

# Renesas RA Family

# Getting Started with CoreMark Benchmarking

#### Introduction

As processors in embedded systems become more complex, more sophisticated benchmarks are needed to better understand performance and analysis.

CoreMark is a modern and sophisticated benchmark that is recommended by ARM<sup>®</sup> and allows you to accurately measure the performance of a processor. Rather than using arbitrary and synthetic code, CoreMark uses basic data structures and algorithms that are common in any embedded application.

Using CoreMark is encouraged due to its ANSI C compliance, and the fact that it is designed to ensure that compilers cannot pre-compute numbers to influence the results and that it does not make any library calls during the benchmarked portion of the code.

Running CoreMark produces a single-number score, allowing users to make quick comparisons between processors. Results can be uploaded to the CoreMark website for certification, as CoreMark has a standard format for reporting results.

This document aims to present and explain the results and the process of benchmarking Renesas RA MCUs using CoreMark.

This application note walks you through all the steps necessary to benchmark using CoreMark.

#### **Required Resources**

#### Development of tools and software

- e<sup>2</sup> studio v2024-07
- Renesas Flexible Software Package (FSP) v5.5.0
- Arm Compiler 6.21
- IAR Embedded Workbench v9.50.2

#### Hardware

• Renesas RA kit: EK-RA6M5

#### **Reference Manuals**

- RA Flexible Software Package Documentation Release v5.5.0
- User's Manual: Renesas RA6M5 Group User's Manual Rev.1.10
- Schematics: EK-RA6M5-v1.0



# Contents

1. CoreMark Project	3
2. Run CoreMark on Renesas RA MCUs	3
2.1 Integrating Toolchains with e <sup>2</sup> studio	3
2.1.1 IAR Embedded Workbench Plugin	3
2.1.2 Integrate with Arm Compiler	5
2.2 Create CoreMark e <sup>2</sup> studio Project Used for Benchmarking using the IAR Compiler	7
2.3 Add CoreMark to e <sup>2</sup> studio Project	10
2.4 Add Timer for Benchmarking	11
2.5 Update Main Stack	12
2.6 Port CoreMark Code	13
2.7 Create CoreMark e <sup>2</sup> studio Project Used for Benchmarking using Arm Compiler	19
2.8 Run CoreMark Project	22
2.8.1 Board Setup	22
2.8.2 Add Run Commands to Print Out Benchmarking Result.	23
2.8.3 Run the e <sup>2</sup> studio Project	24
3. Verify RA Benchmarking Results	26
4. General Guidelines for CoreMark Benchmarking	26
5. References	26
Revision History	28



## 1. CoreMark Project

The official CoreMark source is available at EEMBC <u>GitHub</u>. As we plan to use CoreMark on a bare-metal target, the source files we are going to use consist of the following C source and header files:

- coremark.h
- core\_main.c
- core\_list\_join.c
- core\_matrix.c
- core\_state.c
- core\_util.c
- core\_portme.c
- core\_portme.h
- cvt.c
- ee\_printf.c

The three key algorithms used are related to linked lists, matrix multiplication, and state machines.

At EEMBC <u>GitHub</u>, you will also find more information on the rules for building and running the CoreMark code.

The procedure to create a CoreMark project for Renesas RA MCUs is as follows.

- Create a Bare-Metal Minimal Project using e<sup>2</sup> studio and Flexible Software Package (FSP)
- Copy CoreMark source code to the "src" folder
- Add a 32-bit general-purpose timer (GPT) to the project
- Change the main stack size setting to 0x4000 to accommodate CoreMark benchmarking
- Exclude the main.c generated by FSP from the build
- Update project optimization with the maximum speed option
- Port core\_portme.h, core\_portme.c to add necessary code for the GPT
- Port ee\_printf.c to print benchmarking results.

This document explains the procedure for EK-RA6M5, but the same can be applied to other RA MCUs.

### 2. Run CoreMark on Renesas RA MCUs

Apart from the official CoreMark source, in order to be able to replicate exactly the process used in benchmarking the RA MCUs, you will need the e<sup>2</sup> studio IDE together with FSP. You can download and install setup\_fsp\_v5\_5\_0\_e2s\_v2024-07.exe from <u>https://github.com/renesas/fsp/releases</u>

Moreover, you will need the IAR Arm compiler available at <u>https://www.iar.com/products/architectures/arm/</u> and the Arm compiler available at <u>https://developer.arm.com/documentation/ka005198/latest</u>. You can use the Arm compiler in Keil MDK installation for CoreMark benchmarking.

### 2.1 Integrating Toolchains with e<sup>2</sup> studio

### 2.1.1 IAR Embedded Workbench Plugin

Download and install the IAR Embedded workbench before you integrate the IAR compiler with e<sup>2</sup> studio.

Start e<sup>2</sup> studio, then select "Help -> IAR Embedded Workbench plugin manager".



