Renesas RA Family

Securing Data at Rest Using the Arm® TrustZone®

Introduction

Securing Data at Rest refers to the features and services provided to protect sensitive data residing on a device, which may or may not be modifiable. This application project discusses the considerations for securing **Data at Rest** in an embedded system and provides guidelines on how to use the Arm® TrustZone® hardware feature of the RA Family MCUs to implement a secure Data at Rest solution.

RA MCUs offer data encryption, authentication schemes, and read/write and write-once access protection from CPU and bus masters for secure Data at Rest designs. This application project focuses on providing guidelines on securing data at rest application design using TrustZone® to provide read/write protection. For using Security MPU and Flash Access Window for Securing Data at Rest, see section 5 for the corresponding application project.

TrustZone® provides hardware infrastructure for memory and peripheral access isolation. Sensitive data can be stored in Secure memory spaces and can only be accessed by Secure software. Non-secure software can only gain access to those Secure memory spaces by making use of Non-Secure Callable (NSC) APIs that are created specifically to provide services to the Non-secure domain.

Upon completion of this guide, you will be able to use the TrustZone® feature supported by e² studio and the Flexible Software Package (FSP) efficiently in securing the Data at Rest in your own application. More detailed hardware feature and API descriptions are available in the Renesas RA6M4 Group User's Manual: Hardware and the Flexible Software Package (FSP) User's Manual.

The example project included in this application project uses EK-RA6M4 to provide several demonstrations of the TrustZone® technology regarding Securing Data at Rest.

Required Resources

Development tools and software

- The e² studio IDE v2024-01
- Renesas Flex Software Package (FSP) v5.2.0
- SEGGER J-Link® USB driver
- 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
- SEGGER J-Link V7.94g or later (segger.com/downloads/jlink/)
- Renesas Flash Programmer v3.14.00 or later (renesas.com/us/en/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html)

Hardware

- EK-RA6M4, Evaluation Kit for RA6M4 MCU Group (renesas.com/ra/ek-ra6m4).
- Workstation running Windows® 10.
- One USB device cables (type-A male to micro-B male).

Prerequisites and Intended Audience

This application project assumes you have some experience with the Renesas e² studio IDE and FSP. Before you perform the procedures in this application note, follow the FSP User's Manual to build and run the Blinky project. Doing so enables you to become familiar with e² studio and the FSP and validates that the debug connection to your board functions properly.

The intended audience are users who are interested in Securing the Data at Rest in their application with RA MCUs using Arm® TrustZone®.

Contents

1. Secure Data Overview

With the dawn of AI, IoT, and Cloud connectivity, digital data security has become the number one priority when protecting trade secrets and personal privacy.

Secure data technology includes **Data in Transit** and **Data at Rest**. **Data in Transit**, or data in motion is data actively moving from one location to another, such as across the internet or through a private network. **Data in Transit protection** is the protection of data while it is traveling from network to network or being transferred from local storage to cloud storage. **Data at Rest** is data that is not actively moving from device to device or network to network, such as local data stored on embedded flash, SRAM which are not actively being transferred in and out of the MCU. **Data at Rest protection** aims to secure sensitive data stored on any device or network. This application project focuses on **Data at Rest design** in an embedded environment using the Arm® TrustZone® and Flash Block Protection feature of an RA MCU.

Data at Rest protection uses **Data Encryption** and **Data Access Control** as major security measures. This application project provides reference Data Access Control designs to establish the Write Once and Read/Write Access control for a RA MCU. Data Encryption is not covered in this application project.

1.1 Sensitive Data at Rest Topology

In an embedded system, sensitive data can reside in volatile data storage (MCU's internal SRAM or external SDRAM) or non-volatile data storage (such as MCU's internal flash storage, external QSPI storage, and external EEPROM storage). As part of the application security design, user must consider the topology of the data based on its use case. As an example, in a medical device, some data (like blood pressure measurement taken every 5 minutes) can be stored in volatile memory while other types of data (daily blood pressure averages) may need to be stored in non-volatile memory for future use. User should consider the nature of the data and therefore determine its topology before beginning the design as this decision will have an impact on securing the data.

This application project focuses on the methodology to secure the internal SRAM and internal flash.

1.2 Data at Rest Security Measures

Encryption and **Access Control** are two of the main secure Data at Rest protection schemes that will be discussed in this application project. These two schemes apply to both volatile and non-volatile storage types. Access control for internal storage (both volatile and non-volatile) will be covered with example projects.

1.2.1 Data Encryption

Data Encryption is widely used in both secure **Data at Rest** and **Data in Transit**.

Securing internal data through encryption is desired for small Arm® Cortex® MCUs as these devices are used more and more in networking and communication applications. The Secure Crypto Engine 9 (SCE9) feature of RA MCUs are, for example, is used to generate wrapped keys and perform cryptographic operations.

An example of using encryption of Data at Rest is encryption of data in external storage. An embedded system could use an AES key to encrypt sensitive data and code that resides on the external storage. Upon successful authentication, the external code data can be decrypted and used.

1.2.2 Data Access Control

Increased demands for device connectivity as well as increased complexity in embedded systems result in more potential attack surfaces being exposed. Controlling access to the secure data effectively reduces the attack surface, thus increasing system security. The following is a brief introduction to possible use cases where access controls provided in RA TrustZone® enabled MCUs can be applied.

1.2.2.1 Read and Write Protection

With the TrustZone® technology, sensitive data and code residing in the secure SRAM have read and write protection such that only software granted with permission can access them. Sensitive data and code residing in the secure flash are protected from unauthorized read.

For RA Family MCU, it is possible to lock selected flash blocks from erasing and programming by secure or non-secure software. This locking can be configured as temporary or permanent locking.

1.2.2.2 Write-Once and Read Protection

In some use cases, sensitive **Data at Rest** needs to be protected from access or alteration for the lifetime of the device. For example, a secure bootloader must be immutable for the lifetime of the product. As explained in section 1.2.2.1, RA Family Arm® TrustZone® enabled MCUs have flash block locking functionality for temporary or permanent locking. When the permanent locking is applied to the corresponding secure flash block, it is an effective write-once protection with read protection.

Secure bootloader is a typical type of data at rest assets to protect from attack. Specific handling for protecting the secure bootloader can also involve authentication, encryption, and so forth. However, these handlings are not discussed in this application note.

1.3 Data at Rest Risk Profile and Attack Surface Analysis

To fully consider and design for secure Data at Rest in an embedded environment, one should thoroughly consider the following topics:

1. Consider who will have access to the sensitive data in the embedded system.
2. Consider if the CPU bus can access the sensitive data.
3. Consider if other bus masters can access the sensitive data. If so, determine which peripheral the bus master connects to, and what entity this peripheral communicates with.
4. Consider if the debugger can access the sensitive data.
5. Consider the robustness of the application design such that there are measures taken against the application itself to prevent accidental damage to the sensitive data by overwriting the security policies and measures in place.

Reducing the attack surface helps in all the above situations. Securing the entire MCU's memory may not effectively enhance the overall data security, since a larger attack surface translates to a higher chance that hackers will find a weak point. A good guideline for securing sensitive data is to design the application such that only the minimum amount of data is secured, and access is controlled through strategical interfaces.

The analysis of a system's risk profile and attack surface is outside of the scope of this application note; however, the security measures offered by RA MCUs will be introduced to help in reducing the attack surface and minimizing the system's risk profile.

2. Overview of RA MCU Features for Securing Data at Rest

This chapter provides an overview of the RA Family MCU hardware features for securing Data at Rest.

Following is a list of security elements on the RA Family TrustZone® enabled MCUs using RA6M4 as an example.

- Arm® TrustZone®
- Flash Block Locking
- Secure Crypto Engine 9 (SCE9)
- Debug Protection and Serial Programming Control with Device Lifecycle Management System
- Arm® Memory Protection Unit
- Bus Master Memory Protection Unit

2.1 Hardware Enforced Separation based on TrustZone®

Arm® TrustZone® technology for ARMv8-M is an optional Security Extension that is designed to provide a foundation for improved system level security in a wide range of embedded applications. TrustZone® enables the system and the software to be partitioned into Secure and Non-secure worlds. Secure software can access both Secure and Non-secure memories and resources, while Non-secure software can only access Non-secure memories and resources.

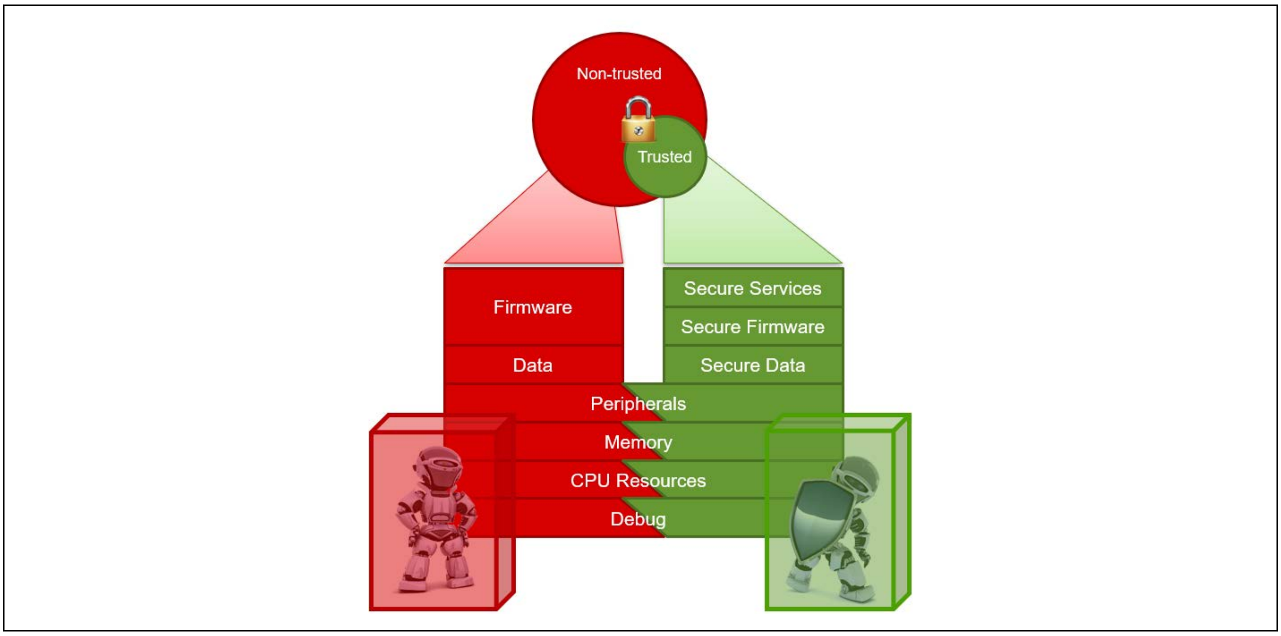


Figure 1. TrustZone® Isolation

To ensure a Secure platform, the security arrangement is not limited to processor architecture. RA Family Arm® TrustZone® enabled MCU entire system-level design is enhanced to address security requirements.

2.1.1 Memory Separation

The code flash, the data flash, and the SRAM are divided into Secure (S), Non-secure (NS) and Non-secure callable (NSC) regions. These memory security attributions are set into the non-volatile memory by the serial programming command when the Device Lifecycle is Secure Software Development (SSD) state. See section 2.2.1 for the definitions of the RA Family Device Lifecycle State definitions for the RA Family TrustZone® enabled MCU.

RA6M4 Group MCUs incorporated IDAU (Implementation Defined Attribution Unit) to configure the secure regions. Following is a summary of the 8 available regions:

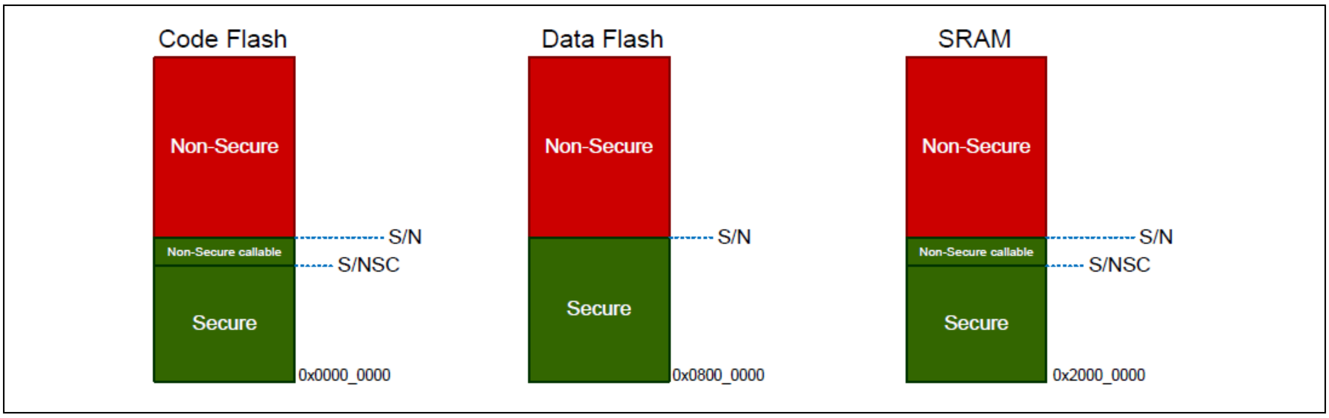


Figure 2. Secure and Non-secure Regions

Sensitive Code and Data should be in the Secure regions so direct access to the Secure regions is protected from Non-secure software access. Non-secure Callable can be used to provide minimum access to the Secure region from the Non-secure region.

