

Azure RTOS sample projects using e² studio or IAR EW

R01AN6455EJ0210 Rev.2.10 2024.7.1

Important Notice:

Notice 1:

On November 21, 2023, Microsoft announced that they have decided to contribute Azure RTOS to Open Source under the stewardship of the Eclipse foundation and Azure RTOS becomes Eclipse ThreadX.

For detailed information, please refer to the announcement titled at Microsoft Contributes Azure RTOS to Open Source.

The support strategy scheme for Eclipse ThreadX will be determined and communicated at a later date.

Microsoft will discontinue the Azure RTOS and Azure RTOS Middleware under the existing agreement LICENSED-HARDWARE.txt.

It's important to note that updates for Azure RTOS on these hardware will no longer be provided.

Notice 2:

Renesas announces to discontinue the existing Sequans-sourced LTE module known as the part number RYZ014A and will no longer be shipping this product.

With the discontinuation of RYZ014A, the CK-RX65N v1 board will also be discontinued. If you are using RYZ014A in a current design or production, the Sequans part numbers, GM01Q is a pin and functionally compatible replacement for RYZ014A.

Below Cellular driver of RX family works the below alternate product combination. - RYZ014A Cellular Module Control Module : Sequans GM01Q is the compatible module.

Regarding EOL notice of the RYZ014A, please see :

[The link] <u>https://www.renesas.com/us/en/document/eln/plc-240004-end-life-eol-process-select-part-</u>numbers?r=1503996

[The product page] <u>https://www.renesas.com/us/en/products/wireless-connectivity/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module</u>



Introduction

Azure RTOS sample projects for each component (ThreadX, FileX, GUIX, NetX Duo, and USBX) can be created using Renesas e² studio or IAR Embedded Workbench (EW) with the on-board emulator. All samples are designed to run on RX family.

This document guides how to create and use these sample projects.

Supported Sample Projects

- ThreadX sample project Contains ThreadX source code
- Minimal sample project

Contains ThreadX source code

Simplest sample for ThreadX

- FileX RAM Disk sample project Contains FileX source code
- NetX Duo Ping sample project Contains NetX Duo ping sample project
- NetX Duo Iperf sample project Contains NetX Duo iPerf sample project
- **IoT Embedded SDK sample project** Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS
- IoT Embedded SDK PnP sample project (Available up to Azure RTOS v.6.2.1_rel-rx-2.0.0) Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS via IoT Plug and Play
- GUIX 8bpp sample project Contains sample for GUIX 8BPP
- GUIX 16bpp sample project Contains sample for GUIX 16BPP
- GUIX 16bpp draw 2d sample project Contains sample for GUIX 16BPP with 2D Draw
- USBX device CDC-ACM Class sample project Contains USBX source code
- USBX Host Mass Storage Class sample project Contains USBX source code
- USBX Host Communication Device Class (CDC-ACM) sample project Contains USBX source code
- ThreadX Low Power sample project Contains ThreadX & low power utility source code
- Azure Device Update (ADU) sample project Sample project for OTA firmware update via Microsoft Azure
- Secure bootloader sample project Used together with ADU sample project to provide a secure boot

Supported Devices

- RX130
- RX140
- RX23E-B
- RX26T



- RX65N
- RX651
- RX660
- RX66T
- RX671
- RX72N

Table 1 Supported Kits

Device		RX	65N		RX651	RX130	RX140	RX23E-B	RX26T	RX660	RX66T	RX72N	RX671
Board	RSK	Cloud Kit	New CK	New CK	RSK RX65N	Target Board	RSK	RSSK	МСВ	RSK	RSK	Envision Kit	RSK
Connectivity	Ethernet	Wi-Fi	Ethernet	Cat M1	-	-		-	-		-	Ethernet	Wi-Fi
ThreadX sample	0	0	0	0	0	0	0	0	0	0	0	0	0
Minimal sample	0	0	0	0	0	0	0	0	0	0	0	0	0
FileX sample	0	0	0	0	0	0	0	0	0	0	0	0	0
Ping sample	0	0	0	-	-	-	-	-	-	-	-	0	0
lperf sample	0	-	0	-	-	-	-	-	-	-	-	0	-
IoT Embedded SDK sample	0	0	0	-	-	-	-	-	-	-	-	0	0
IoT Embedded SDK sample using EWF	-	-	-	0	-	-	-	-	-	-	-	-	-
IoT Embedded SDK PnP sample	0	0	0	-	-	-	-	-	-	-	-	0	0
IoT Embedded SDK PnP sample using EWF	-	-	-	0	-	-	-	-	-	-	-	-	-
GUIX (8bpp, 16bpp, draw 2d) sample	0	-	-	-	-	-	-	-	-	-	-	0	-
USBX PCDC sample	0	0	0	0	-	-	-	-	-	-	-	-	-
USBX HMSC sample	0	-	0	0	-	-	-	-	-	-	-	0	0
USBX HCDC sample	0	-	0	0	-	-	-	-	-	-	-	0	0
Threadx Low Power sample	0	0	0	0	0	0	0	0	-	0	-	0	0
ADU sample	0	0	0	-	-	-	-	-	-	-	-	0	0
Bootloader sample	0	0	0	-	-	-	-	-	-	-	-	0	0

Supported sample projects are different by each device. For details, please refer to the following URL.

https://github.com/renesas/azure-rtos

Download Links for Development Environment

• <u>e² studio : 2024-04 or later</u>

https://www.renesas.com/software-tool/e-studio

- Renesas C/C++ Compiler for RX Family CC-RX : V3.06.00 or later https://www.renesas.com/software-tool/cc-compiler-package-rx-family
- GCC for Renesas RX : 8.3.0.202311 or later https://gcc-renesas.com/rx-download-toolchains/
- IAR Embedded Workbench for RX : 4.20.1 or later

https://www.iar.com/products/architectures/renesas/iar-embedded-workbench-for-renesas-rx/

- RX Smart Configurator : V2.21.0 and later https://www.renesas.com/software-tool/smart-configurator
- <u>Azure IoT Explorer</u>

https://github.com/azure/azure-iot-explorer/releases



Contents

1.	Getting Started	5
1.1	Creating project using e ² studio	5
1.2	Creating project using IAR EW	7
2.	Sample Project Descriptions	8
2.1	ThreadX sample project	8
2.2	Minimal sample project	10
2.3	FileX RAM Disk sample project	12
2.4	NetX Duo Ping sample project	14
2.5	NetX Duo Iperf sample project	16
2.6	IoT Embedded SDK sample project	18
2.7	IoT Embedded SDK PnP sample project	24
2.8	GUIX 8bpp/16bpp/16bpp_draw2d sample project	26
2.9	USBX device CDC-ACM Class sample project	28
2.10	USBX Host Mass Storage Class sample project	30
2.11	USBX Host Communication Device Class (CDC-ACM) sample project	33
2.12	ThreadX Low Power sample project	35
2.12.	1 Overview of sample project	35
2.12.	2 Execute sample project	36
2.12.	3 Configuration of ThreadX Low Power by Smart Configurator	38
2.13	Azure Device Update (ADU) sample project	40
Revi	sion History	41



1. Getting Started

To create new Azure RTOS project, the procedure is different between e² studio and IAR EW.

1.1 Creating project using e² studio

1. Launch e² studio, create new project: [File] > [New] > [Renesas C/C++ Project] and select Renesas RX and create a new workspace.

💽 e² studio Launcher 🛛 🕹]
Select a directory as workspace e ² studio uses the workspace directory to store its preferences and development artifacts.	
Workspace: D\demo v Browse	
Use this as the default and do not ask again Becent Workspaces	
Launch Cancel	

Figure 1.1 Workspace Creation Window

2. Select GCC for Renesas RX C/C++ Executable Project or Renesas CC-RX C/C++ Executable Project.

Note : For those who use CC-RX Evaluation Edition. After the trial period, the CC-RX features become limited to restrictions on the linkage size. RX Azure RTOS sample will exceed this restriction. Consider updating CC-RX Professional edition or using GCC for Renesas RX C/C++ Executable.

