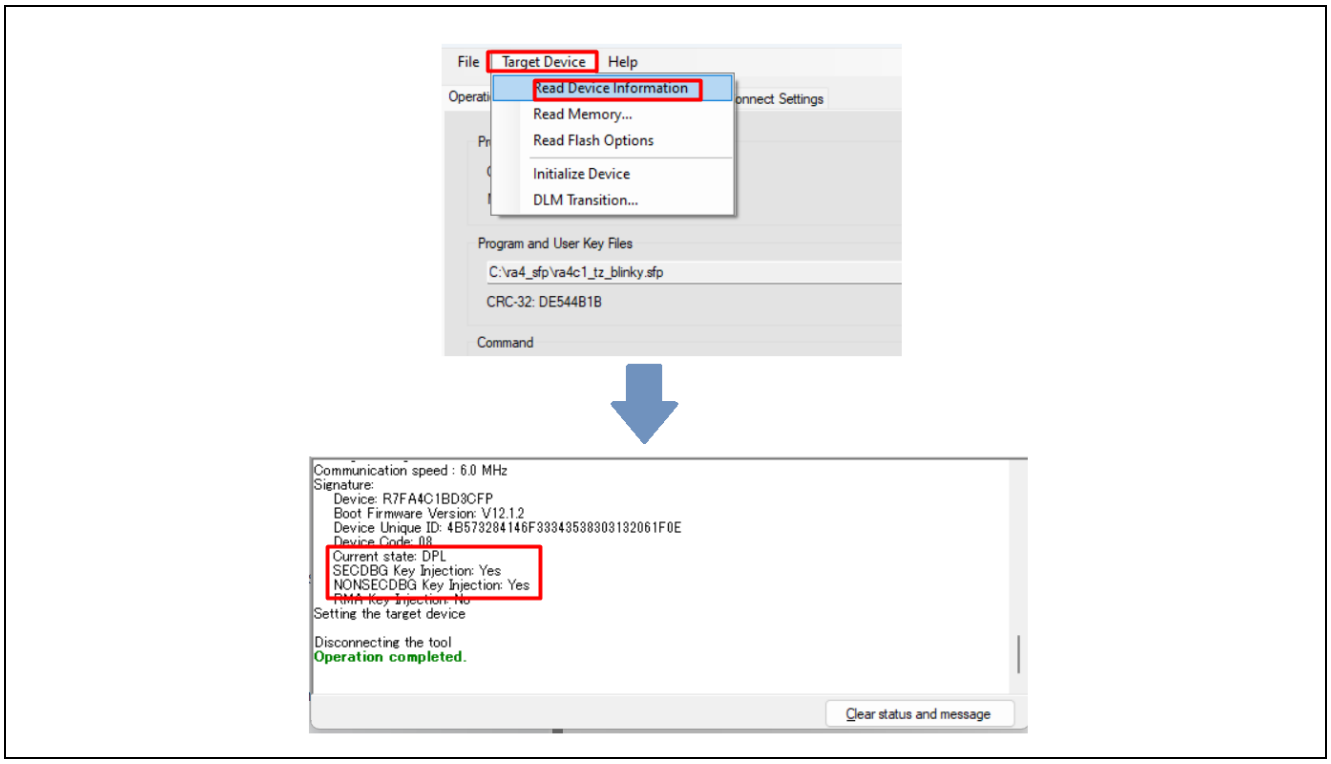

Figure 28. Verify the Final DLM state

As described previously, in this state, the debugger connection is not permitted, and serial programming cannot access the code and data flash areas.

The **NONSECDBG_KEY** must be provided to transition the device to the **NSECSD** state. In this state, debugging is allowed only for non-secure memory regions and peripherals, and serial programming can program, erase, and read only the non-secure code and data flash areas.

Full access to both debugging and the code and data flash areas is enabled by providing the **SECDBG_KEY**, which transitions the device from the **NSECSD** state to the **SSD** state.

