

## Renesas RA Family

# Establishing and Protecting Device Identity using RSIP

### Introduction

This application note offers a general discussion on IoT security and offers a brief introduction to the security features offered by the Renesas RA Family MCUs with Arm® TrustZone® for Cortex-M technology support, including different key generation options.

This package provides an example application using the Renesas Secure IP (RSIP) module to generate a pair of cryptographic keys using a local Certificate Authority (CA) to generate the device certificate based on the public key. The private key and the device certificate establish the MCU device identity, which is unique to each MCU. The device's private key is stored in data flash in wrapped format, and the device certificate is stored in a secure region of code flash.

The RSIP security engines on RA Family MCUs support the creation of a unique cryptographic identity. This identity can then be leveraged for other security solutions, such as secure internet connectivity and secure firmware updates. Creating the device identity on the chip itself has a significant security advantage in that the private component of the identity is never exposed outside the MCU, so it cannot be compromised by an information leak.

The specific types of cryptographic identities that can be created vary depending on the specific RSIP security engine that is integrated into the MCU. Both Protected Mode and Compatibility Mode are supported.

With this release, application projects demonstrate generating a cryptographic unique identity using Protected Mode and Compatibility Mode:

- RSA 4K key pair, RSIP-E51A (RA8M1), Protected Mode.
- ECC secp256r1 key pair, RSIP-E31A (RA4C1), Compatibility Mode.
- ECC secp256r1 key pair, RSIP-E11A (RA4L1), Protected Mode.

### Required Resources

To build and run the RA Family Device Identity Application example, you need the following resources:

#### Development tools and software

- e<sup>2</sup> studio IDE v2025-04.1
- RA Family Flexible Software Package (FSP) v6.0.0
- SEGGER J-link® USB driver V8.44a

The above three software components: the FSP, J-Link USB drivers, and e<sup>2</sup> studio are bundled in a downloadable platform installer available on the FSP webpage at [renesas.com/ra/fsp](https://www.renesas.com/ra/fsp)

- Visual Studio 2022 Community Version (<https://visualstudio.microsoft.com/downloads/>)  
It is only needed if the user wants to customize the PC program. It is not needed to run the application project.

#### Hardware

- EK-RA8M1 with two Micro-USB cables, Evaluation Kit for RA8M1 MCU Group (<http://www.renesas.com/ra/ek-ra8m1>)
- EK-RA4C1, with a Type-C USB cable, Evaluation Kit for RA4C1 MCU Group (<http://www.renesas.com/ra/ek-ra4c1>)
- EK-RA4L1, with two Type-C USB cables, Evaluation Kit for RA4L1 MCU Group (<http://www.renesas.com/ra/ek-ra4l1>)
- Test PC running Windows® 10/11 OS

## Prerequisites and Intended Audience

This application note assumes you have some experience with the Renesas e<sup>2</sup> studio IDE and RA Family Flexible Software Package (FSP). Before you perform the procedures in this application note, follow the procedure in the *FSP User Manual* to build and run the Blinky project. Doing so enables you to become familiar with the e<sup>2</sup> studio and the FSP and validates that the debug connection to your board functions properly. In addition, a prerequisite reading is chapter 3 of application note [Security Design with Arm® TrustZone® using Cortex-M33](#) or [Security Design using Arm TrustZone - Cortex M85](#). The goal of reading this chapter is to understand the two different development models provided by Renesas Tooling for TrustZone® support. Furthermore, this application note assumes that you have some knowledge of cryptography and RA Family RSIP features.

The intended audience are users who want to develop applications with RSIP modules using Renesas RA Family MCUs with RSIP support.

## Contents

|                                                                           |    |
|---------------------------------------------------------------------------|----|
| 1. Introduction to IoT Security .....                                     | 4  |
| 1.1 Overview .....                                                        | 4  |
| 1.2 Importance of Device Identity in an IoT Ecosystem .....               | 5  |
| 1.3 Cryptographic Unique Identity and Identity Generation Use Cases ..... | 5  |
| 2. RA Family MCU Hardware Security Features .....                         | 6  |
| 2.1 Arm® TrustZone® .....                                                 | 6  |
| 2.2 Renesas Secure IP .....                                               | 6  |
| 2.3 Flash Block Protection .....                                          | 7  |
| 3. Overview of Key Generation in RA Family MCUs .....                     | 7  |
| 3.1 Key Wrapping .....                                                    | 7  |
| 3.2 Key Generation in the Device .....                                    | 8  |
| 4. Device Identity Design Overview .....                                  | 8  |
| 5. Device Identity Application Example .....                              | 9  |
| 5.1 Overview .....                                                        | 9  |
| 5.2 Software Architecture Overview .....                                  | 9  |
| 5.3 Operational Overview .....                                            | 12 |
| 5.4 Securely Storing Device Identity .....                                | 13 |
| 5.5 Non-Secure Callable APIs .....                                        | 14 |
| 6. Running the Device Identity Application Example .....                  | 14 |
| 6.1 Importing and Building the Embedded Projects .....                    | 14 |
| 6.2 Powering up the Board .....                                           | 18 |
| 6.3 Downloading and Verifying the Demonstration .....                     | 18 |
| 6.3.1 Download Flat project and run PC application .....                  | 18 |
| 6.3.2 Download TrustZone project .....                                    | 20 |
| 6.4 Customizing the Application Project .....                             | 26 |
| 6.4.1 Customize the PC Application .....                                  | 26 |
| 7. References .....                                                       | 27 |
| 8. Known Issues and Limitations .....                                     | 27 |
| 9. Appendix .....                                                         | 27 |
| 9.1 Glossary .....                                                        | 27 |
| 10. Website and Support .....                                             | 28 |
| Revision History .....                                                    | 29 |

## 1. Introduction to IoT Security

This section provides an overview of IoT Security (in general) and covers the different aspects of the security features offered by RA Family MCUs.

### 1.1 Overview

A typical infrastructure for an operational IoT (Internet of Things) environment consists of the following:

- IoT Devices
- Cloud Server
- Device Management services
- Certificate Authority (CA)

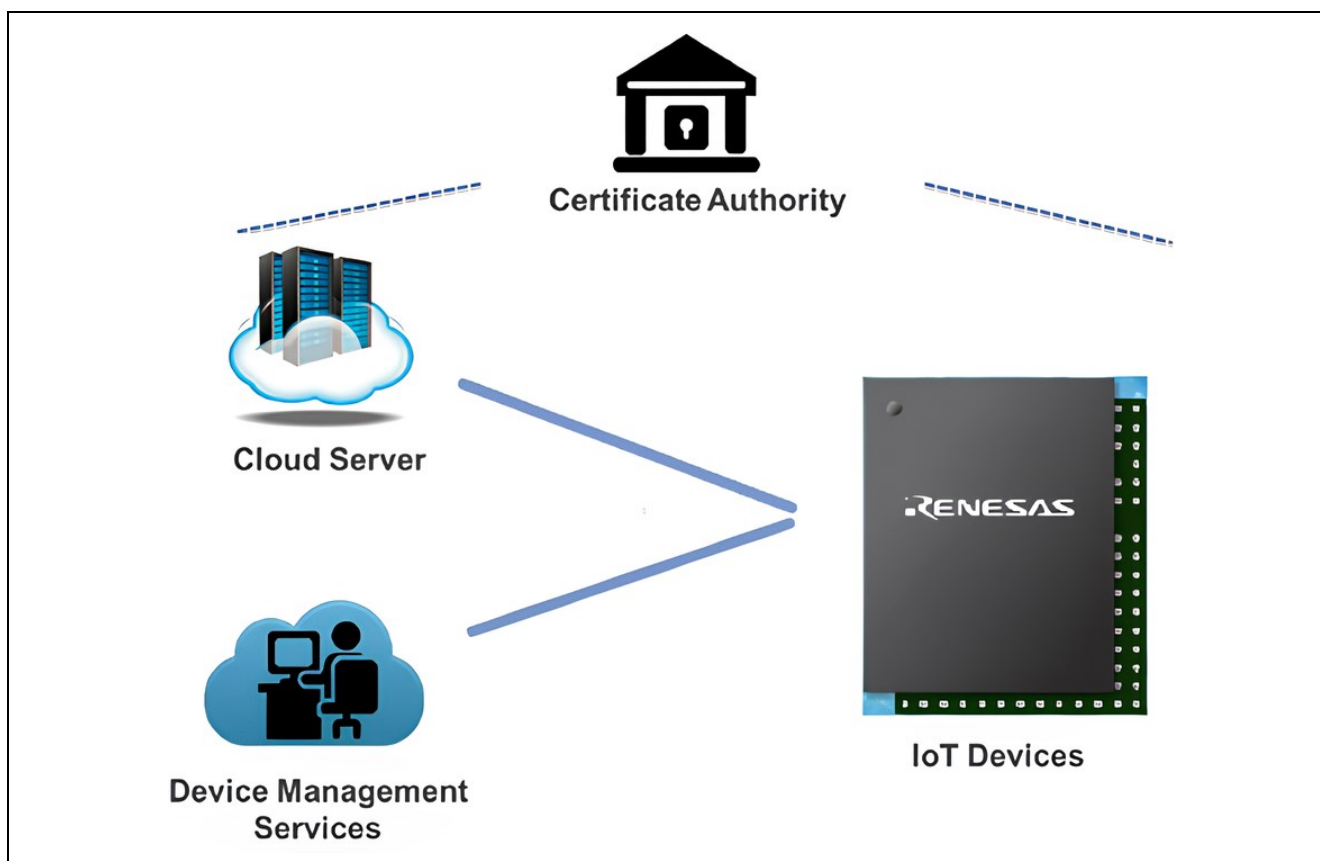


Figure 1. IoT Environment Overview

#### IoT Devices

An IoT Device is a piece of equipment with the mandatory capabilities of communication and the optional capabilities of sensing, actuation, data capture, data storage, and data processing. IoT devices can be in a secure or non-secure location and without any security safeguard, but all are prone to attack.

#### Cloud Server

A Cloud Server is a network server for cloud services providing everyday services and access to those devices. It is typically located in a highly secure and controlled data center.

## Device Management Services

Device Management services offer a comprehensive suite of capabilities associated with managing and configuring IoT devices. The management service enables IoT customers of any size to have complete control over their devices and data. This includes (but is not limited to):

- Application Security
  - Device management solution that ensures your IoT devices' reliability, longevity, and interoperability in addition to providing trusted and useful data. The security entity used in the application security design includes Cloud Server keys/certificates.
- Device Management Security
  - Manage keys/certificates which identifies each unique IoT Device, which may involve Device key and certificate creation, update, and revocation.
  - Initial firmware deployment and subsequent firmware updates. Firmware contains a signature verifying its authenticity and may be encrypted.

## Certificate Authority (CA)

An authorized and trusted entity that issues certificates as a service is commonly referred to as CA. Certificates are used to authenticate public keys and thus the devices that contain those keys. The process by which a certificate is generated for a key is a well-defined process that is part of your security scheme. A Certificate Authority can be public or private. If your devices are managed in a tight ecosystem (for example, devices for industrial settings), the CA will likely be private. If your devices are distributed through a consumer channel where the services and hardware are likely to be provided by different vendors (for example, surveillance cameras, thermostats, home security systems, and so forth), the CA will likely be a public CA.

## 1.2 Importance of Device Identity in an IoT Ecosystem

With the establishment of a strong device identity, IoT devices can be uniquely identified and authenticated when they are connected to ensure secure and encrypted communication between other devices, services, and users.

Strong IoT security can be achieved by providing the following foundations typically agreed upon by the industry. A well-designed Device Identity is the core to these foundations:

- Trust

When a device connects to the network, it must authenticate and establish trust between other devices, services, and users. Once trust is established, devices, users, and services can securely communicate and exchange encrypted data and information.
- Privacy

As more IoT devices connect, more data is generated, collected, and shared. This data can include personal, sensitive, and financial information that must be kept private and secured – often under regulatory compliance. A device identity can provide authentication and identification when the IoT devices are connected to one another.
- Integrity

Device integrity applies to both the devices and data being transmitted within the IoT ecosystem. The integrity of a device starts with proving it is what it says it is. With a strong unique device identity, it can be ensured that the devices are legitimate – reducing counterfeit products and protecting a company's brand. Data integrity is an often-overlooked requirement, but connected devices and systems rely on the authenticity and reliability of the information being transmitted.

## 1.3 Cryptographic Unique Identity and Identity Generation Use Cases

A cryptographic unique identity is a device-specific identity typically made up of a private key and a certificate containing the corresponding public key. It enables secure communication, authentication, and data integrity in embedded and IoT systems.

There are two common approaches to generating this identity:

**Key Injection:** Keys are generated offline and injected into the device.

- **Advantages:** Fast production, supports parallel certificate generation.
- **Disadvantages:** Risk of private key exposure, requires Define HSM integration, costly equipment, and specialized infrastructure. Key duplication must also be avoided.

Renesas supports both secure and plaintext key injection with example projects available.

**On-Chip Key Generation:** The key pair is generated securely within the MCU.

- **Advantages:** Private key never leaves the device, eliminating key exposure risks and third-party dependency.
- **Disadvantages:** Slower process; certificate generation must be coordinated after key creation.

Each method serves different use cases—key injection fits high-volume manufacturing, while on-chip generation is ideal for maximum security requirements.

## 2. RA Family MCU Hardware Security Features

RA Family TrustZone®-enabled MCUs enable hardware root-of-trust mechanisms by providing the ability to protect memory blocks. Using this capability, the protected memory can be accessed only by firmware located in memory regions designated as a secure memory region. These capabilities are provided by the Arm® TrustZone® and Renesas Block Protection hardware feature. The contents of program memory can be locked from future erase/write events using the Renesas Block Protection. Memory protection features offered by RA Family MCU devices can be used for storing the secure boot code and device certificate/keys amongst other sensitive data which are vital for device identity application.