Help		
3	Welcome	
0	Help Contents	
2	Search	
	Show Context Help	
	Show Active Keybindings	Ctrl+Shift+L
	Cheat Sheets	
	Renesas Help	>
	CMSIS Packs Management	>
ø	Add Renesas Toolchains	
۹	Eclipse User Storage	>
*	Perform Setup Tasks	
<i>e</i>	Check for Updates	
<b>6</b> .	Install New Software	
\$	Eclipse Marketplace	
	Install New Device Family Support	
2	IAR Embedded Workbench plugin manager	
•	About e <sup>2</sup> studio	

Figure 1. Select IAR Embedded Workbench Plugin Manager

You choose the desired toolchain and press install.

upported targets			Available IA	R Embedded Workbench install	ations
Target	Installed plugin	^	Version	Status	IAR Embedded Workbench Installation path
ARM (6.50.x)	9.0.0.202407151343		9.40.	1 🌸 Plugin installed	C:\Program Files\IAR Systems\Embedded Workbe
ARM (8.10-8.22)	9.0.0.202407151343		9.50.	1 🌸 Plugin installed	C:\Program Files\IAR Systems\Embedded Workbe
ARM (8.30-8.50)	9.0.0.202407151343		9.50.	2 🌸 Plugin installed	C:\Program Files\IAR Systems\Embedded Worl
ARM (7.20 to 7.80)	9.0.0.202407151343				
ARM (9.x)	9.0.0.202407151343				
R32C (>= 1.30)	not installed				
RH850 (1.10 to 1.40)	not installed				
RH850 (2.x)	not installed				
RL78 (>= 2.10, <= 3.1	not installed				
RL78 (1.x)	not installed				
RL78 (4.x)	not installed				
RX (>= 2.20)	not installed				
RX (>= 4.10)	not installed	~	<		2

Figure 2. Select IAR Plugin

The bottom right corner of the  $e^2$  studio IDE will show configuration progress.

Configuring IAR plugins: (34%)

Figure 3. IAR Configuration Progress



Press "Next", then "Next". Accept the terms of the license agreements then click "Finish".

Check the items that you wish to install.			
Name           Image: Ward of the second state of the	Version 9.0.0.202302241222 4.0.0.202302241222 2.1.0.202302241222 9.0.0.202302241222	ld com.iar.cdt.arm.debugger.feature.featur com.iar.common.feature.feature.group com.iar.common.debugger.feature.featu com.iar.cdt.arm.sdk.feature.feature.group	
Select All Deselect All			
Details			Ċ

Figure 4. Install the IAR Embedded Workbench Plugin

The bottom right corner of the e<sup>2</sup> studio IDE will show the installation progress.

Installing Software: (48%)

Figure 5. IAR Plugin Installation Process

Wait for the plugin to be installed and click "Restart Now" to complete the installation process.

Software Updates		$\times$	
Restart e <sup>2</sup> studio to apply the software update?			
	Restart Now N	0	

Figure 6. Restart e<sup>2</sup> studio

#### 2.1.2 Integrate with Arm Compiler

Download and install the Arm compiler or Keil MDK before you integrate the Arm compiler with e<sup>2</sup> studio. Start e<sup>2</sup> studio, then select "Window -> Preferences" to add toolchains to e<sup>2</sup> studio.



Wi	indow	Help	
	New	w Window	
	Edit	tor	>
	Арр	pearance	>
	Sho	w View	>
	Sho	w C-SPY View	>
	Pers	spective	>
	Nav	/igation	>
	Pref	ferences	

Figure 7. e<sup>2</sup> studio Preferences

Select the desired Arm Compiler toolchain, then click "Apply and Close" to add the Arm compiler to e<sup>2</sup> studio.

Preferences			
type filter text	Toolchains		<
> General > C/C++	Add/Remove toolchains		
> Help	Name		Add
> IAR Embedded Workbench	ARM Compiler 6.19		
> Install/Update	ARM Compiler 6.21		Remove
> Java			
> Language Servers			
Library Hover			
> Oomph			
> Remote Development			
> Renesas			
> Run/Debug			
> Scripting			
Terminal			
> TextMate			
loolchains			
> Iracing			
Varian Control (Team)			
> XML	Name: ARM Compiler 6.21		
	Path: C\Keil v5\APM\APMCLANC\hin		
	Fault C. (Keil_v3 (AKW/AKWCLANG (DII)		
< >			Apply
2 2 2 2 2 2		Apply and Close	Cancol

Figure 8. Add Arm Compiler to e<sup>2</sup> studio



If the Arm compiler is not present in the Toolchains windows, click "Add", then browse to the toolchain folder, e.g., C:\Keil\_v5\ARM\\ARMCLANG\bin

📴 Preferences		_	
type filter text	Toolchains	<u>ب</u>	• ⇒ • 8
<ul> <li>&gt; General</li> <li>&gt; C/C++</li> <li>&gt; Help</li> <li>&gt; IAR Embed</li> <li>&gt; Install/Upd</li> <li>&gt; Java</li> <li>&gt; Language 5</li> <li>Library Hov</li> <li>&gt; Oomph</li> <li>&gt; Remote De</li> <li>&gt; Renesas</li> <li>&gt; Run/Debug</li> <li>&gt; Scripting Terminal</li> <li>&gt; TextMate Toolchains</li> <li>&gt; Tracing Validation</li> <li>&gt; Version Coi</li> <li>&gt; XML</li> </ul>	Add a new Toolchain Select Toolchain Path Please enter the path to the toolchain's binaries Path to toolchain binaries:	directory. Browse Download toolchains from ds.arm.com	Add Remove
<		> Finish Cancel	Apply
? 🖻 🗹		Apply and Close	Cancel

Figure 9. Add Toolchain's Path

# 2.2 Create CoreMark e<sup>2</sup> studio Project Used for Benchmarking using the IAR Compiler

Ensure you integrated the IAR compiler with e<sup>2</sup> studio before creating a CoreMark project.

