Renesas RA Family

RA Family Device Identity Application

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 MCU groups RA6M3, RA6M2**,** and RA6M1, including different key generation options. Note that the rest of the documentation uses RA Family MCU to refer to the above mentioned MCU groups only. The application example provided in this package uses the Secure Crypto Engineer (SCE) module based on RA6M3 to generate a device identity unique to each device which is securely stored in the internal flash using the Security Memory Protection Unit (MPU) and the Flash Access Window (FAW) hardware features of the MCU.

This application note enables you to effectively use the RA Family SCE modules in your own design. Upon completion of this guide, you will be able to add the RA Family Flexible Software Package (FSP) SCE modules 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 advance uses of the module, are in the RA Family FSP User's Manual and serve as a valuable resource in creating more complex designs.

Currently, the RA Family Device Identity Application is implemented and tested on the EK-RA6M3 kit.

Required Resources

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

Development tools and software

- e² studio ISDE v7.8 or later
- RA Family Flexible Software Package (FSP) v1.0.0 or later
- SEGGER J-link® USB driver V6.64b or later

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 2017 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 10 OS
- Two Micro USB cables

Prerequisites and Intended Audience

This application note assumes you have some experience with the Renesas e² studio ISDE 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 SCE features.

The intended audience are users who want to develop applications with SCE modules using Renesas RA Family MCUs RA6M1, RA6M2, or RA6M3 MCU groups.

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 infrastructural for an operational IoT (Internet of Things) environment consists of the following:

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

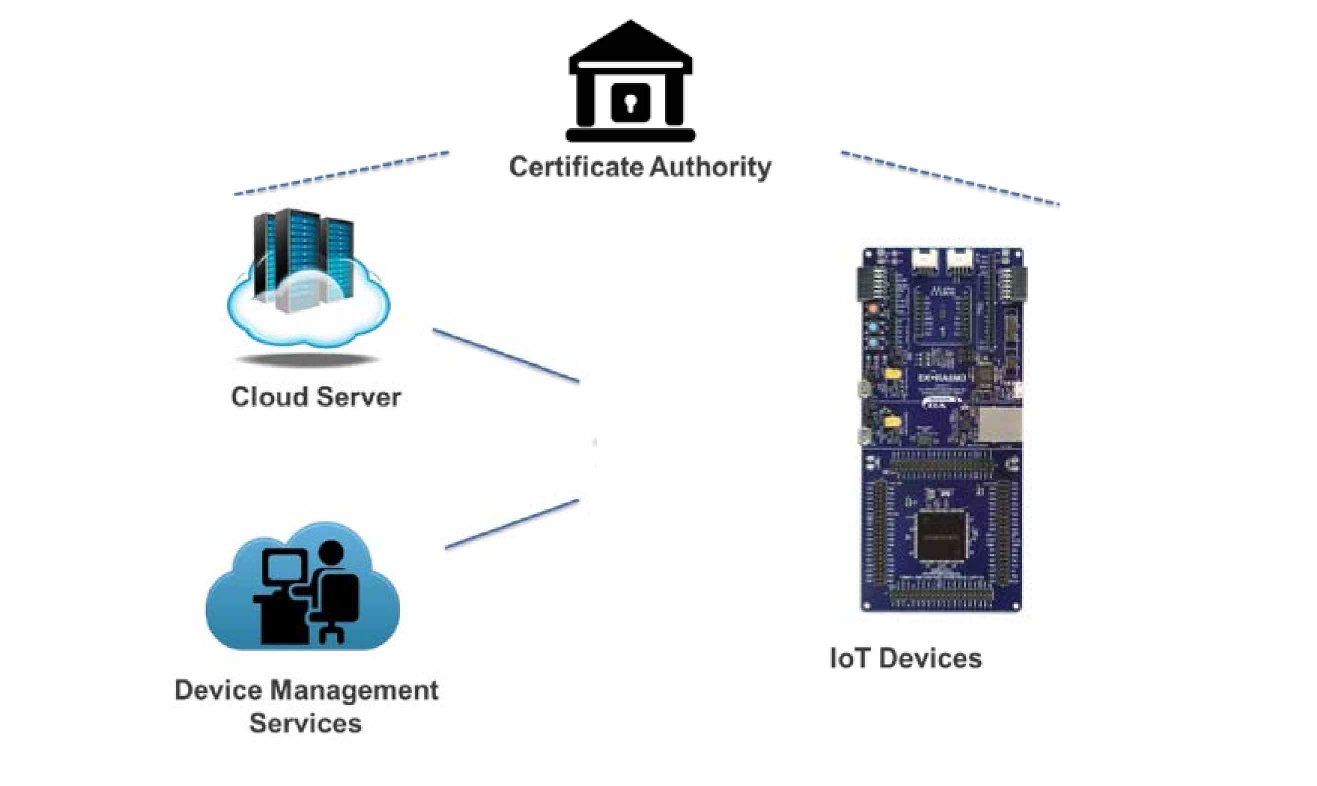

Figure 1. IoT Environment Overview

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 network server for cloud service 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 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 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 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 an establishment of a strong device identity, each IoT device 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 RA Family MCU Hardware Security Features

RA Family MCUs enable hardware root-of-trust mechanisms by providing the ability to protect memory blocks. Using this capability, the protected memory can only be accessed by firmware located in memory regions designated as a secure memory region. These capabilities are provided by the Security MPU. Additionally, the contents of flash memory can be locked from future erase/write events using the FAW. 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.

1.3.1 Security MPU

RA Family MCUs incorporate a security MPU with four secured regions. The four secure regions include individual areas in Code flash, SRAM, and two security function regions. These regions can only be accessed by "secure code". Secure regions are protected from unauthorized accesses by:

- A non-secure program
- Additional bus masters, such as the DMA, DTC
- Debugger interface

This mechanism allows untrusted code to exist and operate alongside trusted code. The security MPU settings are stored in flash and it is activated before fetching the reset vector.

For more detailed information on the Security MPU, please refer to the RA Family MCU Hardware User's Manual and the Secure Data at Rest application project.

1.3.2 FAW (Flash Access Window)

The Flash Access Window 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. User can also refer to 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 inverse of the FAW address range.

FSPR (One Time Programmable Setting)

The FAW register setting can be permanently set using the FSPR bit. If the Security MPU register locations reside outside of the FAW window, the Security MPU setting will also be permanently set. 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.

It is important that this bit is set to prevent the FAW and Security MPU registers from being modified after being deployed. For example, when using the Security MPU, the FAW and FSPR bit must be set to lock the Security MPU register area so the security regions where the device certificate and the secure program reside cannot be accidentally modified.

1.3.3 Secure Crypto Engine Module

The Secure Crypto Engine (SCE) 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. Below is a diagram of the SCE features offered by these MCUs.