### 2.1 Arm® TrustZone®

Renesas RA TrustZone®-based MCUs use different attribution hardware depending on the core:

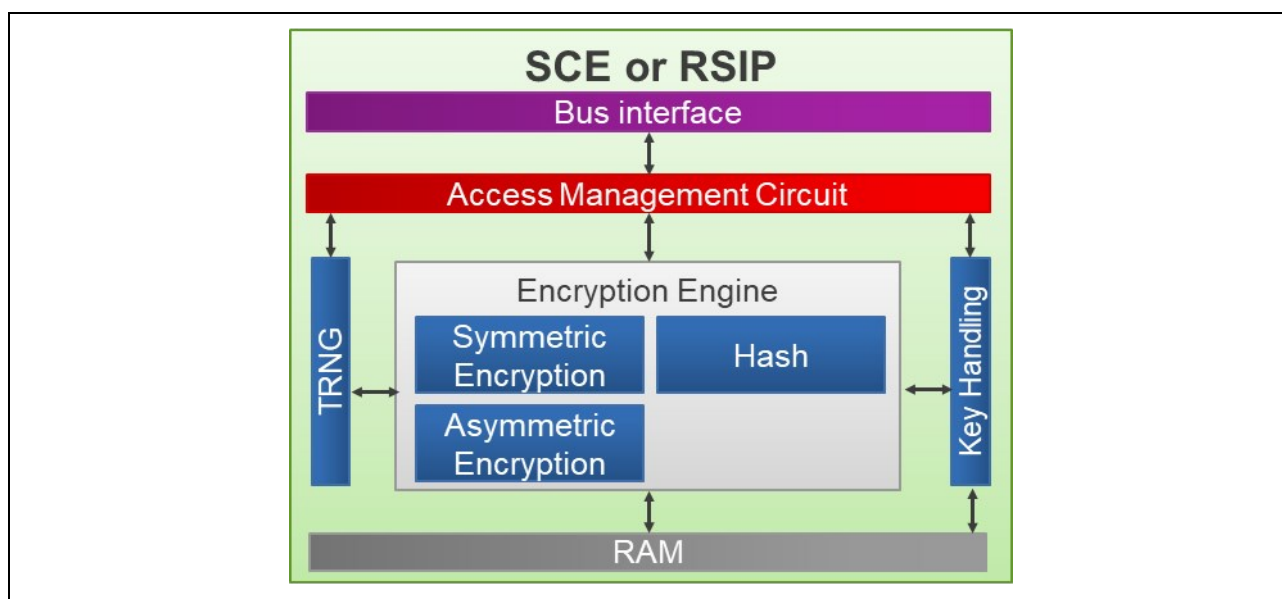
- RA4/RA6 depends on an Implementation Defined Attribution Unit (IDAU) to enforce TrustZone boundaries. In these MCUs, the IDAU settings will define Secure, Non-Secure Callable (NSC), and Non-Secure regions for on-chip flash and SRAM, which are computed at build time.
- RA8 implements the TrustZone security extension using both a fixed, hardware defined IDAU (and its Master Security Attribution Unit (MSAU) for bus masters) and a software configurable Security Attribution Unit (SAU). The IDAU and SAU are queried in parallel with security label (Secure, NSC, Non-Secure) is applied, ensuring that secure memory remains protected even if one attribution unit is misconfigured.

Both approaches initialize their security attribution units before any user code runs, preventing boot-time attacks and guaranteeing the secure data and code remain isolated from the non-secure.

### 2.2 Renesas Secure IP

The Renesas Secure IP (RSIP) is an RA Family MCU hardware peripheral that provides several security features, including NIST certified algorithms and support for cryptographic primitives. In addition, MCUs containing RSIP have a Hardware Unique Key (HUK), unique to every individual MCU, which is stored wrapped by the Hardware Root Key and MCU's unique ID. The HUK is never exposed outside RSIP and will not work on any other MCU. The HUK is used for secure key storage only; it cannot be used for MCU device identity.

Many RA Family TrustZone®-enabled MCUs support asymmetric cryptography as well as symmetric cryptography. Following is a diagram of the RSIP features offered by these MCUs. When keys are generated on the MCU, they are always wrapped by the HUK and will only work on the MCU that generated the keys.



**Figure 2. Renesas Secure IP Features**

The RSIP engine provided by RA Family devices is used by this application project in the following areas:

- Generate key pairs (plaintext public and wrapped private key).
- Sign the challenge string using the private key.

### 2.3 Flash Block Protection

The RA Family TrustZone®-enabled MCUs can provide temporary and permanent flash block locking for code and data flash Programming and Erasing (P/E) mode entry. The e² studio IDE provides configuration options for a user to selectively prevent the erasure and programming of the intended flash block.

This flash block protection is not directly used in this application project. However, for use cases where the device certificate needs to be permanently locked down in the manufacturing stage, this can be used to lock the corresponding flash block at run time to protect the secure code flash region from accidental erasure and reprogramming.

## 3. Overview of Key Generation in RA Family MCUs

### 3.1 Key Wrapping

Device keys generated inside the RA Family TrustZone®-enabled MCU using the RSIP hardware module can only be generated as wrapped keys. Plaintext keys can be injected from external sources and stored as wrapped keys in the MCU.

A wrapped key is a key that has been encrypted by the RSIP, using a method that involves use of the MCU's HUK. Because this method requires the MCU's HUK to unwrap the key, the key can only be unwrapped by the same MCU that wrapped it. Therefore, key wrapping on RA Family MCUs is considered secure, as a wrapped key can only be used on the RA Family MCU on which it was generated, and it cannot be used outside of that MCU. As a result, the scalability of an attack can be substantially reduced.

Wrapped keys provide the following advantages:

- A wrapped key can only be used on the RA Family MCU on which it was generated.
- It cannot be moved to another RA Family MCU. If moved to another RA Family device, the original key cannot be recovered from the wrapped key and cannot be used with RSIP.



### 3.2 Key Generation in the Device

Key generation in the device is the common use case where the device-specific key is natively generated inside the RA Family MCU using the RSIP module. To generate the device key using the RA Family Flexible Software Package (FSP), there are two operation modes available: Compatibility Mode and Protected Mode.

#### Compatibility Mode:

- Purpose: Designed to support integration with existing systems or legacy applications and ensure backward compatibility.
- Characteristics: Allows plaintext key usage, often facilitating easier integration into existing infrastructure. Wrapped keys can still be generated by the security engine.
- Disadvantages: The potential for plaintext key usage in the system can lower the overall security of the application. Extra care is necessary when architecting the application.

#### Protected Mode:

- Purpose: Aims to provide optimal security for key generation and cryptographic operations.
- Characteristics: No plaintext keys can be input to RSIP, ensuring that private keys are never exposed outside the security engine, thereby mitigating the risk of attacks or data leaks. Generated keys are provided in wrapped format.
- Disadvantages: Some existing infrastructure assumes plaintext keys.

## 4. Device Identity Design Overview

This section explains how RA hardware and software features are integrated to create a unique device identity for each device.

### Key Generation

The first step in creating a device identity is key generation. The keys can be either generated inside the RA Family MCU or they can be generated outside in a secure facility and injected into the RA device. Each methodology has its pros and cons. The decision must be made based on the customer use case.

### Certificate Authority (CA)

Once the device keys are generated/injected, we need an entity that takes the public key from the key pair and issues an associated digital certificate. A CA can be either public or private CA located in the cloud or in an on-premises CA (local CA), which would typically be hosted on a secure server.

### Securing the Device Identity

Once the device identity is created and programmed on the RA Family device, it must be securely stored to prevent theft or tampering. Depending on what kind of end application the certificate is for, it may provide access to a controlled environment. If the device private key is stolen, fraudulent access could occur. Therefore, the private key must be securely stored.

Secure storage of the Device Identity can be achieved by using the Arm® TrustZone® and Renesas Block Protection features offered by the RA Family MCU. These features configure a portion of internal program memory as secure partitions for both code and data.

- Code and data in secure partitions are protected from access by non-secure code, non-secure peripherals, and non-secure bus master channels (DMAC and DTC) by Arm® TrustZone® technology. A Secure Fault will be triggered for any violation.
- In addition, Renesas's TrustZone-enabled MCUs provide Device Lifecycle Management capabilities that can be used to disable debugger and serial programming access to secure regions.
- Access to secure region services is only possible via exposed non-secure callable API veneers, located in the Non-secure Callable region.
- The device certificate can be stored in the secure partition, which cannot be accessed or modified directly by any non-secure code running on the RA Family MCU.



## 5. Device Identity Application Example

### 5.1 Overview

The example application project accompanying this document demonstrates natively generating and storing the device identity information using the on-chip RSIP modules available with the Renesas RA Family device. For demonstration purposes, this application uses a local Certificate Authority (CA) running on a Windows® PC to generate a signing key and root CA that will be used to sign the device certificate. USB-PCDC or SCI/UART is used as the primary communication interface between the RA kit and the host console application running on the Windows PC.

This application project demonstrates three use cases, each highlighting the Device ID feature with a specific combination of cryptographic algorithm, RSIP variant, and operation mode:

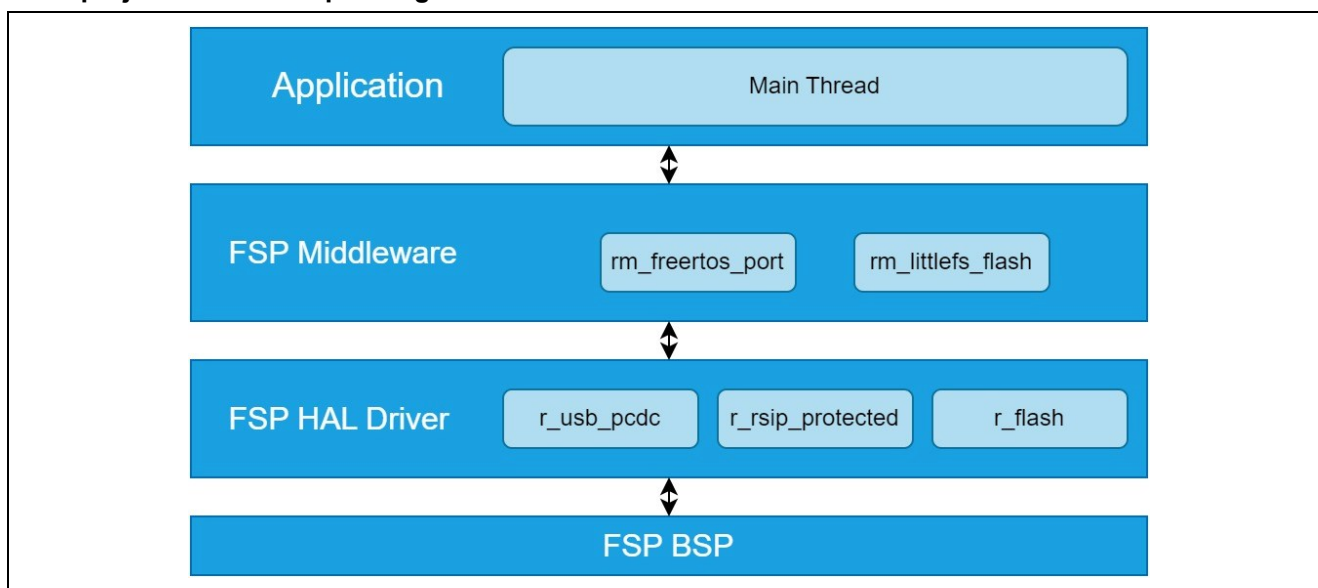
- RSA 4096-bit key pair using RSIP-E51A on RA8M1, operating in Protected Mode.
- ECC secp256r1 key pair using RSIP-E31A on RA4C1, operating in Compatibility Mode.
- ECC secp256r1 key pair using RSIP-E11A on RA4L1, operating in Protected Mode.

These use cases illustrate the flexibility of RSIP in supporting both compatibility-focused and security-centric applications within the RA Family MCU ecosystem.

### 5.2 Software Architecture Overview

The following figure shows the overall software architecture of the RA Family device identity application project in each mode.