On e<sup>2</sup> studio, select "File -> New-> C/C++ Project, then click "Next".

New C/C++ P	enesas RA Project	_		×
All C/C++	Renesas RA C/C++ Project Create an executable or static library C/C++ project for Renesas RA.			
?	< Back Next > Finish		Cancel	I

Figure 10. Select C/C++ Template



Name your project an appropriate name, e.g., RA6M5\_CoreMark\_IAR for EK-RA6M5 kit using the IAR compiler.

Renesas RA C/C++ Project
Renesas RA C/C++ Project     Image: Constraint of the second
Project name
KAbM5_CoreMark_IAR       Use default location
Location:       C:\Users\b3202733\e2_studio\workspace_coremark_ra\RA6M5_CoreM       Browse         Choose file system:       default
You can download more Renesas packs here
? < Back Next > Finish Cancel

Figure 11. Name Your Project

Select the Board, Device, and Toolchain you want to use for benchmarking.

🗐 Renesas RA C/C	E++ Project		- C X	
Renesas RA C/C Device and Tools	++ Project Selection			
Device Selection FSP Version: 5.5. Board: EK Device: R7f Core: CM	0 ~ ~ RA6M5 ~ ~ ~ ~ ~ A6M5BH3CFC ~ ~ ~ 33 ~ ~	Board Description Evaluation kit for RA6 Visit <u>https://www.rene</u> manual, quick start gu projects, etc.	M5 MCU Group esas.com/ra/ek-ra6m5 to get kit user's ide, errata, design package, example	
Language: 🔘	c ○c++	Device Details TrustZone Pins Processor	Yes 176 Cortex-M33	
Toolchains GNU ARM Embed LLVM Embedded	Ided Toolchain for Arm	Debugger J-Link ARM	~	
ARM Compiler 6.	Arm - (9.x) 21 19 V Manage Toolchains			
?		< Back	lext > Finish Cancel	

Figure 12. RA Project Options



Select Flat (Non-TrustZone) Project.



Figure 13. Flat Project Selection

After this step, select Executable project type with No RTOS.

Renesas RA C/C++ Project          Renesas RA C/C++ Project         Build Artifact and RTOS Selection         Build Artifact Selection         Image: Static Library         Image: Project builds to an executable file         Static Library         Image: Project builds to a static library file         Executable Using an RA Static Library         Image: Project builds to an executable file         Image: Project builds to an existing RA static library project         Image: Project builds to an existing RA static library project         Image: Project Builds to a executable file         Image: Project Builds to a existing RA static library project         Image: Project Builds to a executable file         Image: P	×	~	Cancel
<ul> <li>Renesas RA C/C++ Project</li> <li>Build Artifact and RTOS Selection</li> <li>Build Artifact Selection</li> <li>Executable         <ul> <li>Project builds to an executable file</li> <li>Static Library                 <ul> <li>Project builds to a static library file</li> <li>Executable Using an RA Static Library</li> <li>Project builds to an executable file</li> <li>Project uses an existing RA static library project</li> <li>Project uses an existing RA static library project</li></ul></li></ul></li></ul>	×	RTOS Selection No RTOS	Next > Finish Cancel
	Renesas RA C/C++ Project Renesas RA C/C++ Project Build Artifact and RTOS Selection	Build Artifact Selection <ul> <li>Executable</li> <li>Project builds to an executable file</li> </ul> Static Library <ul> <li>Project builds to a static library file</li> <li>Executable Using an RA Static Library</li> <li>Project builds to an executable file</li> <li>Project uses an existing RA static library project</li> </ul>	? < Back

Figure 14. Select No RTOS Project



Select Bare Metal – Minimal Project Template. Click "Finish" to generate the project.

Renesas RA C/C++	Project		$ \sim$ $\rightarrow$	<
Renesas RA C/C++	Project			
Project Template Sele	ction			
Project Template Sele	ction			
Bare Bare the C [Renv	e Metal - Blinky metal FSP project that includes BSP and will blink LEDs if avail runtime environment. esas.RA.5.5.0.pack]	able. This project will initialize clc	ocks, pins, stacks, and	
Bare     Inclusion	e Metal - Minimal metal FSP project that includes BSP. This project will initialize esas.RA.5.5.0.pack]	clocks, pins, stacks, and the C runt	ime environment.	
Code Generation Sett	ings Formatter			

Figure 15. Bare Metal Minimal Option

# 2.3 Add CoreMark to e<sup>2</sup> studio Project

Copy the CoreMark source code to the "src" folder in your newly created project. The project structure should look as follows.

v 📂 RA6M5_CoreMark_IAR [Debug]
> 🔊 Includes
> 🔑 ra
> 🔑 ra_gen
✓ 🔁 src
> 🔂 core_list_join.c
> 🔂 core_main.c
> 🔂 core_matrix.c
> 🔂 core_portme.c
> h core_portme.h
> core_state.c
> 🔂 core_util.c
> h coremark.h
> 尾 cvt.c
> 🖻 ee_printf.c
> 🖻 hal_entry.c

Figure 16. CoreMark Project



You now need to add a periodic timer and modify the core\_portme.c source file in order to use the modified barebones\_clock(), portable\_init(core\_portable \*p, int \*argc, char \*argv[]) and portable\_fini(core\_portable \*p) functions.

