Renesas RA Family

RA Arm® TrustZone® Tooling Primer

Introduction

This application note will introduce the user to the tools supporting Arm® TrustZone® configuration for the RA Family of microcontrollers. It is intended to be read by development engineers implementing RA TrustZone projects for the first time. It will introduce basic concepts followed by workflow and tooling functions designed to simplify and accelerate their first TrustZone development. A background knowledge of e^2 studio and RA device hardware is expected.

Target Device

RA Arm® Cortex®-M33 devices with TrustZone security extension.

Contents

1. Renesas Implementation of Arm® TrustZone® Technology

For brevity, TrustZone will be abbreviated to TZ in this document.

The following section is supplied for reference only. For full details of TZ implementation, please refer to Arm documentation (https://developer.arm.com/ip-products/security-ip/trustzone) or the RA6M4 device manual.

Arm TZ technology divides the MCU and therefore the application into Secure and Non-Secure partitions. Secure applications can access both Secure and Non-Secure memory and resources. Non-Secure code can access Non-Secure memory and resources as well as Secure resources through a set of so-called veneers located in the Non-Secure Callable (NSC) region. This ensures a single access point for Secure code when called from the Non-Secure partition. The MCU starts up in the Secure partition by default. The security state of the CPU can be either Secure or Non-Secure.

The MCU code flash, data flash, and SRAM are divided into Secure (S) and Non-Secure (NS) regions. Code flash and SRAM include a further region known as Non-Secure Callable (NSC). These memory security attributes are set into the non-volatile memory via SCI or USB boot mode commands when the device lifecycle is Secure Software Debug (SSD) state. The memory security attributes are loaded into the Implementation Defined Attribution Unit (IDAU) peripheral and the memory controller before application execution and cannot be updated by application code.

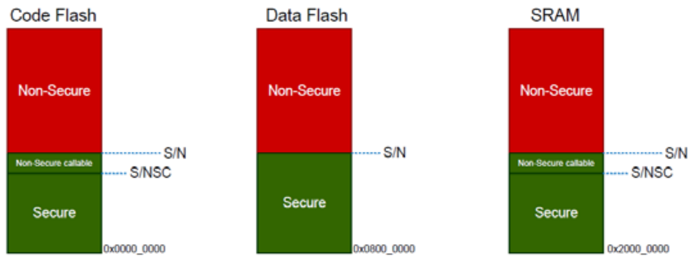


Figure 1. Secure and Non-Secure Regions

Note: All external memory accesses are considered to be Non-Secure.

Code Flash and SRAM can be divided into Secure, Non-Secure, and Non-Secure Callable. All secure memory accesses from the Non-Secure region MUST go through the Non-Secure Callable gateway and target a specific Secure Gateway (SG) assembler instruction. This forces access to Secure APIs at a fixed location and prevents calls to sub-functions and so on. Failing to target an SG instruction will generate a TZ exception.

TZ enabled compilers will manage generation of the NSC veneer automatically using CMSE extensions.

1.1 Calling from Non-Secure to Secure

A new instruction SG (Secure Gateway) has been added to the Armv8-M architecture. This MUST be the destination instruction for any branch within the Non-Secure Callable region. If an attempt is made to branch to any other instruction from the Non-Secure partition, a TZ exception will be thrown.

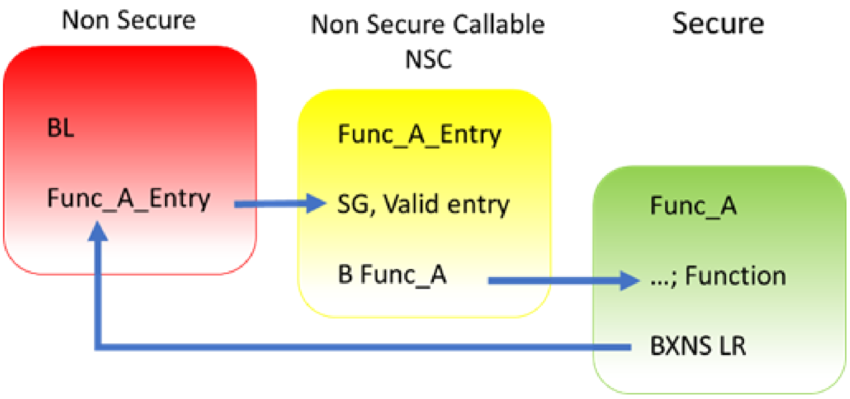


Figure 2. Calling from Non-Secure to Secure Functions

1.2 Calling from Secure to Non-Secure

Secure code uses B(L)XNS instructions to make direct calls to Non-Secure functions. While this is certainly possible, it can create a security vulnerability in the application. It is also challenging for the Secure application to determine the address of the non-secure function during build phase. From the RA Tools and FSP point view, calling directly from Secure to Non-Secure via FSP API is not supported.

Preference is for the Secure code to initialise as necessary from reset, pass control to the Non-Secure partition and manage any data transfers and so forth via FSP call-backs. Security checks, for example, copying Secure data to Non-secure RAM can be performed as necessary as part of the Secure call-back. See the FSP documentation for more details.

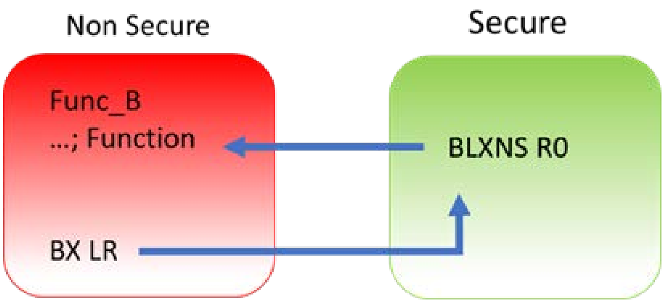


Figure 3. Calling from Secure to Non-Secure Functions

2. Workflow

Arm® TrustZone® MCU development normally consists of two projects within a workspace, Secure and Non-Secure. General project workflows are described in the following sections. The Renesas project generator also supports development with “Flat project” model with no TrustZone awareness.

2.1 Secure Project

1. Start a new Secure project in e^2 studio.
2. Select and configure pins and drivers/stacks that need to be initialized and used in Secure mode. This should be kept to a minimum to reduce the security attack surface.
3. Expose top of stacks as Non-Secure Callable (NSC) ***if*** they need to be accessed from Non-Secure partition. Again, this should be kept to a minimum.
4. Generate project content and write Secure code such as key handling and opening drives as needed.
5. Modify/remove any unnecessary “Guard” functions as needed to control access via NSC.
6. Build project.
7. A Non-Secure project will be needed before debugging. If necessary, prepare a “dummy” Non-Secure project or replace `R_BSP_NonSecureEnter();` with `while(1);` in `hal_entry.c.`

2.2 Non-Secure Project

1. Start a new Non-Secure project.
2. If you have access to the Secure project, choose this option. However, if you only have access to a device with pre-programmed Secure code (commonly referred to as provisioned device) choose “Secure Bundle”.
3. Select and configure pins and drivers/stacks that need to be initialized and used in Non-Secure mode.
4. Note that you can add NSC drivers and stacks as needed.
5. Generate project content and write Non-Secure code as needed
6. Access NSC drivers and Stacks via Guard functions.
7. Build and debug project.

2.3 Flat Project

A flat project does not technically use TrustZone as the developer has made a decision to place the entire application in Secure partition from restart.

Notes:

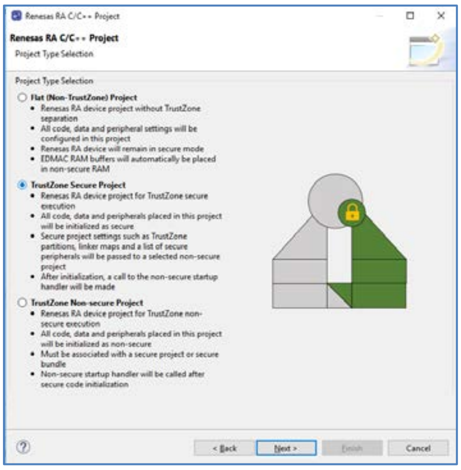
- Any code placed in external memory (such as OSPI or QSPI) will be Non-Secure.
- The Ethernet EDMAC is designed to be a Non-Secure bus master so associated Ethernet RAM buffers will be placed in Non-Secure RAM. The tooling will automatically manage this.