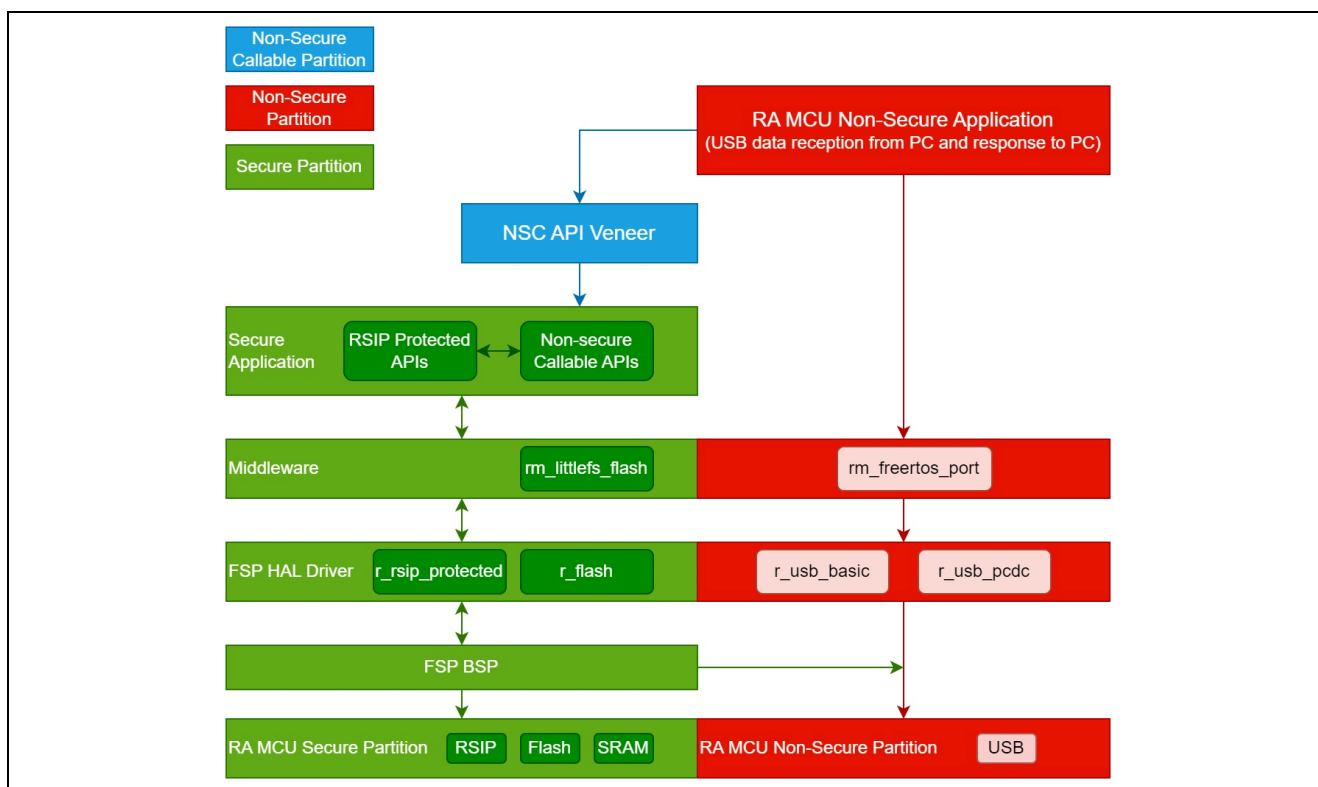
**Flat project on RA4L1 operating in Protected Mode:**



**Figure 3. RA Device Identity Application Software Architecture in Protected Mode (EK-RA4L1)**

The major FSP software components of this application are:

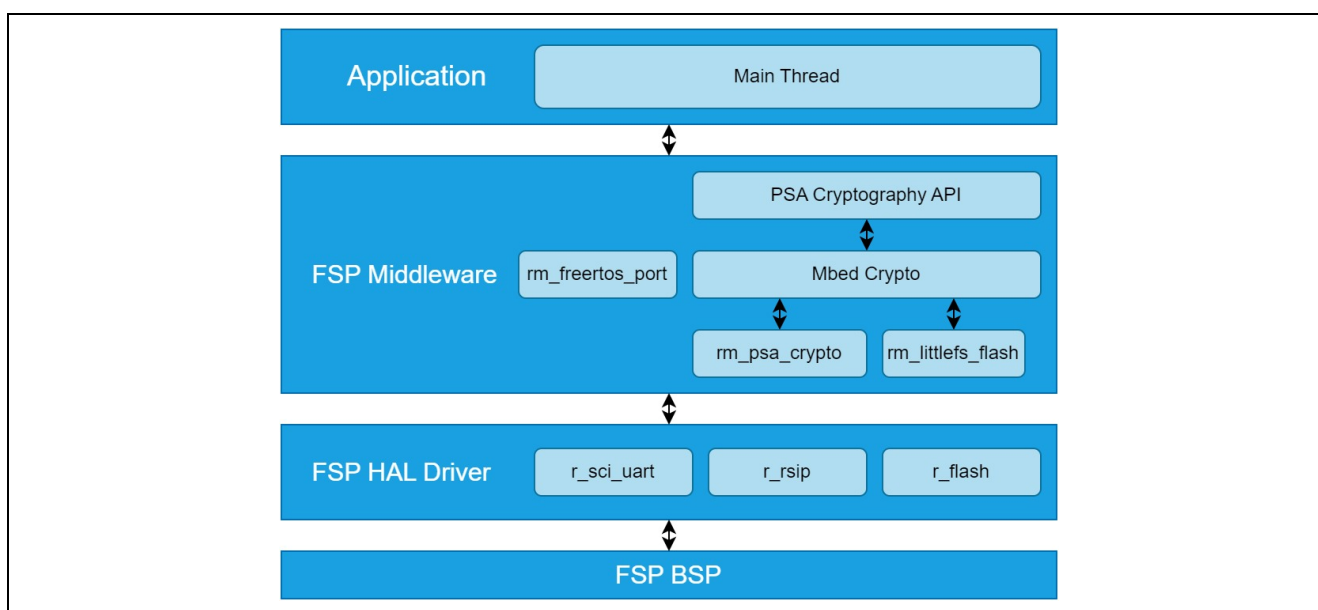
- `r_rsip_protected`: for cryptographic operation.
- `rm_littlefs_flash` and `r_flash`: for ECC key pair (`rm_littlefs`) and device identity (`r_flash`) storage.
- `r_usb_pcdc` and `r_usb_basic`: for communication between PC and MCU.
- `rm_freertos_port`: multithreading framework for scalability.
- The application contains the following thread:
  - Main Thread

**TrustZone project on RA8M1 operating in Protected Mode:**

**Figure 4. RA Device Identity TrustZone-based Application Software Architecture in Protected Mode (EK-RA8M1)**

The major FSP software components of this application are:

- r\_rsip\_protected: for cryptographic operation.
- rm\_littlefs\_flash and r\_flash: for RSA key pair (rm\_littlefs) and device identity (r\_flash) storage.
- r\_usb\_pcdc and r\_usb\_basic: for communication between PC and MCU.
- rm\_freertos\_port: multithreading framework for scalability.
- The application contains the following thread:
  - Main Thread

**Flat project on RA4C1 operating in Compatibility Mode:**

**Figure 5. RA Device Identity Application Software Architecture in Compatibility Mode (RA4C1)**

The major FSP software components of this application are:

- `rm_psa_crypto` and `r_rsip`: for cryptographic operation.
- `rm_littlefs_flash` and `r_flash`: for RSA key pair (`rm_littlefs`) and device identity (`r_flash`) storage.
- `r_sci_uart`: for communication between PC and MCU.
- `rm_freertos_port`: multithreading framework for scalability.
- The application contains the following thread:
  - Main Thread

### Main Thread

This is the main control thread which handles the following functions:

1. Incoming/outgoing USB/UART data from and to PC.
2. Decoding the command and calling the appropriate command handler functions, which in turn handle the corresponding command functionalities.

The following commands are handled by the Main Thread:

- `WRAPPED_KEY_REQUEST`
- `WRAPPED_KEY_CHALLENGE_RESP`
- `WRAPPED_KEY_CERT_PROGRAM`

### WRAPPED\_KEY\_REQUEST

This command is the first command issued from the local CA over USB/UART to request the public key of the device. It is handled by the following API function: `handleWrappedKeyCreation()`.

This function handles the key generation using FSP Crypto modules. This application supports ECC/RSA Key pair generation. Once the key pair is generated, the plaintext public key is sent to the host application to be used for the device certificate generation.

The wrapped private key is stored internally in the data flash and will later be used for signing the challenge response.

### WRAPPED\_KEY\_CHALLENGE\_RESP

This command is issued after the PC-hosted local CA has received the device's public key. It is handled by the following API function: `handleCertChallengeResp()`.

The intention of this challenge request is to allow the target to prove its ownership of the device private key for the corresponding public key being certified.

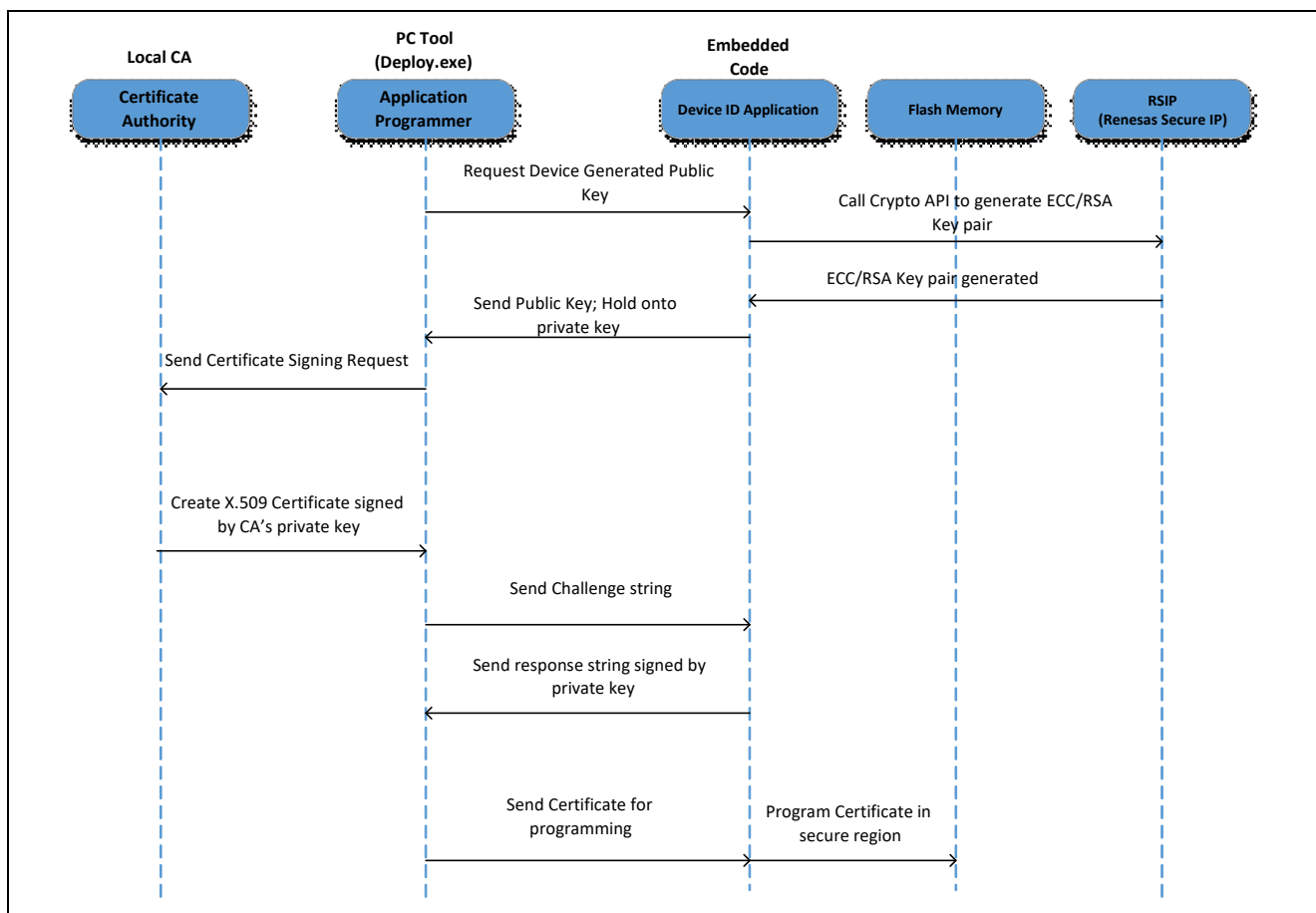
This function handles the challenge response request sent by the host application. Once it receives the request, it signs the string sent as part of the request using the private key generated as part of `WRAPPED_KEY_REQUEST` command. The signed string is sent back to the host application for verification. Once the host application receives the signed string, it verifies the signature using the device public key extracted from the device certificate. When the signature validation is successful, the host application will send the device certificate to the device to be stored securely using Arm® TrustZone® and Block Protection hardware feature.

### WRAPPED\_KEY\_CERT\_PROGRAM

This command is issued from the PC after a successful challenge and response process between the PC and MCU. It is handled by the following API function: `handleCertProgram()`.

This function handles programming the device certificate received from the host application into the secure region of the internal code flash.

### 5.3 Operational Overview



**Figure 6. Operational Overview**

This application project consists of two software projects:

- Embedded project running on evaluation kits.
- Host application running on Windows® 10 PC.

When power is supplied to the evaluation kits, the firmware initializes the MCU and the underlying USB PCDC or SCI/UART stack that is used for communication with the host application running on the Windows PC. At end of initialization, the firmware waits for the USB device connect event. Once the user connects the kit to the Windows PC through a USB cable, the USB enumeration process occurs, and the USB PCDC or SCI/UART instance is created. At this stage, the firmware is waiting for the commands from the host application.

When the user runs the host utility on the Windows PC, it scans the available COM ports and opens the port to which the evaluation kit is connected. Once the COM port is opened successfully, it generates a signing key and root CA certificate that will later be used to sign the device certificate. Now, the host application generates the WRAPPED\_KEY\_REQUEST command and sends it to the kit. On receiving this request, the embedded code running on the target kit generates device key pairs and sends out the public key to the host application. On the host application, it receives the public key from the device and generates a device certificate (signed by CA's signing key).

Before issuing the device certificate, the host application issues a challenge string to the device to prove that the device owns the private key. The embedded software, on receiving the challenge string, signs it using its private key and sends it back to the host application. The host application validates the signature using the device public key and if the validation is successful, the device certificate will be sent to the kit to be securely stored using the Arm®TrustZone® and Renesas Block Protection hardware feature.

## 5.4 Securely Storing Device Identity

The two pieces of information used to establish device identity and created as part of this application are as follows:

- Wrapped ECC/RSA private key
- Device certificate

These two pieces of information need to be securely stored inside the RA Family MCU using the Arm® TrustZone® and Renesas Block Protection hardware features to avoid being accessed and modified. The private key generated as part of this application is already wrapped, so this example skips the step to securely store the device private key. However, in some cases, users may prefer to also store the wrapped key in a secure location to avoid it being misused in the device. This can be done using the same steps used to store the device certificate.

The following is the memory map of the current device identity application projects.