Security Feature of Renesas IDAU Region Setup

- IDAU region can only be set up in boot mode using SCI or USB.
- Not possible to be manipulated during remote update (for example, OTA).

Code and Data Flash

Note that upon MCU power-on-reset, the Security Attribute of Flash Programming and Erasing (P/E) mode entry is set to Secure. This means the P/E mode can only be entered from Secure software control. The Secure region flash driver module can be made Non-secure Callable to allow Non-secure regions to set the flash P/E entry mode. Table 1 summarizes the secure flash read and write access violation error reporting scheme.

Table 1. Secure Flash Region Read/Write Protection

Access Violation	Error Report
Flash read	Arm® TrustZone® Secure Fault: Reset or NMI
Flash P/E mode entry	Flash P/E Error Flag: handled by FSP flash driver

SRAM

SRAM memory, such as SRAM0 include ECC region and Parity can be divided into Secure/Non-secure callable/Non-secure status with Memory Security Attribution (MSA) and can be protected from Non-secure access. When MSA indicates that SRAM memory region is Secure or Non-secure callable status, Non-secure access can’t overwrite them. Table 2 summaries the secure SRAM read and write access violation error reporting scheme.

Table 2. Secure SRAM Region Read/Write Protection

Access Violation	Error Report
SRAM read	TrustZone® Secure Fault: Reset or NMI
SRAM write	TrustZone® Secure Fault: Reset or NMI

Renesas Tools and FSP provide configurability for selection of TrustZone® Error Reporting scheme, by default NMI is selected in the Board Support Package (BSP). Refer to section 3.1.3 for configuration of TrustZone® Secure Fault handling.

Flash Block Locking

The RA Family TrustZone® enabled MCU can provide temporary and permanent flash block locking for code and data flash Programming and Erasing (P/E) mode entry. The e^2 studio provides configuration options for user to selectively prevent the erasure and programing of the intended flash block.

2.1.2 Bus System Separation

Direct Memory Access Controller (DMAC) and Data Transfer Controller (DTC)

The IDAU region setup is consistent for CPU, Direct Memory Access Controller (DMAC) and Data Transfer Controller (DTC). DMAC and DTC are supervised by the Master TrustZone® Filter. Secure flash and secure SRAM area access violation from Non-secure DMAC and DTC accesses will trigger DMA_TRANSERR interrupt and NMI interrupt. Access to address of access violation will not be granted, thus the Data at Rest can be protected.

Bus Master MPU TrustZone® Feature

The Bus Master MPU is a memory protection function for each bus master except for the CPU. Secure software can set up the TrustZone® Security Attributes of the Bus Master MPU. Access-control information can be set up to 8 regions in DMAC/DTC and 4 regions in EDMAC. If access to a protected region is detected, the bus master MPU generates an internal Reset or a Non-Maskable Interrupt (NMI).

Refer to the Renesas RA6M4 User’s Manual: Hardware and FSP User’s Manual for more details of the Security Attribute control for the bus systems.

2.2 Debug and Serial Programming Security Feature

2.2.1 Device Lifecycle Management System

The Debugging and Serial Programming interface security control of the RA TrustZone® enabled MCU are achieved by the Device Lifecyle Management (DLM) system.

The Device Lifecycle States and the Corresponding Debugging and Serial Programming levels are described in Table 3.

Table 3. Arm® TrustZone® Enabled RA Family MCU Group Device Lifecycle States

Lifecycle	Definition and State Features	Debug Level	Serial Programming	Renesas Test Mode
CM	• **C**hip **M**anufacturing • The device is in Renesas factory. • This is the state when the developer received the device. • MCU Unique ID and Hardware Unique Key (HUK) are injected.	DBG2	• Available. • Cannot access code/data flash area.	Not available
SSD	• **S**ecure **S**oftware **D**evelopment • The secure part of the application is being developed. • **SECDBG_KEY** and **RMA_KEY** can be injected.	DBG2	• Available. • Can program/erase/read all code/data flash area.	Not available
NSECSD	• **N**on-**SEC**ure **S**oftware **D**evelopment • The non-secure part of the application is being developed. • **NONSECDBG_KEY** can be injected. • It is possible to regress to SSD state without flash erase if **SECDBG_KEY** is injected in SSD state. • It is possible to erase the entire flash to SSD state.	DBG1	• Available. • Can program/erase/read non-secure code/data flash area.	Not available
DPL	• **D**e**PL**oyed • The device is in-field. • It is possible to regress to NSECSD state without flash erase if **NONSECDBG_KEY** is injected in NSECSD state. • It is possible to erase the entire flash to SSD state.	DBG0	• Available. • Cannot access code/data flash area.	Not available
LCK_DBG	• **L**o**CK**ed **D**e**B**u**G** • **The debug interface is permanently disabled.**	DBG0	• Available • Cannot access code/data flash area.	Not available
LCK_BOOT	• **L**o**CK**ed **BOOT** interface • **The debug interface and the serial programming interface are permanently disabled.**	DBG0	• Not available.	Not available
RMA_REQ	• **R**eturn **M**aterial **A**uthorization **REQ**uest • Request for RMA. • The customer must send the device to Renesas in this state.	DBG0	• Available. • Cannot access code/data flash area.	Not available
RMA_ACK	• "**R**eturn **M**aterial **A**uthorization **ACK**nowledged" • Failure analysis in Renesas.	DBG2	• Available. • Cannot access code/data flash area.	Available

Device Lifecycle State Transitions

All the available transitions are described in Figure 3.

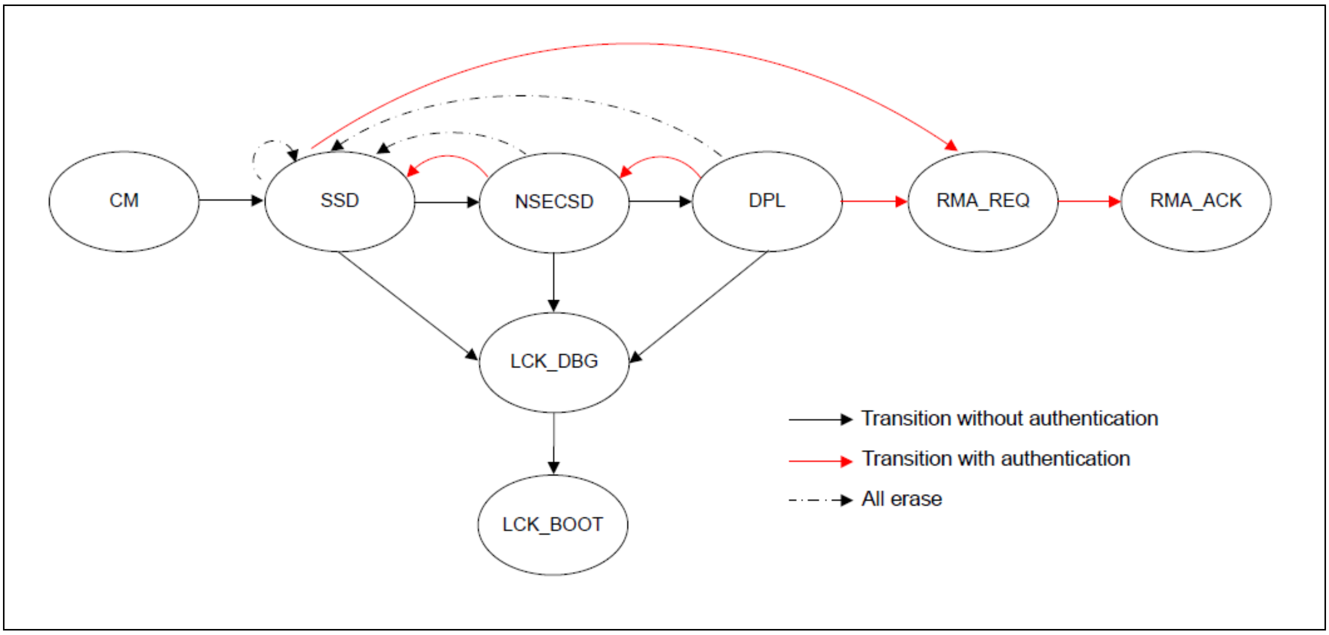

CM
SSD
NSECSD
DPL
RMA_REQ
RMA_ACK
LCK_DBG
LCK_BOOT
Transition without authentication
Transition with authentication
All erase

Figure 3. Device Lifecycle State Transitions

As shown in Figure 3, there are three types of Device Lifecycle state transitions. Each one of these transition types provides secure Data at Rest protection at different stage of the product lifecycle.

- All erase
  - During development.
  - During deployment in DPL state to avoid MCU scrappage and protection of user Data at Rest.
- Transition without authentication
  - Advance the MCU Lifecycle state to create further restrictions to the access level to the Data at Rest from the Debug and Serial Programming interface.
- Transition with authentication
  - Enhanced security control over protected Data at Rest during MCU Lifecycle state regression and failure analysis. See notes below for more details.

Special Notes on the Advantages for the Authenticated State Transition

When the developer requests a DLM state regression to unlock the MCU, the MCU issues a challenge. The developer must then utilize the appropriate DLM state key to construct a response. If the response is correct, the MCU will regress to the requested DLM state, unlocking the defined programming and debugging capabilities. This mechanism prevents eavesdropping and replay attacks, providing enhanced Data at Rest protection while retaining the ability to perform end product failure analysis.

2.2.2 Debugging Security Separation

Debugging and programming interfaces can be completely locked via the Device Lifecycle Management.

RA MCU with Arm® TrustZone® support uses the Device Lifecycle Management system to control three debug levels:

DBG2: The debugger connection is allowed, and no restriction to access memories and peripherals.
DBG1: The debugger connection is allowed, and restricted to access only Non-Secure memory regions and peripherals.
DBG0: The debugger connection is not allowed.

- The MCU can debug the entire secure and non-secure assets when the MCU is in SSD state, which has DBG2 debug level.
- The MCU can debug the non-secure assets only when the MCU is in NSECSD state, which has DGB1 debug level.

  - It is possible to reenable the MCU secure debug interface by regressing from NSECSD state (DBG1) to SSD state (DBG2) via an authenticated process.
- The MCU debug interface is temporarily locked up when the MCU is in DPL state which has DBG0 debug level.
  - It is possible to reenable the MCU debug interface by regressing from DPL state (DBG0) to Non-secure debug state DBG1 via an authenticated process.
    - The MCU debug interface is permanently disabled when the MCU is in LCK_DBG or LCK_BOOT state.

2.2.3 Serial Programming Interface Security

The serial programming interface can communicate with the MCU's factory bootloader via UART or USB. The availability and functionality of the serial programming interface is determined by the device lifecycle states.

- The full functionality of the serial programming interface is available in SSD. In SSD state, the serial programming interface can program/erase/read all code/data flash area.
- In NSECSD state, the serial programming interface can program/erase/read all non-secure code/data flash area.
- In DPL and LCK_DBG state, the serial programming interface is available, but cannot access either Secure or Non-secure code/data flash area. The serial programming interface can still perform state transition and provide answers to queries to the MCU, and so forth.
- The serial programming interface is permanently disabled when the MCU is in LCK_BOOT state.

For more use cases of the Device Lifecycle Management and operational flow to create Device Lifecycle state transitions, see the application note, Installing and Utilizing the Device Lifecycle Management Keys with the link provided in the References section.

2.3 Secure Crypto Engine 9 (SCE9)

The RA Family MCUs with Arm® TrustZone® support integrate Renesas Secure Crypto Engine 9 (SCE9) hardware block to provide cryptographic key generation, data encryption and authentication capability. Following are the encryption and authentication algorithms supported using RA6M4 MCU Group as an example.

2.3.1 Security Algorithm

- Symmetric algorithms: AES
- Asymmetric algorithms: RSA and ECC

Configuration details of the SCE are outside the scope of this application project. Refer to the Renesas RA6M4 Group User's Manual: Hardware and FSP User's Manual for operational details.

2.3.2 Other Crypto Security Features

- HUK (Hardware Unique Key)
  - Provide Key Wrapping and Key Installation services, which can enhance Data at Rest protection via Encryption using the available Security Algorithms.

- TRNG (True Random Number Generator)
  - Provide strong Entropy for all the cryptographic operation to increase Security for Data at Rest protection.

- Hash value generation: SHA1, SHA224, SHA256, GHASH
  - Provide Integrity security control for Data at Rest on the MCU.

- 128-bit unique ID
  - Can be used for the Authenticated Device Lifecycle state transition from DPL to RMA_REQ.
  - Used in conjunction with HUK for Key Wrapping service.

