Integrated Development Environment e² studio

Using CMake with Renesas CC-RX compiler

Introduction

CMake's responsibility is to generate native build tool files from the platform and compiler independent configuration files named CMakeLists.txt.

This document describes how to create the necessary build environment to execute the Build for C/C++ Compiler Package for RX Family (CC-RX) using CMake on a Windows host PC.

Contents

1. Overview

CMake's responsibility is to generate native build tool(1) files from the platform and compiler independent configuration files named CMakeLists.txt.

(1) The native build tool is the real tool used to build your software. Examples of native build tools:

- Xcode
- Visual Studio
- Ninja
- Make

The specifications of the compiler options, that is, the setting items related to build, and the format of the configuration file that stores them are not unified. In other words, it is necessary to use the project file properly according to the development environment used.

CMake can create any project file that CMake supports by creating a configuration file called CMakeLists.txt. Therefore, in any development environment:

1. Generate an arbitrary project file with CMake.
2. Execute build using the generated project file.

You can build in 2 steps.

This document describes how to create the necessary build environment the execute the Build for C/C++ Compiler Package for RX Family (CC-RX) using CMake on a Windows host PC.

For the C Compiler Package for RL78 Family (CC-RL project), there are differences in the build options, but you can construct the build environment using the same procedure.

2. How to install

Please download the following tools from the website and install them.

- Renesas compiler for CC-RX:
  https://www.renesas.com/software-tool/cc-compiler-package-rx-family#downloads
  (version 3.03.00 used in this example)
- CMake:
  Download | CMake
  (version 3.20.3 used in this example)
- GNU Make for windows (If you already installed e^2 studio, you can use "make" in e^2 studio.):
  Make for Windows (sourceforge.net)
  (version 3.81 used in this example)

As a next step, set the following environment variables so that you can run CC-RX, CMake and make on the command prompt.

1. Add the following folders to the environment variable PATH.
   - <Install folder of CC-RX>\bin
   - <Install folder of CMake>\bin
   - <Install folder of GNU make>
2. Next, add the following environment variables that are required to run CC-RX.
   - BIN_RX
     Environment variable name: BIN_RX
     Value: <Install folder of CC-RX>\bin
   - INC_RX
     Environment variables name: INC_RX
     Value: <Install folder of CC-RX>\include

3. File structure

The file structure required to construct a build environment for Build using CMake is as follows. You need to construct a project based on the following example.

```
<project folder>
  \ccrx.cmake              ; Toolchain file for CC-RX
  \CMakeLists.txt          ; CMake configuration file
  \Linker.tmp              ; Subcommand file for Linker
  \generate                ; Folder of source files (You can construct any
     \dbsct.c              folder hierarchy for your source files.)
     \hwsetup.c
     \intprg.c
     \iodefine.h
     \lowlvl.src
     \lowsrc.c
     \lowsrc.h
     \resetprg.c
     \sbrk.c
     \sbrk.h
     \stacksct.h
     \test.h
     \typedefine.h
     \vect.h
     \vecttbl.c
  \src                     ; Folder of source files (You can construct any
     \main.c               folder hierarchy for your source files.)
  \_builds                 ; make execution folder (makefile will be generated
                           in this folder that CMake creates.)
```

Please create the following files in any editor to run CMake.

- Toolchain file for CC-RX
  ccrx.cmake
- CMake configuration file
  CMakeLists.txt
- Subcommand file for Linker
  Linker.tmp

The contents of the above files are described in more detail in the next chapters.

4. Required files to run CMake

4.1 CMake Toolchain file for CC-RX

This file is used to specify information about the compiler and utility paths necessary for cross compiling using the CC-RX toolchain. You need to create a file based on the following contents.

