Renesas RA Family

RA8 Secure Factory Programming

Introduction

The Secure Factory Programming (SFP) feature of the Renesas RA8 MCU Series 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 RA8 MCUs. Guidelines and procedures for utilizing these features and tools are provided. The key operations during SFP are Firmware Image Encryption and Device Lifecycle Management (DLM), which are discussed in detail in this application note. Additionally, SFP is evaluated alongside other security solutions, such as the First Stage Bootloader (FSBL), TrustZone, and Decryption On The Fly (DOTF). This application note provides usage notes on how to use SFP with these solutions.

EK-RA8M1 is used to evaluate the SFP solution in this application note. The procedure steps and screenshots used EK-RA8M1 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) v3.16
  https://www.renesas.com/us/en/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html
- Renesas Security Key Management Tool v1.07
  https://www.renesas.com/software-tool/security-key-management-tool
- Gpg4win
  http://www.gpg4win.org/
- Renesas Trusted Secure IP Key Wrap Service
- e^{2}studio IDE v2024-10
  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:

- RA8M1
- RA8D1
- RA8T1
- RA8E1 (FSBL and DOTF features are not supported)
- RA8E2 (FSBL and DOTF features are not supported)

Hardware

- Evaluation Kit for RA8M1 MCU Group (renesas.com/ra/ek-ra8m1)
- The description in the application note uses 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 device cable (type-A male to micro-B male) to connect between EK-RA8M1 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 e^{2}studio IDE. In addition, background knowledge of RA8 MCU security features is a prerequisite for utilizing the SFP feature, for example Arm®TrustZone®, Secure Key Injection, FSBL Secure Boot, Renesas Secure IP (RSIP-E51A) etc. Please refer to the chapters on *Security Features* and *Renesas Secure IP (RSIP-E51A)* in the *Renesas RA8M1 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 references several key sections from the following two application notes. When exercising the Secure Factory Programming procedure, please have these two Application Notes handy:

- The Renesas Device Lifecycle Management for RA8 MCUs (R11AN0785).
- The Renesas RA Secure Key Injection Application Project (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 RA8 MCU Series

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.

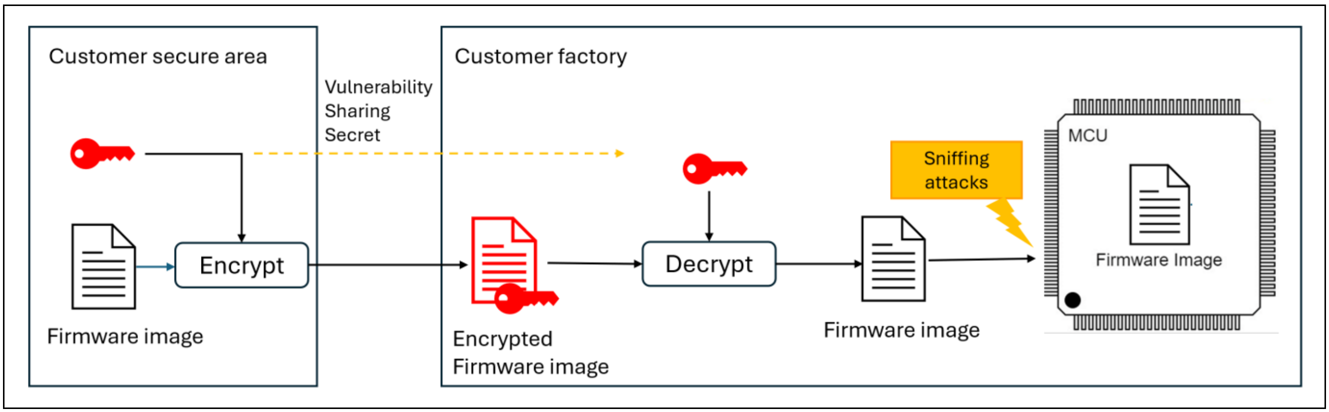


Figure 1. Existing Secure Programming Solution

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 needing a solution to securely program these pre-programmed MCUs.

Additionally, secure key management is crucial in this process, requiring the use of 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 RA8 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 for the RA8 MCU Series 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 MCU’s 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 Renesas SFP process.

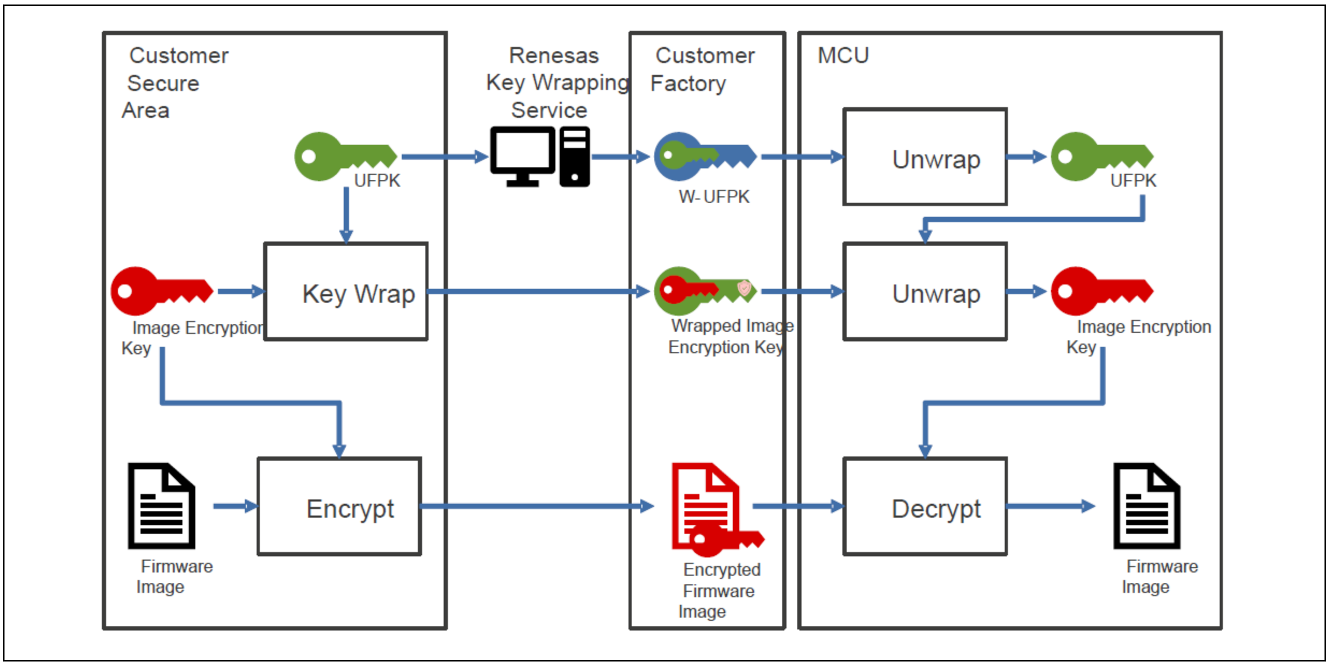


Figure 2. RA8 MCU Secure Factory Programming

1.2.1 Tools Used

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 MCU boot firmware and can be used to securely program the secure factory programming file generated by SKMT and lock the MCU. Refer to section 1.2.4 for the two different MCU locking mechanisms.

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, Authentication Level Key (AL Key). These procedures are described in Chapter 2. A demonstration of using AL2 Key with final OEM/AL state as OEM_PL0 is demonstrated in this application note.
- Prepare the application firmware using the normal RA8 application development procedure.
- Wrap the image encryption key and the AL key and generate the encrypted SFP package. This is achieved using SKMT. The detailed procedure is described in Chapter 3.
- Program the encrypted firmware image: This can be achieved using RFP. The detailed procedure is described in Chapter 3 using four different types of applications. Note that in this process, secure image programming and the image encryption key injection happens in one shot without MCU reset.

Note that when Secure Factory Programming is performed through the boot firmware interface, the entire code flash and data flash except the option-setting memory are erased first prior to programming the application firmware.

1.2.3 Usage Note on the MCU Boot Firmware Version Requirement

The boot firmware version that supports the SFP needs to be version V3.6.32 or above. All MP devices have this boot firmware version or later. To confirm, the Boot Firmware Version can be acquired using the RFP. Refer to the RFP User's Manual for how to establish connection with the Kit. Once connected, navigate to **Target Device** and then select **Read Device Information**.

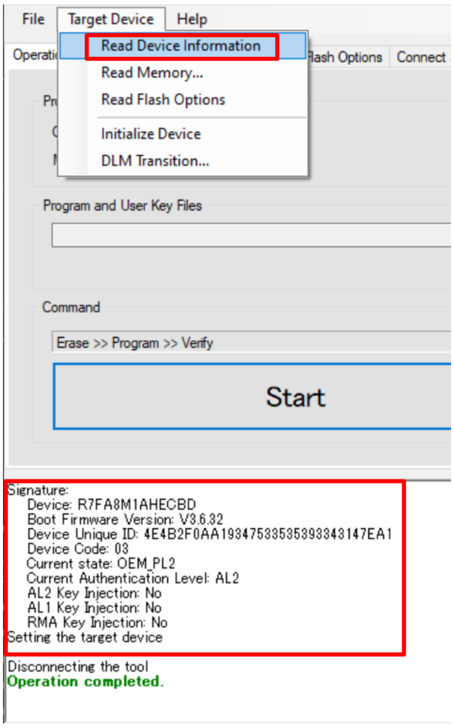


Figure 3. RA8 Boot Firmware Version

1.2.4 Usage Note on the End DLM State after Secure Factory Programming

SKMT provides two options for the final DLM/AL state after programming the encrypted firmware image:

- **OEM PL0 with AL2_KEY**: Final state is OEM_PL0, Authentication level is AL0. AL2 Key is injected so PL can be regressed to AL2 after authentication. Device can be Initialized. Transitioning to OEM_PL0 with AL2_KEY is demonstrated in this application note. It is important to note that after the .sfp file is programmed, the MCU boot programming interfaces and the Debug interfaces are locked. To recover the MCU with these functionalities, the MCU can be regressed to OEM_PL2 using the Initialize command or using the AL2_KEY.
- **LCK_BOOT**: Final state is LCK_BOOT. MCU is locked, the debug interface and the serial programming interfaces are permanently disabled. Be very cautious using this final DLM state when using SFP. It is not recommended to exercise this end state during development stage.

1.3 Collaboration with other RA8 MCU Series Security Solutions

The Renesas Secure Factory Programming (SFP) feature coordinates seamlessly with various security solutions within the RA8 MCU Series. The following are the two key actions to perform when using SFP with other security solutions:

- Prepare a set of credentials. Some credentials can be common when using SFP with other Security solutions, for example, the UFPK and W-UFPK. Other credentials may be specific to the security solutions used.
- Integrate the procedures for other security solutions with the SFP procedure. This application note provides guidelines in this regard.

For the details on the set of credentials needed when using SFP with other security solutions and the description of the integrated procedure, please refer to the corresponding sections in Chapter 3.

- Refer to section 3.3 for the operational procedure when using SFP with TrustZone projects.
- Refer to section 3.4 for the operational procedure when using SFP with FSBL.
- Refer to section 3.5 for the operational procedure when using SFP with DOTF.

