RX Family

How to Compute RI600/4 User Stack and System Stack

R20AN0093ES0100
Rev.1.00
Oct 01, 2010

Introduction

Computing the required stack size is a critical step for embedded developers to avoid wasting any additional, unnecessary memory and ensuring stack overflows does not occur. Deciding how much memory to allocate for the stack has always been a trial and error process.

This document explains the method of computing the user stack and system stack for application written in RI600/4.

Target Device

Applicable MCU: RX Family

Contents

1. Guide in using this Document

This document aims to equip users with the technique of determining the required user/task stack and system stack for RI600/4.

Table 1 Explanation of Document Topics

Topic	Objective	Pre-requisite
Introduction to RI600/4	Explanation of user and system stack	Knowledge in RI600/4
Computing User Stack Size	Explains methods of computing user stack size to be allocated	Knowledge in RI600/4
Computing System Stack Size	Explains methods of computing system stack size to be allocated	Knowledge in RI600/4
Reference Documents	Listing of documents that equip users with knowledge in the pre-requisite requirements	None

2. Introduction to RI600/4 Stack

In a program with RI600/4, there are two types of stacks defined: user stack and system stack. User stacks are allocated for each individual task defined in the configuration file as shown in Figure 1. The size of user stack for each task is also specified in the configuration file. System stack refers to the memory allocated for the kernel and various handlers. The size of the system stack is also defined in the configuration file shown in Figure 1.

Caution will need to be exercised when allocating the sizes of the user stack and system stack. Allocating too much memory will result in wasting of memory. Allocating too less will result in stack overflows, which can corrupt other memory areas and typically trigger a program crash.

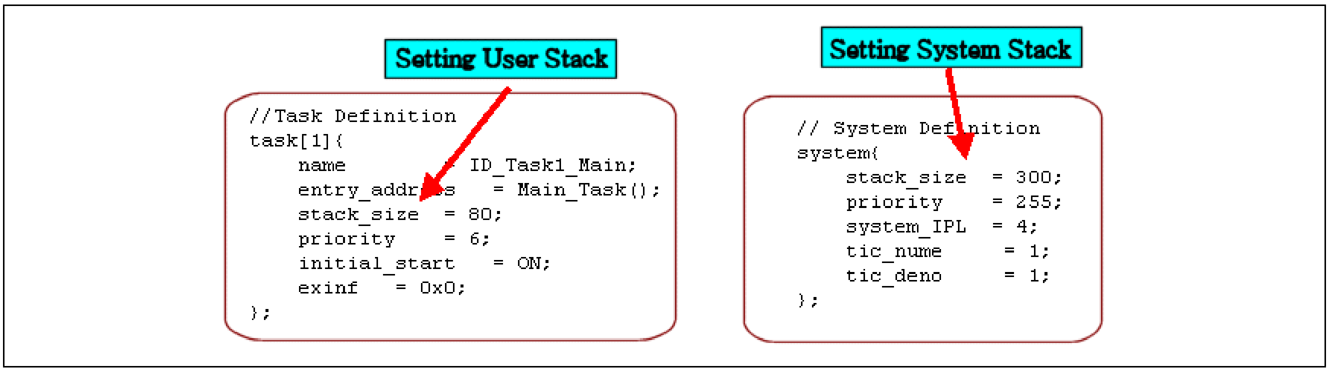

Figure 1 Setting User Stack and System Stack

3. Computing User Stack Size

User stack size of each task is calculated based on the following expression.

Necessary size of the user stack $= \alpha + \beta$

Figure 2 Formulae for User Stack Size Computation

3.1 Calculating α

α denotes the size consumed by task entry function. This size can be computed using Call Walker. To measure the stack size of α using Call Walker, either a stack information file (*.sni) or profile information file (*.pro) is required (Figure 3).

