C/C++ Compiler Package for RX Family V3.08.00

Release Note

Thank you for using our product.

This document describes the restrictions and points for caution. Read this document before using the product.

Contents

1. Related Manuals 2

2. Operating environment 3

3. Changes 4

3.1 Addition of attributes 4

3.1.1 Attribute Specification Formats 4

3.1.2 aligned 5

3.1.3 always_inline 5

3.1.4 naked 6

3.1.5 noinline 6

3.1.6 section 6

3.1.7 weak 7

3.2 Addition of the -linker_script option 7

3.3 Addition of the -library_directory option 11

3.4 Add the __has_include preprocessor directive 12

3.5 Support for the __asm statement 12

3.6 Support for <time.h> 12

3.7 Changes to assembly language naming rules 12

3.8 Improvement to the address directives 12

3.9 Extension of the allowable range for alignment adjustment value 12

3.10 Dependency file output specification options 13

3.11 Improvements to the -whole_program Option 13

3.12 Rectified points for caution 15

3.13 Other Changes and Improvements 15

4. Standard Libraries 16

4.1 Library files 16

4.2 Using the library files 17

Revision History 18

1. Related Manuals

Please refer to the following manuals together with this document.

Name	Document Number
CC-RX Compiler User's Manual	R20UT3248EJ0115
CS+ Integrated Development Environment User's Manual: CC-RX Build Tool Operation	R20UT3478EJ0113

2. Operating environment

The recommended operating environment is as follows.

Windows 10 - 64bit

Windows 11 - 64bit

Ubuntu 22.04 LTS - 64bit

Ubuntu 24.04 LTS - 64bit

3. Changes

This section describes changes to the CC-RX compiler from V3.07.00 to V3.08.00.

3.1 Addition of attributes

The following GCC-compiler attributes have been added:

- aligned
- always_inline
- naked
- noinline
- section
- weak

However, the use of these attributes in C++ source files is not supported.

This feature is not described in the CC-RX Compiler User's Manual(R20UT3248EJ0115). For detailed specifications, please refer to the information below.

3.1.1 Attribute Specification Formats

Attributes can be specified using one of the following formats (1)-(3):

```
(1)  <attribute>[...] <{variable|function|member} declaration>;
(2)  <{variable|function|member} declaration> <attribute>;
(3)  <declaration specifier> (<attribute> <{variable|function|member} declaration>)[, ...];

<attribute>:
   __attribute__((<attribute list>))
<attribute list>:
   <attribute name>[(<attribute arguments>[, ...])][, ...]
```

(1) Declarations that appear after the <attribute> are affected.

(2) Only the declaration immediately preceding the <attribute> is affected.

(3) Only the declaration enclosed together with the <attribute> within the parentheses is affected.

In the example below, __attribute__((section)) applies to var1, var2, var4 and var5, but not to var3 or var6.

```
__attribute__((section("MyBSS"))) int var1, var2;     // Format(1)
int var3, var4 __attribute__((section("MyBSS")));     // Format(2)
int (__attribute__((section("MyBSS"))) var5), var6;   // Format(3)
```

Multiple attributes can be specified simultaneously as long as they do not conflict:

```
__attribute__((noinline)) __attribute__((weak)) void func1();
__attribute__((noinline,weak)) void func2();
__attribute__((noinline)) void func3();
__attribute__((weak)) void func3() { }
```

3.1.2 aligned

[Format]

```
__attribute__((aligned[(<num>)]))
```

[Description]

- Aligns the starting address of the specified variable or member to an address that is a multiple of the integer specified by <num>.
- Applicable targets and values for <num> are as follows.

