RZ/T2ME Group

On The Fly Decryption

Introduction

This application note describes how to perform On-The-Fly Decryption on the RZ/T2ME using the Renesas Starter Kit+ for the RZ/T2ME.

Target Device

RZ/T2ME Group

Contents

1. Overview

1.1 Introduction

The RZ/T2ME Group OTFD sample program package shows how to perform On-The-Fly Decryption (OTFD) on the RZ/T2ME using the Renesas Starter Kit+ for the RZ/T2ME.

OTFD for RZ/T2ME is a function that executes an encrypted program stored on external flash memory in-place while decrypting it. Use this function when the application program needs to be kept secret and you want to execute the application program on the external flash memory.

OTFD can keep an application program confidential, but it cannot detect the program tampering. If integrity is required for the target program of OTFD, the program must be verified in combination with the Secure Boot function for RZ/T2ME.

This sample program package shows how to use the OTFD feature of RZ/T2ME Group in the following ways:

1. Download the sample program including the encrypted program to the evaluation board via the debugger, and execute without secure boot and debug it with the IDE.
2. How to secure boot the loader program and execute the encrypted program on the flash memory securely or as is.

To use the OTFD function of the RZ/T2ME Group, you need the OTFD control software included in the RZ/T2ME Flexible Software Package (FSP), the Renesas Secure IP driver, a program encryption tool for OTFD, and a key injection program.

This sample program in-place-executes the blinking process of the Blinky program for RZ/T2ME Group on the external flash memory and decrypts the encrypted blinking process on the external flash memory on the fly.

The FSP, Renesas Secure IP driver, and Secure Key Management Tool (SKMT) CLI version, one of the program encryption tools for OTFD, are included in this package.

When combined with Secure Boot, key injection uses the Secure Boot sample program package (R01AN7376EJ). The Secure Boot sample program package includes a Python tool with program encryption for OTFD.

1.2 Features

The sample program package has the following features:

1. Demonstration project for OTFD simple operation check:
   Download the sample program including the encrypted program to the evaluation board via the debugger, and execute without secure boot and debug it with the IDE.
   - Download the sample program to RZ/T2ME and flash memory on the target board via debugger and execute it to confirm the operation of OTFD.
   - Use it as a reference when checking the OTFD function or when developing a program that enables the OTFD function.
   - Use Secure Key management Tool (SKMT) CLI version for encryption of the blinking processing part of the sample program.
2. Demonstration project OTFD with Secure Boot:
   How to secure boot the loader program and execute the encrypted program on the flash memory securely or as is.
   - Program the sample program in the flash memory on the target board and boot in xSPI0 boot mode to check the operation of OTFD. This sample program shows how to execute the program to be decrypted by the OTFD function after integrity verification and how to execute it as is.
   - Use this as a reference for software implementation when building a system to boot an OTFD program on flash.
   - Use the Secure Boot Sample Program Package for flash memory programming of the sample program, encryption of the blinking processing part of the sample program, and key injection.

1.3 Limitations

No limitations in this sample program.

1.4 Package Contents

RZ/T2ME Group OTFD sample program package contains several files with software and documents. The following table lists their contents.

Table 1.1 RZ/T2ME Group OTFD Sample Program Package Contents

No.	File Name	Classification	Remarks
1	RZT2ME_RSK_OTFD_Debug_Rev100.zip	Software	Demonstration project for OTFD simple operation check Note1
2	RZT2ME_RSK_OTFD_xSPI_Boot_Rev100.zip	Software	Demonstration project combining OTFD and Secure Boot Note1
3	otfd_encrypt_aes_keyfile.bin	Data	Programs and data for RZ/T2ME stored in the “Pre-built parameters and programs” folder. These files are used in Section 3.2 of this document as a reference for building the environment. (The application's bin file was generated by the IAR Compiler project)
4	parameter_RZT2ME_RSK_OTFD_app_encrypt.bin	Data	
5	parameter_RZT2ME_RSK_OTFD_loader.bin	Data	
6	RZT2ME_RSK_OTFD_app_encrypt.bin	Software	
7	RZT2ME_RSK_OTFD_loader.bin	Software	
8	RZT2ME_RSK_OTFD_xSPI_Boot.srec	Software	
9	sec_boot_image_app.bin	Software	
10	sec_boot_image_app.pubkey	Data	
11	sec_boot_image_loader.bin	Software	
12	sec_boot_image_loader.pubkey	Data	
13	secureboot_common_keyfile.bin	Data	
14	otfd_encrypt_aes_key.bin	Key	Sample key data stored in the “Key Data” folder. These files are used in Section 3.2 of this document as a reference for building the environment.
15	ikey-pair_ssbl.pem	Key	
16	ikey-pair_userapp.pem	Key	
17	otfd_encrypt_nonce.bin	Key	
18	rootkey-pair.pem	Key	
19	secureboot_common_key.bin	Key	
20	r01an7277ej0100-rzt2me-security-otfd.pdf	Document	This document
21	r01an7440ej0100-rzt2me-security-releasenote.pdf	Document	Release Note

Note1 Contains sample program projects for GCC and IAR compilers.

1.5 Related Documents

Table 1.2 lists documents related to this document.

Table 1.2 Related Documents

Title	Document Number
RZ/T2, RZ/N2 Getting Started with Flexible Software Package	R01AN6434EJ****
RZ/T2M Group Renesas Starter Kit+ for RZ/T2M User’s Manual (RZ/T2M and RZ/T2ME)	R20UT4939******
RZ/T2ME Group Quick Start Guide Renesas Starter Kit+ for RZ/T2ME	R20QS0074EJ****
RZ/T2ME Group User’s Manual: Hardware	R01UH1062EJ****
RZ/T2ME Group Security Features User’s Manual	R01UH1091EJ****
RZ/T2ME Group Renesas Secure IP Driver	R01AN7375EJ****
RZ/T2ME Group Device Setup Guide for Secure boot	R01AN7376EJ****

1.6 Explanation of Terms

The meanings of terms used in this document are indicated below.

Term Used in This Document	Meaning of Term
OTFD	On The Fly Decryption
SKMT	Secure Key management Tool https://www.renesas.com/us/en/software-tool/security-key-management-tool
RSIP	Renesas Secure IP
User Key	Encryption key used in the user application AES key, RSA public key, RSA private key, etc
User Factory Programming Key (UFPK)	256-bit symmetric key data used to wrap User Keys
W-UFPK	UFPK wrapped by the Renesas Key Wrapping service
Renesas Key Wrap Service	A service that wraps a UFPK with a device-specific key to generate a W-UFPK https://dlm.renesas.com/keywrap/

2. On-The-Fly Decryption for RZ/T2ME

RZ/T2ME has two units of Expanded Serial Peripheral Interface (xSPI) and On-The-Fly Decryption (OTFD). The OTFD only support to decrypt data read for xSPI.

The decryption algorithm supported by OTFD is AES-CTR with key lengths of 128-bit, 192-bit, and 256-bit. However, since the RSIP driver available for injecting keys into devices does not support the injection of 192-bit AES keys, only 128-bit and 256-bit decryption keys are currently available for OTFD.

To use the OTFD function of the RZ/T2ME group, you need the xSPI driver with OTFD control function (r_xspi_qspi module) included in the RZ/T2ME Flexible Software Package (FSP), Renesas Secure IP driver (r_rsip module), program encryption tool for OTFD, and key injection program.