To change the lifecycle of the device to the desired DLM state, select **Target Device > DLM Transition**, and then specify the target DLM state with the corresponding authentication key (**SECDBG_KEY or NONSECDBG_KEY**), as shown in Figure 29. For detailed information, refer to *R11AN0469 Device Lifecycle Management for Cortex-M33*.

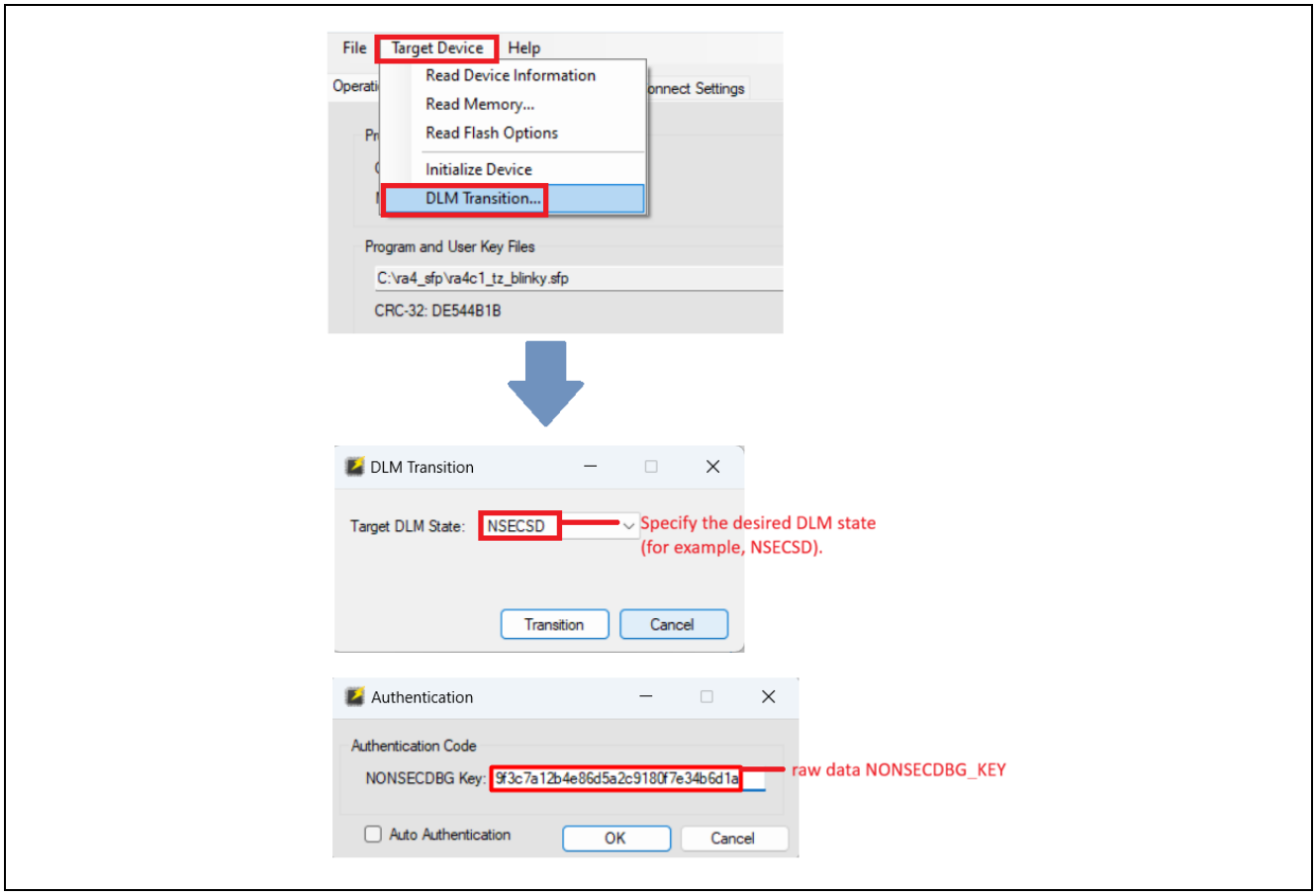

Figure 29. Transitioning the device lifecycle state using RFP

4. Appendix A - Disable Initialize command Using RFP.

The Initialize Command can be disabled to prevent flash erasure of devices that have been integrated into a final product.

