RX Compiler CC-RX V2.05.00

Release Notes

R20UT3814EJ0100
Rev.1.00
June. 30, 2016

Contents

Chapter 1. Changes

This section describes changes on CC-RX from V2.04.01 to V2.05.00.
The features indicated as [Professional edition] can only be used if the compiler is registered under the professional license.

1.1 Improvements to the feature for checking source code against MISRA-C:2012 rules [Professional edition]

The following rule numbers have been added to the arguments for -misra2012 option, which allows the compiler to check the source code against MISRA-C:2012 rules.

2.6 2.7 9.2 9.3 12.1 12.3 12.4 14.4 15.1 15.2 15.3 15.4 15.5 15.6 15.7
16.1 16.2 16.3 16.4 16.5 16.6 16.7 17.1 17.7 18.4 18.5 19.2 20.1 20.2
20.3 20.4 20.5 20.6 20.7 20.8 20.9 20.10 20.11 20.12 20.13 20.14

1.2 Enhancing the security of dynamic memory management [Professional edition]

A feature for the detection of illegal operations in the releasing of heap space has been added. To create a standard library with this feature enabled, specify the -secure_malloc option and run the library generator.
When this feature is enabled, the standard library functions will work as follows.

(1) As well as the actual areas allocated for users in the heap by the calloc, malloc, and realloc functions, four extra bytes are added before and after each area for the detection of illicit operations.
(2) When called, the free and realloc functions determine if the argument is a pointer to an actual area allocated by calloc, malloc, or realloc, the pointer is to an area that has already been released, or the four-byte area for detecting illicit operations has been overwritten.
(3) If any of the above is the case, an illicit operation is assumed to have proceeded, and _heap_chk_fail will be called.

The __heap_chk_fail function needs to be defined by the user. Write the processing which should be executed when any illicit operation has been detected in the heap space. For example, in the following program, if the string “ABCDEF” is copied from str to the buffer for four letters in the 6th line, the heap space will be corrupted since "EF" and a null character ('\0') overflows from the buffer. In this case, _heap_chk_fail will be executed when the heap space is released at the 8th line.

```
1: #include <string.h>
2: #include <stdlib.h>
3: void func(char *str) {
4:  char *buf;
5:  buf = (*char)malloc(4);
6:  strcpy(buf, str);  // Copy ”ABCDEF” from str
7:  …
8:  free(buf);
9: }
```

By using this feature, you can easily counter security problems through measures against the dual release of memory and against buffers overflowing.

1.3 Added intrinsic functions

1.3.1 New intrinsic functions

We have added the following intrinsic functions for generating bit-manipulation instructions.

- __bclr()
  Generates a BCLR instruction.
- __bset()
  Generates a BSET instruction.
- __bnot()
  Generates a BNOT instruction.

1.3.2 Aliases for existing intrinsic functions

We have added aliases for each of the intrinsic function that have been available on V2.04.01 or earlier versions of CC-RX. Each alias takes the form of "__" (two underscores) preceding the name of the corresponding intrinsic functions*. The alias for the max() function, for example, is __max(). The interface for calling the aliases and the result of expansion is the same as for the corresponding intrinsic functions. In the above example, expanding max(a, b) and __max(a, b) will have the same results. The aliases are usable even when the header file (machine.h) is not included.

* Identifiers with "__" appended are reserved for the compiler and thus should not be used in user programs.

1.4 Eliminated point for caution

Scope of optimization(RXC#038)

1.5 Other improvements

Other improvements are listed below.

(a) Reading and writing of temporary files

The problem of the compiler generating errors when attempting to read from or write to temporary files has been resolved.

(b) Precision of operations by pow() and powf()

The margin of error in the results of operations by pow() and powf() has been reduced.

(c) Internal errors

The problem of the compiler generating internal errors when compiling has been improved.

1.6 Enhanced optimization

For V2.05.00, optimization has been further enhanced on points (a) to (e), listed and described below.

(a) Merging of stack areas allocated for auto arrays in different local scopes (reducing the stack size)

The compiler merges stack areas allocated for auto arrays that belong to different blocks ({}) whose lifetimes do not overlap.

```
< Example of source code >
int *g;
void func01(void){
  {
    int array01[10];
    g = array01;
        func();
  }
  {
    int array02[10];     // Lifetimes of array01[10] and
                         // array02[10] do not overlap.
    g = array02;
  }
}
```

```
< Code generated by V2.04.01>
_func01:
        .STACK  _func01=88      ; Size of allocated
                                ; stack = 88 bytes
        PUSH.L R6
        MOV.L #_g, R6
        ADD #0FFFFFFB0H, R0
        MOV.L R0, R14
        MOV.L R14, [R6]
        BSR _foo
        ADD #28H, R0, R14
        MOV.L R14, [R6]
        RTSD #54H, R6-R6
```

```
< Code generated by V2.05.00>
_func01:
        .STACK  _func01=52      ; Size of allocated
                                ; stack = 52 bytes
        PUSHM R6-R7
        ADD #0FFFFFFD8H, R0
        MOV.L #_g, R6
        MOV.L R0, R7
        MOV.L R7, [R6]
        BSR _foo
        MOV.L R7, [R6]
        RTSD #30H, R6-R7
```