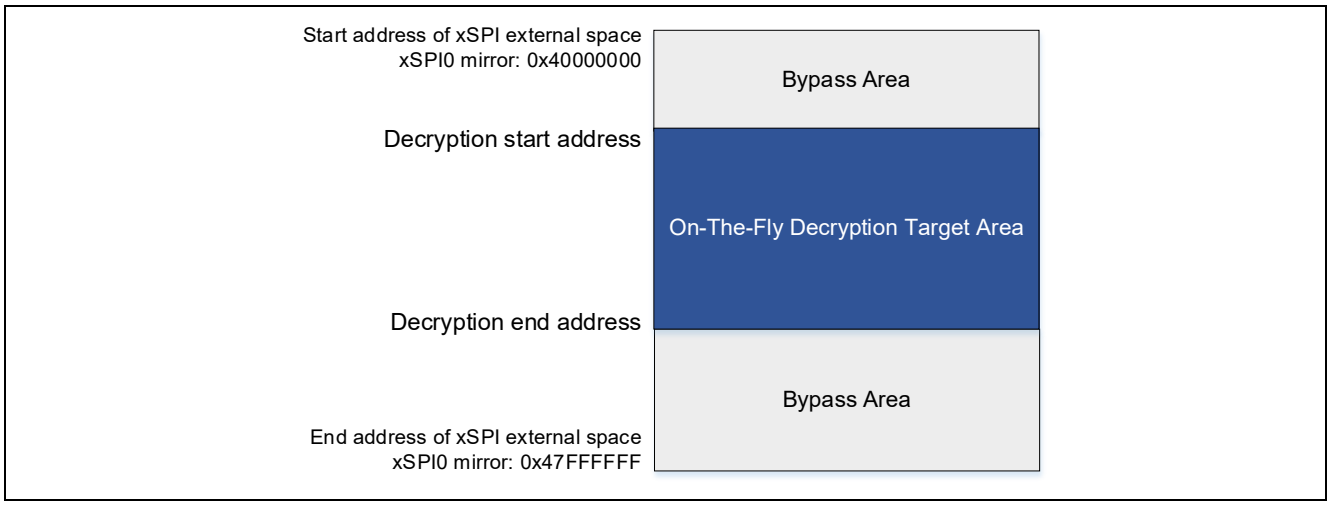


Figure 2.1 On-The-Fly Decryption target area

2.1 FSP Configuration

2.1.1 r_xspi_qspi module

To use OTFD with r_xspi_qspi module, open the FSP configuration screen in the project, select the QSPI module in the Stack tab, and set the items required to use OTFD. For a detailed explanation of how to use FSP, refer to *RZ/T2, RZ/N2 Getting Started with Flexible Software Package (R01AN6434EJ****)*.

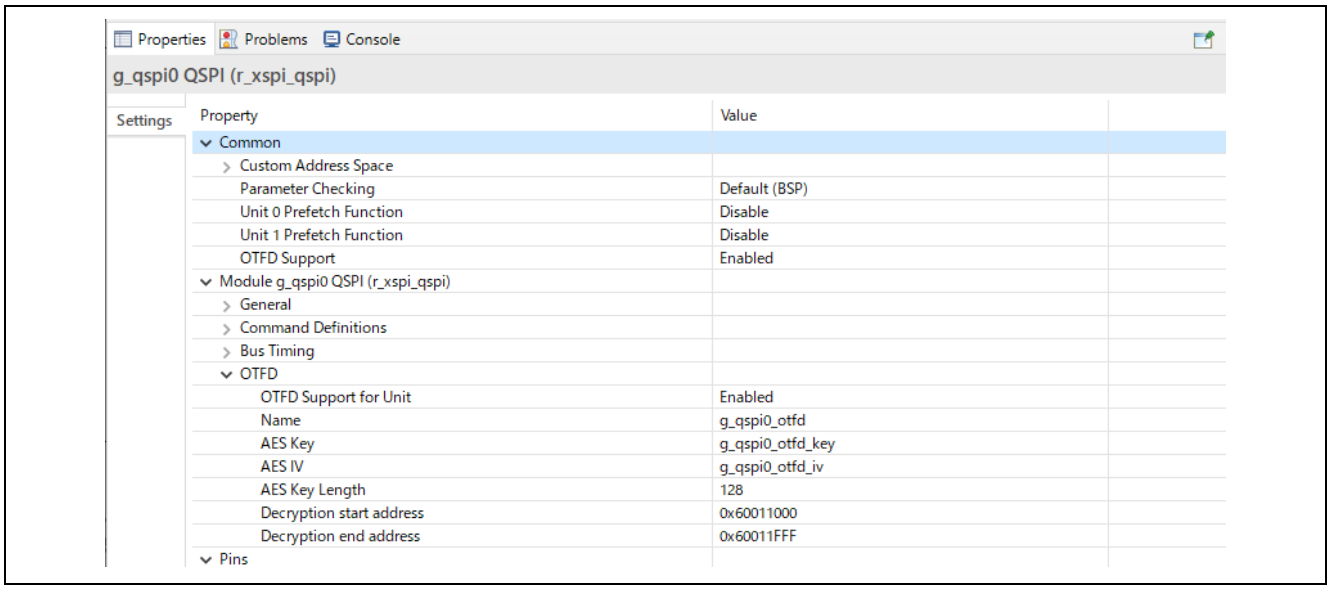


Figure 2.2 Configuration items of QSPI module for OTFD

Table 2.1 Configuration items of QSPI module for OTFD

Configuration items	Setting	Description
OTFD Support	Enable	Enable OTFD support for the xSPI module.
OTFD Support Unit	Enable	Enable OTFD support for the unit.
Name	Name must be a valid C symbol	OTFD Configuration name. Example: g_qspi0_otfd
AES Key	Name must be a valid C symbol	Name of Key variable. Example: g_qspi0_otfd_key
AES IV	Name must be a valid C symbol	Name of IV variable. Example: g_qspi0_otfd_iv
AES Key Length	128 or 256	Select AES key length.
Decryption start address	Value must be an integer between 0x60000000 and 0x6FFFFFFF	xSPI decryption start address. Do not select mirror area.
Decryption end address	Value must be an integer between 0x60000000 and 0x6FFFFFFF	xSPI decryption end address. Do not select mirror area.

2.1.2 r_rsip module

The RSIP driver is distributed separately from the FSP.

Please follow the instructions in Chapter 2 of the *RZ/T2ME Group Renesas Secure IP Driver (R01AN7375EJ****)* to integrate it into the FSP.

2.2 OTFD Encryption Method

To decrypt an application program in RZ/T2ME using OTFD, encrypt the application program in the following method.

1. Convert the target program to big-endian by AES 1 block (16 bytes)

2. Encrypt big-endian target program with AES-CTR.
   Encryption algorithm: AES-CTR
   Key Length: 128-bit or 256-bit
   Counter: Nonce[127:28] || Mirror area value for *Decryption start address*[31:4]

3. Convert the encrypted program to little-endian by AES 1 block (16 bytes).

The following tools can be used to encrypt application programs for OTFD:

- Secure Key Management Tool (SKMT)
- secureboot_utility.py included in RZ/T2ME Group Secure boot sample program package (R01AN7376EJ)

2.3 Demonstration Project Overview

2.3.1 OTFD Simple Operation Check

The OTFD simple operation check project (RZT2ME_RSK_OTFD_Debug) is designed for the purpose of debugging the OTFD function in the IDE, so that the program can be downloaded and executed without secure boot by the debugger.

When using only OTFD without secure boot, please refer to the OTFD Simple Operation Check project.