2. Cryptographic Key Preparation for Secure Factory Programming

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

2.1 Generate the UFPK and Wrapped UFPK

A User Factory Programming Key (UFPK) and Wrapped User Factory Programming Key (W-UFPK) are necessary for wrapping the Image Encryption Key. Refer to the following two sections in the Application Project for RA8 DLM (R11AN0785) to establish the encrypted communication with the DLM server and generate these credentials.

Refer to section ***Establish PGP-Encrypted Communication with the Renesas DLM Server*** in Application Note R11AN0785 to establish the PGP encrypted communication channel with the Renesas DLM server.
Refer to section ***Create the UFPK and W-UFPK*** in Application Note R11AN0785 to generate UFPK and W-UFPK. The rest of this application project assumes that the following UFPK and W-UFPK have already been created:
`ra8x1_ufpk.key`: the UFPK

`ra8x1_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. A NIST CAVP test vector is used for this example.

https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program/Block-Ciphers

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

Figure 4. NIST AES 128 Test Vector

2.3 Authentication Level 2 Key

The following is a brief description of the AL2_KEY. For more details on the Device Lifecycle Management and Authenticate Keys for RA8, please refer to Application Note R11AN0785.

Authentication Level 2 Key (AL2_KEY) can be injected in the OEM_PL2 state. It is used to transition the MCU from AL0 or AL1 to AL2 when in the OEM DLM state. The AL2_KEY can be permanently disabled in the OEM DLM state at AL2 by using the boot firmware parameter setting command.

Difference in System Behavior with and without AL2_KEY:

- With AL2_KEY: When the AL2_KEY is injected, the MCU can transition to the AL2 state, enabling both non-secure and secure debug functions. This allows for full access to the MCU's debug capabilities, which is essential for development and troubleshooting.
- Without AL2_KEY: If the AL2_KEY is not present, the MCU cannot transition to the AL2 state without erasing the flash memory content. This restriction enhances security by limiting access to sensitive areas of the MCU.

In the SFP programming demonstration described in this application note, the AL2 Key will be injected during the SFP programming. This allows the device to be transitioned to OEM_PL2 without erasing the programmed application.

In this example, the following raw 128-bit raw key is used as the plaintext AL2 Key.
`000102030405060708090A0B0C0D0E0F`

3. Firmware Image Encryption and Programming

This section provides a demonstration of SFP using OEM PL0 with AL2_KEY as final DLM/AL 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.

3.1 Set up the Hardware and Initialize the MCU

Follow the below two guidelines to connect the EK-RA8M1 to the PC.

- Use the default jumper setting on the EK-RA8M1. Refer to the EK-RA8M1 User’s Manual to confirm the jumper settings.
- Connect J10 on EK-RA8M1 to the Development PC using the USB type-A male to micro-B male cable to provide power, Debug access, and MCU boot firmware access to the board.

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 usage of RDPM. For RFP usage, please refer to the RFP User’s Manual.

Launch e^{2}studio, then launch the **RDPM.**

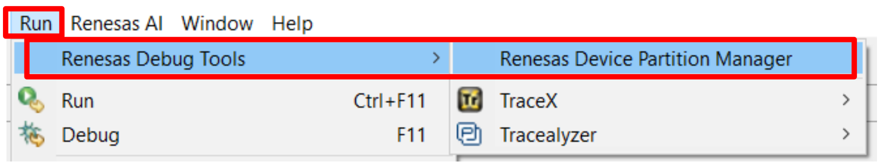


Figure 5. Open the Renesas Device Partition Manager

Next, check **Initialize device back to factory default**, choose the connection method, then click **Run**. For EK-RA8M1, choose either SCI or SWD as the connection method if the default jumpers are in place (Refer to the EK-RA8M1 User’s Manual for the default jumper setting). For a custom PCB board, the Connection Type should be selected based on the Boot Mode interfaces available.

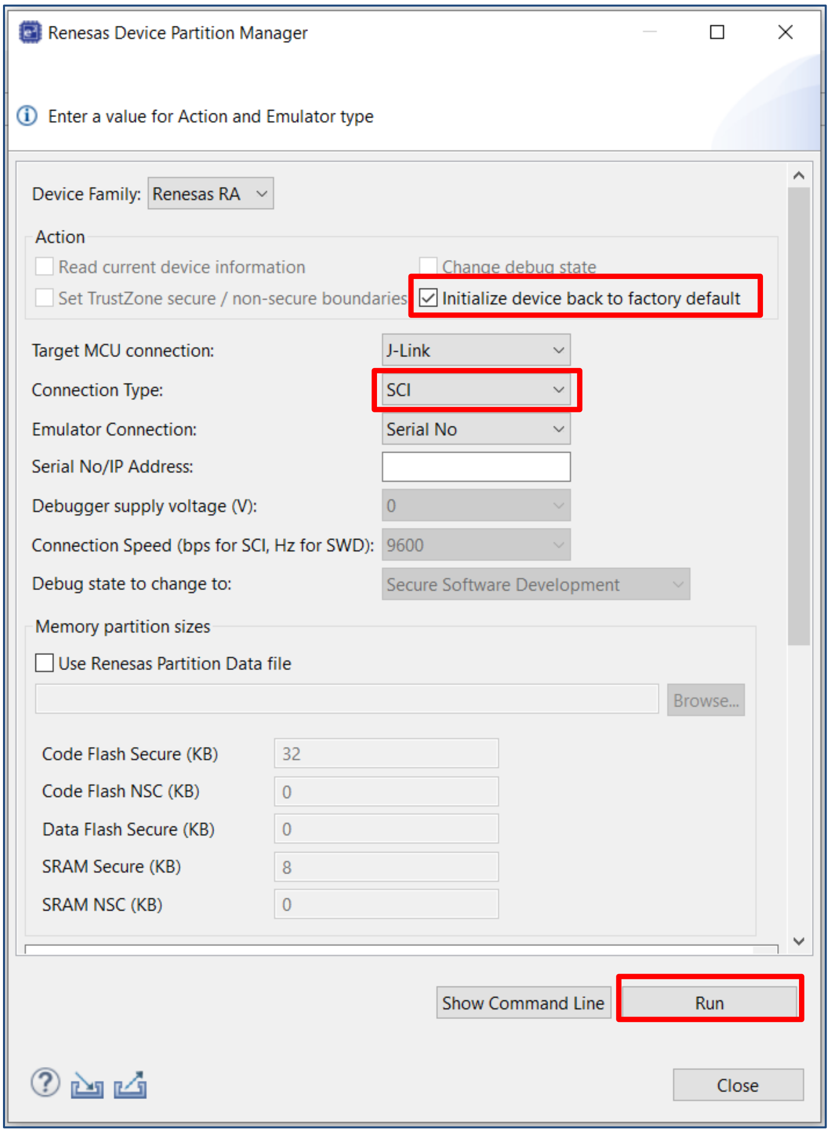


Figure 6. Initialize RA8M1 using Renesas Device Partition Manager

Ensure the following output is achieved.

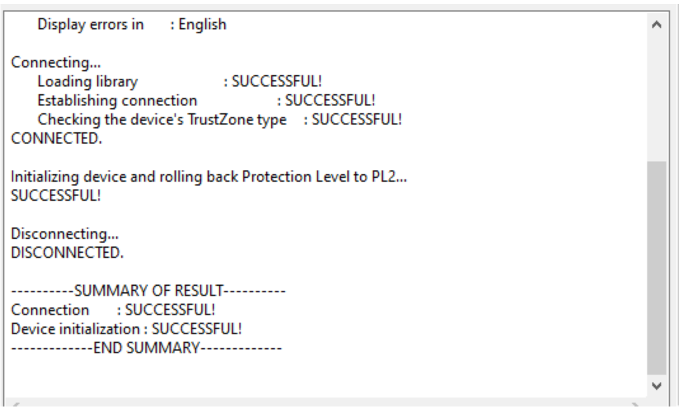


Figure 7. MCU Initialization

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

3.2 Using a Flat Blinky Project as an Example

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.

Care should be taken when using SFP with other security solutions. Refer to sections 3.3, 3.4 and 3.5 for some typical use cases.

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**.

