RX63N Group

Simple Timer Framework

R01AN0952EU0100
Rev.1.00
Jan 15, 2012

Introduction

This application note demonstrates the features and capabilities of Simple Timer Framework. Although the sample demo project accompanying the application note runs on RSK+RX63N, the Simple Timer Framework can run on any hardware platform.

Target Devices

RX63N Group.

Contents

1. Overview

The Simple Timer Framework is a linked list timers with callback feature. It is designed to schedule future events and to take necessary actions required when these events occur. The framework is very flexible and can be used on any hardware platform. Any module or a software component of a user application can use the services of Simple Timer Framework. There is no limit of timers that can be in the framework.

2. Simple Timer Framework

This section provides details for the Simple Timer Framework. The features and limitations of the framework and the requirements to include the framework into a user application are described.

2.1 Requirements

The Simple Timer Framework requires a periodic timer tick generated by the hardware. Any simple timer peripheral can be used for this purpose. Renesas MCUs provide several timer peripherals such as TMR and CMT that can generate a timer tick. The tick rate is depended on the requirements of the user application.

The only two files are needed to include the Simple Timer Framework into a user application are simple_timer.c and simple_timer.h files. The C source file contains the implementation of the Simple Timer Framework APIs. The H header file includes API and type definitions used by the framework. The Simple Timer Framework uses less than 200 bytes of ROM and less than 10 bytes of RAM.

The number of timer objects defined and used by a user application is only limited by the amount of available RAM. Any module or software component that wants to use the services of the framework must include the H header file and make the proper API calls. A Simple Timer Framework object uses 36 bytes of RAM.

2.2 Internal Workings

The Simple Timer Framework object type structure is shown in Figure 1. Name field is an ASCII string containing the name of the timer. Field is_active indicates whether the timer has started or not. Fields start_time and stop_time store exactly what they indicate by their names. The callback_func and param are used to call a registered function when a timer expires. The param argument is passed to this callback function. The callback function depends on the user application requirements and each timer in the list can point to a different function. Fields *prev and *next are used to link the timers.

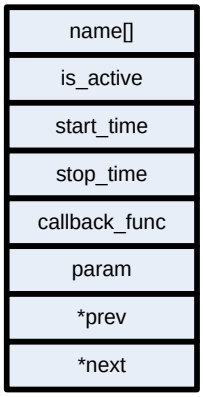

Figure 1 Simple Timer Type

Active timers are sorted by stop_time value and the timer with lowest stop_time is pointed by the list_head pointer. This is shown in Figure 2. For simplicity only stop_time and link pointers are shown.

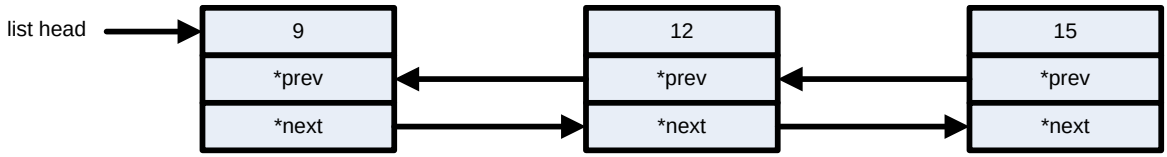

Figure 2 Linked List Timers

A new timer is inserted in the list by first searching the stop_time values in the list against the timer's stop_time and then by modifying the link pointers. Figure 3 shows a new timer with stop_time value equal to 13 inserted into the linked list.