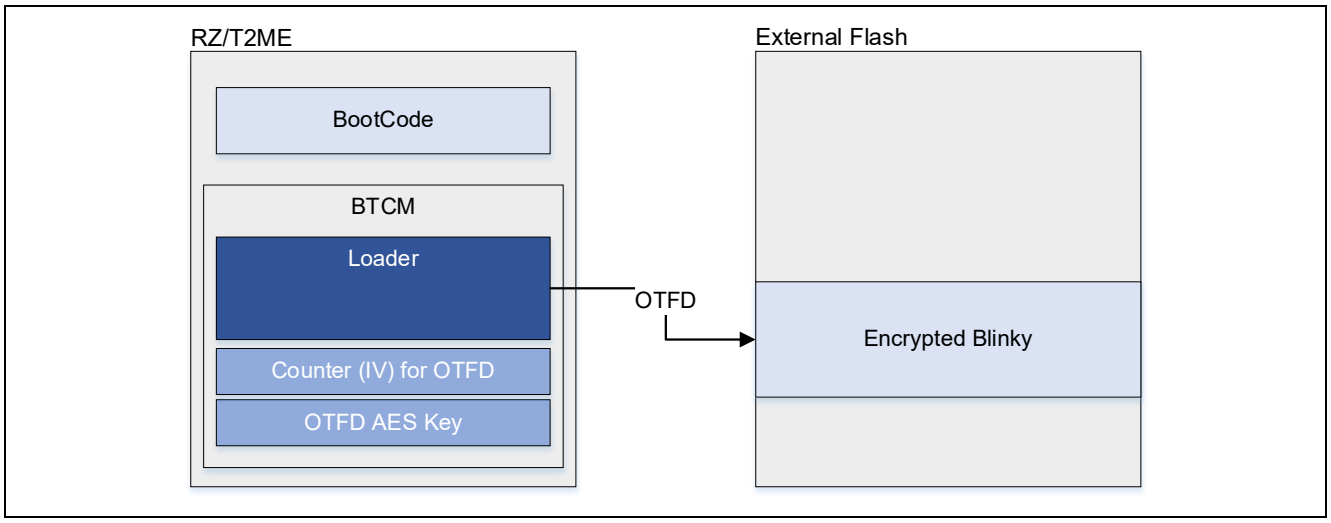


Figure 2.3 Software configuration of the project for OTFD simple operation check

The loader and application of the OTFD simple operation check project perform the following processes:

Loader: Injects the AES key for OTFD and configures OTFD, flash, etc., and jumps to the application on the flash encrypted for OTFD. The loader shall be placed in the BTCM.

Application: Blinking LED. The application is encrypted for OTFD and placed on the flash.

This sample project is intended as a simple operation check of OTFD, so the loader includes a key injection process. However, we recommend that key injection be performed during the manufacturing process of your product. Please be careful not to mix the key injection process into the source code of your product.

When debugging this program in the IDE, the loader and application shall be downloaded when the debugger is connected, depending on the project settings.

For the application, encryption for OTFD is performed by Secure Key Management Tool (SKMT) included in the project after build, and the encrypted program is downloaded.

The OTFD function in this program targets the following flash areas and uses 128-bit or 256-bit AES keys. (128-bit keys are used by default)

xSPI0 flash area: 0x60011000 - 0x60011FFF

2.3.2 OTFD with Secure Boot

The OTFD with Secure Boot project (RZT2ME_RSK_OTFD_xSPI_Boot) is a sample program intended to guide how to flash boot a program using the OTFD function with the secure boot function enabled.

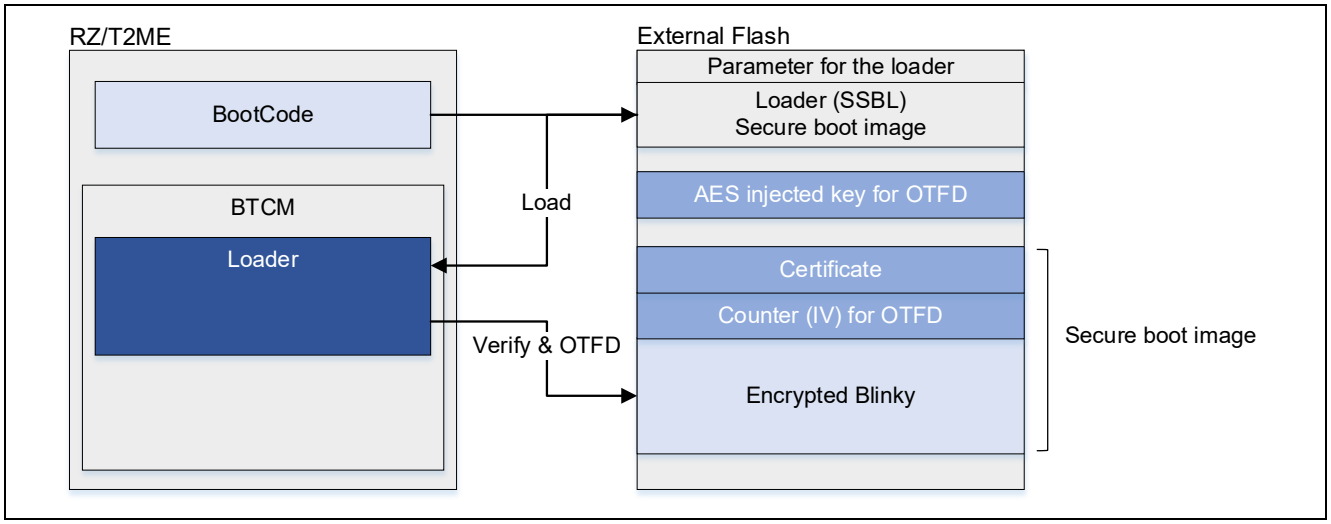


Figure 2.4 Software configuration of the project for OTFD with Secure Boot

The loader and application of the OTFD with Secure Boot project perform the following processes:

Loader: Injects the AES key for OTFD and configures OTFD, flash, etc., and jumps to the application on the flash encrypted for OTFD. If application verification is enabled, it verifies the application before jumping to the application. The loader section shall be placed in RAM.

Application: Blinking LED. The application is encrypted for OTFD and placed on the flash.

This sample program shows how to execute the program to be decrypted by the OTFD function after integrity verification and how to execute it as is.

OTFD can keep an application program confidential, but it cannot detect the program tampering. We recommend executing the program with OTFD after integrity verification of the program.

To build a secure boot system, this program should convert the generated S-Record format program files and key data into data for secure boot and OTFD after the project is built, and program them into flash.

The Secure Boot sample program package for RZ/T2ME is used for this setup work.

The OTFD function in this program targets the following flash areas and uses 128-bit or 256-bit AES keys. (128-bit keys are used by default)

xSPI0 flash area: 0x60011000 - 0x60011FFF

3. Quick Start

This chapter describes how to build, run, and check the operation of the two demo projects included in this package: the OTFD simple operation check demo and the OTFD with Secure Boot demo.

When using only OTFD without secure boot, please refer to the OTFD Simple Operation Check project.

3.1 OTFD Simple Operation Check

IAR Embedded Workbench for ARM (EWARM) or e^2 studio is used as the development environment of OTFD simple operation check demo.
When you unzip RZT2ME_RSK_OTFD_Debug_Rev100.zip, you will find folders named gcc and iccarm, which contain the projects for each compiler.

3.1.1 Board Settings

Table 3.1 show the environment required for configuring the RSK+. For details on how to install and use tools, refer to the document "Getting Started with Flexible Software Package".

Table 3.1 Setup Environment for RZ/T2ME RSK+

Name	Remarks
RZ/T2ME evaluation board	Renesas Starter Kit+ for RZ/T2ME
USB cables	1 (Mini-B, type-A), For USB to Serial
	1 (Type-C, type-A), For power supply
Windows host PC	EWARM or e^2 studio installed
Debugger	I-jet or J-LINK or USB cable (micro B, type A) USB cable is required when using J-Link™ OB on RSK+ as debugger

1. Set the switch as follows:

Table 3.2 Operation Mode Switch Settings for RSK+ RZT2ME

SW	Setting	Description
SW4.1	ON	16-bit bus boot mode (NOR Flash) External flash memory must be blank to start this sample program (RAM execution). NOR flash memory on this board is blank, so set the operating mode switch for 16-bit bus boot mode (NOR flash).
SW4.2	OFF	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait This sample program runs on TCM and CPU frequency is set to 800MHz in the start up process. Therefore, the following settings (MDW = 1) are required.

2. Connect the PC to the RZ/T2ME RSK+ USB-Serial connector (CN16) with USB cable (MiniB, type-A).
3. Connect the debugger to the PC and the RZ/T2ME RSK+. For details on connecting the debugger, refer to the document “Getting Started with Flexible Software Package”.

3.1.2 Build and Run the Demo

This section describes the steps to build and run the demo using RZT2ME_RSK_OTFD_Debug_Rev100.zip.

In this demonstration project, the command for SKMT to encrypt the OTFD target area is already registered in the post-build command of the project properties.