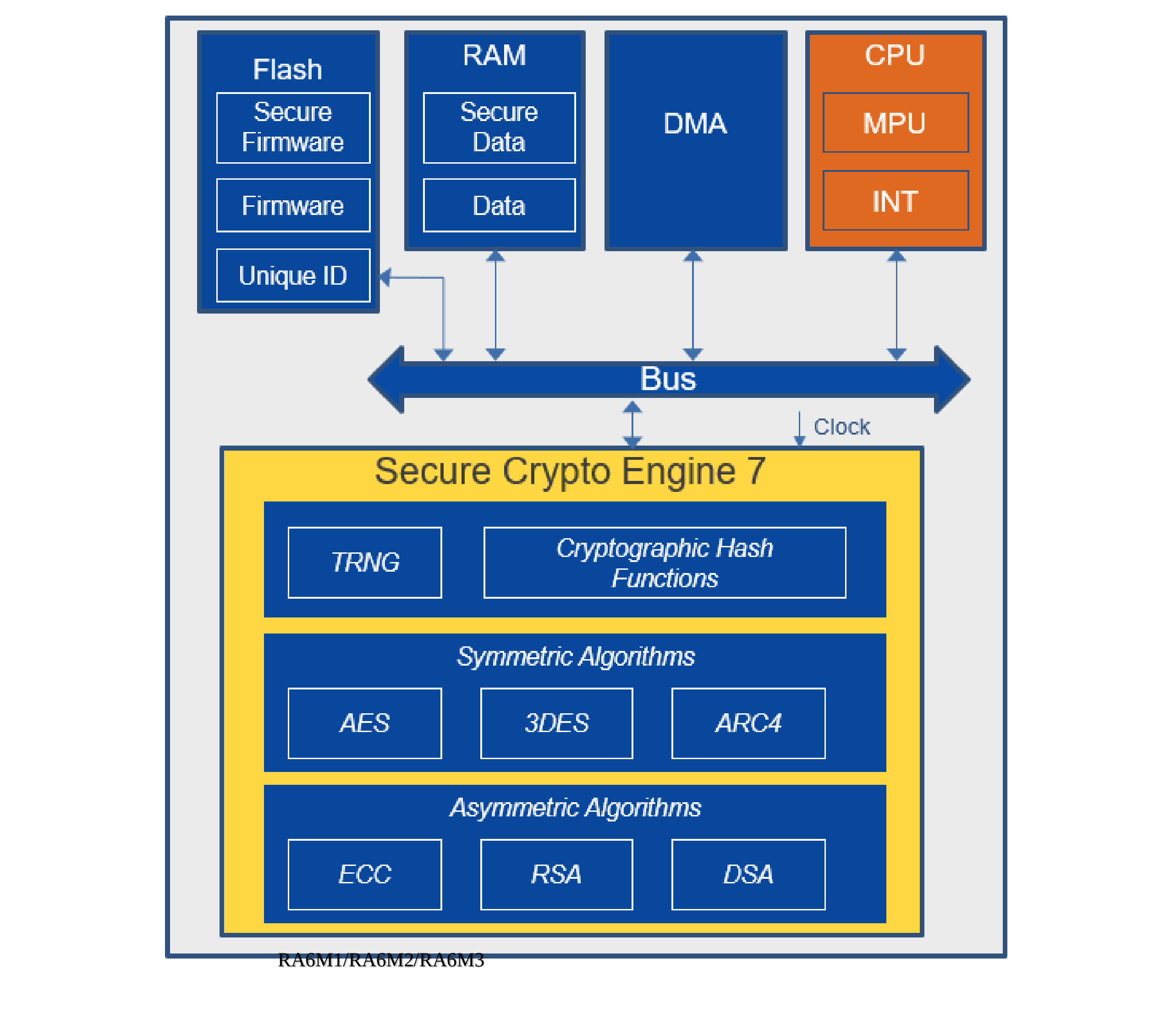

Figure 2. Security Hardware Peripherals Available in RA6M1/2/3

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

1.4 Notes on Arm MPU, Bus Master MPU, Bus Slave MPU

This section explains how the Arm MPU, Bus Master MPU, and Bus Slave MPU relate to Data at Rest design. Refer to the Arm® Cortex technical user manual to understand the definition and settings of the Arm MPU. Refer to RA Family MCUs hardware user's manuals to understand the definition and settings of the Bus Master and Bus Slave MPUs.

While these three MPUs intend to catch inadvertent accesses to the regions defined by these MPUs, they do not provide protection of reading and updating the register settings from non-secure code The register settings of these MPUs are not protected from reading by a debugger nor by non-secure code. Both secure and non-secure code can modify these registers.

In addition, the MPU regions defined by these three MPUs do not provide the same level of security compared to the protection provided by the Security MPU:

- A debugger can access the protected regions
- A read-protected region (without write protection) can be written by secure and non-secure code
- A write-protected region (without read protection) can be read by secure and non-secure code
- A read/write-protected region cannot be read nor written by either secure or non-secure code

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 use of the MCU's unique ID. 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, 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.

2.2 Key Generation in the Device

This 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 (SCE) 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 2048-bit, 1024-bit plaintext public keys in standard and Chinese Reminder Theorem (CRT) format
- RSA 2048-bit, 1024-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, 256-bit plaintext public keys and wrapped private keys.

In this application, ECC `secp256r1` plain-text public key 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.

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 on their approach. 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 theft or tampering. This can be achieved by using the Security MPU and FAW features offered by the RA Family MCU. The features configure a portion of internal code flash as secure code and data regions. The secure code region contains API functions that are only authorized to work on the secure data region. The secure data region contains key information such as device certificates. This section cannot be accessed or modified by any non-secure code running on the RA Family MCU.

The Security MPU settings are locked using the FAW feature (using the one-time programmable FPSR bit) before leaving the secure facility (programming center) to prevent them from being modified.

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.