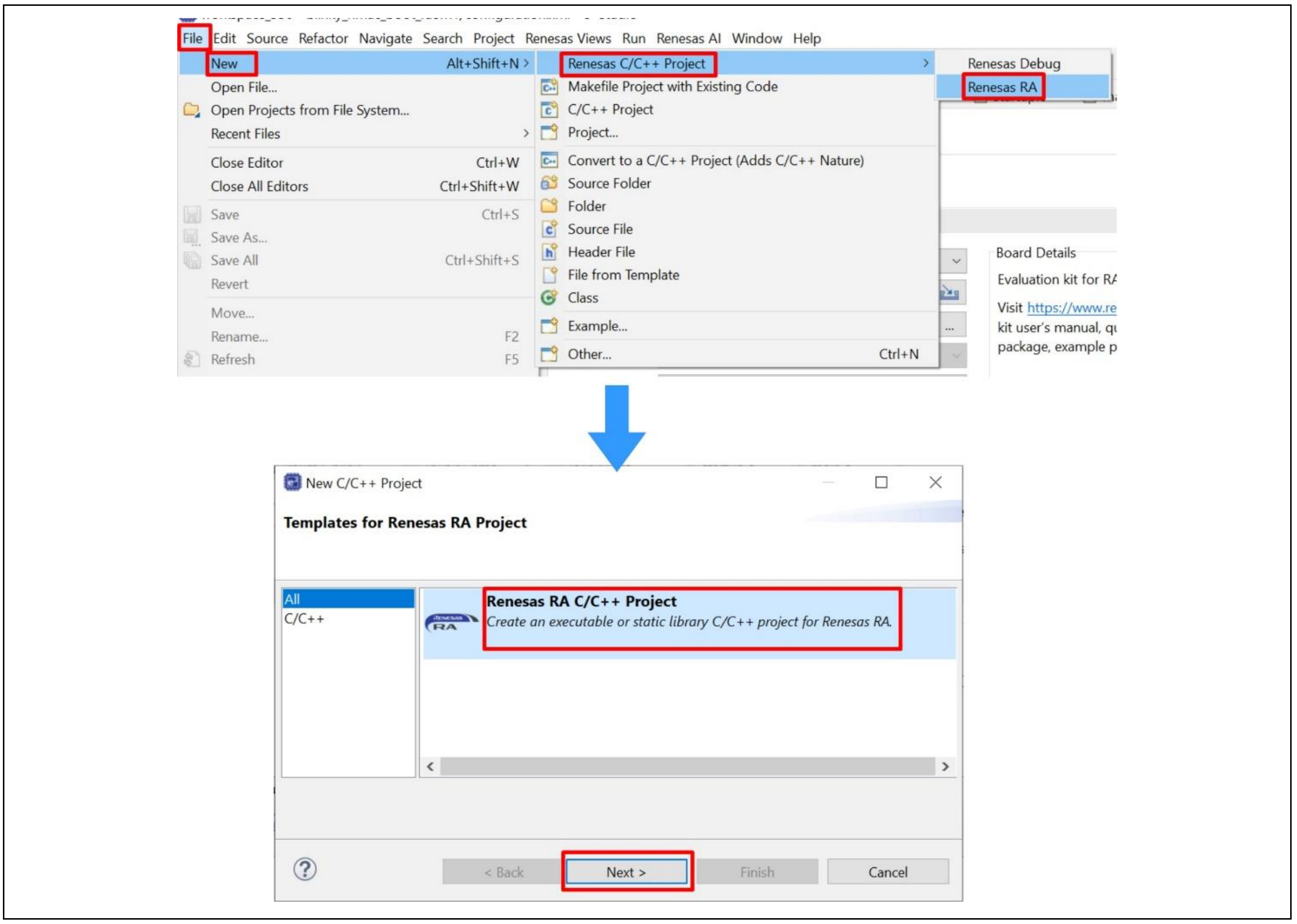


Figure 8. Select Renesas RA C/C++ Project

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

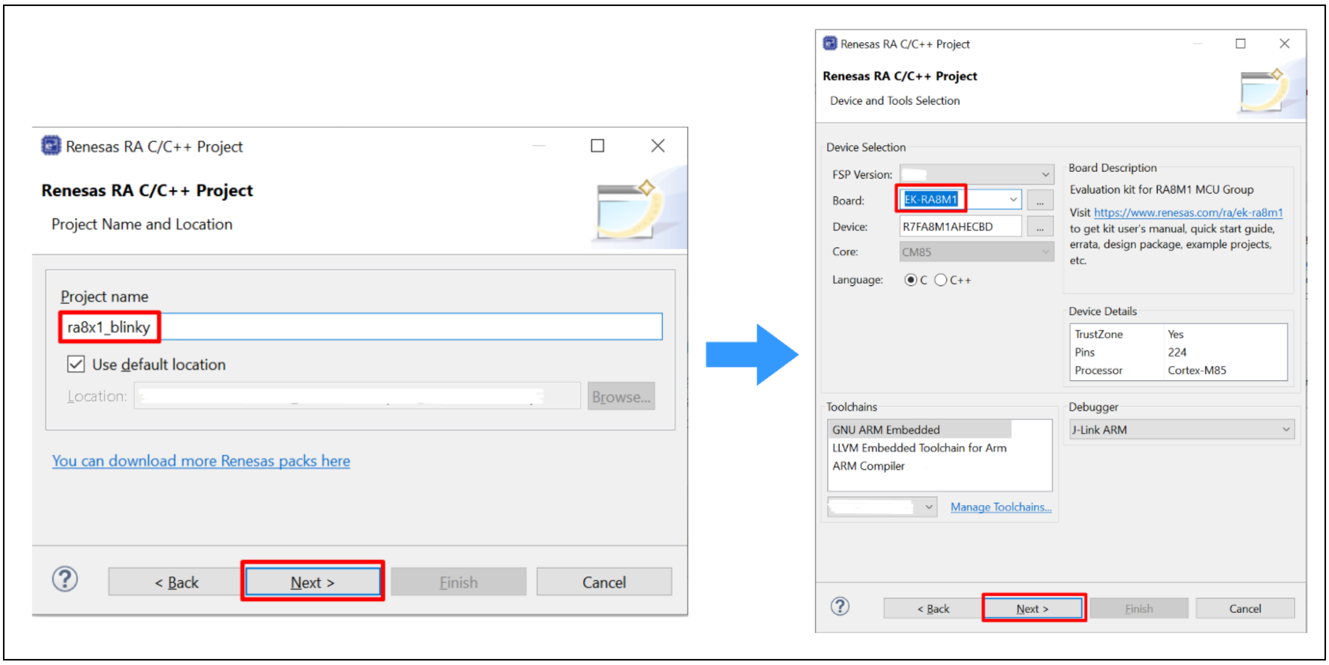


Figure 9. Assign project's name and select a board

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**.

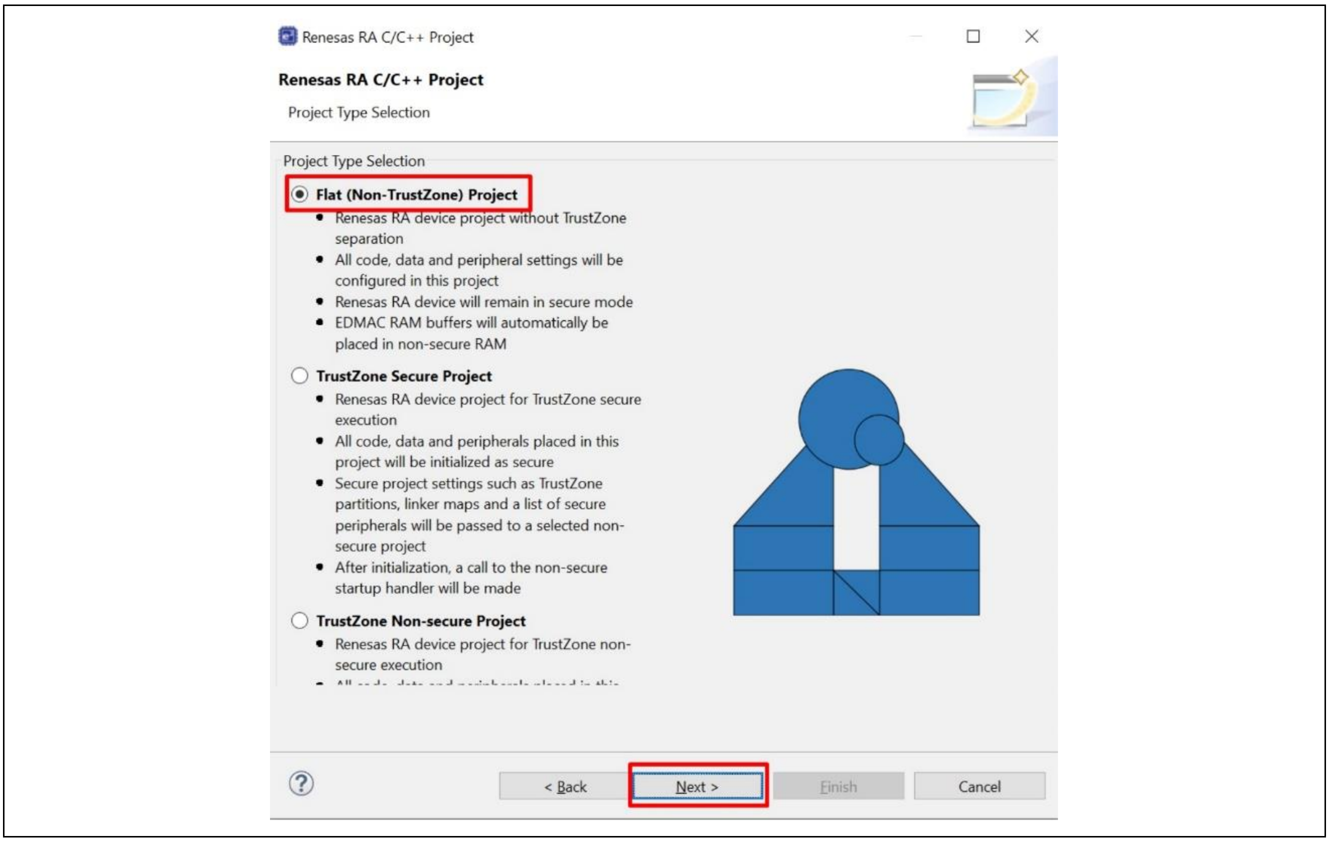


Figure 10. Blinky Project Type Selection

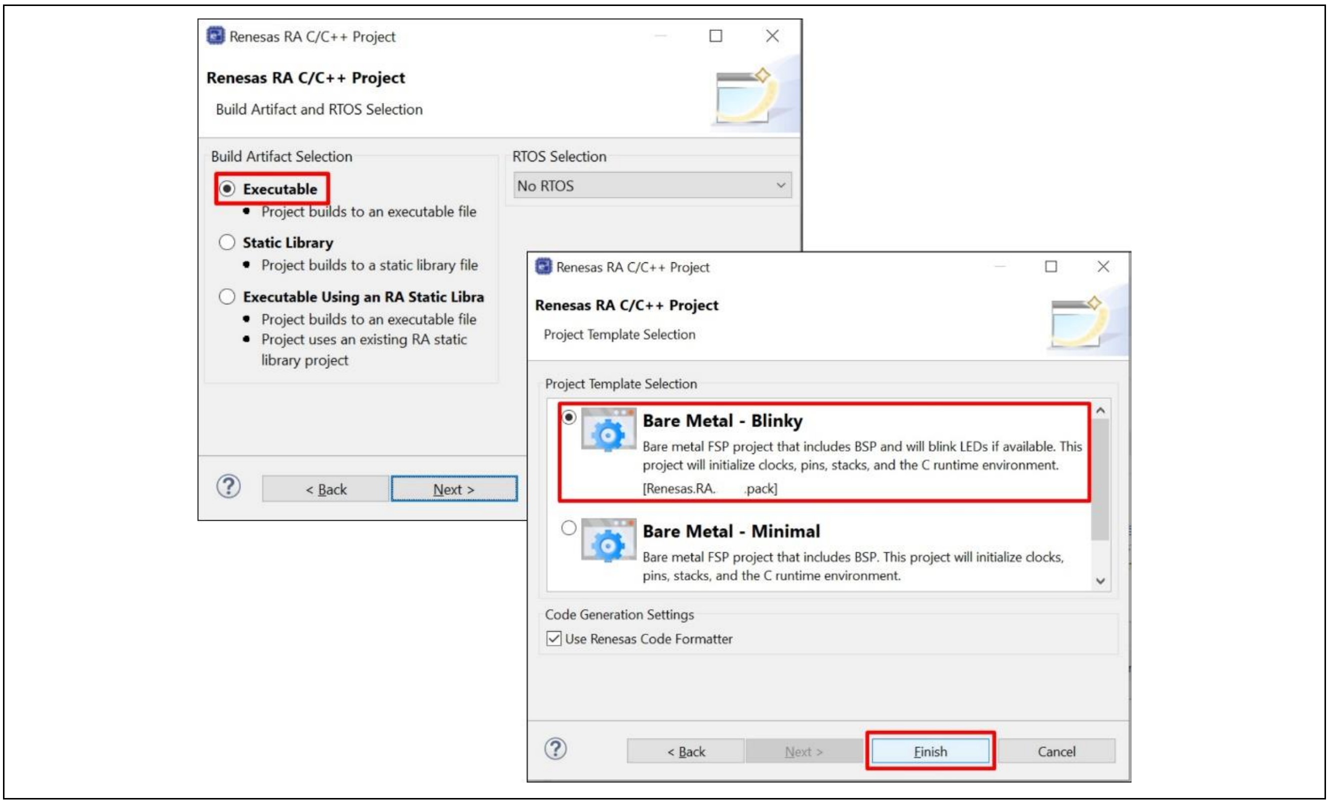
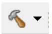


Figure 11. Blinky Project Template Selection

Once the project has been created, the name of the project will show up in the **Project Explorer** window of e[2]studio. 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 (`ra8x1_blinky.srec`) is generated.