🕲 New C/C++ I	Project – 🗆 🗙
Templates for	Renesas RX Project
All C/C++	GCC for Renesas RX C/C++ Executable Project $A \subset C ++ Executable Project for Renesas RX Toolchain$
	GCC for Renesas RX C/C++ Library Project GCC for Renesas RX C/C++ Library Project Markow Project for Renesas RX using the GCC for Renesas RX Toolchain.
	Renesas CC-RX C/C++ Executable Project
	Renesas CC-RX C/C++ Library Project
?	< Back Next > Finish Cancel

Figure 1.2 Toolchain Setting Window

- 3. Input the project name.
- 4. Click [Next].



8		- 🗆 X
New Renesas	CC-RX Executable Project	Ď
roject name: d	emo	
🗹 Use default I	ocation	
Location:	D:\demo\demo	Browse
	Create Directory for Project	
Choose file syste	im: default \vee	
Working sets		
Add project	to working sets	New
Working sets:		 Select
1		
? <	Back Next > Finish	Cancel

Figure 1.3 Project Creation Window

- 5. At RTOS, select "Azure RTOS".
- 6. Click Manage RTOS Versions... to download software package.
- 7. At **RTOS Version**, select a version that downloaded at step 6.
- 8. At Target Board, select a board that you are working on. Configurations are automatically set based on the target board.
- 9. Click [Next].

G		- 🗆 X
New	w Renesas CC-RX Executable Project	
Sele	lect toolchain, device & debug settings	
- Too Lan Too Too RTG	olchain Settings anguage: C C ++ olchain: Renesas CC-RX olchain Version: v3.06.00 Manage Toolchains. TOS: Azure RTOS	
RTC	TOS Version: 6.4.0_rel-rx-1.0.0 Manage RTOS Versions	
Der	levice Settings	Configurations
Tary Tary	rget Board: CX-RX6SN V rget Device RSFSGSNEH+FB IInlock Devices Endian: Little V Bank Mode Single Bank V	Create Hardware Debug Configuration E2 Lite (RX) Create Debug Configuration RX Simulator Create Release Configuration
0	Back Nex	xt > Finish Cancel



10. Click [Next].

×
New Renesas CC-RX Executable Project
Use Smart Configurator Use Peripheral Code Generator
Smart Configurator is a single Lue Infinites that controlless the functionalities of Code Generator and II Configurator with the information and the configuration is driven and middleware information configuration is configuration in the information and provide and the configuration is driven and middleware information and provide and provi
⑦ < Back Next > Finish Cancel

Figure 1.5 Coding Assistant Setting Window



- 11. Select an application.
- 12. Click [Finish].



Figure 1.6 Select Application Window

13. Azure RTOS sample project including each component is created.

Ele Edit Navigate Search Project Re	merar Viewr, Rup, Window, Help		
		The call the set of sectors as the residence of the call control of	(A = E = N // A = A = I = A
e e bebug	no hardwarebeuog + +	Carletate - a cross to result to brain	
Project Explorer	C & C T T T W demoisting 1		MCU/MPU Package = = MMU Layout
 Biocholas 	Overview information	Generate Code Generate Report	Assigned O Debuilt Board
> as libs	Allow general pin configuration and pin configu	ration for selected software component	
> 😁 src			
> 🐸 trash	Interrupt		
# demo.scfg	Allow general interrupt configuration and interru	pt configuration for selected software component	
demo HardwareDebug.launch			
	Click here to get more information on User's M	anual, Release Note Application Notes Tool News	
	 Current Configuration 		2 ENESAS
	Selected board/device: R5F565NEDxFC (ROM si	re: 2MB, RAM size: 640KB, Pin count: 176)	T(CIVE 3/353
	Generated location (PROJECT_LOC\): src\smc_ge	n Edit	
	Selected components:		REFERENCE
	Component ^ Versio	Configuration	
	 Azure RTOS ThreadX 6.1.11 	rel threadx(used)	
	 AzureRTOS Object 1.0.10 	4 azurertos_object(used)	· · · · · · · · · · · · · · · · · · ·
	Board Support Packages. (r_bsp) 7.10	r_bsp(used)	
	CMT driver (r_cmt_n) 5.10	r_cmt_rx(used)	
	K	, v	
	Overview Board Clocks System Components Pir	is Interrupts	▶ Legend
Console	ik 🗿 🕪 🖻 💌 (3 • • • • • • • • • • • • • • • • • • •	810
Smart Configurator Output	(Sinc_Ben ()_com xB ()_cosp_inter (opt_com xB ()	0 items	-
M03000004: File modified:src\	smc_gen\r_config\r_bsp_config.h	Description	lype
noocooc. code generación is	auccessiver.p. (deno (deno (sirc (sirc_gen	~	
<		>	
		🧭 demo	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			

Figure 1.7 Created Sample Project Window

- 14. Build project: Select the sample project in the e² studio workspace and right click and select build to build the sample project.
- 15. Make sure that target board is set to Debug mode in Jumper Settings. For the detail, see each board User manual.
- 16. Select Download and Debug to download and start execution of the project. By default, execution stops at a breakpoint set at main.

Note: Other debugger settings may be required depending on the board type you specify.

In the case of Renesas Starter Kit+ for RX65N-2MB: click **Debugger > Connection Settings > Power Target From The Emulator**, and set **No**.

17. Please review the sample descriptions later in this guide for additional setup and expected behavior.

1.2 Creating project using IAR EW

Please refer to following FAQ for the detailed instructions: English : <u>https://en-support.renesas.com/knowledgeBase/20533128</u> Japanese : <u>https://ja-support.renesas.com/knowledgeBase/20533124</u>



2. Sample Project Descriptions

Additional setup and expected behavior of each sample project are described in this section.

2.1 ThreadX sample project

This sample is the standard 8-thread ThreadX example, that illustrates the use of the main ThreadX services, including threads, message queues, timers, semaphores, byte memory pools, block memory pools, event flag groups, and mutexes.

Supported Kits :

- Target Board for RX130
- Renesas Starter Kit for RX140
- Renesas Solution Starter Kit for RX23E-B
- MCB-RX26T Type A/B/C
- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit
- Renesas RX65N Cloud Kit
- Renesas Starter Kit for RX660
- Renesas Starter Kit for RX66T
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit

To run this sample, simply follow these steps (assuming the steps described in the previous section were done):

- 1. Set a breakpoint at any line.
- 2. Select Go to start execution of the sample project.



Figure 2.1 e² studio Debugger Screen





Figure 2.2 IAR EW Debugger Screen

After hitting **Break**, the debugger screen shot above shows various counters incremented by the ThreadX sample as each of the main components of the ThreadX are exercised.

To learn more about Azure RTOS ThreadX, view https://docs.microsoft.com/azure/rtos/threadx/.



2.2 Minimal sample project

This is minimal sample with one thread. It illustrates the usage of main ThreadX service. The sample outputs the message to serial terminal and blinks LED every second.

Supported Kits:

- Target Board for RX130
- Renesas Starter Kit for RX140
- Renesas Solution Starter Kit for RX23E-B
- MCB-RX26T Type A/B/C
- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit
- Renesas RX65N Cloud Kit
- Renesas Starter Kit for RX660
- Renesas Starter Kit for RX66T
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit

To run this sample, simply follow these steps (assuming the steps described in the previous section were done):

[For RX130 Target Board and MCB RX26T]

- 1. Select **Go** to start execution of the sample project
- 2. Open "Renesas Debug Virtual Console" .



Figure 2.3 Renesas Debug Virtual Console



3. As the project runs, it will output "Hello, RX AzureRTOS sample" to serial terminal per one second, and it will blink an LED on the board per one second.



Figure 2.4 Minimal sample output [For target boards except for RX130 Target Board and MCB RX26T]

1. Verify the serial port in your OS's device manager. It should show up as a COM port



Figure 2.5 Device Manager

2. Open your favorite serial terminal program such as Putty and connect to the COM port discovered above. Configure the following values for the serial port:

Baud rate: **115200** Data bits: **8** Parity: **none** Stop bits: **1** Flow control: **none**

- 3. Select Go to start execution of the sample project
- 4. As the project runs, it will output "Hello, RX AzureRTOS sample" to serial terminal per one second, and it will blink an LED on the board per one second.