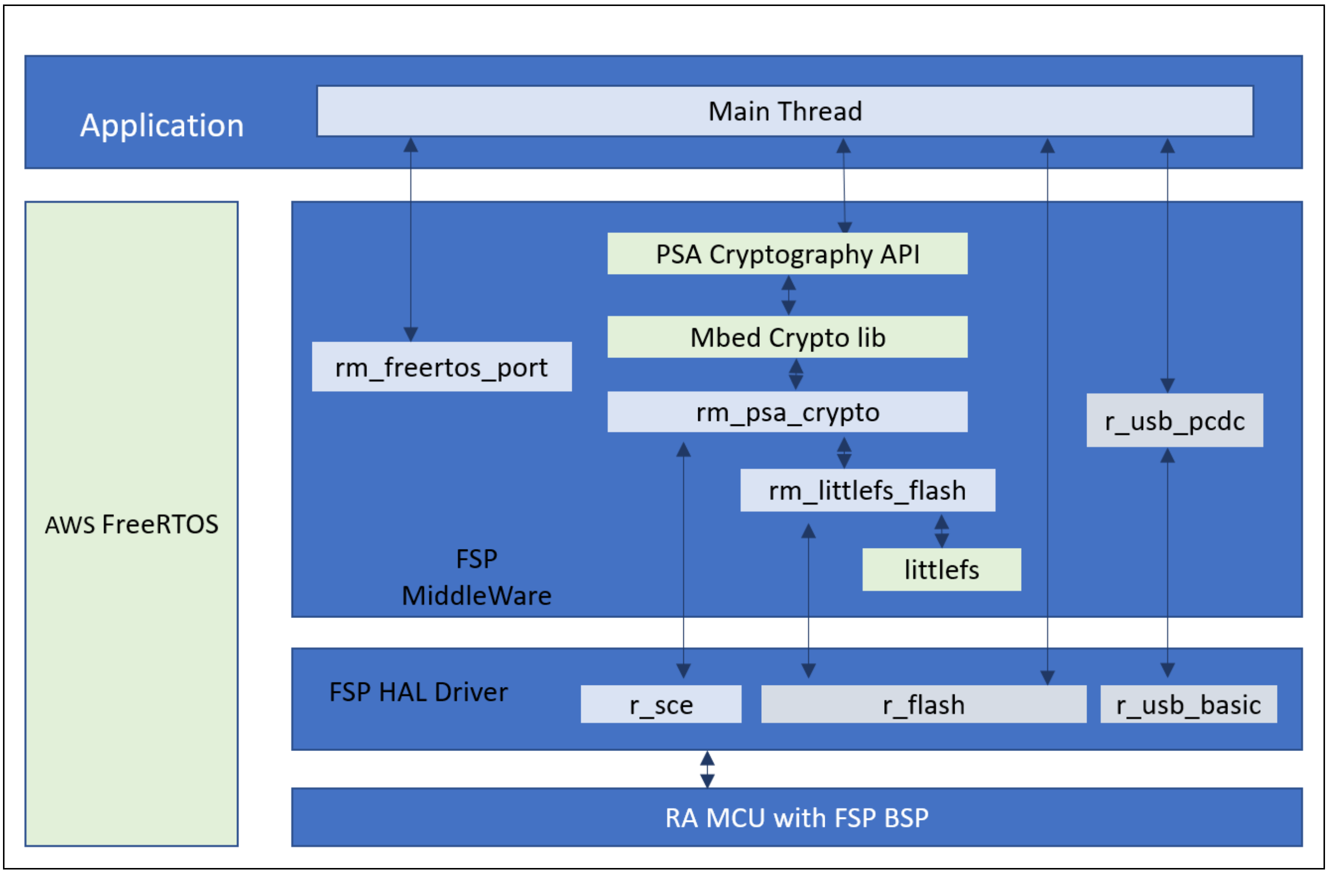

Figure 3. RA Device Identity Application Software Architecture

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 device identify and private key 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. Decodes the command and calls 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

WRAPPER_KEY_REQUEST

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

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 for signing the challenge response.

WRAPPED_KEY_CERT_PROGRAM

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

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: `handleHrkCertChallengeResp ()`

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 will send the device certificate to the device to be stored securely using Security MPU and FAW.

4.3 Operational Overflow

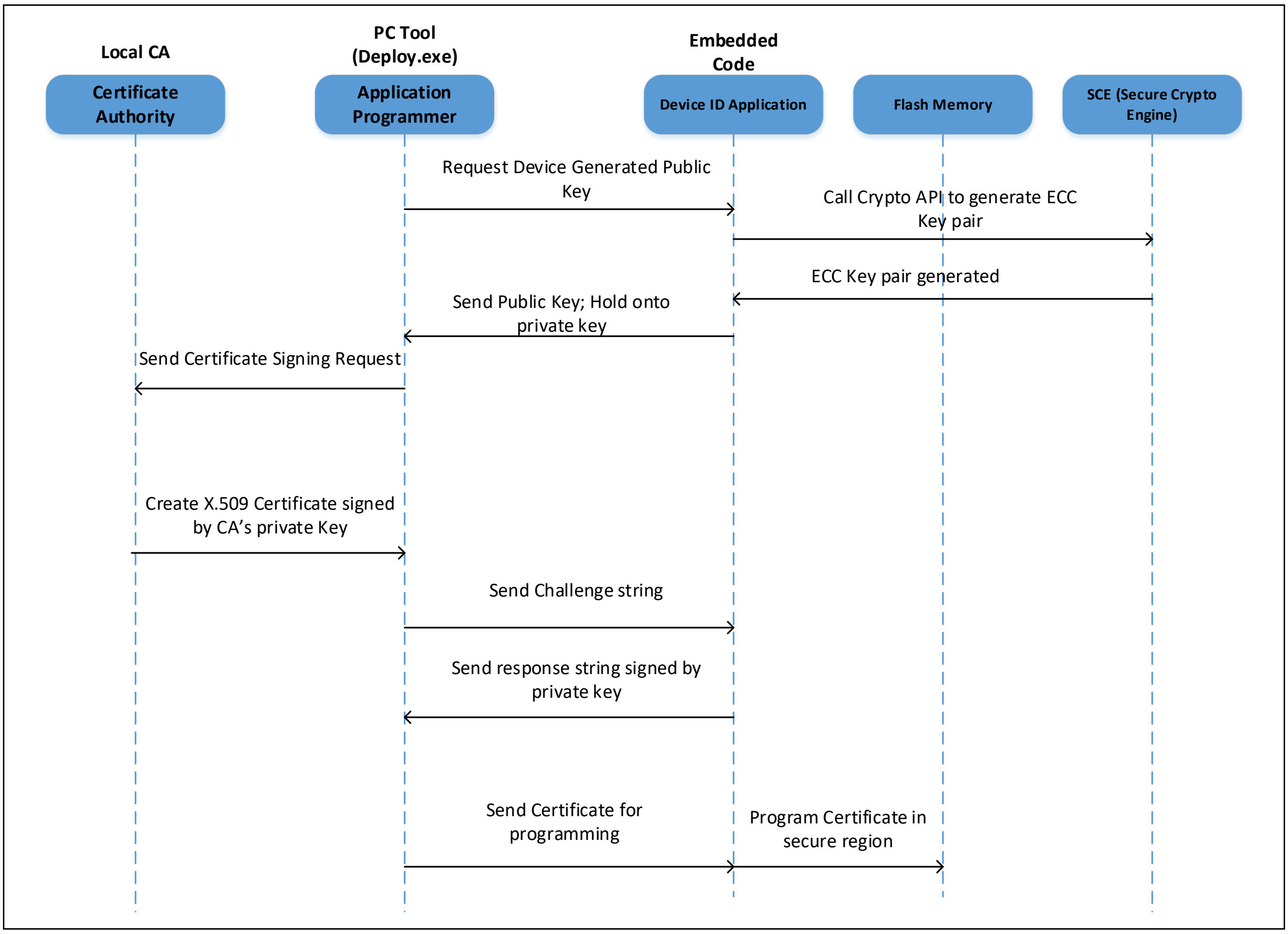

Figure 4. Operational Overflow

This application project consists of two software projects:

- Embedded project running on the EK-RA6M3 kit
- Host application running on Windows (7/10) 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 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 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. 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 EK-RA6M3 kit to be securely stored using the Security MPU and 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 securely stored inside the RA Family MCU using the Security MPU and 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.