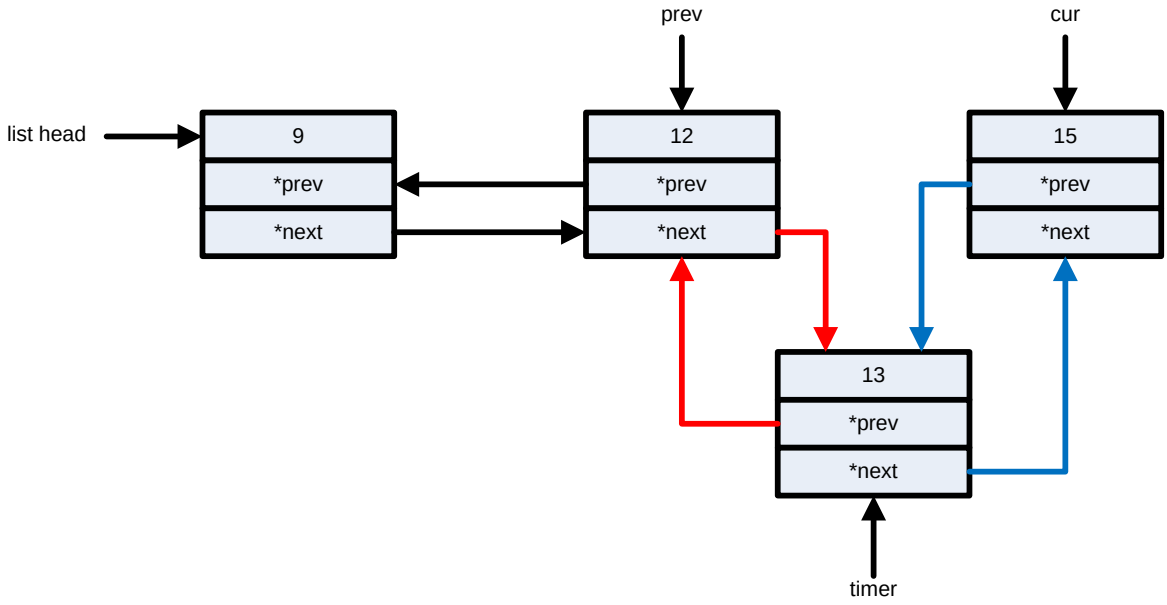

Figure 3 Inserting a New Timer

A timer expires if the Simple Timer Framework timer tick becomes equal to or greater than the timer's stop_time. In this case, the timer is removed from the list and the registered callback function is called with the param argument. The Simple Timer Framework provides the capability to cancel an active timer. Canceled timer is removed from the linked list by updating the link pointers as needed.

3. Sample Demo Project

This application note is provided with a sample demo project that runs on RSK+RX63N. However, the Simple Timer Framework can be made to run on any hardware platform. This section describes the sample demo project.

3.1 Demo Directory Structure

Figure 4 shows the directory structure of the Simple Timer Framework demo project. The bsp folder contains the Renesas board specific source code. The code folder has the Simple Timer Framework source files along with debug console to test the framework and TMR module to generate a timer tick.

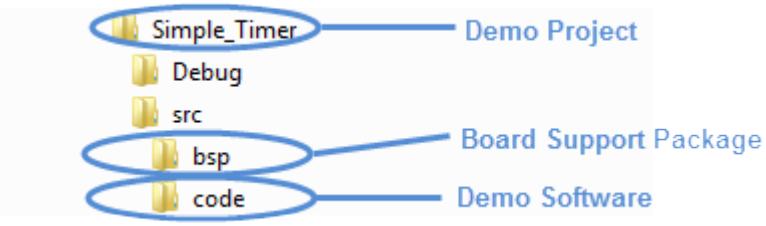

Figure 4 Directory Structure of Simple Timer Framework Demo

3.2 Demo Setup

RSK+RX63N board initializations are done in files in bsp folder. The main.c source code initializes the LED, LCD, and push button switches. The sample demo project uses 1 second timer tick generated by TMR module and it is initialized in main.c file. The Simple Timer Framework is also initialized in main.c.

The sample demo project uses HEW debug console to send and receive simple commands to start, cancel and dump the list of active timers. HEW debug console is particularly useful tool since it does not tide up any serial port resources on RX63N. Further, it uses the Renesas E1 debug connection so there is no need for other cables or USB to serial converters.

Following is the message format for the serial test commands.

Start a timer: s<space><timer id><space><duration><CR>

Cancel a timer: c<space><timer id><CR>

Dump timer list: d<CR>

The sample demo project uses 3 timers to display how the Simple Timer Framework works. Examples below show sample outputs from the debug console.

```
d
dumping timer list -- current time(4)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========

s 1 30
start timer: timer1 for 30 seconds -- Success

s 0 10
start timer: timer0 for 10 seconds -- Success

d
dumping timer list -- current time(26)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
    timer0          23          33  0xfff8109b  0x00000000
    timer1          13          43  0xfff8109b  0x00000001

*** simple timer test callback: timer0 expired -- current time(33) ***

*** simple timer test callback: timer1 expired -- current time(43) ***

d
dumping timer list -- current time(47)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
```

Initially there are no active timers in the list. Timer 1 is started with 30 second duration. Then timer 0 starts with 10 seconds. A list of active timers shows them sorted by their stop time. The list also shows the callback function and the parameter. When the timers expire, the callback function is called and timers are removed from the list.

```
s 0 100
start timer: timer0 for 100 seconds -- Success

s 1 200
start timer: timer1 for 200 seconds -- Success

s 2 300
start timer: timer2 for 300 seconds -- Success

d
dumping timer list -- current time(595)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
    timer0         582         682  0xfff8109b  0x00000000
    timer1         588         788  0xfff8109b  0x00000001
    timer2         594         894  0xfff8109b  0x00000002

c 1
cancel timer: timer1 -- Success

d
dumping timer list -- current time(605)
name        start time  stop time   callbackFn  param
==========  ==========  ==========  ==========  ==========
    timer0         582         682  0xfff8109b  0x00000000
    timer2         594         894  0xfff8109b  0x00000002
```

In this case, timers 0, 1, and 2 are started with 100, 200, and 300 second durations. The list shows all 3 timers are active and sorted by their stop time. The next command cancels timer 1 and this is shown in the second list.

4. Simple Timer Framework APIs

The following functions serve as the API for the Simple Timer Framework. This section describes each API function.

```
st_status_code_t   simple_timer_init(void);
st_status_code_t   simple_timer_init_timer(uint8_t const *name, st_timer_t *timer);
st_status_code_t   simple_timer_start_timer(st_timer_t *timer,
                                            uint32_t duration,
                                            st_callback_func_t callback_func,
                                            uint32_t param);
st_status_code_t   simple_timer_cancel_timer(st_timer_t *timer);
uint32_t           simple_timer_get_time(void);
st_status_code_t   simple_timer_set_time(uint32_t time);
st_timer_t         *simple_timer_get_head(void);
void               simple_timer_increment_time(void);
void               simple_timer_check_timer_expiry(void);
```

4.1 simple_timer_init

This API initializes the Simple Timer Framework.

