Renesas RA Family

Establishing and Protecting Device Identity using SCE7 and FAW

Introduction

Covering general IoT security concepts, this application note details the security features and key generation options for the Renesas RA Family MCU groups specified in the Target Devices section.

The example application provided in this package uses the Secure Crypto Engine 7 (SCE7) module based on RA6M3 to generate a unique device identity. This identity is securely stored in the internal flash using the MCU's Flash Access Window (FAW) hardware feature.

This application note enables you to effectively use the RA Family SCE7 module and the Mbed Crypto middleware in your own design. Upon completion of this guide, you will be able to add the RA Family Flexible Software Package (FSP) Mbed Crypto middleware and the SCE7 module to your own design, configure them correctly for the target application, and write code using the included application example code as a reference and efficient starting point. References to more detailed API descriptions and other application projects that demonstrate more advanced uses of the module are in the RA Family FSP User's Manual and serve as a valuable resource in creating more complex designs.

The example application is based on EK-RA6M3. It can be easily adapted to the other MCUs listed in the **Target Devices** section if the USB Full Speed peripheral is provided on the development board.

Required Resources

Target Devices

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

- RA6M1
- RA6M2
- RA6M3
- RA6T1

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

Development tools and software

- e² studio IDE v2025-12
- RA Family Flexible Software Package (FSP) v6.4.0
- SEGGER J-link® USB driver V9.14a

The above three software components: the FSP, J-Link USB drivers, and e² studio are bundled in a downloadable platform installer available on the FSP webpage at renesas.com/ra/fsp

- Visual Studio 2026 Community Version (https://visualstudio.microsoft.com/downloads/)

Hardware

- EK-RA6M3, Evaluation Kit for RA6M3 MCU Group (www.renesas.com/ra/ek-ra6m3)
- Test PC running Windows® 11 OS
- Two Micro USB cables

Prerequisites and Intended Audience

Users of this application note should have some prior experience with the Renesas e² studio 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 e² studio and the FSP and validates that the debug connection to your board functions properly. In addition, this application note assumes that you have some knowledge of cryptography and RA Family Secure Crypto Engine 7 (SCE7) features.

The intended audience are users who want to develop applications with SCE7 modules using Renesas RA Family MCUs with SCE7 support.

Contents

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)

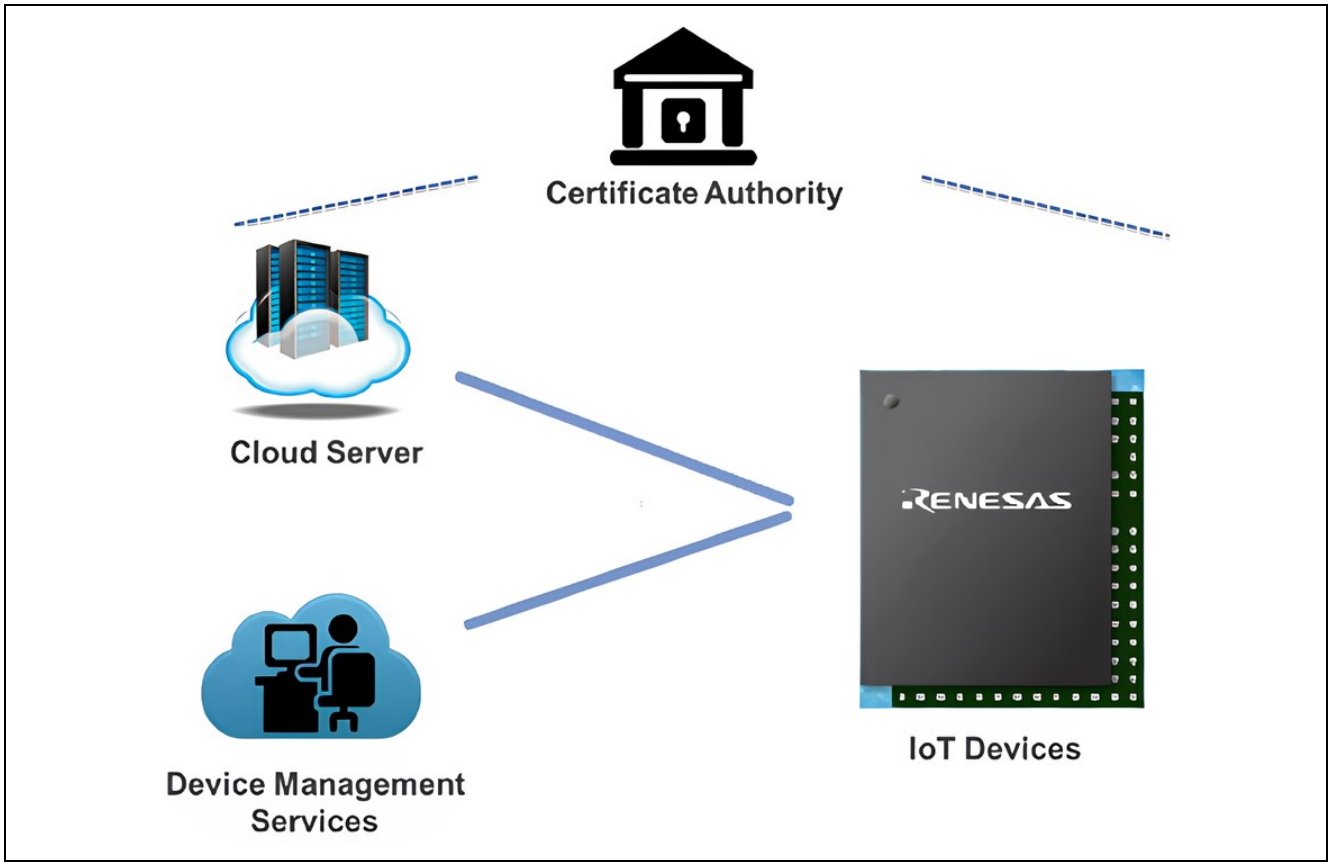

Figure 1. IoT Environment Overview

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. Due to their connectivity to publicly accessible infrastructure, IoT devices are susceptible to malicious attacks.

