RX Family

Unique ID Read Module
Using Firmware Integration Technology

Abstract

This document describes the unique ID read (UID) module using firmware integration technology (FIT). The module reads 32 bytes of the unique ID stored in the extra area and stores it in the specified area. Hereinafter this module is referred to as UID FIT module.

Products

- RX110 Group
- RX111 Group
- RX113 Group
- RX130 Group
- RX13T Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details of the confirmed operation contents of each compiler, refer to “4.1 Confirmed Operation Environment".

Related Documents

- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)

Contents

1. Overview 3
1.1 UID FIT Module 3
1.2 Overview of the UID FIT Module 3
1.2.1 Procedure to Read the Unique ID 3
1.2.2 Sections for Program Allocation 4
1.3 Outline of the APIs 4
1.4 Processing Example 5
1.5 Limitations 6

2. API Information 7
2.1 Hardware Requirements 7
2.2 Software Requirements 7
2.3 Supported Toolchain 7
2.4 Header Files 7
2.5 Integer Types 7
2.6 Configuration Overview 7
2.7 Code Size 8
2.8 Return Values 9
2.9 Adding the FIT Module to Your Project 10
2.10 “for”, “while” and “do while” statements 11
2.11 Setting a Project 12
2.11.1 Renesas Electronics C/C++ Compiler Package for RX Family 12
2.11.2 GCC for Renesas RX 14
2.11.3 IAR C/C++ Compiler for Renesas RX 16
2.12 Note on Using API Functions 20
2.13 Note During On-Chip Debugging 20

3. API Functions 21
3.1 R_UID_Open () 21
3.2 R_UID_Read () 22
3.3 R_UID_GetVersion () 23

4. Appendices 24
4.1 Confirmed Operation Environment 24
4.2 Troubleshooting 25

5. Reference Documents 26

Related Technical Updates 26

Revision History 27

1. Overview

1.1 UID FIT Module

The UID FIT module can be used by being implemented in a project as an API. See section 2.9 Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the UID FIT Module

The UID FIT module provides API functions to read 32 bytes of the unique ID stored in the extra area and store the read unique ID in the specified area.

This module uses the self-programming function of the flash memory to read the unique ID from the extra area. The sequencer enters ROM P/E mode to read the unique ID. Values in the ROM cannot be read in ROM P/E mode. Thus the program is transferred to the RAM and executed in the RAM. After the unique ID is read, the program is executed in the ROM again.

1.2.1 Procedure to Read the Unique ID

The procedure to read the unique ID is described here.

1. Call the R_UID_Open() function.
   1-1. Transfer the program from the ROM to the RAM.

2. Call the R_UID_Read() function.
   2-1. Jump to the RAM.
   2-2. Enter the P/E mode (the ROM cannot be read in this mode).
   2-3. Read the unique ID by the unique ID read command.
   2-4. Enter read mode (the ROM can be read in this mode).
   2-5. Jump to the ROM.

1.2.2 Sections for Program Allocation

In this module, sections are provided to allocate the program executed in the RAM. The program to be transferred to RAM is set to be placed in the PFRAM section.

Table 1.1 lists the Sections for Program Allocation.

Table 1.1 Sections for Program Allocation

Source File	Function	Allocated Section
r_uid_rx.c	R_UID_Open R_UID_Read R_UID_GetVersion uid_codecopy	P section
r_uid_ram.c	uid_read_ram uid_pe_mode_enter uid_pe_mode_exit uid_read uid_write_fpmcr uid_enable_dataflashaccess uid_delay uid_delay_us	PFRAM section

1.3 Outline of the APIs

Table 1.2 lists the API functions included in the module.

Table 1.2 API Functions

Function	Description
R_UID_Open()	Transfers the program to read the unique ID from the ROM to the RAM for using the module.
R_UID_Read()	Reads the unique ID from the extra area.
R_UID_GetVersion()	Returns the version of the module.

1.4 Processing Example

Figure 1.1 shows a Processing Example of the UID FIT Module.

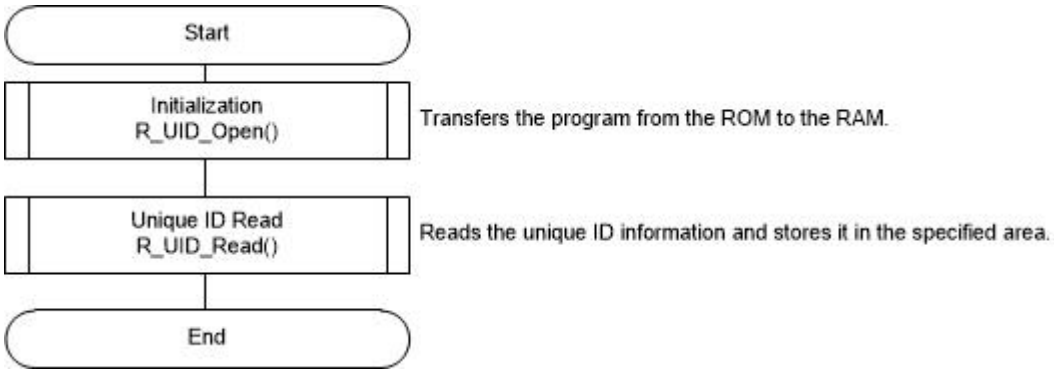


Figure 1.1 Processing Example of the UID FIT Module

1.5 Limitations

This module has the following limitation:

- Areas in the ROM or E2 DataFlash cannot be specified as the area to store the read unique ID.