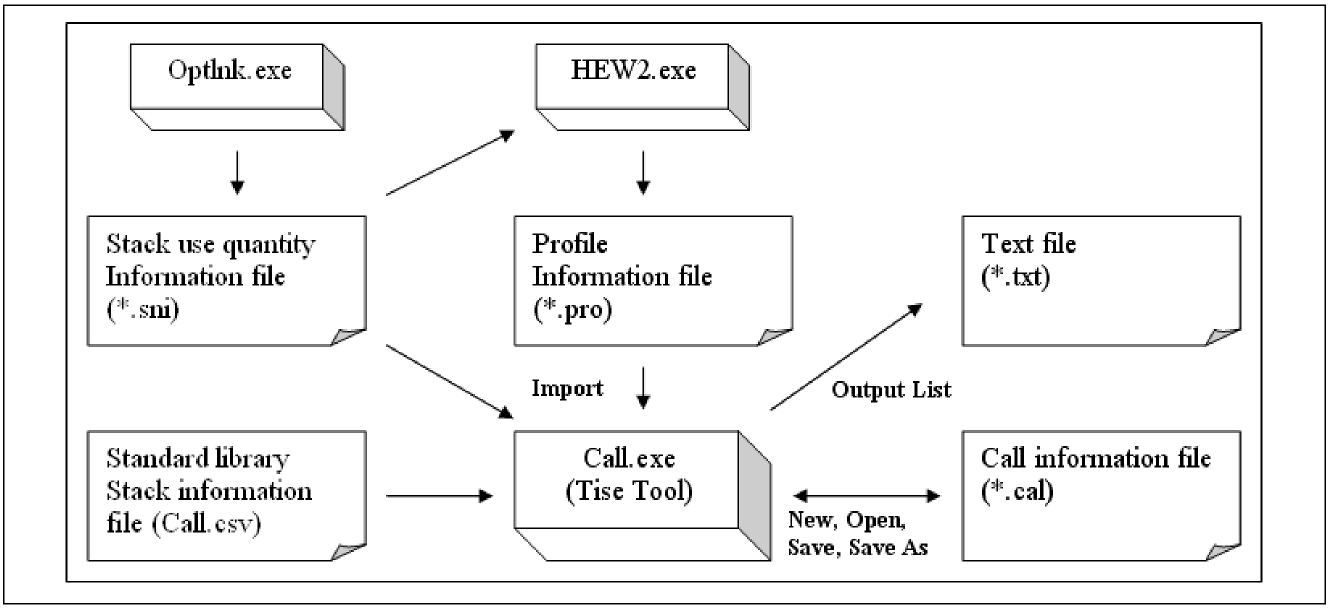

Figure 3 Linkage of each File in Call Walker

For the optimizing linkage editor to generate the stack information file, it is required to enable the "Stack information output" option in HEW. Figure 4 illustrates how to select the option.

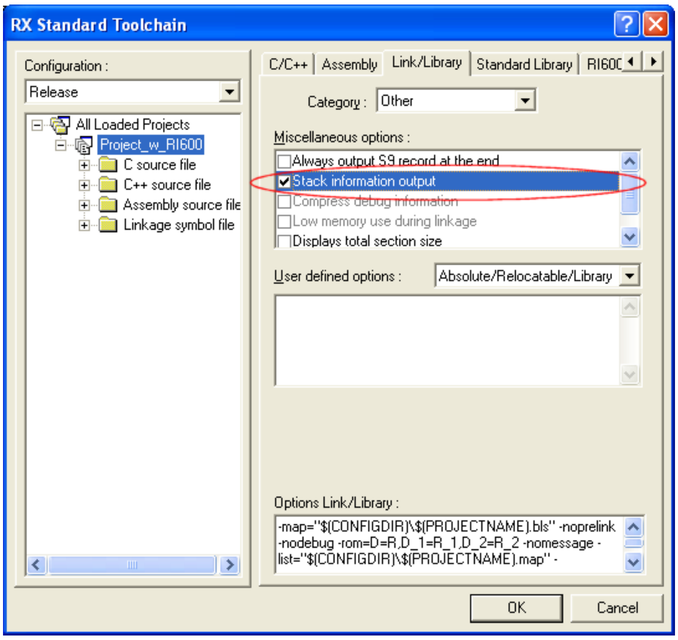

Figure 4 Enabling Generation of Stack Information File

Upon the compilation of the project, its corresponding stack information file (*.sni) will be generated shown in Figure 5.

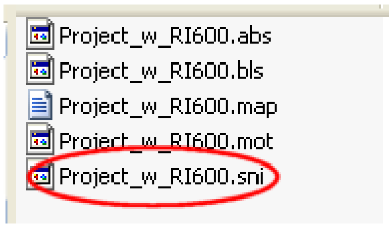

Figure 5 Stack Information File

The stack information file is then loaded into Call Walker by choosing [File]->[Import Stack File…]. Figure 6 provides a screenshot of the Call Walker with the loaded stack information file.