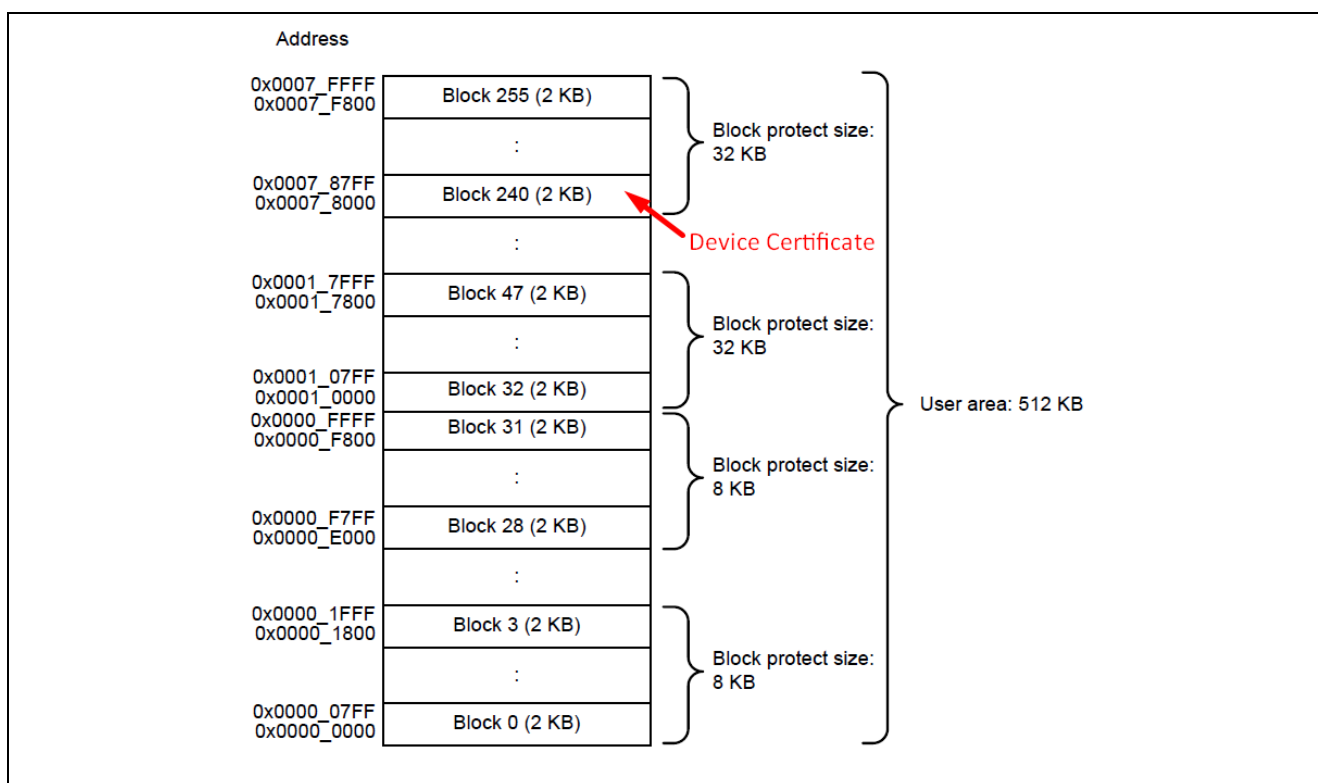


Figure 7. Memory Map used in the Application Project of RA4L1 and RA4C1

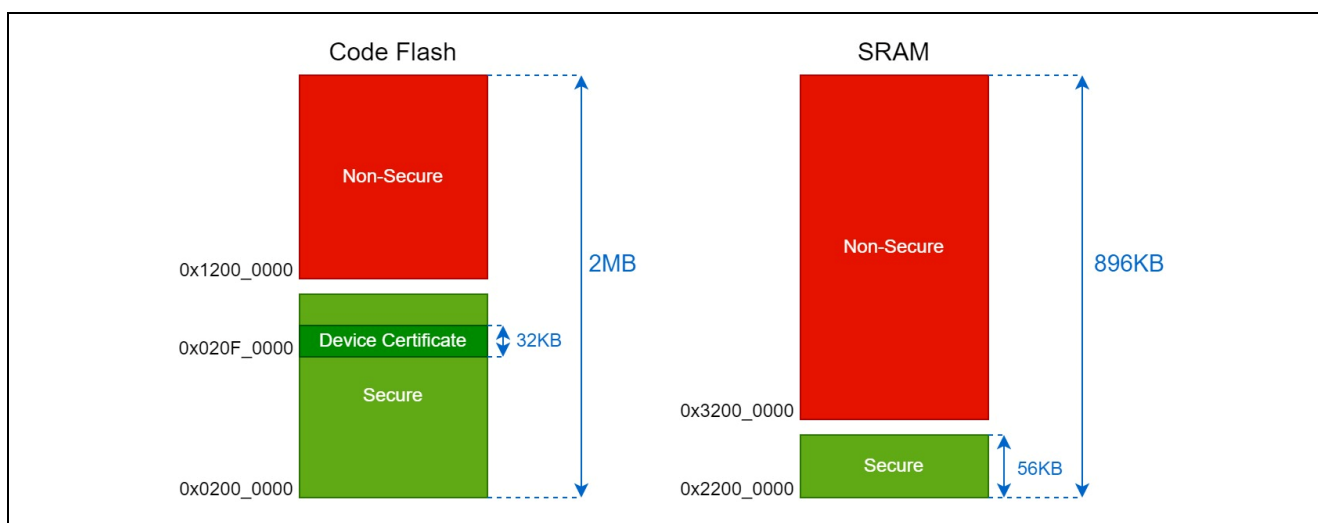


Figure 8. Memory Map used in the Application Project of RA8M1

Figure 8 show the memory layout of the code flash in the application project of RA8M1.

The MCU operates in secure mode when executing code in the secure region or non-secure callable region. The Device Certificate is programmed in the flash block from 960 kB to 992 kB (start from 0x020F0000). The MCU operates in non-secure mode when executing code in the non-secure region. When executing in non-secure mode, the system cannot access information in the secure region.

The security attributes are set up such that the LittleFS middleware, RSIP Protected Mode module are located in the TrustZone® secure partition as shown in Figure 4. Therefore, the RSIP Protected Mode module code which accesses the keys are protected from direct access from the non-secure partition. In addition, the FSP flash driver module (r\_flash\_hp) is mapped to the bottom portion of the secure SRAM (0x22000000). This protects the non-secure code from copying the flash routines for illegal flash operations.

## 5.5 Non-Secure Callable APIs

There are two non-secure callable APIs that the non-secure code can call and activate the corresponding secure partition operations.

- The first non-secure callable receives one byte from the non-secure region and parses the received byte. Refer to `\embeddedCommon/src/framedProtocolTarget.c` for the definition of this function.

```
/* Handle a received byte of serial data */
BSP_CMSE_NONSECURE_ENTRY void fpReceiveByte(const uint8_t byte);
```

- The second non-secure callable returns secure region responses to non-secure region.

```
/* Provide response to Non-secure region */
BSP_CMSE_NONSECURE_ENTRY void share_with_ns(uint8_t *pBuffer, uint16_t
*numBytes);
```

## 6. Running the Device Identity Application Example

### 6.1 Importing and Building the Embedded Projects

There are 3 projects in `ra_device_id_using_rsip` package:

Flat projects:

- `RA4C1_Flat_Compatibility`: demonstrates for ECC secp256r1 key pair, RSIP-E31A, Compatibility Mode on EK-RA4C1.
- `RA4L1_Flat_Protected`: demonstrates for ECC secp256r1 key pair, RSIP-E11A, Protected Mode on EK-RA4L1.

TrustZone project:

- `RA8M1_TZ_Protected`: demonstrates for RSA 4K key pair, RSIP-E51A, Protected Mode on EK-RA8M1.

The embedded projects are included in the folder

`ra_device_id_using_rsip\<Project_name>\embedded`. The following instructions will show the user how to import these projects in their e<sup>2</sup> studio workspace.

First, extract `ra_device_id_using_rsip.zip`

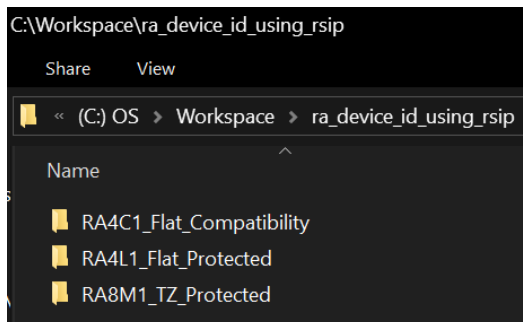
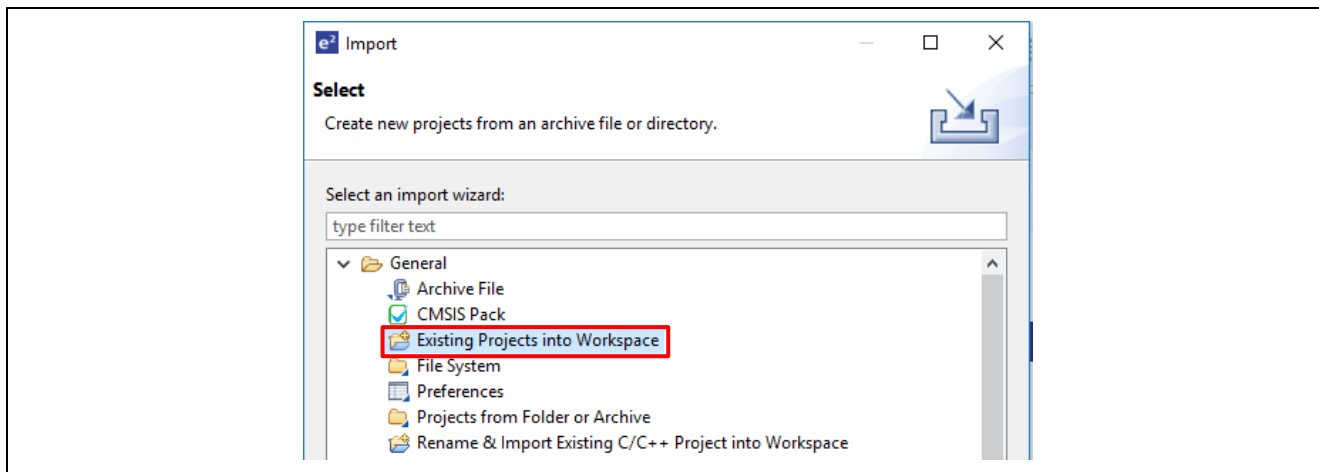


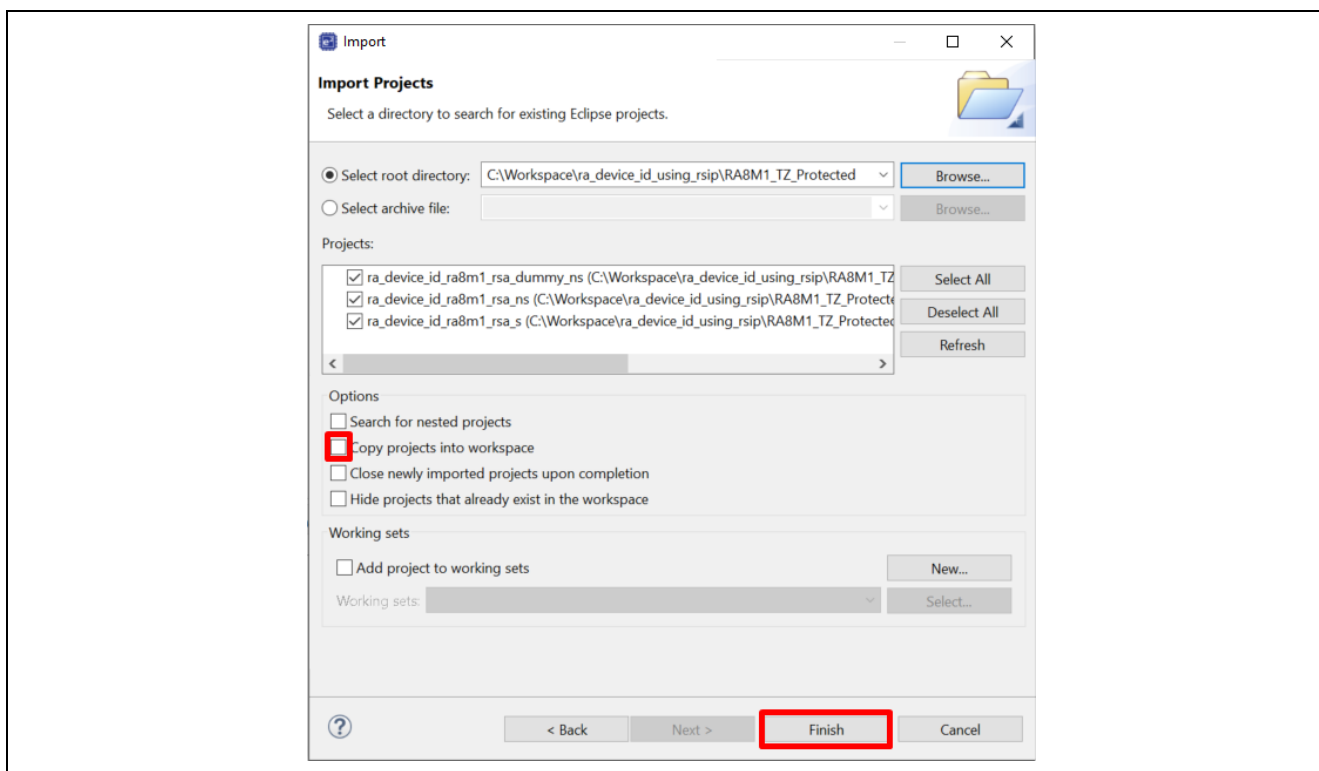
Figure 9. Project folder after extract

In the e<sup>2</sup> studio IDE, select **File -> Import... -> Existing Projects into Workspace**.



**Figure 10. Choose to import “Existing Projects”**

Select to import the project user would run as an example in following figure. **DO NOT CHECK** the “Copy projects into workspace” box.



**Figure 11. Selection for Importing the Embedded Project.**

For Flat projects of RA4C1 and RA4L1, there is one project under the `\embedded\EK-RA4C1` or `\embedded\EK-RA4L1`. After importing the project, open the RA configurator, click **Generate Project Content**, and then compile the project. It should compile without errors.