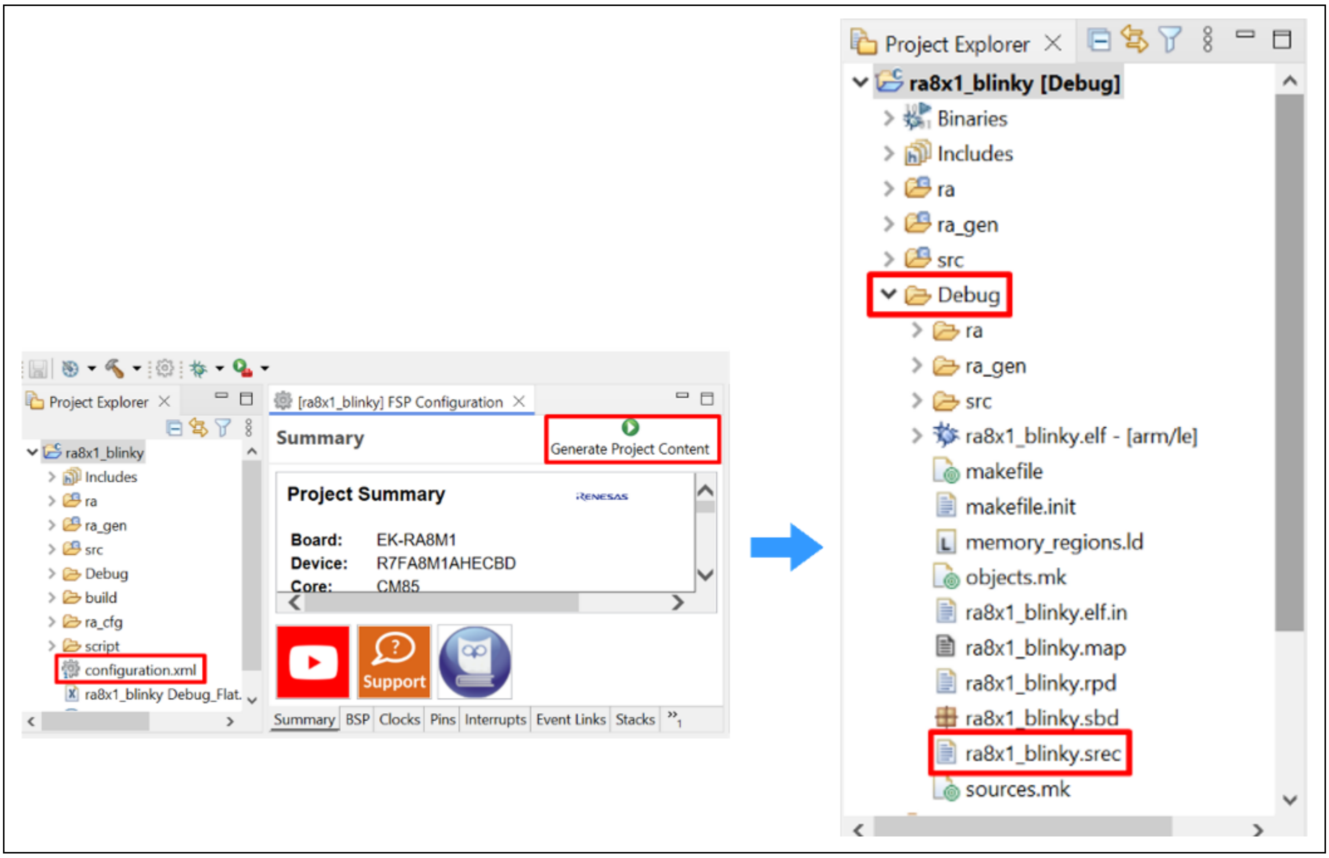


Figure 12. Build the Blinky Project

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 **SKMT GUI**. Under the **Overview** tab, select **RA Family, RSIP-E51A Security Functions and Protected Mode**.

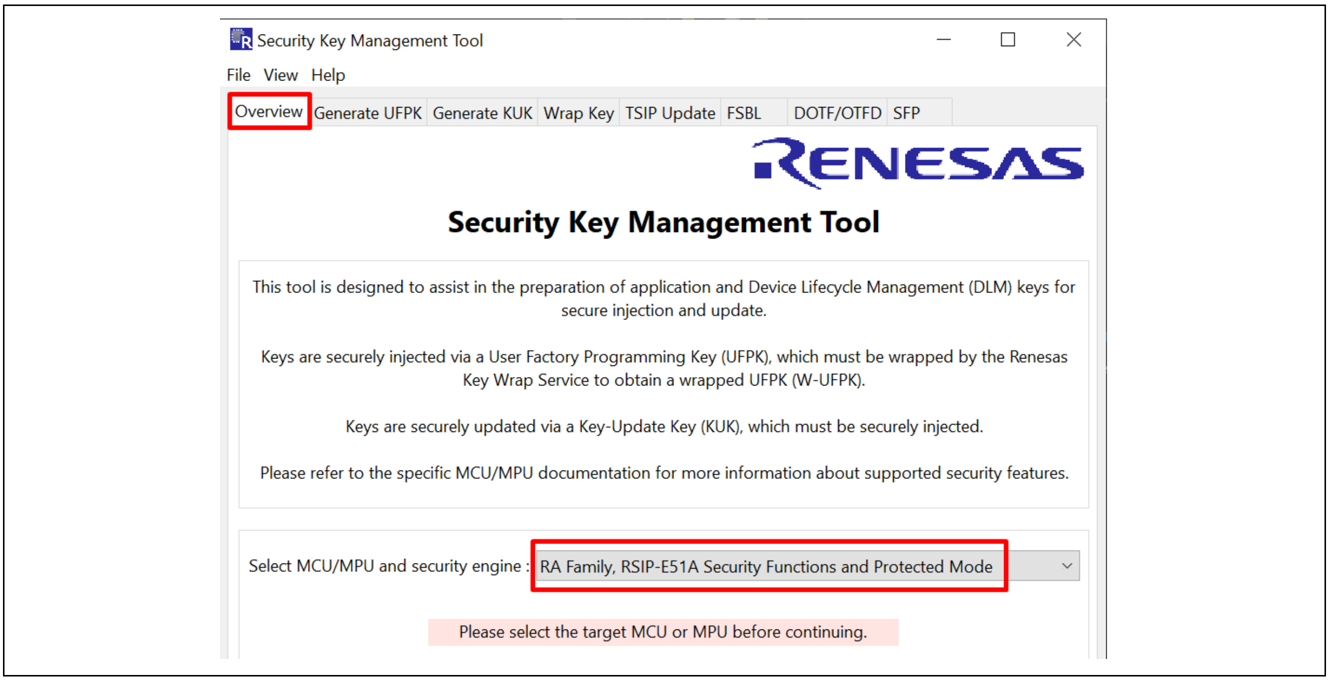


Figure 13. Select RA Family, RSIP Security Functions and Protected Mode

On the **SFP** tab, **Firmware Images** tab, browse the **Plaintext Image** (`ra8x1_blinky.srec`) and add it by clicking the **Add** button.

Select the destination DLM/AL state after writing the firmware image in **Final DLM/AL State**. In this example demonstration, select "**OEM PL0 with AL2_KEY**".

Specify the output file name in **Secure Programming File** (`ra8x1_blinky.sfp`).

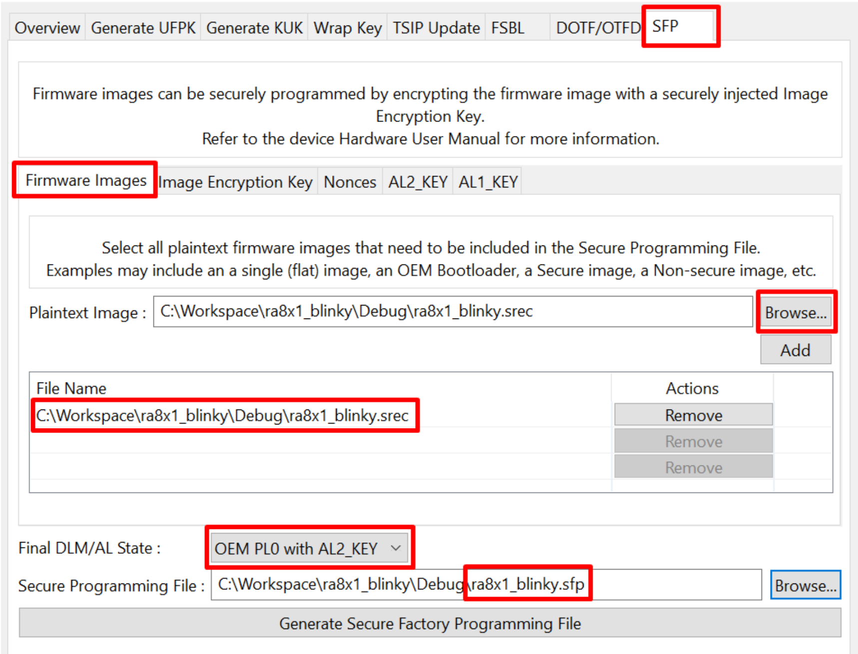


Figure 14. Setting the Firmware Image

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.

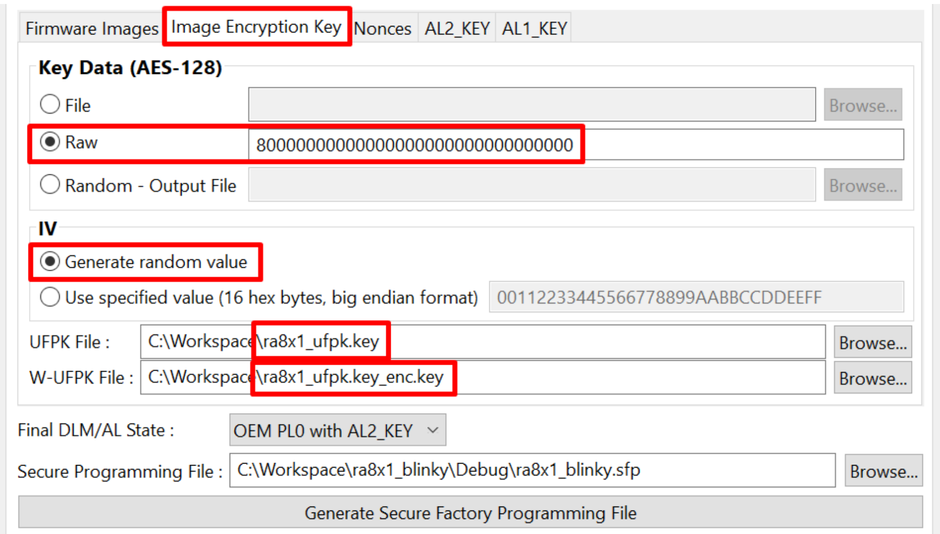


Figure 15. Setting the Image Encryption Key

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

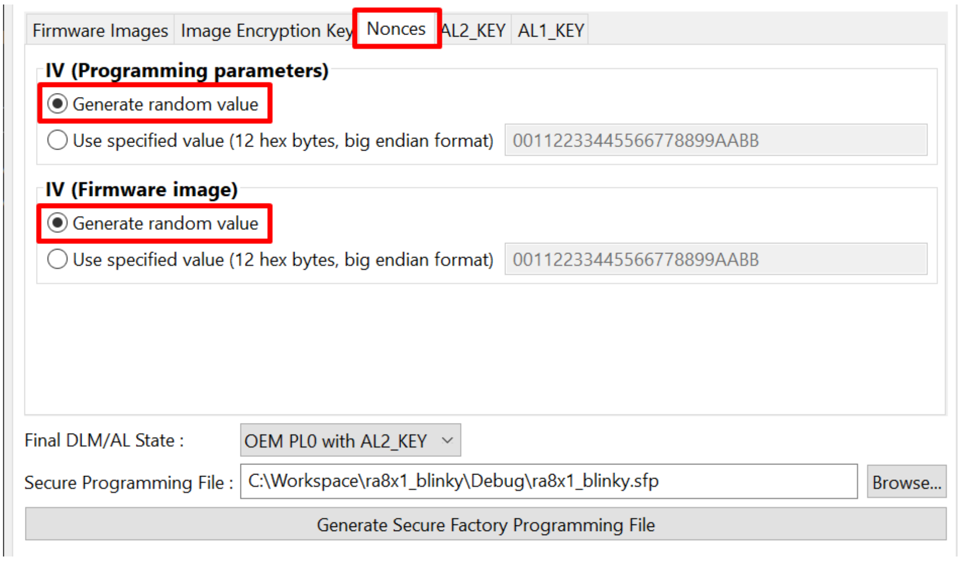


Figure 16. Setting the Nonces

In **AL2_KEY** tab, specify the AES 128-bit key data to be used. For this example, we will specify the key as raw data by selecting **Raw** and entering the key value (`000102030405060708090A0B0C0D0E0F`), but we could also specify a binary key file containing the key value.