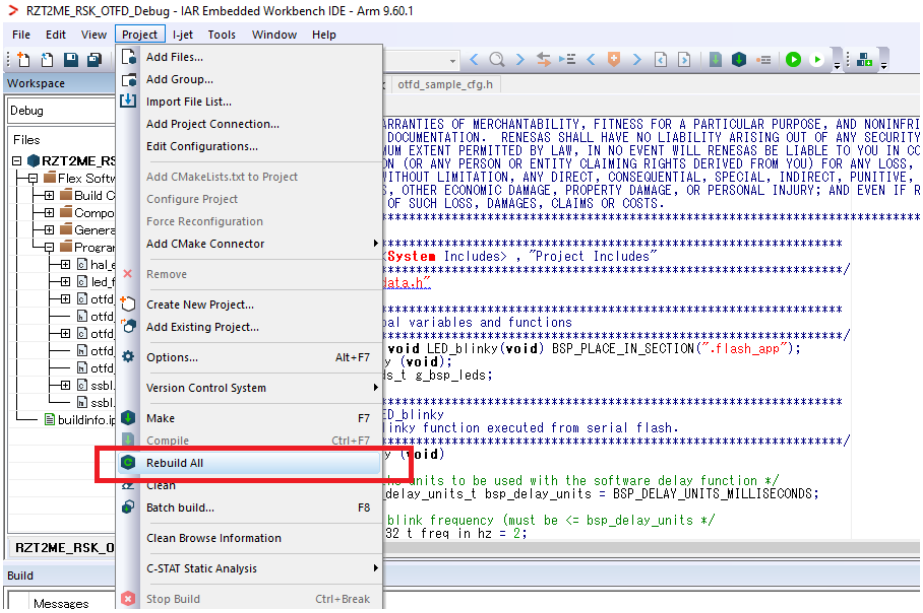
If you want to change a registered SKMT command, refer to section 4.4.3.

EWARM:

1. Open the OTFD Simple Operation Check demo project and build the project.
   When using EWARM, open the following workspace, select the OTFD Simple Operation Check demo project, then build the project.

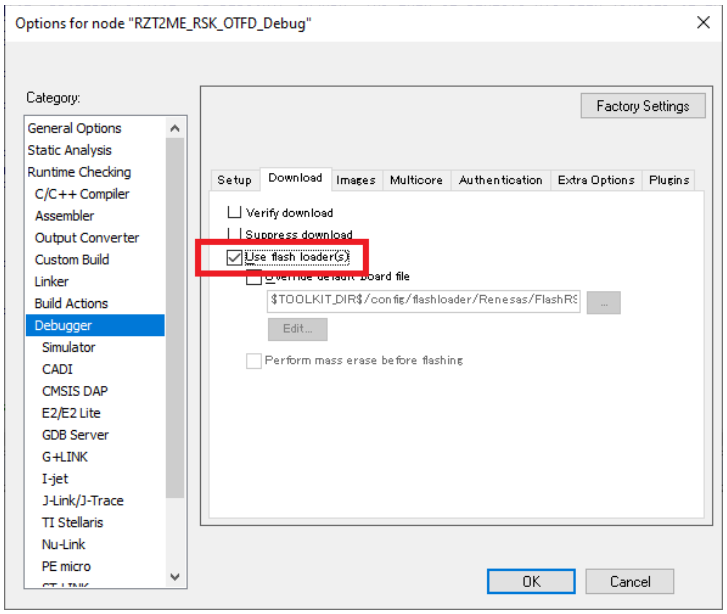
iccarm\RZT2ME_RSK_OTFD_Debug_Rev100.eww

   After selecting the project, click “Project > Rebuild All” on the toolbar to build the project.

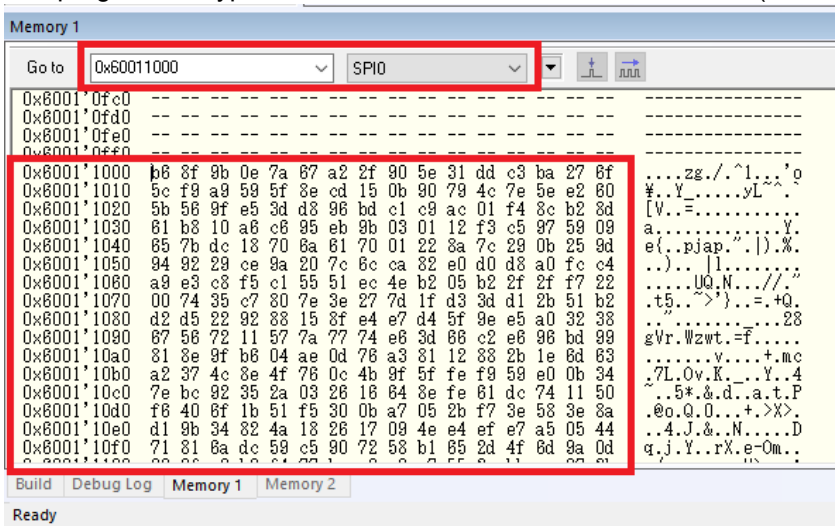


   An S-Record file (RZT2ME_RSK_OTFD_Debug.srec) of the sample program is generated, followed by a file (RZT2ME_RSK_OTFD_encrypt.srec) in which the blinking process part is encrypted with SKMT.

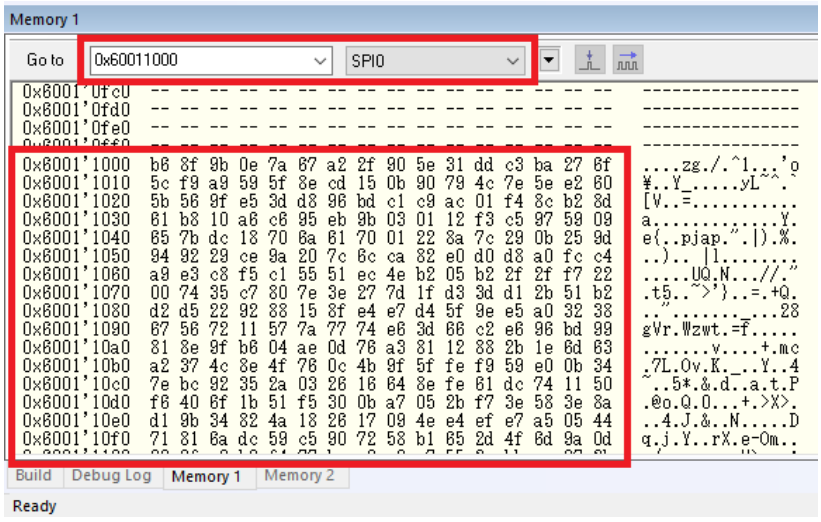
2. Enable the "Use Flash Loader" debugger option and download the S-Record files of the sample program. You will get some warnings that the data in RAM will not be flushed, but you can ignore it. After that, disable the "Use Flash Loader" option in the debugger options and download the S-Record files again, and only the data in RAM will be downloaded.



The program encrypted for OTFD can be seen in the flash area (0x60011000 -) of the memory window.



3. Run the sample program in the debugger.
   If the RZ/T2ME OTFD works correctly, the LED will light up 10 times. In EWARM 9.60.1, decryption of an encrypted program cannot be confirmed in the memory window, the program before decryption is displayed.