Cloud Server

The Cloud Server is a network-connected server that provides services to the IoT devices. It is typically located in a highly secure and controlled data center.

Device Management Services

Device Management services offer a comprehensive suite of IoT device management capabilities to enable IoT customers of any size to have complete control over their devices and data. This includes (but is not limited to):

- Application Security
  - Keys/certificates identifying the Cloud Server.
- Device Management Security
  - Keys/certificates identifying each unique IoT Device.
  - Keys/certificates identifying the Device Management Services Server.
  - 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 a 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 of 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 Using the RA Family MCU SCE7 and FAW Hardware Security Features

RA Family MCUs enable hardware root-of-trust mechanisms by providing the ability to protect memory blocks. The contents of flash memory can be locked from future erase/write events using the Flash Access Window (FAW). The cryptographic operations are supported using the Secure Crypto Engine 7 (SCE7) module. The details of the usage of these hardware features are explained in this section.

1.3.1 FAW (Flash Access Window)

The FAW registers are used to set the code flash address range that can be erased/programmed. The addresses that are outside this range, referred to as outside the FAW, cannot be modified after the FAW window is set. This feature is used to prevent the device identity (keys/certificates) from being erased or reprogrammed.

The example application project provided along with this package includes code reference to configure the FAW using APIs provided by the FSP for storing information that establishes device identity. Users can also refer to the Secure Data at Rest application project for more use cases on FAW configuration.

For more detailed information on the FAW, see the FSP User's Manual link in the reference section.

Note: The FAW is set to the area of flash that can be written, so the area of memory that is LOCKED is the area outside of the FAW address range.

FSPR (Protection of Access Window and Startup Area Select Function)

The FAW register setting can be permanently set using the FSPR bit. This bit is **one-time programmable** and so must be set only when all the settings are confirmed and the device is ready to leave the production floor.

1.3.2 Secure Crypto Engine 7 Module

The Secure Crypto Engine 7 (SCE7) is an RA Family MCU hardware peripheral that provides several security features, including NIST-certified algorithms and support for cryptographic primitives.

The RA Family MCUs which this application note is targeting support asymmetric cryptography as well as symmetric cryptography. Following is a diagram of the SCE features offered by these MCUs.