#	ccrx.cmake
1	### BEGIN CMAKE_TOOLCHAIN_FILE
2	# "Generic" is used when cross compiling
3	set(CMAKE_SYSTEM_NAME Generic)
4	# Optional, this one not so much
5	set(CMAKE_SYSTEM_VERSION 1)
6	# Set the environment root directory
7	# It can be used to specify the target environment location
8	# e.g compiler's location. This variable is most useful when cross-compiling.
9	set(CMAKE_FIND_ROOT_PATH "C:/Program Files (x86)/Renesas/RX/3_3_0/bin")
10	# CMake variables for compiler, assembler, native build system
11	# Specify the C compiler
12	set(CMAKE_C_COMPILER ccrx)
13	set(CMAKE_C_COMPILER_ID RXC)
14	set(CMAKE_C_COMPILER_ID_RUN TRUE)
15	set(CMAKE_C_COMPILER_FORCED TRUE)
16	# Specify the CPP compiler
17	set(CMAKE_CXX_COMPILER ccrx)
18	set(CMAKE_CXX_COMPILER_ID RXC)
19	set(CMAKE_CXX_COMPILER_ID_RUN TRUE)
20	set(CMAKE_CXX_COMPILER_FORCED TRUE)
21	# Specify the ASM compiler
22	set(CMAKE_ASM_COMPILER asrx)
23	set(CMAKE_ASM_COMPILER_ID RXA)
24	set(CMAKE_ASM_COMPILER_ID_RUN TRUE)
25	set(CMAKE_ASM_COMPILER_FORCED TRUE)
26	# Specify the linker
27	set(CMAKE_LINKER rlink)
28	# Specify options for command
29	set(COMMON_OPTIONS "-isa=rxv2 -fpu")
30	set(CMAKE_C_FLAGS "${COMMON_OPTIONS} -lang=c -include=\"../generate\" -define=DEBUG_CONSOLE -utf8 -outcode=utf8 -nomessage -debug -nologo")
31	set(CMAKE_CXX_FLAGS "${COMMON_OPTIONS} -lang=cpp -include=\"../generate\" -define=DEBUG_CONSOLE -utf8 -outcode=utf8 -nomessage -debug -nologo")
32	set(CMAKE_ASM_FLAGS "${COMMON_OPTIONS} -include=\"../generate\" -utf8 -debug -nologo")
33	set(CMAKE_EXE_LINKER_FLAGS "-subcommand=../Linker.tmp")
34	# Specify the make
35	set(CMAKE_MAKE_PROGRAM make)
36	### END CMAKE_TOOLCHAIN_FILE

- #9: Edit this line to specify "<Install folder of CC-RX>\bin" according to your environment.
- #29: Edit this line to specify an option of Compiler/Assembler according to your environment.
- #30-31: Edit this line to specify an option of Compiler according to your environment.
- #32: Edit this line to specify an option of Assembler according to your environment.
- #33: Create Linker.tmp as subcommand file of Linker. (The contents of this file will be discussed in a later chapter.)

If you specify a double-quote (") character for an option parameter, please add the '\' character before the double-quote character.

4.2 CMake Configuration file (CMakeLists.txt)

CMake Language commands are ready by cmake from a file named CMakeLists.txt. This file specifies the source files and build parameters, which cmake will place in the project's build specification (in this case a Makefile).

By specifying assembler, compiler and linker directives in this file, object files will be generated and linked to create a ".abs" file.

Create a file, named CMakeLists.txt, with the following contents.