For TrustZone project, there are three projects under the `\embedded\EK-RA8M1` folder. In a split project development model, the secure project `ra_device_id_ra8m1_rsa_s` and the non-secure dummy project `ra_device_id_ra8m1_rsa_dummy_ns` are developed by the secure project development team, which is a different development team from the team that develops the non-secure project `ra_device_id_ra8m1_rsa_ns`. In addition, the secure project and a dummy non-secure project (whose linkage to the secure project is established with the Combined Development model) must be downloaded first before changing the MCU device lifecycle to OEM\_PL1.



Follow the steps below to compile and download the application to the MCU. For more information on the definition of Split Project Development and Combined Project Development model, reference application project or [Security Design using Arm TrustZone - Cortex M85](#).

After importing the project, follow the steps below in sequence to compile the projects:

1. Expand project `ra_device_id_ra8m1_rsa_s` and double click `configuration.xml`. Once the RA configurator is opened, click **Generate Project Content** and then compile the project. It takes 1-2 minutes to compile the project. There are warnings from third-party software components.
2. Expand project `ra_device_id_ra8m1_rsa_dummy_ns` and double click `configuration.xml`. Once the RA configurator is opened, click **Generate Project Content** and then compile the project.
3. For the project `ra_device_id_ra8m1_rsa_ns`, follow the two steps below to compile the project:
  - A. Update the matching secure bundle based on current project folder location. The secure bundle is linked as an absolute path. Right click on `ra_device_id_ra8m1_rsa_ns` project and select **Properties** as shown in Figure 12.

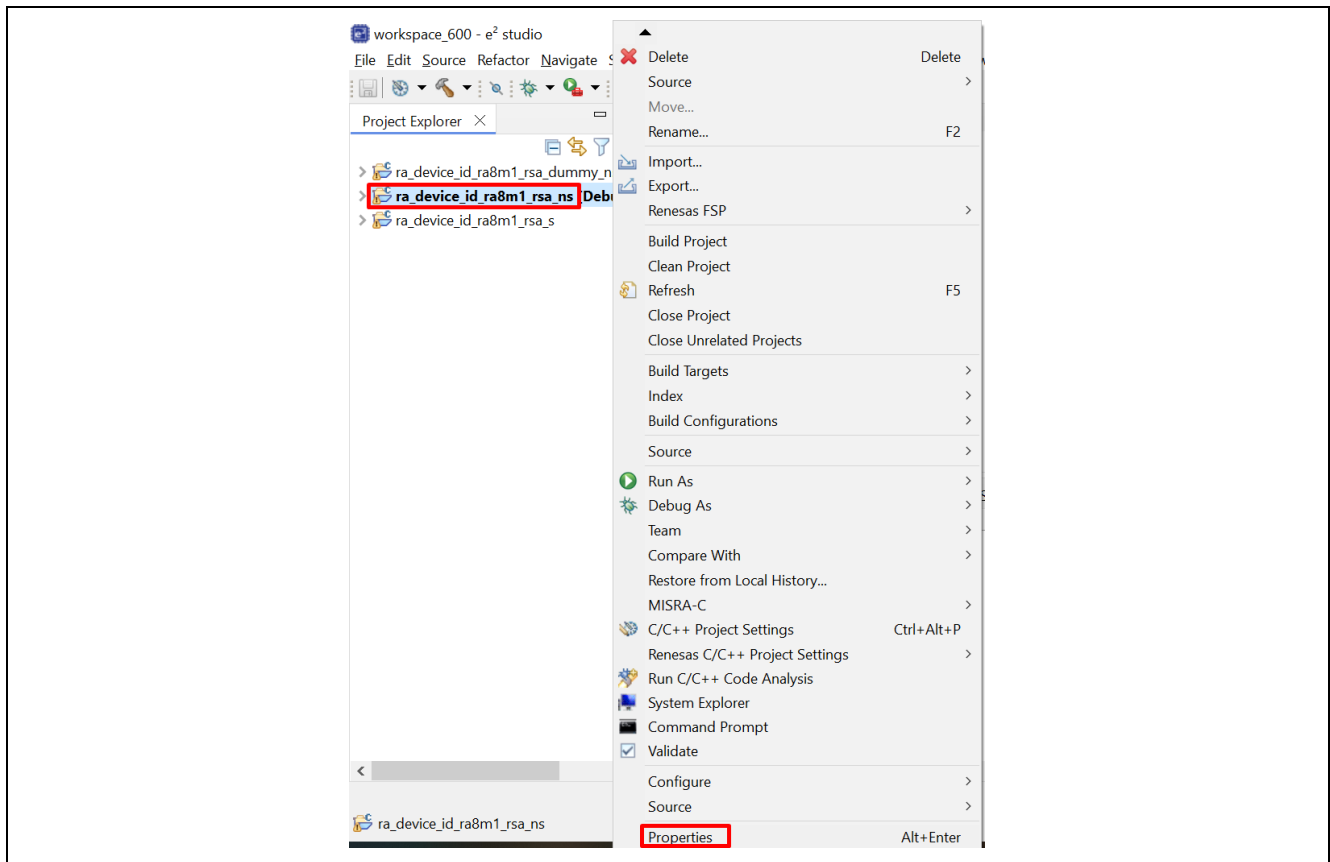
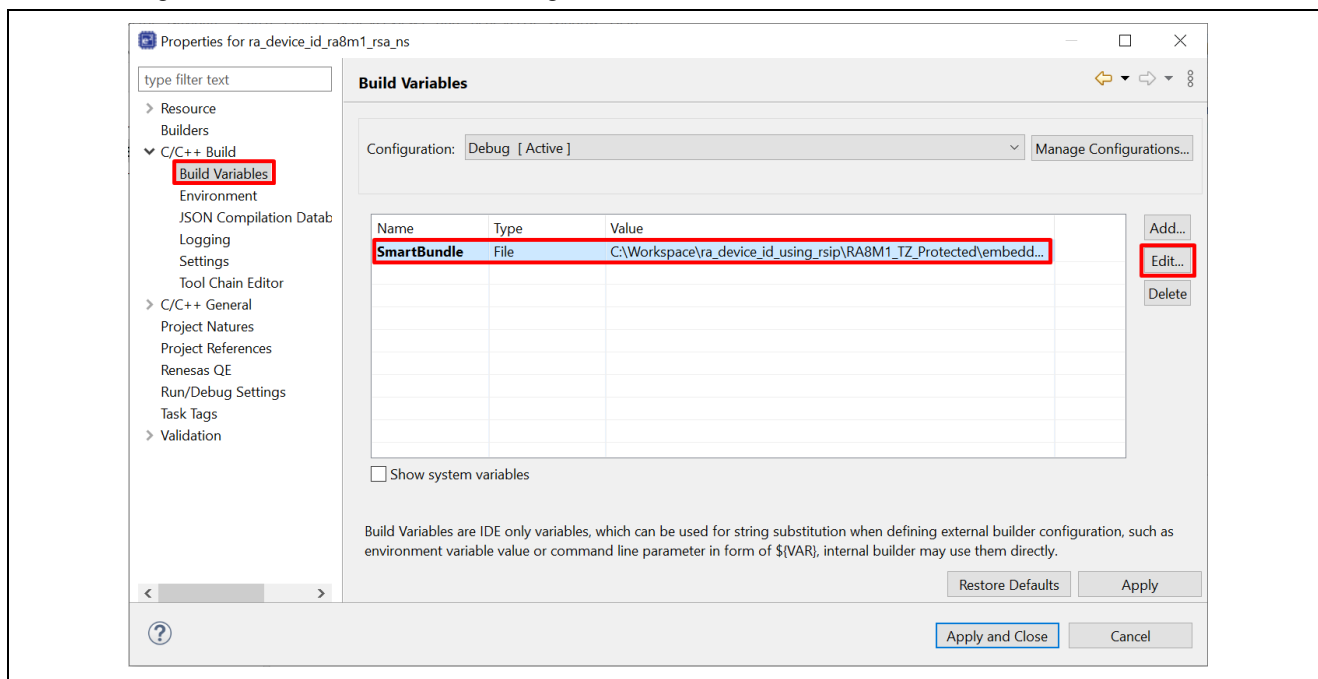


Figure 12. Open the Properties Menu

Navigate to the **Build Variables** configuration for **SmartBundle**.

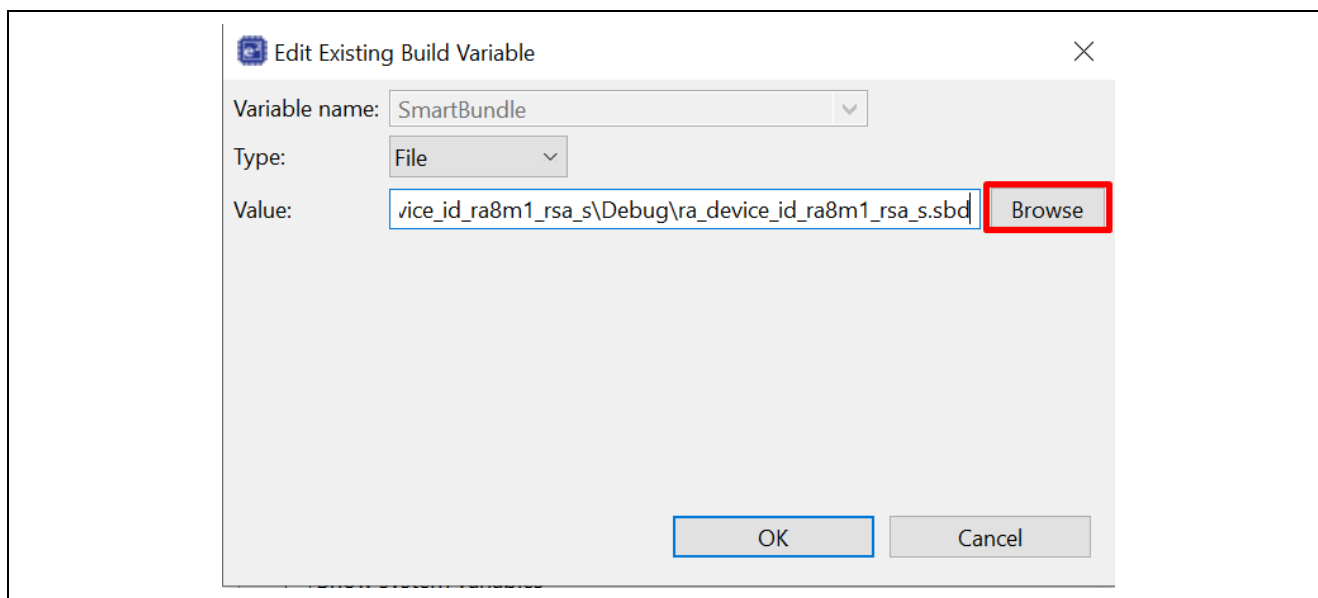


**Figure 13. Edit the Build Variable “SmartBundle”**

Next, click **Browse** and select

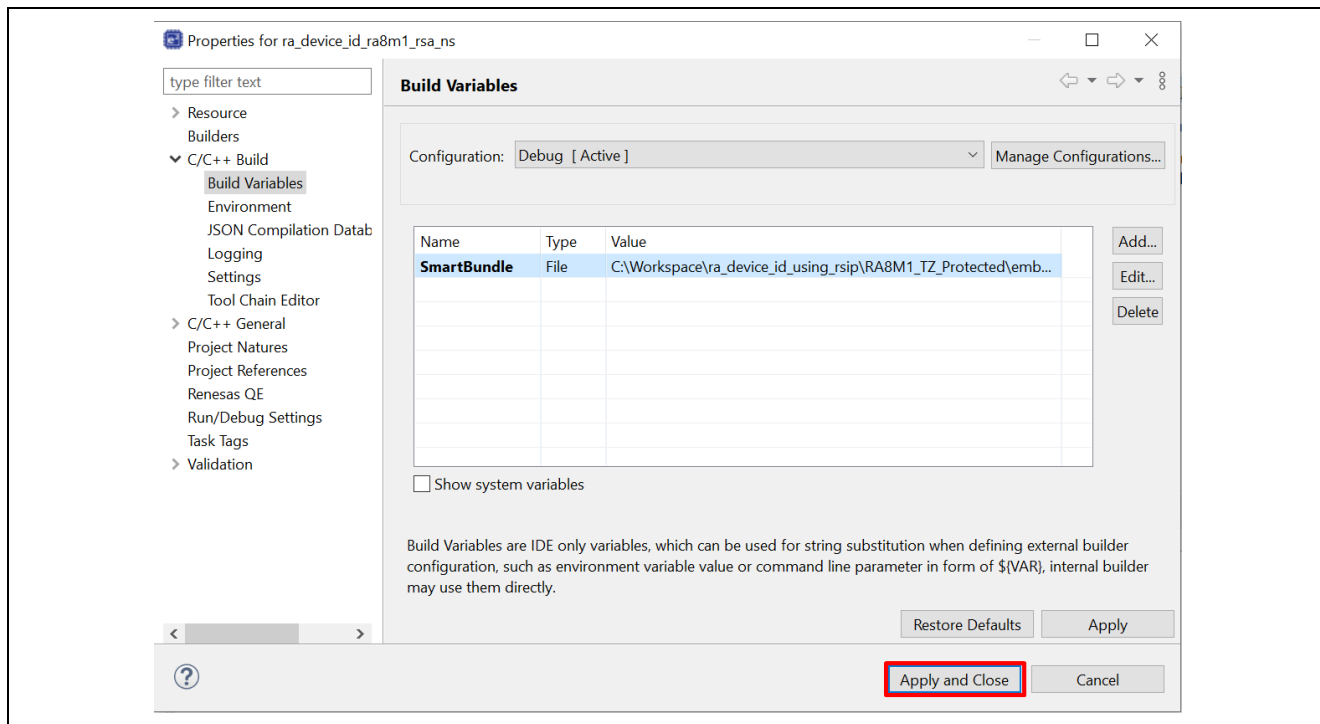
`\ra_device_id_ra8m1_rsa_s\Debug\ra_device_id_ra8m1_rsa_s.sbd`. Click **OK**.