Note that once the Initialize Command is disabled, it cannot be executed again. Therefore, disabling this command is not recommended during the development phase.

The following steps describe how to disable the Initialize Command using the RFP tool.

Step 1: Create a new RFP project and establish communication with the RA evaluation board (EK-RA4C1 is used in this example).

Open RFP and Select:

[**File**] > [**New Project**]. Enter the required **Project Information** and select the **Communication** interface corresponding to the boot mode used by the device (SCI or SWD boot mode).

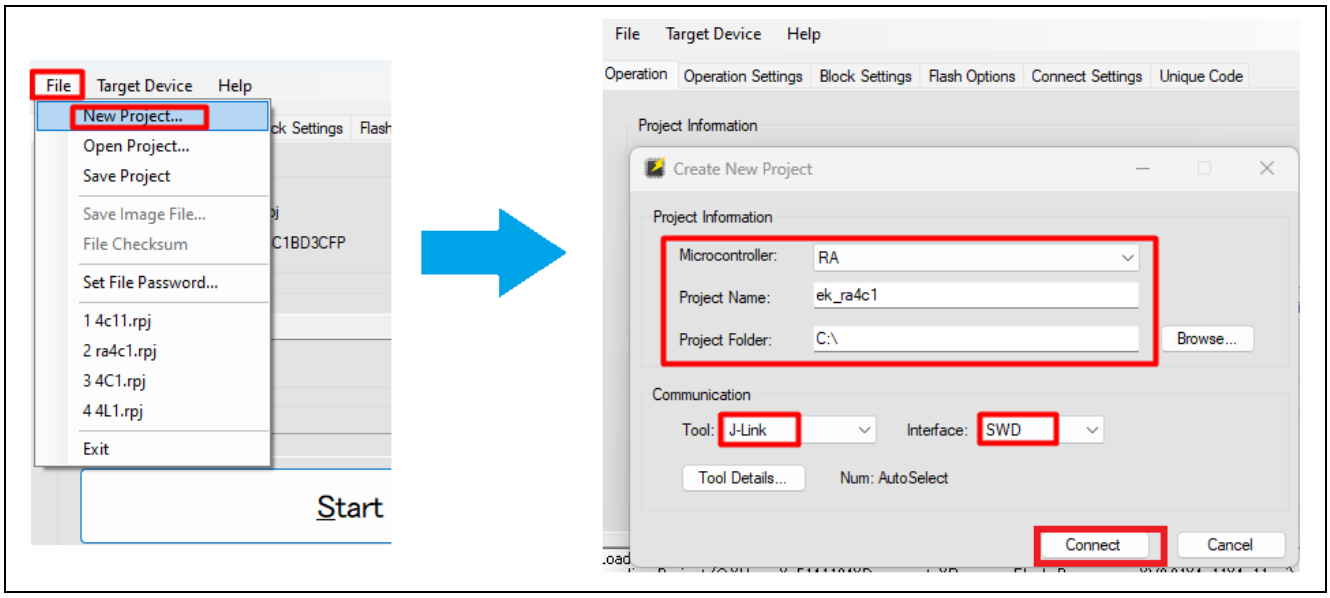

Figure 30. Create an RFP Project and establish the connection with EK-RA4C1

Step 2: In the **Flash Options** tab, under Security settings, **set the option** to **Set** and **Disable Initialize Command** to **Yes,** as shown in the Figure 31.

A warning dialog appears, as shown in the Figure 32, indicating that once this setting is applied, the Initialize Command will be permanently disabled. Again, this setting is not recommended in the development phase.