#	CMakeLists.txt
1	cmake_minimum_required(VERSION 3.16)
2	# set the project name
3	project(CCRX_Project)
4	enable_language(C ASM)
5	########## Specify the C/C++/ASM compiler and linker commands
6	set(CMAKE_C_COMPILE_OBJECT "\"${CMAKE_C_COMPILER}\" ${CMAKE_C_FLAGS} -output=obj=<OBJECT> <SOURCE>")
7	set(CMAKE_CXX_COMPILE_OBJECT "\"${CMAKE_CXX_COMPILER}\" ${CMAKE_CXX_FLAGS} -output=obj=<OBJECT> <SOURCE>")
8	set(CMAKE_ASM_COMPILE_OBJECT "\"${CMAKE_ASM_COMPILER}\" ${CMAKE_ASM_FLAGS} -output=<OBJECT> <SOURCE>")
9	set(CMAKE_C_LINK_EXECUTABLE "\"${CMAKE_LINKER}\" ${CMAKE_EXE_LINKER_FLAGS} <OBJECTS> -output=<TARGET>.abs")
10	set(CMAKE_CXX_LINK_EXECUTABLE "\"${CMAKE_LINKER}\" ${CMAKE_EXE_LINKER_FLAGS} <OBJECTS> -output=<TARGET>.abs")
11	########## Set C/C++ Source language
12	file(GLOB_RECURSE SOURCE ${CMAKE_CURRENT_SOURCE_DIR} "*.cpp" "*.c")
13	########## Set ASM Source language
14	file(GLOB_RECURSE src_asm ${CMAKE_CURRENT_SOURCE_DIR} "*.asm" "*.s" "*.src")
15	foreach(X IN ITEMS ${src_asm})
16	set_source_files_properties(${X} PROPERTIES LANGUAGE ASM)
17	endforeach()
18	########## Add function to scan and generate dependency files
19	function(dep_scan out_objects)
20	string(REPLACE " -" ";-" DEPENDENCY_C_OPTIONS ${CMAKE_C_FLAGS})
21	string(REPLACE " -" ";-" DEPENDENCY_CPP_OPTIONS ${CMAKE_CXX_FLAGS})
22	string(REPLACE " -" ";-" DEPENDENCY_ASM_OPTIONS ${CMAKE_ASM_FLAGS})
23	set(result)
24	foreach(in_f ${ARGN})
25	get_filename_component(ext ${in_f} LAST_EXT)
26	file(RELATIVE_PATH rel_f ${CMAKE_CURRENT_SOURCE_DIR} ${in_f})
27	set(depend_f "CMakeFiles/${PROJECT_NAME}.dir/${rel_f}.d")
28	set(obj_f "CMakeFiles/${PROJECT_NAME}.dir/${rel_f}.obj")
29	if(${ext} STREQUAL ".c")
30	add_custom_command (
31	COMMENT "Generate dependency: ${depend_f}"
32	OUTPUT ${depend_f}
33	COMMAND ${CMAKE_C_COMPILER} ${DEPENDENCY_C_OPTIONS} -MM -MP -output=dep=${depend_f} -MT=${obj_f} -MT=${depend_f} ${in_f}
34	DEPENDS "${in_f}"
35	VERBATIM
36	)
37	elseif(${ext} STREQUAL ".cpp")
38	add_custom_command (
39	COMMENT "Generate dependency: ${depend_f}"
40	OUTPUT ${depend_f}
41	COMMAND ${CMAKE_CXX_COMPILER} ${DEPENDENCY_CPP_OPTIONS} -MM -MP -output=dep=${depend_f} -MT=${obj_f} -MT=${depend_f} ${in_f}

```
42          DEPENDS "${in_f}"
43          VERBATIM
44        )
45      elseif((${ext} STREQUAL ".asm") OR (${ext} STREQUAL ".s") OR (${ext} STREQUAL ".src"))
46        add_custom_command (
47          COMMENT "Generate dependency: ${depend_f}"
48          OUTPUT ${depend_f}
49          COMMAND ${CMAKE_ASM_COMPILER} ${DEPENDENCY_ASM_OPTIONS} -MM -MP -output=${depend_f
    } -MF=${depend_f} -MT=${obj_f} -MT=${depend_f} ${in_f}
50          DEPENDS "${in_f}"
51          VERBATIM
52        )
53      endif()
54      list(APPEND result ${depend_f})
55    endforeach()
56    set(${out_objects} "${result}" PARENT_SCOPE)
57  endfunction()
58  ########## Call dep_scan
59  dep_scan(d_files ${SOURCE} ${src_asm})
60  ########## Add the executable
61  add_executable(${PROJECT_NAME} ${SOURCE} ${src_asm} ${d_files})
62  ########## Execute Library Generator
63  # Command rule for build .lib file
64  set(RENESAS_LIBRARY_GENERATOR "lbgrx.exe")
65  string(REPLACE " -" ";-" LIBRARY_GENERATOR_FLAGS "${COMMON_OPTIONS} -lang=c -head=runtime,
    stdio,stdlib,string,new -nologo")
66  add_custom_command(
67    TARGET ${PROJECT_NAME}
68    PRE_LINK
69    COMMAND ${RENESAS_LIBRARY_GENERATOR} ${LIBRARY_GENERATOR_FLAGS} -output=${PROJECT_NAME}.
    lib
70    COMMENT "Library Generator:"
71    VERBATIM
72  )
73  ########## Build finished
74  add_custom_command(
75    TARGET ${PROJECT_NAME}
76    COMMAND ${CMAKE_COMMAND} -E cmake_echo_color --cyan "Build Finished."
77  )
78  ########## Include dependencies files
79  # This process should be put at the end
80  foreach(in_f ${d_files})
81    set(include_command ${include_command}\n-include ${in_f})
82  endforeach()
83  add_custom_command(
84    TARGET ${PROJECT_NAME}
85    COMMAND ${include_command}
86    VERBATIM
87  )
```