**Note:** For the Split project development model, the non-secure project is linked to the secure project through the secure bundle file. The non-secure developer does not have access to the secure project source code. In a real-world use case, the `.sbd` file may be stored in any development folder. The user needs to manually select the secure bundle file.



**Figure 14. Select the Secure Bundle**

Click **Apply and Close**.



**Figure 15. Finish Updating the Secure Bundle Linkage for the Non-secure Project**

- B. Expand project `ra_device_id_ra8m1_rsa_ns` and double click `configuration.xml`. Once the RA configurator is opened, click **Generate Project Content** and then compile the project.

## 6.2 Powering up the Board

Follow the steps below to set up the hardware and jumpers to prepare debugging the system using the J-Link debugger:

1. EK-RA8M1 jumper setting: J6 closed, J9 open, J12 jumper on pins 2-3. Other jumpers keep out-of-box setting.
  - Connect J10 on EK-RA8M1 using a micro-USB cable to the workstation to provide power and debugging capability using the on-board debugger.
  - Connect J11 on EK-RA8M1 using a micro-USB cable to the workstation to provide USB connection.
2. EK-RA4L1 jumper setting: J6 open, J9 open. Other jumpers keep out-of-box setting.
  - Connect J10 on EK-RA4L1 using a type-C USB cable to the workstation to provide power and debugging capability using the on-board debugger.
  - Connect J11 on EK-RA4L1 using a type-C USB cable to the workstation to provide USB connection.
3. EK-RA4C1 jumper setting: J6 jumper on pins 2-3, J9 jumper on pins 2-3, SW4-4 is OFF. Other jumpers keep out-of-box setting.
  - Connect J10 on EK-RA4C1 using a type-C USB cable to the workstation to provide power and debugging capability using the on-board debugger and for using UART connection.

## 6.3 Downloading and Verifying the Demonstration

### 6.3.1 Download Flat project and run PC application

Once the projects are compiled and the evaluation kit is connected to the development PC. We can follow the steps below to download the projects.

For Flat projects on RA4C1 or RA4L1, user can start a debug session for the e<sup>2</sup>studio project and click “Resume” twice to run past the `main()`.

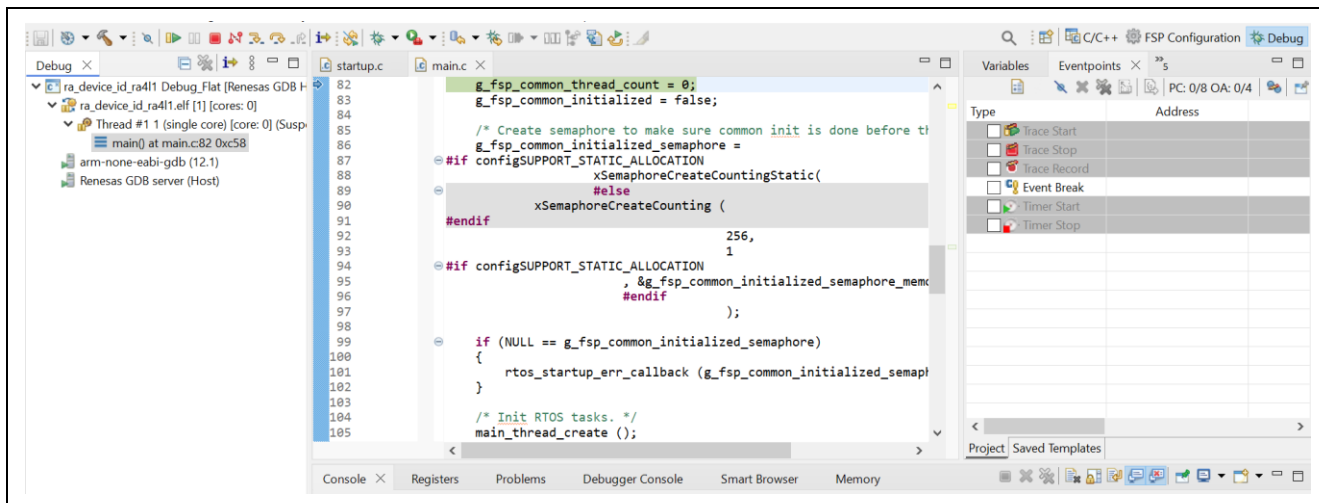


Figure 16. Debug session

The target kit will now show up as the **USB Serial Device (RA4L1)** or **CDC UART Port (RA4C1)** in the Device Manager. Make a note of the COM port number of the target kit from the Device Manager.

Now, run the host application on the Windows PC. To run the application, open the command window on your Windows PC and navigate to the folder where this application project is stored. The `deploy.exe` file will be located under the `ra_device_id_using_rsis\<Project_name>\pc\apps\deploy\Release` directory.

To run host application, type the following command on the command window:

```
deploy.exe connect <COM port number>
```

```
argc = 3, argv[1] = connect, argv[2] = 05
Scanning for devices on port 05
Initialising connection to COM port [5]
Initialised connection to COM port [5] OK
Issuing Generate Key command to device
Create the cert signed by signing key (local CA instance)
Successful challenge/response
Sending device Certificate to the device, len = 936
Device Cert succesfully created and programmed into device
```

Figure 17. Host application console messages on RA4L1

```
argc = 3, argv[1] = connect, argv[2] = 52
Scanning for devices on port 52
Initialising connection to COM port [52]
Initialised connection to COM port [52] OK
Issuing Generate Key command to device
Create the cert signed by signing key (local CA instance)
Successful challenge/response
Sending device Certificate to the device, len = 936
Device Cert succesfully created and programmed into device
```

Figure 18. Host application console messages on RA4C1

At this stage, the host application communicates with the target through the USB-PCDC or SCI/UART communication interface. The user application does the following tasks as shown in section 5.3:

1. Scans for the USB COM port provided by the user. If it finds it, it opens the serial connection.
2. On successful serial connection, it issues the Generate Key command to the target kit.
3. On the device side, the ECC key pairs are generated using the RSIP crypto modules. The public key is sent back to the host application.

4. The host application creates root CA and signing key to be used to sign the device certificate at the later stage.
5. Generates a challenge/response string and sends it to the target kit.
6. On successful challenge/response, the host application will sign and send the device certificate to the device.
7. The device certificate will be securely stored in the internal code flash and can be protected by Flash Block Protection.

### 6.3.2 Download TrustZone project

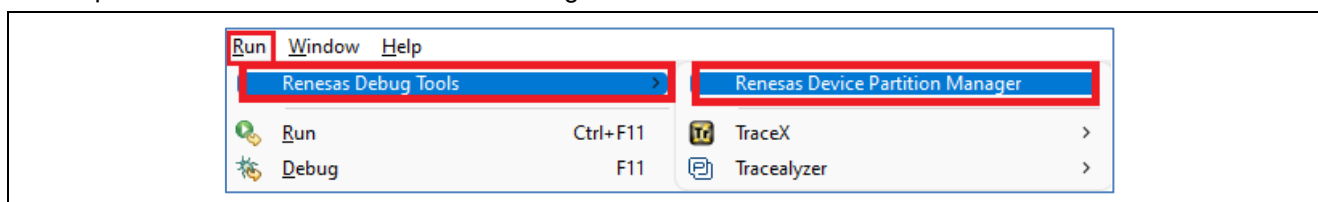
Once the projects are compiled and the EK-RA8M1 is connected to the development PC. We can follow below steps to download the projects.

Note: We advise that you read the Split Project Development Model explanation in chapter 3.2 Split Project Development in application note [Security Design using Arm TrustZone - Cortex M85](#) to understand why a dummy non-secure project (developed with Combined Project development model) needs to be downloaded with the secure project first.

Prior to downloading the application projects, it is recommended to check on the current Device Lifecycle State of the MCU. This can be achieved by using the Renesas Device Partition Manager which is integrated in e<sup>2</sup>studio.

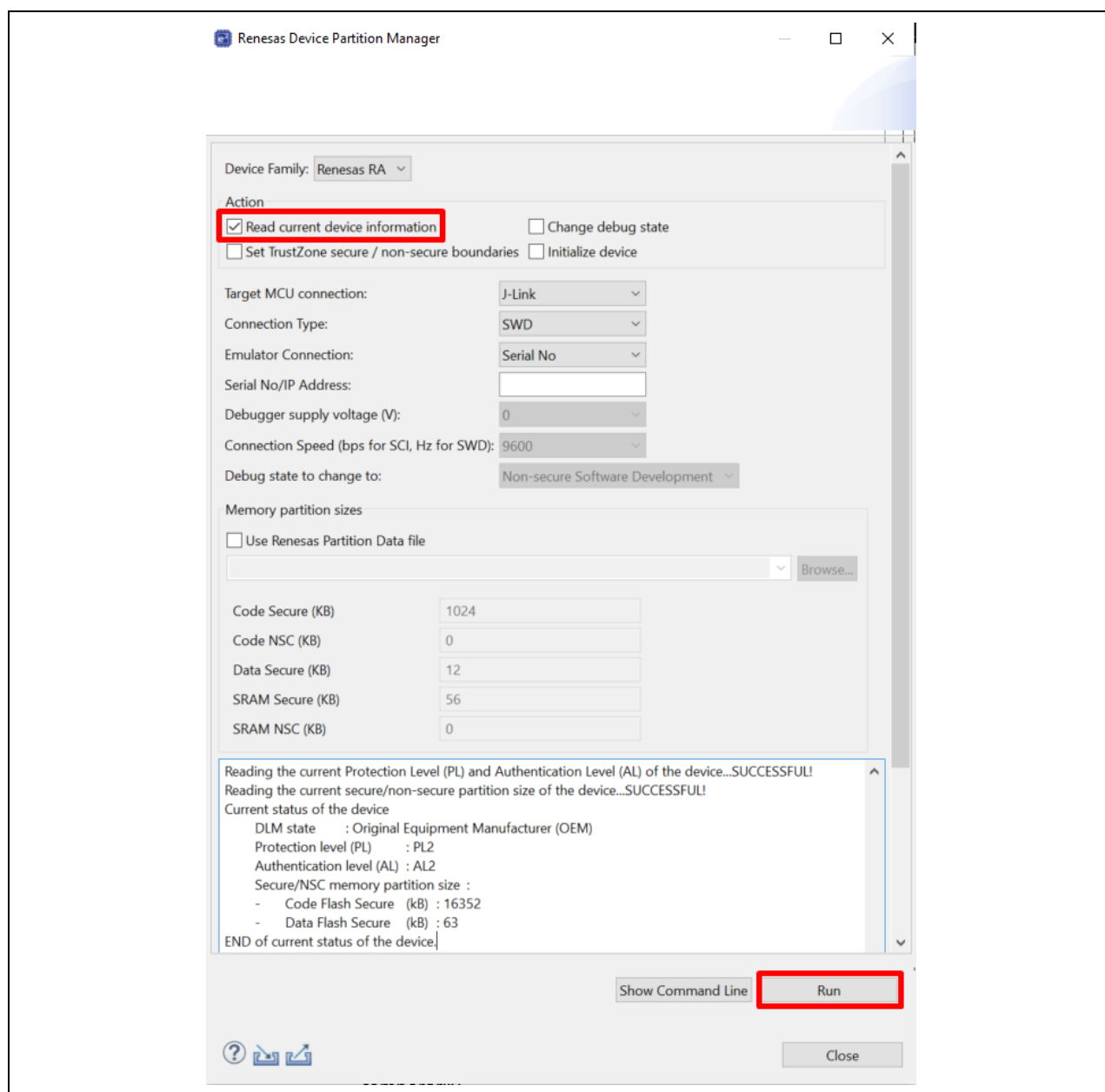
Power cycle the board prior to working with the Renesas Device Partition Manager after a debug session if using J-Link as the connection interface. This is needed to transition to MCU Boot mode, so the Renesas Device Partition Manager can take control of the device via the SCI/SWD interface. For details on the hardware connections, please see section IDAU Registers in the FSP User's Manual.