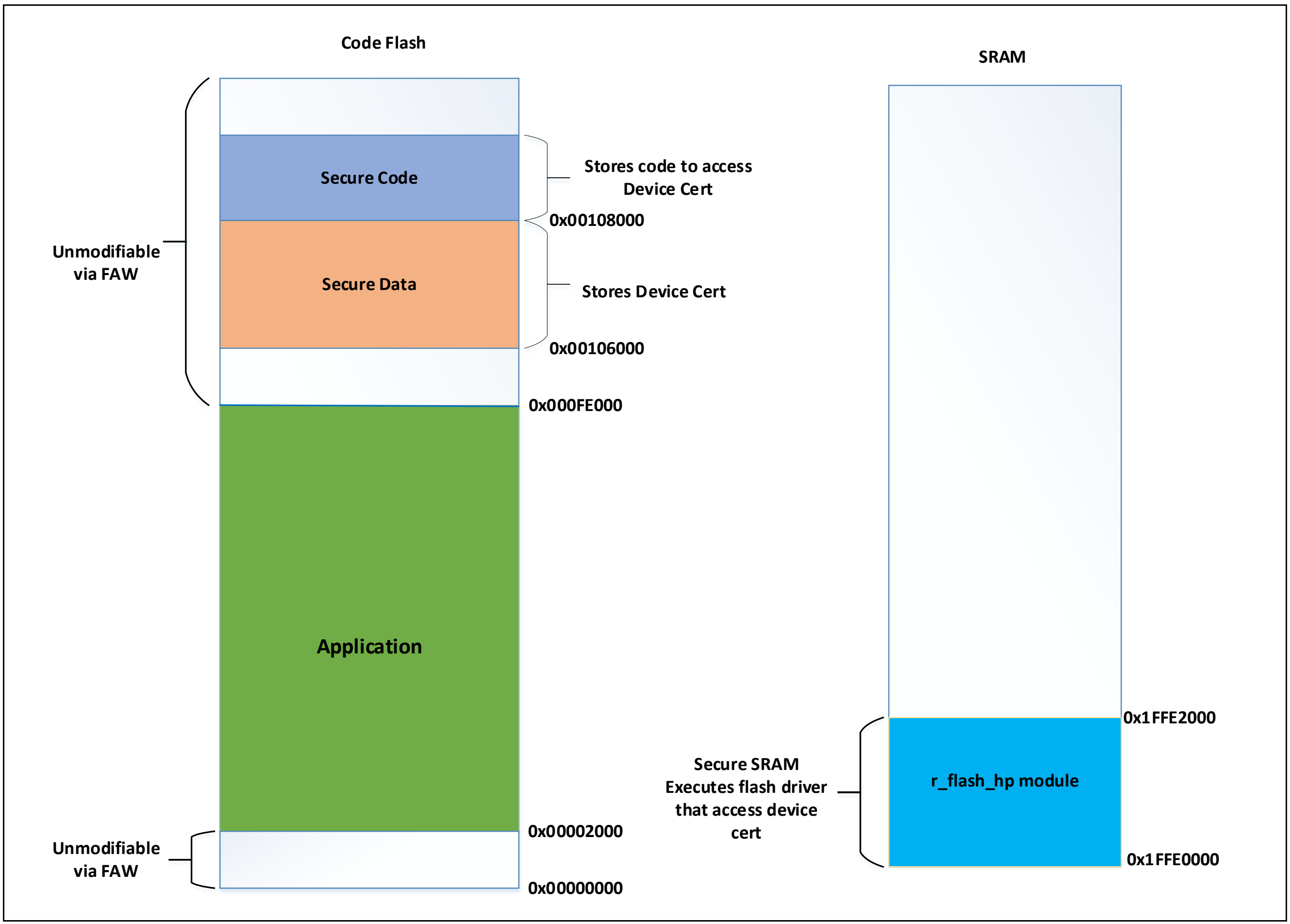

Figure 5. Memory Map used in the Application Project

The left side of Figure 5 shows the memory layout of the code flash in this application project. The green color denotes the area where the application resides, orange color denotes the area reserved for storing the device certificate, and blue color denotes the area reserved for the secure code region that accesses the secure data region.

As shown in the preceding memory map, the address region between 0x0 to 0x2000 contains the Security MPU settings and the address region from 0xFE000 (block aligned address boundary) contains secure data and secure code that needs to be protected from being erased by the user application. To achieve this, the FAW is configured to protect the memory regions from 0x2000 to 0xFE000. By setting the FAW start address to 0x2000 and end address to 0xFE000, the sections outside these address regions are protected from being modified. See the flash_FAW_Set() API found in the `RA_Device_Identity_Solution\embedded\common\src\framedProtocolTarget.c` file, which implements the FAW settings.

The right side of Figure 5 shows the SRAM region memory map in this application project. The FSP flash driver module (`r_flash_hp`) is mapped to the bottom portion of the SRAM that has been configured as a secured SRAM region. This is needed to allow the flash driver to erase and program the secure data region while loading the device certificate.

See the linker script associated with the example application project to see how these secured data/code/SRAM regions are allocated and mapped. Also see the RA Family configurator settings as shown in Figure 6, used in the application project for details on how to configure a region in internal code flash as a secure region. The user can use this example as a reference in their design when allocating a specific region in code flash/SRAM as a secured location for storing the device identities and other application specific modules such as protected IP or third-party licensed firmware.