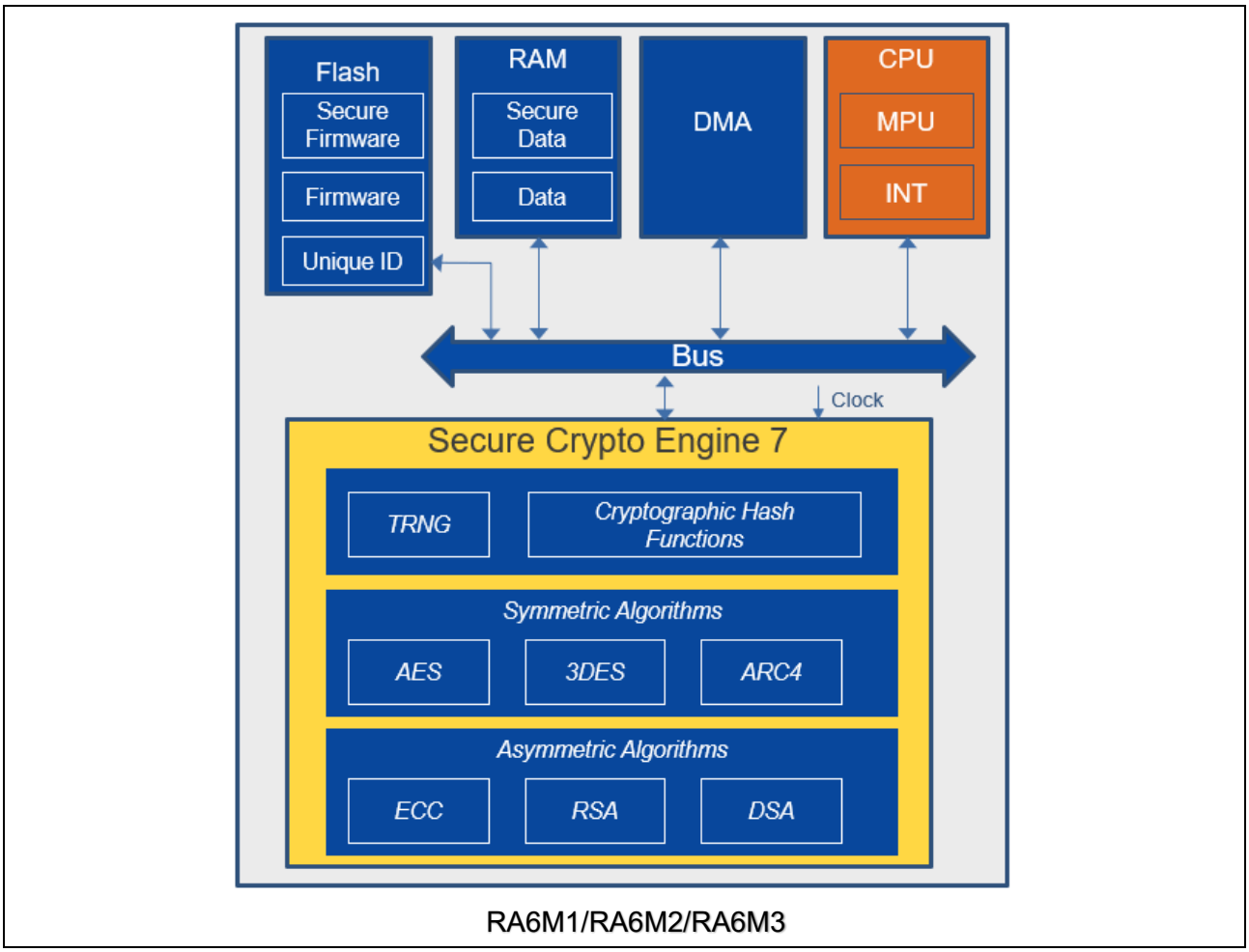

Figure 2. Security Hardware Peripherals Available in RA6M1/2/3

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

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

2. Overview of Key Generation in RA Family MCUs

2.1 Key Wrapping

Device keys generated inside the RA Family MCU using the SCE hardware module can be either in plain text or wrapped, depending on the type of key.

Plaintext Keys

Plaintext refers to information or data in an unencrypted or unprotected form that is readable by either a human or a machine and can be used without the need for any special processing.

Wrapped Keys

A wrapped key is a key that has been encrypted by the SCE using a method that involves the use of MCU-unique information. For RA Family MCUs containing the SCE7, the MCU's unique ID is used. Because this method requires the MCU's unique ID 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; 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 wrapped.
- 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.

2.2 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 SCE module. To generate the device key using the RA Family Flexible Software Package (FSP), the **Mbed Crypto** module is used. The Mbed Crypto module implements PSA Crypto APIs which call the Secure Cryptographic Engine 7 (SCE7) HAL module, which in turn drives the SCE IP on the device.

Mbed Crypto Module Features

The following key types can be generated using the services of the Mbed Crypto module using SCE7 hardware:

- RSA 1024-bit and 2048-bit plaintext public keys in standard format.
- RSA 1024-bit and 2048-bit standard format wrapped private keys.
- AES 128-bit, 192-bit, and 256-bit wrapped keys for ECB, CBC, CTR, and GCM chaining modes.
- AES 128-bit and 256-bit wrapped keys for XTS chaining mode.
- ECC 192-bit and 256-bit plaintext public keys and wrapped private keys.

In this application, ECC secp256r1 plain-text public keys and wrapped private keys are generated.

2.3 Key Injection from Secure Infrastructure

Key injection is a security feature that is meant for use cases where a key is generated external to the MCU device in a secure facility and then injected into the MCU. In general, the RSA key generation takes more time when generated inside the MCU compared to if it was generated external to the device (through a PC tool) inside a secure facility, especially if the corresponding public key needs to be placed in a signed certificate.

If hardware-based unique identity is not a requirement for the application, and/or if it is necessary to securely inject customer-specific keys, key injection can be utilized. This process will be covered by a separate Application Project.

3. 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. Based on the customer use case, the decision must be made.

Certificate Authority (CA)

Once the device keys are generated/injected, we need an entity that issues digital certificates. 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 Device Identity

Once the device identity is created and programmed on the RA Family device, it must be securely stored to prevent alteration. This can be achieved by using the FAW feature offered by the RA Family MCU. Once the one-time programmable FPSR bit is cleared, the FAW register setting and the code flash section locked by FAW cannot be modified anymore.

4. Device Identity Application Example

4.1 Overview

The example application project accompanying this document demonstrates natively generating and storing the device identity information using the on-chip SCE 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-CDC is used as the primary communication interface between the EK-RA6M3 kit and the host console application running on the Windows® PC.

4.2 Software Architecture Overview

The following figure shows the overall software architecture of the RA Family device identity application project. The light green blocks are components from FSP ecosystems: AWS FreeRTOS block is a component from AWS and the other light green background blocks **PSA Cryptography API**, **Mbed Crypto lib**, and **littlefs** are components from Arm.