2.4 Notes on Arm MPU and Bus Master MPU

The Arm® MPU is not intended as a security feature in and of itself. It is most beneficial if viewed as a safety feature both from the standpoint of limiting damage due to errant code in a deployed product and forcing safety into product design. RA Family MCU integrated the MPU to the Arm® TrustZone® enabled devices. There can be 8 Arm® MPU regions in the Secure and Non-secure domain respectively. Currently configuration of these regions with FSP module and e^2 studio IDE smart configurator are not supported. Refer to the *Arm® Cortex® technical user manual* to understand the definition and settings of the Arm MPU. Refer to RA MCU hardware user's manuals to understand the definition and settings of the bus master MPUs.

The bus master MPU monitors the addresses accessed by the bus masters in the entire address space (0x0000_0000 to 0xFFFF_FFFF). Access-control information can be set up to 8 regions in DMAC/DTC and 4 regions in EDMAC and monitoring for access to each region is in accordance with this information. If access to a protected region is detected, the bus master MPU generates an internal reset or a non-maskable interruption. The bus master MPU has security attributes that can be set up using Secure code with its security attribute for each of the 8-bus master MPU regions as Secure or Non-secure individually. While the bus master MPUs intend to catch inadvertent accesses to the regions defined by the MPUs, they do not provide protection of reading of the register settings from debugger nor non-secure program.

3. Configuring the Security Elements

This section explains the detailed process to setup the TrustZone® regions and Block Protection Settings.

3.1 Configuring the Arm® TrustZone® Regions

The Arm TrustZone® IDAU regions can be set up via e^2 studio prior to application code execution during boot time. The IDAU regions can also be set up using the standalone Renesas Flash Programmer (RFP) or the Renesas Device Partition Manager, which is integrated with e^2 studio.

3.1.1 Overview the TrustZone® Regions

The RA MCU TrustZone® support utilizes the Implementation Defined Attribution Unit (IDAU) to set up the secure regions. Following is a summary of the possible TrustZone® IDAU regions for the RA MCU Group. The IDAU setting is common for the CPU, DMAC, and DTC.

- Up to three or six regions for the code flash, depending on the bank mode.
- Up to two regions for the data flash.
- Up to three regions for the SRAM.

In addition, RA implements the following security regions:

- Secure or Non-secure region for the Standby SRAM.
- Secure or Non-secure region for the VBATT backup registers.
- Individual Secure or Non-secure security attribution for each peripheral.
- Some peripherals support both Secure and Non-secure security attributions.

3.1.2 Set up the IDAU Regions

Renesas RA TrustZone® enabled MCU depends on IDAU to set up the Secure regions. The IDAU regions can only be set up via SCI or USB boot mode. It is not possible to change the IDAU settings in firmware. As the IDAU region settings are set up before exiting the MCU boot mode, this removes the security vulnerability of possible IDAU region alteration during MCU application execution.

Setting up the IDAU with e^2 studio

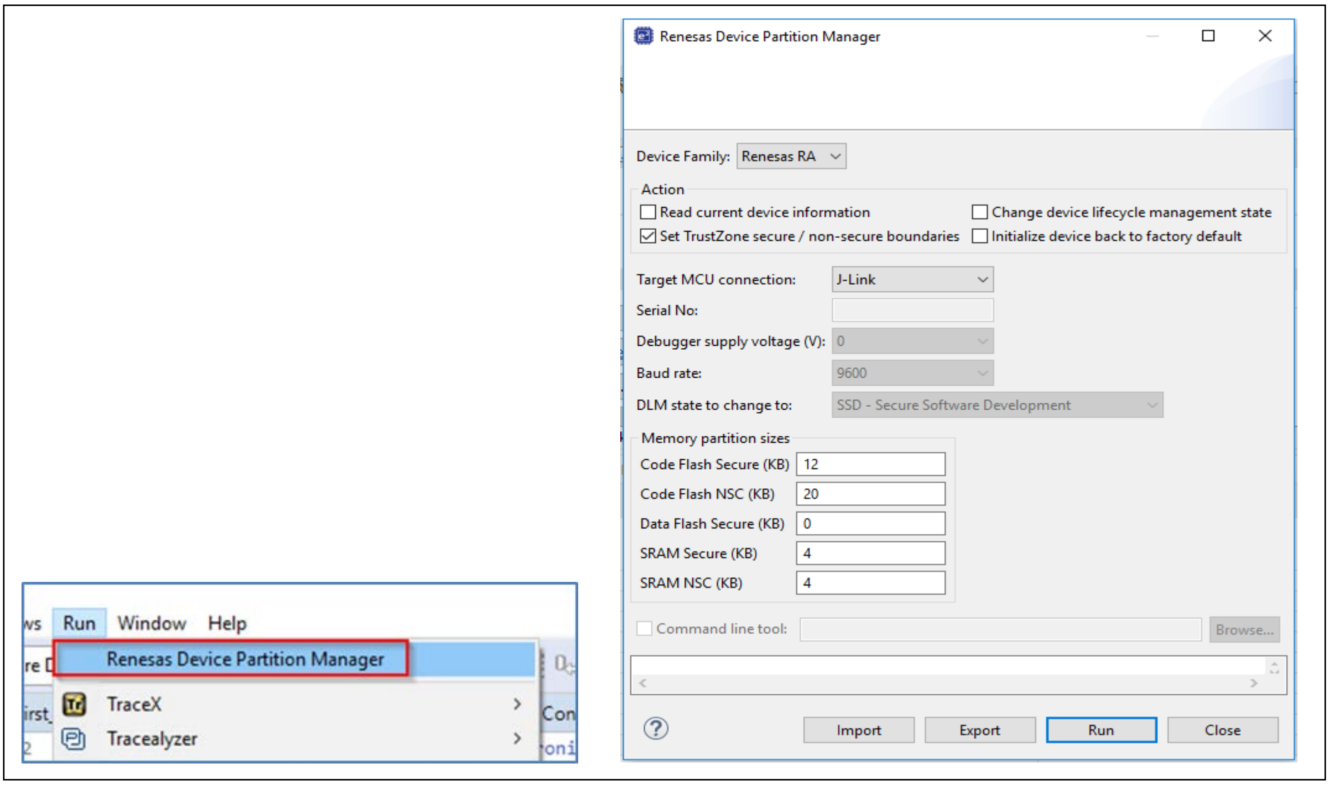
The IDAU region is automatically set up when the first Secure Project is compiled and downloaded to the MCU. There is special handling from both the IDE and the Evaluation Kit's perspective. See *FSP User's Manual section IDAU registers* to understand the details and reference the information for customized hardware system design.

Setting up the IDAU region with e^2 studio can also be performed manually using the following two methods:

- Using the *Renesas Device Partition Manager* which is integrated with e^2 studio.
- Using the standalone **Renesas Flash Programmer** tool.

Manually set up IDAU Region using Renesas Device Partition Manager

Following is an example configuration when using **Renesas Device Partition Manager** to manually set up the IDAU region. Note that the settings provided are for the example project included in this application project. User should adjust these settings based on their specific application.
Note: User needs to power cycle the board prior to work with **Renesas Device Partition Manager** after a debug session if using J-Link as connection interface.



Renesas Device Partition Manager

Device Family: Renesas RA

Action
- □ Read current device information
- ☑ Set TrustZone secure / non-secure boundaries
- □ Change device lifecycle management state
- □ Initialize device back to factory default

Target MCU connection: J-Link
Serial No:
Debugger supply voltage (V): 0
Baud rate: 9600
DLM state to change to: SSD - Secure Software Development

Memory partition sizes

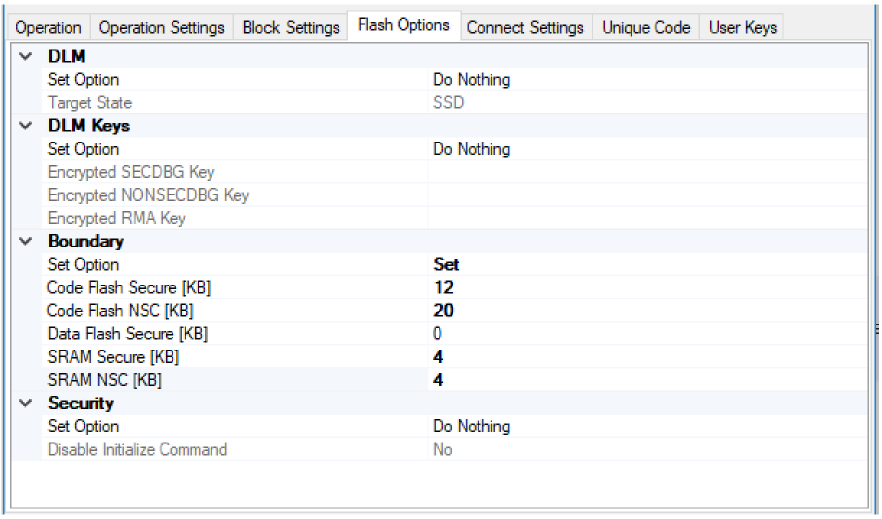
Code Flash Secure (KB)	12
Code Flash NSC (KB)	20
Data Flash Secure (KB)	0
SRAM Secure (KB)	4
SRAM NSC (KB)	4

□ Command line tool:

Figure 4. Set up IDAU Regions using Renesas Device Partition Manager

Manually Set up IDAU Region using Renesas Flash Programmer

Note that the settings in Figure 5 are based on the example project provided in this application project. Users need to adjust the settings based on their specific application.



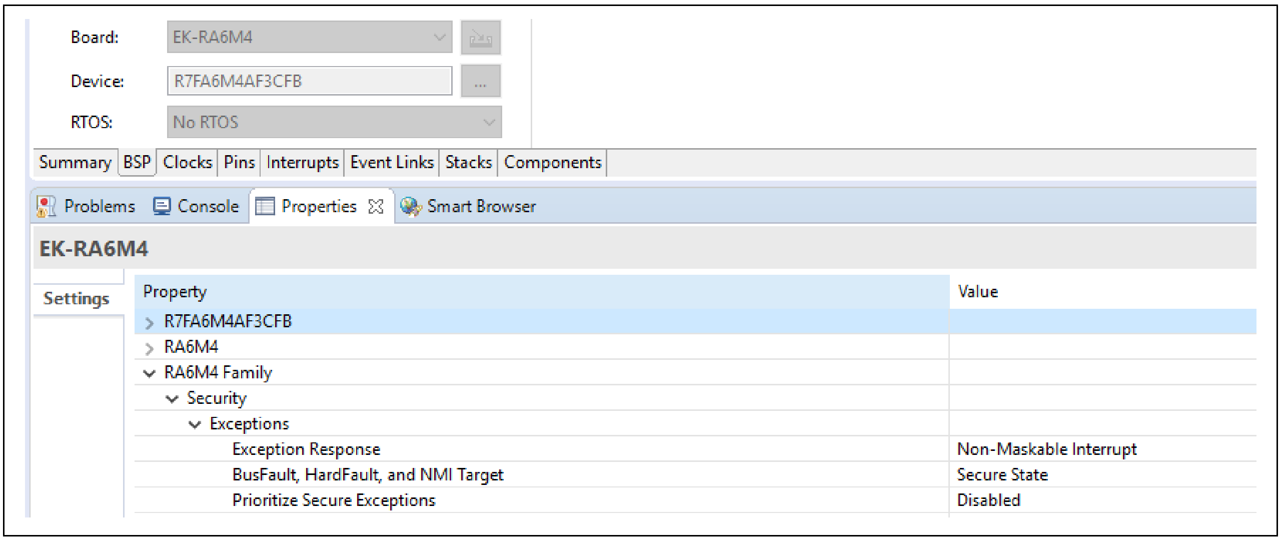
DLM	
Set Option	Do Nothing
Target State	SSD
DLM Keys	
Set Option	Do Nothing
Encrypted SECDBG Key	
Encrypted NONSECDBG Key	
Encrypted RMA Key	
Boundary	
Set Option	**Set**
Code Flash Secure [KB]	**12**
Code Flash NSC [KB]	**20**
Data Flash Secure [KB]	0
SRAM Secure [KB]	**4**
SRAM NSC [KB]	**4**
Security	
Set Option	Do Nothing
Disable Initialize Command	No

Figure 5. Set up IDAU Region using Renesas Flash Programmer

For an example RFP project which can set up the IDAU regions and perform Device Lifecycle state transition, user can refer to the application project, *Security Design with Arm® TrustZone® – IP Protection Appendix A*.

3.1.3 Configure the Secure Fault Handling using BSP Property

As shown in Figure 5, by default, NMI is selected as the exception response method and the fault will target Secure state. See section 4 for examples of handling of the Secure Fault.



Board: EK-RA6M4

Device: R7FA6M4AF3CFB

RTOS: No RTOS

Property	Value
R7FA6M4AF3CFB	
RA6M4	
RA6M4 Family	
Security	
Exceptions	
Exception Response	Non-Maskable Interrupt
BusFault, HardFault, and NMI Target	Secure State
Prioritize Secure Exceptions	Disabled

Figure 6. Secure Fault Exception Handling (Secure Project)

Note that although the security attribution configurations shown in Non-secure project Figure 6, the current tools support does not enable the Non-secure BSP project settings to be configured. Only Secure project settings are used in the Secure BSP configuration.