- #3: The role of this processing is to specify project name. The project name specified here will be the output file name. Please edit it as necessary.
- #11-17: The role of this processing is to collect source files (file extension: .c/.cpp/.asm/.s/.src) to include for Build. Please place your source files to include for Build in the folder containing this file (CMakeLists.txt) or its hierarchical folder.
- #18-59: The role of this processing is to generate the dependency scan file for each source file. (No need to edit)

- #62-72: The role of this processing is to execute Library Generator. Please edit #65 as options of Library Generator for your environment.
- #78-87: The role of this processing is to include dependency scan files in makefile. It is the trick processing, and it is necessary to always place this processing at the end of file. If you want to add other processing, please add it before this processing.

4.3 Generate MOT file

To create a mot file from the generated abs file, you need to add the following processing before #73 or later processing to the configuration file.

#	Add the following processing to CMakeLists.txt
1	########## Execute Converter
2	# Command rule for build .mot file from .abs file
3	set(CONVERTER_FLAGS -form=stype -nologo -nomessage)
4	add_custom_command(
5	TARGET ${PROJECT_NAME}
6	POST_BUILD
7	COMMAND ${CMAKE_LINKER} ${PROJECT_NAME}.abs ${CONVERTER_FLAGS}
8	COMMENT "Converter:"
9	VERBATIM
10	)

- #3: Edit this line to an option for Converter according to your environment. If you want to output the other format, please modify "-form" option.

4.4 Generate x file

To debug on e² studio, you need to generate x file from abs file. You need to add the following processing before #73 or later processing to the configuration file.