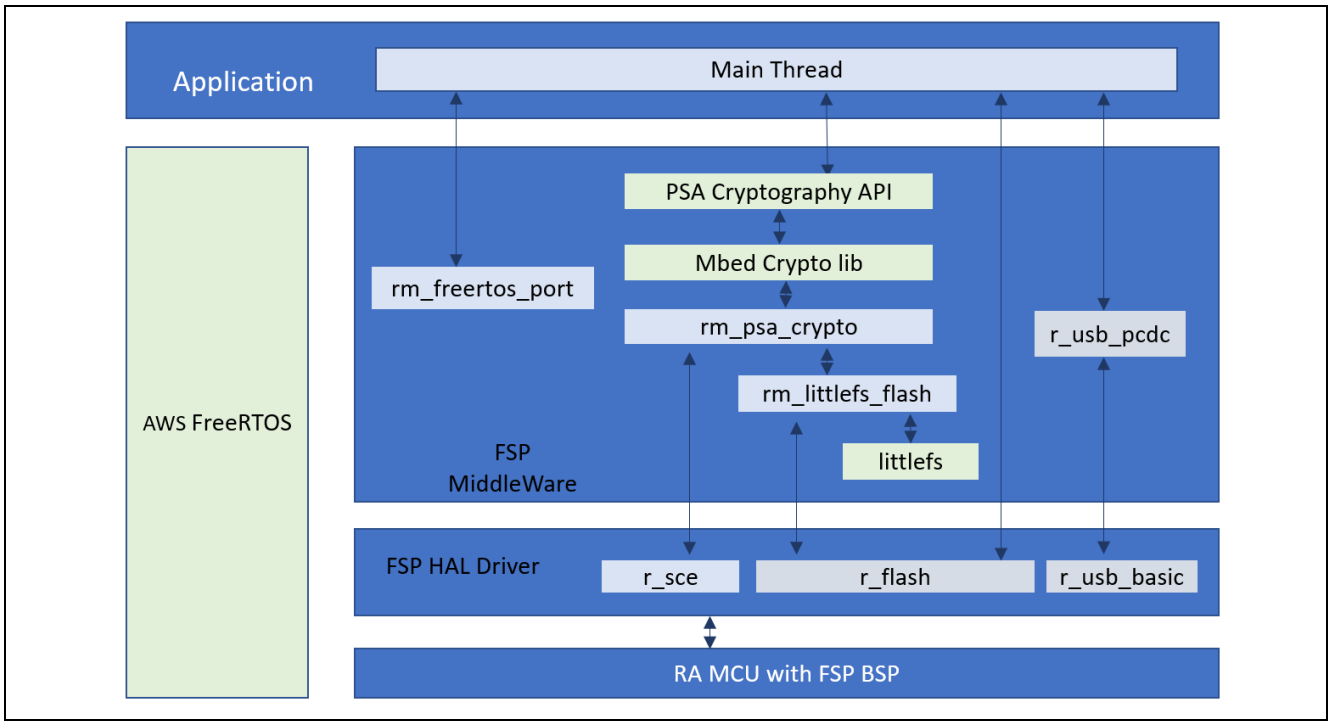

Figure 3. RA Device Identity Application Software Architecture

The major FSP software components of this application are:

- rm_psa_crypto and r_sce: 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

Main Thread

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

1. Incoming/outgoing USB 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_CERT_PROGRAM
- WRAPPED_KEY_CHALLENGE_RESP

WRAPPED_KEY_REQUEST

This command is handled by the following API function: `handleWrappedKeyCreation()`

This function handles the key generation using FSP Crypto modules. This application supports ECC 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 to sign the challenge response.

WRAPPED_KEY_CERT_PROGRAM

This command 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.

WRAPPED_KEY_CHALLENGE_RESP

This command 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 the request is received, 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 sends the device certificate to the device to be stored securely using FAW.