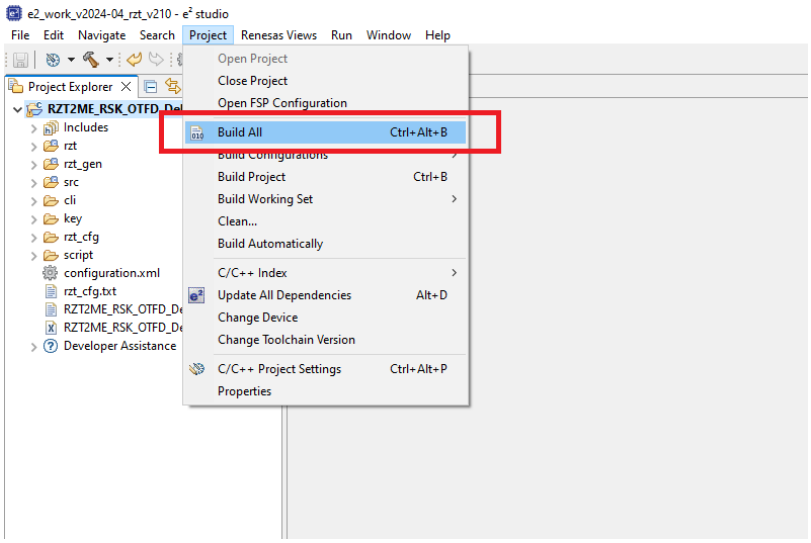


e² studio:

1. Open the OTFD Simple Operation Check demo project and build the project.
   When using e² studio, import the project (RZT2ME_RSK_OTFD_Debug_Rev100) below, then build the project.

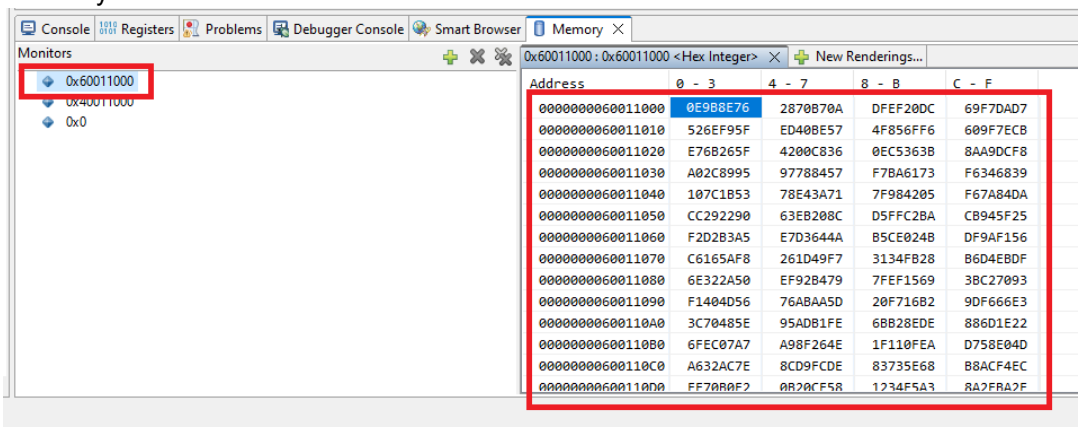
   gcc\RZT2ME_RSK_OTFD_Debug_Rev100

   After project import, click “Project > Build All” on the toolbar to build the project.

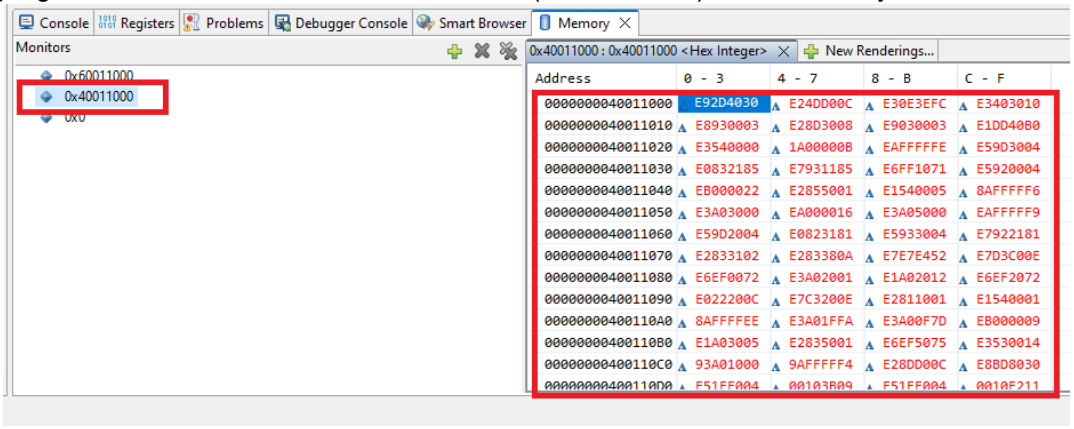


   An S-Record file (RZT2ME_RSK_OTFD_Debug.srec) of the sample program is generated, followed by a file (RZT2ME_RSK_OTFD_encrypt.srec) in which the blinking process part is encrypted with SKMT.

2. Then download and run the S-Record and encrypted S-Record files of the sample program in the debugger. The program encrypted for OTFD can be seen in the flash area (0x60011000 -) of the memory window.



3. If the RZ/T2ME OTFD works correctly, the LED will light up 10 times. The decryption of the encrypted program can be seen in the flash mirror area (0x40011000-) on the memory window.



3.2 OTFD with Secure Boot

IAR Embedded Workbench for ARM (EWARM) or e² studio is used as the development environment of the OTFD with Secure Boot demo.
When you unzip RZT2ME_RSK_OTFD_xSPI_Boot_Rev100.zip, you will find folders named gcc and iccarm, which contain the projects for each compiler.

This section describes the procedure for setting up and running the demo using the pre-build OTFD with Secure Boot demo in the RZ/T2ME RSK+.

3.2.1 Set Up the Demo

Table 3.3 shows the environment required for setup, and Table 3.4 shows the files to be prepared in advance for setup.

Table 3.3 Setup Environment for RZ/T2ME RSK+

Name	Remarks
RZ/T2ME evaluation board	Renesas Starter Kit+ for RZ/T2ME
USB cables	1 (Mini-B, type-A)
	1 (Type-C, type-A)
Windows host PC	EWARM or e² studio installed
parameter_generator.py Note1	Generation tool for the parameter for the loader and the parameter for the user application program
secureboot_utility.py Note1	Tools to generate secure boot images, key files, password files, OTFD encrypted image, and signed data required to set up RZ/T2ME products
secure_device_setup.py Note1	Command sending tool for secure device setup
RZT2ME_RSK_SecureDeviceSetup_*.out.srec Note1	Pre-built secure device setup program for RZ/T2ME

Note1 Included in RZ/T2ME Group Secure Boot sample program package.

Table 3.4 Files to be prepared in advance for setup

Name	Remarks
RZT2ME_RSK_OTFD_xSPI_Boot.srec	S-Record file of OTFD sample program with secure boot function, including loader and application parts.
otfd_encrypt_aes_key.bin	AES key for OTFD 128 / 256bit binary data
otfd_encrypt_nonce.bin	NONCE for OTFD (Upper 100 bits used) 128bit binary data
secureboot_common_key.bin	Image encryption key for secure boot 128bit binary data
rootkey-pair.pem	root key pair for secure boot
ikey-pair_ssbl.pem	Image key pair for secure boot (for loader)
ikey-pair_userapp.pem	Image key pair for secure boot (for application)
secureboot_common_keyfile.bin	A key file for injecting the key for Secure Boot. This was created using secureboot_utility.py and specifying that it should be injected into Secure Boot common key area 0 in the OTP area.
otfd_encrypt_aes_keyfile.bin	A key file for injecting the AES key used by the OTFD function. This was created using secureboot_utility.py and specifying that it should be injected into 0x6001000- in the flash area.

Set up the demo according to Chapter 2 of the *RZ/T2ME Group Device Setup Guide for Secure boot (R01AN7376EJ)*.

After setting up the Host PC and board according to section 2.1 of the *RZ/T2ME Group Device Setup Guide for Secure boot (R01AN7376EJ)*, follow the steps below to setup the pre-built demo into the device.

For details on, refer to section 2.7 in *RZ/T2ME Group Device Setup Guide for Secure boot*.