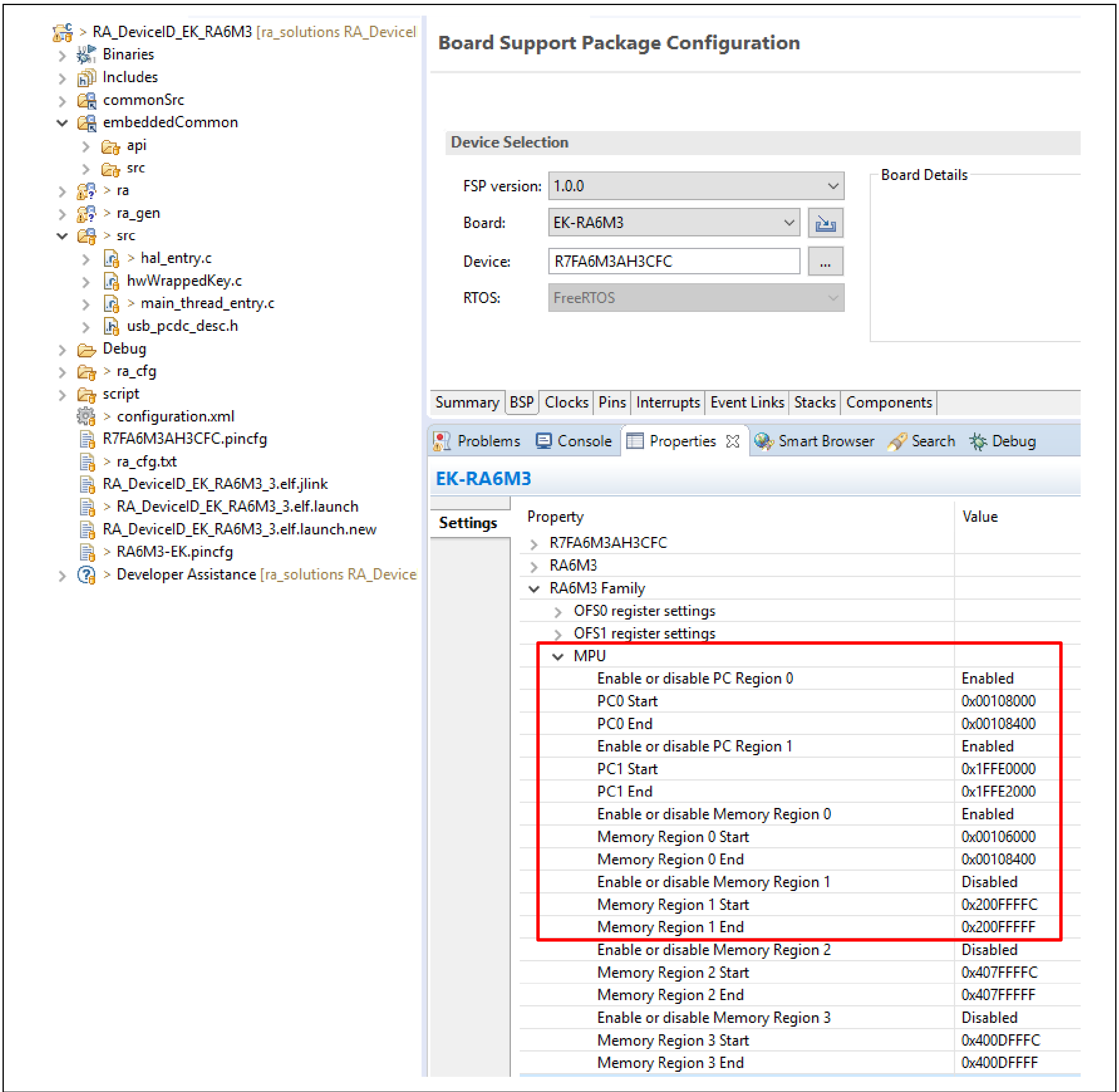

Figure 6. Security MPU settings using e² studio ISDE Configurator

5. Running the Device Identity Application Example

5.1 Importing, Building, and Running the Embedded Project

The embedded projects are included in the folder `RA_Device_Identity_Solution\embedded`. The following instructions will show the user how to import these projects in their e² studio workspace.