💻 COM4 - Tera Term V	л		_	×
File Edit Setup Cor	ntrol Window	Help		
Hello, RX AzureR Hello, RX AzureR Hello, RX AzureR	TOS sample. TOS sample. TOS sample	[0] [1] [2]		
Hello, RX AzureR Hello, RX AzureR Hello, RX AzureR	TOS sample. TOS sample. TOS sample.	[3] [4]		
Hello, RX AzureR Hello, RX AzureR	TOS sample. TOS sample.	[5] [6]		
Hello, KX AzureK Hello, RX AzureR Hello. RX AzureR	TUS sample. TOS sample. TOS sample.	L7J [8] [9]		<i>_</i>

Figure 2.6 Minimal sample output



LED
LED4
LED1
LED0
LED2
LED2
LED0
LED0
LED0
LED0
LED1
LED1

Table 2 Assigned LED on the board

2.3 FileX RAM Disk sample project

This sample illustrates the use of the FileX embedded FAT file system. The example creates a small RAMdisk with a sample file and data, and reads the file data back into memory. The debugger can show the data being read.

Supported Kits:

- Target Board for RX130
- Renesas Starter Kit for RX140
- Renesas Solution Starter Kit for RX23E-B
- MCB-RX26T Type A/B/C
- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit
- Renesas RX65N Cloud Kit
- Renesas Starter Kit for RX660
- Renesas Starter Kit for RX66T
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit

To run this sample, simply follow these steps (assuming the workspace is already open):

- Open sample_filex_ram_disk.c and set a breakpoint around Line 201 at if (status != FX_SUCCESS)
- 2. Select Go to start execution of the sample project
- 3. In the **Expression** window for e² studio or **Watch** window for IAR EW, ensure you watch the **local_buffer** variable as expression.









Figure 2.8 IAR EW Debugger Screen

The debugger screen shot above shows the file data read back in the RAM disk sample.

To learn more about Azure RTOS FileX, view https://docs.microsoft.com/azure/rtos/filex/.



2.4 NetX Duo Ping sample project

This sample project illustrates the setup and use of NetX Duo IPv4/IPv6 TCP/IP stack via ping from another node on the local network. By default, this demonstration requests an IP Address via DHCP, and displays the status and assigned IP Address via Terminal program.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit (Ether)
- Renesas RX65N Cloud Kit
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit
- To run this sample project, simply follow these steps (assuming the workspace is already open):
- 5. Verify the serial port in your OS's device manager. It should show up as a COM port



Figure 2.9 Device Manager

6. Open your favorite serial terminal program such as Putty and connect to the COM port discovered above. Configure the following values for the serial port:

Baud rate: **115200** Data bits: **8** Parity: **none** Stop bits: **1** Flow control: **none**

- 7. Select **Go** to start execution of the sample project
- 8. As the project runs you should observe the IP address assigned via DHCP in the output window



Figure 2.10 IP Address Assigned via DHCP

9. The example above shows that the assigned IP address of the RX MCU is 192.168.2.115. When the demonstration is running it can be pinged by any machine on the network. The following is an example of a ping from a Windows machine on the same local network (using the DOS command window).





Figure 2.11 Ping Response

To learn more about Azure RTOS NetX Duo, view https://docs.microsoft.com/azure/rtos/netx/.



2.5 NetX Duo Iperf sample project

This demonstration illustrates TCP and UDP network throughput, using NetX Duo IPv4/IPv6 TCP/IP stack, and the industry-standard Iperf network throughput benchmark, with Jperf GUI. By default, this demonstration requests an IP Address via DHCP, and displays the status and assigned IP Address via Terminal program.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit (Ether)
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit

To run the NetX Duo Iperf Sample project, simply follow these steps (assuming the workspace is already open):

Note: This sample is Ethernet based and therefore assumes an Ethernet cable is connected to the Ethernet connector on the board.

1. Verify the serial port in your OS's device manager. It should show up as a COM port.

Figure 2.12 Device Manager

2. Open your favorite serial terminal program such as Putty and connect to the COM port discovered above. Configure the following values for the serial port:

Baud rate: **115200** Data bits: **8** Parity: **none** Stop bits: **1** Flow control: **none**

- 3. Select **Go** to start execution of the sample project.
- 4. As the project runs you should observe the IP address assigned via DHCP in the output window.

DHCP In Progress

Figure 2.13 IP address assigned via DHCP

5. Once running, simply browse to target IP address (in the screen shot above it is 10.172.14.40) to view the NetX Duo Iperf server page, which provides options for running each Iperf test as well as displays the results of each test. Here is as sample view after browsing 10.172.14.40:



Azure RTOS sample projects using e2 studio or IAR EW

	>
$ \begin{array}{c} \square \text{ NetX IPert Demonstration} \\ \hline \\ $	
O Vot secure 10.172.14.40	<u> </u>
Microsoft Azure	N E T 🗶
NetX IP Address: 10.172.14.40 fe80:0:0:0:0:0:0:3Start UDP Transmit Test Destination IP Address: Test Time(Seconds):Destination Port: Exact Size:Start UDP Receive Test Test Time(Seconds):Start TCP Transmit Test Destination IP Address: Exact Size:Destination IP Address: Start TCP Transmit Test Destination PAddress:Start TCP Transmit Test Destination IP Address: Exact Time(Seconds):10Start TCP Receive Test Test Time(Seconds):Test Time(Seconds):10Start TCP Receive Test Test Time(Seconds):10Start TCP Receive Test Test Time(Seconds):10	Choose a test from the left.

Figure 2.14 NetX Duo Iperf Server Page

Note: Static IP address assignment is also possible by disabling NX_ENABLE_DHCP in the project settings and modifying the default static IP address of 192.168.1.211 in the source file "sample_netx_duo_iperf.c" file. To learn more about Azure RTOS NetX Duo, view https://docs.microsoft.com/azure/rtos/netx/.



2.6 IoT Embedded SDK sample project

This demonstration connects to Azure IoT Hub using Azure IoT middleware for Azure RTOS. This demonstration also publishes the message to IoT Hub every few seconds.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit (Ether/ Cellular)
- Renesas RX65N Cloud Kit
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit

It is also possible to view device properties, view device telemetry, update device twin, call a direct method on device and send cloud-to-device message using Azure IoT Explorer.

Following videos guide how to set up and run this Azure RTOS sample project in detail. This video uses CK-RX65N to introduce, but the setup follow is common to every board.

Azure RTOS Tutorial (1/3) CK-RX65N Azure RTOS Tutorial (2/3) CK-RX65N: Program Build Azure RTOS Tutorial (3/3) CK-RX65N: Cloud Operation

Projects with cellular connectivity have "with EWF" at the end of the project name on Select Application Window.

 Prepare Azure resources such as creating an IoT Hub and registering an IoT device by referring Microsoft document.
 For details, places refer to the Application Nate (DX65N Groups Visualization of Sensor Data using

For details, please refer to the <u>Application Note (RX65N Group: Visualization of Sensor Data using</u> <u>RX65N Cloud Kit and Azure RTOS</u>), specifically chapters 3.1.

- 2. Confirm that you have the copied the following values to use in the next step.
 - hostname
 - deviceID
 - primaryKey
- 3. Open **sample_config.h** to set the Azure IoT device information constants to the values that you saved in step 2.

Constant name	Value
HOST_NAME	{Your loT hub hostName value}
DEVICE_ID	{Your deviceID value}
DEVICE_SYMMETRIC_KEY	{Your primaryKey value}

4. **[Wi-Fi]** Open **main.c** to set the Wi-Fi network parameters when you use the boards of which connectivity is Wi-Fi.

Constant name	Value
WIFI_SSID	{Your Wi-Fi SSID value}
WIFI_PASSWORD	{Your Wi-Fi password}

5. **[Cellular]** Open **scfg file in your project** and choose components on the tab. Then click **ewf** in components on the left side and **set** your **SIM APN** in "The SIM operator APN".



Software component configuration
Components 🚵 🚵 $\mathbb{P}_{\mathbb{Z}} \boxdot \mathfrak{P}_{\mathbb{Z}} \mathfrak{D}$ Configure
yper filter test yper filter test

- 6. **[Ethernet]** You don't need to set specific parameters when you use the boards of which connectivity is ethernet.
- 7. Verify the serial port in your OS's device manager. It should show up as a COM port.