The workflow is as follows:

1. Start a new Flat project.
2. Select and configure pins and drivers/stacks as needed.
3. Generate project content and write code as needed.
4. Build and debug project.

3. RA Project Generator (PG)

The RA project generators have been created to help users through setting up new TZ enabled projects. User will be prompted for project settings such as Project Type (Secure, Non-Secure, or Flat), compiler, RTOS and debugger. Care is needed when setting up a TZ project to ensure that the connection between Secure and Non-Secure partitions are managed correctly.



Renesas RA C/C++ Project

Project Type Selection

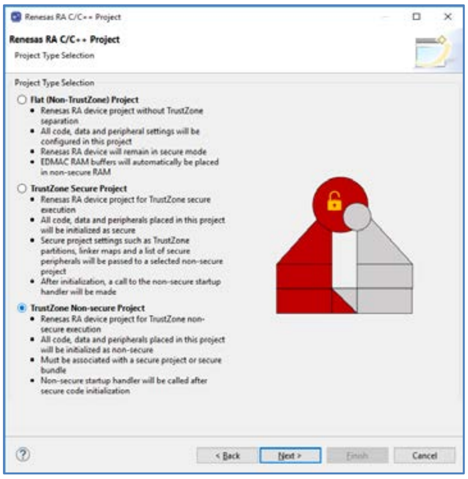
Project Type Selection

○ **Flat (Non-TrustZone) Project**
- Renesas RA device project without TrustZone separation
- All code, data and peripheral settings will be configured in this project
- Renesas RA device will remain in secure mode
- EDMAC RAM buffers will automatically be placed in non-secure RAM

● **TrustZone Secure Project**
- Renesas RA device project for TrustZone secure execution
- All code, data and peripherals placed in this project will be initialized as secure
- Secure project settings such as TrustZone partitions, linker maps and a list of secure peripherals will be passed to a selected non-secure project
- After initialization, a call to the non-secure startup handler will be made

○ **TrustZone Non-secure Project**
- Renesas RA device project for TrustZone non-secure execution
- All code, data and peripherals placed in this project will be initialized as non-secure
- Must be associated with a secure project or secure bundle
- Non-secure startup handler will be called after secure code initialization

Figure 4. Secure Project (following Arm notation as green)



Renesas RA C/C++ Project

Project Type Selection

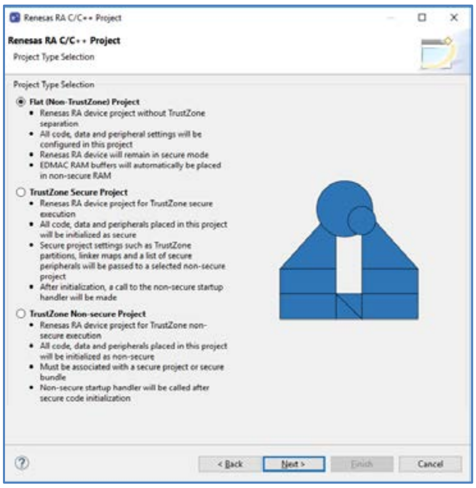
Project Type Selection

○ **Flat (Non-TrustZone) Project**
- Renesas RA device project without TrustZone separation
- All code, data and peripheral settings will be configured in this project
- Renesas RA device will remain in secure mode
- EDMAC RAM buffers will automatically be placed in non-secure RAM

○ **TrustZone Secure Project**
- Renesas RA device project for TrustZone secure execution
- All code, data and peripherals placed in this project will be initialized as secure
- Secure project settings such as TrustZone partitions, linker maps and a list of secure peripherals will be passed to a selected non-secure project
- After initialization, a call to the non-secure startup handler will be made

● **TrustZone Non-secure Project**
- Renesas RA device project for TrustZone non-secure execution
- All code, data and peripherals placed in this project will be initialized as non-secure
- Must be associated with a secure project or secure bundle
- Non-secure startup handler will be called after secure code initialization

Figure 5. Non-Secure Project (following Arm notation as red)



Renesas RA C/C++ Project

Project Type Selection

Project Type Selection

● **Flat (Non-TrustZone) Project**
- Renesas RA device project without TrustZone separation
- All code, data and peripheral settings will be configured in this project
- Renesas RA device will remain in secure mode
- EDMAC RAM buffers will automatically be placed in non-secure RAM

○ **TrustZone Secure Project**
- Renesas RA device project for TrustZone secure execution
- All code, data and peripherals placed in this project will be initialized as secure
- Secure project settings such as TrustZone partitions, linker maps and a list of secure peripherals will be passed to a selected non-secure project
- After initialization, a call to the non-secure startup handler will be made

○ **TrustZone Non-secure Project**
- Renesas RA device project for TrustZone non-secure execution
- All code, data and peripherals placed in this project will be initialized as non-secure
- Must be associated with a secure project or secure bundle
- Non-secure startup handler will be called after secure code initialization

Figure 6. Flat Project

3.1 Secure Project Set Up

All code, data, and peripherals in this project will be configured as Secure using the device Peripheral Security Attribution (PSA) registers. Although it is very application specific, we recommend keeping the Secure project code as small as possible to reduce the attack surface. For example, secure key handling may be the only application code in the secure project.

Necessary values to set up the TZ memory partition (IDAU registers) will be automatically calculated after the project is built to ensure they match the code and data size, keeping the attack surface as small as possible.

Typically, ANSI C start up code (clearing of RAM, variable initialisation, etc) , clock, and secure peripheral initialisation will occur in this project.

At the end of the Secure code, a call will be made to `R_BSP_NonSecureEnter();` to pass control to the Non-Secure partition.

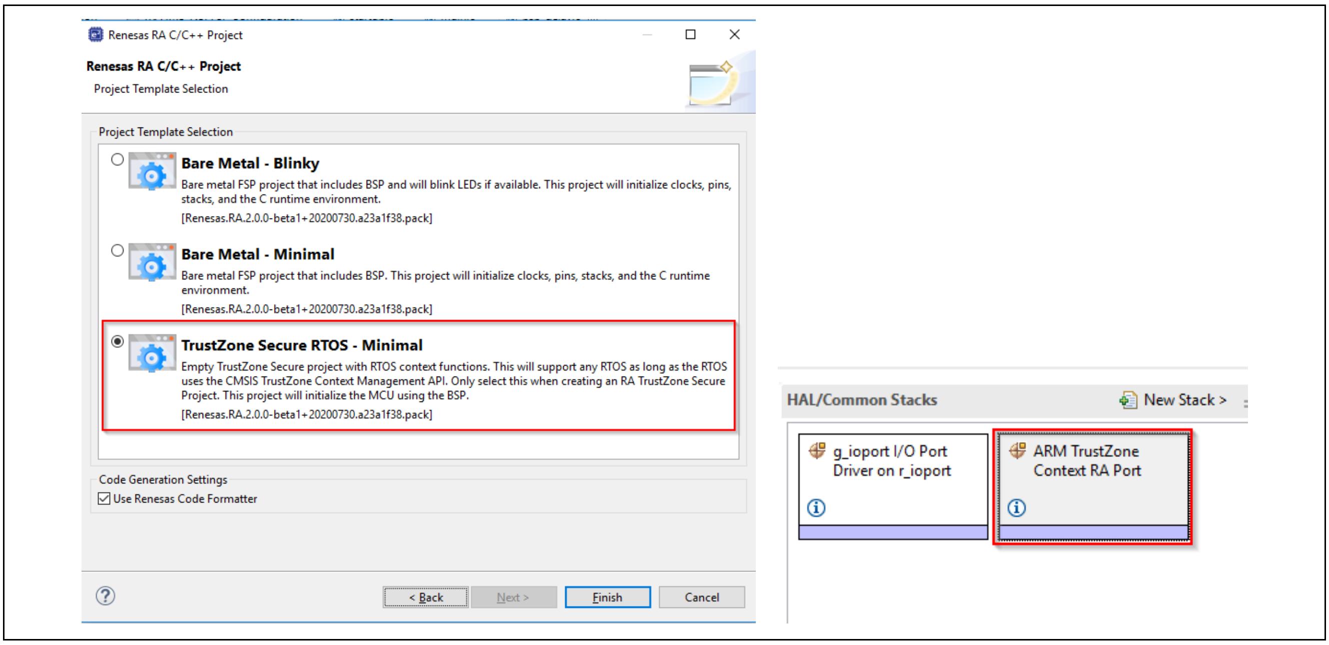
Non-Secure Callable (NSC) “Guard” functions are added to the project and expose selected modules to Non-Secure projects. User can add application-specific access checks as needed in these functions.