1. Open the Renesas Device Partition Manager.



**Figure 19. Open Renesas Device Partition Manager**

Next, select **Read current device information** and click **Run**. If the device lifecycle is read as OEM\_PL1 or OEM\_PL0, then you need to **Initialize the MCU**. **Initialize the MCU**



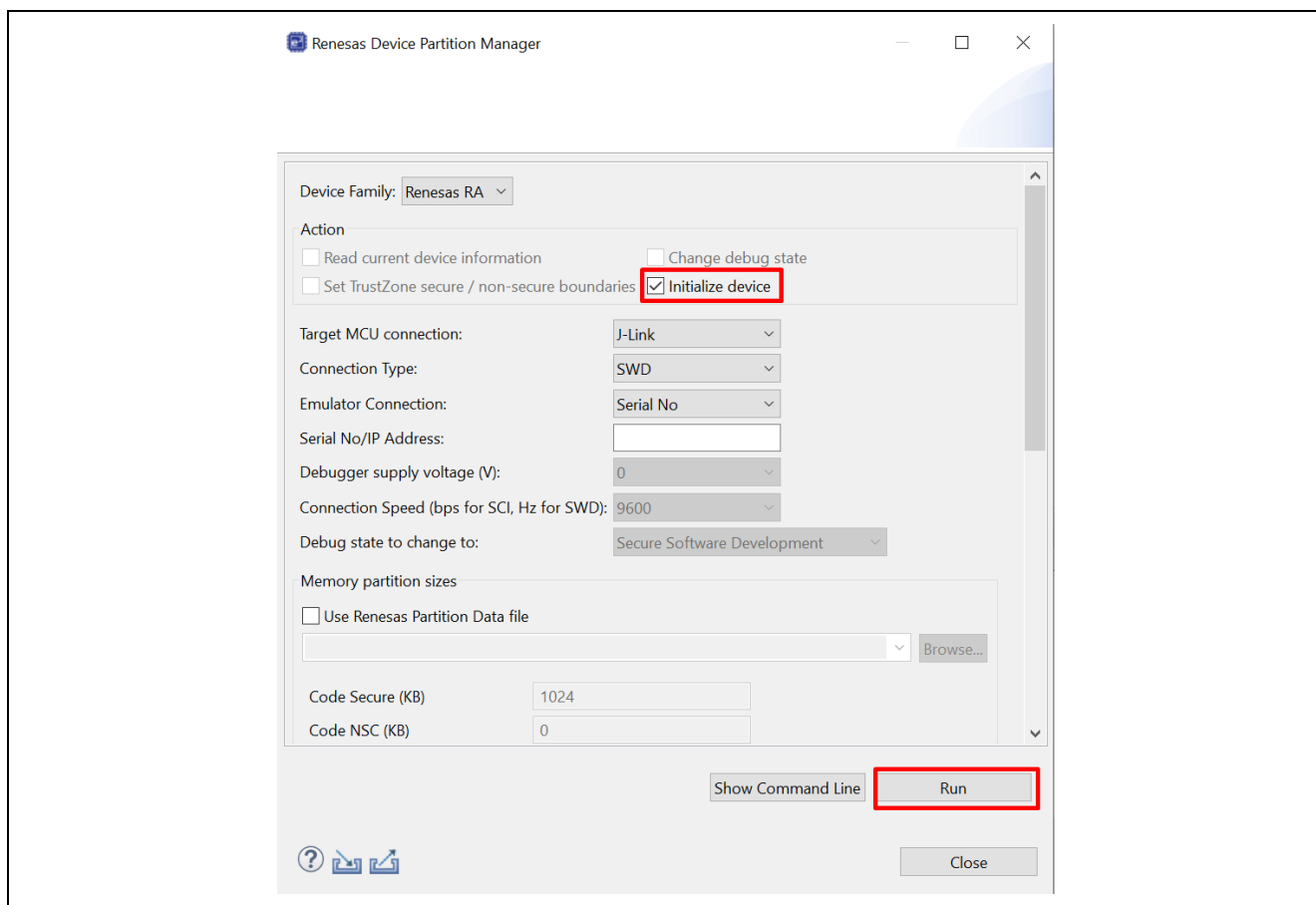
**Figure 20. View the Current Device Lifecycle State**

### 6.3.2.1 Initialize the MCU

This step is needed if the Device Lifecycle State is read as OEM\_PL1 or OEM\_PL0. This step **MUST** be done after running the Device Identity projects included in this application project prior to running any other TrustZone® based application because the Device Lifecycle State will be in OEM\_PL1 after getting the project up and running.

Note: Flash content not permanently locked down will be erased during this process. This is particularly helpful if the device was previously used in OEM\_PL1 state or has certain flash block locked up temporarily.

Select **Initialize device**, choose **J-Link** as the connection method and then click **Run**.



**Figure 21. Initialize RA8M1 using Renesas Device Partition Manager**

Power cycle the EK-RA8M1 after successfully initializing the device to OEM\_PL2 state by disconnecting both USB cable and reconnecting them to the development PC.

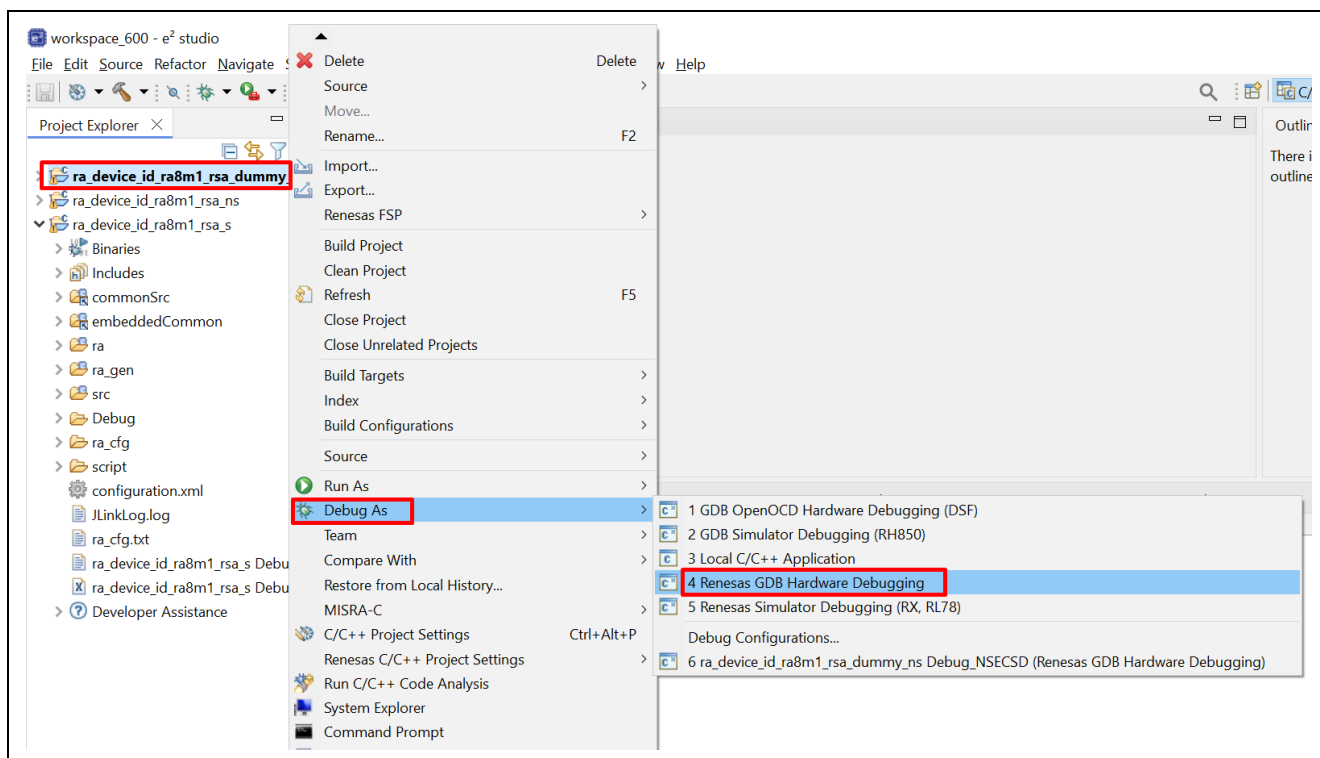
### 6.3.2.2 Download the Secure Project and the Dummy Non-secure Project

The dummy non-secure project is developed with the Combined Project Development Mode with the following features.

- It is debugged in the OEM\_PL2 state.
- It performs no real functionality (only while loop).
- It is used to create the conditions to allow debugging of the real Non-Secure Project (which is created with the Split Project Development model and can only be debugged under OEM\_PL1 state).

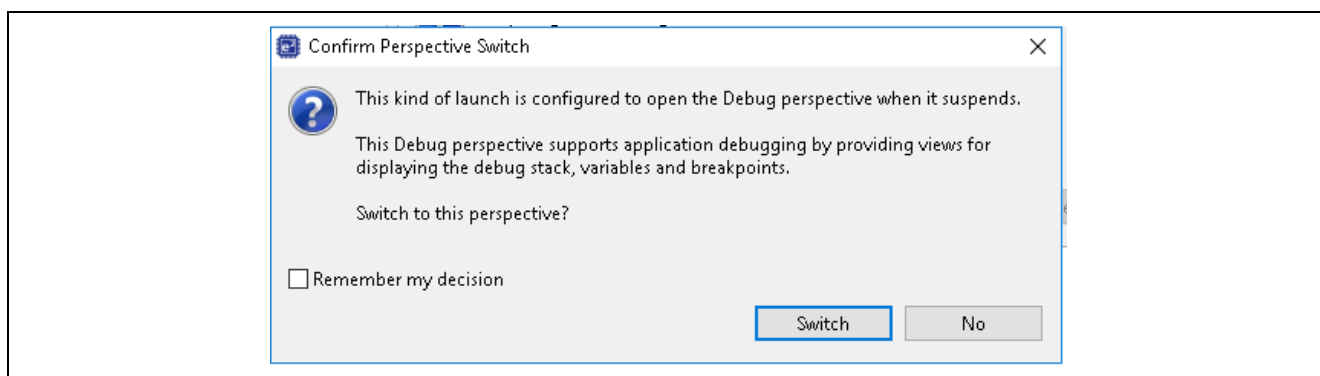
After the power recycle, right click on `ra_device_id_ra8m1_rsa_dummy_ns`, and select **Renesas GDB Hardware Debugging**.



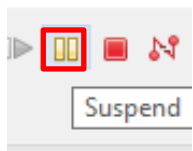


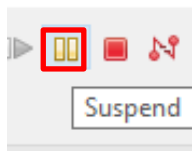
**Figure 22. Download the Secure Project and Non-secure Dummy Projects**

Click **Switch** if the following window pops up. Then click Resume  twice to run the project. At this point, the secure image and the dummy non-secure image are both downloaded.



**Figure 23. Select Switch to Debug perspective**



Click Suspend button . The execution should be pointing to the `while(true);` loop in the dummy non-secure project. Click Stop/Terminate  to end the debug session.

### 6.3.2.3 Change the MCU Device Lifecycle State to OEM\_PL1

Prior to this step, the MCU is running in OEM\_PL2 state where both secure and non-secure regions can be debugged. For security consideration, the MCU state can be transitioned to OEM\_PL1 state so only Non-secure regions can be debugged. For more understanding on the Device Lifecycle Management, please reference application note Device Lifecycle Management for RA8 MCUs ([R11AN0785](#)).

To change the Device Lifecycle State during development, we can use Renesas Device Partition Manager which is integrated with e2 studio.

Power cycle the board prior to working with the Renesas Device Partition Manager after a debug session when using J-Link as connection interface. This is needed to transition to MCU Boot mode, so the Renesas Device Partition Manager can take control of the device via the SCI/SWD interface.

Disconnect both USB cables from the development PC and then reconnect them to the PC.

Navigate to the **Run** tab in e<sup>2</sup> studio and select **Renesas Device Partition Manager**. Once the Renesas Device Partition Manager is opened, make the selections shown in Figure 24 and then click **Run**.

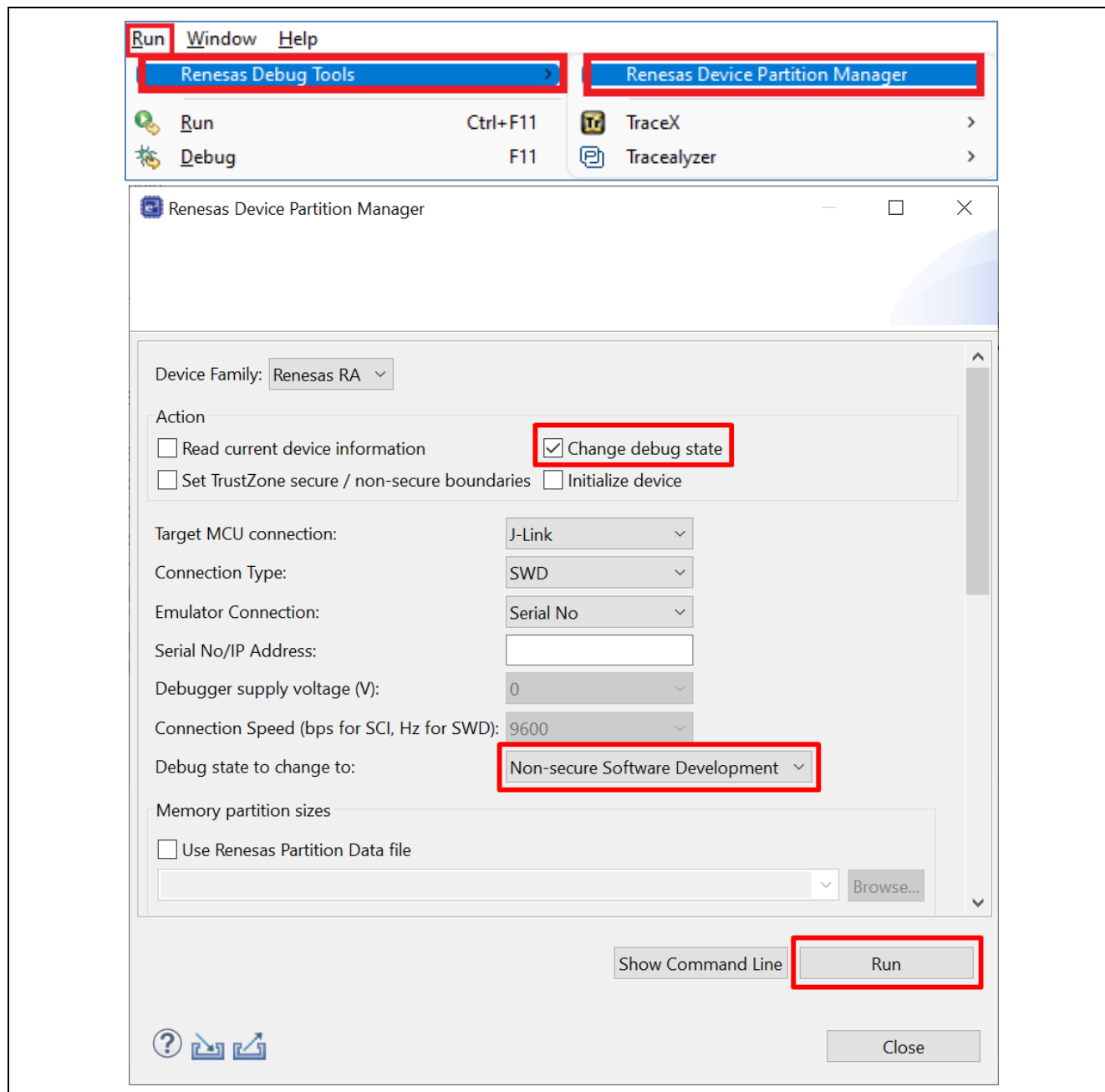
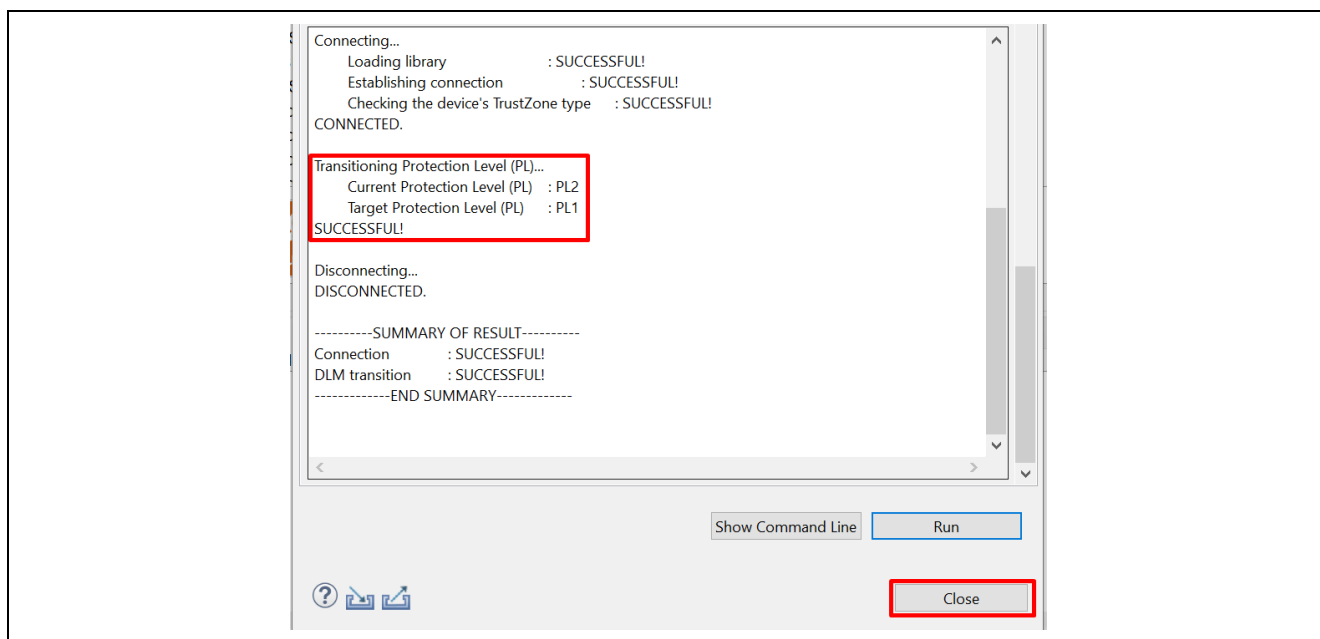