3.2 Configuring the Flash Block Protection

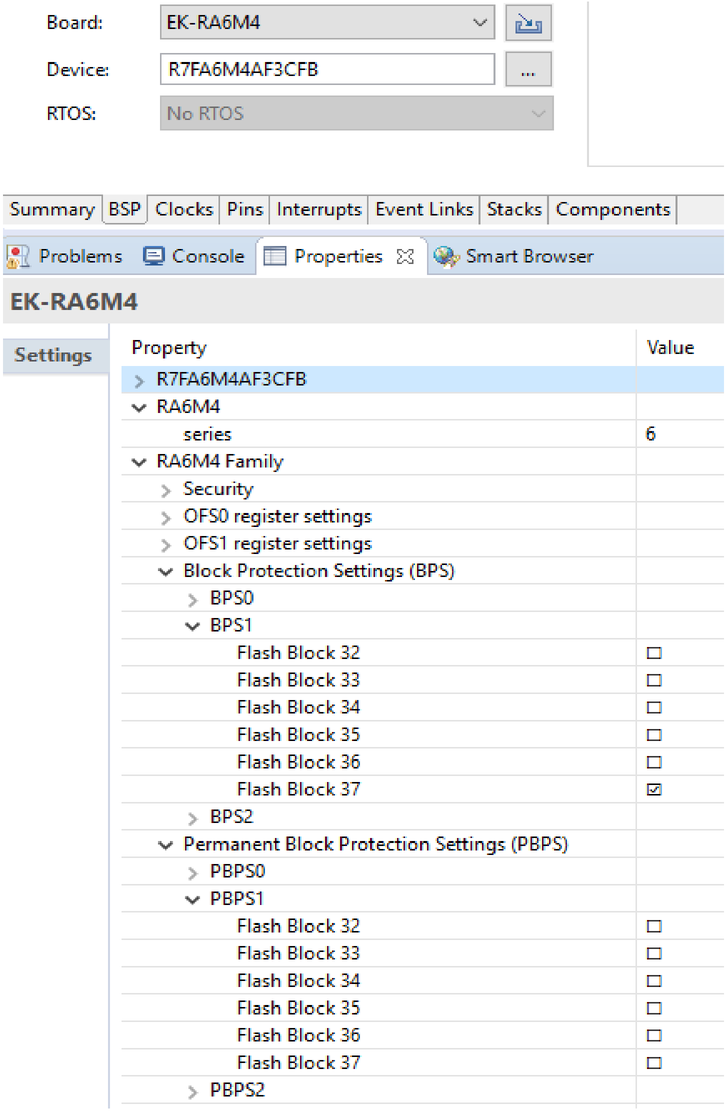
The Flash Block Protection feature protects secure or non-secure flash region from being erased or reprogrammed by secure or non-secure software. It is worth noting that the protection is for both Secure and Non-secure software accesses.

3.2.1 Setting up the Flash Block Protection

Both the Secure and Non-Secure projects can set up the flash block for protection from erase and programming as shown in Figure 7.

- BPS settings are for temporary protection.
- PBPS settings are for permanent protection.
- BPS0 and PBPS0 are for the protection of regions from 0 to 36 in flash linear mode.
- BPS1 and PBPS1 are for the protection of regions from 37 to in flash linear mode.
- PBS2 and PBPS2 are for the protection of regions when the flash is in dual bank mode. At the time of the creation of this application project, dual bank support is not available. These settings are not exercised in this application project.

Note: Once the flash block protection is set up, if a new program is downloaded to the locked region, user needs to follow section 3.2.2 to erase the flash prior to programming the new binary to the MCU.



Board: EK-RA6M4

Device: R7FA6M4AF3CFB

RTOS: No RTOS

Property	Value
R7FA6M4AF3CFB	
RA6M4	
series	6
RA6M4 Family	
Security	
OFS0 register settings	
OFS1 register settings	
Block Protection Settings (BPS)	
BPS0	
BPS1	
Flash Block 32	□
Flash Block 33	□
Flash Block 34	□
Flash Block 35	□
Flash Block 36	□
Flash Block 37	☑
BPS2	
Permanent Block Protection Settings (PBPS)	
PBPS0	
PBPS1	
Flash Block 32	□
Flash Block 33	□
Flash Block 34	□
Flash Block 35	□
Flash Block 36	□
Flash Block 37	□
PBPS2	

Figure 7. Set up Flash Block Protection

3.2.2 Clearing the Flash Block Protection

Clearing the temporary Flash Block Locking can be achieved by the Renesas Flash Programmer (RFP) or Renesas Device Partition Manager.

Note: If there is a permanently locked block or register, these two methods explained below will not work. The locked region will be permanently locked out of erasure and programming.

Utilizing Renesas Flash Programmer

The **Initialize Device** command erases all flash blocks that are not permanently locked and initializes the MCU to SSD state. After this command is successfully executed, the device lifecycle is updated to SSD and the contents on the flash memory are erased.

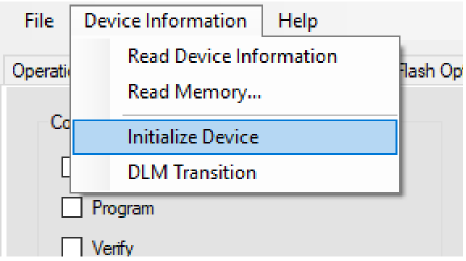
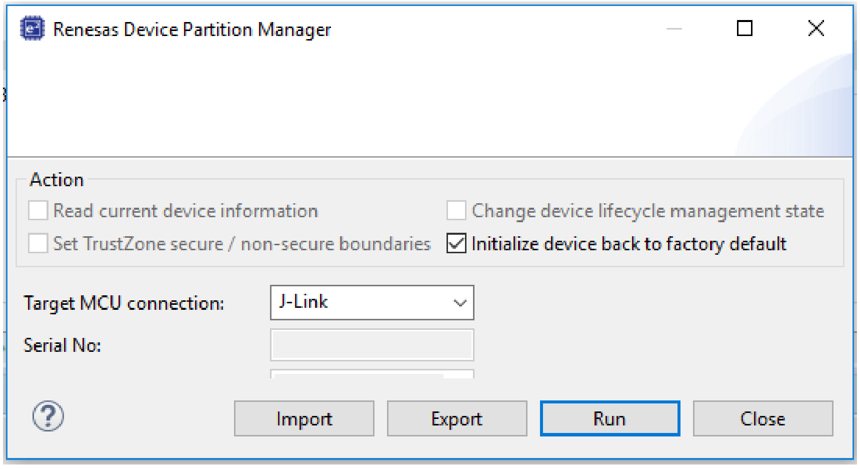


Figure 8. Clear the Flash Block Protection using RFP

Utilizing Renesas Device Partition Manager

User can use the following setting to perform the Initialization.



Renesas Device Partition Manager

Action

☐ Read current device information ☐ Change device lifecycle management state

☐ Set TrustZone secure / non-secure boundaries ☑ Initialize device back to factory default

Target MCU connection: J-Link

Serial No:

Figure 9. Clear the Flash Block Protection using Renesas Device Partition Manager

3.3 Configuring the Security Control for Debugging and Serial Programming

For RA Arm® TrustZone® enabled MCUs, setting up the Security Control for Debugging and Serial Programming is primarily concerned with setting the Device Lifecycle State.

Security considerations regarding the Debugging and Serial Programming span the entire Device Lifecycle of the RA TrustZone® enabled MCUs such as through:

- Development
- Production
- Deployment
- Failure Analysis

As shown in **Figure 3**, there are three types of Device Lifecycle Transitions:

- All Erase: typically used during Development.
  - The All Erase operation is the same operation described in section 3.2.2, which is used to clear the Flash Block Locking. See section 3.2.2 for the operation flow of All Erase.
- Unauthenticated Transitions: typically used during Development, Production and Deployment.
- Authenticated Transitions: typically used for MCU Device Lifecycle regression and Failure Analysis.

This section provides a brief explanation on setting up the Device Lifecycle state for Unauthenticated Transitions. User can refer to the application note, Renesas RA Family Installing and Utilizing the Device Lifecycle Management Keys for information on the use cases and process of performing Authenticated Transitions.

Operation details for setting up the Device Lifecycle using RFP and Renesas Device Partition Manager are common for the following three design processes using RA6M4:

- Combined Project Development
- Split Project Development
- Flat Project Development

For details on which steps will be used at which stage of the development process, refer to the application project, Renesas RA Family Security Design using TrustZone® – IP Protection.

3.3.1 Using Renesas Flash Programmer (RFP)

As shown below, user selects **Device information** > **DLM Transition** to reach the window on the right, to select the DLM State to transition into.

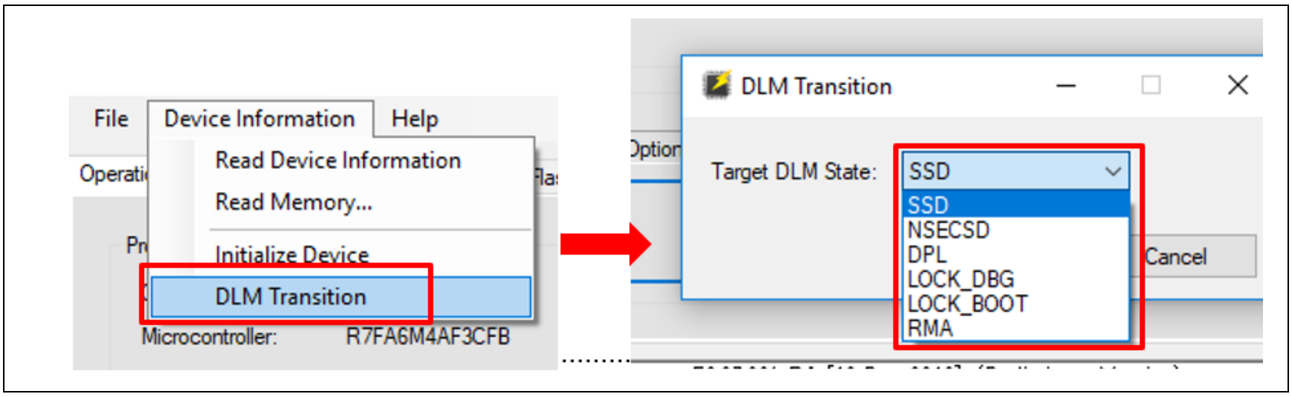
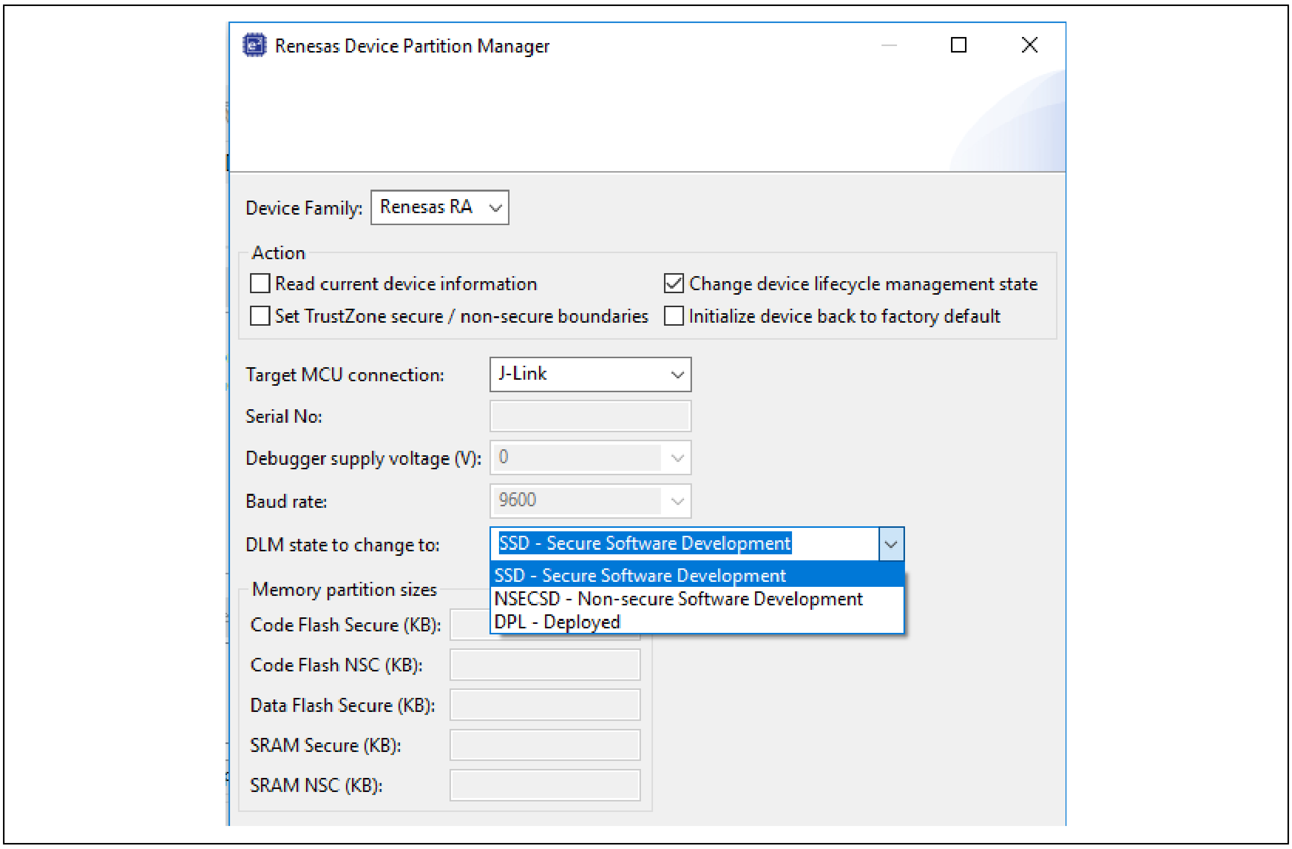


Figure 10. Unauthenticated Transition using RFP

Note that the States can only be transitioned following the paths shown in Figure 3. Following is a summary of possible unauthenticated transitions that can be made with the RFP interface shown in Figure 10. Information on when the transitions are typically performed is also provided. User can reference Table 3 for the corresponding debug interface changes caused by these transitions.