Figure 2.15 Device Manager

8. Open your favorite serial terminal program such as Putty and connect to the COM port discovered above. Configure the following values for the serial port:

Baud rate: **115200** Data bits: **8** Parity: **none** Stop bits: **1** Flow control: **none**

- 9. Build project
- 10. Select **Download and Debug** to download and start execution of the project
- 11. As the project runs, the demo prints out status information to the terminal output window. The demo also publishes the telemetry message to IoT Hub every few seconds. Check the terminal output to verify that messages have been successfully sent to the Azure IoT hub.

Initializing Wi-Fi Wi-Fi connected. IP address: 192.168.0.5 Mask: 255.255.255.0 Gateway: 192.168.0.1 Syncing time Time server IP address: 132.246.148.30 Time: 1632723050 Time: 1632723050 Time: Sync successful. IoTHub Host Name: Connected to IoTHub. Telemetry message send: ["Message ID":0].
Telemetry message send: ["Message ID":4]. Telemetry message send: ["Message ID":5]. Telemetry message send: ["Message ID":6]. Telemetry message send: ["Message ID":7].

Figure 2.16 Status Information and Telemetry Message



You can use the **Azure IoT Explorer** to view and manage the properties of your devices. In the following steps, you'll add a connection to your IoT hub in IoT Explorer.

- 1. Download and install latest (above v0.14.5) Azure IoT Explorer from: <u>https://github.com/Azure/azure-iot-explorer/releases</u>
- Copy the connection string: <u>Microsoft Azure Portal</u> > sign in > select your IoT Hub > [Share access policies] > [iothubowner] > [Primary connection string].

Home >			iothubowner	×
<u> </u>	Shared access	policies 🖈 …		
P Search (Ctrl+/)	Shared access policies may be used to g	generate security tokens to consume IoT hub functional	Regenerate primary key Regenerate secondary key Swap keys	
Security settings	 Connect using shared access policies 		••••••	D
🐍 Identity	🗟 Save 🏷 Discard change		Secondary key	
Shared access policies	 Allow 		•••••	D
Networking	Deny		Primary connection string	٦
🔎 Certificates			HostName=kuwada-rx65n-cloud-kit-1.azure-devices.net;SharedAccessKeyName=io 🧠	Ð
Defender for IoT	Manage shared access policies		Secondary connection string	_
Overview	+ Add shared access policy O Re	alresh III Delete	······ •	D
() Security Alerts	Policy Name	Permissions	Permissions	
E Recommendations	D tabularian	Desides Dead Desides With Control Conner	Registry Read	
Settings	Iothubowher	Registry Read, Registry Write, Service Connect	Registry Write	
Monitoring	service	Service Connect	Service Connect	
			Device Connect	

Figure 2.17 Primary Connection String

- 3. In Azure IoT Explorer, select **IoT hubs > Add connection**.
- 4. Paste the connection string into the **Connection string** box.
- 5. Select Save.

Home > IoT hubs × • v6 112 hab been interace. Please clock here is down • v6 112 hab been interace. Please clock here is down • v6 v6 v12 hab been interace. Please clock here is down • v6 v6 v12 hab been interace. Please clock here is down • v6 v6 v12 hab been interace. Please clock here is down • v6	Azure for Explorer (p	neview)		Settings
○ d112 has been released. Revese dick here to develop and Play ○ Hold holds Shared access p Shared access policy here do 1 get an IOT hub connection string? Please do not save your hub connection string? Please do not save your hub connection string? View devices int Abared access policy name Intro-pp-but Intro-pp-but Host name Shared access policy key	Home > IoT hubs			×
Add connect A	() v0.11.2 has been released. (Please click here to down	Edit connection string	
In Thubs Shared access p \$\$^{0}\$ toT Plug and Play Connection String Connection String Where do I get an IoT hub connection string? Please do not save your hub connection string to any unsafe locations Host name Rost name Host name Host name	=	+ Add connect		1.1
e ^o Tot Plug and Play Connection String Connection String Where do I get an IoT hub connection string? Please do not save your hub connection string to any unsafe locations Wew devices in Assume devices net View devices in Essent access policy name rtos-pp-but Iothubowner Host name Shared access policy key Host name Iothubowner	and IoT hubs	Shared access p		
Connection Strip Please do not save your hub connection string? Please do not save your hub connection string to any unsafe locations View devices in Shared access policy name inchubowner Host name Shared access policy key Most name	S IoT Plug and Play			
Please do not save your hub connection string to any unsafe locations View devices int Shared access policy name rtos-pp-bug Iost name Shared access policy key		Connection Strin	Where do I get an IoT hub connection string?	
View devices ini Host name Host name Host name Host name Host name Host name Host name Host name C			Please do not save your hub connection string to any unsafe locations	
Shared access policy name © rtos-pp-bug iothubowner Host name Shared access policy key		View devices in t	Host name azura-devices nat	
rtos-pp-bu; iothubowner Host name Shared access policy key			Shared access policy name	
Host name Shared access policy key 💿		rtos-pp-buc	iothubowner	D
		Host name	Shared access policy key	
		Tiost nume		٥

Figure 2.18 Azure IoT Explorer

6. If the connection succeeds, the Azure IoT Explorer switches to a Devices view and lists your device.

To view device properties using Azure IoT Explorer:

- 1. Select the link for your device identity. IoT Explorer displays details for the device.
- 2. Inspect the properties for your device in the **Device identity** panel.



Home > > Devices > MyDevKit > Device identity □ Device identity □ Device identity □ Device identity □ Device identity □ Device identity □ □ □ Telemetry MyDevKit □ > Direct method Primary key □ □ © Cloud-to-device m □ □ % Module identity Secondary key □ □ %* IoT Plug and Play c Primary connection string □ □	Azure IoT Explorer (preview) File Edit View Window Help Azure IoT Explorer (pr	, review)	Notifications	Setting
■ Save Manage keys ∨ ■ Device identity ● Device identity > Direct method ● Primary key ◎ ■ Cloud-to-device m Secondary key ◎ ● S ^d IoT Flug and Flay c Primary connection string ◎	Home >	> <u>Devices</u> > MyDevKit > Device identity		
■ Device identity ● Device identity > Direct method ● Primary key ◎ □ Cloud-to-device m ♀ IooT Plug and Play c ● Primary connection string ◎	=	🗟 Save 🔍 Manage keys 🗠		
□ Telemetry Device ID ○ □ >> Direct method MyDevKit ID >> Cloud-to-device m □ □ Primary connection string ○ □	 Device identity Device twin 	Device identity		
S Direct method Primary key ○ Cloud-to-device m C Module identity Secondary key ○ Primary connection string ○ Primary connection string ○	🖵 Telemetry	Device ID 0		
Cloud-to-device m Control to the secondary key Control to the secondary Control to the secondary key Control to the sec	✓ Direct method	Primary key 💿		
Module identity Secondary key © Secondary key © Image: Constraint of the secondary key © Primary connection string ©	Cloud-to-device m			•
S ^r 101 Plug and Play c Primary connection string ◎	Module identity	Secondary key 🕖		•
	S IoT Plug and Play c	Primary connection string ①		

Figure 2.19 Azure IoT Explorer

To view device telemetry using Azure IoT Explorer:

- 1. In IoT Explorer select **Telemetry**. Confirm that **Use built-in event hub** is set to Yes.
- 2. Select Start.
- 3. View the telemetry as the device sends messages to the cloud.