To add a new periodic timer, open the configuration.xml file and go to Stacks. You should see something similar to the picture below.

tion		Generate Project Content
E New Thread Remove	HAL/Common Stacks	new Stack > 🐣 Extend Stack > 🔊 Remove
new Object > 紀 Remove		
	Etion   New Thread Remove	Image: Second

Figure 17. Stack Configuration

# 2.4 Add Timer for Benchmarking

The next step is to add a New Stack, then select Timers and, finally, Timer, General PWM(r\_gpt).

			Generate Project C	ontent
	New Stack	> 4	Extend Stack > Re Analog Artificial Intelligence Audio Bootloader CapTouch Connectivity DSP Input Monitoring Motor Networking Power Security	ontent move > > > > > > > > > > > > > > > > > > >
			Sensor Storage System	>
⊕ ⊕ ⊕	Port Output Enable for GPT (r_poeg) Realtime Clock (r_rtc) Three-Phase PWM (r_gpt_three_phase) Timer, General PWM (r_gpt)	AP.	Timers Transfer Search	>
<b>+</b>	Timer, Low-Power (r_agt)			

Figure 18. Add GPT Timer



After adding the GPT timer, you need to edit the settings. Clicking on the block representing the newly added GPT timer, then go to the Properties Window. You use this Properties window to change the timer's name to g\_timer\_periodic, the period to 50, and the period unit to Seconds. You also need to expand the Interrupts block, add the Callback as timer\_callback and set the Priority to 2. The following image captures the changes needed.

Propertie	es × Roblems Smart Browser	
g_umer_	periodic filler, General P will (1_gpt)	
Settings	Property	Value
API Info	✓ Common	
	Parameter Checking	Default (BSP)
	Pin Output Support	Disabled
Write Protect Enable		Disabled
Clock Source		PCLKD
	<ul> <li>Module g_timer_periodic Timer, General PWM (r_gpt)</li> </ul>	
	✓ General	
	Name	g_timer_periodic
	Channel	0
	Mode	Periodic
	Period	50
	Period Unit	Seconds
	> Output	
	> Input	
	✓ Interrupts	
	Callback	timer_callback
	Overflow/Crest Interrupt Priority	Priority 2
	Capture A Interrupt Priority	Disabled
	Capture B Interrupt Priority	Disabled
	Underflow/Trough Interrupt Priority	Disabled
	> Extra Features	

Figure 19. GPT Configuration

# 2.5 Update Main Stack

Change the Main Stacks Size in BSP properties to 0x4000.

Summary	Summary BSP Clocks Pins Interrupts Event Links Stacks Components						
🖳 Problem	💦 Problems 📮 Console 🔲 Properties 🗙 💸 Smart Browser 🤑 Smart Manual						
EK-RA6	M5						
Settings	Property > R7FA6M5BH3CFC	Value					
	KADMO     KADMO     KAGMO     KAGMO     KAGMO     KAGMO						
	Main stack size (bytes) Heap size (bytes)	0x4000 0					
	MCU Vcc (mV) Parameter checking	3300 Disabled					
	Assert Failures Error Log	Return FSP_ERR_ASSERTION No Error Log					
	Clock Registers not Reset Values during Startup Main Oscillator Populated	Disabled Populated					
	PFS Protect C Runtime Initialization	Enabled Enabled					
	Early BSP Initialization Main Oscillator Clock Source	Disabled Crystal or Resonator					
	Subclock Populated Subclock Drive (Drive capacitance availability varies by MCU)	Populated Standard/Normal mode					
	Subclock Stabilization Time (ms)	1000					

Figure 20. Change Main Stack Size



Click "Generate Project Content", then the next step is to modify the source files.

Right click on the ra\_gen\main.c and exclude it from the Build, so there is no conflict with the main from core\_main.c.



Figure 21. Exclude main.c from Build

### 2.6 Port CoreMark Code

You modify the core\_portme.h and the core\_portme.c in the "src" folder.

In core\_portme.h, add "#include <stddef.h>" before the code "typedef size\_t ee\_size\_t;", as shown below.



Figure 22. Add "#include <stddef.h>"



Also, in core\_portme.h, modify the "#define COMPILER\_FLAGS" depending on the toolchain used. If you use IAR Compiler version 9.50.2, change it to #define COMPILER\_FLAGS "High Speed; No size constraints".

The code should look as follows.



Figure 23. Modify core\_portme.h

In core\_portme.c, before the barebones\_clock() function, add the below code.