2. API Information

This FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

The MCU used must support the following functions:

- Unique ID Read
- The FlashIF clock (FCLK) is 1 MHz or higher.
- When the FCLK is lower than 4 MHz, the frequency that can be set for the FCLK is 1 MHz, 2 MHz, or 3 MHz.
  Note that when 1 MHz < frequency ≤ 4 MHz, a frequency with a decimal point cannot be used.

2.2 Software Requirements

This driver is dependent upon the following FIT module:

- Renesas Board Support Package (r_bsp) v5.00 or higher

2.3 Supported Toolchain

This driver has been confirmed to work with the toolchain listed in 4.1 Confirmed Operation Environment.

2.4 Header Files

All API calls and their supporting interface definitions are located in r_uid_rx_if.h.

2.5 Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.6 Configuration Overview

This module does not have settings of the configuration options. Therefore the r_uid_rx_config.h (configuration header file) is not included.

2.7 Code Size

The sizes of ROM, RAM, and maximum stack usage associated with this module are listed below.

The values in the table below are confirmed under the following conditions.

Module Revision: r_uid_rx rev1.11

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00

(The option of "-lang = c99" is added to the default settings of the integrated development environment.)

GCC for Renesas RX 4.8.4.201801

(The option of "-std = gnu99" is added to the default settings of the integrated development environment.)

IAR C/C++ Compiler for Renesas RX version 4.10.1

(The default settings of the integrated development environment)

Configuration Options: Default settings

ROM, RAM and Stack Code Sizes

Device	Category	Memory Used		
		Renesas Compiler	GCC	IAR Compiler
RX110	ROM	**5613 bytes**	**8152 bytes**	**4065 bytes**
	RAM	**3294 bytes**	**3484 bytes**	**1982 bytes**
	STACK *1	**196 bytes**	**-**	**136 bytes**
RX111	ROM	**5846 bytes**	**8457 bytes**	**4417 bytes**
	RAM	**3354 bytes**	**3576 bytes**	**2043 bytes**
	STACK *1	**196 bytes**	**-**	**136 bytes**
RX113	ROM	**5881 bytes**	**8529 bytes**	**4469 bytes**
	RAM	**3438 bytes**	**3660 bytes**	**2127 bytes**
	STACK *1	**196 bytes**	**-**	**136 bytes**
RX130	ROM	**5852 bytes**	**8424 bytes**	**4465 bytes**
	RAM	**3414 bytes**	**3636 bytes**	**2103 bytes**
	STACK *1	**196 bytes**	**-**	**136 bytes**

Note 1. The sizes of maximum usage stack of interrupts functions is included.

Note 2. The size includes BSP.

2.8 Return Values

This section describes return values of API functions. This enumeration is located in r_uid_rx_if.h as are the prototype declarations of API functions.

```
typedef enum
{
       UID_SUCCESS = 0,     /* R_UID_Open executed successfully. */
                            /* Unique ID read successfully with R_UID_Read. */
       UID_ERR_UNINITIALIZED, /* R_UID_Read executed before executing R_UID_Open. */
       UID_ERR_LOCK_FUNC,   /* R_UID_Open or R_UID_Read executed while either
                               R_UID_Open or R_UID_Read is executed.  */
       UID_ERR_FAILURE,     /* R_UID_Open executed twice or more. */
                            /* Failed to read the unique id with R_UID_Read */
} uid_err_t;
```

2.9 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) or (5) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using the Smart Configurator in e^2 studio
By using the Smart Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: e^2 studio (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e^2 studio
By using the FIT Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: CS+ (R20AN0470)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

(5) Adding the FIT module to your project using the Smart Configurator in IAREW
By using the Smart Configurator Standalone version, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: IAREW (R20AN0535)” for details.

2.10 “for”, “while” and “do while” statements

In this module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for register to be reflected and so on. For these loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if user incorporates fail-safe processing into loop processing, user can search the corresponding processing with “WAIT_LOOP”.