4.3 Operational Overview

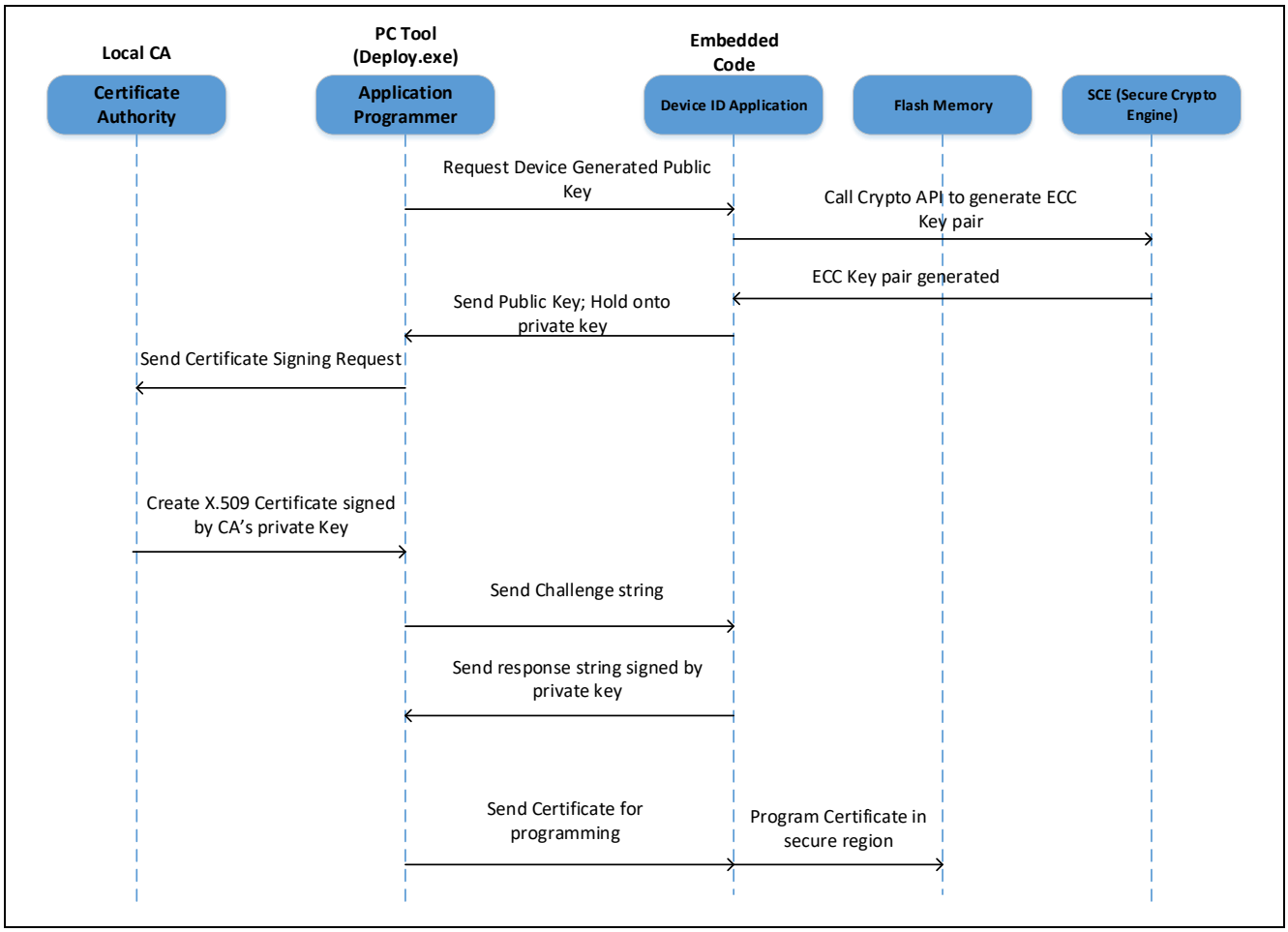

Figure 4. Operational Overview

This application project consists of two software projects:

- Embedded project running on the EK-RA6M3 kit.
- Host application running on Windows® 11 PC.

Upon supplying power to the EK-RA6M3 kit, the firmware initializes the platform and the underlying USB CDC stack that is used for communication with the host application running on Windows PC. At the end of initialization, the firmware waits for the USB device connection event. Once the user connects the kit to the Windows PC through a USB cable, the USB enumeration process occurs, and the USB CDC 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 EK-RA6M3 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. The host application receives the public key from the device and generates a device certificate (signed by the 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 EK-RA6M3 kit to be securely stored using the FAW on the RA Family MCU.

4.4 Securely Storing Device Identity

The two unique device identities created as part of this application are as follows:

- Wrapped ECC private key
- Device certificate

These two device identities need to be stored inside the RA Family MCU using the FAW 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 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 project.