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

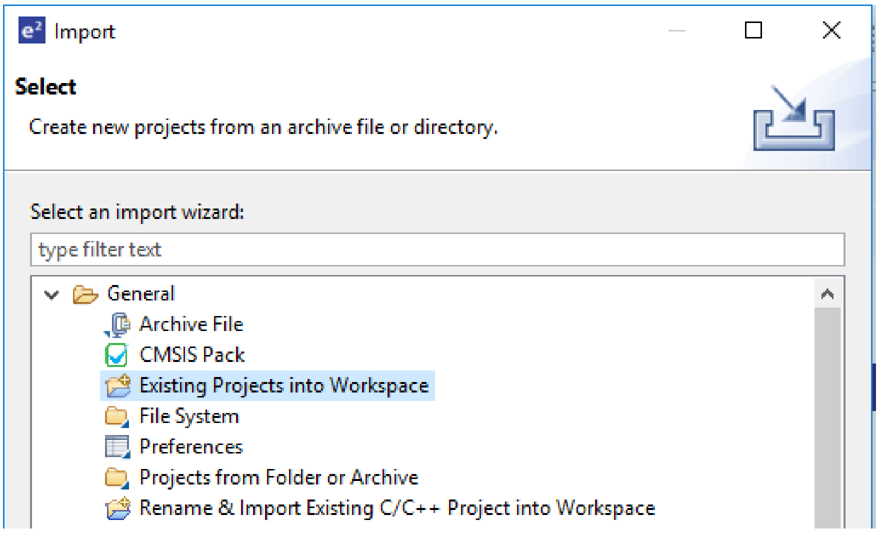

Figure 7. Importing the Project

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

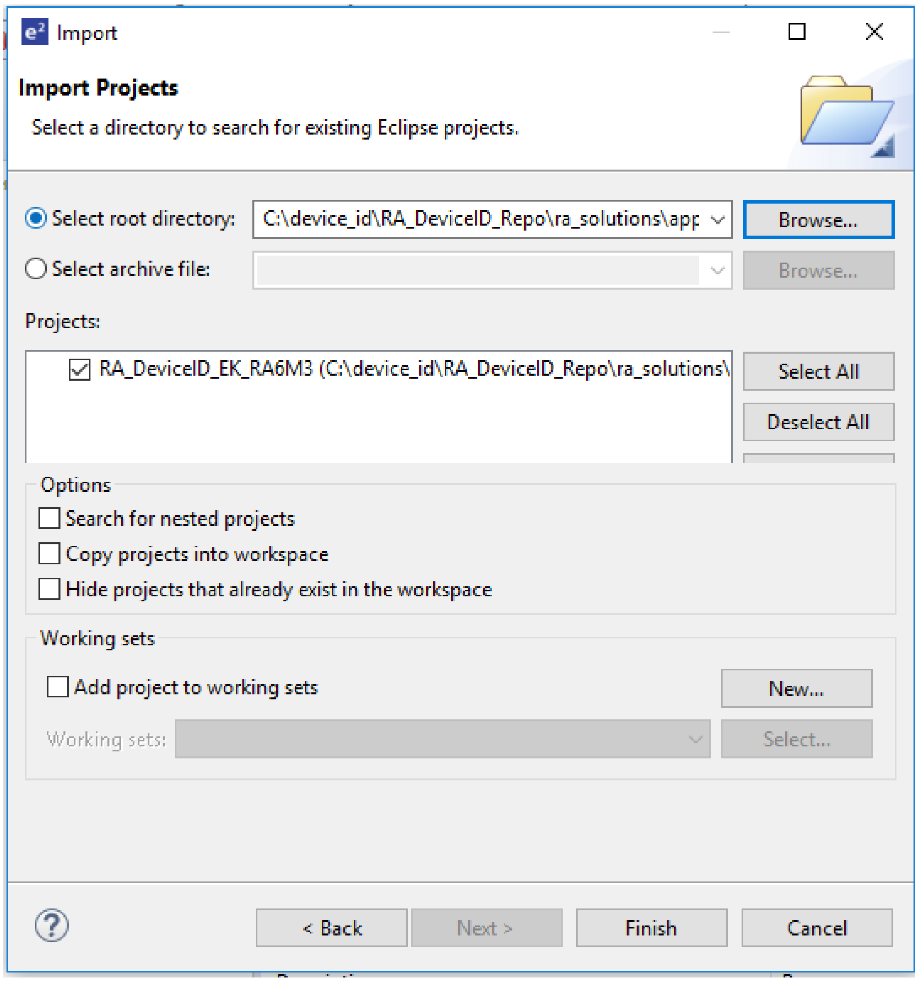

Figure 8. Selection for Importing the Embedded Project

After importing the project, open the RA configurator, click **Generate Project Content** and then compile the project. The project should compile with new errors.

Notice that there are over 600 warnings after compilation. These warnings are from third-party software components. Renesas are not responsible for fixing these warnings.

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. 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 Debugging using ITM printf

In the embedded application project given in this package, there are number of `printf ()` statements to output information about the system. Rather than using the **Renesas Debug Console** in e² studio that uses semi-hosted printf that is intrusive into the real-time execution of the RA processor, the project is setup to use printf via SWO. The output of the SWO printf can be captured and viewed directly within e² studio using the **Live Trace Console** window. However, the performance of the **Live Trace Console** is poor for capturing printf output, as every packet (that is, character) that is captured from SWO is time stamped and saved into a trace file in the project. Instead, the SWO Viewer that is shipped with the J-Link software tools can be used to display the output of the SWO printf with much better performance.

5.4 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 run to `main ()`.

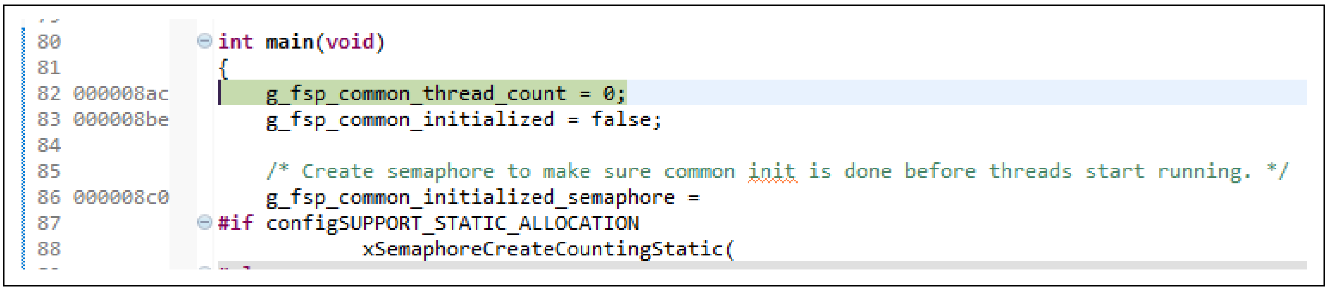

Figure 9. Click "Resume" to reach "main"

Now, open the J-Link SWO viewer. In this example, J-Link SWO V6.64b is used. We recommended using J-Link SWO V6.64b or later versions. Note that the CPUClock setting might have minor variations from the value shown below, it is normal and expected. Click **OK** at the window below.

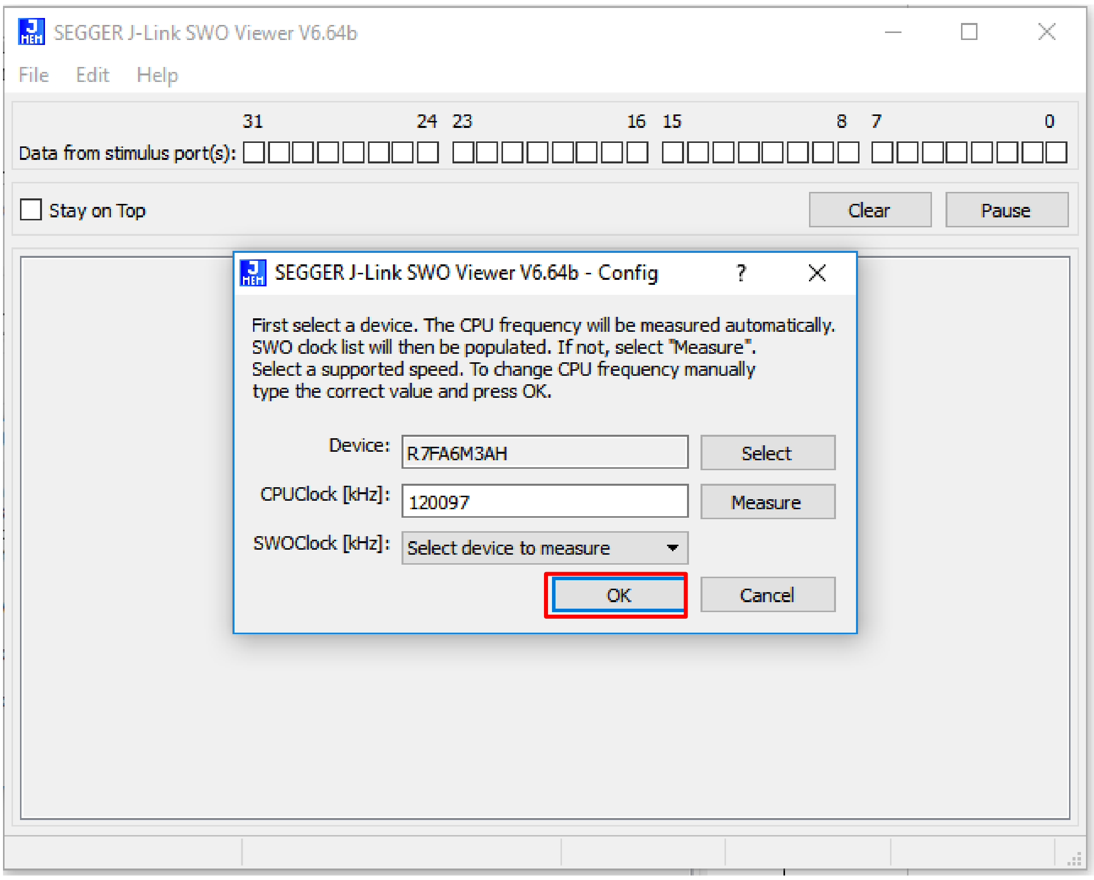

Figure 10. Open J-Link SWO Viewer

Click **Resume** in the e² studio debug window to run the application. The SWO Viewer will print the following message.