The following shows example of description.

while statement example :

```
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}
```

for statement example :

```
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}
```

do while statement example :

```
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

2.11 Setting a Project

The sequencer enters ROM P/E mode to read the unique ID. The program cannot be executed in the ROM during ROM P/E mode. Therefore, to read the unique ID, a part of the program needs to be transferred to and executed in the RAM.

The settings differ depending on the compiler so configure settings according to the complier used.

2.11.1 Renesas Electronics C/C++ Compiler Package for RX Family

The following settings are required in the e^2 studio linker setting.

- Add the section.
- Specify the section for ROM to RAM mapping.

Adding the Section

Add the RPFRAM section to the RAM.

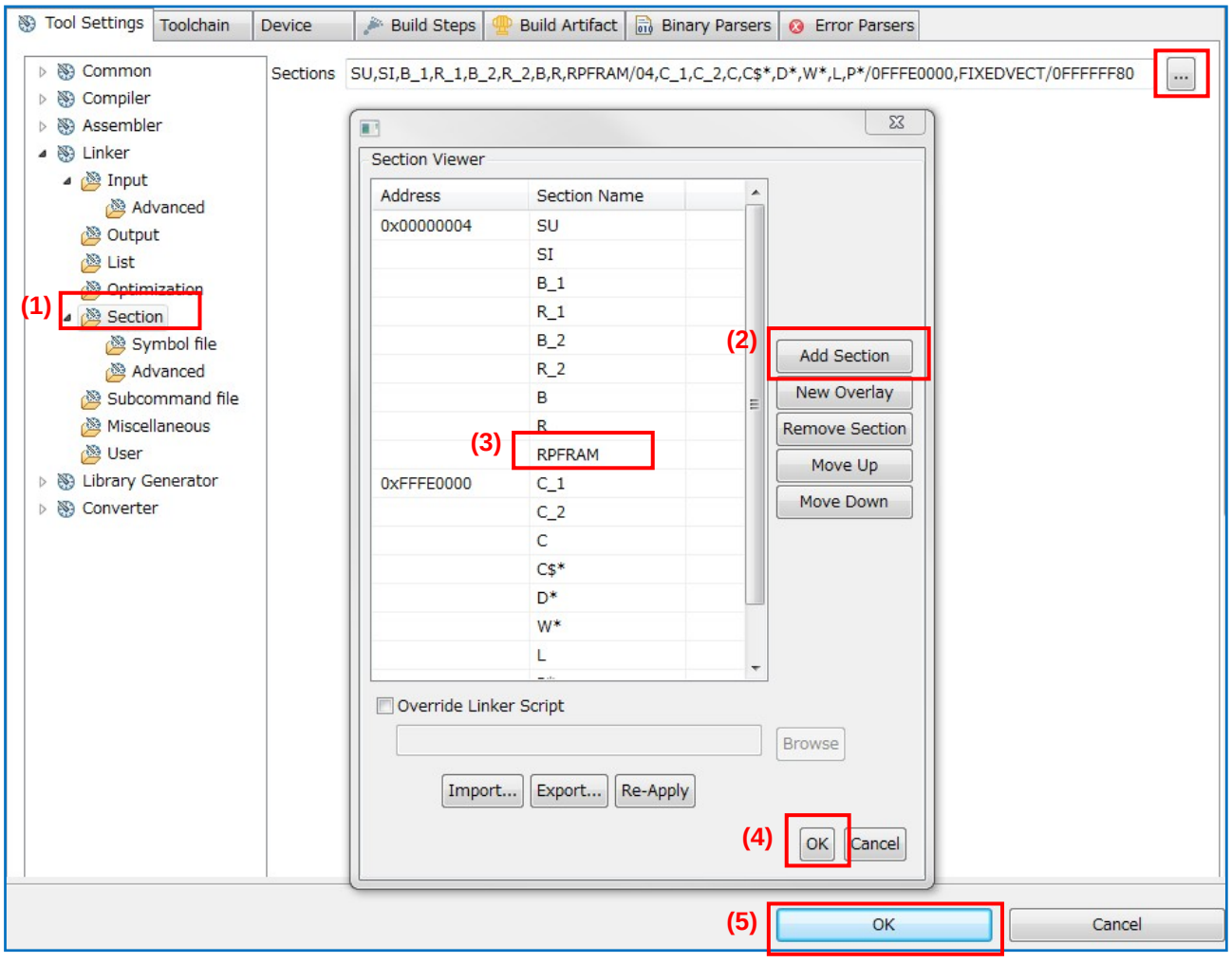
(1) Click the Section.

(2) Click the Add section button.

(3) Type “RPFRAM”.

(4) Click the OK button.

(5) Click the OK button.



Specifying the Section for ROM to RAM Mapping

Add “PFRAM=RPFRAM”.

(1) Click the Symbol file.

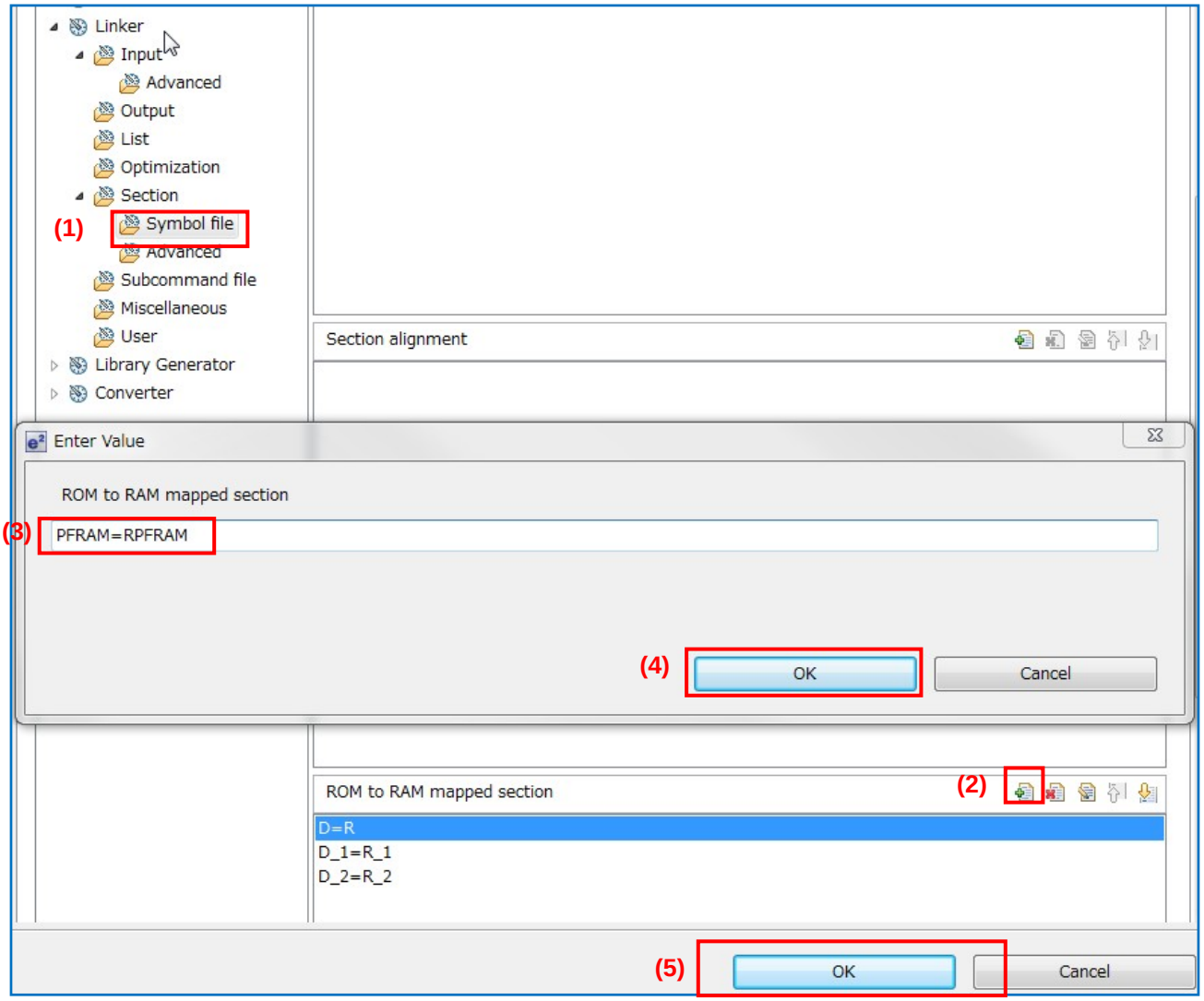
(2) Click the Add icon.

(3) Type “PFRAM=RPFRAM”.

(4) Click OK.

(5) Click the OK button.

The program transferred to the RAM has the setting “R_BSP_ATTRIB_SECTION_CHANGE(P,FRAM)” and is allocated in the PFRAM section. With this setting, addresses in the PFRAM section (ROM) can be mapped to the RPFRAM section (RAM).



2.11.2 GCC for Renesas RX

Edit the linker_script.ld file to add sections and symbols.

Adding Sections and Symbols

Add the following three codes.

(a)