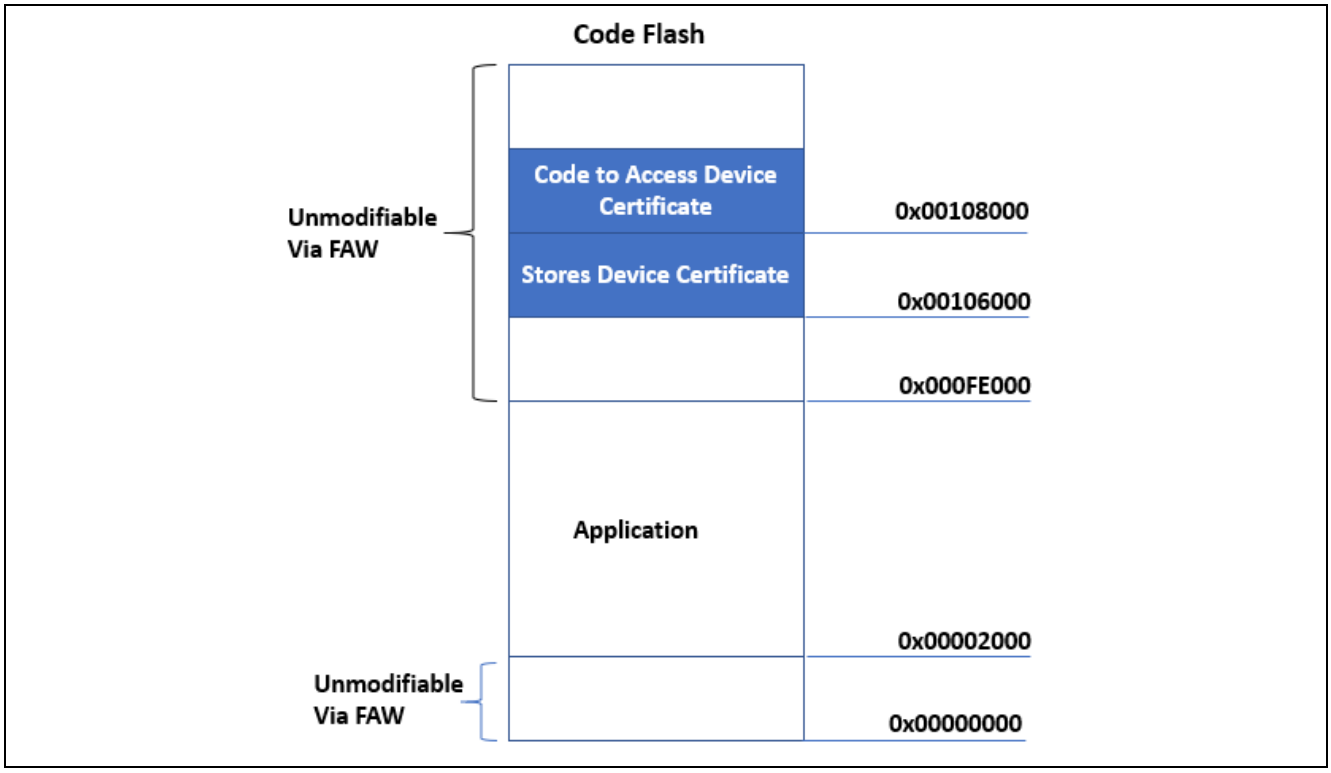

Figure 5. Memory Map used in the Application Project

Figure 5 shows the memory layout of the code flash in this application project. By setting the FAW start address to 0x2000 and the end address to 0xFE000, the sections outside these address regions are protected from being modified. See the `flash_FAW_Set()` API found in the `estab-protect-device-ID-SCE7-FAW \embedded\common\src\framedProtocolTarget.c` file, which implements the FAW settings.

5. Running the Device Identity Application Example

5.1 Importing, Building, and Running the Embedded Project

Unzip the `estab-protect-device-ID-SCE7-FAW.zip` file, the embedded projects are included in the folder `estab-protect-device-ID-SCE7-FAW\embedded`. The following instructions will show the user how to import these projects into their e² studio workspace.

In e² studio ISDE, select **File > Import… > General > Existing Projects into Workspace** and browse to the above folder in the **Select Root Directory** section:

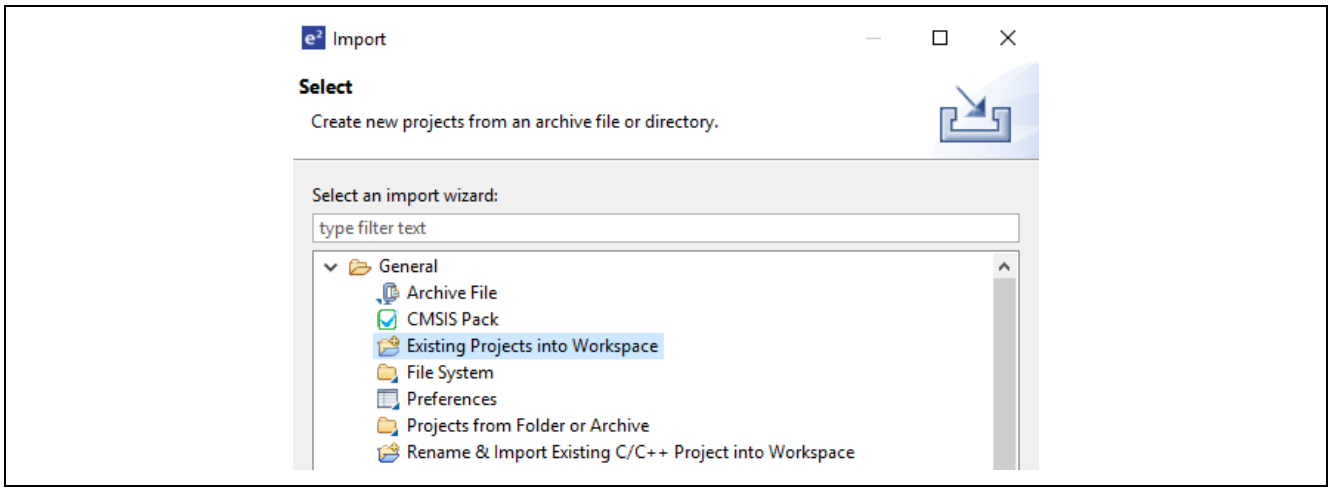

Figure 6. Importing the Project

Select to import all projects as shown in the following figure. **DO NOT CHECK the "Copy projects into workspace" box.**