Applicable targets	Allowed value for <num>
global variable	A power of two not exceeding 4096
static variable	A power of two not exceeding 4096
structure or union member of static variable	A power of two not exceeding 4096
structure or union member of non-static local variable	One of the values 1,2,or 4

- If this is specified multiple times for the same target, the largest <num> value will be applied.
- It is not possible to specify a value smaller than the default alignment of the specified variable or member as <num>. An error will occur.
- When the -stuff option is enabled, global variables, as well as variables and members declared with the static specifier, are placed in sections with the suffix “_X”. The following two points should be noted.
  - When using the linker options -start and -rom, it is necessary to specify sections with the suffix “_X”. If there are other options that specify section names, please ensure they are also properly configured.
  - Depending on the section, initialization in the startup program may be required.
- If <num> is omitted, it is treated as if <num> is specified as 4.

3.1.3 always_inline

[Format]

```
__attribute__((always_inline))
```

[Description]

- The specified function is expanded inline.
- Inline expansion is not performed in the following cases.
  - The specified function takes variable arguments.
  - The specified function is called indirectly via its address.
- __attribute__((always_inline)) does not guarantee inline expansion.
- If specified together with #pragma noinline, __attribute__((always_inline)) takes precedence.
- If specified together with __attribute__((noinline)), __attribute__((noinline)) takes precedence.
- The following #pragma directives cannot be specified together; an error will occur:
  - interrupt
  - inline_asm
  - entry
  - stack_protector

- __attribute__((weak)) cannot be specified together; an error will occur.

3.1.4 naked

[Format]

```
__attribute__((naked))
```

[Description]

- Suppresses generation of function prologue and epilogue code for the specified function.
- The function body must consist only of asm statements. Behavior is not guaranteed if anything other than asm statements is used.
- The following #pragma directives cannot be specified together; an error will occur:
  - inline_asm
  - stack_protector
- __attribute__((always_inline)) cannot be specified together; an error will occur.

3.1.5 noinline

[Format]

```
__attribute__((noinline))
```

[Description]

- Prevents the specified function from being expanded inline, regardless of compiler options or optimization levels.
- If specified together with #pragma inline, __attribute__((noinline)) takes precedence.
- If specified together with __attribute__((always_inline)), __attribute__((noinline)) takes precedence.
- #pragma inline_asm cannot be specified together; an error will occur.

3.1.6 section

[Format]

```
__attribute__((section("<section name>")))
```

[Description]

- Places the specified variable or function into the section named <section name>.
- If multiple section names are specified for the same variable or function, the last specified section name is used.
- <section name> must conform to the rules for valid section names in assembly language files. Otherwise, an assembly error will occur.
- If specified together with #pragma section, __attribute__((section)) takes precedence.

3.1.7 weak

[Format]

```
__attribute__((weak))
```

[Description]

- When specified for a variable or function definition, a weak definition is generated.
  When specified for a variable or function declaration, the reference is treated as a weak reference.
  A weak definition and a weak reference differ from normal definitions and references as follows:
  - Weak definition:
    At link time, a normal definition with the same name may exist in another module. In that case, the linker does not issue an error; the normal definition is used and the weak definition is ignored.
  - Weak reference:
    At link time, even if the referenced variable or function definition does not exist, the linker does not issue an error; the reference is treated as NULL.
- #pragma inline cannot be specified together; an error will occur.
- __attribute__((always_inline)) cannot be specified together; an error will occur.

3.2 Addition of the -linker_script option

A linker script based on the GNU linker script format can now be used during linking. The linker script is specified by optimizing linkage editor (rlink) option -linker_script.

This feature is not described in the CC-RX Compiler User's Manual(R20UT3248EJ0115). For detailed specifications, please refer to the information below.

[Format]

```
-linker_script=<file>
```

[Description]

- Specifies a linker script file.
- The constructs (such as keywords and syntax elements) supported by rlink are as follows.
  If unsupported constructs are used, unexpected behavior may occur.
  Refer to the GNU linker manual for details of each item.
  - MEMORY command
    The MEMORY command in the following format is supported.

```
MEMORY
{
   name [(attr)]:ORIGIN=address,LENGTH=size
   ...
}
```

name specifies the region (memory area) name, *address* specifies the start address, *size*specifies the size.

(*attr*) is optional. The *attr* can be specified as 'r' (read-only), 'w' (read/write), or 'x' (executable).