Figure 24. Modify core\_portme.c

You can check and correct the CLOCKS\_PER\_SEC setting by getting the correct value from the clock\_frequency, shown in the figure below, when running the project for the first time.

if (FSP_SUCCESS != err)				
<pre>i ee_printf("ERROR: R_GPT_Start!\n"); }</pre>				
err = R GPT InfoGet(&g timer periodic ctrl.	&g timer info:			
if (FSP SUCCESS != err)	-0			
{	Expression	Туре	Value	Address
<pre>ee_printf("ERROR: R_GPT_InfoGet!\n");</pre>	✓ (= g_timer_info	timer_info_t	{}	0x20004250
}	(x)= count_direction	timer_direction_t	TIMER_DIRECTION_UP	0x20004250
<pre>if (sizeof(ee_ptr_int) != sizeof(ee_u8 *)) </pre>	(x)= clock_frequency	uint32_t	5000000	0x20004254
i ee printf(	(x)= period_counts	uint32_t	250000000	0x20004258
<pre>"ERROR! Please define ee_ptr_int to "pointer!\n");</pre>	a Name : g_timer_info Details:{count_di	rection = TIMER_DIREC	TION_UP, clock_freque	ncy = 50000000, peri
if (sizeof(ee_u32) != 4)	Default:{} Decimal:{}			
<pre>ee_printf("ERROR! Please define ee_u32 t }</pre>	Hex:{} Binary:{}			
J a sanatable da da	<			>

Figure 25. Check CLOCK\_PER\_SEC Setting



Change the barebones\_clock() functions as follows.

```
/* Porting : Timing functions
   How to capture time and convert to seconds must be ported to whatever is
  supported by the platform. e.g. Read value from on board RTC, read value from
  cpu clock cycles performance counter etc. Sample implementation for standard
  time.h and windows.h definitions included.
*/
CORETIMETYPE
barebones_clock()
ł
   fsp_err_t err = FSP_SUCCESS;
   timer_status_t status;
    err = R_GPT_StatusGet (&g_timer_periodic_ctrl, &status);
    if (FSP SUCCESS != err)
    Ł
       ee_printf("ERROR: R_GPT_StatusGet!\n");
    }
    /* The period is set to 50s we shouldn't overflow but just in case
      report an error if we do. If we set the a shorter period we need to do:
      info.period_counts * g_capture_overflows */
    if (g_capture_overflows > 0)
    Ł
       ee_printf("ERROR: Timer overflow!\n");
    }
    return status.counter;
}
```

Figure 26. bareborns\_clock() Function



Then change the portable\_fini(core\_portable \*p), portable\_init(core\_portable \*p, int \*argc, char \*argv[]) to add the GPT timer that is needed for benchmarking.

```
/* Function : portable_init
       Target specific initialization code
       Test for some common mistakes.
*/
void
portable_init(core_portable *p, int *argc, char *argv[])
-
   fsp_err_t err = FSP_SUCCESS;
   /* Flush C cache */
   uint32_t * c_cache = (uint32_t *)0x40007004;
   *c_cache = 1;
   /* Enable C cache */
   c cache = (uint32 t *)0x40007000;
   *c cache = 1;
   /* Flush S cache */
   uint32 t * s cache = (uint32 t *)0x40007044;
   *s cache = 1;
   /* Flush S cache */
   s cache = (uint32 t *)0x40007040;
   *s cache = 1;
   /* Initialize GPT Timer */
   err = R_GPT_Open(&g_timer_periodic_ctrl, &g_timer_periodic_cfg);
   if (FSP SUCCESS != err)
    £
       ee_printf("ERROR: R_GPT_Open!\n");
    }
   err = R_GPT_Start(&g_timer_periodic_ctrl);
   if (FSP SUCCESS != err)
    Ł
        ee printf("ERROR: R GPT Start!\n");
    1
   err = R GPT InfoGet(&g timer periodic ctrl, &g timer info);
   if (FSP SUCCESS != err)
    ł
       ee printf("ERROR: R GPT InfoGet!\n");
    3
    if (sizeof(ee_ptr_int) != sizeof(ee_u8 *))
    ł
        ee printf(
            "ERROR! Please define ee_ptr_int to a type that holds a "
           "pointer!\n");
    if (sizeof(ee u32) != 4)
        ee_printf("ERROR! Please define ee_u32 to a 32b unsigned type!\n");
   p->portable_id = 1;
}
```

#### Figure 27. portable\_init Function



Figure 28. portable\_fini Function

At the end of the file, add the callback method function of the GPT timer.



Figure 29. Add timer\_callback to core\_portme.c

In ee\_printf.c, change the uart\_send\_char(char c) and add the code below for printing benchmarking results.

```
#define MAXBUFFER 1000
volatile char uart_buffer[MAXBUFFER + 1];
volatile unsigned int uart_buffer_cnt = 0;
void
uart_send_char(char c)
{
    if(uart_buffer_cnt < MAXBUFFER)</pre>
    {
        uart_buffer[uart_buffer_cnt++] = c;
        uart buffer[uart buffer cnt] = '\0';
    }
    else
    {
        uart_buffer[uart_buffer_cnt] = '\0';
    }
}
```

Figure 30. Add uart\_send\_char Function



In the project properties setting, add "ITERATIONS=9000" to the IAR C/C++ Compiler for ARM->Preprocessor.



Figure 31. Preprocessor Setting

In the project properties setting, change IAR C/C++ Compiler for ARM->Optimization to "High, Speed" with "No size constraints".



