RX Family

Examples of Controlling to Low Power Consumption (Intermittent Operation) using LTE Cat-M1 Module (RYZ014A) and FreeRTOS

Introduction

This application note describes examples of controlling low power consumption operation using the RX65N Cloud Kit board from Renesas and the LTE Cat-M1 cellular module (RYZ014A) and communicating with Amazon Web Services™(AWS) using Amazon FreeRTOS™.

Features of this application note

- You can learn the most suitable low power consumption mode and settings for your use case. (See 2.4)
- To learn how to operate with low power consumption when using FreeRTOS with the RX family. (See 2.1)
- You can learn how to operate RYZ014A with low power consumption. (See 2.2)
- You can learn the low power consumption operation method of the RX family by using RX65N as an example. (See 2.3)

■ **Renesas RYZ014A Module**

The Renesas RYZ014A module is an all-in-one, single-mode LTE category M1 module with worldwide deployment and roaming capability. The RYZ014A module includes a carrier-proven LTE Cat-M1 protocol stack and comprehensive software.

Notice: RYZ014A (cellular module for CAT-M1) is not included in the RX65N Cloud Kit but is sold separately.

■ **Amazon FreeRTOS**

Amazon FreeRTOS is a real-time operating system that enhances the FreeRTOS kernel with functionality for connections, security, and over-the-air (OTA) updates. It includes demo applications for demonstrating the functionality of Amazon FreeRTOS. It is distributed free of charge under the MIT Open Source License.

Target Device

RX65N group (RX65N Cloud Kit)

- Information on the boards, related programs, and development environment required to develop the RX cloud solution is summarized in the links below.
  https://www.renesas.com/rx-cloud
- Information about RYZ014A is summarized in the link below.

  https://www.renesas.com/jp/ja/products/interface-connectivity/wireless-communications/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module

Contents

1. Overview

1.1 Overview of this document

The system that combines the RX65N and RYZ014A module can connect IoT devices directly to the cloud using LTE Cat-M1 of the low power wide area network (LPWAN). The main feature is that there is no need a gateway or router to connect to the cloud. For this reason, cellular connections can be placed anywhere as long as there is a mobile LTE network, and the infrastructure construction can be done flexibly. It also has an advantage compared to Wi-Fi in terms of power consumption.

While such systems are not tied to the location, it is often used for battery-powered equipment, so it is required that it can be operated for a long time with a battery. The device must be switched to a low power consumption state to effectively utilize the limited battery power.

RX65N, RYZ014A, and Amazon FreeRTOS each have a function to reduce power consumption, and this document shows an example of a control that saves power by combining these functions to perform intermittent operations.

The intermittent operation is an operation in which the microcomputer is used by switching between the normal operation mode and the low power consumption mode.

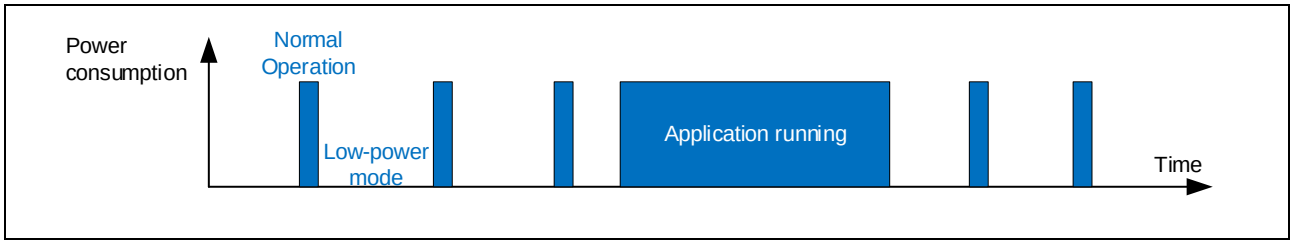

Figure 1-1 Overview of the intermittent Operation

1.2 Examples of Low-Power Control

This document describes how to use the low-power mode of RX65N and RYZ014A with Amazon FreeRTOS. Three examples are shown depending on the system use case. The outline of each system and the operation image of RX65N are as follows.

Table 1-1 Case 1. Overview of the System

System operation		A system that keeps programs running at all times, such as for communication with AWS™ and sensors. If you want to reduce power consumption as much as possible.
Assumed standby time		Several hundred msec
Power saving control example	FreeRTOS	Tickless idle mode
	RYZ014A	No power saving setting
	RX65N	Sleep Mode
Advantage		Even a system that cannot operate intermittently can be operated with low power consumption.
Disadvantage		It is not suitable when you want to count accurately because the tick timer value may shift.