- SECTIONS command
  The SECTIONS command in the following format is supported.

```
SECTIONS
{
   sections-statement
   sections-statement
   ...
}
```

The following can be described in *sections-statement*

- ENTRY command
- Symbol definition (assignment to symbols)
  - Simple symbol definition in the form: *symbol* = *expression*
  - Symbol definition using keywords: HIDDEN, PROVIDE, PROVIDE_HIDDEN
- Data declarations (Supported keywords: BYTE, SHORT, LONG)
- Output section description in the following format:

```
section_name [address] [(type)] : [AT(load_address)] [ALIGN(section_align)]
{
   output-section-statement
   output-section-statement
   ...
} [>region] [AT>load_region] [=fill]
```

  - *section_name* specifies the output section name
  - *address* specifies the execution-time address (virtual address)
  - *type* can be NOLOAD, READONLY, TYPE=SHT_PROGBITS or TYPE=SHT_NOBITS
  - *load_address* specifies the load-time address
  - *section_align* specifies the execution memory region
  - *load_region* specifies the load memory region
  - *fill* specifies the fill value for unused space
  - Items enclosed in [] are optional
  - The following can be written in *output-section-statement*
    - Symbol definition in the same format as *sections_statement*
    - Input section description in the following format:

```
input_file_name(input_section_name)
```

      Wildcard characters and the following keywords are supported:
      SOFT_BY_NAME, SORT (same as SORT_BY_NAME), EXCLUDE_FILE, KEEP

- Other Commands

  The following commands are supported:

  - ENTRY
  - INCLUDE
  - GROUP
  - OUTPUT
  - OUTPUT_FORMAT (elf32-rx-be or elf32-rx-le)
  - TARGET (same as OUTPUT_FORMAT)
  - SEARCH_DIR
  - STARTUP
  - ASSERT
  - PROVIDE
  - HIDDEN
  - PROVIDE_HIDDEN

- Builtin Functions

  The following builtin functions are supported:

  - ABSOLUTE
  - ADDR
  - ALIGN
  - DEFINED
  - LENGTH
  - LOADADDR
  - MAX
  - MIN
  - ORIGIN
  - SIZEOF

[Remarks]

- If a linker script file is recursively specified using INCLUDE, a message is displayed and processing terminates.
- This option is invalid when form={object | relocate | library} option or strip option is specified.
- Section placement specified by this option and the start option is interpreted in the order specified.

- This option and library_directory option are interpreted in the order specified. Therefore, if the folder containing the linker script file is specified with library_directory option, specify it before this option.
- BYTE, SHORT, LONG always insert data in little-endian format.

[Note]
The following differences exist compared to the GNU linker:

- rlink behaves as if GROUP is specified for all libraries, regardless of explicit GROUP specification.
- If OUTPUT is specified multiple times, multiple output files are generated.
- OUTPUT_FORMAT and TARGET are accepted but have no effect.
- Input sections without output section specification are placed at the end.
- ADDR returns an absolute value, not section-relative.

[Messages]

The following table lists the messages displayed when using the linker script.

Number	Message	Explanation
M0560800	Detailed error information	Detailed error information for syntax analysis of the linker script.
W0561703	Duplicate definition of “*name*”	The definition of "*name*" is duplicated in the linker script. The last definition specified is used.
W0561705	Invalid alignment value	The specified alignment value in the linker script is invalid.
E0562023	Duplicate file specified “*file*”	The description of "*file*" is duplicated in the linker script.
E0562115	Linker script specified section cannot be specified in “*section*” option : “*option*”	The section "*section*" specified in the linker script cannot be used with the option "*option*".
E0562312	Undefined section symbol “*section*” referenced in “*file*”	The section symbol "*section*" is referenced in the file "*file*", but the corresponding section does not exist.
E0562327	Section “*section1*” overlaps section “*section2*” in load memory	The load memory addresses (LMA) of sections specified in the linker script overlap.
E0562700	ASSERT:”*character string*”	The condition specified by the ASSERT command in the linker script is not satisfied, so the message "character string" is displayed.
E0562701	Syntax error	There is a syntax error in the linker script.
E0562702	Syntax error	There is a syntax error in the linker script.
E0562703	Nested Include file “*file*”	The INCLUDE command in the linker script directly or indirectly includes the same file. Review the include hierarchy to remove the cyclic inclusion.
E0562705	Division or modulo by zero	A division or modulo operation by zero is performed.

