RX Family

System Timer Module Firmware Integration Technology

Introduction

This application note describes the System Timer module which uses Firmware Integration Technology (FIT). This module uses System Timer to counts the year/month/day/hour/minute/seconds, and has simple scheduler function. In this document, this module is referred to as the System Timer FIT module.

Target Device

RX Family

Target Compilers

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

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)
- RX Smart Configurator User's Guide: e² studio (R20AN0451)
- RX Smart Configurator User's Guide: CS+ (R20AN0470)
- RX Smart Configurator User's Guide: IAREW (R20AN0535)
- CMT Module Using Firmware Integration Technology (R01AN1856)

Contents

1. Overview

1.1 System Timer FIT Module

The System Timer FIT module can be used by being implemented in a project as an API. See section 1.12 Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the System Timer FIT Module

The system timer counts the year/month/day/hour/minute/seconds using CMT in RX Family MCUs. And, this module also has the scheduler function. Users can be executed cyclic function by the system timer to register the function pointer with interval.

1.3 API Overview

Table 1.1 lists the API functions included in this module.

Table 1.1 API Functions

Function	Description
R_SYS_TIME_Open()	Open system timer module
R_SYS_TIME_GetCurrentTime()	Get system time from system timer
R_SYS_TIME_SetCurrentTime()	Set system timer to system timer
R_SYS_TIME_ConvertUnixTimeToSystemTime()	Convert Unix time to system time format
R_SYS_TIME_RegisterPeriodicCallback()	Register the cyclic function (max: 30)
R_SYS_TIME_UnregisterPeriodicCallback()	Unregister the cyclic function
R_SYS_TIME_IsPeriodicCallbackRegistered()	Confirm cyclic functions
R_SYS_TIME_Close()	Close system timer module
R_SYS_TIME_GetVersion()	Get the system timer version information

1.4 File Structure

This application note includes following files.

Table1.1 File Structure 1

File/Directory(Bold) Names	**Description**
r20an0431ej0101-rx-middle.pdf	System timer module application note. (English, this document)
r20an0431jj0101-rx-middle.pdf	System timer module application note. (Japanese)
FITModules	FIT Module Folder
r_sys_time_rx_v1.01.zip	System timer module
r_sys_time_rx_v1.01.xml	System timer module e2 studio XML file for FIT configurator

The folder to which the contents of r_sys_time_rx_v1.01.zip is extracted will contain the files listed in table 1.2 below.

Table1.2 File Structure 2

File/Directory(Bold) **Names**	**Description**
r_config	System timer config file folder
r_sys_time_rx_config.h	System timer config file (default settings)
r_sys_time_rx	System timer FIT Module folder
src	System timer source code folder
r_sys_time_rx.c	System timer source code
r_sys_time_rx_private.h	System timer header file for internal
doc	System timer document folder
ja	System timer document folder (Japanese)
r20an0431jj0101-rx-middle.pdf	System timer application note (Japanese)
en	System timer document folder (English)
r20an0431ej0101-rx-middle.pdf	System timer application note (English)
ref	System timer config file (template) folder
r_sys_time_rx_config_reference.h	System timer config file (template)
r_sys_time_rx_if.h	System timer header file
readme.txt	readme

2. API Information

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

2.1 Hardware Requirements

- CMT

2.2 Software Requirements

This driver is dependent upon the following FIT module:

- r_bsp
- r_cmt_rx

2.3 Supported Toolchain

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

2.4 Interrupt Vector

None.

2.5 Header Files

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

2.6 Integer Types

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

2.7 Configuration Overview

The configuration option settings of this module are located in r_sys_time_rx_config.h. The option names and setting values are listed in the table below:

Configuration options in r_sys_time_rx_config.h	
None	-

2.8 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_sys_time_rx Rev.1.01

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 4.11.1

(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
RX64M	ROM	**2.0k bytes**	**3.2k bytes**	**2.5k bytes**
	RAM	**0. 5k bytes**	**0.5k bytes**	**0.5k bytes**
	STACK (Note 1)	**108 bytes**	**-**	**80 bytes**

Note 1: in executing R_SYS_TIME_SetCurrentTime().