Output of this project type will be an elf file that must be either pre-programmed (provisioned) into a device or referenced by a Non-Secure project (via Secure bundle `*.SBD`) to build a final image.

This project type will NOT typically be debugged in isolation and will normally require a Non-Secure project such as a call to a `R_BSP_NonSecureEnter()` to be made. This can be replaced with `while(1);` if needed.

3.2 RTOS Support in TZ Project

Although the RTOS kernel and user tasks will reside in the Non-Secure partition, the Secure partition needs to allocate stack space and so on. It is essential when starting a new RTOS project that the Arm® TrustZone® Secure RTOS-Minimal template is selected. This will add the Arm TrustZone Context RA Port as below.



Renesas RA C/C++ Project

Project Template Selection

Project Template Selection

○ **Bare Metal - Blinky**
Bare metal FSP project that includes BSP and will blink LEDs if available. This project will initialize clocks, pins, stacks, and the C runtime environment.
[Renesas.RA.2.0.0-beta1+20200730.a23a1f38.pack]

○ **Bare Metal - Minimal**
Bare metal FSP project that includes BSP. This project will initialize clocks, pins, stacks, and the C runtime environment.
[Renesas.RA.2.0.0-beta1+20200730.a23a1f38.pack]

● **TrustZone Secure RTOS - Minimal**
Empty TrustZone Secure project with RTOS context functions. This will support any RTOS as long as the RTOS uses the CMSIS TrustZone Context Management API. Only select this when creating an RA TrustZone Secure Project. This project will initialize the MCU using the BSP.
[Renesas.RA.2.0.0-beta1+20200730.a23a1f38.pack]

Code Generation Settings

☑ Use Renesas Code Formatter

Figure 7. Secure RTOS-Minimal Template

3.3 Peripheral Security Attribution

Each peripheral can be configured to be Secure or Non-Secure. Peripherals are divided into two types.

Type-1 peripherals have one security attribute. Access to all registers is controlled by one security attribute. The Type-1 peripheral security attribute is set in the PSARx (x = B to E) register by the secure application.

Type-2 peripherals have the security attribute for each register or for each bit. Access to each register or bit field is controlled according to these security attributes. The Type-2 peripheral security attribute is set in the Security Attribution register in each module by the Secure application. For more information about the Security Attribution register, see sections in the Appropriate MCU’s User’s Manual for each peripheral.

Table 1. Secure and Non-Secure Peripherals

Type	Peripheral
Type 1	SCI, SPI, USBFS, CAN, IIC, SCE9, DOC, SDHI, SSIE, CTSU, CRC, CAC, TSN, ADC12, DAC12, POEG, AGT, GPT, RTC, IWDT, WDT
Type 2	System control (Resets, LVD, Clock Generation Circuit, Low Power Modes, Battery Backup Function), FLASH CACHE, SRAM controller, CPU CACHE, DMAC, DTC, ICU, MPU, BUS, Security setting, ELC, I/O ports
Always Non-Secure	CS Area Controller, QSPI, OSPI, ETHERC, EDMAC

FSP will initialise the arbitration registers during Secure project BSP start up. User code may also be written to set or clear further arbitration. However, care must be taken not to undermine FSP.

3.4 Non-Secure

All code, data, and peripherals in this project will be configured as Non-Secure. This project type must be associated with a Secure project to enable access to secure code, peripherals, linker scripts and others.

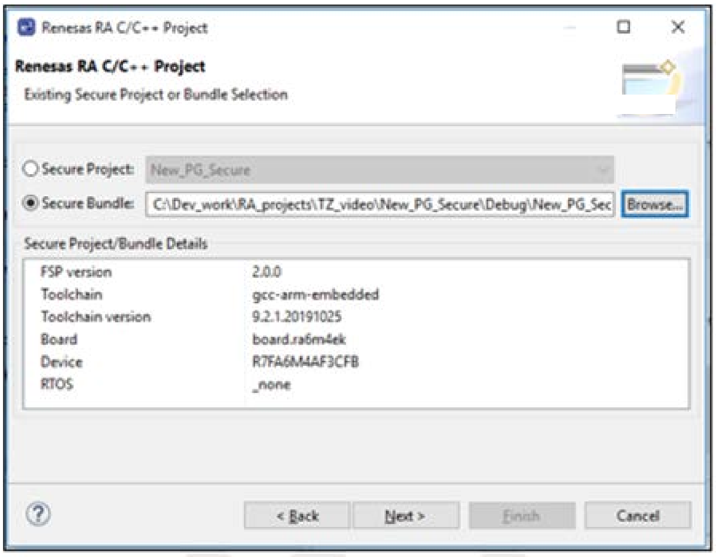
3.5 Flat Project Type

All code, data, and peripherals are configured in a Secure single partition except for the EDMAC RAM buffers that will remain in the Non-Secure partition. Effectively, TZ is disabled.

3.6 Secure Connection to Non-Secure Project

When starting a new Non-Secure Project, the user will be prompted for either a Secure Project or Secure Bundle. In each case, details of the linker settings, Non-Secure Callable functions, and Secure peripherals will be read to enable the Non-Secure project setup.

Should the Secure project or bundle be rebuilt, the Non-Secure editor will detect this and prompt user to regenerate the Non-Secure project configuration.



Renesas RA C/C++ Project

Renesas RA C/C++ Project

Existing Secure Project or Bundle Selection

○ Secure Project: New_PG_Secure

● Secure Bundle: C:\Dev_work\RA_projects\TZ_video\New_PG_Secure\Debug\New_PG_Sec

Secure Project/Bundle Details

FSP version	2.0.0
Toolchain	gcc-arm-embedded
Toolchain version	9.2.1.20191025
Board	board.ra6m4ek
Device	R7FA6M4AF3CFB
RTOS	_none

Figure 8. Secure Project or Bundle Selection

3.6.1 Secure Project (Combined)

A Secure project must reside in the same Workspace as the Non-Secure project and will typically be used when a design engineer has access to both the Secure and Non-Secure project sources. This is sometimes known as “Combined model”.

A Secure `.elf` file will be referenced and included in the debug configuration for download to the target device. The development engineer will have visibility of Secure and Non-Secure project source code and configuration.

3.6.2 Secure Bundle (Split)

A Secure Bundle will ONLY include linker memory ranges, symbol references, and details of locked Secure peripheral configuration settings but no access to Secure source code (API header files will be included as necessary).

The Secure bundle file (`*.SBD`) must be supplied to the Non-Secure developer by the Secure project developer.

The development engineer will typically not have access to the Secure project or `.elf` file which MUST be pre-programmed or provisioned into the target MCU.

The DLM state of target device should then be switched to NSECSD (see section 6.2) before the device is provided to the non-secure developer.

This is often referred to as "Split model" where a basic security set up is developed by a Secure team and then passed to the Non-Secure team in the same facility or at a third party. The Non-Secure team has no access to the Secure source code and cannot directly access Secure peripherals, data, or APIs.

3.7 Debug Configurations

After each project type has been selected, a suitable debug configuration will be generated.

3.7.1 Non-Secure with Secure Project (Combined)

Both Secure and Non-Secure `.elf` files will be downloaded.

A debug configuration called `<project name>_SSD` will be generated.

3.7.2 Non-Secure with Secure Bundle (Split)

Only a Non-Secure elf will be downloaded. This configuration must be used with a pre-provisioned device (Secure project pre-programmed into MCU Flash).

A debug configuration called `<project name>_NSECSD` will be generated.

3.7.3 Flat Debug

A single `.elf` file will be downloaded.