Figure 32. Optimization Setting



Now, you can build the project without errors.

### 2.7 Create CoreMark e<sup>2</sup> studio Project Used for Benchmarking using Arm Compiler

Ensure you integrated the Arm compiler with e<sup>2</sup> studio before creating a CoreMark project. Select the Board, Device, and Toolchain you want to use for benchmarking and process to create an Arm compiler-based project similar to the IAR compiler. Follows sections 2.3, 2.4, 2.5, and 2.6 to add the GPT module, configure your project, and port the CoreMark. Note that you need a commercial license to use "--Ito" option in Arm Compiler.

Device Selection FSP Version: 5.5.0  Board: EK-RA6M5   Device: R7FA6M5BH3CFC  Core: CM33 Language:  CC++		Board Description Evaluation kit for RA6M5 MCU Group Visit <u>https://www.renesas.com/ra/ek-ra6m5</u> to get kit user's manual, quick start guide, errata, design package, example projects, etc.				
Languager			Device Details			
			TrustZone Pins Processor	Yes 176 Cortex-M33		
oolchains			Debugger			
GNU ARM E LLVM Embed IAR Toolchai ARM Compi	mbedded dded Toolchain for Arm in for Arm - (9.x) ler 6.21	Manage Toolchains	J-Link ARM	~		

Figure 33. Create An Arm Compiler - Based Project Options



Also, in core\_portme.h, modify the "#define COMPILER\_FLAGS" depending on the toolchain used. In the case of Arm Compiler, change it to "-Omax".

The code should look as follows.



Figure 34. Modify "#define COMPILER\_FLAGS"

In the project's Properties-> ARM C Compiler 6.15->Miscellaneous->Other flags, add "-Omax".

e filter text	Settings	⇔ ◄ ⇔
esource uilders C/C++ Build Build Variables	Configuration: Debug [Active]	✓ Manage Configurations.
Environment Logging Settings	😵 Tool Settings 🎤 Build Steps 🤮 Build Artifact 🔛 Binary Parsers 🥝 Error Parsers	
1001 Chain Editor //C++ General 4CU roject Natures Iroject References lenesas QE lun/Debug Settings sick Tans	White Compared of J     White Compared on J     W	ninsing uccanauons - wunninuanceu - wunsed - Who-ildense-management - wetta- Umat
idation	Image: Second Secon	
>	Beologying     Warnings and Errors     WinfordInneurs	
		Apply and Close Canci

Figure 35. Add "-Omax" Option to Project Settings



In the project's Properties-> ARM Linker 6.15->Miscellaneous->Other flags, add "--Ito".



--library\_type=microlib --no\_startup --via="\${workspace\_loc:/\${ProjName}/script}/ac6/fsp\_keep.via" --lto

Figure 36. Add "--Ito" Option to Project Settings.



#### 2.8 Run CoreMark Project

#### 2.8.1 Board Setup

The EK-RA6M5 kit has a few switch settings that must be configured before running the projects associated with this application note. In addition to these switch settings, the boards also contain a USB debug port and connectors to access the J-Link<sup>®</sup> programming interface.

#### Table 1. Switch settings for EK-RA6M5

Switch	Setting
J8	Jumper on pins 1-2
J9	Open



Figure 37. J8 and J9 on EK-RA6M5

The figure below shows the picture of the EK-RA6M5 kit.



Figure 38. EK-RA6M5



Connect the board to your PC using the USB cable into the port labeled "Debug1".

#### 2.8.2 Add Run Commands to Print Out Benchmarking Result.

In Debug Configuration, add the below command.

dprintf portable\_fini,"%s",uart\_buffer

Debug Configurations Create, manage, and run configurations						- □ ×
Image: Second Science of Science o	Name: RA6M5_CoreMark_IAR Dr	ebug_Flat tup 1 Source Co	ommon			
	Filename Program Binary [RA6M	Load type Image and Symbols	Offset (hex)	On connect Yes		Add Edit Remove Move up Move down
	Runtime Options Set program counter at (hex Set breakpoint at: Resume Run Commands dprintf portable_fini,"%s",uart	): main _buffer				^
< > > Filter matched 10 of 12 items					Revert	Apply

Figure 39. Add dprint Command



#### 2.8.3 Run the e<sup>2</sup> studio Project

After successfully building the project, it can be debugged using Renesas GDB Hardware Debugging. Rightclick on the project -> Debug AS -> Renesas GDB Hardware debugging or "Debug Configurations..." and choose the desired one.