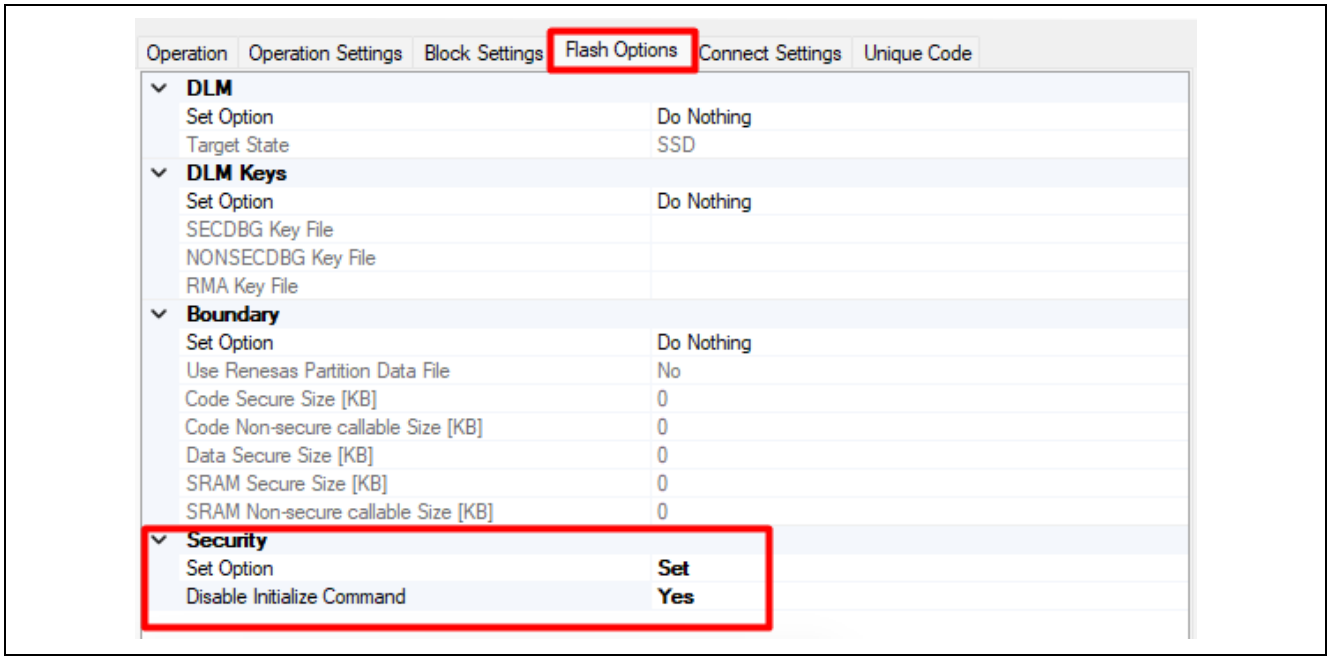

Figure 31. Setting for Flash Options tab to Disable Initialize Command

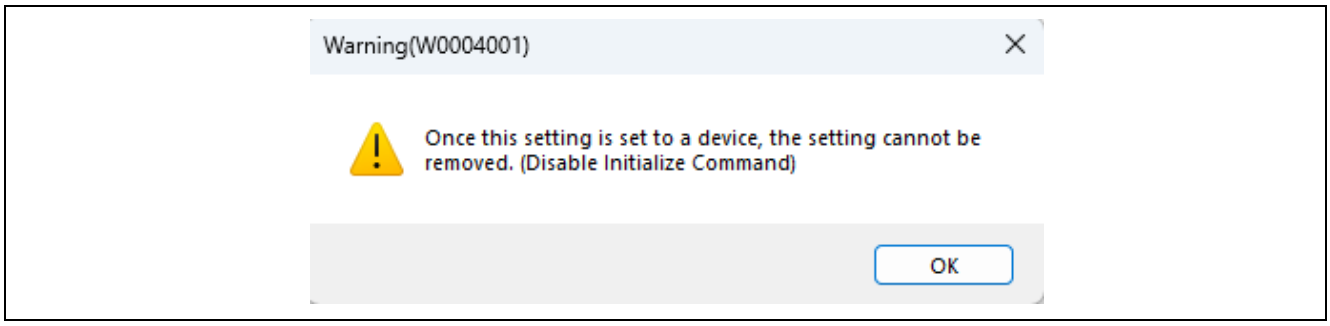

Figure 32. Warning Dialog Message if Disable Initialize Command Set.