2.9 Parameters

This section describes the parameter structure used by the API functions in this module. The structure is located in r_sys_time_rx_if.h as are the prototype declarations of API functions.

2.10 Return Values

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

```
typedef enum e_sys_time_err
{
      SYS_TIME_SUCCESS=0,                 /* Normally terminated. */
      SYS_TIME_ERR_BAD_CHANNEL,           /* Non-existent channel number. */
      SYS_TIME_ERR_BAD_INTERVAL,          /* Bad interval parameter is
                                             specified. */
      SYS_TIME_ERR_BAD_TIME_OFFSET,       /* Bad time offset is set. */
      SYS_TIME_ERR_BAD_FUNCTION_POINTER,  /* Bad function pointer is set. */
      SYS_TIME_ERR_BAD_SYS_TIME,          /* Bad system timer value is input */
      SYS_TIME_ERR_ALREADY_STARTED,       /* System timer is already started. */
      SYS_TIME_ERR_NOT_STARTED,           /* System timer is not started. */
      SYS_TIME_ERR_FULL_REGISTERED,       /* All register table is used. */
      SYS_TIME_ERR_ALREADY_REGISTERED,    /* Specified function pointer has been
                                             already registered. */
}
sys_time_err_t;
```

2.11 Callback Function

In this module, the callback function specified by the user is called when the CMT interrupt occurs.

The callback function is specified by storing the address of the user function in the "function_pointer" argument in R_SYS_TIME_ConvertUnixTimeToSystemTime() function.

```
/* Callback function usage example */
  void my_sys_time_callback(void)
  {
       …
  }
```

2.12 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) 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 “Renesas e^2 studio Smart Configurator User Guide (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 “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 “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” 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 “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