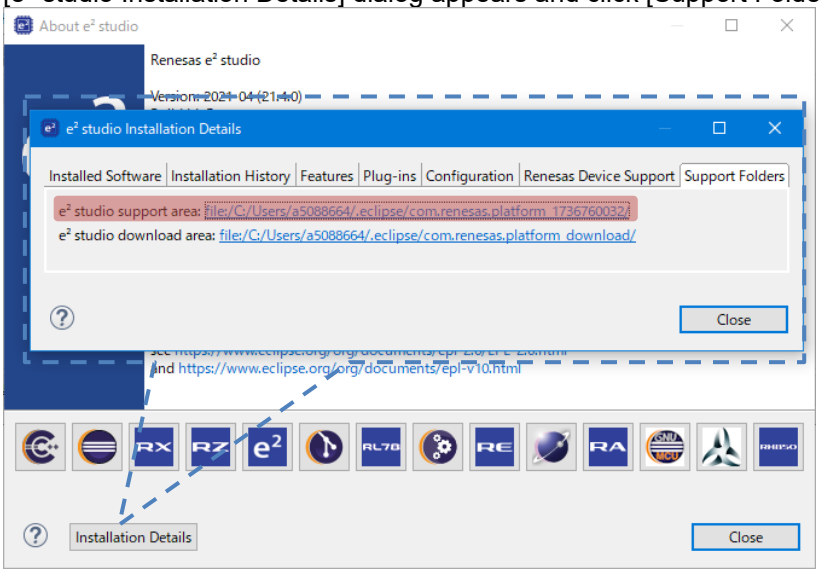
#	Add the following processing to CMakeLists.txt
1	########## Execute X Converter
2	# Command rule for build .x file from .abs file
3	set(RENESAS_XCONVERTER renesas_cc_converter.exe)
	add_custom_command(
4	TARGET ${PROJECT_NAME}
5	POST_BUILD
6	COMMAND ${RENESAS_XCONVERTER} ${PROJECT_NAME}.abs ${PROJECT_NAME}.x
7	COMMENT "X Converter:"
8	VERBATIM
9	)

- #3: This program can be found in the "\Utilities\ccrx" under the e² studio support folder. Add that folder to the environment variable PATH.

You can check the support folder of e² studio via the following procedure.

1. Select [Help] > [About e² studio] menu on e² studio.
2. [About e² studio] dialog appears and click [Installation Details] button.

3. [e² studio Installation Details] dialog appears and click [Support Folders] tab.



4.5 Subcommand file for Linker

This file is to specify options for Linker. You need to create a file for Linker options based on the following contents.

#	Linker.tmp
1	-noprelink
2	-start=SU,SI,B_1,R_1,B_2,R_2,B,R/04,PResetPRG,C_1,C_2,C,C$*,D*,W*,L,PIntPRG,P/0FFE00000,EXCEPTVECT/0FFFFFF80,RESETVECT/0FFFFFFC
3	-form=absolute
4	-nomessage
5	-list=CCRX_Project.map
6	-nooptimize
7	-rom=D=R,D_1=R_1,D_2=R_2
8	-cpu=RAM=00000000-0003ffff,FIX=00080000-00083fff,FIX=00086000-00087fff,FIX=00088000-0009ffff,FIX=000a0000-000a3fff,RAM=000a4000-000a5fff,FIX=000a6000-000bffff,FIX=000c0000-000dffff,FIX=000e0000-000fffff,ROM=00100000-00107fff,FIX=007fc000-007fcfff,FIX=007fe000-007fffff,RAM=00800000-0085ffff,RAM=fe7f5d00-fe7f5d7f,RAM=fe7f7d70-fe7f7d9f,ROM=ffe00000-ffffffff
9	-nologo
10	-library=CCRX_Project.lib

5. Build

Building a program using CMake is a two stages process.

Stage 1. invokes CMake to generate standard build files (Makefiles in this case) using the CMake configuration file (CMakeLists.txt) and CMake toolchain file.

Stage 2. invokes the platform's native build tools (native toolchain) to actually build the program.

5.1 Run CMake

Open Command Prompt and change directory to where the configuration file "CMakeLists.txt" is located. Then, set options and run CMake to generate makefile. Frequent use CMake options are as below:

- -B: An output path for the result.
  In here, specify "_builds" as the generation folder for makefile.
- -G: A build system generator.
  In here, specify "Unix Makefiles" as project file format.
- -D: A cmake cache entry.
  In here, specify "CMAKE_TOOLCHAIN_FILE=ccrx.cmake" as toolchain file for CC-RX.