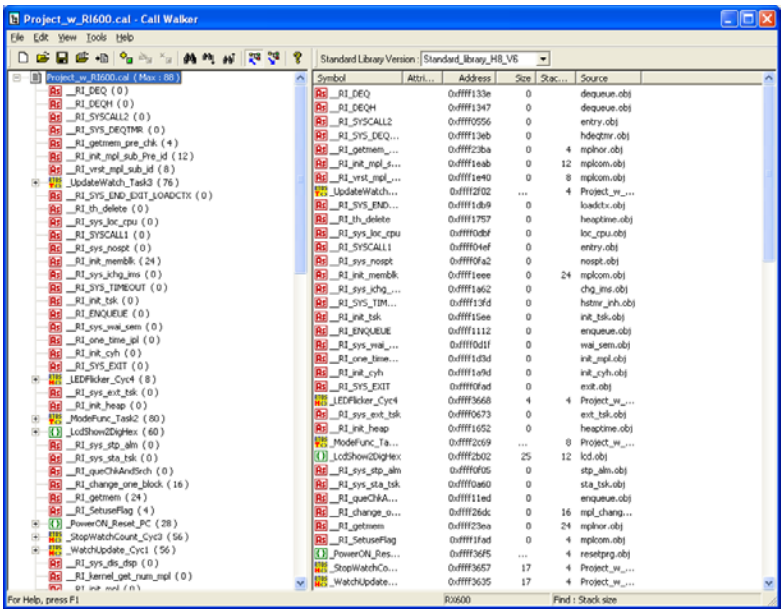

Figure 6 Call Walker with loaded Stack Information File

To identify α of individual task, user may refer to the numerical value append next to the task function name. User may choose to view the required or used size of individual task. Figure 7 shows the required α stack size of task “_Main_Task1 (72)” is 72 bytes.

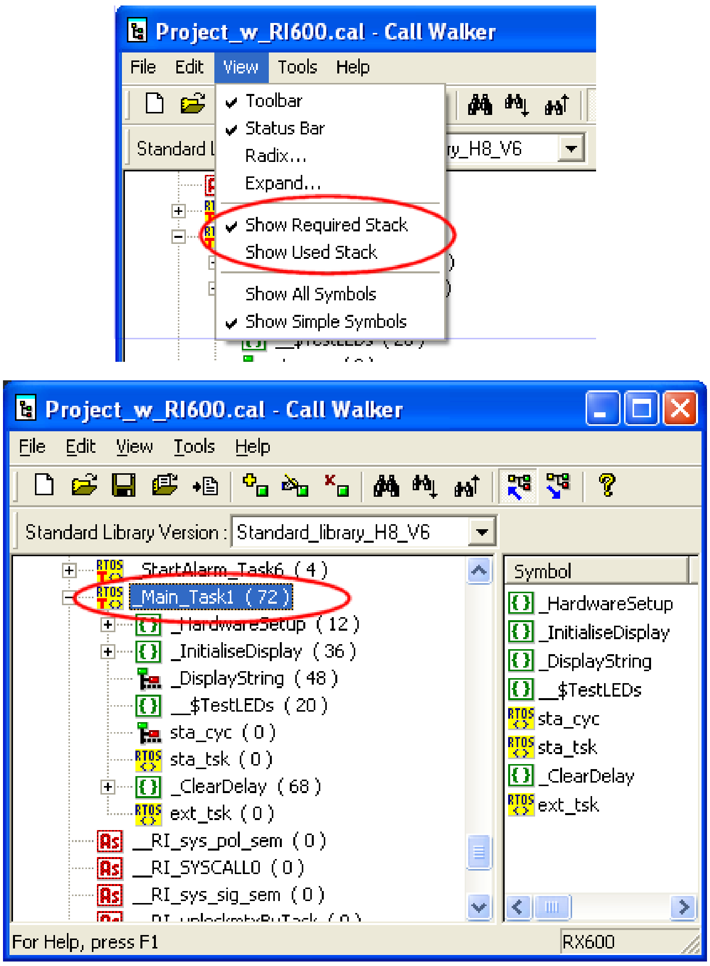

Figure 7 α Stack Size of “Main_Task1”

3.2 Calculating β

β defines the task context size. The size of β is dependent on the system.context defined in the configuration file (Figure 8).

```
system {
        stack_size      = (1) System stack size;
        priority        = (2) Maximum value of task priority;
        system_IPL      = (3) Kernel interrupt mask level;
        message_pri     = (4) Maximum value of message priority;
        tic_deno        = (5) Time tick denominator;
        tic_nume        = (6) Time tick numerator;
        context         = (7) Task context register;
};
```