```
. += _edata - _data;
```

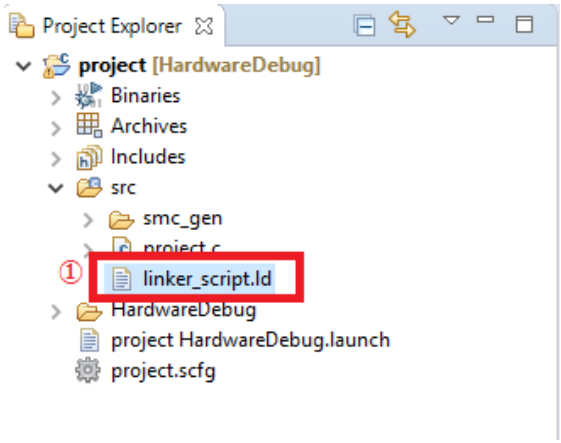
(b)

```
.pfram ALIGN(4):
{
    _PFRAM_start = .;
    . += _RPFRAM_end - _RPFRAM_start;
    _PFRAM_end = .;
} > ROM
```

(c)

```
.rpfram ALIGN(4): AT(_PFRAM_start)
{
    _RPFRAM_start = .;
    *(PFRAM)
    . = ALIGN(4);
    _RPFRAM_end = .;
} > RAM
```

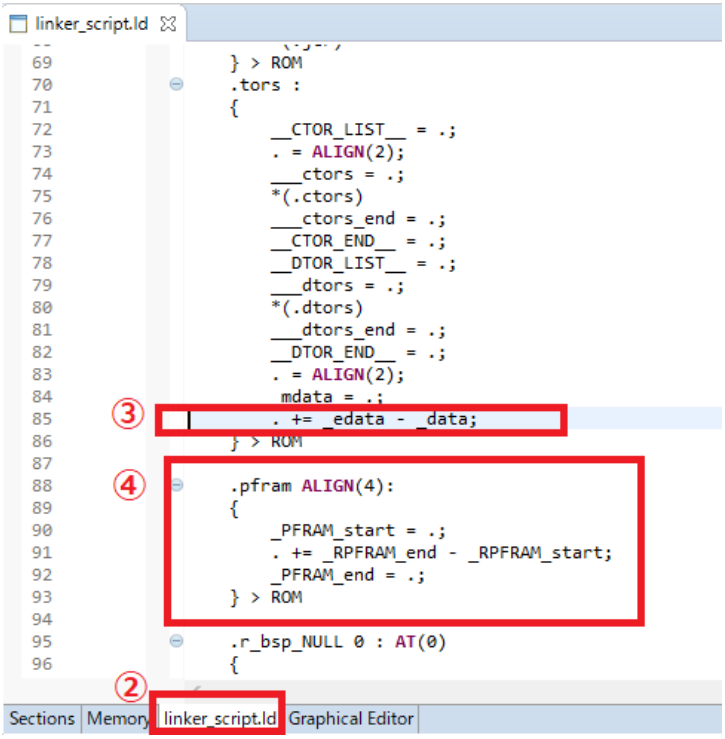
(1) Open "linker_script.ld" from Project Explorer.



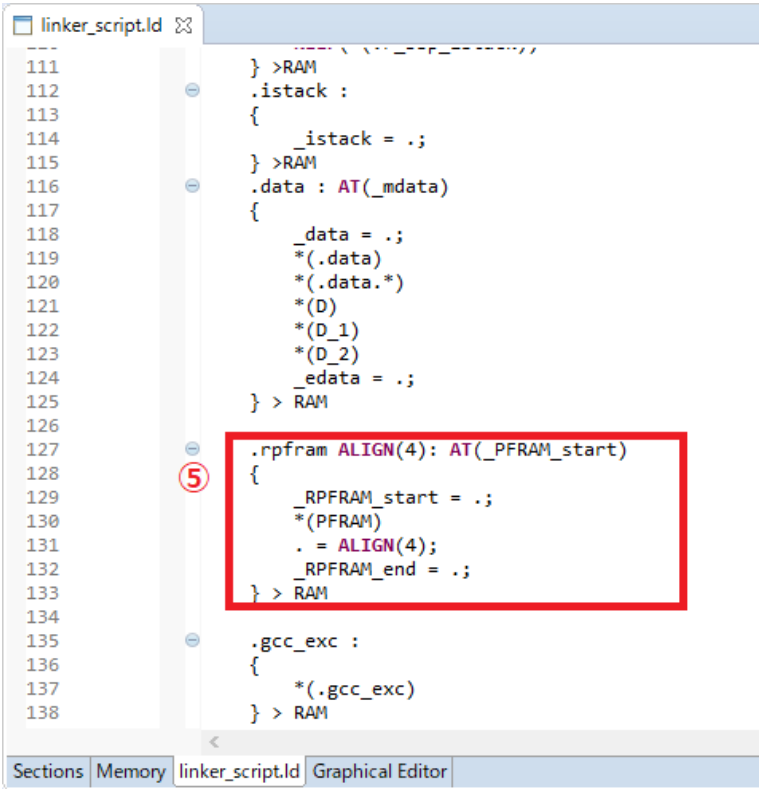
(2) Click "linker_script.ld".

(3) Enter the code (a) under " _madta = .;".

(4) Enter the code (b) under " .tors section".



(5)Enter the code (c) under " .data section".



2.11.3 IAR C/C++ Compiler for Renesas RX

Edit the icf file to add the section setting.

The icf file to be edited differs depending on the target device of the project.

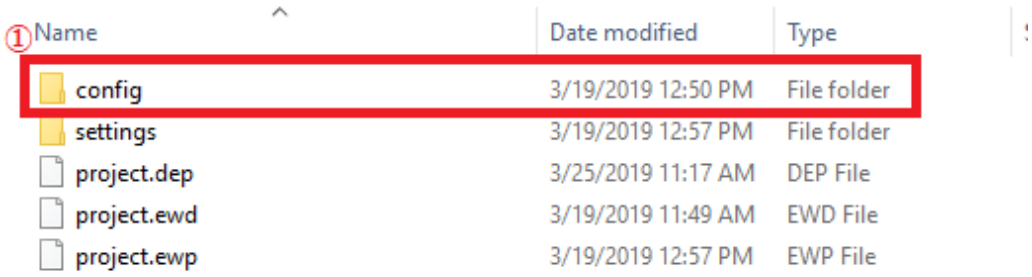
Confirm and edit the upper 8 digits of the model name of the device to be used.

For example, edit "lnkr5f51138.icf" for RX113 (R5F51138ADFP).

The following is an example of editing on RX113 (R5F51138ADFP).

Adding the Section Setting

(1) Create the "config" folder in the project folder.



(2) Copy "lnkr5f51138.icf" to the "config" folder of the project folder from "\rx\config" in the folder in which IAR C/C++ Compiler for Renesas RX (hereinafter described as "EWRX") is installed.

The default at installation is "C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.1".

(3) Open the copied "lnkr5f51138.icf" file and add "section PFRAM" to "initialize manually".

(4) Enter the following below " define block ISTACK…".