Choose **Generate random value** for the **IV**.

Click the **Generate Secure Factory Programming File** button to create the output file. If the file is successfully generated, the "OPERATION SUCCESSFUL" will appear and the output file `ra8x1_blinky.sfp` will be created.

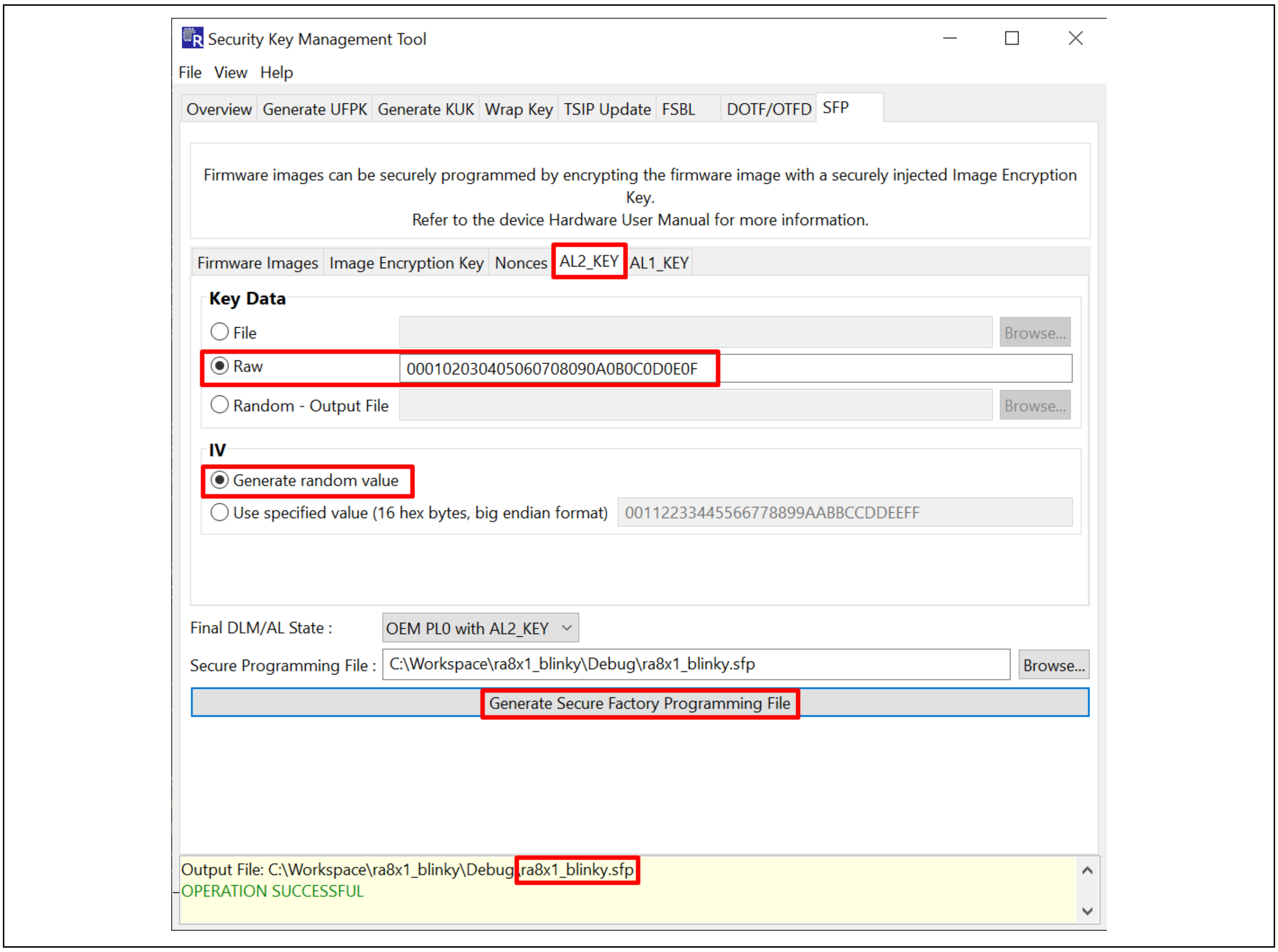


Figure 17. AL2 Key setting and the Output

Note that during the generation of the .sfp file, the UFPK, W-UFPK, Image Encryption Key, AL2_KEY information as well as the application image are used in this process. During this process, the AL2_KEY and the Image Encryption Key are wrapped with the UFPK, and the application is encrypted by the Image Encryption Key. The final DLM state and Protection Level information are also included in the generated .sfp file. When the .sfp file is programmed on the MCU through the boot interface, the AL2_KEY will be stored wrapped by the MCU HUK, and the UFPK and Image Encryption Key will be discarded.

3.2.3 Program the Encrypted Firmware Image with RFP

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**. Use the **Initialize Device** command within RFP or follow section 3.1 to initialize the MCU.

After the Initialize command is successfully executed, the DLM state of the MCU is OEM_PL2/AL2.

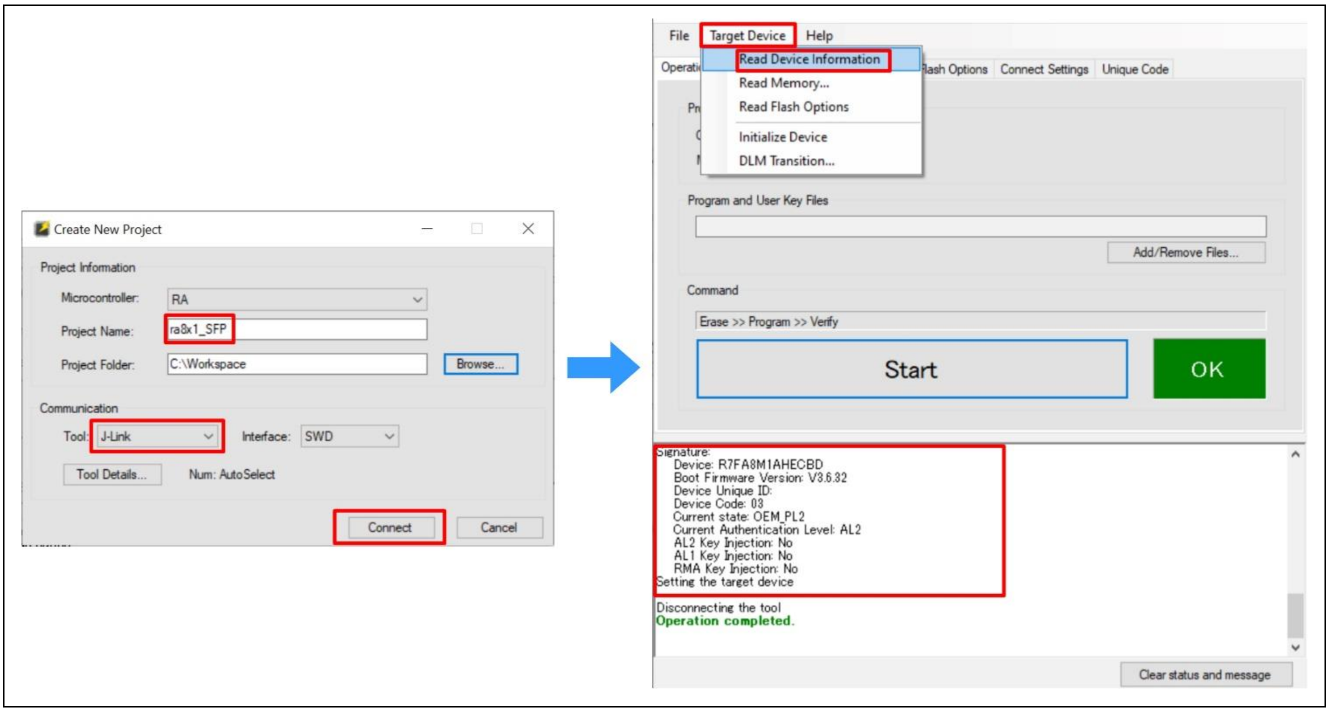


Figure 18. Create a new RFP project

Add the `ra8x1_blinky.sfp` file, then click **Start**. The encrypted firmware image will be programmed, and LEDs (red, green, blue) will blink on the board.

After the .sfp file is programmed, the system is in OEM_PL0 state, and the debugging and boot firmware programming interfaces are locked.

To re-enable the Debug interface, the OEM state can be transitioned to OEM_PL2 using the plaintext AL2_KEY without erasing the programed firmware. This approach allows failure analysis of the existing firmware. Alternatively, the Initialize command can be used to regress the OEM state from OEM_PL0 to OEM_PL2. However, this will erase the programmed firmware image.

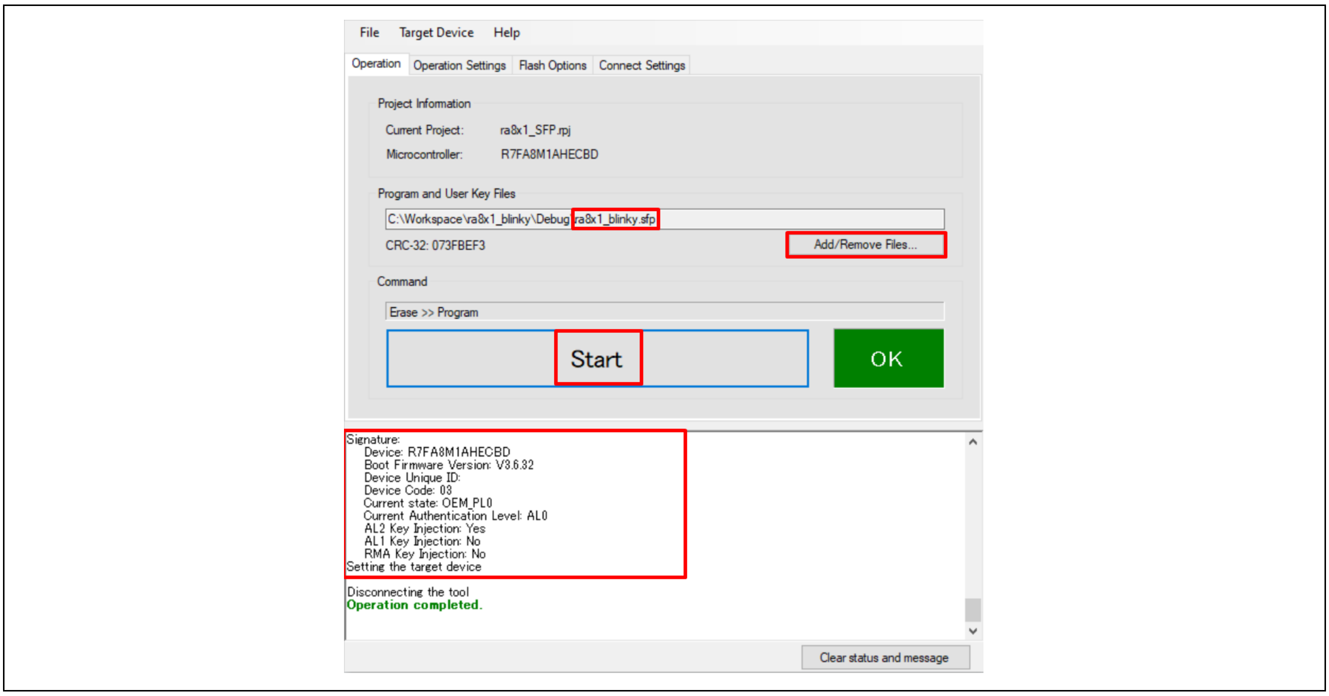


Figure 19. Encrypted Firmware Image is programmed by RFP

3.3 Using SFP with TrustZone Projects

3.3.1 Application Preparation

SFP can be used with TrustZone projects. Refer to Application Note R11AN0897 for a set of example TrustZone projects. Assume the following two .srec files are generated:

- Secure project: `pre_programmed_sensor_algorithm_s.srec`

This project contains secure temperature monitor algorithms.