=	Stop Show system properties 🔋 Clear events	
 Device identity Device twin 	Telemetry 🛈	
C Telemetry	Consumer group	
✓ Direct method	Use built-in event hub	
Cloud-to-device m	Receiving events	
St Module identity), monoperative": "000-07-3070756:03.0362", "propertyM: "valueA", "propertyM: "valueA",)	
	3:55:58 PM, 07/30/2020:	
	<pre>{ "Dody": { "Hessage ID": 15 }, requesedTime": "2020-07-J0T07:55:58.0402", "propertien"; "propertien"; "propertyA": ValueA", "propertyM": ValueA", "propertyM": ValueA", "propertyM": ValueA", "propertyM": ValueA", "propertyM": ValueA", "propertyM": ValueA", "propertyM: ValueA",</pre>	

Figure 2.20 Telemetry Message

To update device twin using Azure IoT Explorer:

- 1. In IoT Explorer select **Device twin**.
- 2. Modify the **desired** section of the Device twin, you can add a custom twin:



3. Select Save.



Azure IoT Explorer (preview)	🗘 Notifications 🛛 🎯 Setti
Home >	> <u>Devices</u> > MyDevKit > Device twin	
 Device identity 	🖒 Refresh 📓 Save	
🔁 Device twin	Device twin ①	
 □ Telemetry > Direct method □ Cloud-to-device m ♀ Module identity ∞ IoT Plug and Play c 	<pre>is</pre>	.86854142*

Figure 2.21 Device Twin

- 4. View the notification for the device twin update status.
- 5. In the terminal output window, you can view the desired device twin properties are received.

Telemetry message send: {"Message ID":68}.	^
Telemetry message send: {"Message ID":69}.	
Telemetry message send: { "Message ID":70}.	
Telemetry message send: {"Message ID":71}.	
Telemetry message send: {"Message ID":72}.	
Telemetry message send: { "Message ID":73}.	
Telemetry message send: { Message ID : 74}.	
Telemetry message send: {"Message ID" 75}	
Receive desired property call: {"weather":{"temperature":"25"},"\$version":2}	
Telemetry message send. ("Message 1D".70).	
Telemetry message send: {"Message ID":77}.	
Telemetry message send: {"Message ID":78}.	
Telemetry message send: {"Message ID":79}.	
Telemetry message send: {"Message ID":80}.	
Telemetry message send: {"Message ID":81}.	
Telemetry message send: {"Message ID":82}.	
Telemetry message send: {"Message ID":83}.	
Telemetry message send: { Message ID :84}.	
LiTelenstur verses cond. ["Message Th".011	

Figure 2.22 Received Desired Device Twin Properties

To call a direct method on device using Azure IoT Explorer:

You can also use Azure IoT Explorer to call a direct method that you have implemented on your device. Direct methods have a name, and can optionally have a JSON payload, configurable connection, and method timeout. To call a direct method in Azure IoT Explorer:

- 1. In IoT Explorer select **Direct method**.
- 2. Send a direct method to mimic the device reboot with payload. The device will receive and output the payload as dummy data.
 - Method name: reboot
 - Payload: {"timeout": 500}

=	✓ Invoke method	
 Device identity Device twin 	Direct method ①	
C Telemetry	Method name *	Î
> Direct method	Payload 💿	
Cloud-to-device m	{ 'timeout : 500}	
	Connection timeout in seconds 0	A
	-0	10

Figure 2.23 Direct Method



3. Select Invoke method.

4. In the terminal output window, you can view the method is invoked on the IoT Device.

Telemetry message send: {"Message ID":227}.	
Telemetry message send: {"Message ID":228}.	
Telemetry message send: {"Message ID":229}.	
Telemetry message send: {"Message ID":230}.	
Telemetry message send: {"Message ID":231}.	
Telemetry message send: {"Message ID":232}.	
Telemetry message send: {"Message ID":233}.	
Telemetry message send: {"Message ID":234}.	
Telemetry message send: {"Message ID":235}.	
Telemetry message send: {"Message ID":236}.	
Telemetry message send: {"Message ID":237}.	
Telemetry message send: {"Message ID":238}.	
Telemetry message send: {"Message ID":239}.	
Telemetry message send: { Message ID:240}.	
Telemetry message send: { Message ID : 241 }.	
lelemetry message send: { Ressage 1D : 242}.	
Telemetry message send: { Hessage ID : 243}.	
Description between the state and st	
Receive method call: reboot, with payload:{ timeout :500}	

Figure 2.24 Invoked Method

To send cloud-to-device message using Azure IoT Explorer:

- 1. In IoT Explorer select Cloud-to-device message.
- 2. Enter the message in the Message body:

{ "Hello": "Azure RTOS" }	
- J	

3. Check Add timestamp to message body.

	Send message to device	
Device identity	Cloud-to-device message 📀	
🖵 Telemetry	Message body 💿 (''Hello': ''Azure RTOS'')	
Direct method Cloud-to-device		
X Module identity	Add timestamp to message body	
S ^Q IoT Plug and Play c	Properties Add custom property Add system property Belete	

Figure 2.25 Cloud-to-device message

- 4. Select Send message to device.
- 5. In the terminal output window, you can view the message is received by the IoT Device.



Figure 2.26 Received Message



2.7 IoT Embedded SDK PnP sample project (Available up to Azure RTOS v.6.2.1_rel-rx-2.0.0)

This demonstration connects to Azure IoT Hub using Azure IoT middleware for Azure RTOS. This demonstration also publishes the message to IoT Hub every few seconds.

It is also possible to view device properties, view device telemetry, update device twin and call a direct method on device using Azure IoT Explorer.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit (Ether/ Cellular)
- Renesas RX65N Cloud Kit
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit

To run this project, simply follow **2.5 IoT Embedded SDK sample project**.

Moreover, this sample can interact with IoT Plug and Play components using Azure IoT Explorer.

To interact with IoT Plug and Play components using Azure IoT Explorer:

You can use Azure IoT Explorer to interact with IoT Plug and Play components.

Azure IoT explorer needs a local copy of the model file that matches the **Model ID** your device sends. The model file lets Azure IoT explorer display the telemetry, properties, and commands that your device implements.

To use the Azure IoT explorer to verify the IoT Plug and Play device application is working:

- 1. In IoT Explorer, select the IoT Plug and Play Settings.
- 2. Select Add and select Public Repository.
- 3. Select Save.

=		Save + Add \sim 7 Revert ? Help
뮵 loT hubs		Model repository locations specify where the application looks to find IoT Plug and Play model
		definitions. Locations are saved to application storage and can be edited or removed at any time.
		Help:
I Notification Center		What is loT Plug and Play
		Model Repository Locations:
		We'll look for your model definition in the following order. Please drag and drop to change it. Click 'Add' to enable more ways to can resolve your model definitions.
		Before enabling us to retrieve model definition from a local folder, please read Microsoft Privacy Statement
	1	Public Repository X The ict-plugandplay-models GitHub repository includes DTDL models that are made publicly available on https://devicemodels.azure.com. Repository endpoint https:// devicemodels.azure.com

Figure 2.27 IoT Plug and Play Setting



- 4. On the **IoT hubs** page, click on the name of the hub you want to work with. You see a list of devices registered to the IoT hub.
- 5. Click on the **Device ID** of the device you created previously.
- 6. The menu on the left shows the different types of information available for the device.
- 7. Select **IoT Plug and Play components** to view the model information for your device.

=	C) Refresh
Device identity	IoT Plug and Play components IoT Plug and Play documentation
🔁 Device twin	
S Direct method	Step 1. four device has been discovered as a lot plug and play device Model ID
Cloud-to-device message	dtmi:com:example:Thermostat:4
🛠 Module identities	Step 2. We've resolved your IoT Plug and Play model
$\mathcal{S}^{\mathcal{G}}$ loT Plug and Play components	You model definition has been resolved from: Public Repository 🛞 Configure
	Step 3. Continue your IoT Plug and Play journey by drilling down to each component
	If you have defined 'Property', 'Command' or 'Telemetry' in model dtmi:com:example:Thermostat;4, you would be able to see 'Default component' in the table below. If you have defined 'Component', you would be able to see a list of components down below.
8.	Components Model content

Figure 2.28 Model Information

- 9. You can view the different components of the device. The default component and any additional ones. Select a component to work with.
- 10. Select the **Telemetry** page and then select Start to view the telemetry data the device is sending for this component.
- 11. Select the **Properties (read-only)** page to view the read-only properties reported for this component.
- 12. Select the **Properties (writable)** page to view the writable properties you can update for this component.
- 13. Select a property by its name, enter a new value for it, and select Update desired value.
- 14. To see the new value show up select the **Refresh** button.
- 15. Select the Commands page to view all the commands for this component.
- 16. Select the command you want to test set the parameter if any. Select **Send command** to call the command on the device. You can see your device respond to the command in the command prompt window where the sample code is running.