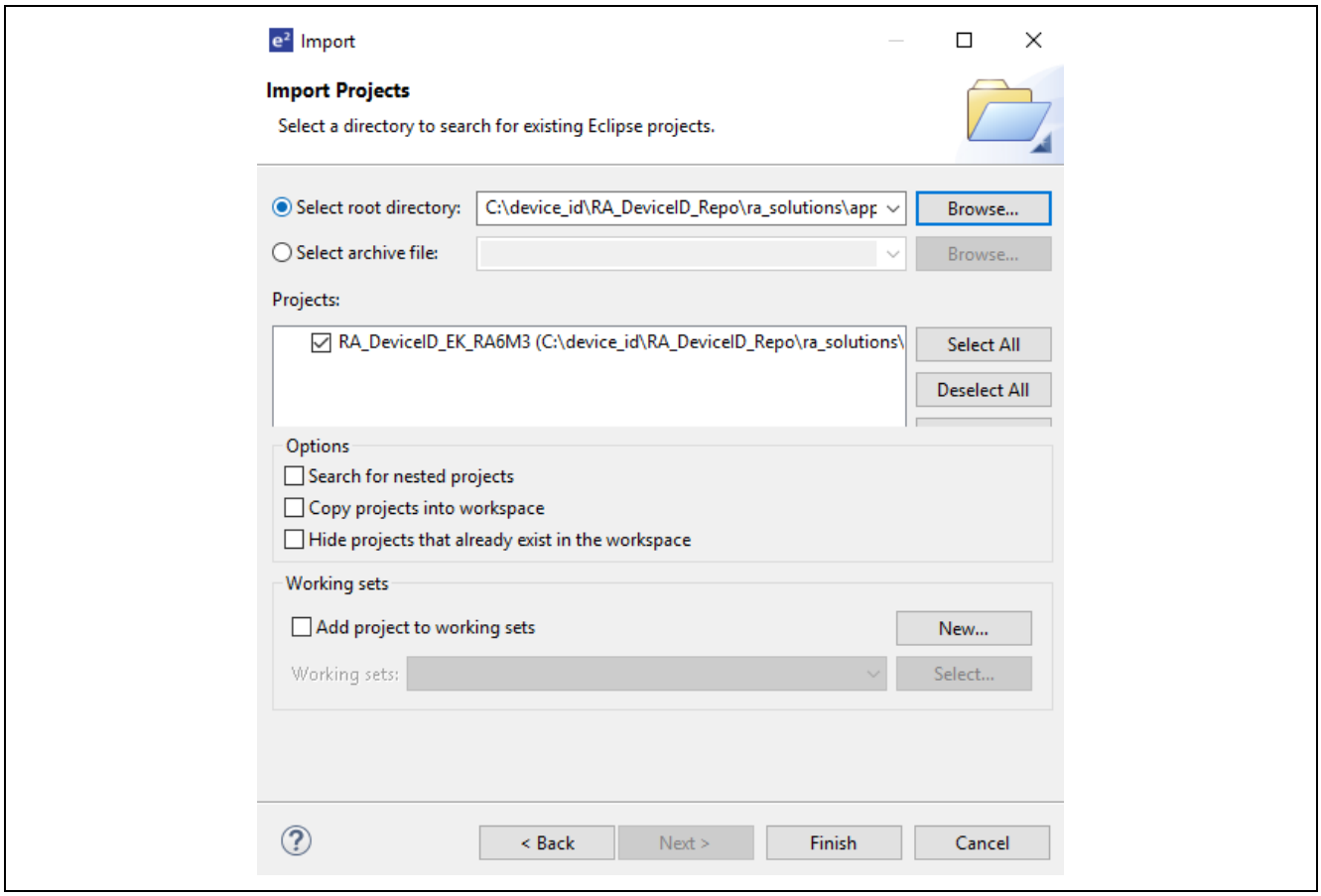

Figure 7. Selection for Importing the Embedded Project

After importing the project, open the RA configurator, click **Generate Project Content**, and then compile the project. It should compile without errors.

Notice that there are over 700 warnings after compilation. These warnings are from third-party software components, and Renesas is not responsible for fixing them.

5.2 Powering up the Board

To connect power to the board, use the following instructions.

1. Connect the micro-USB end of the supplied USB cable to the EK-RA6M3 board J10 connector (DEBUG_USB)
   Note: The kit contains a SEGGER J-Link® On-board (OB). J-Link provides full debug and programming for the EK-RA6M3 board. Connect the other end of the USB cable to the USB port on your workstation.
2. USB FS device mode jumper setting: connect pins 2 and 3 on J12.
3. Connect the micro-USB end of the other USB cable to the EK-RA6M3 board J11 connector (USB FS). Connect the other end of the USB cable to the USB port on your workstation.

5.3 Verifying the Demonstration

At this stage, it is assumed that you followed the instructions in section 5.1 to import, build, and load the application project into the target kit. If not, go back to section 5.1 and follow the steps before moving further in this section.

Now start a debug session for the e² studio project and click “Resume” twice to run past the `main()`.

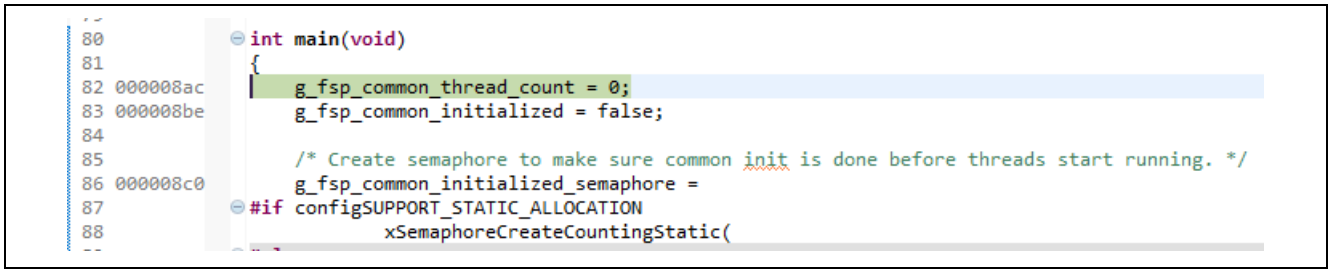

Figure 8. Click “Resume” to pass “main”

The target kit will now show up as the **USB Serial Device** in the Device Manager. Make a note of the COM port number of the target kit from the device manager.

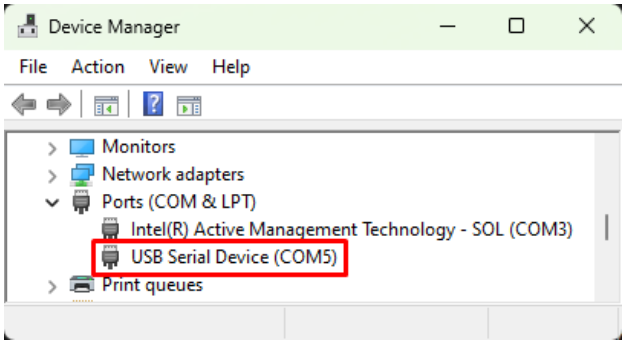

Figure 9. Get COM port from the Device Manager

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 `estab-protect-device-ID-SCE7-FAW\pc\apps\deploy\Release` directory.