A debug configuration called `<project name>_FLAT` will be generated.

4. Secure Projects

As mentioned, Secure code will be called immediately after device reset and run ANSI C start up, clock, interrupt vector table, and secure peripheral initialization before starting user code. All selected peripheral configuration settings will be automatically initialised as Secure.

4.1 Secure Clock

Device clock settings are the possible exception in that they will be initialised in the Secure project (to enable faster start up from reset) but can be set as Secure or Non-Secure as user application may need to change settings during execution (for low-power mode and so on). The Secure and Non-Secure FSP BSPs can both change the clock settings.

However, clock settings can be locked as Secure should the developer choose to do so.

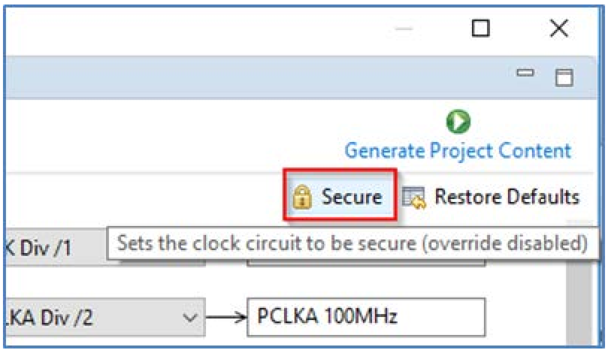


Figure 9. Secure Clock Setting

4.2 Setting Drivers as NSC

Some driver and middleware stacks in the Secure project may need to be accessed by the Non-Secure partition. To enable generation of NSC veneers, set "Non-Secure Callable" from the right-click context menu for the selected modules in the Configurator.

Note: It is only possible to "expose" top of stacks as NSC.

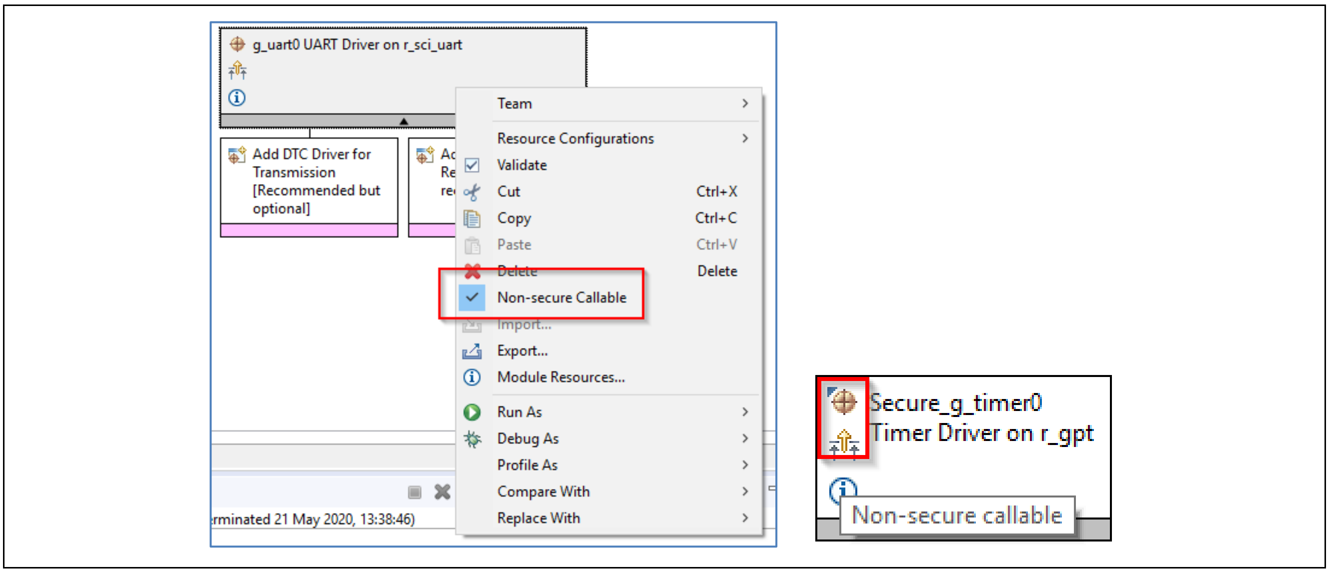


Figure 10. Generate NSC Veneers

The top of the stack will be marked with a new icon and tool tip to signify NSC access.

4.3 Guard Functions

Access to NSC drivers from a Non-Secure project is possible through the Guard APIs. FSP will automatically generate Guard functions for all the top of stack/driver APIs added to the project as Non-Secure Callable.

User can choose to add further levels of access control or delete guard function if they wish to only expose a limited range of APIs to a Non-Secure developer.

```
BSP_CMSE_NONSECURE_ENTRY fsp_err_t g_uart0_open_guard(
   uart_ctrl_t *const p_api_ctrl, uart_cfg_t const *const p_cfg) {
  /* TODO: add your own security checks here */

  FSP_PARAMETER_NOT_USED(p_api_ctrl);
  FSP_PARAMETER_NOT_USED(p_cfg);

  return R_SCI_UART_Open(&g_uart0_ctrl, &g_uart0_cfg);
}
```

For example, an SCI channel may be opened and configured for a desired baud rate by the Secure developer, but only enable the Write API to the Non-Secure developer. In which case, all but `g_uart0_write_guard()` could be deleted. CTRL structures are not required as they will be added on the Secure side.

For example, the call from the Non-Secure partition would be as follows:

```
err = g_uart0_open_guard(0,0);
```

See FSP documentation for more details.

5. Non-Secure projects

Configuration of the project can continue as for other RA devices, but certain resources will be locked if they have been previously set up as Secure.

The Non-Secure project will be called from the Secure project via

```
R_BSP_NonSecureEnter();
```

5.1 Clock Set Up

You may recall that clocks can be set as Secure or Non-Secure. If they are set as Secure, settings will only be available to view, and user will not be able to change them. The Override button will be greyed. This is useful to preserve CGC sync with secure project by not overriding unless necessary. If it is NOT set as Secure, user can choose to override the initial Secure settings

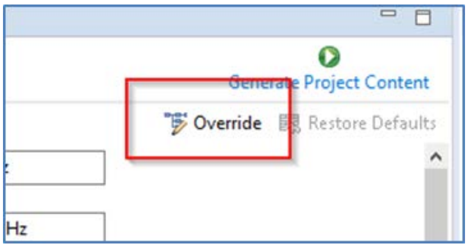


Figure 11. Clock Setting as Non-Secure

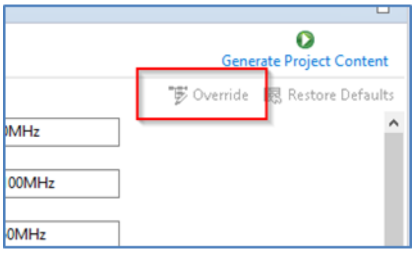


Figure 12. Clock Setting as Secure

5.2 Selecting NSC Drivers

Drivers declared as NSC in a Secure project can be selected and added to Non-Secure project and will be decorated as before.

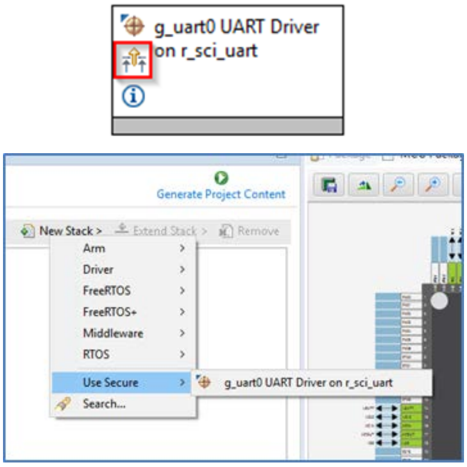
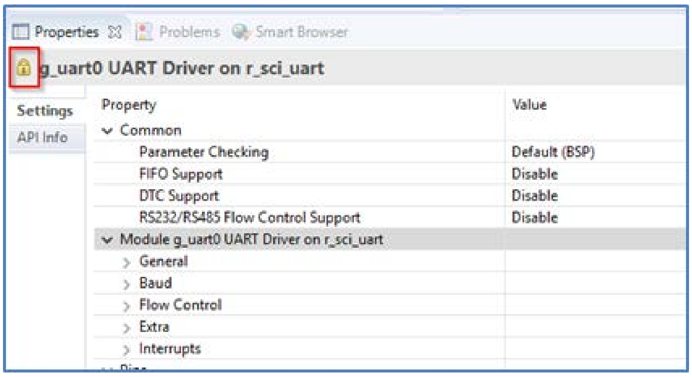


Figure 13. Selecting NSC Drivers

5.3 Locked Resources

When a NSC Secure driver is added to a Non-Secure project, the configuration settings are locked and are available for information only. A padlock is added for indication.