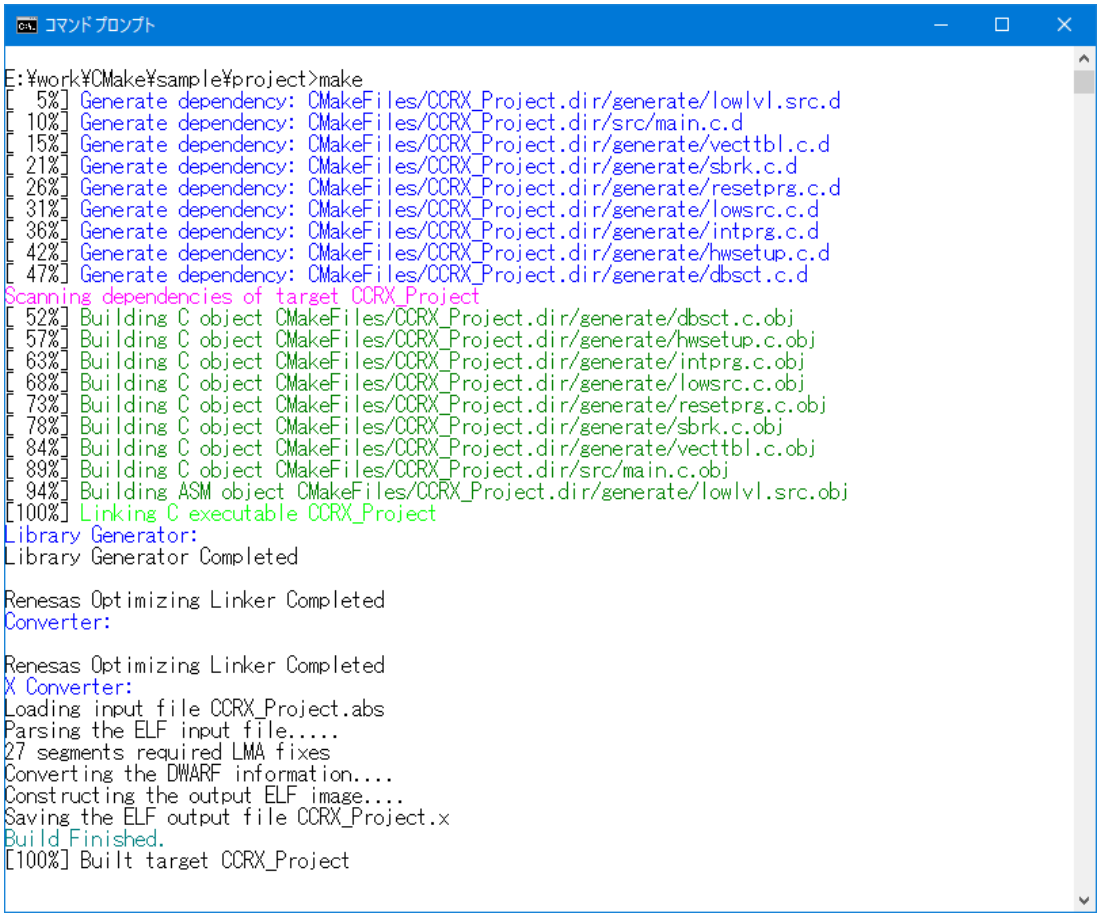
```
cmake -H. -B_builds -G "Unix Makefiles" -DCMAKE_TOOLCHAIN_FILE=ccrx.cmake
```

5.2 Run GNU Make

After CMake successfully generates the native build files. GNU Make needs to be executed to build the program.

```
cd _builds
make
```

<Example for Build result>



6. Sample code

Sample code which this document describes can be downloaded from the Renesas Electronics website. That sample code is a project targeting the RX65N.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.24. 20	-	First edition
2.00	Jun.30.21	all	Update whole. - Remove unnecessary information. - Replace file contents to general information.
2.01	Sep.15.21	Page 11	Add "Sample Code" section.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.