	Interface Properties (read-only) Properties (writable) Commands Telemetry
Device identity	Č) Refresh
🔁 Device twin	You model definition has been resolved from: Public Repository 🛞 Configure
C Telemetry	Interface Id
✓ Direct method	dtmi:com:example:Thermostat:4
☑ Cloud-to-device message	Name
	Thermostat D
🛠 Module identities	Description
🔗 IoT Plug and Play compor	Reports current temperature and provides desired temperature control.
	<pre>1 * [0 2 "@context": "dtmi:dtdl:context;2", 3 "@id": "dtmi:com:example:Thermostat;4", 4 "@type: "Interface", 5 "displayMamed: "Thermostat", 6 "description: "Reports current temperature and provides desired temperature control.", 7 "extends: "dtmi:azure:iot:deviceUpdateContractModel;2", 8 " "contents": [</pre>

Figure 2.29 IoT Plug and Play Components



2.8 GUIX 8bpp/16bpp/16bpp_draw2d sample project

This demonstration illustrates Washing Machine application using advanced GUIX features such as:

- Widget creation
- Creating multiple screens inside the main screen
- Attaching and detaching the child screen when you switch screens
- Double-buffer toggle control for screen transition without tearing
- Radial slider, vertical and horizontal slider creation
- Running animation

It also illustrates 2 kinds color depth and use of 2D drawing engine (DRW2D) on RX family.

- sample_guix_8bpp: sample for display of size 480 * 272 with 8 bits color look-up table (CLUT8).
- sample_guix_16bpp: sample for display of size 480 * 272 with 16 bits RGB 565.
- sample_guix_16bpp_draw2d: sample for display of size 480 * 272 with 16 bits RGB 565 with 2D drawing engine.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- RX72N Envision Kit

To run each GUIX Sample project, simply follow these steps (assuming the steps described in the previous section were done):

1. Select **Go** to start execution of the demonstration. As the project runs you should observe Washing Machine GUI on board TFT panel. The four different screens are demonstrated as:



Figure 2.30 Main Screen



Figure 2.31 Garments selection screen





Figure 2.32 Water level selection screen

Microsoft Azure 12:31 PM Tuesday	≜
Feb 21,2017	Derature
Cold Vish Spin Rinse Spin	Start Garments Water Level Vivater Level Siss 70 ° Temperature OP Power Off

Figure 2.33 Temperature selection screen

The application demonstrates the simulation of the Washing Machine controller from the GUI perspective. This project initializes the GUIX system, configures the GUIX drivers, initializes Canvas, creates screens using widget creation APIs, starts the GUIX and handles the Touch Events from the Touch driver. All these

are done from the Application Thread.

To learn more about Azure RTOS GUIX, view https://docs.microsoft.com/azure/rtos/guix/.



2.9 USBX device CDC-ACM Class sample project

This demonstration illustrates the setup and use of USBX device CDC-ACM Class to communicate with the host as a serial device. This project initializes the USBX system and device stack, set the parameters for callback when insertion/extraction of a CDC device, read from the CDC class and write to the CDC instance using device CDC-ACM APIs.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit
- Renesas RX65N Cloud Kit

Before build the sample and run, you need to connect the USB0 Function on Renesas Starter Kit+ for RX65N-2MB to your computer using the USB-MiniB cable: (assuming Renesas Starter Kit+ for RX65N-2MB is specified as Target Board)



Figure 2.34 USB0 Function on Renesas Starter Kit+ for RX65N-2MB

To run the device CDC-ACM Sample project, simply follow these steps (assuming the steps described in the previous section were done):

- 1. Select Go to start execution of the demonstration.
- 2. Verify the serial port in your OS's device manager. It should show up as a COM port for the CDC-ACM device.

Figure 2.35 Device Manager

- 3. Open your favorite serial terminal program such as Putty and connect to the COM port discovered above. In this sample project, it is not necessary to set any other settings on the terminal program.
- 4. As the project runs, you should be able to observe "abcdef" returned from the CDC-ACM device when you input **enter** key to the CDC-ACM device via the terminal.





Figure 2.36 Serial Terminal Window

To learn more about Azure RTOS USBX, view https://docs.microsoft.com/azure/rtos/usbx/.



2.10 USBX Host Mass Storage Class sample project

This demonstration illustrates the setup and communication with MSC device (USB flash drive) using USBX HMSC. The sample program initializes the FileX, USBX system and USB driver stack. When a MSC device is inserted, it reads and writes a file to MSC device using device FileX APIs.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit
- Renesas RX65N Cloud Kit
- Renesas Starter Kit+ for RX671
- 1. Change the jumper pins (J7 and J16) on Renesas Start Kit+(RSK) for RX65N-2MB to set to USB Host mode. (assuming Renesas Starter Kit+ for RX65N-2MB is specified as Target Board)

Note: Jumper pin numbers are different for each RSK.

- 2. Build USBX HMSC sample project and run.
- 3. Connect MSC device to USB Standard A connector (red frame) on RSK.



Figure 2.37 USB Standard A Connector on Renesas Starter Kit+ for RX65N-2MB

When the USBX HMSC driver recognizes that MSC device is connected, the sample application program creates a "counter.txt" file to MSC device using FileX API.

- 4. Disconnect MSC device from RSK and connect MSC drive to PC.
- 5. Confirm that "counter.txt" file is generated at the root folder in the MSC device.



Azure RTOS sample projects using e2 studio or IAR EW

→ I 🗹 🗖 🔫	Share View Dr	Nanage D:\	
Pin to Quick Cop access	Paste Shortcut	Move Copy to * Copy Organize	New ite New folder New
$\leftarrow \rightarrow \checkmark \uparrow$	→ (D:) USB Drive	~	ට 🔎 Sear
I My PC: 이가 3D Obje 이 Desktog 한 Docum 나 Downlo	ents ads	counter.txt	

Figure 2.38 Root Folder in MSC Device

6. Open "counter.txt" file using the binary editor on PC. It contains count up numbers from 0x0000 to 0x00FF from the address 0x00000000 as following.

		0122456789ARCDEE
		0120400700ADODEL
00000010 08 00	0 09 00 0A 00 0B 00 0C 00 0D 00 0E 00 0F 00	
00000020 10 00	0 11 00 12 00 13 00 14 00 15 00 16 00 17 00	
00000030 18 00	0 19 00 1A 00 1B 00 1C 00 1D 00 1E 00 1F 00	
00000040 20 00	0 21 00 22 00 23 00 24 00 25 00 26 00 27 00	.!.″.#.\$.%.&.'.
00000050 28 00	0 29 00 2A 00 2B 00 2C 00 2D 00 2E 00 2F 00	(.).*.+.,/.
00000060 30 00	0 31 00 32 00 33 00 34 00 35 00 36 00 37 00	0.1.2.3.4.5.6.7.
00000070 38 00	0 39 00 3A 00 3B 00 3C 00 3D 00 3E 00 3F 00	8.9.:.;.<.=.>.?.
	0 41 00 42 00 43 00 44 00 45 00 46 00 47 00	W.A.B.U.D.E.F.G.
00000090 48 00	U 49 UU 4A UU 4B UU 4C UU 4D UU 4E UU 4F UU	H.I.J.K.L.M.N.U.
UUUUUUAU 50 00	0 51 00 52 00 53 00 54 00 55 00 56 00 57 00 0 50 00 54 00 55 00 50 00 50 00 55 00 55 00	Г.U.K.S.I.U.V.W. У У 7 Г У 1 ^
		∧.I.∠.L.∓.J
00 83 0000000		.a.b.c.d.e.r.g.
00000E0 70 00		parstuvw
000000F0 78 00	0 79 00 7A 00 7B 00 7C 00 7D 00 7F 00 7F 00	x.v.z.{. .}.~
00000100 80 00	0 81 00 82 00 83 00 84 00 85 00 86 00 87 00	
00000110 88 00	0 89 00 8A 00 8B 00 8C 00 8D 00 8E 00 8F 00	
00000120 90 00	0 91 00 92 00 93 00 94 00 95 00 96 00 97 00	
00000130 98 00	0 99 00 9A 00 9B 00 9C 00 9D 00 9E 00 9F 00	
00000140 A0 00	0 A1 00 A2 00 A3 00 A4 00 A5 00 A6 00 A7 00	[.]ヲ.ァ.
00000150 10 00	0 40 00 44 00 40 00 40 00 40 00 4E 00 4E 00	A data of the distance of the

Figure 2.39 Content of "counter.txt"

- 7. Disconnect MSC device from PC and connect the MSC device to RSK. This sample program reads "counter.txt" in MSC device and adds the count up data from the address (0x00000200) in this file.
- 8. Disconnect MSC device from RSK and connect the MSC drive to PC.
- 9. Open "counter.txt" file using the binary editor on PC. It contains count up numbers from 0x0000 to 0x00FF from the address 0x00000200 as following.