- CM > SSD
  - Typically used during initial development of the Secure Project.
- SSD > NSECSD
  - Typically used after the secure project is fully developed in the Split Project Development Model.
  - Typically used during production stage following the development phase of the Complete Project Development Model and the Flat Project Development Model.
- NSECSD > DPL
  - Typically used after the non-secure project is fully developed in the Complete Project Development Model.
  - Typically used during production stage following the development phase of the Complete Project Development Model and the Flat Project Development Model.
- DPL > LCK_DBG
  - Typically used during mass production stage for all three development models.
  - Note that the Debug interface is permanently locked down for the lifetime of the device in LCK_DBG.
- LCK_DBG > LCK_BOOT
  - Typically used during mass production stage for all three development models.
  - Note that the Serial Programming interface is permanently locked down for the lifetime of the device in LCK_BOOT.
- SSD > LCK_DBG
  - Typically used in mass production for matured products.

3.3.2 Using "Renesas Device Partition Manager"



Renesas Device Partition Manager

Device Family: Renesas RA

Action

☐ Read current device information
☑ Change device lifecycle management state
☐ Set TrustZone secure / non-secure boundaries
☐ Initialize device back to factory default

Target MCU connection: J-Link

Serial No:

Debugger supply voltage (V): 0

Baud rate: 9600

DLM state to change to: SSD - Secure Software Development

SSD - Secure Software Development
NSECSD - Non-secure Software Development
DPL - Deployed

Memory partition sizes

Code Flash Secure (KB):

Code Flash NSC (KB):

Data Flash Secure (KB):

SRAM Secure (KB):

SRAM NSC (KB):

Figure 11. Unauthenticated Transition using Renesas Device Partition Manager

As shown in Figure 11, following are the possible unauthenticated transitions that can be made with the **Renesas Device Partition Manager**. See section 3.3.1 for the use cases of these transitions.

- CM > SSD
- SSD > NSECSD
- NSECSD > DPL

Note: User needs to recycle the board prior to working with Renesas Device Partition Manager after a debug session if using J-Link as a connection interface.

4. Securing Data at Rest Demonstrations

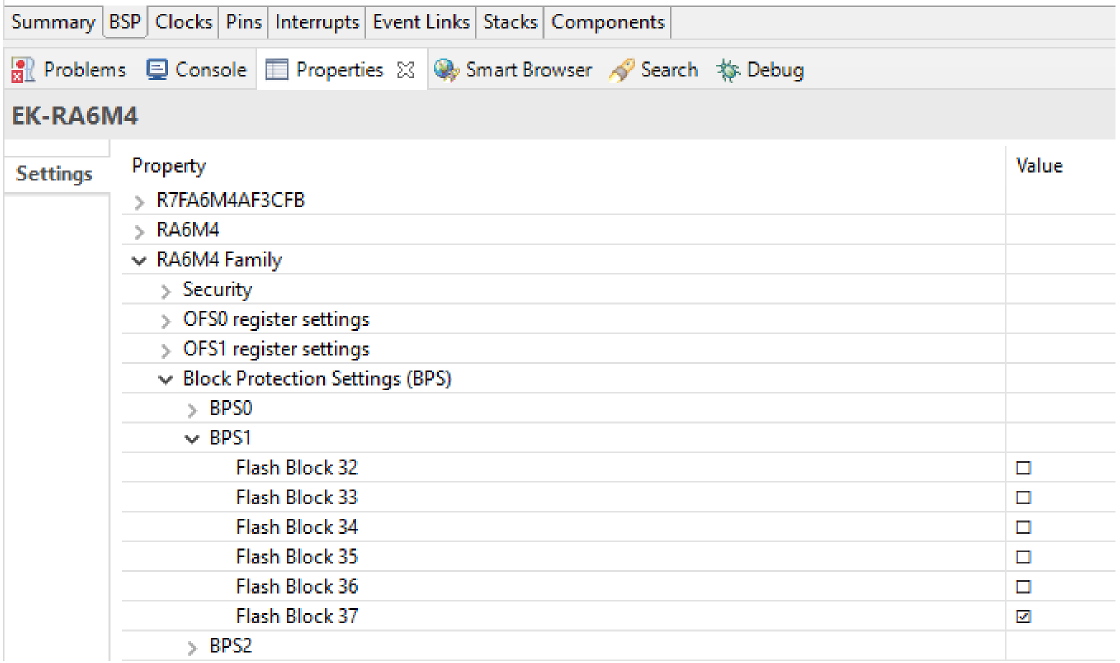
This section introduces a software project developed with EK-RA6M4, which demonstrates **Securing Data at Rest using TrustZone®**.

4.1 Software Architecture Overview

This project includes the following software components that demonstrate the flash and SRAM read and write protection.

Secure Project

- Secure FSP flash driver (Configured as Non-secure Callable).
- Secure flash code which implements the fault handler to catch Secure Fault and perform system reset.
- Secure project locks Flash block 37 from Programming and Erasing.



Summary | BSP | Clocks | Pins | Interrupts | Event Links | Stacks | Components

Problems | Console | Properties | Smart Browser | Search | Debug

EK-RA6M4

Settings

Property	Value
› R7FA6M4AF3CFB	
› RA6M4	
⌄ RA6M4 Family	
› Security	
› OFS0 register settings	
› OFS1 register settings	
⌄ Block Protection Settings (BPS)	
› BPS0	
⌄ BPS1	
Flash Block 32	☐
Flash Block 33	☐
Flash Block 34	☐
Flash Block 35	☐
Flash Block 36	☐
Flash Block 37	☑
› BPS2	

Figure 12. Flash Block 37 is Temporarily Locked

Non-secure Project

- J-Link RTT Viewer User Interface code.
- Non-secure flash code which accesses secure flash and secure SRAM region.
- Non-secure SRAM code which accesses secure SRAM region.

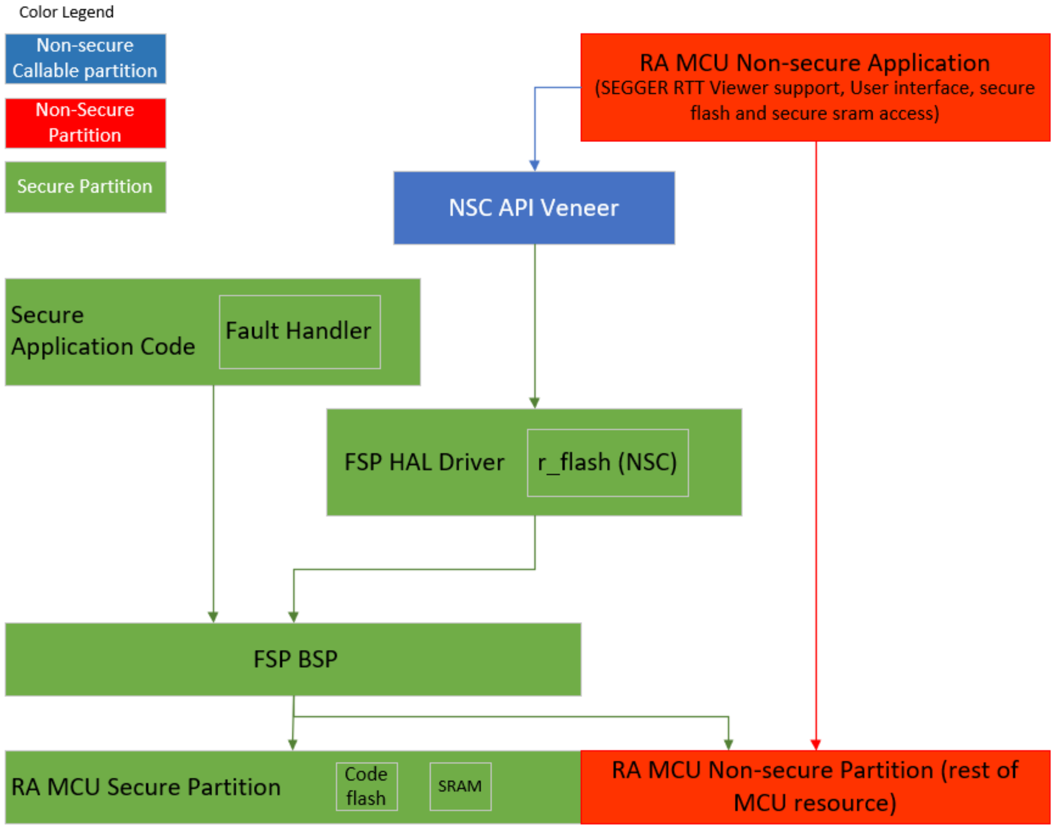

Color Legend
Non-secure Callable partition
Non-Secure Partition
Secure Partition
RA MCU Non-secure Application
(SEGGER RTT Viewer support, User interface, secure flash and secure sram access)
NSC API Veneer
Secure Application Code
Fault Handler
FSP HAL Driver
r_flash (NSC)
FSP BSP
RA MCU Secure Partition
Code flash
SRAM
RA MCU Non-secure Partition (rest of MCU resource)

Figure 13. Software Block Diagram

4.2 Functionality Description

The J-Link RTT user code runs in the non-secure flash region. It monitors user input from the J-Link RTT Viewer and activates non-secure flash and non-secure SRAM software components to take the following actions:

- Illegal read to secure flash, secure SRAM from Non-secure partition
  - System will reset upon catching these Secure Faults.
- Illegal writes to secure SRAM from Non-secure partition
  - System will reset upon illegal write to secure SRAM.
  - System will print error message upon illegal write to secure flash.
- Writes to secure flash region which are not locked from programming and erasing from Non-secure callable flash driver
  - Guard function for the Non-secure callable flash driver will catch illegal copy from secure flash region to write to Non-secure flash region.
  - Guard function for the Non-secure callable flash driver will catch illegal write to secure flash region. This use case is provided as an example of user enforced security.
- Writes to Non-secure code flash region which is locked from programming and erasing from Non-secure callable flash driver
  - Programming and erasing are not granted.
  - System will print error message returned from FSP.
- Write to Non-secure region which are not locked from programming and erasing from Non-secure callable flash driver
  - This operation will succeed after all security check from the guard function.

The software project uses the default Security configuration as shown in Figure 4. The secure fault is handled in MCU Secure state. The Secure Fault is handled by the Secure project function `HardFault_Handler()` implemented in `fault_handling_system_reset.c`.

4.3 Running the Demonstration Project

User can use the following steps to run the application.

4.3.1 Import and Build the Secure and Non-secure Projects

Follow *FSP User's Manual* section**,** Importing an Existing Project into e^2 studio to import the Secure Project and Non-secure into the workspace and compile in the order instructed below.

1. First step, double click the `configuration.xml` from the Secure project. Click **Generate Project Content** and then build the Secure project.
2. Next step, double click the `configuration.xml` from the Non-secure project. Click **Generate Project Content** and then build the Non-secure project.

4.3.2 Setting up the Hardware

Connect J10 on EK-RA6M4 using the micro-USB cable to the workstation to provide power and debugging capability using the on-board debugger.

Initialize the MCU

This step is optional but recommended. Prior to download the example application, it is recommended user Initializes the device to SSD state. Unlocked flash content will be erased during this process. This is particularly helpful if the device was previously used in NSECSD state or has certain flash block locked up temporarily.

Use Renesas Device Partition Manager and J-Link Debugger to Initialize the MCU

Establish following connection prior to use "Renesas Device Partition Manager" and the Onboard J-Link debugger to perform "initialize device back to factory default" action. Note that "initialize device back to factory default" performs the same functionality as "Initialize Device" when using RFP.

- EK-RA6M4 jumper setting: J6 closed, J9 open. Other jumpers keep out-of-box setting.
- USB cable connected between J10 and development PC.

Note: User needs to power cycle the board prior to work with **Renesas Device Partition Manager** after a debug session if using J-Link as connection interface.

Open the **Renesas Device Partition Manager**.