Format

```
st_status_code_t simple_timer_init(void);
```

Parameters

None.

Return Values

ST_SUCEESS = Operation successful.

Any value other than ST_SUCCESS indicates a failure.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API initializes the Simple Timer Framework variables. It must be called before using the framework.

4.2 simple_timer_init_timer

This API initializes a Simple Timer Framework timer.

Format

```
st_status_code_t simple_timer_init_timer(uint8_t const *name,
                                         st_timer_t *timer)
```

Parameters

`name`

An ASCII string indicating the name of the timer.

`timer`

Pointer to timer object to initialize.

Return Values

ST_SUCEESS = Operation successful.

Any value other than ST_SUCCESS indicates a failure.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API initializes a Simple Timer Framework timer object that is instantiated. The API takes a pointer to this timer and an ASCII string containing the name of the timer. It must be called after a timer object is created.

4.3 simple_timer_start_timer

This API starts a Simple Timer Framework timer.

Format

```
status_code_t simple_timer_start_timer(st_timer_t *timer,
                                       uint32_t duration,
                                       st_callback_func_t callback_func,
                                       uint32_t param)
```

Parameters

`timer`
Pointer to timer object to start.

`duration`
Timeout duration for the timer.

`callback_func`
Function pointer to call when timer expires.

`param`
Parameter that is passes to the callback function.

Return Values

ST_SUCEESS = Operation successful.

Any value other than ST_SUCCESS indicates a failure.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API starts a Simple Timer Framework timer. The timer to start is indicated by the timer pointer, timeout value is given in duration argument. The Simple Timer Framework uses the callback function to notify the use application that the timer expired. The param is a generic field to pass an argument to the callback function. It is recommended to typecast this field if a different type than uint32_t is needed by the user application. Since this field is 4-byte long, it can be used to pass a pointer as well.

If the timer is already stared (e.g. active) ST_FAILURE is returned. Successfully started timer is inserted in the linked list of timer sorted by the stop time.

4.4 simple_timer_cancel_timer

This API cancels a Simple Timer Framework timer.

Format

```
st_status_code_t simple_timer_cancel_timer(st_timer_t *timer)
```

Parameters

`timer`

Pointer to timer object to cancel.

Return Values

ST_SUCEESS = Operation successful.

Any value other than ST_SUCCESS indicates a failure.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API cancels an active Simple Timer Framework timer. Attempt to cancel a timer that is not started returns ST_FAILURE. Cancelled timer is removed from the linked list of active timers.

4.5 simple_timer_get_time

This API returns the Simple Timer Framework timer tick.

Format

```
uint32_t simple_timer_get_time(void)
```

Parameters

None.

Return Values

Returns the current value of timer tick in uint32_t type.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API is used to query the current value of Simple Timer Framework time tick.

4.6 simple_timer_set_time

This API sets the Simple Timer Framework timer tick.

Format

```
st_status_code_t simple_timer_set_time(uint32_t time)
```

Parameters

`time`

Timer tick value.

Return Values

ST_SUCEESS = Operation successful.

Any value other than ST_SUCCESS indicates a failure.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API is used to set the value of Simple Timer Framework timer tick. The default value for the timer tick is zero if this it is not explicitly set by this API.

4.7 simple_timer_get_head

This API returns the Simple Timer Framework head pointer.

Format

```
st_timer_t *simple_timer_get_head(void)
```

Parameters

None.

Return Values

Returns timer_list_head variable that points to the head of the active linked list timers.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API is used to get the value of Simple Timer Framework timer_list_head variable. A NULL indicates that there are no active timers in the list.

4.8 simple_timer_increment_time

This API increments the Simple Timer Framework timer tick.

Format

```
void simple_timer_increment_time(void)
```

Parameters

None.

Return Values

None.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API increments the Simple Timer Framework timer tick. A usual place to add this API call is in a periodic timer interrupt service routine.

4.9 simple_timer_check_timer_expiry

This API checks if a Simple Timer Framework timer expired.

Format

```
void simple_timer_check_timer_expiry(void)
```

Parameters

None.

Return Values

None.

Properties

Prototyped in file simple_timer.h.

Implemented in file simple_timer.c.

Description

This API is used to check if a Simple Timer Framework timer in the active timers list expired. Since the active timers are sorted by their stop time, it starts searching from the head of the list. If the timer pointed by the head pointer expired, it is removed from the list and the callback function associated with this timer is called. The API updates the head pointer with the next timer in the list and checks if this new timer expired. This continues either until there are no more timers in the list or the timer pointer by the head pointer still has more time to expire.

A usual place to add this API call is in the main loop.

5. References

1. Group Hardware Manuals for the Renesas device on the RSK board.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jan 15.2012	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. Handling of Unused Pins

   Handle unused pins in accord 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.1