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

`deploy.exe connect <COM port Number>`

```
C:\Workspace\r11an0449\pc\apps\deploy\Release>deploy.exe connect 05

 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 10. Host application console messages

Open the J-Link RTT Viewer using the setup in Figure 11. Click **OK** at the window below.

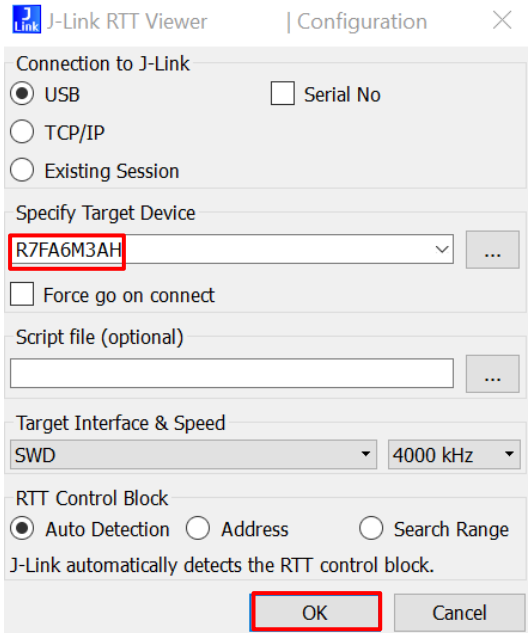

Figure 11. Open J-Link RTT Viewer

You will also notice the RTTViewer output messages which match the terminal events with some details on the commands and responses.

```
00> Enabling framed data protocol over USB
00> Connecting the kit to Host PC...
00> USB CDC instance created. Run the Device Id Host application!!!
00>
00> Received WRAPPED_KEY_REQUEST request
00> hwCreateKey successful
00>
00>  Received WRAPPED_KEY_CHALLENGE_RESP command
00> ChallengeResp successful. Responding with signed Hashed data
00>
00>  WRAPPED_KEY_CERT_PROGRAM request received
00> Successfully wrote the Device Cert in the flash
00> WRAPPED_KEY_CERT_PROGRAM cmd successful !!!
00>  Flash write validation of Device Cert succeeded
```

Figure 12. JLink RTT Viewer Messages

At this stage, the host application communicates with the target through the USB-CDC communication interface.

The user application does the following tasks as shown in section 4.3:

1. Scans for the USB COM port provided by the user. If found, 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 SCE 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 the FAW.

5.4 Customizing the PC Application Project

To customize the PC application, download the Visual Studio development environment using the software 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 it as desired from this point onward.

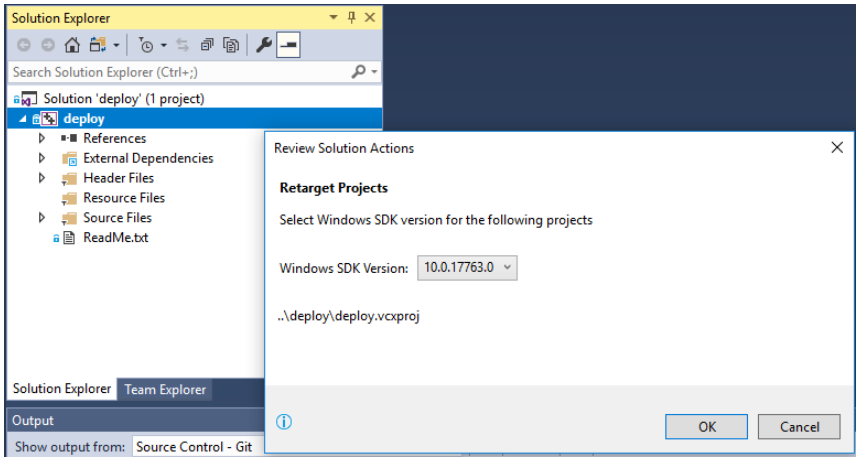

Figure 13. Retarget to the installed Windows SDK

6. References

Available on www.renesas.com:

- Renesas RA Family RA6M3 User's Manual: Hardware (R01UH0886)
- RA Flexible Software Package Documentation (FSP)

7. Appendix

7.1 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. (https://csrc.nist.gov/projects/hardware-roots-of-trust)
SCE	Secure Crypto Engine – 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,

Term	Meaning
	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.
Non-Secure code	A function or group of functions resides in a non-secure region of internal flash or SRAM. Please reference the Secure Data at Rest for RA Family to understand how to access and control the non-Secure code.
Challenge String	Randomly generated string at the host application. This string is used by the host application to validate the target's ownership of the private key.
Unique ID	An identification value, unique to each individual RA Family MCU is stored inside the MCU. The SCE uses a unique ID when it wraps a key.

8. Website and Support

Visit the following URLs to learn about key elements of the RA family, download components and related documentation, and get support:

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.30	Jul.12.23	—	Initial release.
1.40	Feb.02.24	—	Update to FSPv5.0.0.
1.50	Oct.03.24	—	Update to FSPv5.5.0.
1.60	Mar.20.26	—	Update to FSPv6.4.0.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

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

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Trademarks

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

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

© 2026 Renesas Electronics Corporation. All rights reserved.