Step 3: Click **OK** to confirm the warning message. Then return to the **Operation Settings** tab and select **Program Flash Options**.

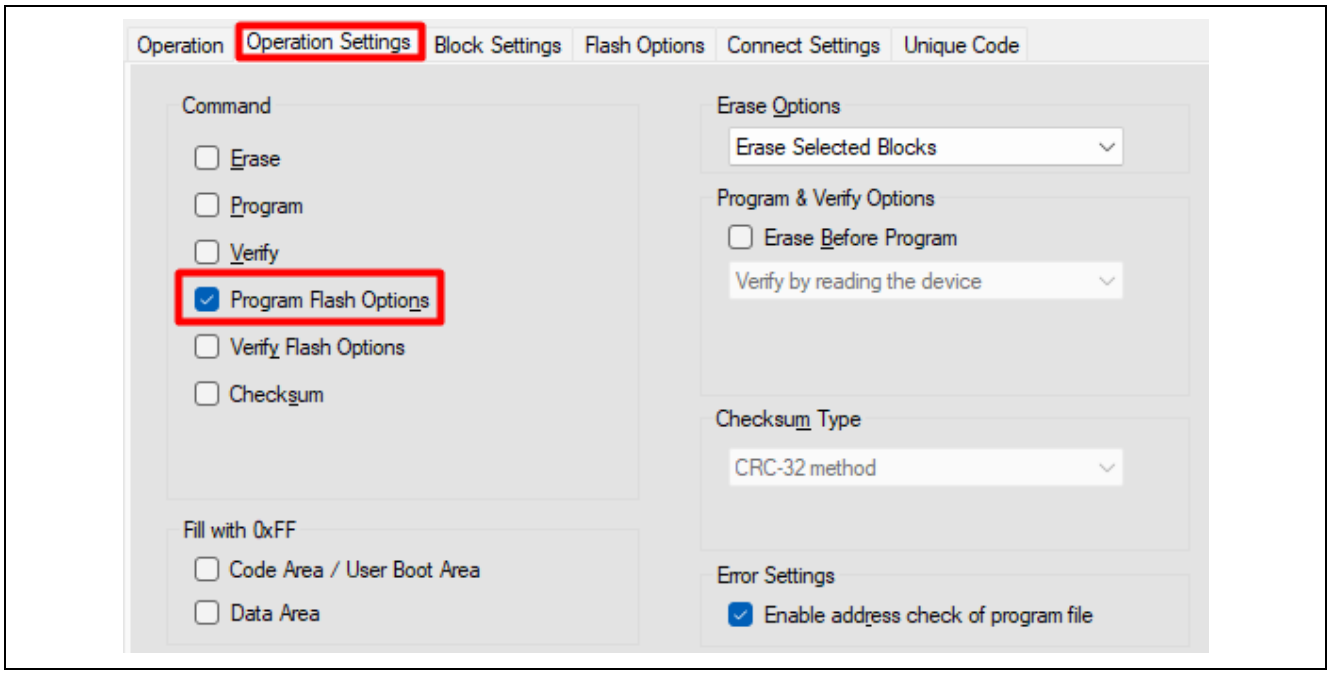

Figure 33. Select Program Flash Options from Operation Settings

Step 4: Click **Start** in the **Operation** tab to perform Program Flash Options.