- Non-Secure project: `pre_programmed_sensor_algorithm_ns.srec`

This project runs in non-secure region and accesses the secure algorithm via non-secure callable API.

3.3.2 Encrypt the firmware image

Follow the steps in section 3.2.2 to use SKMT to encrypt the firmware images and generate the output file: `TrustZone.sfp`. Notice that in **Firmware Images** tab, add `pre_programmed_sensor_algorithm_s.srec` and `pre_programmed_sensor_algorithm_ns.srec` as shown in Figure 20. The secure and non-secure .srec files will be combined into one file.

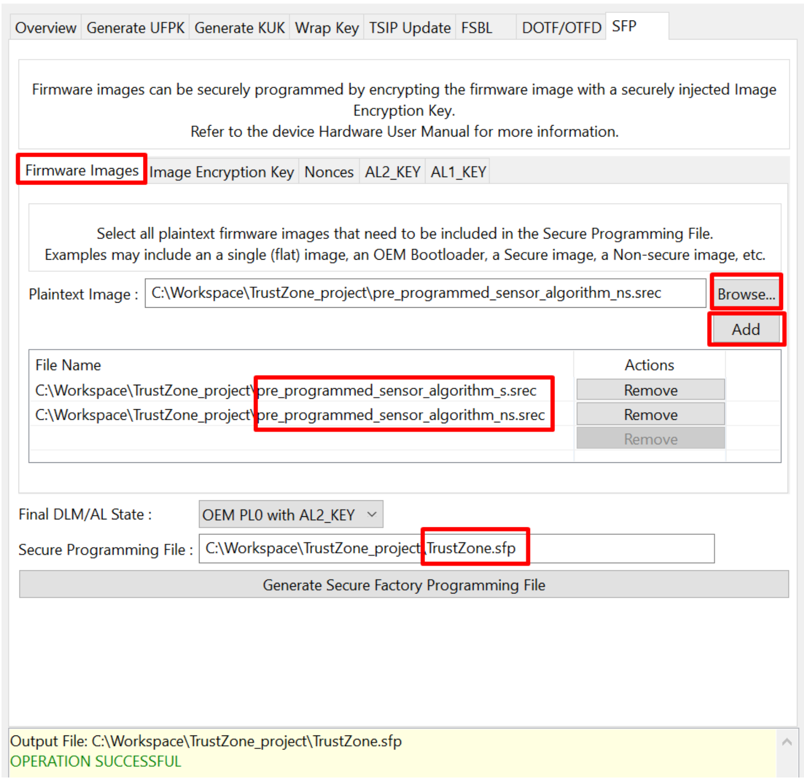


Figure 20. Select the Secure and Non-secure Application to Generate the SFP File

3.3.3 Program TrustZone Application

Using RFP, select `TrustZone.sfp` file to program to the MCU.

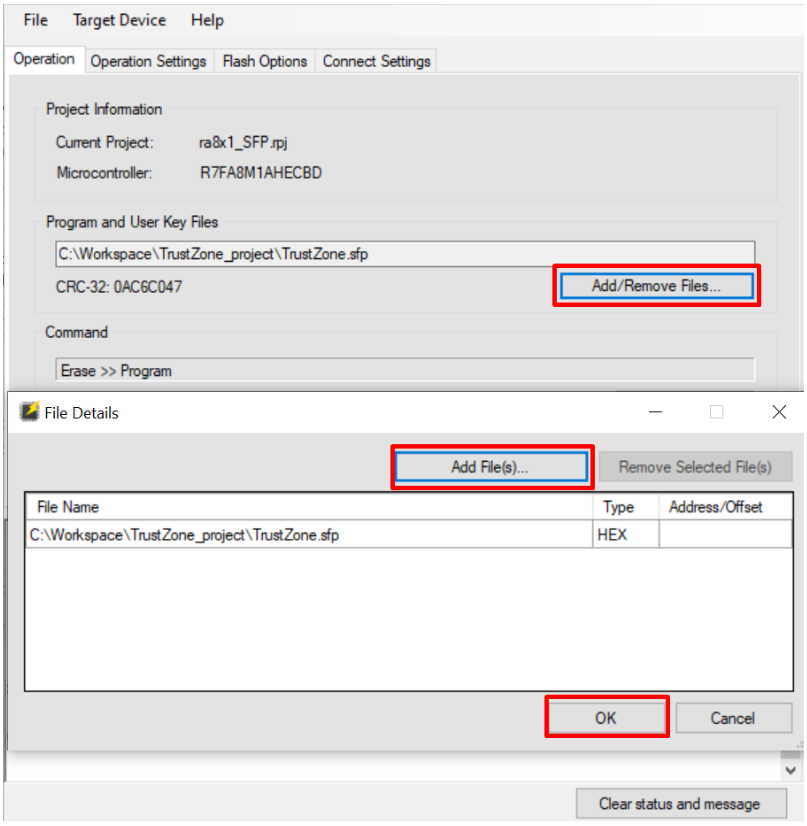


Figure 21. Select the TrustZone file

Select the **Program/Verify Flash Options** in **Operation Settings** tab.

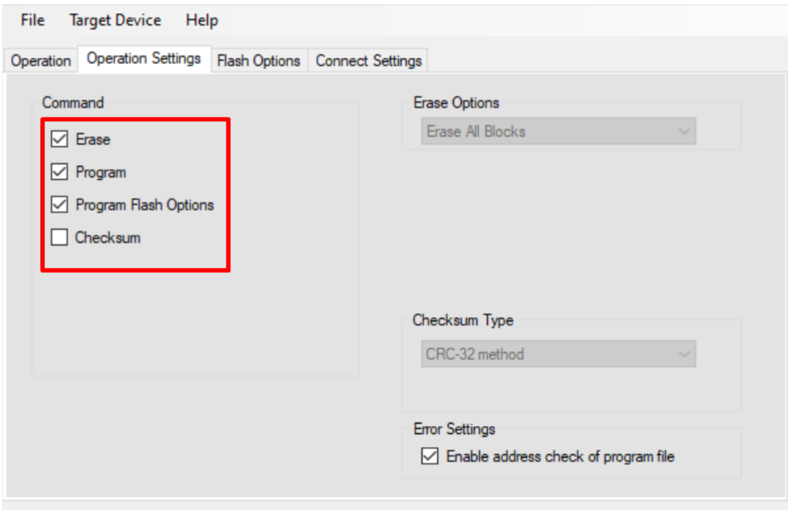


Figure 22. Configure the Operation Settings

Set up the **Flash Options** to configure the TrustZone memory boundary.

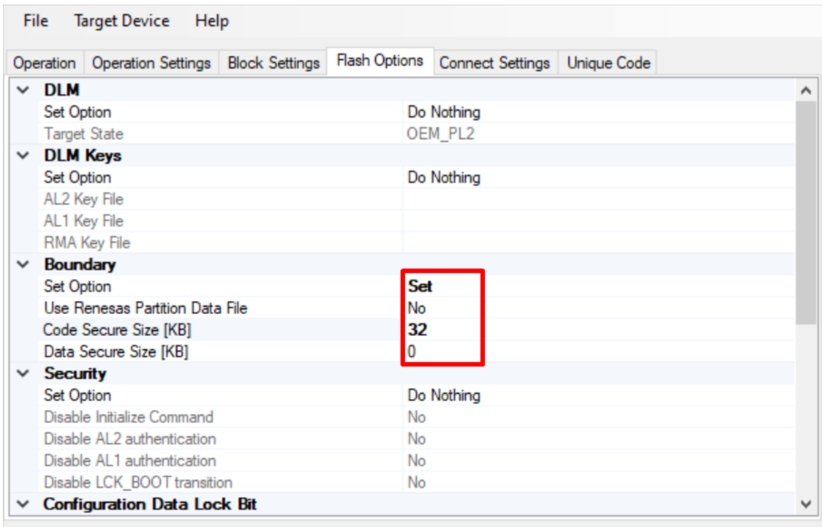


Figure 23. Configure the Boundary

Then go back to the **Operation** tab and click **Start** button to program the image with SFP.

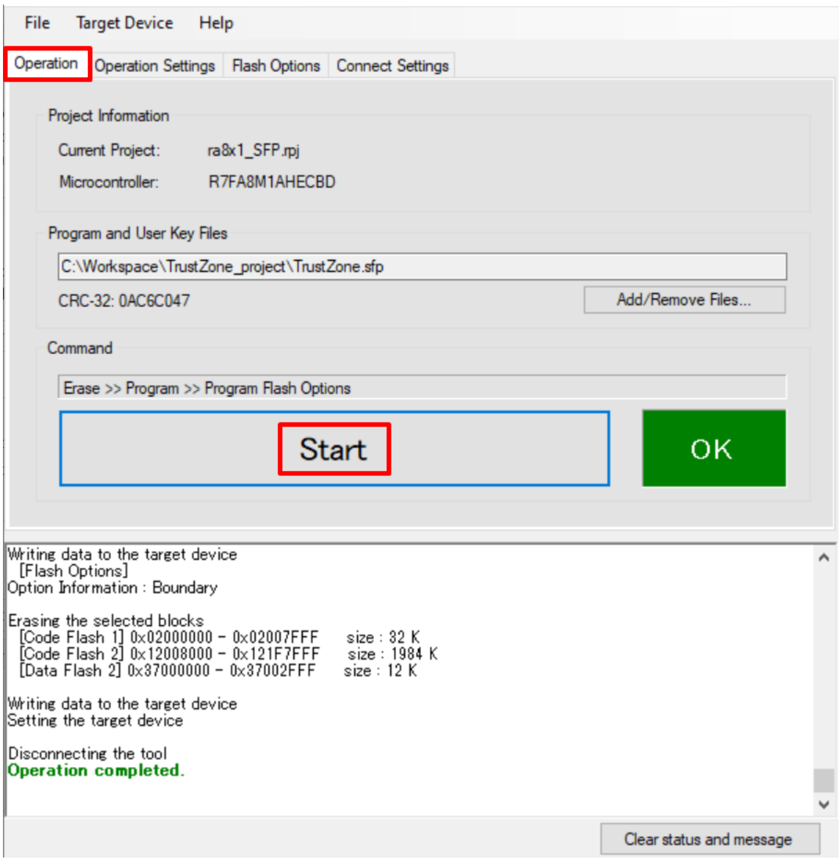


Figure 24. Program the secure application project

3.4 Using SFP with FSBL Enabled

3.4.1 Credentials Preparation

When using SFP with **FSBL Secure Boot** enabled, follow Application Note R11AN0774 to create the following credentials.