```
define block PFRAM with alignment = 4 {section PFRAM};
define block PFRAM_init with alignment = 4 {section PFRAM_init};
```

(5) add "block PFRAM_init" to "place in ROM_region32".

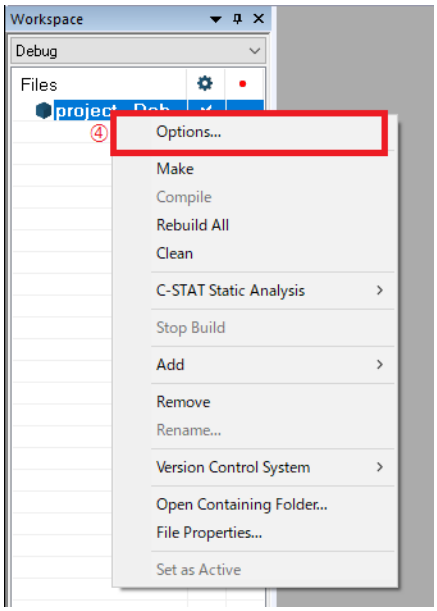
(6) add “block PFRAM” to "place in RAM_region32" under “ro section D_2”

lnkr5f51138.icf - Notepad

File Edit Format View Help

```
define region DATA_FLASH    = mem:[from 0x00100000 to 0x00101FFF];

③ initialize manually { rw section .textrw,section PFRAM};
initialize by copy { rw, ro section D, ro section D_1, ro section D_2 };
initialize by copy with packing = none { section __DLIB_PERTHREAD };
do not initialize  { section .*.noinit };

define block HEAP     with alignment = 4, size = _HEAP_SIZE { };
define block USTACK   with alignment = 4, size = _USTACK_SIZE { };
define block ISTACK   with alignment = 4, size = _ISTACK_SIZE { };

④ define block PFRAM with alignment = 4 {section PFRAM};
define block PFRAM_init with alignment = 4 {section PFRAM_init};

define block STACKS with fixed order { block USTACK,
                                       block ISTACK };

place at address mem:0xFFFFFFFC      { ro section .resetvect };
place at address mem:0xFFFFFF80      { ro section .exceptvect };