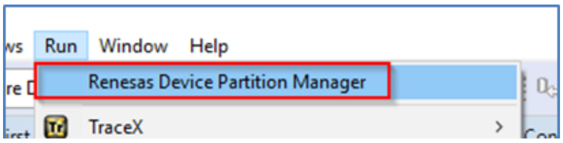
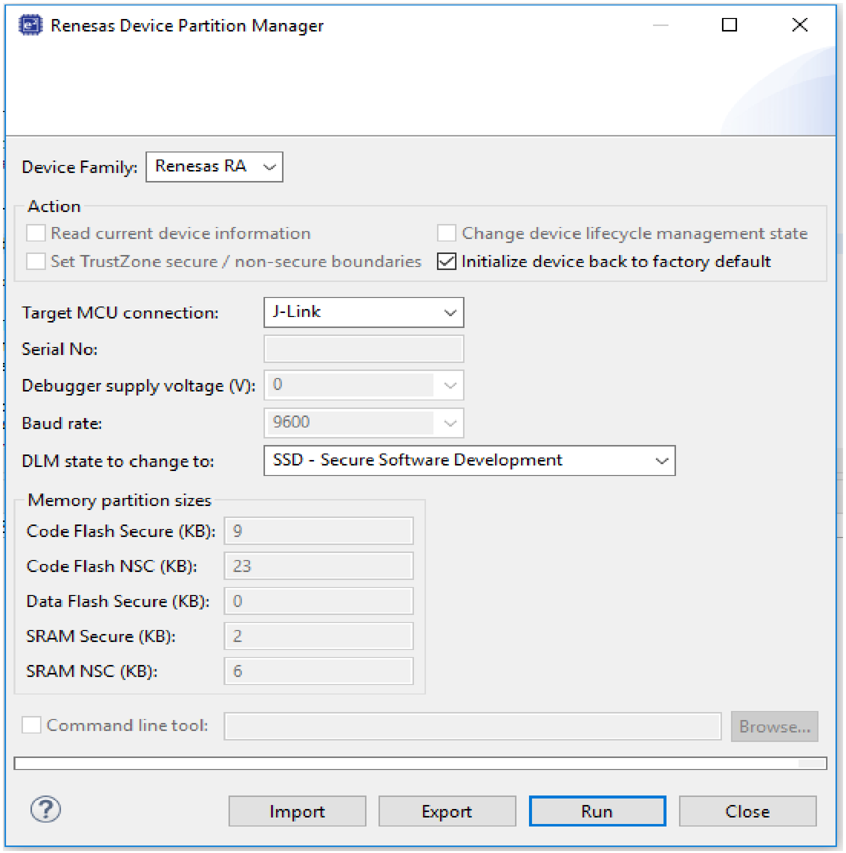


Figure 14. Open Renesas Device Partition Manager

Next, check **initialize device back to factory default**, choose the connection method and then click **Run**.


Renesas Device Partition Manager

Device Family: Renesas RA

Action

□ Read current device information
□ Change device lifecycle management state
□ Set TrustZone secure / non-secure boundaries
☑ Initialize device back to factory default

Target MCU connection: J-Link

Serial No:

Debugger supply voltage (V): 0

Baud rate: 9600

DLM state to change to: SSD - Secure Software Development

Memory partition sizes

Code Flash Secure (KB): 9

Code Flash NSC (KB): 23

Data Flash Secure (KB): 0

SRAM Secure (KB): 2

SRAM NSC (KB): 6

□ Command line tool:

Figure 15. Initialize RA6M4 using Renesas Device Partition Manager

4.3.3 Verifying the Functionalities

To run the application, right click on `securing_data_at_rest_ek_ra6m4_ns` and select **Debug As** > **Renesas GDB Hardware Debugging**.

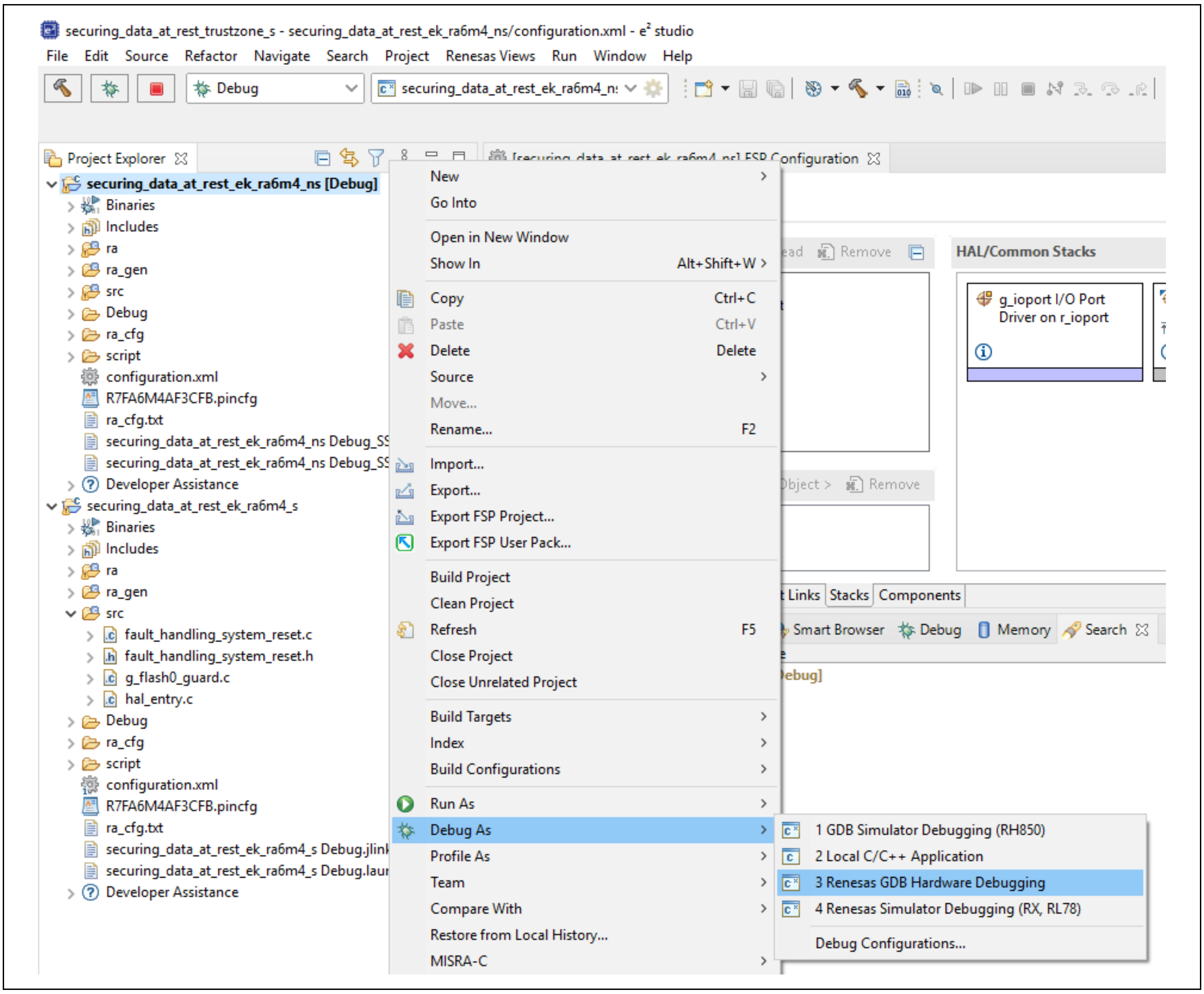
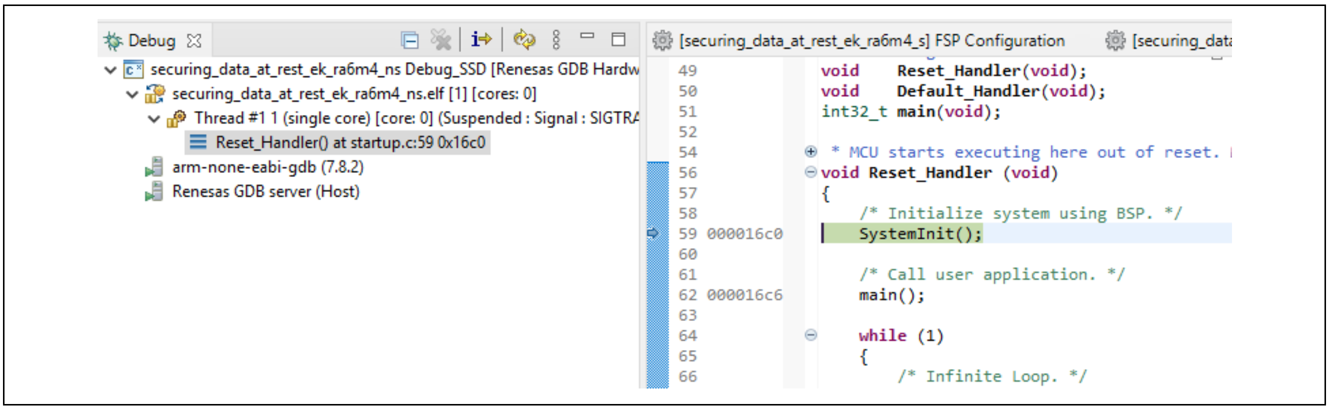

securing_data_at_rest_trustzone_s - securing_data_at_rest_ek_ra6m4_ns/configuration.xml - e² studio
File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
Debug
securing_data_at_rest_ek_ra6m4_n:
Project Explorer
securing_data_at_rest_ek_ra6m4_ns [Debug]
Binaries
Includes
ra
ra_gen
src
Debug
ra_cfg
script
configuration.xml
R7FA6M4AF3CFB.pincfg
ra_cfg.txt
securing_data_at_rest_ek_ra6m4_ns Debug_SS
securing_data_at_rest_ek_ra6m4_ns Debug_SS
Developer Assistance
securing_data_at_rest_ek_ra6m4_s
Binaries
Includes
ra
ra_gen
src
fault_handling_system_reset.c
fault_handling_system_reset.h
g_flash0_guard.c
hal_entry.c
Debug
ra_cfg
script
configuration.xml
R7FA6M4AF3CFB.pincfg
ra_cfg.txt
securing_data_at_rest_ek_ra6m4_s Debug.jlink
securing_data_at_rest_ek_ra6m4_s Debug.laun
Developer Assistance
New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Export FSP Project...
Export FSP User Pack...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Project
Build Targets
Index
Build Configurations
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
MISRA-C
1 GDB Simulator Debugging (RH850)
2 Local C/C++ Application
3 Renesas GDB Hardware Debugging
4 Renesas Simulator Debugging (RX, RL78)
Debug Configurations...
[securing_data_at_rest_ek_ra6m4_ns] FSP Configuration
Remove
HAL/Common Stacks
g_ioport I/O Port
Driver on r_ioport
Remove
Links Stacks Components
Smart Browser Debug Memory Search
[Debug]

Figure 16. Running the Application

Note that prior to the application execution, the IDAU regions will be set up to assume the values shown in Figure 4 through the debugger interaction with the MCU bootloader.

The projects are now loaded and the debugger should be paused in the `Reset_Handler()` at the `SystemInit()` call in the Secure project.



securing_data_at_rest_ek_ra6m4_ns Debug_SSD [Renesas GDB Hardw
securing_data_at_rest_ek_ra6m4_ns.elf [1] [cores: 0]
Thread #1 1 (single core) [core: 0] (Suspended : Signal : SIGTRA
Reset_Handler() at startup.c:59 0x16c0
arm-none-eabi-gdb (7.8.2)
Renesas GDB server (Host)

```
         void    Reset_Handler(void);
         void    Default_Handler(void);
         int32_t main(void);

          * MCU starts executing here out of reset.
         void Reset_Handler (void)
         {
             /* Initialize system using BSP. */
000016c0     SystemInit();

             /* Call user application. */
000016c6     main();

             while (1)
             {
                 /* Infinite Loop. */
```

Figure 17. Secure Project Reset Handler

Click ▶ twice to run the project.

Next, launch **J-Link RTT Viewer** V7.94g or later.

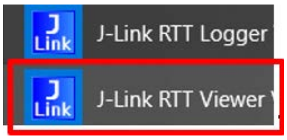
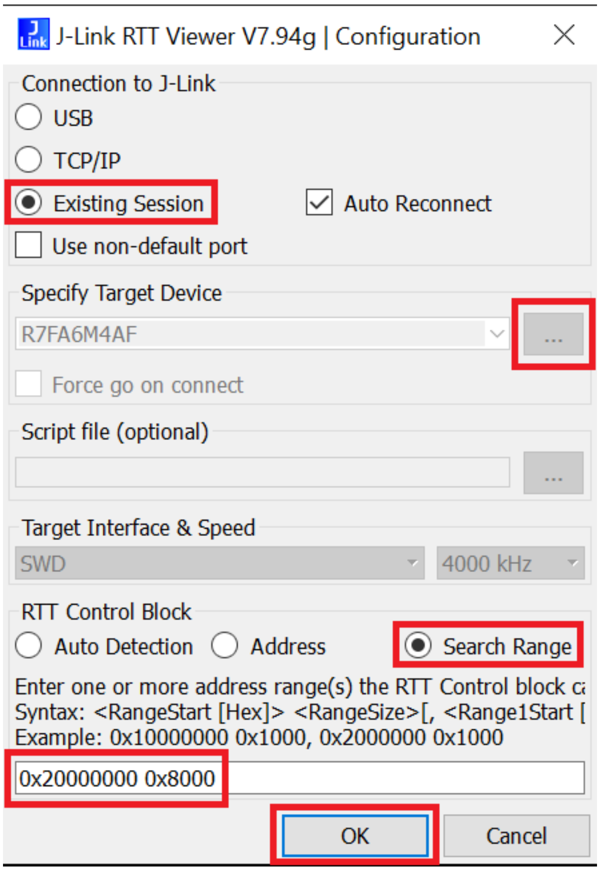


Figure 18. Launch J-Link RTT Viewer

Select **Existing Session** as connection type. Set up the RTT Control Block to **Search Range**. Set the search range to 0x20000000 0x8000 and then **OK** to start RTT.