Number	Message	Explanation
E0562706	Undefined format “*format*”	The OUTPUT_FORMAT or TARGET specification in the linker script is invalid.Specify one of the following values: - elf32-rx-be - elf32-rx-le
E0562707	Undefined symbol “*symbo*l”	The symbol "*symbol*" specified in the linker script is not defined.
E0562708	Undefined section “*section*”	The section "*section*" specified in the linker script is not defined.
E0562709	Undefined memory “*memory*”	The memory region "*memory*" specified in the linker script is not defined.
E0562711	Memory definition out of range “*memory*”	The placement in the memory region "*memory*" specified in the linker script exceeds the region size.
E0562712	Section “section” unfit in memory “*memory*” range	The section "*section*" specified in the linker script does not fit in the memory region "*memory*".
E0562714	Cannot move location counter backwards	The location counter cannot be moved backwards.
E0562715	Section “*section*” conflicts with option “*option1*” and option “*option2*”	The section "*section*" conflicts with option "*option1*" and option "*option2*".
E0562717	Reserved section name in SECTIONS : "*section*"	The section name "*section*" specified in the SECTIONS command in the linker script is reserved. Specify a different name.

3.3 Addition of the -library_directory option

Optimizing linkage editor (rlink) option -library_directory has been added to specify folders containing library files.

This feature is not described in the CC-RX Compiler User’s Manual(R20UT3248EJ0115). For detailed specifications, please refer to the information below.

[Format]

```
-library_director=<folder>
```

[Description]

- Adds folders where the following are located:
  - Library files
  - Relocatable files
  - Object files
  - Linker script files
- Files are searched in the following order:
  - Current folder

- Folders specified by this option
- Folders specified by environment variable HLINK_DIR

- Specify this option before the target files.

[Examples]

Inputs a.lib located in \libdir.

```
> rlink main.obj -library_directory=\libdir -library=a.lib
```

3.4 Add the __has_include preprocessor directive

The __has_include preprocessor directive allows users to determine whether a specified header file exists.

It returns 1 if the header exists, and 0 otherwise.

3.5 Support for the __asm statement

The __asm__ statement enables inline assembly similar to that supported by GCC.

Only the basic syntax is supported. Extended syntax is not supported.

The qualifiers volatile and inline are accepted for compatibility with GCC, but they have no special effect.

The statement can be used only within functions. Using it outside a function results in an error.

3.6 Support for <time.h>

Partial support for the standard header <time.h> has been added, including types and macros.

3.7 Changes to assembly language naming rules

Support for using periods(.) in names has been added.

Periods(.) can now be used in symbol names, section names, and other identifiers.

3.8 Improvement to the address directives

In the assembly language, for address directives (.ALIGN, .ORG, .OFFSET), the value written to padding areas generated by address alignment was previously fixed to NOP(03H). A parameters has been added to allow users to specify an arbitrary value.

3.9 Extension of the allowable range for alignment adjustment value

The allowable range for alignment values specified in the address directives (.ALIGN) and the link directives (.SECTION) in the assembly language has been expanded to powers of two from 2 to 65536.

3.10 Dependency file output specification options

Added an option to output dependencies between source files and include files to a dependency file.

compile options: -output=dep, -MM, -MP, -MT

assemble options: -MM, -MP, -MT, -MF

In addition, the following options have been added to improve the usability of the dependency file.

compile options: -MD, -MMD, -MF

assemble options: -MMD

The generated dependency file can be included in a Makefile for use.

For details on the options, please refer to the User's Manual.

3.11 Improvements to the -whole_program Option

Optimization on the assumption that the whole program consists of the input file has been enhanced.