(b) Optimization of constant propagation

Obviously recognizable calculations of constants within loops are omitted.

```
< Example of source code >
j = 1;
k = 2;
l = 3;

for (ix=0; ix<xtra; ix++) {
    for (i=0; i<n4; i++) {
        j = j*(k-j)*(l-k);     // j is always 1
        k = l*k-(l-j)*k;       // k is always 2
        l = (l-k)*(k+j);       // l is always 3
        e1[l-2] = j+k+l;       // l-2 is always 1, j+k+l is always 6.
        e1[k-2] = j*k*l;       // k-2 is always 0, j+k*l is always 6.
    }
}
x = e1[0]+e1[1];
```

```
< Code generated by V2.04.01>
          :
L13:   ; bb47
          CMP R2, R6
          BGE L15
L14:   ; bb3
          SUB R5, R15, R7
               ; j*(k-j)*(l-k),
               ; l*k-(l-j)*k, (l-k)*(k+j),
               ; ; are calculated each time.
          SUB R14, R5, R8
          MUL R7, R14
          MUL R8, R14
          MUL R5, R15, R8
          SUB R14, R15, R7
          MUL R5, R7
          ADD #01H, R6
          SUB R7, R8
          SUB R8, R15
          ADD R14, R8, R5
          MUL R5, R15
          ADD R15, R5
          SHLL #02H, R15, R7
          :
```

```
< Code generated by V2.05.00>
          :
L13:  ; bb46
          CMP R2, R15
          BGE L15
L14:  ; bb2
          ADD #01H, R15
          MOV.L #00000006H, 04H[R3]
              ; 6 is always assigned to e1[1].
          MOV.L #00000006H, [R3]
              ;  6 is always assigned to e1[0].
          BRA L13
          :
```

(c) Optimization of induction variables

The compiler does not generate code for redundantly updating loop induction variables.

```
< Example of source code >
void callee(unsigned i);
void caller(void){
  unsigned i;
  for(i=128; i != 0; --i){
    callee(i);
  }
}
```

```
< Code generated by V2.04.01>
          :
          MOV.L #00000080H, R6
          MOV.L R6, R7              ; Loop induction variable
                                    ; is redundantly initialized.
L11:      ; bb
          MOV.L R7, R1
          BSR _callee
          SUB #01H, R7              ; Loop induction variable
                                    ; is redundantly updated.
          SUB #01H, R6
          BNE L11
          :
```

```
< Code generated by V2.05.00>
          :
          MOV.L #00000080H, R6
L11:  ; bb
          MOV.L R6, R1
          BSR _callee
          SUB #01H, R6
          BNE L11
          :
```

(d) Using min, max, and abs instructions in optimization

The compiler uses min, max, and abs instructions more frequently.

< Example of source code >

```
int min_test(int a) {
  if (a >= 17) {
    a = 17;
  }
  return a;
}
```

< Code generated by V2.04.01>

```
_min_test:
          .STACK   _min_test=4
          CMP #10H, R1
          MOV.L #00000011H, R14
          BGT L12
L11:      ; entry
          MOV.L R1, R14
L12:      ; entry
          MOV.L R14, R1
          RTS
```

< Code generated by V2.05.00>

```
_min_test:
          .STACK   _min_test=4
          MIN #11H, R1      ; A min instruction is used.
          RTS
```

(e) Deleting unused code

The ability to delete unused code has been further enhanced.

```
< Example of source code >
unsigned long test(unsigned long long variable, int var){
  if (var){
    variable &= 0x012345678abcdefULL;
  }
  return (variable >> 32);
}
```

```
< Code generated by V2.04.01>
_test:
          .STACK  _test=4
          CMP #00H, R3
          BEQ L12
L11:      ; if_then_bb
          AND #78ABCDEFH, R1   ; R1 is not referenced
                               ; in the subsequent lines.
          AND #00123456H, R2
L12:      ; if_break_bb
          MOV.L R2, R1
          RTS
```

```
< Code generated by V2.05.00>
_test:
          .STACK  _test=4
          CMP #00H, R3
          BEQ L12
L11:      ; if_then_bb
          AND #00123456H, R2
L12:      ; if_break_bb
          MOV.L R2, R1
          RTS
```

Chapter 2. Points for Caution

This section describes points for caution regarding CC-RX.