1. Use secureboot_utility.py to extract the loader part in binary from the OTFD sample program with secure boot function.

```
> python secureboot_utility.py mot2bin --startaddr 00102800 --endaddr
0010FFFF -i RZT2ME_RSK_OTFD_xSPI_Boot.srec -o RZT2ME_RSK_OTFD_loader.bin
```

2. Use secureboot_utility.py to extract the application part in binary from the OTFD sample program with secure boot function and encrypt it for OTFD.
   Note that --destaddr must be the value of the mirror area of the starting address of the OTFD target area specified in --startaddr. This value is used as the value of the least significant 28 bits of the counter when encrypting OTFD with AES-CTR.

```
> python secureboot_utility.py encotfd --enckey otfd_encrypt_aes_key.bin -
-nonce otfd_encrypt_nonce.bin --startaddr 60011000 --endaddr 60011FFF -i
RZT2ME_RSK_OTFD_xSPI_Boot.srec -o RZT2ME_RSK_OTFD_app_encrypt.bin --
destaddr 40011000
Output File: D:\_work\RZT2ME_RSK_OTFD_app_encrypt.bin
Key: 4BF34EDDE418A23B4EA7F49744BD5073
Counter: BF89F643B5849AA0F19502CBC4001100
```

3. Use parameter_generator.py and secureboot_utility.py to generate a secure boot image for the loader.
   The following commands will generate parameter_RZT2ME_RSK_OTFD_loader.bin, sec_boot_image_loader.bin and sec_boot_image_loader.pubkey.

```
> python parameter_generator.py loader --mpu rzt2me --src_addr 60000050 --
dest_addr 00102000 --mode xspi0 -i RZT2ME_RSK_OTFD_loader.bin -o
parameter_RZT2ME_RSK_OTFD_loader.bin --secureboot
parameter_generator.py : notice: For secure boot, the start address of the
loader program must be the specified address plus 0x800 : 00102000

> python secureboot_utility.py image -i RZT2ME_RSK_OTFD_loader.bin -o
sec_boot_image_loader.bin --param parameter_RZT2ME_RSK_OTFD_loader.bin --
rkey rootkey-pair.pem --ikey ikey-pair_ssbl.pem --enckey
secureboot_common_key.bin --sign_target cert-img --keyselect ckey0
```

4. Use parameter_generator.py and secureboot_utility.py to generate a secure boot image for the application.
   The following commands will generate parameter_RZT2ME_RSK_OTFD_app_encrypt.bin, sec_boot_image_app.bin and sec_boot_image_app.pubkey.
   Note that the application is already encrypted for OTFD and not for secure boot images.

```
> python parameter_generator.py userapp --src_addr 60010FF0 --
app_start_addr 60010FF0 -i RZT2ME_RSK_OTFD_app_encrypt.bin -o
parameter_RZT2ME_RSK_OTFD_app_encrypt.bin --secureboot

> python secureboot_utility.py image -i RZT2ME_RSK_OTFD_app_encrypt.bin -o
sec_boot_image_app.bin --param parameter_RZT2ME_RSK_OTFD_app_encrypt.bin -
-rkey rootkey-pair.pem --ikey ikey-pair_userapp.pem --sign_target cert-img
```

5. Boot the RZ/T2ME in SCI boot mode or USB boot mode and load the secure device setup program (RZT2ME_RSK_SecureDeviceSetup_*.out.srec) into the RAM of the device.
   If the program is successfully loaded, the secure device setup program will start.

   In SCI boot mode:

   a-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/T2ME:

SW	Setting	Description
SW4.1	OFF	SCI (UART) boot mode.
SW4.2	ON	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

   a-2) Load the secure device setup program (RZT2ME_RSK_SecureDeviceSetup_*.out.srec) into the RZ/T2ME using the secure device setup tool (secure_device_setup.py).
   The following command loads RZT2ME_RSK_SecureDeviceSetup_SCI.out.srec (for RZ/T2ME):

```
> python secure_device_setup.py start --port COM9 --boot_mode sci -i
RZT2ME_RSK_SecureDeviceSetup_SCI.out.srec

SCI Download mode.
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

   In USB boot mode:

   b-1) Set SW4 on the RSK+ board to the following and press S3 RESET.
   RZ/T2ME:

SW	Setting	Description
SW4.1	ON	USB boot mode.
SW4.2	OFF	
SW4.3	OFF	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

   b-2) Load the secure device setup program (RZT2ME_RSK_SecureDeviceSetup_*.out.srec) into the RZ/T2ME using the secure device setup tool (secure_device_setup.py).
   The following command loads RZT2ME_RSK_SecureDeviceSetup_USB.out.srec (for RZ/T2ME):

```
> python secure_device_setup.py start --port COM9 --boot_mode usb -i
RZT2ME_RSK_SecureDeviceSetup_USB.out.srec

USB Open.
USB Download mode (Normal USB boot)
Send program data. (S0)
-- Load Program to BTCM  ---------------
Send program data. (S3)
-- Start Boot Program on BTCM ---------
```

6. Use secure_device_setup.py to program and inject the data and keys created in steps 1-4 into the flash and enable secure boot.
   Program the parameters for the loader, user program and keys into the external flash on the RSK+ using the secure device setup tool (secure_device_setup.py).
   If you have started the secure device setup program in USB boot mode, the COM port of the RZ/T2ME may change from the boot time; to check the COM port again.

```
> python secure_device_setup.py writeflash --port COM5 --addr 60000000 -
i parameter_RZT2ME_RSK_OTFD_loader.bin
writeflash : Setup success.
> python secure_device_setup.py writeflash --port COM5 --addr 60000050 -
i sec_boot_image_loader.bin
writeflash : Setup success.
> python secure_device_setup.py writeflash --port COM5 --addr 60010DF0 -
i sec_boot_image_app.bin
writeflash : Setup success.