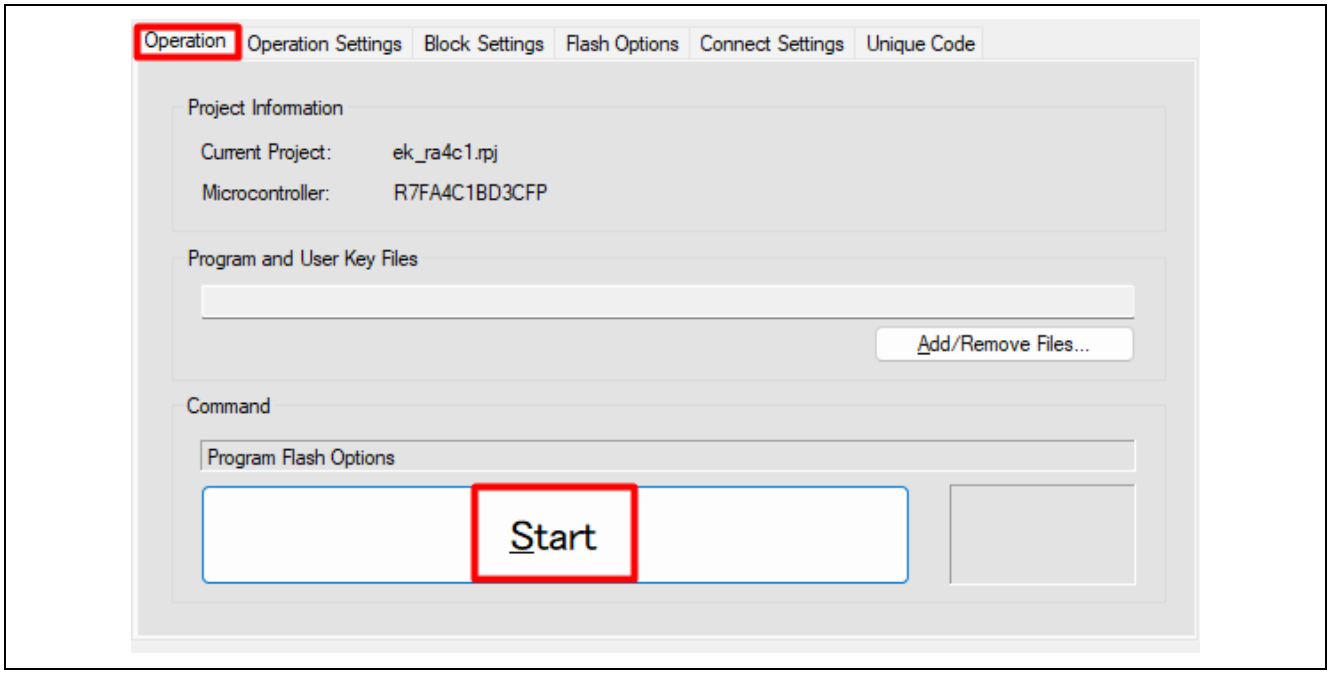

Figure 34. Start Disabling the Initialize Command.

Confirm the successful status message and verify that **Option Information** indicates **Security** and **Protection Information** is **Disable Chip Erase**.