2.1 Note on a case of the W052304 message [C/C++ Compiler]

When the int_to_short option is specified and a file including a C standard header is compiled as C++ or EC++, the compiler may show the W052304 message. In this case, simply ignore the message because there are no problems.

[NOTE]

In compilation of C++ or EC++, the int_to_short option will be invalid.

Data that are shared between C and C++ (EC++) program must be declared as the long or short type rather than as the int type.

2.2 Note on using MVTC or POPC instructions [Assembler]

In the assembly language, the program counter (PC) cannot be specified for MVTC or POPC instructions.

2.3 Note on the delete option for linkage [Optimizing linkage editor]

When a function symbol is removed by the delete option, its following function in the source program is not allowed to have a breakpoint at its function name on the editor in your debugging. If you would like to set a breakpointvia the Label window at the function entrance, set the breakpoint via the Label window or at the program code of the function.

Chapter 3. Restrictions

This chapter describes restrictions on CC-RX V.2.05.00.

3.1 Restriction on usage of math.h functions (frexp, ldex, scalbn and remquo) in C++ language (including EC++)

An object is generated which will be an infinit-loop at execution when the actual argument of some function (frexp, ldexp, scalbn or remquo) of math.h is int-type, at compiling C++ or EC++ program.

Conditions:

This problem occurs when both (1) and (2) are satisfied.

(1) This program is in C++ or the lang=cpp option is effective.

(2) math.h is included and any of the following functions is called.

(a) frexp(double, long*) with 'int *' type second argument (except when the first argument is float-type and the dbl_size=8 option is effective).

(b) ldexp(double, long) with 'int *' type second argument (except when the first argument is float-type and the dbl_size=8 option is effective).

(c) scalbn(double, long) with 'int *' type second argument (except when the first argument is float-type and the dbl_size=8 option is effective).

(d) remquo(double, double, long*) with 'int *' type third argument (except when the both the first and second arguments are float-type and the dbl_size=8 option is effective).

Examples:

file.cpp:

```
// Example of compiling C++ source that generates an infinity-loop
#include <math.h>
double d1,d2;
int i;
void func(void)
{
   d2 = frexp(d1, &i);
}
```

Command Line:

```
ccrx -cpu=rx600 -output=src file.cpp
```

file.src: Example of the generated assembly program

```
_func:
   ; ...(Omitted)
    ; Calling substitute function of frexp
   BSR __$frexp__tm__2_f__FZ1ZPi_Q2_21_Real_type__tm__4_Z1Z5_Type
   ; ...(Omitted)

__$frexp__tm__2_f__FZ1ZPi_Q2_21_Real_type__tm__4_Z1Z5_Type:
L11:
   BRA L11 ; Calls itself ==> infinity-loop
```

Countermeasures:

Select one of the following ways to avoid the problem.

(1) Compile the program with the lang=c or lang=c99 option.

(2) Change int or int * into long or long *.

(3) Append the following declarations to each function that is being used.

```
/* For the frexp function */
static inline double frexp(double x, int *y)
{ long v = *y; double d = frexp(x,&v); *y = v; return (d); }
/* For the ldexp function */
static inline double ldexp(double x, int y)
{ long v = y; double d = ldexp(x,v); return (d); }
/* For the scalbn function */
static inline double scalbn(double x, int y)
{ long v = y; double d = scalbn(x,v); return (d); }
/* For the remquo function */
static inline double remquo(double x, double y, int *z)
{ long v = *z; double d = remquo(x,y,&v); *z = v; return (d); }
```

Example of (2):

Change in file.cpp:

```
#include <math.h>
double d1,d2;
int i;
void func(void)
{
  long x = i; /* Accept as long type temporary */
  d2 = frexp(d1, &x); /* Call with long type argument */
  i = x; /* Set the result for variable 'i' */
}
```

Example of (3):

Change in file.cpp:

```
#include <math.h>
/* Append declaration */
static inline double frexp(double x, int *y)
{ long v = *y; double d = frexp(x,&v); *y = v; return (d); }
double d1,d2;
int i;
void func(void)
{
  d2 = frexp(d1, &i);
}
```

3.2 Restriction of PIC/PID function (pic and pid options)

When a standard library is created by the library generator (lbgrx) with the pic or pid option specified, the following warning may appear once or more.

```
W0591301:"-pic" option ignored (When the pic option has been specified)
W0591301:"-pid" option ignored (When the pid option has been specified)
```

Despite the warning, the created standard library has no problems.