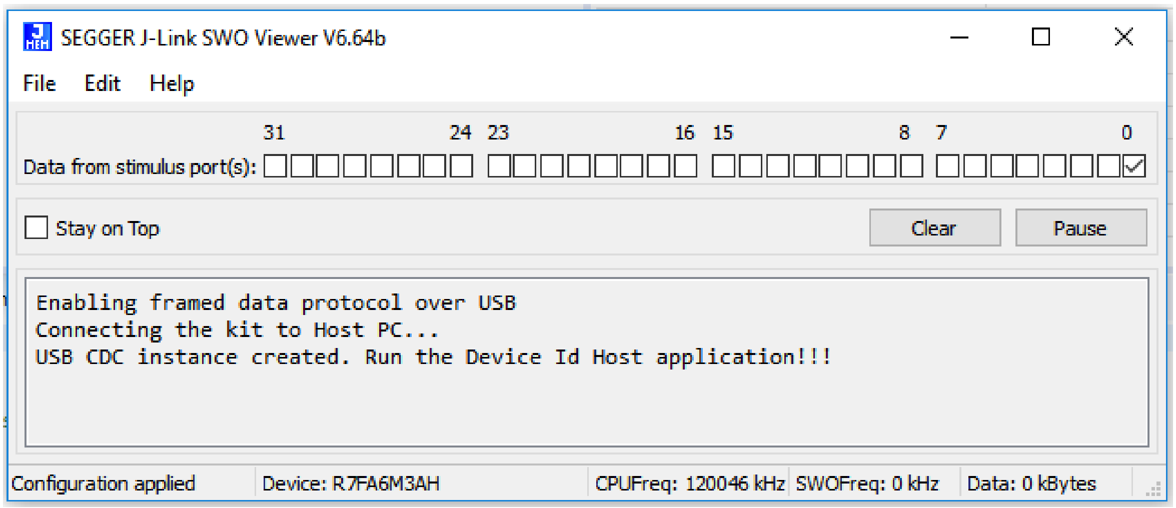

Figure 11. J-Link SWO Viewer 'Measure' Finished

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.