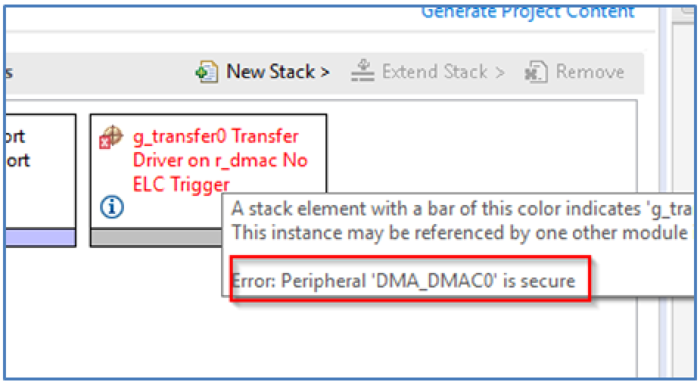
g_uart0 UART Driver on r_sci_uart

Property	Value
Common	
Parameter Checking	Default (BSP)
FIFO Support	Disable
DTC Support	Disable
RS232/RS485 Flow Control Support	Disable
Module g_uart0 UART Driver on r_sci_uart	
General	
Baud	
Flow Control	
Extra	
Interrupts	

Figure 14. Locked Resources

5.4 Locked Channels

In a peripheral with multiple channels, for example, DMA, if a Non-Secure developer tries to select a channel that has already been defined as Secure, the following error message type will be displayed.



g_transfer0 Transfer Driver on r_dmac No ELC Trigger

A stack element with a bar of this color indicates 'g_tra
This instance may be referenced by one other module

Error: Peripheral 'DMA_DMAC0' is secure

Figure 15. Error Message when Selecting a Secure Channel

6. IDAU registers

Renesas RA TZ-enabled devices include a set of registers known as Implementation Defined Attribution Unit (IDAU) that are used to set up partitions between Secure, Non-Secure Callable, and Non-Secure regions. The IDAU registers can only be programmed during MCU ***boot mode*** and NOT through the debug interfaces. Because of this, special debugger firmware has been developed to manage bringing the device up in SCI boot mode to set up the IDAU registers (automatically drives MD pin) and then switch back to debug mode as needed.

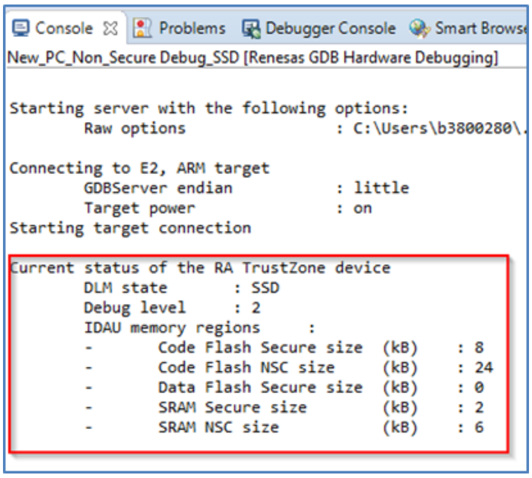
Note: Please be aware of the extra signal connection (MD pin) needed on the debug interface connector. The Renesas Evaluation Kit (EK) for your selected device is a good reference.

Pin No.	SWD	JTAG	Serial Programming using SCI
1	VCC	VCC	VCC
2	P108/SWDIO	P108/TMS	NC
4	P300/SWCLK Wired OR with MD	P300/TCK Wired OR with MD	P201/MD
6	P109/SWO/TXD9	P109/TDO/TXD9	P109/TXD9
8	P110/RXD9	P110/TDI/RXD9	P110/RXD9
9	GNDdetect	GNDdetect	GNDdetect
10	nRESET	nRESET	nRESET
12	P214/TRACECLK	P214/TRACECLK	NC
14	P211/TRACEDATA[0]	P211/TRACEDATA[0]	NC
16	P210/TRACEDATA[1]	P210/TRACEDATA[1]	NC
18	P209/TRACEDATA[2]	P209/TRACEDATA[2]	NC
20	P208/TRACEDATA[3]	P208/TRACEDATA[3]	NC
3, 5, 15,17, 19	GND	GND	GND
7	NC	NC	NC
11, 13	NC	NC	NC

The e^2 studio build phase automatically extracts the IDAU partition register settings from the Secure `.elf` file and programs them into the device during debug connection, which can be observed in the console.

This is an important phase of TZ development as the Secure partitions should be set as small as possible to ensure that the security attack surface is as small as possible.

However, should the developer wish to make these partitions larger to accommodate, for example during field firmware updates, const or data arrays should be placed in the Secure project as needed.



New_PC_Non_Secure Debug_SSD [Renesas GDB Hardware Debugging]

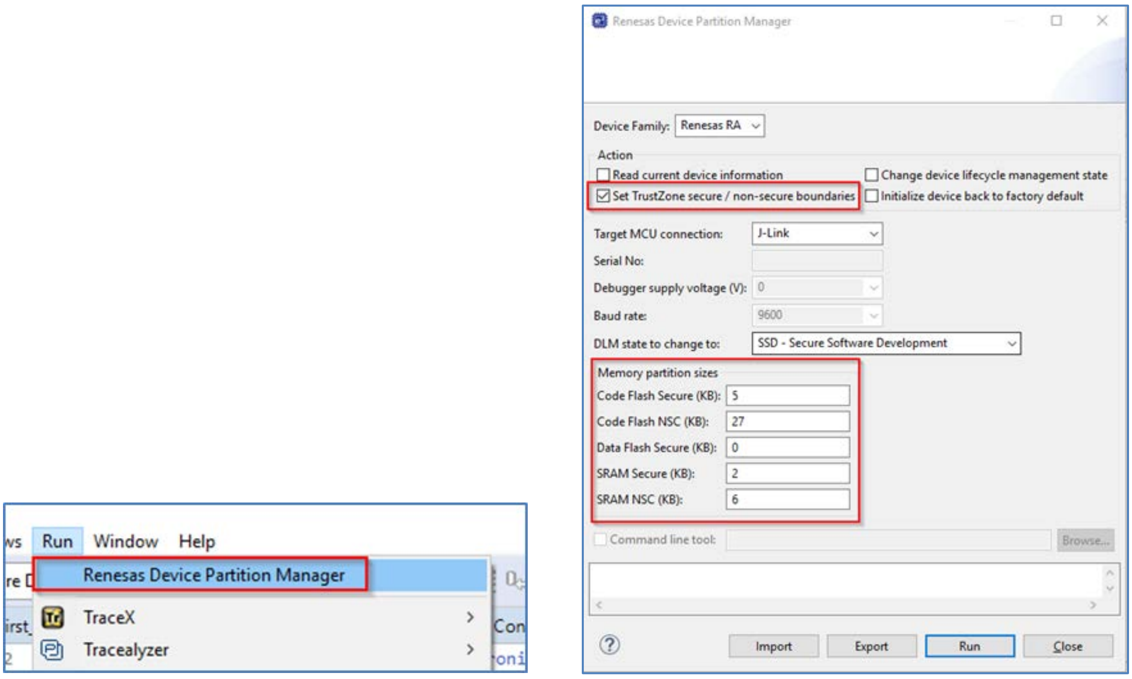
```
Starting server with the following options:
        Raw options                   : C:\Users\b3800280\.

Connecting to E2, ARM target
        GDBServer endian              : little
        Target power                  : on
Starting target connection

Current status of the RA TrustZone device
        DLM state         : SSD
        Debug level       : 2
        IDAU memory regions       :
        -       Code Flash Secure size  (kB)    : 8
        -       Code Flash NSC size     (kB)    : 24
        -       Data Flash Secure size  (kB)    : 0
        -       SRAM Secure size        (kB)    : 2
        -       SRAM NSC size           (kB)    : 6
```

Figure 16. RA TrustZone Device Current Status

It is also possible to manually set up the partition registers through the **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

Memory partition sizes

Code Flash Secure (KB): 5

Code Flash NSC (KB): 27

Data Flash Secure (KB): 0

SRAM Secure (KB): 2

SRAM NSC (KB): 6

☐ Command line tool:

Figure 17. Renesas Device Partition Manager

6.1 SCI Boot Mode

Example of MD mode pin connection to debugger connector (from EK schematic).