Figure 8 Defining β in Configuration File

User has eight options to choose from for the system.context entry. And each option will represent a different value of β. Figure 9 shows the respective options and its corresponding task context size.

system.context	Task context size (bytes)
NO	68
FPSW	72
ACC	76
FPSW,ACC	80
MIN	44
MIN,FPSW	48
MIN,ACC	52
MIN,FPSW,ACC	56

Figure 9 System.context and corresponding Task Context Size

Figure 10 shows the definition of task context size to be 80 bytes.

```
// System Definition
system{
    stack_size  = 1024;
    priority    = 10;
    system_IPL  = 4;
    message_pri = 1;
    tic_deno    = 1;
    tic_nume    = 1;
    context     = FPSW,ACC;
};
```

Figure 10 Example of a System.context Definition

The user stack size for task "Main_Task1" can therefore be computed at 152 bytes (72+80). Take note 152 bytes is the minimum size that will be used by task "Main_Task1".

4. Computing System Stack Size

The system stack size can be computed based on the following expression.

$$\text{Necessary size of the system stack} = \alpha + (\Sigma\beta i) + \gamma$$

Figure 11 Formulae for System Stack Size Computation

4.1 Calculating α

α denotes the maximum size among the service calls used. Figure 12 provides a sample table listing of the system stack size for respective service call. The value α depends on the kernel version.

Service Call	System Stack Size ((b).(C))	Service Call	System Stack Size ((b).(C))	Service Call	System Stack Size ((b).(C))	Service Call	System Stack Size ((b).(C))	Service Call	System Stack Size ((b).(C))	Service Call	System Stack Size ((b).(C))
act_tsk	24	sus_tsk	24	iref_flg	24	unl_mtx	56	ref_mpl	24	unl_cpu	24
iact_tsk	24	isus_tsk	24	snd_dtq	36	ref_mtx	24	iref_mpl	24	iunl_cpu	24
can_act	24	rsm_tsk	24	psnd_dtq	32	snd_mbf	44	set_tim	24	dis_dsp	16
ican_act	24	irsm_tsk	24	ipsnd_dtq	52	psnd_mbf	44	iset_tim	24	ena_dsp	24
sta_tsk	24	frsm_tsk	24	tsnd_dtq	40	ipsnd_mbf	60	get_tim	24	sns_ctx	24
ista_tsk	24	ifrsm_tsk	24	fsnd_dtq	32	tsnd_mbf	44	iget_tim	24	sns_loc	24
ext_tsk	64	dly_tsk	24	ifsnd_dtq	52	rcv_mbf	56	sta_cyc	24	sns_dsp	24
ter_tsk	132	sig_sem	44	rcv_dtq	32	prcv_mbf	56	ista_cyc	24	sns_dpn	24
chg_pri	36	isig_sem	60	prcv_dtq	32	trcv_mbf	56	stp_cyc	24	vsta_knl	52
ichg_pri	52	wai_sem	32	iprcv_dtq	52	ref_mbf	24	istp_cyc	24	ivsta_knl	52
get_pri	24	pol_sem	24	trcv_dtq	32	iref_mbf	24	ref_cyc	24	vsys_dwn	16
iget_pri	24	ipol_sem	24	ref_dtq	24	get_mpf	48	iref_cyc	24	ivsys_dwn	16
ref_tsk	28	twai_sem	36	iref_dtq	24	pget_mpf	36	sta_alm	24	chg_ims	28
iref_tsk	28	ref_sem	24	snd_mbx	40	ipget_mpf	36	ista_alm	24	ichg_ims	16
ref_tst	24	iref_sem	24	isnd_mbx	56	tget_mpf	48	stp_alm	24	get_ims	0
iref_tst	24	set_flg	48	rcv_mbx	32	rel_mpf	36	istp_alm	24	iget_ims	4
slp_tsk	24	iset_flg	64	prcv_mbx	28	irel_mpf	52	ref_alm	24	ret_int	32
tslp_tsk	24	clr_flg	24	iprcv_mbx	28	ref_mpf	24	iref_alm	24	ref_ver	24
wup_tsk	40	iclr_flg	24	trcv_mbx	36	iref_mpf	24	rot_rdq	24	iref_ver	24
iwup_tsk	52	wai_flg	44	ref_mbx	24	get_mpl	96	irot_rdq	24	vrst_dtq	48
can_wup	24	pol_flg	24	iref_mbx	24	pget_mpl	112	get_tid	24	vrst_mbx	24
ican_wup	24	ipol_flg	24	loc_mtx	40	ipget_mpl	112	iget_tid	24	vrst_mbf	48
rel_wai	116	twai_flg	48	ploc_mtx	24	tget_mpl	96	loc_cpu	16	vrst_mpf	48
irel_wai	132	ref_flg	24	tloc_mtx	44	rel_mpl	108	iloc_cpu	16	vrst_mpl	68