ADDRESS	<u> 00 01 02 03 04 C</u>	5 06 07 08 09 0A OB OC OD OE (F 0123456789ABCDEF	
00000200	00 01 01 01 02 0	1 03 01 04 01 05 01 06 01 07 (1	
00000210	08 01 09 01 0A C	1 OB 01 OC 01 OD 01 OE 01 OF (1	
00000220	10 01 11 01 12 0	1 13 01 14 01 15 01 16 01 17 (1	
00000230	18 01 19 01 1A C	1 1B 01 1C 01 1D 01 1E 01 1F (1	
00000240	20 01 21 01 22 0	1 23 01 24 01 25 01 26 01 27 0	1 .!.".#.\$.%.&.'.	
00000250	28 01 29 01 2A C	1 2B 01 2C 01 2D 01 2E 01 2E 0	1 (.).*.+/.	
00000260	30 01 31 01 32 0	1 33 01 34 01 35 01 36 01 37 0	1 0.1.2.3.4.5.6.7.	
00000270	38 01 39 01 34 0	1 3B 01 3C 01 3D 01 3E 01 3E 0	1 89::<=>?	
00000280		1 43 01 44 01 45 01 46 01 47 0	1 @ARCDEEG	
00000290		1 4B 01 4C 01 4D 01 4E 01 4E 0		
00000200	50 01 51 01 52 0			
00000280	58 01 59 01 54 0	1 5B 01 5C 01 5D 01 5E 01 5E 0	1 ¥ Y 7 [¥] ^	
00000200			1 `abcdefg	
00000200	60 10 10 10 00 00 00 00 00 00 00 00 00 00		1 hiiklmpa	
00000200	70 01 71 01 72 0		1 n.r.j.k.i.m.n.o.	
00000220		1 70 01 70 01 70 01 70 01 77 0 1 70 01 70 01 70 01 7E 01 7E 0	1	
000002F0	00 01 78 01 7A 0		1 X.Y.Z.L.I.J	
00000300			1	
00000310	88 UI 89 UI 8A U		1	
00000320	90 01 91 01 92 C	1 93 UI 94 UI 95 UI 96 UI 97 U		
00000330	98 01 99 01 9A C	I AR OL AC OL AD OL AF OL AF (
00000340	AU UT AT OT AZ C	I A3 UI A4 UI A5 UI A6 UI A7 U		
0000000370		1 ND 01 NC 01 ND 01 NE 01 NE 0	1 / 1 1 6 - 1 6	

Figure 2.40 Content of "counter.txt"



10. By repeating steps 8 and 9 above, the sample program keeps updating count data to "counter.txt" file in the MSC device.

To learn more about Azure RTOS USBX, view <u>https://docs.microsoft.com/azure/rtos/usbx/</u>.



2.11 USBX Host Communication Device Class (CDC-ACM) sample project

This demonstration illustrates the setup and communication with HCDC device using USBX HCDC driver (HCDC-ACM). This project initializes the USBX system and USB driver, set the parameters for callback when insertion/extraction of a CDC device, read from the CDC class and write to the CDC instance using Host CDC-ACM APIs.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit
- Renesas RX65N Cloud Kit
- Renesas Starter Kit+ for RX671
- 1. Change the jumper pins (J7 and J16) on Renesas Start Kit+(RSK) for RX65N-2MB to set to USB Host mode. (assuming Renesas Starter Kit+ for RX65N-2MB is specified as Target Board)

Note: Jumper pin numbers are different for each RSK.

- 2. Build USBX HCDC sample project and run.
- 3. Connect CDC device to USB Standard A connector (red frame) on RSK.



Figure 2.41 USB Standard A Connector on Renesas Starter Kit+ for RX65N-2MB

Note:

Please connect this CDC device to a PC (Windows) via UART and start the terminal program on this PC.

- 4. When user input key data on the terminal, the input key data send to CDC device via UART.
- 5. CDC device received key data sends the key data to USB Host CDC device (RSK) via USB.
- 6. USB Host CDC device (RSK) received key data sends the key data to CDC device. (Loopback)
- 7. CDC device received key data sends to terminal program on PC via UART.
- 8. User is able to observe key data on terminal program.





Figure 2.42 Serial Terminal Window

To learn more about Azure RTOS USBX, view https://docs.microsoft.com/azure/rtos/usbx/.



2.12 ThreadX Low Power sample project

This sample project illustrates how to use ThreadX's Low Power feature. You can confirm the transition to and resume from the following low power modes supported by the device using the Low Power Consumption Device Driver Module (r_lpc_rx).

Kits	Target Board for RX130	Renesas Starter Kit+ for RX65N- 2MB	
	Renesas Solution Starter Kit for RX23E-B	CK-RX65N cloud kit	
		Renesas RX65N Cloud Kit	
		Renesas Starter Kit+ for RX671	
		RX72N Envision Kit	
Device	RX130, RX140, RX23E-B	RX65N, RX651, RX660, RX72N, RX671	
Supported low power mode	Sleep Mode	Sleep Mode	
	Deep Sleep Mode	Software Standby Mode	
	Software Standby Mode	Deep Software Standby Mode	

2.12.1 Overview of sample project

- 1. The sample project creates one thread **thread_0**. The **thread_0** turns on the LED when it starts.
- 2. After executing for about 3 seconds, suspend the own thread by **tx_thread_suspend**.
- 3. Since there is no other thread to run, **Demo_LowPower_Enter** configured in ThreadX "Enter low power function" configuration is called from **tx_low_power_enter** of ThreadX.
- 4. **Demo_LowPower_Enter** turns off the LED and transitions to the low power consumption mode.
- 5. The low power consumption mode is resumed by the interruption of pressing the user switch. The interrupt handler **Demo_callback** is called and **tx_thread_resume** resumes **thread_0**. At this point, **thread_0** does not run.
 If it has transitioned to the deep software standby made, it will be resumed by the user switch press.

If it has transitioned to the deep software standby mode, it will be resumed by the user switch press interrupt or RTC alarm interrupt and reboots from the reset vector.

- Next, the Demo_LowPower_Exit configured in the ThreadX "Exit low power function" configuration is called from tx_low_power_exit of ThreadX. Demo_LowPower_Exit turns on the LED and returns to ThreadX.
- 7. The resumed **thread_0** runs.
- 8. Repeat the transition to the same low power consumption mode in steps 2 to 7 three times in total and execute all low power consumption modes in the following order.

For RX130, RX140 and RX23E-B:

Sleep Mode (3 times) => Deep Sleep Mode (3 times) => Software Standby Mode (3 times)

For RX65N, RX651, RX660, RX72N, RX671:

Sleep Mode (3 times) => Software Standby Mode (3 times) => Deep Software Standby Mode (1 time)



The figure shows the execution flow from suspending the thread_0 with tx_thread_suspend to resuming.