Figure 24. Open the Renesas Device Partition Manager

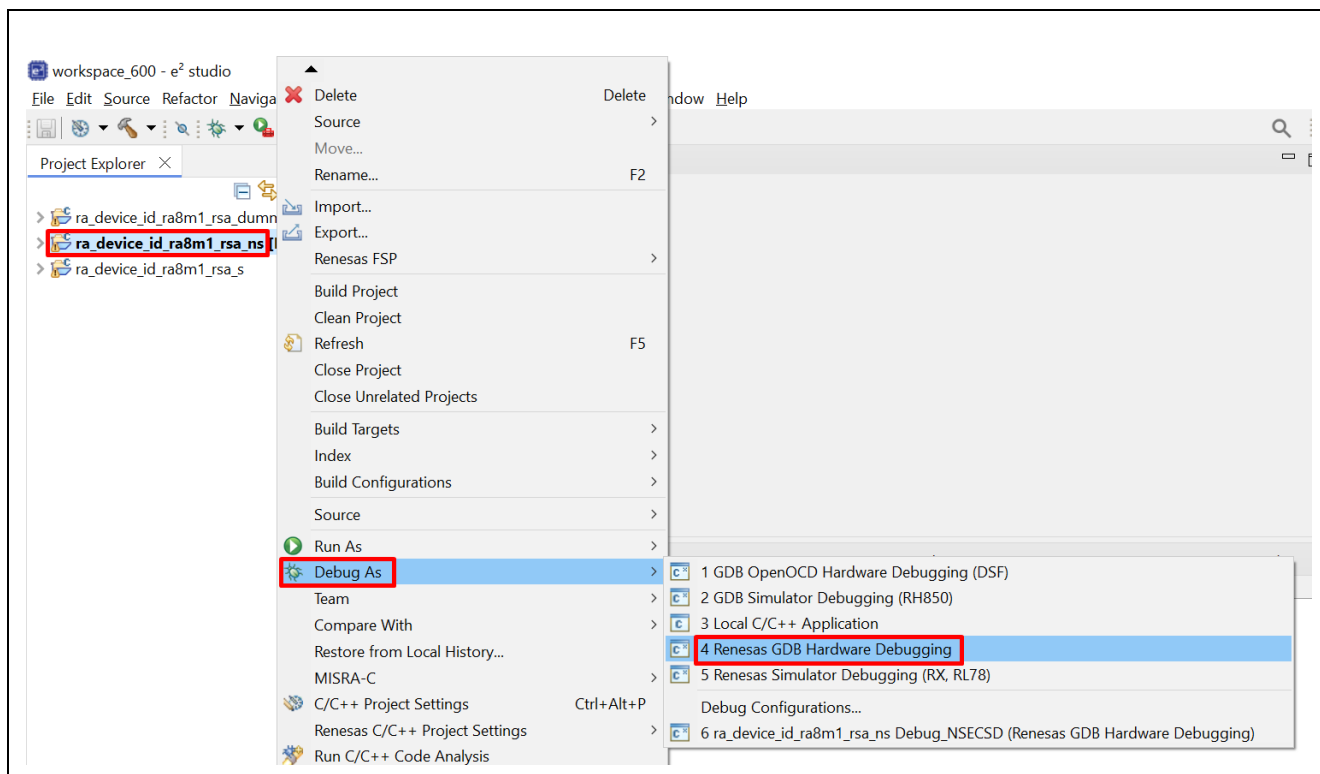


**Figure 25. Successfully Transitioned to OEM\_PL1**

Click **Close** to close the Renesas Device Partition Manager.

#### 6.3.2.4 Download the Non-Secure Project and Run the Application

Right click on project **ra\_device\_id\_ra8m1\_rsa\_ns** and select **Renesas GDB Hardware Debugging**. Note that this download will not touch the secure project content previously downloaded, but this Non-Secure project will overwrite the previously downloaded Dummy Non-Secure project. When the MCU starts execution, the secure project will transition to non-secure execution to this non-secure project which is being downloaded in this step.



**Figure 26. Debug the Non-secure Project with Secure Project Already Downloaded**

Click Resume  twice to run the project.

### 6.3.2.5 Run the PC Application to Communicate with the MCU

To run the PC application, open the command window in your Windows PC and navigate to the folder where this application project is stored. The `deploy.exe` file will be located under the `ra_device_id_using_rsip\RA8M1_TZ_Protected\pc\apps\deploy\Release` directory.

To run the host application, type the following command on the command window as shown below.

```
deploy.exe connect <COM port Number>
```

```
argc = 3, argv[1] = connect, argv[2] = 05
Scanning for devices on port 05
Initialising connection to COM port [5]
Initialised connection to COM port [5] OK
Issuing Generate Key command to device
Successfully set the RSA public key in devKey
Create the cert signed by signing key (local CA instance)
Successful challenge/response
Sending device Certificate to the device, len = 1229
Device Cert succesfully created and programmed into device
```

**Figure 27. Host application console messages on RA8M1**

At this stage, the host application communicates with the target through the USB-PCDC communication interface. The user application does the following tasks as shown in section 5.3:

1. Scans for the USB COM port provided by the user. If it finds it, it opens the serial connection.
2. On successful serial connection, it issues the Generate Key command to the target kit.
3. On the device side, the ECC key pairs are generated using the RSIP crypto modules. The public key is sent back to the host application.
4. The host application creates root CA and signing key to be used to sign the device certificate at the latter stage.
5. Generates a challenge/response string and sends it to the target kit.
6. On successful challenge/response, the host application will sign and send the device certificate to the device.
7. The device certificate will be securely stored in the internal code flash and protected by Arm® TrustZone®.

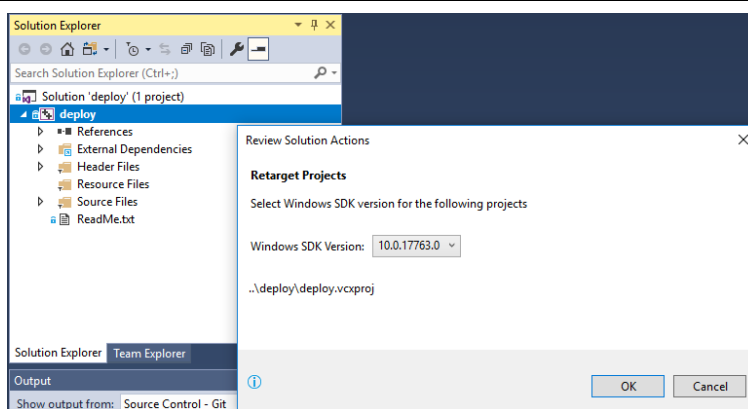
**Note:** After running this application project, user **MUST** follow section 6.3.2.1 to Initialize the MCU.

## 6.4 Customizing the Application Project

### 6.4.1 Customize the PC Application

To customize the PC application, first download the Visual Studio development environment using the software pointed out in the Required Resources section. Before compiling the project, retarget the project to use the Windows SDK installed on the development PC.

Next, compile the project and update as desired from this point onward.



**Figure 28. Retarget to the installed Windows SDK**

## 7. References

- Renesas RA Family Device Lifecycle Management for Cortex-M33 ([R11AN0469](#))
- Renesas RA Family Device Lifecycle Management for RA8 MCUs ([R11AN0785](#))
- Renesas RA Family Security Design with Arm® TrustZone® using Cortex-M33 ([R11AN0467](#))
- Renesas RA Family Security Design using Arm TrustZone – Cortex M85 ([R11AN0897](#))
- Renesas RA Family Injecting Plaintext User Keys ([R11AN0473](#))
- Renesas RA Family Injecting and Updating Secure User Keys ([R11AN0496](#))
- Renesas RA Family Establishing and Protecting Device Identity using SCE7 and FAW ([R11AN0449](#))
- Renesas RA Family Establishing and Protecting Device Identity using SCE9 and Arm® TrustZone® ([R11AN0475](#))

## 8. Known Issues and Limitations

The host application is tested only on Windows® 10/11 PC.

## 9. Appendix

### 9.1 Glossary

| Term                       | Meaning                                                                                                                                                                                                                                                               |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Certificate Authority (CA) | An entity that issues digital certificates according to policy-based rules. A CA could be public or private, located in the Cloud, or in the case of an on-premises CA, typically hosted on a secure appliance.                                                       |
| Device Certificate         | Certificate uniquely identifying an individual device. It is digitally signed, asserting that the certificate comes from a known source and has not been modified, and that the device is trusted.                                                                    |
| Root of Trust              | Roots of trust are highly-reliable hardware, firmware, and software components that perform specific, critical security functions.<br>( <a href="https://csrc.nist.gov/projects/hardware-roots-of-trust">https://csrc.nist.gov/projects/hardware-roots-of-trust</a> ) |
| RSIP                       | Renesas Secure IP – A module in the MCU that provides for efficient, low-power cryptographic acceleration, TRNG (True Random Number Generation), creation and isolation of cryptographic keys.                                                                        |
| PKI                        | Public Key Infrastructure – A set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates, which are typically used to manage secure identity via public key cryptography.                               |
| Key Pair                   | Asymmetric keys are generated in pairs – a public and private key. The private key is held in secret by only one party and can be used to assert that party's identity. The public key is freely distributed and is uniquely associated with the private key.         |
| Secure Code                | A function or group of functions that resides in a secure region of internal flash or internal SRAM, as defined and enforced by the TrustZone®. Please reference the Secure Data at Rest for RA Family to understand the access control of the Secure Code.           |
| Non-Secure code            | A function or group of functions that resides in a non-secure region of internal flash or internal SRAM. Please reference the Secure Data at Rest for RA Family to understand the access control of the non-Secure code.                                              |
| Challenge String           | Randomly generated string at the host application. This string is used by the host application to validate the ownership of the private key by the target.                                                                                                            |
| Unique ID                  | An identification value, unique to each individual RA Family MCU, that is stored inside the MCU. The unique ID is used by the RSIP when it wraps a key.                                                                                                               |

## 10. 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              | <a href="https://renesas.com/ra/ek-ra8m1">renesas.com/ra/ek-ra8m1</a> |
| EK-RA4L1 Resources              | <a href="https://renesas.com/ra/ek-ra4l1">renesas.com/ra/ek-ra4l1</a> |
| EK-RA4C1 Resources              | <a href="https://renesas.com/ra/ek-ra4c1">renesas.com/ra/ek-ra4c1</a> |
| RA Product Information          | <a href="https://renesas.com/ra">renesas.com/ra</a>                   |
| Flexible Software Package (FSP) | <a href="https://renesas.com/ra/fsp">renesas.com/ra/fsp</a>           |
| RA Product Support Forum        | <a href="https://renesas.com/ra/forum">renesas.com/ra/forum</a>       |
| Renesas Support                 | <a href="https://renesas.com/support">renesas.com/support</a>         |

**Revision History**

| Rev. | Date      | Description |                 |
|------|-----------|-------------|-----------------|
|      |           | Page        | Summary         |
| 1.00 | Aug.06.25 | -           | Initial version |

# 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](http://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/](http://www.renesas.com/contact/).