2.13 "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 */
```

2.14 Section information

System timer module uses the default sections.

3. API Functions

R_SYS_TIME_Open ()

Format

#include “r_sys_time_rx_if.h”

sys_time_err_t R_SYS_TIME_Open(void);

Parameters

None

Return Values

SYS_TIME_SUCCESS	Normal terminated
SYS_TIME_ERR_BAD_CHANNEL	No CMT channel is exist
SYS_TIME_ERR_ALREADY_STARTED	Already started

Description

Open system timer module.

R_SYS_TIME_GetCurrentTime ()

Format

#include “r_sys_time_rx_if.h”

sys_time_err_t R_SYS_TIME_GetCurrentTime(SYS_TIME *sys_time);

Parameters

sys_time	input/output	Area for acquiring the system time from the system timer

Return Values

SYS_TIME_SUCCESS	Normal terminated

Description

Get the system time from system timer.

R_SYS_TIME_SetCurrentTime ()

Format

#include “r_sys_time_rx_if.h”

sys_time_err_t R_SYS_TIME_SetCurrentTime(SYS_TIME *sys_time);

Parameters

sys_time	input/output	Area of system time setting to system time

Return Values

SYS_TIME_SUCCESS	Normal terminated
SYS_TIME_ERR_BAD_SYS_TIME	Illegal system time is set

Description

Set the system time to the system timer. And, update Unix time is included into the system timer.

R_SYS_TIME_ConvertUnixTimeToSystemTime ()

Format

#include “r_sys_time_rx_if.h”

sys_time_err_t R_SYS_TIME_ConvertUnixTimeToSystemTime(

uint32_t unix_time, SYS_TIME *sys_time, uint8_t *time_offset);

Parameters

unix_time	input	Unix time
sys_time	input/output	Area of system time setting to system time
time_offset	input	String to specify the time zone

Return Values

SYS_TIME_SUCCESS	Normal terminated
SYS_TIME_ERR_BAD_TIME_OFFSET	Illegal time zone is set

Description

Set the system time to the system timer using Unix time. To set the string that represents the time zone to time_offset. The strings that represents the time zone are defined into r_sys_time_rx_if.h.

R_SYS_TIME_RegisterPeriodicCallback ()

Format

#include “r_sys_time_rx_if.h”

sys_time_err_t R_SYS_TIME_RegisterPeriodicCallback(

callback_from_sys_time_t function_pointer, uint32_t interval)

Parameters

function_pointer	input	Function pointer
interval	input	Cyclic interval (unit=10ms)

Return Values

SYS_TIME_SUCCESS	Normal terminated
SYS_TIME_ERR_BAD_FUNCTION_POINTER	Illegal function pointer is set
SYS_TIME_ERR_BAD_INTERVAL	Illegal interval is set
SYS_TIME_ERR_FULL_REGISTERED	Reach the upper limit of registration
SYS_TIME_ERR_ALREADY_REGISTERED	Known function pointer is specified

Description

Users can be executed cyclic function by the system timer to register the function pointer with interval. Max 30 functions can be registered. The function pointer is executed into CMT interrupt. This CMT interrupt enables interrupt. Please execute process that needs realtime into the interrupt function that has higher priority than CMT interrupt priority (can be set at r_cmt_rx_config.h).

R_SYS_TIME_UnregisterPeriodicCallback ()

Format

#include “r_sys_time_rx_if.h”

sys_time_err_t R_SYS_TIME_UnregisterPeriodicCallback(callback_from_sys_time_t function_pointer);

Parameters

function_pointer	input	Function pointer

Return Values

SYS_TIME_SUCCESS	Normal terminated
SYS_TIME_ERR_BAD_FUNCTION_POINTER	Illegal function pointer is set

Description

Unregister the cyclic function.

R_SYS_TIME_IsPeriodicCallbackRegistered ()

Format

#include “r_sys_time_rx_if.h”

bool R_SYS_TIME_IsPeriodicCallbackRegistered(callback_from_sys_time_t function_pointer);

Parameters

function_pointer	input	Function pointer

Return Values

true	Already registered
false	Not registered yet

Description

Confirm the registration of cyclic function

R_SYS_TIME_Close ()

Format

#include “r_sys_time_rx_if.h”

sys_time_err_t R_SYS_TIME_Close(void);

Parameters

None.

Return Values

SYS_TIME_SUCCESS	Normal terminated
SYS_TIME_ERR_BAD_CHANNEL	Failed CMT channel close
SYS_TIME_NOT_STARTED	Not executed system timer yet

Description

Stop the system timer

R_SYS_TIME_GetVersion ()

Format

#include “r_sys_time_rx_if.h”

uint32_t R_SYS_TIME_GetVersion(void);

Parameters

None

Return Values

Version information about this module.

Description

The function returns the version of this module. The version number is encoded such that the top two bytes are the major version number and the bottom two bytes are the minor version number.

4. Appendices

4.1 Confirmed Operation Environment

This section describes confirmed operation environment for the System Timer FIT module.

Table 4.1 Confirmed Operation Environment (Rev. 1.01)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.3.0 IAR Embedded Workbench for Renesas RX 4.11.1
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 4.11.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.1.01
Board used	Renesas Starter Kit+ for RX65N-2MB (RTK50565Nxxxxxx) Renesas Starter Kit+ for RX64M (R0K50564Mxxxxxx)

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)".

Revision History

Rev.	Date	Description	
		Page	Summary
1.01	Jun.28.2019	-	Added Target Compiler: GCC for Renesas RX, IAR C/C++ Compiler for Renesas RX Changed Return Values from SYS_TIME_BAD_XXX to SYS_TIME_ERR_BAD_XXX. Bug Fix: - Fixed the problem that time information is incorrect when an interrupt occurs during R_SYS_TIME_GetCurrentTime ().
1.00	Nov 30, 2016	-	First Release.

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/.