Figure 2.43 Execution Flow after tx_thread_suspend (&thread_0)

2.12.2 Execute sample project

To run the sample project, simply follow these steps for each board:

Target Board for RX130, Renesas Starter Kit for RX140 and Renesas Solution Starter Kit for RX23E-B:

- 1. Select Launch to download the program.
- 2. Select **Resume** to start execution of the project. The program stops at the breakpoint of main function.
- 3. Select Resume to restart.
- 4. The program turns LED0(RX130 and RX140)/LED1(RX23E-B) on and runs for 3 seconds.
- 5. The program turns LED0(RX130 and RX140)/LED1(RX23E-B) off and transitions to sleep mode. e² studio status bar will change from Running to Sleeping as below:

Sleeping

- 6. The program is resumed by pressing the user switch (SW1). This cycle is repeated 3 times.
- 7. Similarly, transitions to deep sleep mode and resume by pressing the user switch is repeated 3 times. e² studio status bar will change from Running to Sleeping as below:

Sleeping

8. Similarly, transitions to software standby mode and resume by pressing the user switch is repeated 3 times. e² studio status bar will change from Running to Standby as below:

Standby

- 9. Repeat from sleep mode to software standby mode.
 - (*) e2 studio status bar when sleep mode and deep sleep are the same. So please check MSTPCRC.DSLPE register value before executing wait instruction.
 - sleep mode: MSTPCRC.DSLPE =0
 - deep sleep: MSTPCRC.DSLPE =1

RX65N Cloud Kit:

- 1. Select Launch to download the program.
- 2. Select **Resume** to start execution of the project. The program stops at the breakpoint of main function.
- 3. Select Resume to restart.



- 4. The program turns LED1 on and runs for 3 seconds.
- 5. The program turns LED1 off and transitions to sleep mode. e² studio status bar will change from Running to Sleeping as below:

Sleeping

- 6. The program is resumed by pressing the user switch. This cycle is repeated 3 times.
- 7. Similarly, transitions to software standby mode and resume by pressing the user switch is repeat 3 times. e² studio status bar will change from Running to Standby as below: ^(*)

Standby

8. The program transitions to deep software standby. e² studio status bar will change from Running to Standby as below: (*)

Standby

- 9. The program reboots by pressing the user switch.
- (*) e2 studio status bar when deep software standby and software standby are the same. So please check SBYCR.SSBY and DPSBYCR.DPSBY register value before executing wait instruction.
 - software standby: SBYCR.SSBY=1, DPSBYCR.DPSBY=0
 - deep software standby: SBYCR.SSBY=1, DPSBYCR.DPSBY=1

Renesas Starter Kit+ for RX65N-2MB, Renesas Starter Kit for RX660, Renesas Starter Kit for RX671, RX72N Envision Kit and CK-RX65N:

- 1. Select Launch to download the program.
- 2. Select **Resume** to start execution of the project. The program stops at the breakpoint of main function.
- 3. Select **Resume** to restart.
- 4. The program turns LED (usually LED0) on and runs for 3 seconds.
- 5. The program turns LED off and transitions to sleep mode. e² studio status bar will change from Running to Sleeping as below:

Sleeping

- 6. The program is resumed by pressing the user switch (usually SW1). This cycle is repeated 3 times.
- 7. Similarly, transitions to software standby mode and resume by pressing the user switch is repeat 3 times. e² studio status bar will change from Running to Standby as below: ^(*)

Standby

8. The program transitions to deep software standby. e² studio status bar will change from Running to Standby as below: ^(*)

Standby

9. The program reboots by RTC alarm interrupt after about 30 seconds.

(*) e2 studio status bar when deep software standby and software standby are the same. So please check SBYCR.SSBY and DPSBYCR.DPSBY register value before executing wait instruction.



- software standby: SBYCR.SSBY=1, DPSBYCR.DPSBY=0
- deep software standby: SBYCR.SSBY=1, DPSBYCR.DPSBY=1

2.12.3 Configuration of ThreadX Low Power by Smart Configurator

• You can develop own system low power operation for your product referring to this sample project and using Smart Configurator's component configuration feature as below. Each configurable item description is displayed in Macro definition view by clicking the configuration item.

# Enable low power mode	☑ Enable Demo_LowPower_Enter() Demo_LowPower_Exit()	
# Enter low power function		
# Exit low power function		
# Enable tickless operation in low power mode	🖾 Disable	
# Enable threadx timer setup	🗷 Enable	
# Low power timer setup function	Demo_LowPower_Timer_Setup	
# Enable threadx user timer adjust	🗷 Enable	
# Low power user timer adjust function	Demo_LowPower_User_Timer_Adju	
# Enable threadx wait	Disable	
		~
acro definition: TA_LOW_POWER		~
LOW_POWER macro can be used together with TX_ENABLE_WAIT se 1: TX_LOW_POWER == 1 and TX_ENABLE_WAIT == 0: execute of se 2: TX_LOW_POWER == 0 and TX_ENABLE_WAIT == 1: execute of se 3: TX_LOW_POWER == 0 and TX_ENABLE_WAIT == 0: no supplies se 4: TX_LOW_POWER == 1 and TX_ENABLE_WAIT == 1: execute of the sector of the table sector of table sector o	T macro user-defined low power consumption function (call tx_low_power_enter/є only WAIT instruction in ThreadX (tx_low_power_enter/exit are not called) ort for low power consumption is low power enter WAIT instruction, and tx low power exit	exit)

Figure 2.44 Configuration of ThreadX Low Power

- If the Low Power Consumption Device Driver Module (r_lpc_rx) is used, the module executes "WAIT" instruction inside the r_lpc_rx module. Therefore, please note that "Enable threadx wait" must be disabled.
- If you define your own function for "Enter low power function", "Exit low power function", "Low power timer setup function" and "Low power user timer adjust function", please modify the prototype definition for each function in libs/threadx/tx_user.h manually as well.





Figure 2.45 libs/threadx/tx_user.h

- The "tx_low_power_next_expiration" parameter is passed to the "TX_LOW_POWER_TIMER_SETUP" function. Since the tx_low_power_next_expiration is the next timer deadline (i.e., the number of ticks before the next wakeup), a low power mode timer must be set so that the low power mode is resumed before this tick number elapses.
 When the tx_low_power_next_expiration is 0xffffffff, there is no next timer expiration date (there is no thread waiting for a timeout), so the user may resume from the low power mode at any time.
 When the tx_low_power_next_expiration is very small value, the transition to the low power consumption mode may be omitted by judging from the transition process time and the resume process time because it depends on the processing time of the user-defined function.
- For the latest information of Low Power APIs, please refer to <u>https://github.com/azure-rtos/threadx/blob/master/utility/low_power/low_power.md</u>.



2.13 Azure Device Update (ADU) sample project

This sample project illustrates over-the-air (OTA) firmware update via Microsoft Azure. Azure ADU is a cloud service provided by Microsoft that enables deployment of OTA updating of IoT devices.

When implementing ADU, secure boot loader sample project must be used together with this project. The secure bootloader function is to verify that firmware to be run is reliable, make sure it has not been tempered, and update it.

Supported Kits:

- Renesas Starter Kit+ for RX65N-2MB
- CK-RX65N cloud kit (Ether/ Cellular)
- Renesas RX65N Cloud Kit
- Renesas Starter Kit+ for RX671
- RX72N Envision Kit

Development Environment:

• e2 studio : 2023-10 or later

To run this sample, please see the document <u>"RX Family How to implement OTA by using Microsoft Azure Services"</u>.

To learn more about Azure ADU, view https://learn.microsoft.com/azure/iot-hub-device-update/.



Revision History

		Description	
Rev.	Date	Page	Summary
1.00	Jul. 20, 2022		First edition issued
1.01	Oct. 20, 2022	1, 22	Changed project name from "PnP Temperature Control sample project" to "IoT Embedded SDK with IoT Plug and Play sample project"
		2	Added Azure IoT Explorer
1.02	Jan. 20, 2023	6	Improved creation procedure for IAR EW project
		24, 25	Added USBX Host Mass Storage Class sample project
		31	Added Azure Device Update sample project and secure bootloader sample project
1.03	July. 28, 2023	-	Add minimal sample explanation Remove IoT Embedded SDK with IoT Plug and Play sample project
2.00	Jan. 12, 2024	-	Supported RX23E-B Added USBX HCDC sample Updated Azure Device Update (ADU) sample project explanation.
2.10	July. 1, 2024	-	Supported Azure RTOS v.6.4.0 Unsupported IoT Embedded SDK PnP sample project from Azure RTOS v.6.4.0



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 is supplied until the power is supplied until the power is supplied until the 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 systemevaluation 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 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 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.
- 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

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/.