"ROM16":place in ROM_region16        { ro section .code16*,
                                       ro section .data16*};
"RAM16":place in RAM_region16        { rw section .data16*,
                                       rw section __DLIB_PERTHREAD};
"ROM24":place in ROM_region24        { ro section .code24*,
                                       ro section .data24* };
"RAM24":place in RAM_region24        { rw section .data24* };
"ROM32":place in ROM_region32        { ro,
⑤                                      block PFRAM_init};
 RAM32 :place in RAM_region32        { rw,
                                       ro section D,
                                       ro section D_1,
                                       ro section D_2,
                                ⑥     block PFRAM,
                                       block HEAP };
"STACKS":place at end of RAM_region32 { block STACKS };
```

(7) Open the project you want to use from EWRX, right-click on the project in the workspace and open the



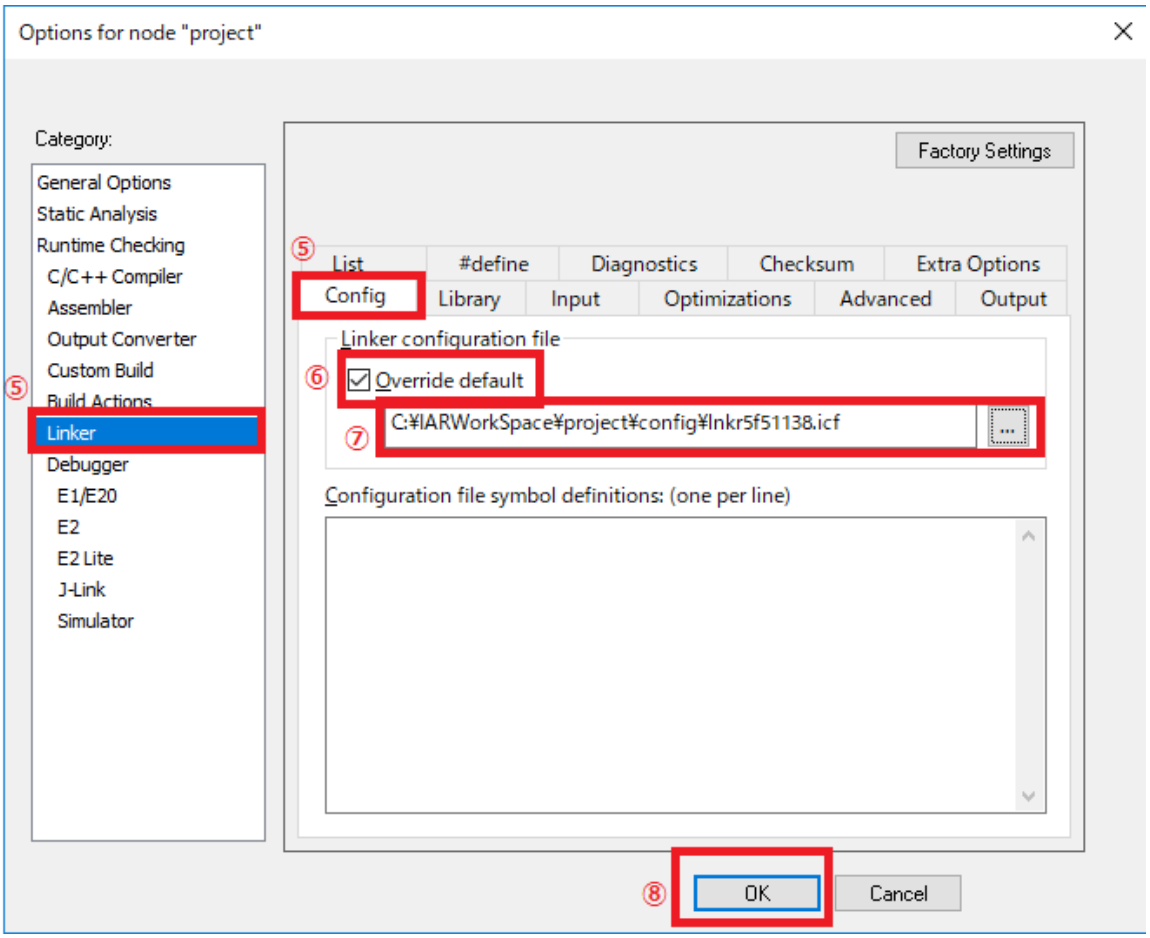
option.

(8) Select "Category: Linker" and click “Config”

(9) Select the "Override default" check box.

(10) Set the file that was edited in (3) to the reference destination.

(11) Click OK.


Options for node "project"
Category:
General Options
Static Analysis
Runtime Checking
C/C++ Compiler
Assembler
Output Converter
Custom Build
Build Actions
Linker
Debugger
E1/E20
E2
E2 Lite
J-Link
Simulator
Factory Settings
⑤
List
#define
Diagnostics
Checksum
Extra Options
Config
Library
Input
Optimizations
Advanced
Output
Linker configuration file
⑥
Override default
⑦
C:¥IARWorkSpace¥project¥config¥lnkr5f51138.icf
...
Configuration file symbol definitions: (one per line)
⑧
OK
Cancel

2.12 Note on Using API Functions

The sequencer enters ROM P/E mode to read the unique ID. Values in the ROM cannot be read during ROM P/E mode. The vector table is placed in the ROM as default. Therefore if an interrupt occurs while the unique ID is read (during ROM P/E mode), the interrupt handler cannot be executed, and then the program goes out of control. To keep this from happening, disable generating interrupt requests before executing the R_UID_Read() function, or allocate the vector table and the program for interrupt handler in the RAM, and change the value in the interrupt table register (INTB). Also do not generate a non-maskable interrupt request.

2.13 Note During On-Chip Debugging

Values in the ROM cannot be read while the R_UID_Read function is executed.

3. API Functions

3.1 R_UID_Open ()

This function transfers the program to read the unique ID from the ROM to the RAM for using this module.

Format

uid_err_t R_UID_Open(void)

Parameters

None

Return Values

UID_SUCCESS */* R_UID_Open executed successfully. */*
UID_ERR_LOCK_FUNC */* R_UID_Open or R_UID_Read executed while either R_UID_Open or R_UID_Read is executed. */*
UID_ERR_FAILURE */* R_UID_Open executed twice or more. */*

Properties

Prototyped in r_uid_rx_if.h.

Description

Prepares to read the unique ID. The program to read the unique ID is transferred from the ROM to the RAM.

Example