> python secure_device_setup.py instkey --port COM5 -i
secureboot_common_keyfile.bin
instkey : Setup success.
> python secure_device_setup.py instkey --port COM5 -i
otfd_encrypt_aes_keyfile.bin
instkey : Setup success.
> python secure_device_setup.py setboot --port COM5 --hash
sec_boot_image_loader.pubkey --enable
setboot : Setup success.
```

3.2.2 Run the Demo

RZ/T2ME is booted in xSPI0 boot mode, the device's boot function refers to the parameters for the loader and the user program written to the QSPI Flash is extracted to RAM and booted securely.

If secure boot fails for any reason, the user program will not boot. If the program does not start after running the device setup for secure boot according to section 3.2.1, check the section 2.2.10 of the RZ/T2ME Group Device Setup Guide for Secure boot (R01AN7376EJ).

Note that if SCI/USB boot is not disabled, the flash memory can be reprogrammed by SCI/USB boot even if secure boot fails.

Set SW4 on the RSK+ board to the following and press S3 RESET.

RZ/T2ME:

SW	Setting	Description
SW4.1	ON	xSPI0 boot mode (x1 boot Serial flash)
SW4.2	ON	
SW4.3	ON	
SW4.4	ON	JTAG Authentication by Hash is disabled.
SW4.5	OFF	ATCM 1 wait

If the RZ/T2ME OTFD works correctly, the LED will light up 10 times.

4. Sample Program

4.1 Development Environment

Refer to RZ/T2, RZ/N2 Getting Started with Flexible Software Package.

4.2 File Structure

The zip file in the package contains the sample program code for the GCC compiler and the IAR compiler. When you unzip the zip file, you will find folders named gcc and iccarm, which contain the projects for each compiler.

Table 4.1 list the main files contained in the OTFD sample program project for each compiler.

Table 4.1 File Structure of SSBL

Folder Name	File Name	Description
RZT2ME_RSK_OTFD_ *\		
├	*.jlink, *.launch, *project, *.eww, *.ewd, *.ewp	Project files
├	*.pincfg, *.xml, *.ipcf	Flexible Software Package Files
├	rz*_cfg.txt	
├ rz*\		
├ rz*_cfg\		
├ rz*_get\		
├ script\	*.ld, *.icf	Memory allocation
├ cli\		SKMT V1.06 Note1
├ key\		Sample key data Note1
└ src\	*.c, *.h	OTFD sample program source code

Note1 Only included in RZT2ME_RSK_OTFD_Debug_Rev*.zip.

4.3 Functions

Table 4.2 lists the main functions defined in the OTFD sample program.

Table 4.2 Functions of OTFD Sample Program

File Name	Function Name	Description
led_func.c	LED_blinky	Application for OTFD
ssbl.c	second_application_boot_loader	Main routine of loader program
	check_secure_boot_image	Verify the OTFD program
	qspi_set_Quad_Mode_enable	QSPI Enable/Disable
	qspi_set_Quad_Mode_disable	

4.4 OTFD Simple Operation Check

4.4.1 Memory Maps

Table 4.3 show memory maps for the sample program.

Table 4.3 Memory Map for RZ/T2ME

Memory Type	Address	Size	Description
ATCM	0x00000000 - 0x00002FFF	512 Kbyte	OTFD Sample Programs heap/stack
	0x00003000 - 0x0007FFFF		Not used
BTCM	0x00100000 - 0x00101FFF	64 Kbyte	Reserved (bootloader area)
	0x00102000 - 0x0010FFFF		OTFD sample program loader part
External Memory (xSPI0 Flash)	0x60000000 - 0x6FFFFFFF	256 MByte	Not used
	0x60011000 - 0x60011FFF		OTFD sample program Application part (OTFD target area)
	0x60000000 - 0x6FFFFFFF		Not used

4.4.2 Configure

In this sample project, you can configure the key length of the OTFD AES key. In the default configuration, the OTFD key length is 128-bit. The configurations can be changed with the following files.

RZT2ME_RSK_OTFD_Debug_Rev*\src\otfd_sample_cfg.h

The configuration of the OTFD key length is shown in Table 4.4.

Table 4.4 Configurations for OTFD Key length

Configuration items	Configurable values	Description
CFG_OTFD_SAMPLE_USE_KEY	USE_KEY128	Default setting. The OTFD function uses a 128-bit key length.
	USE_KEY256	OTFD function uses 256-bit key length.

4.4.3 How to change the OTFD key

The procedure for changing the key in this sample project is shown below.

1. Set the key length of OTFD in the following file.
   RZT2ME_RSK_OTFD_Debug_Rev*\src\otfd_sample_cfg.h
2. Select the QSPI module in the Stack tab of the FSP configuration of the project, and set the AES Key Length to match the key length in step 1.

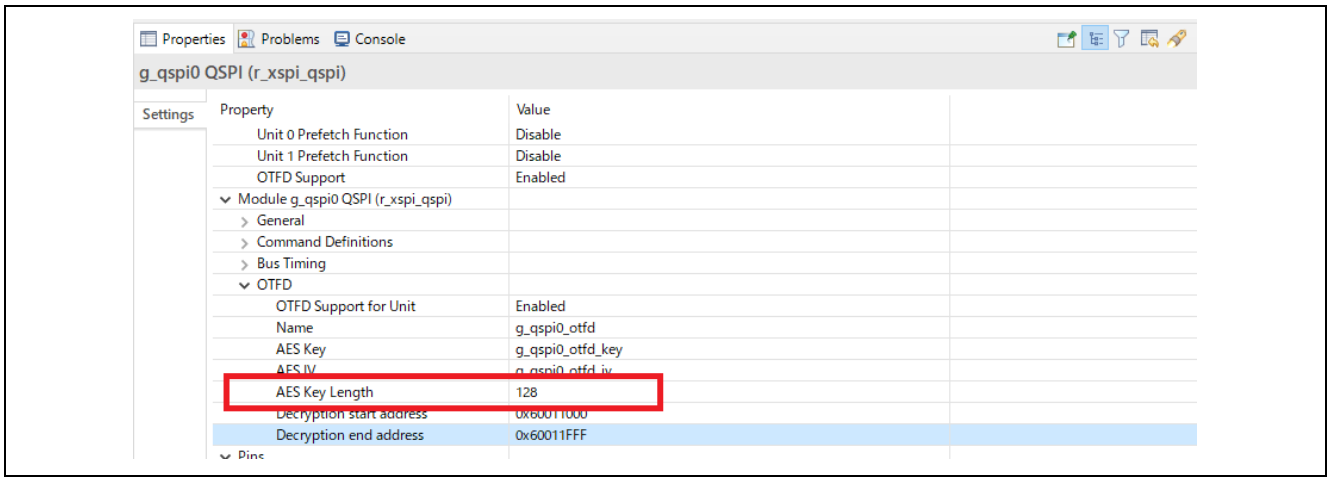
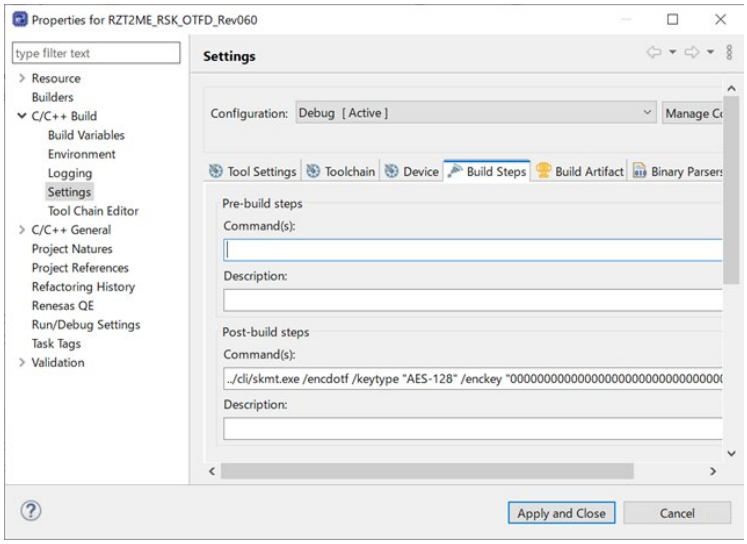


Figure 4.1 QSPI module settings on the Stack tab of the FSP configuration

3. Store key files under the src folder of the project.
   To use a key included in the sample project, copy the key length file to be used from the key folder.
   key\AES-*\otfd_key.c
   key\AES-*\otfd_key.h

4. Set commands for program encryption with SKMT
   Open the project properties and change the commands for SKMT registered in the post-build execution commands to match the key to be used. For details on SKMT, refer to the SKMT specification.
   In addition, if you wish to use the keys included in the sample project, simply set the commands described below.

   e² studio:
   Set commands for Post-build steps.



   Setting up e² studio to use AES 128-bit sample key:

```
../cli/skmt.exe /encdotf /keytype "AES-128" /enckey
"4BF34EDDE418A23B4EA7F49744BD5073" /nonce
"BF89F643B5849AA0F19502CBC0000000" /startaddr "60011000" /endaddr
"60011FFF" /prg "RZT2ME_RSK_OTFD_Debug.srec" /output
"RZT2ME_RSK_OTFD_encrypt.srec" /destaddr "40011000"
```

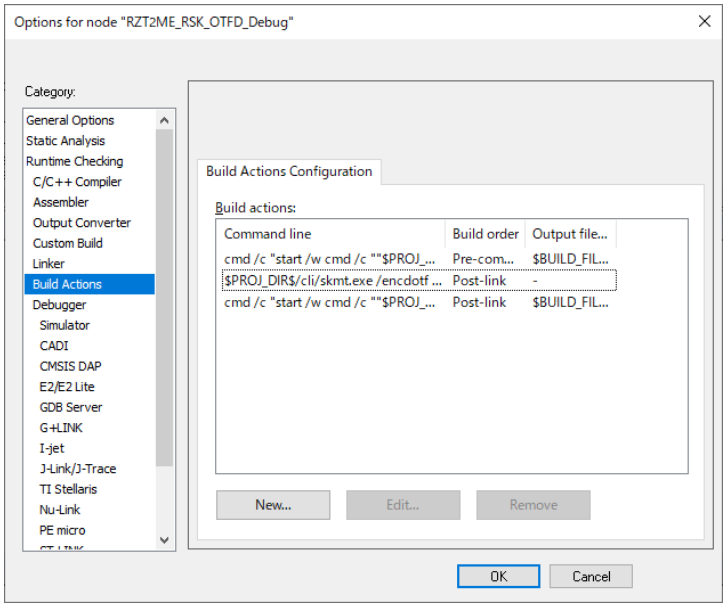
   Setting up e² studio to use AES 256-bit sample key:

```
../cli/skmt.exe /encdotf /keytype "AES-256" /enckey
"FB3577D4E8F7251869DEEC618E04C1D1767E6A17975CEB93FC976E92CCC355F0" /nonce
"BF89F643B5849AA0F19502CBC0000000" /startaddr "60011000" /endaddr
"60011FFF" /prg "RZT2ME_RSK_OTFD_Debug.srec" /output
"RZT2ME_RSK_OTFD_encrypt.srec" /destaddr "40011000"
```

   Note that /destaddr must be the value of the mirror area of the starting address of the OTFD target area specified in /startaddr. This value is used as the value of the least significant 28 bits of the counter when encrypting OTFD with AES-CTR.

EWARM:

Set commands for Build Actions.



Setting up EWARM to use AES 128-bit sample key:

```
$PROJ_DIR$/cli/skmt.exe /encdotf /keytype "AES-128" /enckey
"4BF34EDDE418A23B4EA7F49744BD5073" /nonce
"BF89F643B5849AA0F19502CBC0000000" /startaddr "60011000" /endaddr
"60011FFF" /prg "$PROJ_DIR$/Debug/Exe/RZT2ME_RSK_OTFD_Debug.srec" /output
"$PROJ_DIR$/Debug/Exe/RZT2ME_RSK_OTFD_encrypt.srec" /destaddr "40011000"
```

Setting up EWARM to use AES 256-bit sample key:

```
$PROJ_DIR$/cli/skmt.exe /encdotf /keytype "AES-256" /enckey
"FB3577D4E8F7251869DEEC618E04C1D1767E6A17975CEB93FC976E92CCC355F0" /nonce
"BF89F643B5849AA0F19502CBC0000000" /startaddr "60011000" /endaddr
"60011FFF" /prg "$PROJ_DIR$/Debug/Exe/RZT2ME_RSK_OTFD_Debug.srec" /output
"$PROJ_DIR$/Debug/Exe/RZT2ME_RSK_OTFD_encrypt.srec" /destaddr "40011000"
```

Note that /destaddr must be the value of the mirror area of the starting address of the OTFD target area specified in /startaddr. This value is used as the value of the least significant 28 bits of the counter when encrypting OTFD with AES-CTR.

When creating a new key, SKMT can be used to create a new C file format key instead of step 3. For more information on SKMT, please refer to the SKMT manual.

An example of a command to create an AES key file using SKMT is shown below. Please refer to the SKMT manual to create ufpk.key and wufpk_rzt2me.key.

AES 128-bit key length:

```
skmt.exe /genkey /ufpk file="..\key\ufpk.key" /wufpk
file="..\key\wufpk_rzt2me.key" /mcu "RZ-TSIP" /keytype "AES-128" /key
"4BF34EDDE418A23B4EA7F49744BD5073" /filetype "csource" /output
"..\key\otfd_key.c"
```

AES 256-bit key length:

```
skmt.exe /genkey /ufpk file="..\key\ufpk.key" /wufpk
file="..\key\wufpk_rzt2me.key" /mcu "RZ-TSIP" /keytype "AES-256" /key
"FB3577D4E8F7251869DEEC618E04C1D1767E6A17975CEB93FC976E92CCC355F0"
/filetype "csource" /output "..\key\otfd_key.c"
```

4.5 OTFD with Secure Boot

4.5.1 Memory Maps

Table 4.5 show memory maps for the sample program.

Table 4.5 Memory Map for RZ/T2ME

Memory Type	Address	Size	Description
ATCM	0x00000000 - 0x00002FFF	512 Kbyte	OTFD Sample Programs heap/stack
	0x00003000 - 0x0007FFFF		Not used
BTCM	0x00100000 - 0x00101FFF	64 Kbyte	Reserved (bootloader area)
	0x00102000 - 0x0010FFFF		OTFD sample program loader part
External Memory (xSPI0 Flash)	0x60000000 - 0x6000E04F	256 MByte	Reserved (bootloader area)
	0x6000E050 - 0x6000FFFF		Not used
	0x60010000 - 0x600100FF		Key data area for OTFD
	0x60010100 - 0x60010DEF		Not used
	0x60010DF0 - 0x60010FEF		Certificate area
	0x60010FF0 - 0x60010FFF		Counter value area for OTFD
	0x60011000 - 0x60011FFF		OTFD sample program Application part (OTFD target area)
	0x60000000 - 0x6FFFFFFF		Not used

4.5.2 Configure

In this sample project, you can configure whether or not the OTFD application is verified. In the default configuration, the OTFD application is verified. The configurations can be changed with the following files.

RZT2ME_RSK_OTFD_xSPI_Boot_Rev*\src\otfd_sample_cfg.h

The configurations are shown in Table 4.6.

Table 4.6 Configurations for OTFD Key length

Configuration items	Configurable values	Description
CFG_OTFD_VERIFY_CERT_FOR_FLASH_APP	VERIFY_CERT_ENABLE	Default setting. Enable application integrity verification. Run OTFD applications on flash with integrity verification.
	VERIFY_CERT_DISABLE	Disables application integrity verification. Runs OTFD applications on flash without integrity verification.

4.5.3 How to change the OTFD key

The procedure for changing the key in this sample project is shown below.

1. Select the QSPI module in the Stack tab of the FSP configuration of the project, and set the AES Key Length.

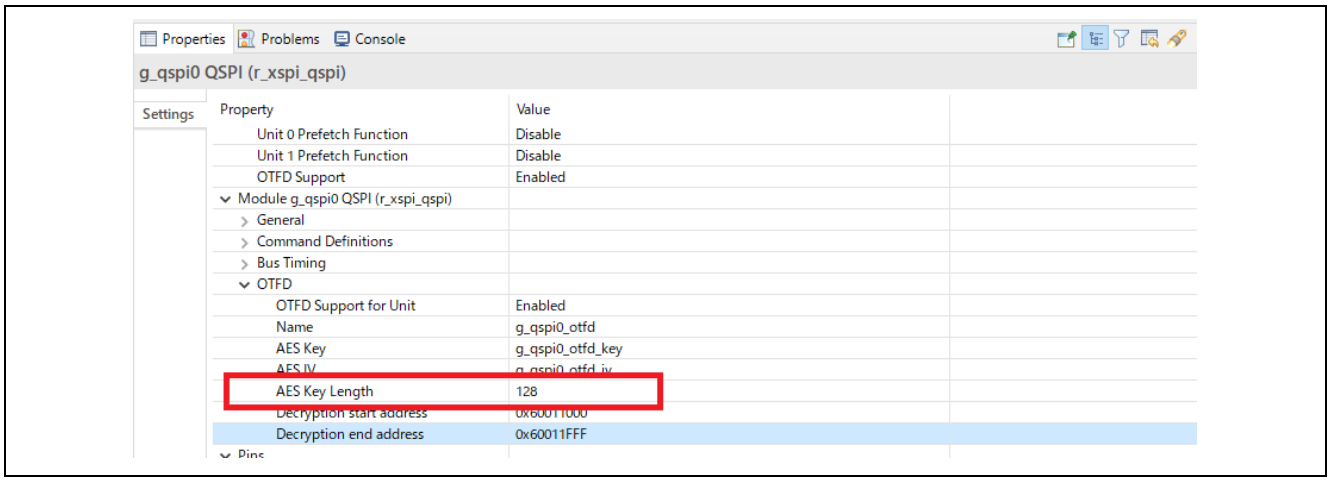


Figure 4.2 QSPI module settings on the Stack tab of the FSP configuration

2. Generate a key file with a key length of your select.
   Using OpenSSL's random number generation, etc., generate a key of your selected key length as a binary file.

For details on how to inject a new key into the flash, refer to section 2.4 of the *RZ/T2ME Group Device Setup Guide for Secure boot*.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug 21, 2024	-	First edition issued

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

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

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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

Contact information

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