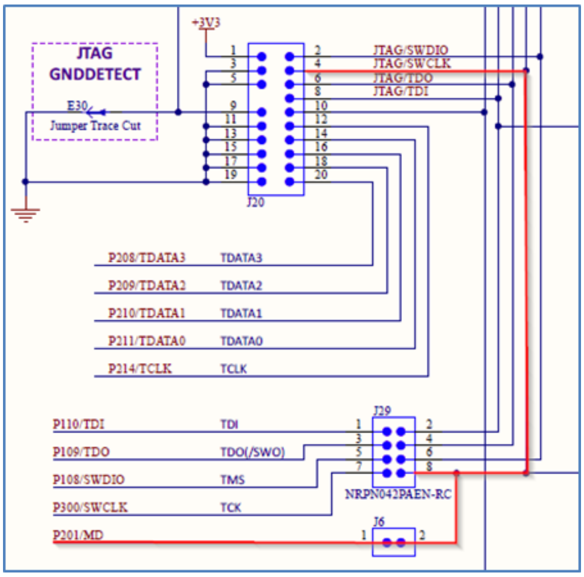


Figure 18. Example of MD Mode Pin Connection to Debugger Connector (from EK schematic)

6.2 DLM States

Device lifecycle defines the current phase of the device and controls the capabilities of the debug interface, the serial programming interface and Renesas test mode. The following illustration shows the lifecycle definitions and capability in each lifecycle.

Note: All authentication key exchange and transitioning to LCK_DBG, LCK_BOOT, RMA_REQ is only managed by Renesas Flash Programmer (RFP) and NOT within e^2 studio.

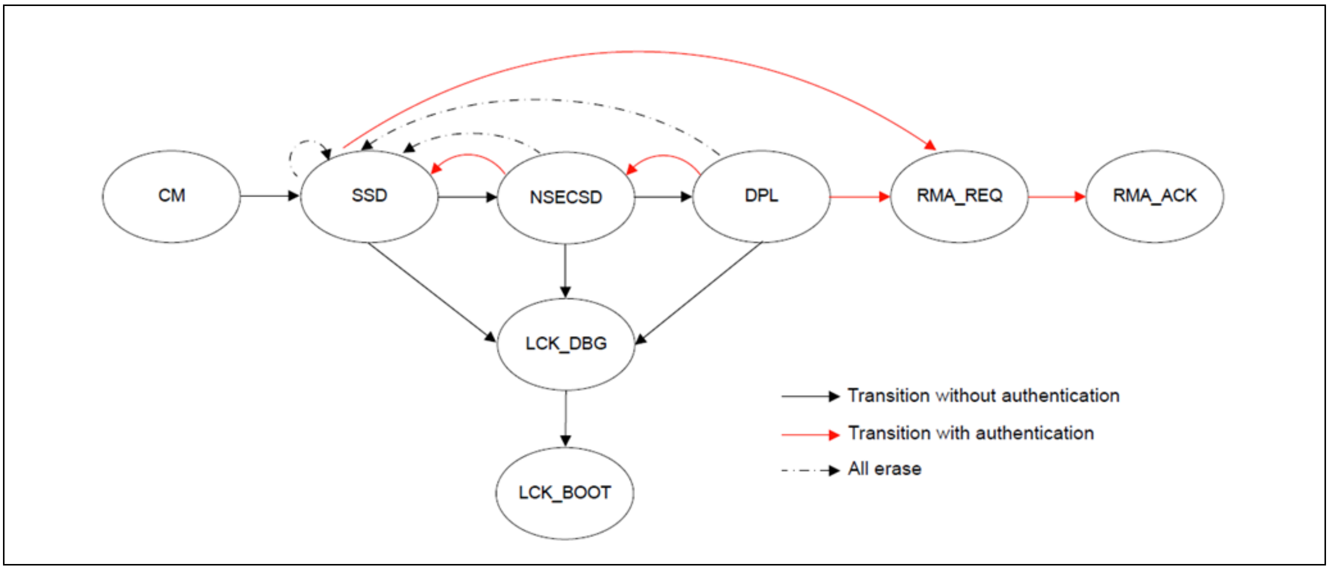
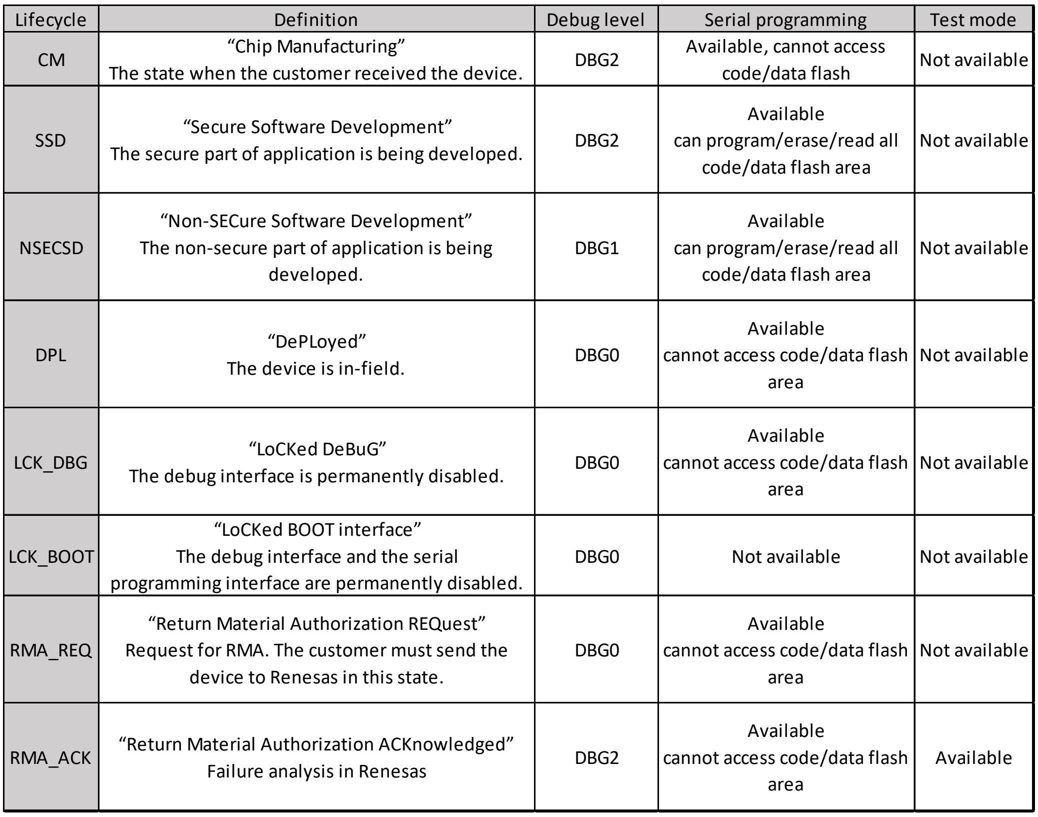

CM
SSD
NSECSD
DPL
RMA_REQ
RMA_ACK
LCK_DBG
LCK_BOOT
Transition without authentication
Transition with authentication
All erase

Figure 19. Lifecycle Stages



Lifecycle	Definition	Debug level	Serial programming	Test mode
CM	"Chip Manufacturing" The state when the customer received the device.	DBG2	Available, cannot access code/data flash	Not available
SSD	"Secure Software Development" The secure part of application is being developed.	DBG2	Available can program/erase/read all code/data flash area	Not available
NSECSD	"Non-SECure Software Development" The non-secure part of application is being developed.	DBG1	Available can program/erase/read all code/data flash area	Not available
DPL	"DePLoyed" The device is in-field.	DBG0	Available cannot access code/data flash area	Not available
LCK_DBG	"LoCKed DeBuG" The debug interface is permanently disabled.	DBG0	Available cannot access code/data flash area	Not available
LCK_BOOT	"LoCKed BOOT interface" The debug interface and the serial programming interface are permanently disabled.	DBG0	Not available	Not available
RMA_REQ	"Return Material Authorization REQuest" 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	"Return Material Authorization ACKnowledged" Failure analysis in Renesas	DBG2	Available cannot access code/data flash area	Available

Figure 20. Lifecycle Stages and Debug Levels

There are three debug access levels. The debug access level changes according to the lifecycle state.