```
uid_err_t err;

/* Initialization of the API. */
err = R_UID_Open();

/* Check error */
if ( UID_SUCCESS != err)
{
   …
}
```

Special Notes

None

3.2 R_UID_Read ()

This function reads the unique ID from the extra area.

Format

uid_err_t R_UID_Read(uint8_t *pdest_addr)

Parameters

*uint8_t** *pdest_addr*

Specify the pointer to store the unique ID.

32-byte of the unique ID data is stored in the area specified with the pointer.

The size of the area specified with the pointer must be 32 bytes or more.

Return Values

UID_SUCCESS	*/* Unique ID read successfully with R_UID_Read. */*
UID_ERR_UNINITIALIZED	*/* R_UID_Read executed before executing R_UID_Open. */*
UID_ERR_LOCK_FUNC	*/* R_UID_Open or R_UID_Read executed while either R_UID_Open or R_UID_Read is executed. */*
UID_ERR_FAILURE	*/* Failed to read the unique id with R_UID_Read. */*

Properties

Prototyped in r_uid_rx_if.h.

Description

Reads the unique ID from the extra area and stores it in the specified area.

Example

```
uid_err_t err;
uint8_t unique_id[32];     /* Variable that stores the unique ID read */

/* Unique ID read */
err = R_UID_Read(unique_id);

/* Check error */
if ( UID_SUCCESS != err)
{
   …
}
```

Special Notes

Before executing this function, disable generating interrupt requests, or allocate the vector table and the program for interrupt handler in the RAM, and change the value in the interrupt table register (INTB). Also do not generate a non-maskable interrupt request.

3.3 R_UID_GetVersion ()

This function returns the API version.

Format

uint32_t R_UID_GetVersion(void)

Parameters

None

Return Values

Version number

Properties

Prototyped in r_uid_rx_if.h.

Description

Returns the API version number.

Example

```
uint32_t version;

version = R_UID_GetVersion();
```

Special Notes

None

4. Appendices

4.1 Confirmed Operation Environment

This section describes confirmed operation environment for the UID FIT module.

Table 4.1 Confirmed Operation Environment (Rev. 1.13)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.4.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev. 1.13
Board used	RX13T CPU Card (product No.: RTK0EMXA10C00000BJ)

Table 4.2 Confirmed Operation Environment (Rev. 1.11)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.3.0 IAR Embedded Workbench for Renesas RX 4.10.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201801 Compiler option: The following option is added to the default settings of the integrated development environment. -std = gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev. 1.11
Board used	Renesas Starter Kit for RX113 (product No.: R0K505113S900BE)

Table 4.3 Confirmed Operation Environment (Rev. 1.10)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 6.1.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev. 1.10
Board used	Renesas Starter Kit for RX130 (product No.: RTK5005130S00000BE)

4.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"

- Using e^2 studio:

  Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"

When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)".

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_uid_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

5. Reference Documents

User’s Manual: Hardware
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools
RX Family C/C++ Compiler CC-RX User's Manual (R20UT3248)
The latest version can be downloaded from the Renesas Electronics website.

Related Technical Updates

This module reflects the content of the following technical updates.

None

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.01.14	—	First edition issued
1.10	Dec.06.17	1	Target Device: Added the RX130 support.
		6	2.2 Software Requirements: Renesas Board Support Package (r_bsp) v3.60 or higher for RX130 is updated.
		6	2.3 Supported Toolchains: Renesas RX Toolchain version is updated to v2.07.00 for RX130 is added.
1.11	May.20.19	—	Supported the following compilers: - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX
		1	Added the section of Target Compilers.
		3	Updated the section of 1.2 Overview of the UID FIT Module.
		6	Added the section of 1.5 Limitations.
		8	Added the section of 2.7 Code Size.
		11	Added the section of 2.10 “for”, “while” and “do while” statements.
		12-16	Added the sections of 2.11.2 GCC for Renesas RX and 2.11.3 IAR C/C++ Compiler for Renesas RX to the section of 2.11 Setting a Project.
		20	Updated the section of R_UID_GetVersion().
		21	Added Table 4.1 Confirmed Operation Environment (Rev. 1.11) to the section of 4.1 Confirmed Operation Environment.
		program	Deleted the inline expansion of the R_UID_GetVersion function.
1.12	Jul.05.19	14-15	Updated the sections of 2.11.2 GCC for Renesas RX.
1.13	Jul.31.19	1	Target Device: Added the RX13T support.
		8	Updated the sections of 2.7 Code Size.
		23	Added Table 4.1 Confirmed Operation Environment (Rev. 1.13) to the section of 4.1 Confirmed Operation Environment.
1.14	Jun.10.20	—	Changed API function comments to Doxygen style.
		1	Deleted Related Documents R01AN1833.
		10	Changed Section 2.9 Adding the FIT Module to Your Project.
		20-22	Deleted “Reentrant” item on the API description page.

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 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/.