- Wrapped OEM Root Key (`oem_root_key.rkey`)

- Code Certificate (`blinky_hmac_boot_ra8m1_code_cert.bin`)
- Key Certificate (`blinky_hmac_boot_ra8m1_key_cert.bin`)
- Blinky firmware image with FSBL Secure Boot configuration (`blinky_hmac_boot_ra8m1.srec`)

When using SFP with **FSBL CRC Boot** enabled, follow Application Note R11AN0774 to create the following credentials:

- Code Certificate (`blinky_crc_boot_ra8m1_code_cert.bin`)
- Blinky firmware image with FSBL CRC Boot configuration (`blinky_crc_boot_ra8m1.srec`)

Encrypt the firmware image (`*.srec`) using the same steps in section 3.2.2 to generate the output file `blinky_hmac_boot_ra8m1.sfp`/`blinky_crc_boot_ra8m1.sfp`.

3.4.2 Program Applications with FSBL Enabled

When using the FSBL with **Secure Boot**, follow these steps to utilize the SFP solution.

Select the `blinky_hmac_boot_ra8m1.sfp` file under the **Operation** tab.

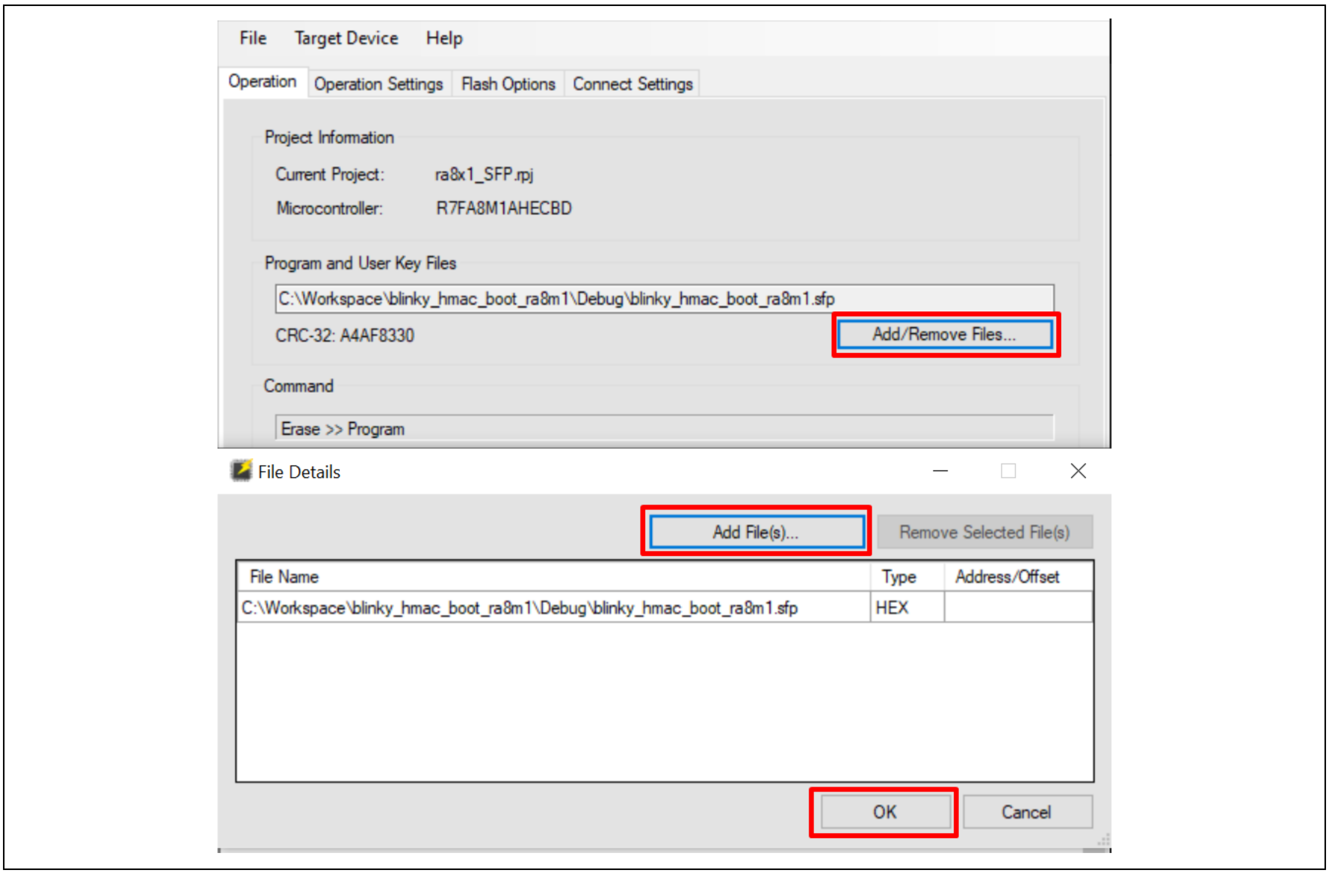


Figure 25. Select the Application Project

Select the **Erase**, **Program** and **Program Flash Options** under the **Operation Settings** tab.

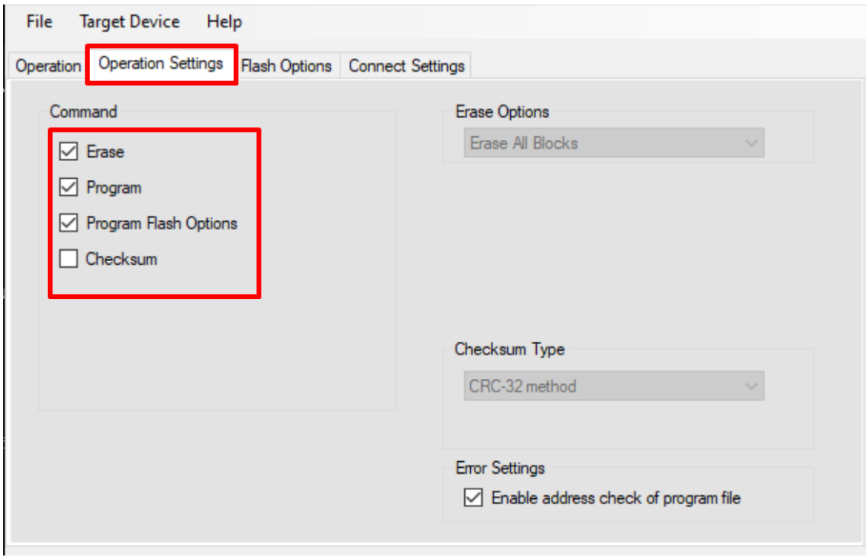


Figure 26. Configure the Operation Settings

Set up the **Flash Options** to configure the **OEM Root Public Key**, **Code Certificate** and **Key Certificate**.

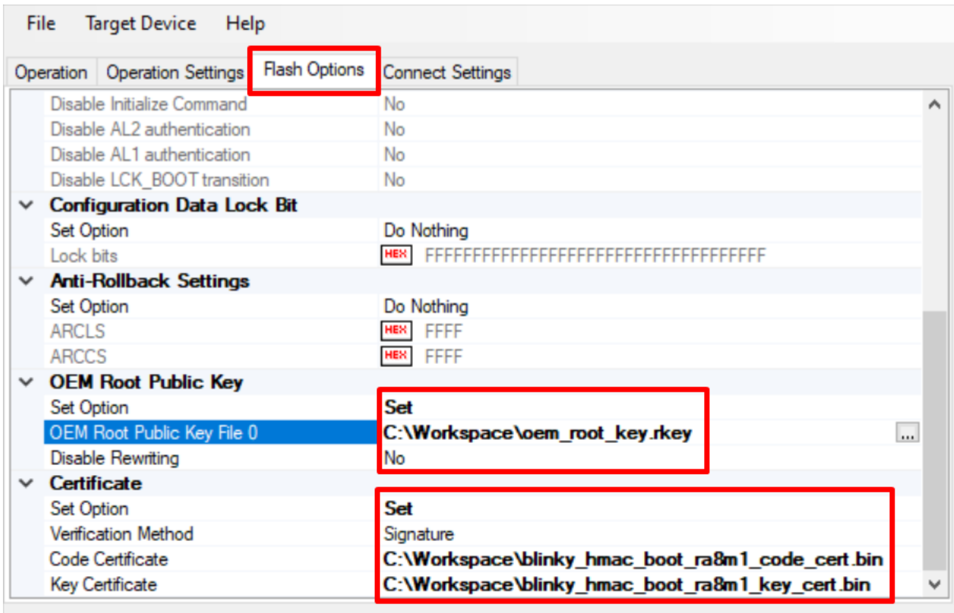


Figure 27. Configure the OEM Root Key and the Certificates

Next, under the **Operation** tab, click **Start.** All the credentials and the firmware will be programmed, the application will be authenticated, and the DLM/AL state will change transitioned to OEM_PL0.

When using the FSBL with **CRC Boot**, follow these steps to utilize the SFP solution.

Select the `blinky_crc_boot_ra8m1.sfp` file under the **Operation** tab.

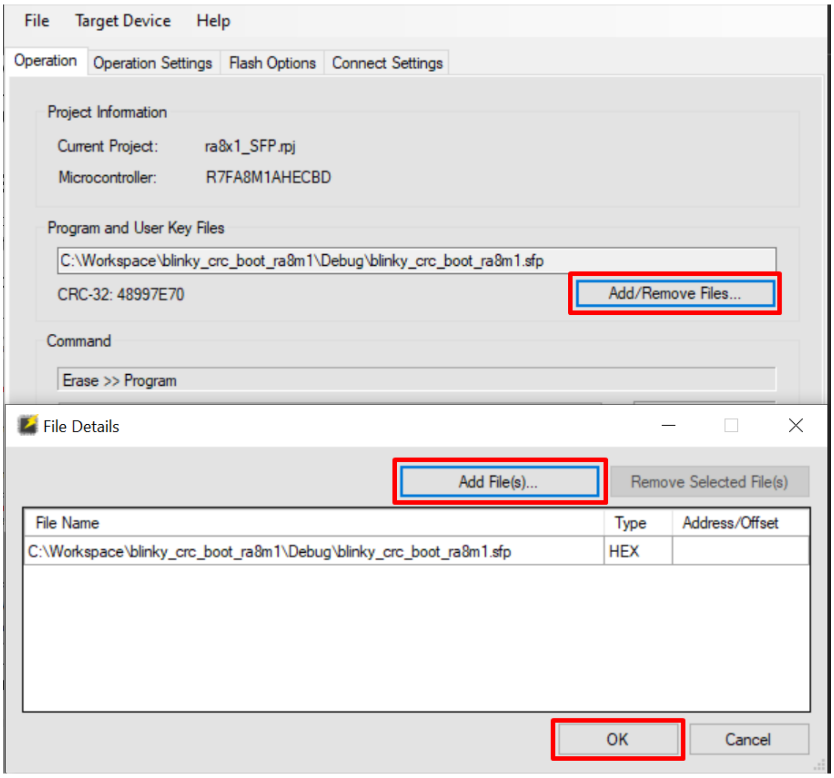


Figure 28. Select the Application Project

Select the **Erase**, **Program** and **Program Flash Options** under the **Operation Settings** tab as shown in Figure 29.