Figure 12 System Stack Size of RI600/4 Service Calls

4.2 Calculating Σβi

Σβi defines the total size consumed by the interrupts and system clock handlers across the different priority levels. For multiple interrupts in the same priority level, the maximum size among the handlers is chosen. Below depicts a scenario on how to calculate Σβi.

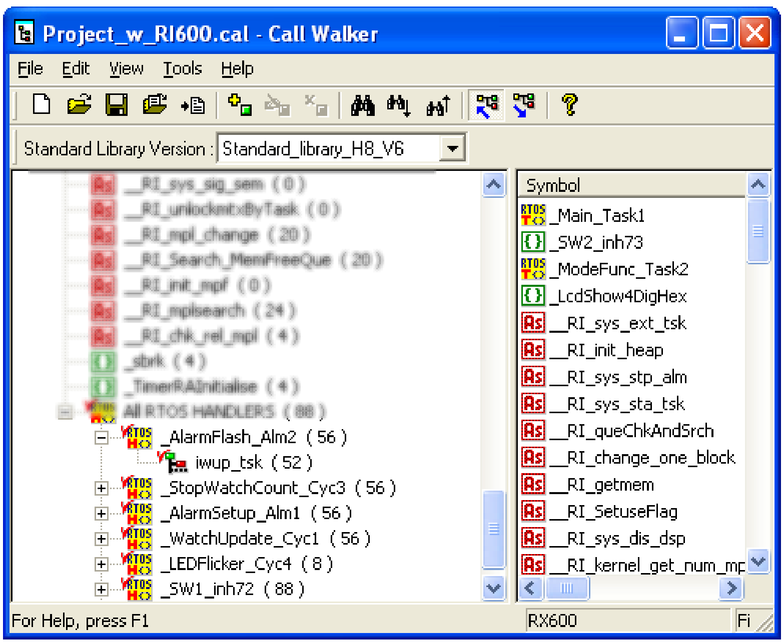

Figure 13 A Scenario of Computing System Stack

RENESAS

As shown in Figure 13, there are

- 2 alarm handlers (i.e. "_AlarmFlash_Alm2" and "_AlarmSetup_Alm1")
- 3 cyclic handlers (i.e. "_StopWatchCount_Cyc3", "_WatchUpdate_Cyc1" and "_LEDFlicker_Cyc4")
- 1 OS-dependent interrupt (i.e. "_SW1_inh72")

Therefore, Σβi can be computed as shown in Figure 14.

Σβi

= stack size of "_SW1_inh72" +
max {ε2 + max (stack size of "_AlarmFlash_Alm2", stack size of "_AlarmSetup_Alm1"),
ε1 + max (stack size of "_StopWatchCount_Cyc3", stack size of "_WatchUpdate_Cyc1",
stack size of "_LEDFlicker_Cyc4")}

= 88 + max {104 + max (56, 56), 104 + max (56, 56, 8)}

= 88 + max {104 + 56, 104 +56}

= 88 + 160

= 248 bytes

Figure 14 Computing Σβi

4.3 Calculating γ

γ denotes the summation of size required by the system down routine and 40bytes. The γ for Figure 15 can be computed to be 44 bytes (4+40). If no system down routine is used, γ is zero.

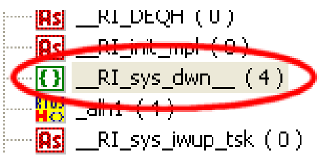

Figure 15 Size of System Down Routine

5. Reference Documents

User's Manual

- RI600/4 V.1.00 User's Manual
- Compiler Package Application Note Call Walker

The latest version can be downloaded from the Renesas Electronics website

Website and Support

Renesas Electronics Website

- http://www.renesas.com/

Inquiries

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Oct.01.10	—	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

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: +49-211-6503-0, 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.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
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 1.0