Note: This Search Size 0x8000 is based on this example application project, if your application uses RTT viewer in Non-secure region and there is a large secure binary, user needs to increase the Search Size to cover the Non-secure project SRAM regions.

If the host PC has more than one J-Link debugger connected to the PC, set the **Serial No** (by default Serial No is set to 0).



J-Link RTT Viewer V7.94g | Configuration

Connection to J-Link

○ USB

○ TCP/IP

● Existing Session ☑ Auto Reconnect

□ Use non-default port

Specify Target Device

R7FA6M4AF ...

□ Force go on connect

Script file (optional)

...

Target Interface & Speed

SWD 4000 kHz

RTT Control Block

○ Auto Detection ○ Address ● Search Range

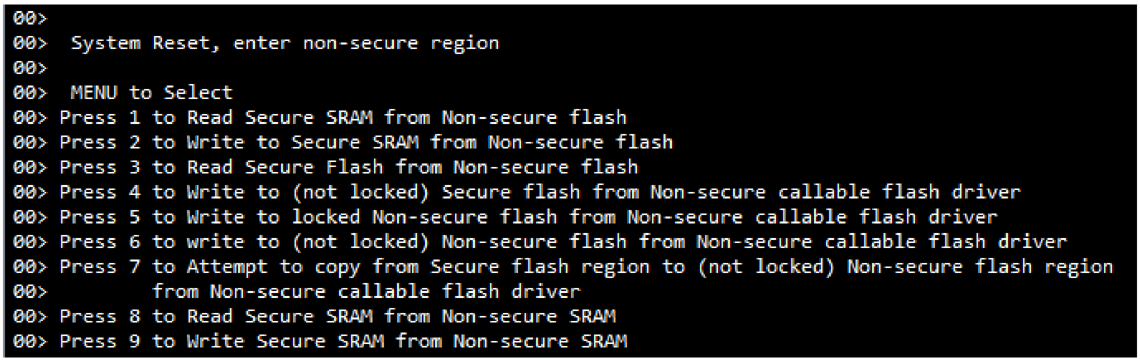
Enter one or more address range(s) the RTT Control block ca
Syntax: <RangeStart [Hex]> <RangeSize>[, <Range1Start [
Example: 0x10000000 0x1000, 0x2000000 0x1000

0x20000000 0x8000

Figure 19. Launch SEGGER RTT Viewer

Following steps provide a walkthrough of the functionalities provided by the system:

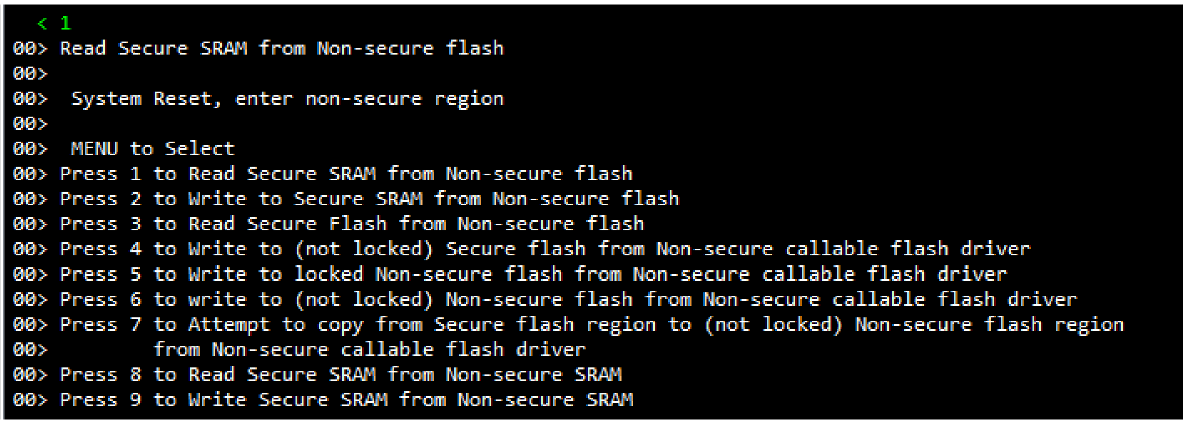
Step 1: The main menu items are printed on the RTT Viewer Terminal 0.



```
00>
00>  System Reset, enter non-secure region
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>        from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 20. Main Menu

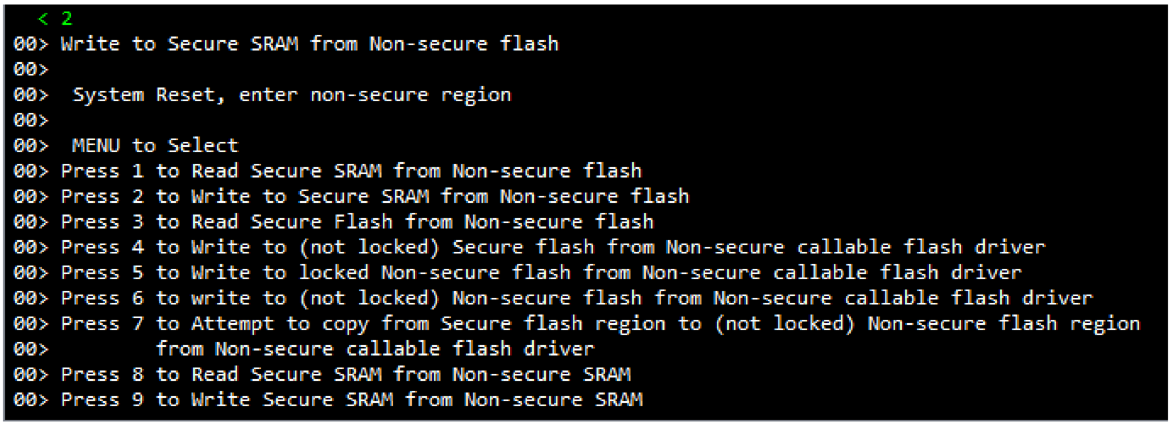
Input **1** to read the secure SRAM from the non-secure flash and confirm that Arm® TrustZone® secure fault is generated and system is reset.



```
  < 1
00> Read Secure SRAM from Non-secure flash
00>
00>  System Reset, enter non-secure region
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>        from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 21. Non-secure Flash Program Reads Secure SRAM Region

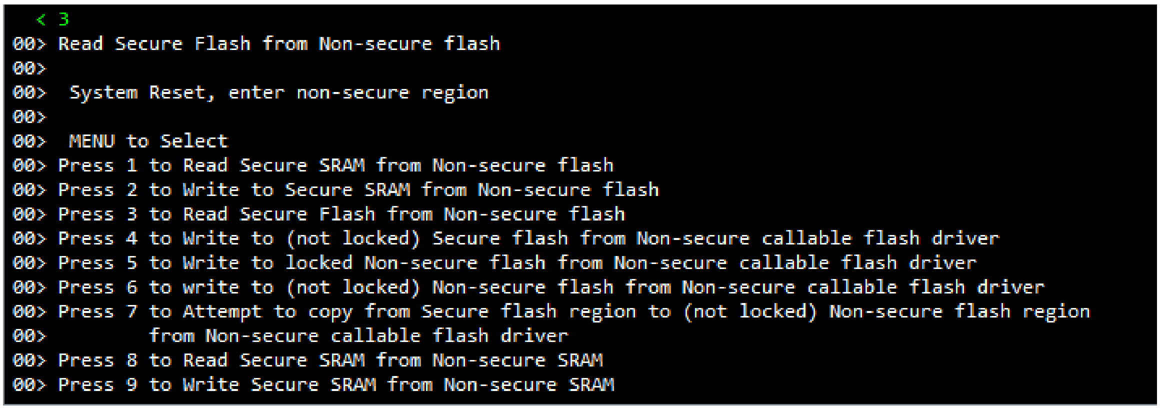
Input **2** to write the secure SRAM from the non-secure flash and confirm that TrustZone® secure fault is generated and system is reset.



```
  < 2
00> Write to Secure SRAM from Non-secure flash
00>
00>  System Reset, enter non-secure region
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>        from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 22. Non-secure Program Reads Secure SRAM Region

Input **3** to read the secure flash from the non-secure flash and confirm that TrustZone® secure fault is generated and system is reset.

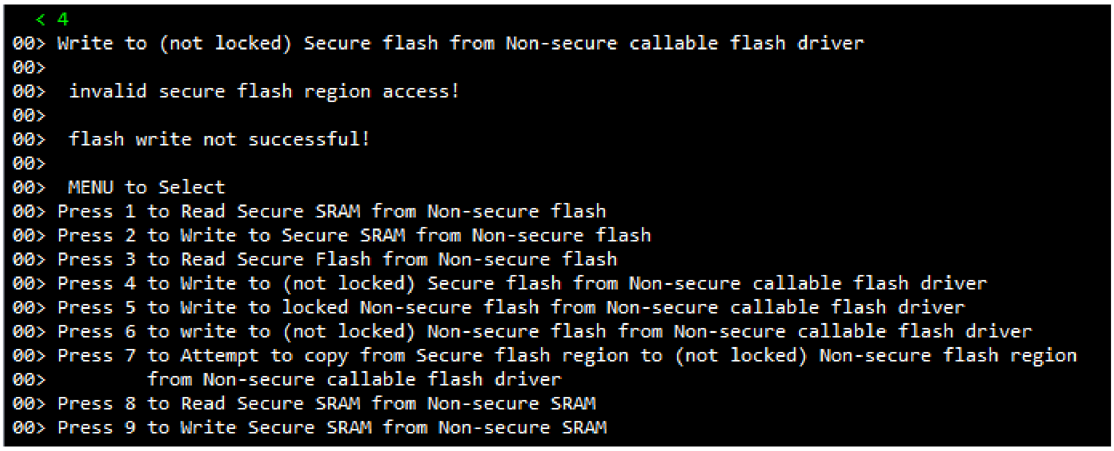


```
  < 3
00> Read Secure Flash from Non-secure flash
00>
00>  System Reset, enter non-secure region
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>         from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 23. Non-secure Program Read Secure Flash Region

Input **4** to write to the not locked Secure flash region. Function `g_flash0_write_guard` implemented security check to disable writing to secure region from non-secure callable. Therefore, this operation failed.

Note: No Arm® TrustZone® secure fault is generated in this case.

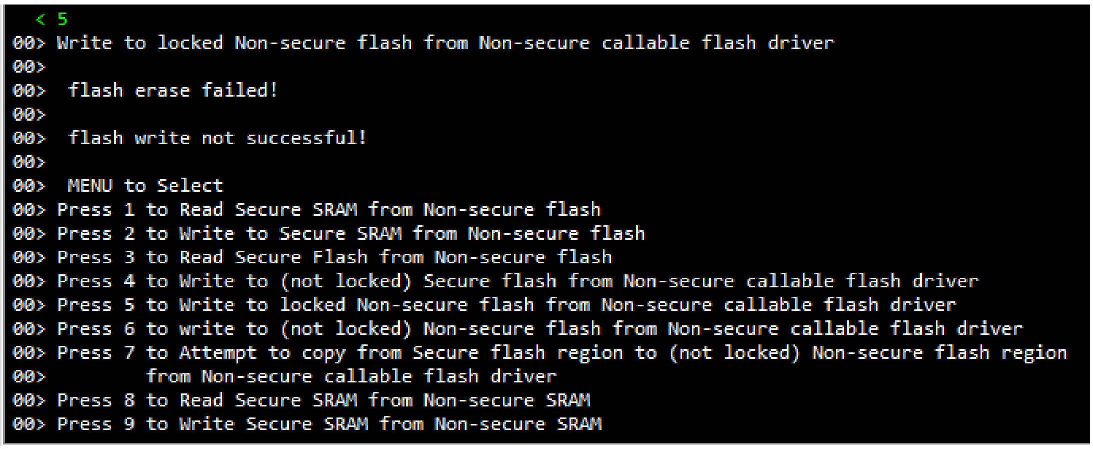


```
  < 4
00> Write to (not locked) Secure flash from Non-secure callable flash driver
00>
00>  invalid secure flash region access!
00>
00>  flash write not successful!
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>         from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 24. Non-secure Flash Program Writes to Secure Flash Region

Input **5** to write to the Non-secure flash region locked by secure code. Flash erase will fail because this flash block is locked for programming and erase.

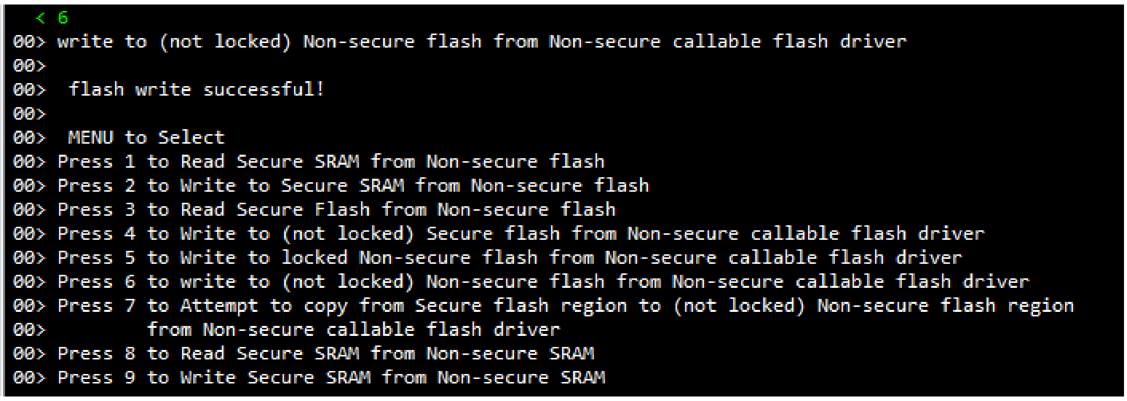
Note: No TrustZone® secure fault is generated in this case.