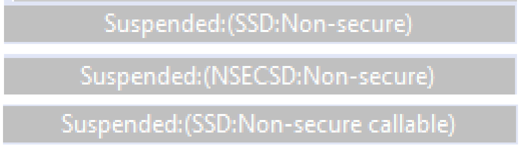
- DBG2: The debugger connection is allowed, and no restriction to access memories and peripherals
- DBG1: The debugger connection is allowed, and restricted to access only Non-Secure memory regions and peripherals
- DBG0: The debugger connection is not allowed

Transitions for one state to another can be performed using the Renesas Flash Programmer (RFP, see section below) or using the Renesas Device Partition Manager (limited number of states possible). It is possible to secure transitions between states using authentication keys. For more information on DLM states and transitions (device specific), please refer to device user manual.

7. Debug

By default, the device will be in SSD mode and so allow access to Secure and Non-Secure partitions. In this mode both Secure and Non-Secure `.elf` files will be downloaded.

The current debugger status is displayed in the lower left corner and includes the DLM state (SSD or NSECSD) and current partition (Secure, Non-Secure, or Non-Secure Callable) when the debugger is stopped, for example.



Suspended:(SSD:Non-secure)

Suspended:(NSECSD:Non-secure)

Suspended:(SSD:Non-secure callable)

Figure 21. Current Debugger Status

7.1 Non-Secure Debug

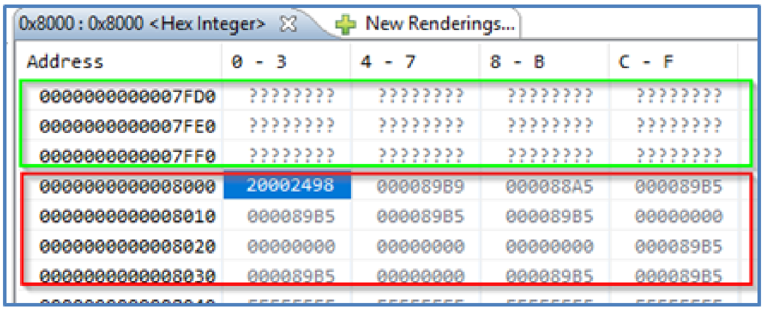
Once the device is transitioned to NSECSD mode, only Non-Secure Flash, RAM and Peripherals can be accessed. In this mode, a Secure `.elf` must be pre-programmed (provisioned) into the device, and only a Non-Secure `.elf` file will be downloaded.

When in NSECSD mode access to Secure elements will be blocked and data displayed as ????????.

In NSECSD mode, it is not possible to set breakpoints on Secure code or data.

It is not possible to step into Secure code; the debugger will perform a step-over of any Secure function calls. Should the user press the Suspend button during execution, the debugger will stop at the next Non-Secure code access.

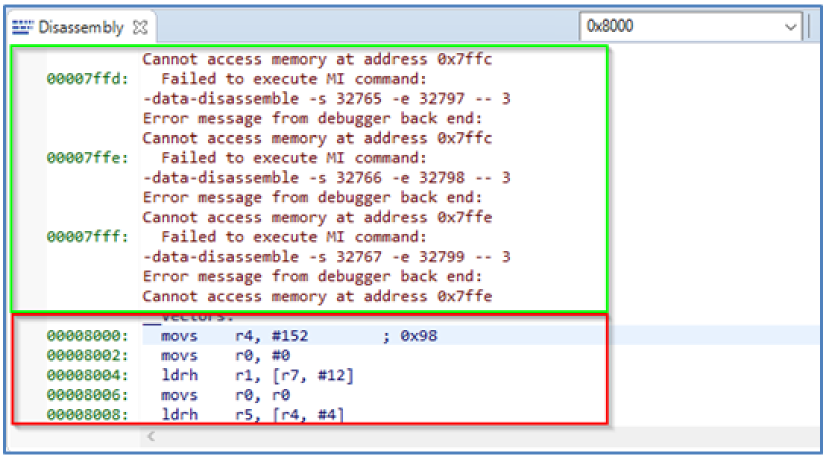
Assuming Secure memory region finishes at 32K (0x8000) in NSECSD debug mode (colour coding added for indication only), memory will be displayed as shown in the following figure.



```
Address             0 - 3      4 - 7      8 - B      C - F
0000000000007FD0    ????????   ????????   ????????   ????????
0000000000007FE0    ????????   ????????   ????????   ????????
0000000000007FF0    ????????   ????????   ????????   ????????
0000000000008000    20002498   000089B9   000088A5   000089B5
0000000000008010    000089B5   000089B5   000089B5   00000000
0000000000008020    00000000   00000000   00000000   000089B5
0000000000008030    000089B5   00000000   000089B5   000089B5
```

Figure 22. Memory Display in NSECSD Debug Mode

Disassembly will be displayed as shown in the following figure.



```
             Cannot access memory at address 0x7ffc
00007ffd:      Failed to execute MI command:
             -data-disassemble -s 32765 -e 32797 -- 3
             Error message from debugger back end:
             Cannot access memory at address 0x7ffc
00007ffe:      Failed to execute MI command:
             -data-disassemble -s 32766 -e 32798 -- 3
             Error message from debugger back end:
             Cannot access memory at address 0x7ffe
00007fff:      Failed to execute MI command:
             -data-disassemble -s 32767 -e 32799 -- 3
             Error message from debugger back end:
             Cannot access memory at address 0x7ffe
             __vectors:
00008000:      movs    r4, #152        ; 0x98
00008002:      movs    r0, #0
00008004:      ldrh    r1, [r7, #12]
00008006:      movs    r0, r0
00008008:      ldrh    r5, [r4, #4]
```

Figure 23. Disassembly Display in NSECSD Debug Mode

8. Debugger support

Renesas E2, E2 Lite, and SEGGER J-Link are supported in e^2 studio for TZ projects.

Table 1. Debugger Support for TZ Projects

Feature	E2 Lite	E2	J-Link	J-Link OB	ULINK	IAR i-Jet
JTAG	Yes	Yes	Yes	No	Yes	Yes
SWD	Yes	Yes	Yes	Yes	Yes	Yes
ETB trace	Yes	Yes	Yes	Yes	Yes	Yes
ETM trace	No	Yes	Yes	No	Yes	Yes
TZ partition programming	Yes	Yes	Yes	Yes	No	No
Non secure debug	Yes	Yes	Yes	Yes	Yes	Yes
e^2 studio	Yes	Yes	Yes	Yes	No	TBC
IAR EW Arm	Under consideration		Yes	Yes	No	Yes
Keil MDK			Yes	Yes	Yes	No

9. Third-Party IDEs

Third-party IDEs such as IAR Systems EWARM and Keil MDK (uVision) are supported by the RA Smart Configurator (RA SC).