Now, run the host application on the windows PC. To run the 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_Identity_Solution\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>
```

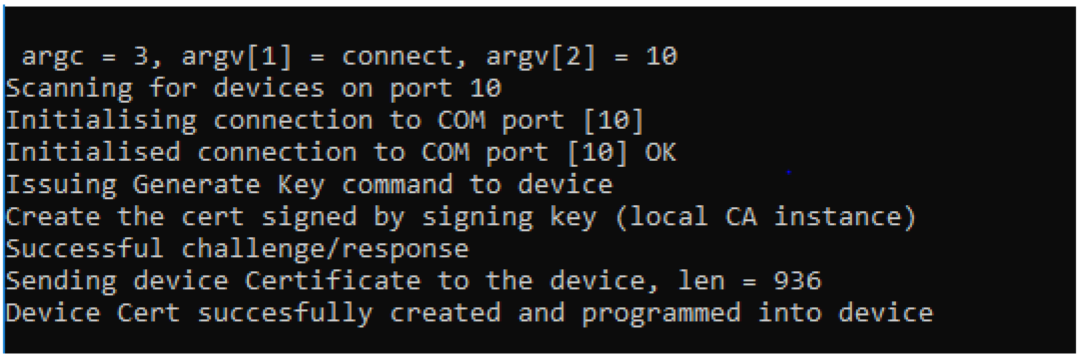

Figure 12. Host application console messages

You will notice the console messages from the host application which matches the above operation with some details on the commands and responses.

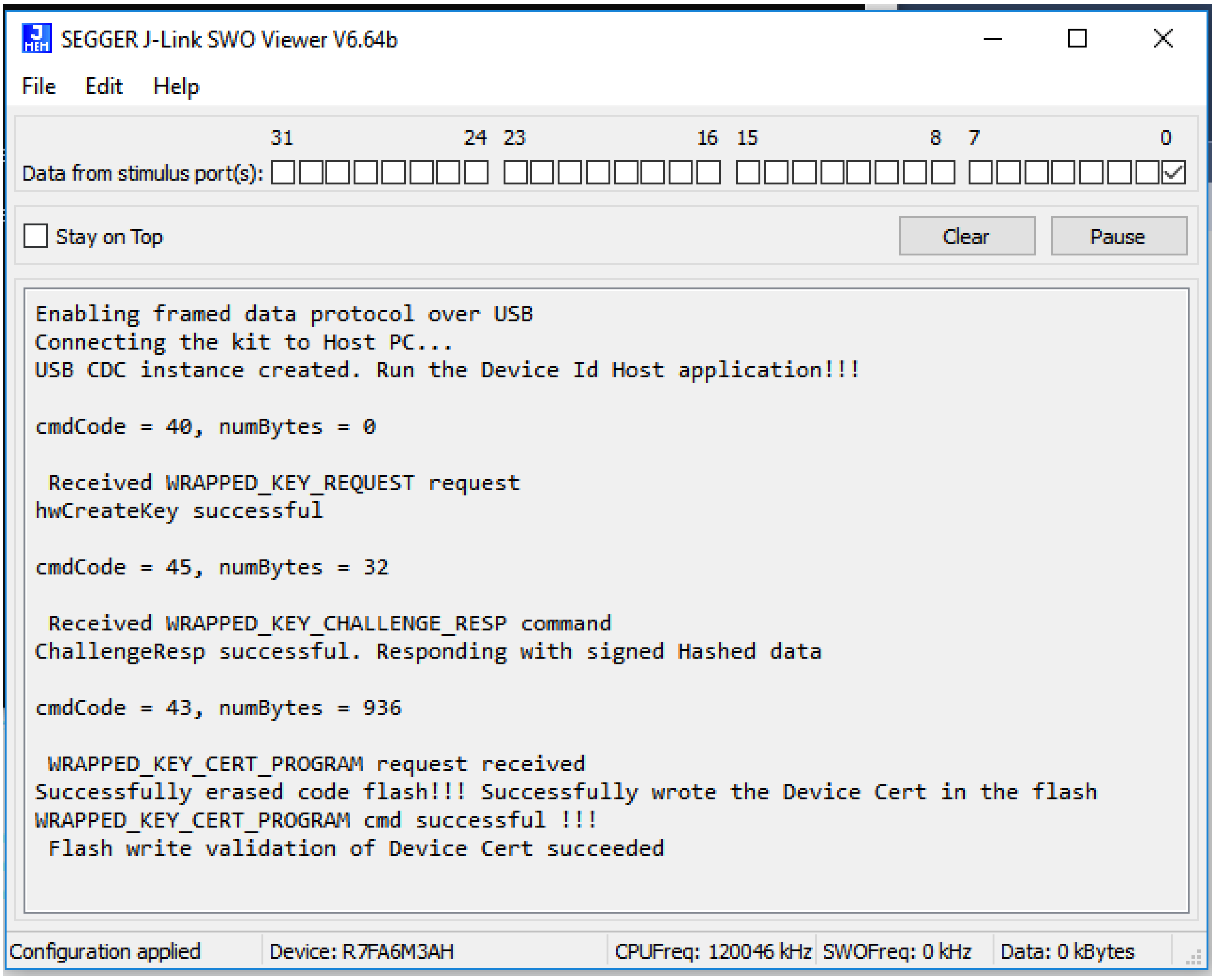

Figure 13. Console Messages on Target Kit

At this stage, the host application starts communicating with the target through the USB-CDC communication interface. 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 Security MPU and FAW.

5.5 Customizing the Demonstration

5.5.1 Migrating to RA6M2 and RA6M1 MCUs

Renesas RA6M1 and RA6M2 MCU groups have smaller internal flash and SRAM code space compared with RA6M3. Follow these steps to migrate this application project for RA6M1 and RA6M2:

1. Start from the RA6M3 embedded project and switch the Board to intended RA6M1 or RA6M2 kits.
2. Modify the BSP Security MPU region settings to fit the memory size of MCU selected.
3. Rename `RA6M3.ld` to the intended linker script for the new kit.
4. Modify the linker script to take care of the new secure flash and SRAM regions defined in the BSP tab.
5. Modify the FAW region range if needed in the application code based on the flash size of the new MCU.
6. This entire application embedded code size fits any part from the RA6M1 or RA6M2 family. In addition, all the peripheral and core functionality used in this application project are compatible with RA6M1, RA6M2 and RA6M3. No application code update is needed when migrating to RA6M1 and RA6M2.

5.5.2 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.

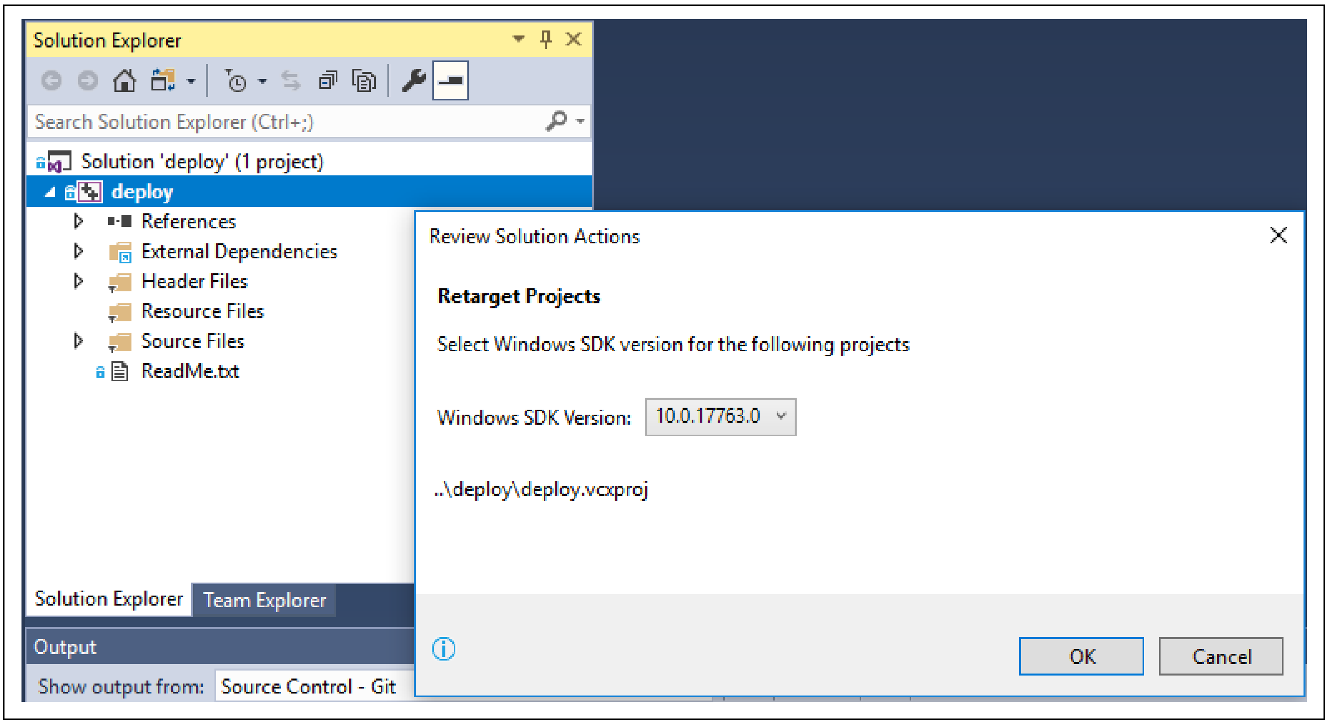

Figure 14. Retarget to the installed Windows SDK

6. References

Available on www.renesas.com:

- FSP v1.0.0 User Manual
- Secure Data-at-rest application project
  - AN number: r11an0416eu

7. Known Issues and Limitations

The host application is tested only on Windows (7/10) PC.

8. Appendix

8.1 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 evice. 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), and 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, as defined and enforced by the MPUs. These secure functions can access both secure data and non-secure data regions.
Non-Secure code	A function or group of functions that resides in a non-secure region of internal flash. These non-secure codes cannot access the secure region. They can only access the non-secure region.
HRK	Hidden Root Key. This is a unique key stored inside the RA Family MCU.
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 SCE when it wraps a key.
Challenge Response String	The response to the challenge string. The Challenge Response String is the signature of the challenge data as created by signing the Challenge String with the receiver's private key.

9. Website and Support

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

EK-RA6M3 Resources	renesas.com/ra/ek-ra6m3
RA Product Information	renesas.com/ra
Flexible Software Package (FSP)	renesas.com/ra/fsp
RA Product Support Forum	renesas.com/ra/forum
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May 6, 2020	-	Initial version

All trademarks and registered trademarks are the property of their respective owners.

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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

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

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

© 2018 Renesas Electronics Corporation. All rights reserved.

Colophon 7.2