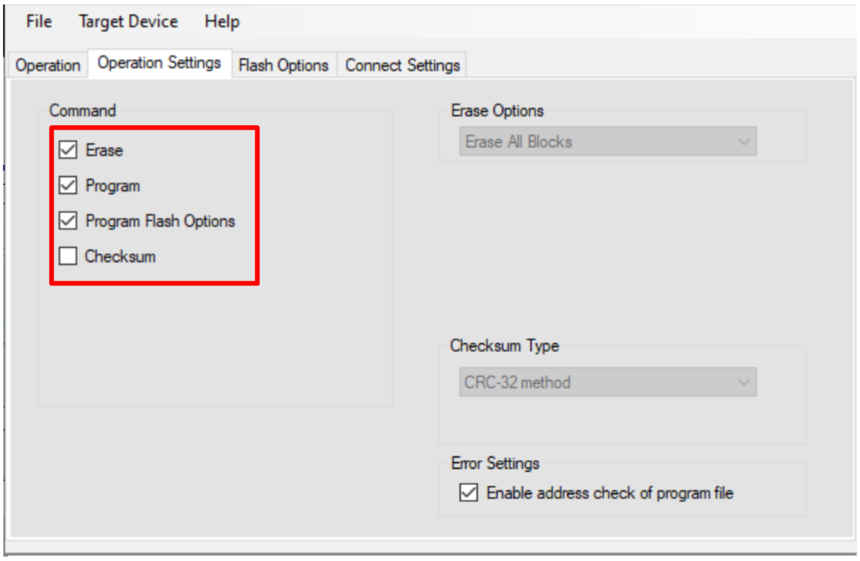


Figure 29. Configure the Operation Settings

Set up the **Flash Options** to configure the **Code Certificate**.

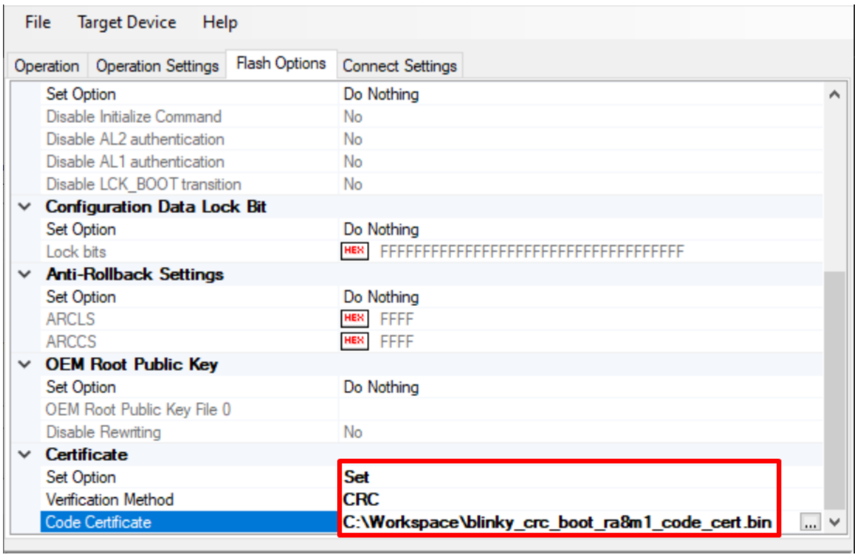


Figure 30. Configure the CRC Code Certificate

Next, click the **Start** button under the **Operation** tab. The encrypted firmware image is authenticated and programmed, and LEDs (red, green, blue) should blink.

3.4.3 Usage Note on Using SFP with FSBL and MCUboot Solution

Usage of MCUboot solution with FSBL and SFP are supported. Please follow the following guidelines:

1. Refer to Application Note R11AN0909 to set up the MCUboot projects.
2. Update the bootloader project to enable FSBL and set up the related registers by referencing Application Note R11AN0774.
3. Generate an .srec file that include the bootloader image and the signed Primary application image. Assume this .srec file is named `combined.srec`. The following guidelines should be used to generate this .srec file.

   1) Generate the .srec file for the Primary signed image using `srec_cat.exe`. The `srec_scat.exe` can be downloaded from https://srecord.sourceforge.net.

   Assuming the signed Primary application image is `primary_app.signed.bin` and its load address is 0x2010000, the following is an example command to generate the .srec file:

   ```
   srec_cat.exe primary_app.signed.bin -binary -offset 0x2010000 -o
   primary_app_signed.srec
   ```

   For TrustZone based applications, the Primary image should include the signed Secure and Non-secure application.

   2) Combine the bootloader .srec file and the Primary signed image .srec file using the same procedure as shown in section 3.3.2. As an alternative, `srec_scat.exe` can also be used to generate the `combined.srec` file.

   3) With Dualbank flash mode, the `combined.srec` file should include the upper bank bootloader, the lower bank bootloader, and the Primary image.

4. Encrypt the `combined.srec` file to generate `combined.sfp` (refer to section 3.2.2).
5. Generate all credentials based on the boot mode (refer to section 3.4.1).
6. Use the one step MCU provisioning of SFP functionality and RFP to program the .sfp and the credentials in the same way as shown in section 3.4.2.

3.5 Using SFP with DOTF Solution

When using SFP with DOTF solution, if J-Link is used to program the OSPI, then the OSPI area data/code (pre-encrypted and plaintext) must be programmed to the MCU first in the same way as when SFP is not used.

The firmware that resides on the MCU should be encrypted using the same procedure as shown in section 3.2.2.

When using DOTF with RSIP Protected Mode, the wrapped DOTF key and the encrypted firmware image (excluding the OSPI area) must be programmed together to the MCU. Assume that the Wrapped DOTF Key is `DOTF_AES_128_RA8M1.rkey`, and it is injected to the start of the first sector of the data flash 0x27000000. Also assume the encrypted firmware image is `ra_app_image_dotf.sfp`. Both files should be added to the **Program and User Key Files** configuration in RFP.

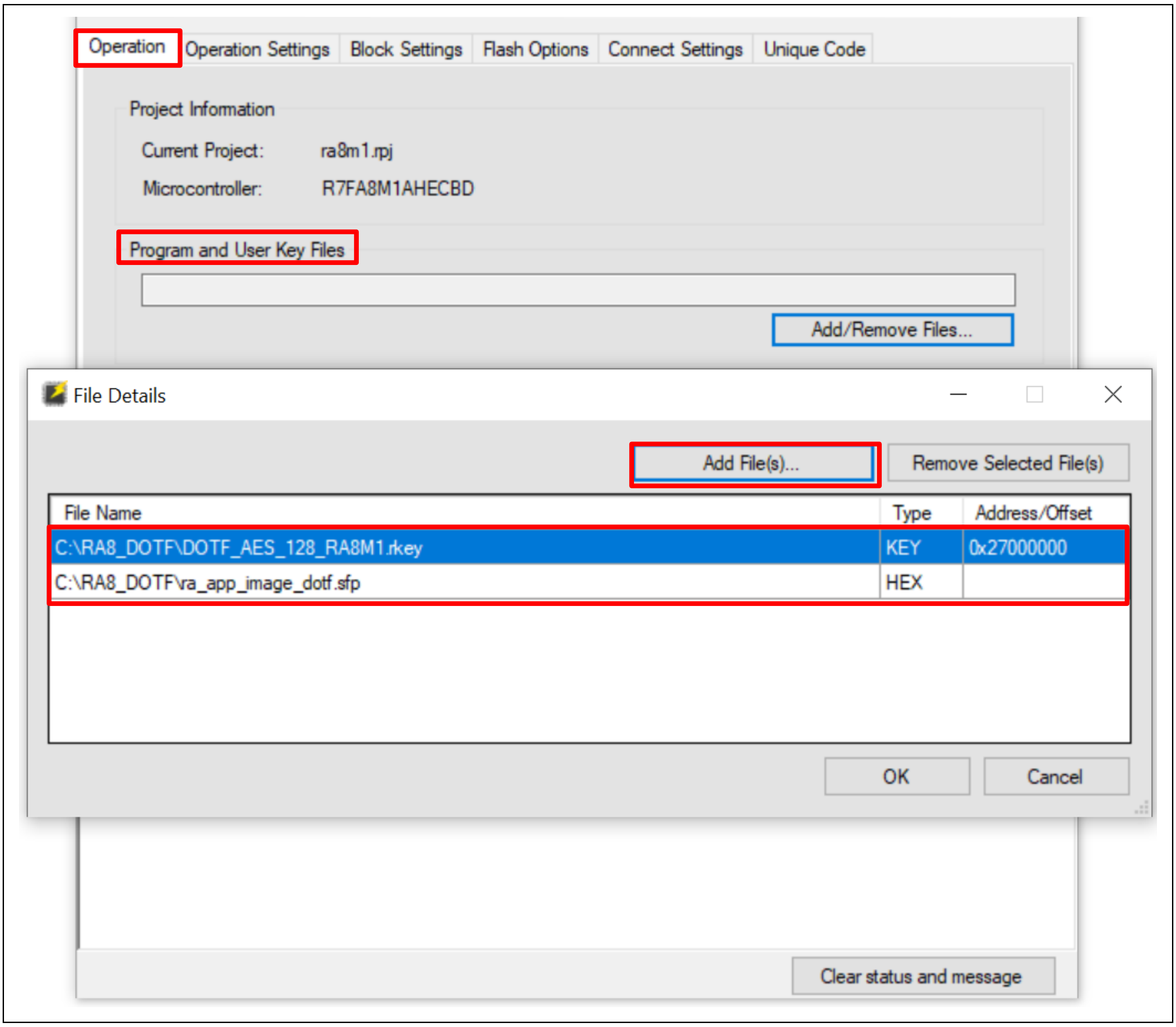


Figure 31. Inject Wrapped DOTF Key and Program the Encrypted Firmware

Once the encrypted firmware and the Wrapped DOTF key are added to be programmed, click **OK** and then **Start** on the next screen to program them to the MCU. The DOTF RSIP Protected Mode application will be programmed, and the DLM/AL state will be transitioned to OEM_PL0.

4. References

1. Flexible Software Package (FSP) User's Manual
2. Renesas RA8M1 Group User's Manual: Hardware (R01UH0994)
3. Renesas RA Family RA8 MCU Series Device Lifecycle Management (R11AN0785)
4. Renesas RA Family MCU Injecting and Updating Secure User Keys (R11AN0496)
5. Renesas RA Family MCU Injection Plaintext User Keys (R11AN0473)
6. Renesas RA Family MCU Renesas Secure IP and Secure Crypto Engine Operational Modes (R11AN0498)
7. Renesas RA Family Application Design using RA8 MCU Decrypt on The Fly (R11AN0773)
8. Renesas RA Family Application Design using RA8 First Stage Bootloader (R11AN0774)
9. Renesas RA Family RA8 Basic Secure Bootloader Using MCUboot and Internal Code Flash (R11AN0909)

5. Website and Support

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

EK-RA8M1 Resources	renesas.com/ra/ek-ra8m1
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	Nov.04.24	—	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

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

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/.