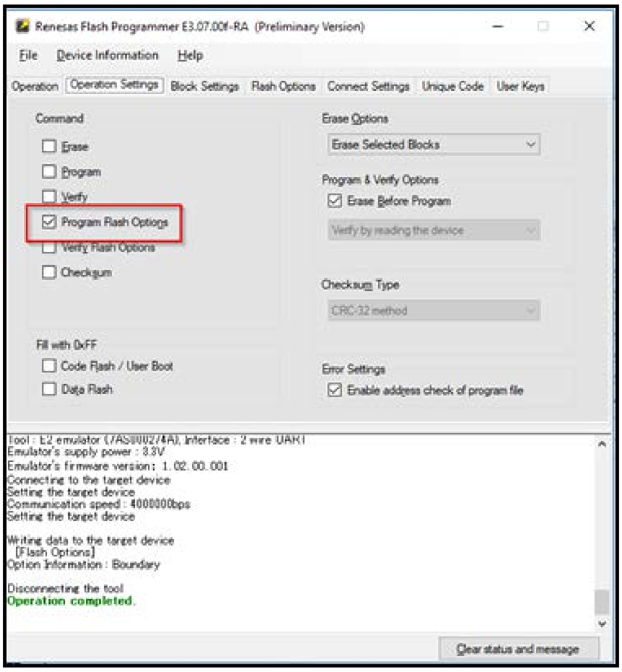
In general, RA SC offers the same configurator functionality as e^2 studio documented above. Project generators are available to initialise workspaces in the target IDEs as well as setting up debug configurations and so forth. However, there are some limitations that need to be noted especially with regards to IDAU TZ partition register programming. See the specific RA SC documentation for usage details.

10. Renesas Flash Programmer (RFP)

Updated versions of Renesas Flash Programmer (RFP) are available to support setting of partitions, DLM state and Authentication keys.

RFP can be downloaded free of charge on the Renesas web site.

A new mode has been added to Program Flash Options as shown in the following graphics.



Command
- ☐ Erase
- ☐ Program
- ☐ Verify
- ☑ Program Flash Options
- ☐ Verify Flash Options
- ☐ Checksum

Fill with 0xFF
- ☐ Code Flash / User Boot
- ☐ Data Flash

Erase Options: Erase Selected Blocks

Program & Verify Options
- ☑ Erase Before Program
- Verify by reading the device

Checksum Type: CRC-32 method

Error Settings
- ☑ Enable address check of program file

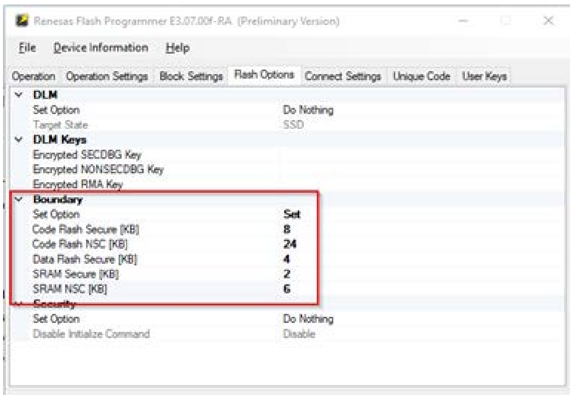
```
Tool : E2 emulator (7AS000274A), Interface : 2 wire UART
Emulator's supply power : 3.3V
Emulator's firmware version: 1.02.00.001
Connecting to the target device
Setting the target device
Communication speed : 4000000bps
Setting the target device

Writing data to the target device
  [Flash Options]
Option Information : Boundary

Disconnecting the tool
Operation completed.
```

Figure 24. RFP Program Flash Options

Options to set partition boundaries are shown in the following figure.

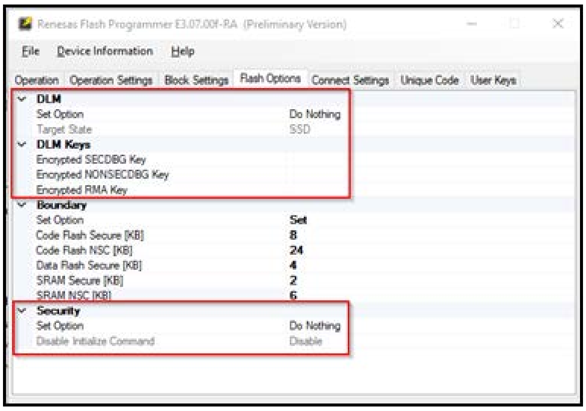


Renesas Flash Programmer E3.07.00f-RA (Preliminary Version)

DLM	
Set Option	Do Nothing
Target State	SSD
DLM Keys	
Encrypted SECDBG Key	
Encrypted NONSECDBG Key	
Encrypted RMA Key	
Boundary	
Set Option	**Set**
Code Flash Secure [KB]	**8**
Code Flash NSC [KB]	**24**
Data Flash Secure [KB]	**4**
SRAM Secure [KB]	**2**
SRAM NSC [KB]	**6**
Security	
Set Option	Do Nothing
Disable Initialize Command	Disable

Figure 25. RFP Partition Boundaries

Options to set DLM state, Authentication keys, and Security settings are shown in the following figure.



Renesas Flash Programmer E3.07.00f-RA (Preliminary Version)

DLM	
Set Option	Do Nothing
Target State	SSD
DLM Keys	
Encrypted SECDBG Key	
Encrypted NONSECDBG Key	
Encrypted RMA Key	
Boundary	
Set Option	**Set**
Code Flash Secure [KB]	**8**
Code Flash NSC [KB]	**24**
Data Flash Secure [KB]	**4**
SRAM Secure [KB]	**2**
SRAM NSC [KB]	**6**
Security	
Set Option	Do Nothing
Disable Initialize Command	Disable

Figure 26. RFP DLM State, Authentication Keys, and Security Settings

Great care is needed here as some DLM states can **permanently** turn off debug and boot mode on the devices. Equally programming a security access authentication key can lead to permanently locked devices if the key is lost.

11. Glossary

Term	Definition
IDAU	Implementation Defined Attribute Unit. Used to program TZ partitions in SCI book mode.
NSECSD	Non-Secure Software Development mode
SSD	Secure Software Development mode
NSC	Non-Secure Callable. Special Secure memory region used for Veneer to allow access to Secure APIs from Non-Secure code.
Provisioned	Device with Secure code pre-programmed and DLM state set to **NSECSD**
Flat project	All code, data and peripherals are configured as secure with the exception of the EDMAC RAM buffer which are placed in Non-Secure RAM due to the configuration of the internal bus masters.
Veneer	Code that resides in Non-Secure Callable region
Combined model	Development engineer has access to both Secure and Non-Secure project and source code
Split model	Development Engineer has access to only the Non-Secure partition. No visibility of Secure source code. Secure code will be provisioned into device.

11.1 Configurator Icon Glossary

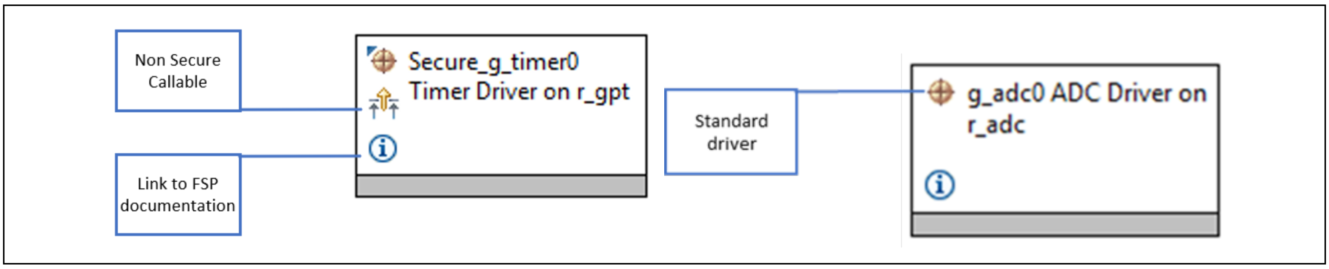

Non Secure Callable
Link to FSP documentation
Secure_g_timer0 Timer Driver on r_gpt
Standard driver
g_adc0 ADC Driver on r_adc

Figure 27. Configurator Icons

12. Next Steps

1. Read the RA FSP User’s Manual for details of guard functions and general Arm® TrustZone® support.
2. Read the RA Device Hardware Manual for detailed Arm® TrustZone® security information.
3. Read the *Renesas e^2 studio Getting Started Guide*.

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	Aug.20.20	—	First release document
1.01	Oct. 01.20	—	Updated trademark and target device information

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.

(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.4.0-1 November 2017)

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/.