When all input files are merged, the following optimizations for global variables are performed.

- If a global variable that has not been qualified as const in never subject to writing, it is optimized in the same way as a variable that is qualified as const.
- A global variable that has not been qualified as static is optimized in the same way as a variable that is qualified as static.

Example of source code (test.c)

```
unsigned char globalIndex;
unsigned char neverReadArray[10];
void
autoVariableReplacementAndStoreEliminationExample(void){
  for(globalIndex = 0; globalIndex < 10; ++globalIndex){
    neverReadArray[globalIndex] = globalIndex;
  }
}

int neverOverwrittenVariable = 0;
int* writeOncePointer;
void constantPropagationExample0(void){
  writeOncePointer = &neverOverwrittenVariable;
}
int constantPropagationExample1(void){
  return (*writeOncePointer);
}
```

Example of generated code (ccrx -isa=rxv3 -whole_program -output=src test.c)

<V3.07.00>

```
_autoVariableReplacementAndStoreEliminationExample:

        MOV.L #00000000H, R14
        MOV.L #_neverReadArray, R15
        MOV.L R14, R5
L11:    ; bb6
        CMP #0AH, R5
        BEQ L13
L12:    ; bb
        MOVU.B R5, R4
        ADD #01H, R5
        MOV.B R14, [R15,R4]
        ADD #01H, R14
        BRA L11
L13:    ; return
        MOV.L #_globalIndex, R14
        MOV.B #0AH, [R14]
        RTS
_constantPropagationExample0:
        .STACK  _constantPropagationExample0=4
        MOV.L #_writeOncePointer, R14
        MOV.L #_neverOverwrittenVariable, R15
        MOV.L R15, [R14]
        RTS
_constantPropagationExample1:
        .STACK  _constantPropagationExample1=4
        MOV.L #_writeOncePointer, R14
        MOV.L [R14], R14
        MOV.L [R14], R1
        RTS
```

<V3.08.00>

```
_autoVariableReplacementAndStoreEliminationExample:
        .STACK _autoVariableReplacementAndStoreEliminationExample=4
        MOV.L #0000000AH, R14
L11:    ; bb
        SUB #01H, R14
        BNE L11
L12:    ; return
        RTS
_constantPropagationExample0:
        .STACK  _constantPropagationExample0=4
        RTS
_constantPropagationExample1:
        .STACK  _constantPropagationExample1=4
        MOV.L #00000000H, R1
        RTS
```

3.12 Rectified points for caution

The following points for caution no longer apply. For details, refer to Tool News.

- Note on Using Designators for Anonymous Unions or Structures (No.69)
- Note on Using the -avoid_cross_boundary_prefetch Option (No.70)

3.13 Other Changes and Improvements

The following changes and improvements have been made:

- Rectification of Linker

  In V3.07.00, Error C0564001 occurred when linking .rel files. This issue has been fixed.

- Improvement of internal errors.

  An internal error could occur during the build process. This issue has been addressed.

4. Standard Libraries

This chapter describes restrictions on standard libraries included in the 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 4.1 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 4.1. Otherwise these library files are not usable, so specify your compiler options in the library generator to generate your own library file.

Table 4.1 Library Files

Library File	**Purposes**	**Optimize Options**	**Microcontroller Options** *1		
			-endian	**-cpu** **-rtti** **-exception** **-noexception**	**Others** *2
rx600lq.lib	For use with RX600 MCUs Priority in optimization: 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 use with RX600 MCUs Priority in optimization: Size Little endian	-size -goptimize			
rx600bq.lib	For use with RX600 MCUs Priority in optimization: Speed Big endian	-speed -goptimize	-endian=big		
rx600bs.lib	For use with RX600 MCUs Priority in optimization: Size Big endian	-size -goptimize			

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

2. The listed option settings produce the same behavior as the default settings.

4.2 Using the library files

Copy the library file(s) included in the package from the "lib" directory into a desired directory.

Then specify one of the copied library files for the **-library** option and start the linkage processing.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.01.26	-	Newly created.

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

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/.