```
  < 5
00> Write to locked Non-secure flash from Non-secure callable flash driver
00>
00>  flash erase failed!
00>
00>  flash write not successful!
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>         from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 25. Non-secure Flash Program Writes to Locked Non-secure Flash Region

Input **6** to write to the Non-secure flash region, not locked, and confirm that write is successful and the main menu is reprinted.

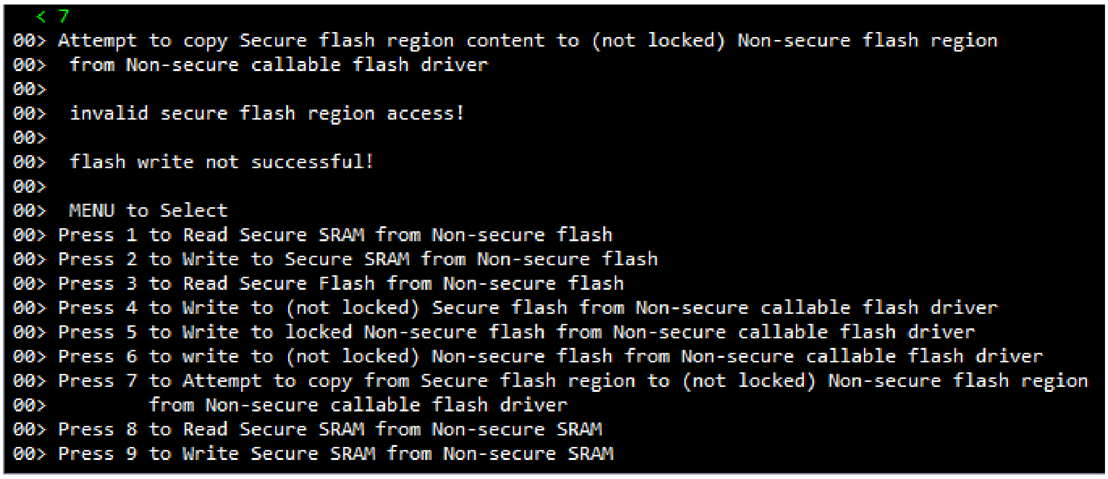


```
  < 6
00> write to (not locked) Non-secure flash from Non-secure callable flash driver
00>
00>  flash write successful!
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>         from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 26. Non-secure Flash Program Writes to Not Locked Non-secure Flash Region

Input **7** to take secure flash region as the source buffer to write to not locked non-secure flash region. Function `g_flash0_write_guard` implemented security check to disable copying data from secure flash region and trying to write to non-secure flash region.

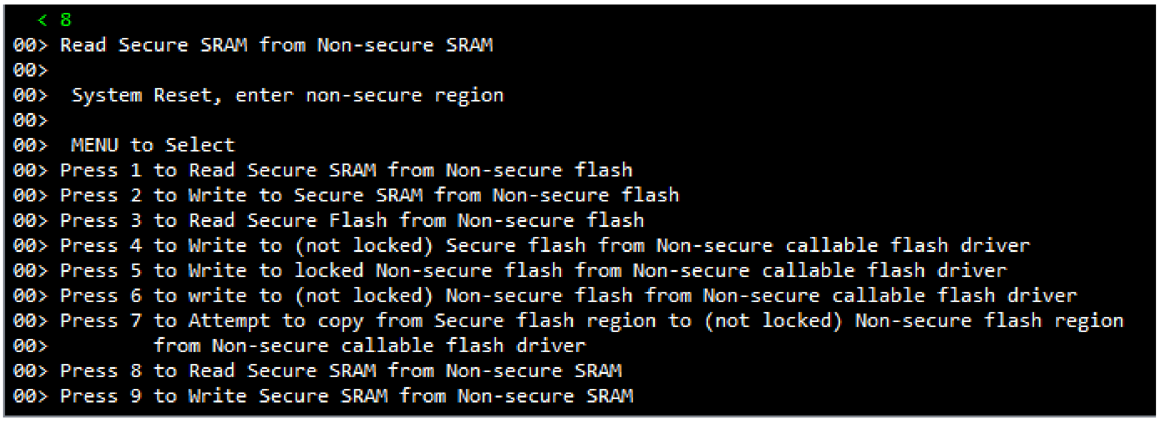
Note: No Arm® TrustZone® secure fault is generated in this case.



```
  < 7
00> Attempt to copy Secure flash region content to (not locked) Non-secure flash region
00>  from Non-secure callable flash driver
00>
00>  invalid secure flash region access!
00>
00>  flash write not successful!
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>         from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 27. Non-secure SRAM Program Reads from Secure SRAM Region

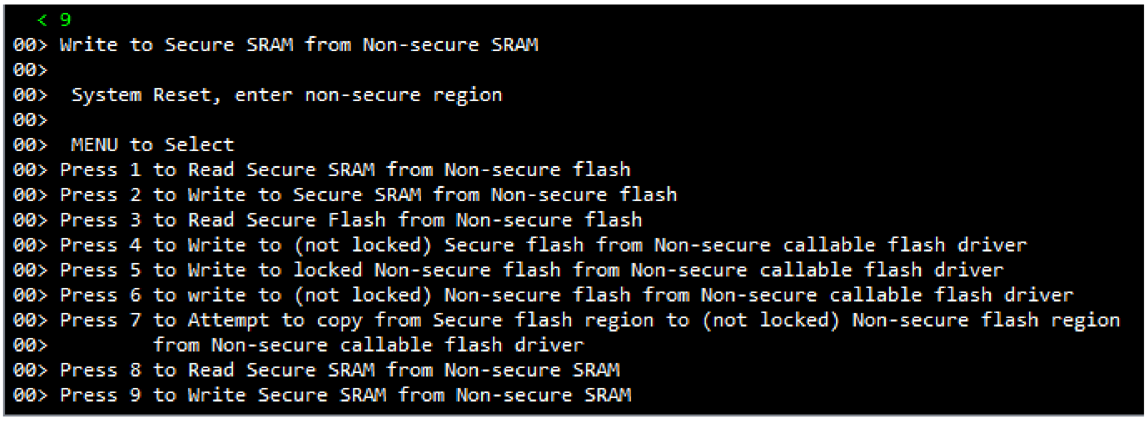
Input **8** to read secure SRAM region from Non-secure SRAM region and confirm TrustZone® secure fault is generated and system is reset.



```
  < 8
00> Read Secure SRAM from Non-secure SRAM
00>
00>  System Reset, enter non-secure region
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>         from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 28. Non-secure SRAM Program Reads from Secure SRAM Region

Input **9** to write to the secure SRAM region from Non-secure SRAM region and confirm that Arm® TrustZone® secure fault is generated and system is reset.



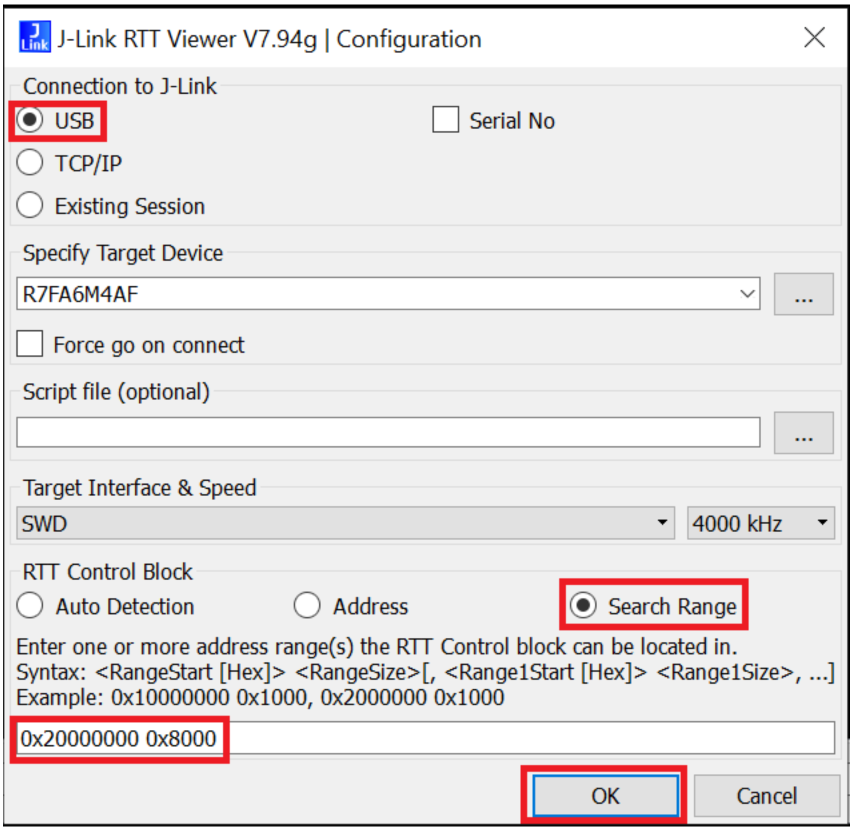
```
  < 9
00> Write to Secure SRAM from Non-secure SRAM
00>
00>  System Reset, enter non-secure region
00>
00>  MENU to Select
00> Press 1 to Read Secure SRAM from Non-secure flash
00> Press 2 to Write to Secure SRAM from Non-secure flash
00> Press 3 to Read Secure Flash from Non-secure flash
00> Press 4 to Write to (not locked) Secure flash from Non-secure callable flash driver
00> Press 5 to Write to locked Non-secure flash from Non-secure callable flash driver
00> Press 6 to write to (not locked) Non-secure flash from Non-secure callable flash driver
00> Press 7 to Attempt to copy from Secure flash region to (not locked) Non-secure flash region
00>         from Non-secure callable flash driver
00> Press 8 to Read Secure SRAM from Non-secure SRAM
00> Press 9 to Write Secure SRAM from Non-secure SRAM
```

Figure 29. Non-secure SRAM Program Writes to Secure SRAM Region

Note: After running this application project, user must follow section 3.2.2 to clear the flash block that is locked.

Notes on Running the Application in Standalone mode

After the MCU is programmed with the application code, the user can run the application in standalone mode (with no debugging session). In this case, choose the "**USB**" as the "**connection to J-Link**":



J-Link RTT Viewer V7.94g | Configuration

Connection to J-Link

● USB □ Serial No

○ TCP/IP

○ Existing Session

Specify Target Device

R7FA6M4AF

□ Force go on connect

Script file (optional)

Target Interface & Speed

SWD 4000 kHz

RTT Control Block

○ Auto Detection ○ Address ● Search Range

Enter one or more address range(s) the RTT Control block can be located in.
Syntax: <RangeStart [Hex]> <RangeSize>[, <Range1Start [Hex]> <Range1Size>, ...]
Example: 0x10000000 0x1000, 0x2000000 0x1000

0x20000000 0x8000

Figure 30. SEGGER RTT Viewer Connection Setup when MCU Running in Standalone Mode

4.3.4 Migrating to Other TrustZone® Enabled RA MCUs

As each MCU group and each MCU within a group may have different internal flash and SRAM size, use the following steps to migrate this application:

1. Modify the BSP Flash Block Protection region settings to fit the memory size.
2. Modify the flash test block address to match the MCU.

5. References

1. Renesas RA6M4 Group User's Manual: Hardware
2. Flexible Software Package (FSP) User's Manual
3. Arm® TrustZone® Technology for the Armv8-M Architecture
4. Renesas RA Family Installing and Utilizing the Device Lifecycle Management Keys (R11AN0469)
5. Security Design with Arm® TrustZone® - IP Protection (R11AN0467)
6. Arm®v8-M Architecture Reference Manual
7. Arm® Cortex®-M33 Processor Technical Reference Manual
8. Arm® Cortex®-M33 Devices Generic User Guide

6. Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.01.20	—	First release document
1.10	Jul.10.23	—	Updated to FSP Version 4.4.0
1.20	May.01.24	—	Updated to FSPv5.2.0

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 November 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/.

© 2024 Renesas Electronics Corporation. All rights reserved.