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help					
🐔 株 🔳 🔅 Debug		✓ RA6M5_CoreMark_IAR Debug_Flat ✓	🌞 🗄 📩 👻	🗐 🐚   🗞 <del>- 4</del> 5 - 🔜 i 🗶   🕪 II 🔳 💉 22, 12-	
🎦 Project Explorer 🗙 📄	¢1	🍸 🖇 🗖 🗐 🔯 [RA6M5_CoreMark_IAR] FSP Co	onfiguration	€ main.c € startup.c × € core_main.c €	
✓		New	>	TION(BSP_SECTION_HEAP);	
> 🐉 Binaries		Go Into			
> 🔊 Includes		Open in New Window		ions in the vector table are weak references t	
> 🗁 ra		Show In	\lt+Shift+W >	in their code they should define their own fun	
✓ 🖉 src			AIL+ DHILL+ W		
ic core_list_join.c		Сору	Ctrl+C	TRIBUTE	
> 🔂 core_main.c	Ē	Paste	Ctrl+V	ult Handler = Default H	
> core_matrix.c	×	Delete	Delete	age_Handler = Default_H	
> [c] core_portme.c		Source	>	lt_Handler = Default_H	
> Core state.c		Move		Fault_Handler = Default_H	
> core_util.c		Rename	F2	ndler = Default_H	
> h coremark.h	n de la	Import		Handler = Default H	
> 💽 cvt.c	4	Export		k_Handler = Default_H	
> 🙋 ee_printf.c	<b>₽</b> 5	Send project settings to Reality AI Tools®		'	
> 🕞 Debug		Renesas FSP Export	>	TRIBUTEattribute((weak, alias("Default	
> 🕷 None		Puild Designt			
> 🗁 ra_cfg		Clean Broject		d); // NMI has many sources and	
> 🧀 script	5	Defreeh	55	er(void) WEAK_REF_ATTRIBUTE; er(void) WEAK REF ATTRIBUTE;	
	ф <u>г</u>	Clean Deviant	F.J	r(void) WEAK_REF_ATTRIBUTE;	
R7FA6M5BH3CFC.pincfg				ler(void) WEAK_REF_ATTRIBUTE; dler(void) WEAK_REF_ATTRIBUTE;	
ra_cfg.txt				d) WEAK REF ATTRIBUTE:	
RA6M5_CoreMark_IAR De		Build Targets	>		
KAOM5_CoreMark_IAK De     Developer Assistance		Index	>	Smart Browser 🐺 Smart Manual 🔅 Debug	
y (j) berelopei Assistance		Build Configurations	>	Renesas GDB Hardware Debugging] Renesas GDB server (Host	
	0	Run As	>		
	*	Debug As	>	1 GDB OpenOCD Hardware Debugging (DSF)	
		Team	>	C 2 GDB Simulator Debugging (RH850)	
		Compare With	>	3 IAR C-SPY Application	
		Restore from Local History		C 4 Local C/C++ Application	
		MISRA-C	>	5 Renesas GDB Hardware Debugging	
		C/C++ Project Settings	Ctrl+Alt+P	c 8 Renesas Simulator Debugging (RX, RL78)	
		Renesas C/C++ Project Settings	>	Debug Configurations	
		Select Device			

Figure 40. Debug the Project



The program should stop in the Reset\_handler.

Debug 🗙 📄 🦮 📩 🤜 🖬 🕴 🗖 🗖	🖻 startup.c 🗙 🗖	core_main.c 🗈 core_portme.c 👍 core_portme.h 🗈 hal_entry.c 💽 _iar_progra »2
RA6M5_CoreMark_IAR Debug_Flat [Renesas GDB	48	⊕ * Exported global variables (to be accessed by other files).
RA6M5_CoreMark_IAR.out [1] [cores: 0]	50	
✓ Inread #1 1 (single core) [core: 0] (Suspend	52	Private global variables and functions.     And the second secon
Reset Handler() at startup.c:62 0x4dc0	54	Vold Reset_Handler(Vold);
Oxffffffe	56	void Deraulmanuter(void);
arm-none-eabi-adb (7.8.2)	57	incor_c main(void);
Banana CDR annua (Last)	59	$\oplus$ * MCU starts executing here out of reset. Main stack pointer is set up already.
Refiesas ODB server (Host)	61	⊖void Reset Handler (void)
	63	/* Initialize system using BSP. */
	64 00004dc4	SystemInit();
	65	
	66	/* Call user application. */
	67 00004dc8	main();
	68	
	69 00004dcc	⊖ while (1)
	70	{
	71	/* Infinite Loop. */
	72	, }
	75	3
	74	• * Default augentian handles
	70	• belative exception manifer
	70	
	20	$\mathbf{x}$
	00	7 A error has occurred. The user will need to investigate the cause. Common problems are state

Figure 41. Debug the Project (cont'd)

Click the Resume button. The program will stop in main from core\_main.c, and click the Resume button again to run the project.

After a while, the program will stop in portable\_fini, and the CoreMark scores will be available in the Debugger Console window, as shown below.

```
CoreMark 1.0 : 805.740195 / IAR Compiler version 9.50.2 High Speed; No size constraints / STACK
2K performance run parameters for coremark.
CoreMark Size : 666
Total ticks
                : 558492679
Total time (secs): 11.169854
Iterations/Sec : 805.740195
                : 9000
Iterations
Compiler version : IAR Compiler version 9.50.2
Compiler flags : High Speed; No size constraints
Memory location : STACK
            : 0xe9f5
seedcrc
              0xe9f5
: 0xe714
: 0x1fd7
[0]crclist
[0]crcmatrix
                : 0x8e3a
[0]crcstate
             : 0x382f
[0]crcfinal
Correct operation validated. See README.md for run and reporting rules.
CoreMark 1.0 : 805.740195 / IAR Compiler version 9.50.2 High Speed; No size constraints / STACK
```