3.3 Eliminated options (for the C/C++ compiler)

(a) -file_inline, -file_inline_path

Specifying these options has no effect and the compiler will output a warning. Instead of **-file_inline** or **-file_inline_path**, write **#include** in the source code.

(b) -enable_register

This option is simply ignored and does not affect the generated code.

3.4 Notes on C/C++ source-level debugging (for the C/C++ compiler)

(1) Even when **-debug** is specified, you may not be able to set a breakpoint or stop stepped execution on lines that

contain a dynamic initialization expression for a global variable (in C++),

are the first lines of functions that begin with a loop statement (e.g. **do** or **while**) and do not have an **auto** variable or of functions for which **#pragma inline_asm** has been specified, or

contain the control section and body of a loop statement (e.g. **for**, **while**, or **do**) written as a single line.

(2) The values of members of union type and of dummy variables that are to be passed via registers may be displayed incorrectly (e.g. in the [Watch] window).

3.5 Note on using sections that include the address 0xffffffff (in assembler)

If two or more **.section** directives in the assembly source code contain **.org** directives, the sections have the same name, and the sections overlap at 0xffffffff, the assembler outputs an internal error message (C0554098).

Example)

```
.section SS,ROMDATA
.org 0fffffffeh
.byte 1
.byte 2 ; 0xffffffff
.section SS,ROMDATA
.org 0ffffffffh
.byte 3; ; 0xffffffff
.end
```

3.6 Note on using **-form** and **-output** at the same time (in the linkage editor)

When **-form=rel** and **-output=<filename>** are specified for the linkage editor (**rlink**) at the same time, the filename extension given as **<filename>** is ignored and replaced with **.rel**.

Example)

```
rlink -form=relocate -output=DefaultBuild\lib_test.lib
```

The filename specified for output, **test.lib**, is changed to **test.rel**.

3.7 Note on using function names that begin with _**builtin** (for the C/C++ compiler)

Declaration of a function with a name that begins with _**builtin** and for which the definition is in **machine.h** in the **include** directory may lead to an internal error. In general, do not use any names that begin with an underscore (_) in your source code, since such names are reserved.

3.8 Note on using #**pragma interrupt** with functions for which **save_acc** is enabled and that have dummy arguments (for the C/C++ compiler)

When **#pragma interrupt** is specified for a function and the **save_acc** flag is enabled (including where this is done by using the **-save_acc** compiler option), the compiler may not output code that reflects the correct values of dummy arguments which are passed via R4.Note: In general, we do not recommend defining arguments for functions with the **#pragma interrupt** specification.

Chapter 4. Standard Libraries Included

This chapter describes restrictions on standard libraries included in RX Family C/C++ Compiler.

.

This compiler package includes four library files (*.lib) for the RX600. You can use any of the library files if they correspond to the options that you wish to specify. Using these files shortens the time required for building.

4.1 Library files

Table 1 shows the standard library files and compiler options.

Note:

The compiler options you specify should be the same as the microcontroller options defined for each of the library files listed in table 1. Otherwise these library files are not usable, so specify your compiler options in the library generator to generate your own library file.

Library File	Purposes	Optimize*2 Options	Microcontroller Options *1 *2		
			-endian	-cpu -rtti -exception -noexception	Others *3
rx600lq.lib	For the RX600 Optimization type:Speed Little endian	-speed -goptimize	-endian=little	-cpu=rx600 -rtti=on -exception	-round=nearest -denormalize=off -dbl_size=4 -unsigned_char -unsigned_bitfield -bit_order=right -unpack -fint_register=0 -branch=24
rx600ls.lib	For the RX600 Optimization type:Size Little endian	-size -goptimize			
rx600bq.lib	For the RX600 Optimization type: Speed Big endian	-speed -goptimize	-endian=big		
rx600bs.lib	For the RX600 Optimization type: Size Big endian	-size -goptimize			

Table 1 Library Files

*Notes:

*1 For details on microcontroller options, please see the “Microcontroller Options” columns of the “(1) Compile Options” of “section B.1.3 Options”, in the Integrated Development Environment User’s Manual:RX Build.

*2 These option selections are same from the each default of them.

4.2 Using the library files

The library files included in the compiler package must be linked in either of the ways given in sections 4.2.1 and 4.2.2.

4.2.1 Using the library files

When the e^2 studio has been installed in C:\Renesas\e2_studio, the library files are stored in the following location:

```
C:\Program Files\Renesas\RX\V2_4_0\lib
```

("V2.04.00" indicates the version and revision number of the compiler package.)

4.2.2 Directory specifying a library file in the optimizing linkage editor

Copy the library file(s) included in the package (stored in the location given in section 4.2.1) into a desired directory. Then specify one of the copied library files for the Library option and start the linkage processing.

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 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 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. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0