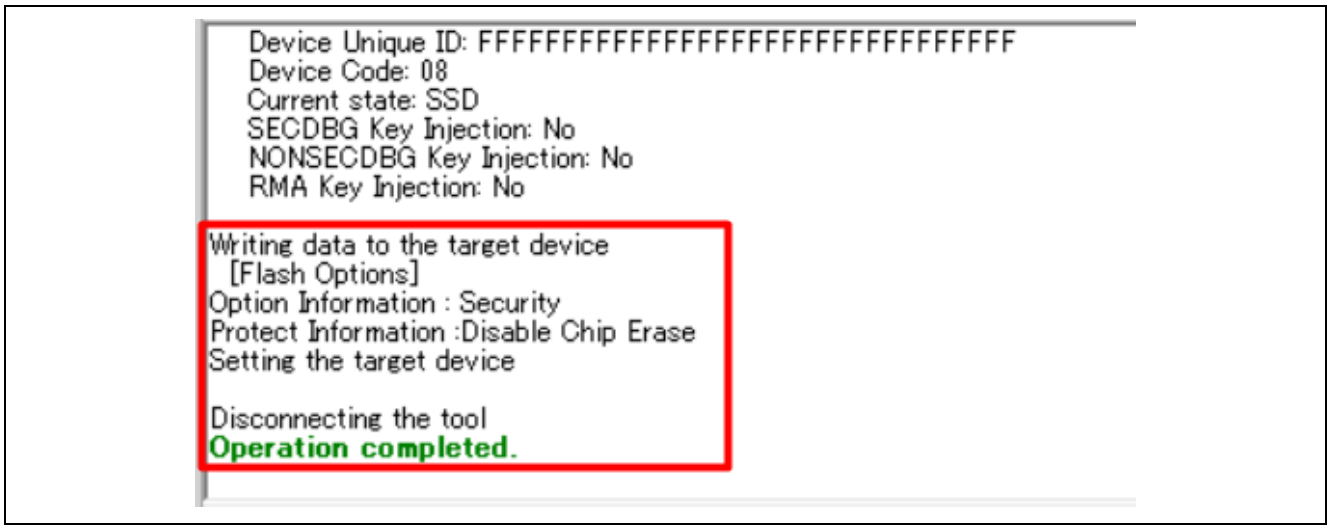

Figure 35. Disable Initialize Command Success Log

After the Initialize Command is disabled, the command can no longer be issued. Any attempt to execute the Initialize Command will result in the message shown in Figure 36.

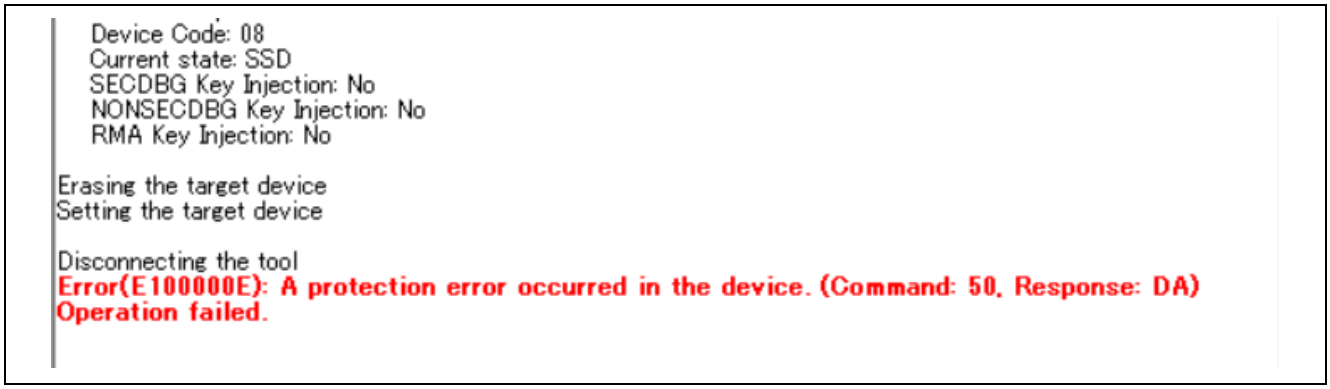

Figure 36. Initialize Command Error After Disable

5. References

1. Flexible Software Package (FSP) User's Manual
2. Renesas RA4C1 Group User's Manual: Hardware (R01UH1137)
3. Renesas RA4L1 Group User's Manual: Hardware (R01UH1081)
4. Renesas RA Family MCU Device Lifecycle Management for Cortex-M33 (R11AN0469)
5. Renesas RA Family MCU Injecting and Updating Secure User Keys (R11AN0496)
6. Renesas RA Family MCU Security Design with Arm® TrustZone® using Cortex-M33 (R11AN0467)
7. Renesas Boot Firmware for RA4C1 MCU Group (R01AN7910)

6. Website and Support

Visit the following URLs to learn about the RA family of microcontrollers, download tools and documentation, and get support.

EK-RA4C1 Resources	renesas.com/ra/EK-RA4C1
RA Product Information	renesas.com/ra
Flexible Software Package (FSP)	renesas.com/ra/fsp
RA Product Support Forum	renesas.com/ra/forum
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.31.26	—	Initial release

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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

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/.