Figure 42. CoreMark Score with IAR Compiler



2K performance run parameters for coremark.
CoreMark Size : 666
Total ticks : 580600338
Total time (secs): 11.612007
Iterations/Sec : 775.059831
Iterations : 9000
Compiler version : GCCClang 18.0.0
Compiler flags : -Omax
Memory location : STACK
seedcrc : 0xe9f5
[0]crclist : 0xe714
[0]crcmatrix : 0x1fd7
[0]crcstate : 0x8e3a
[0]crcfinal : 0x382f
Correct operation validated. See README.md for run and reporting rules.
CoreMark 1.0 : 775.059831 / GCCClang 18.0.0 -Omax / STACK
-



# 3. Verify RA Benchmarking Results

You can verify your results by referring to RA CoreMark results published on the EEMBC website, as shown below.

Clear									CoreMark/		
Sel.	Vendor	Processor	Cert.	Compiler	Execution Memory	MHz	Cores	CoreMark	MHz	Threads	Date↓
	Renesas Electronics	RA6T2	$\checkmark$	ARM Clang Compile	internal flash, intern	240	1	962.45	4.01	1	2022-03-17
	Renesas Electronics	RA6T2	$\checkmark$	IAR C/C++ Compiler	internal flash, intern	240	1	950.68	3.96	1	2022-03-17
	Renesas Electronics	RA2E2	~	IAR C/C++ Compiler	internal flash, intern	48	1	110.24	2.29	1	2021-12-14
	Renesas Electronics	RA4E1	~	IAR C/C++ Compiler	internal flash, intern	100	1	386.67	3.86	1	2021-09-23
	Renesas Electronics	RA4E1	$\checkmark$	ARM Clang Compile	internal flash, intern	100	1	398.30	3.98	1	2021-09-23
	Renesas Electronics	RA6E1	~	IAR C/C++ Compiler	internal flash, intern	200	1	770.75	3.85	1	2021-09-23
	Renesas Electronics	RA6E1	$\checkmark$	ARM Clang Compile	internal flash, intern	200	1	790.27	3.95	1	2021-09-23
	Renesas Electronics	RA6M5	√	IAR C/C++ Compiler	internal flash, intern	200	1	770.82	3.85	1	2021-04-26
	Renesas Electronics	RA6M5	$\checkmark$	ARM Clang Compile	internal flash, intern	200	1	790.76	3.95	1	2021-04-26
	Renesas Electronics	RA4M2	~	ARM Clang Compile	internal flash, intern	100	1	398.30	3.98	1	2021-04-26
	Renesas Electronics	RA4M2	$\checkmark$	IAR C/C++ Compiler	internal flash, intern	100	1	386.00	3.86	1	2021-04-26
	Renesas Electronics	RA2E1	$\checkmark$	IAR C/C++ Compiler	internal flash, intern	48	1	111.73	2.32	1	2021-04-26
	Renesas Electronics	RA6T1	$\checkmark$	IAR C/C++ Compiler	internal flash, intern	120	1	405.90	3.38	1	2021-03-10
	Renesas Electronics	RA6M4	~	ARM Clang Compile	internal flash, intern	200	1	790.75	3.95	1	2021-03-10
	Renesas Electronics	RA6M4	$\checkmark$	IAR C/C++ Compiler	internal flash, intern	200	1	770.52	3.85	1	2021-03-10
	Renesas Electronics	RA4M3	~	ARM Clang Compile	internal flash, intern	100	1	397.30	3.97	1	2021-03-10
	Renesas Electronics	RA4M3	$\checkmark$	IAR C/C++ Compiler	internal flash, intern	100	1	386.11	3.86	1	2021-03-10
	Renesas Electronics	RA2L1	$\checkmark$	IAR C/C++ Compiler	internal flash, intern	48	1	111.73	2.32	1	2021-03-10
	Broadcom Corporation	Broadcom BCM283		GCC 7.2.1	LPDDR2 900MHz	1200	4	15363.93	12.80	4	2018-01-06

Figure 44. RA Coremark scores published on the EEMBC website

# 4. General Guidelines for CoreMark Benchmarking

Since target devices that contain Arm processors may have a wide variety of memories and memory hierarchies, your CoreMark project should be compiled using memory correctly and efficiently. Depending on the compiler, you can achieve this by correctly editing your linker script or scatter files.

Since CoreMark is a small benchmark, it should be run multiple times to obtain reproducible numbers.

Arm recommends performing two validation runs followed by at least ten profile runs. The results can be calculated by the average for the profile runs. These steps are necessary to minimize the variation caused by inconsistent processor states.

### 5. References

EEMBC's CoreMark® https://www.eembc.org/coremark/



# Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	www.renesas.com/ra
RA Product Support Forum	www.renesas.com/ra/forum
RA Flexible Software Package	www.renesas.com/FSP
Renesas Support	www.renesas.com/support



# **Revision History**

		Description	
Rev.	Date	Page	Summary
1.00	March.20.23	-	Initial version
1.10	Sep.11.24	-	Update to FSP v5.5.0



#### 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 reaches the level at which reseting 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

- 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

"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, 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.
- This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
   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

#### 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: <a href="http://www.renesas.com/contact/">www.renesas.com/contact/</a>.