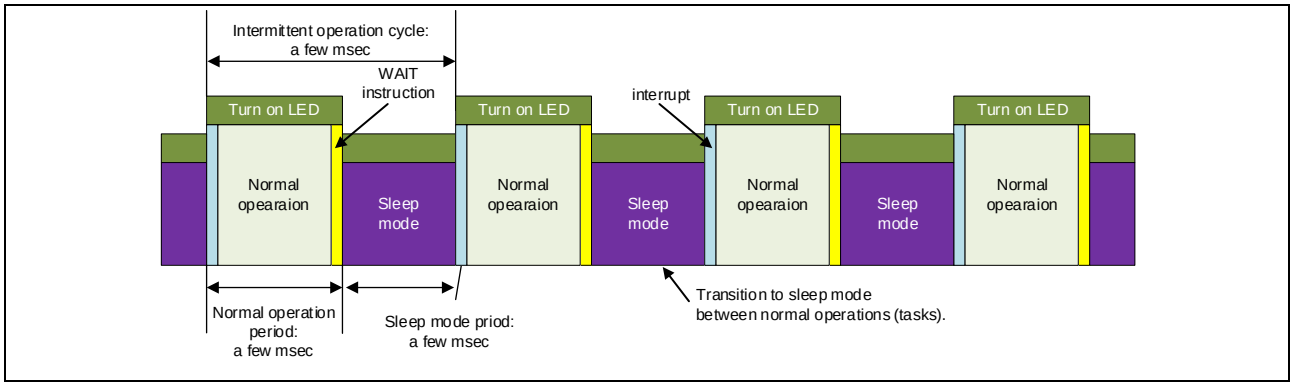

Figure 1-2 Case 1. Operation Image of RX65N

Table 1-2 Case 2. Overview of the System

System operation		A system in which there is a period during which the program does not operate due to intervals in communication with AWS and sensors.
Assumed standby time		A few seconds to a few minutes
Power saving control example	FreeRTOS	Tickless idle mode
	RYZ014A	Enable “eDRX”
	RX65N	Software Standby Mode
Advantage		By performing intermittent operation, it is possible to reduce power consumption while keeping the responsiveness of the device.
Disadvantage		Since transition time is required, it cannot be used in a system that requires processing speed.

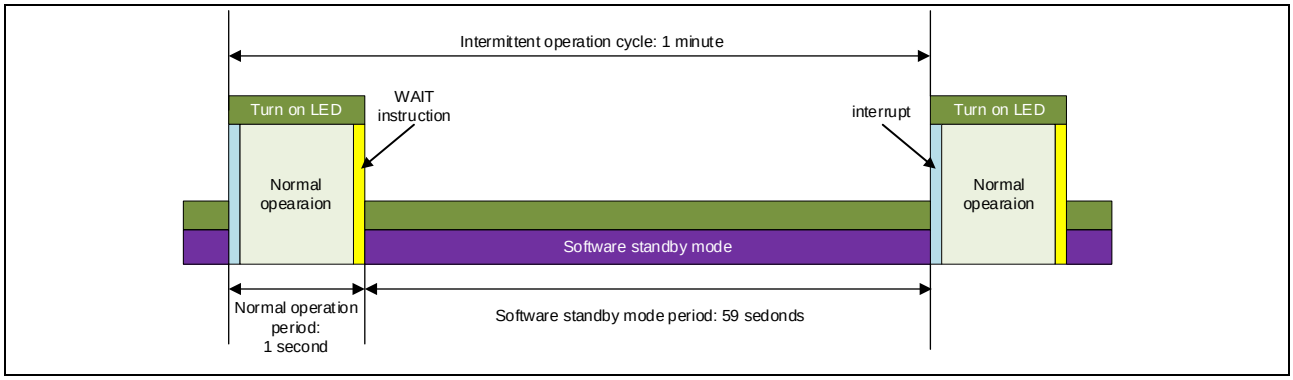

Figure 1-3 Case 2. Operation Image of RX65N

Table 1-3 Case 3. Overview of the System

System operation		A system that takes intervals to communicate with AWS and sensors, and the program does not operate for a long period.
Assumed standby time		Dozens of minutes to over an hour
Power saving control example	FreeRTOS	Tickless idle mode
	RYZ014A	Enable “PSM"
	RX65N	Software Standby Mode and Deep Software Standby Mode
Advantage		Achieves micro-order current consumption by setting it to the lowest power consumption state.
Disadvantage		Since returning from the standby mode is in the same state as an internal reset, the program is initialized. Since rush current flows at restarting, frequent restarting will result in higher power consumption.

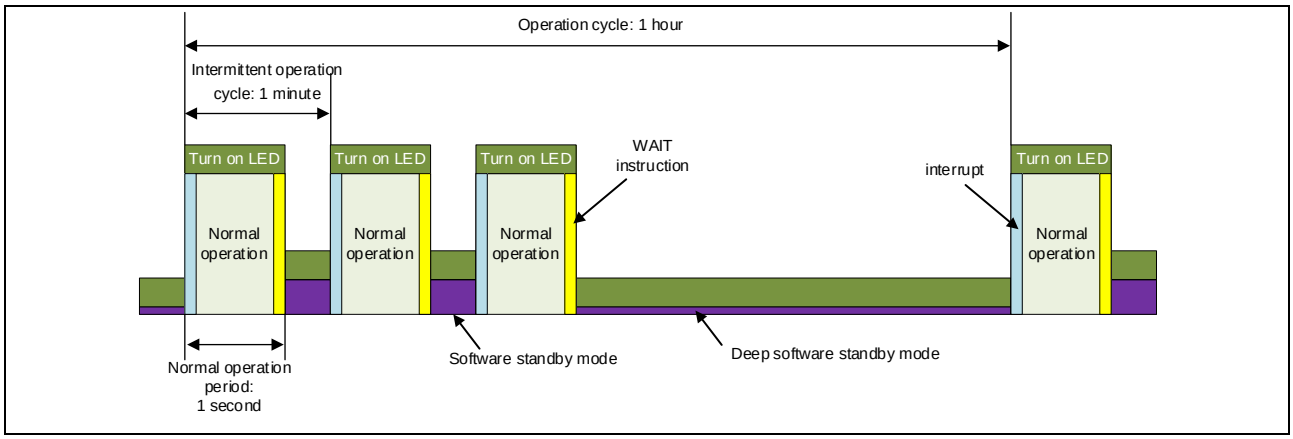

Figure 1-4 Case 3. Operation Image of RX65N

1.3 Diagram of the System

Below is a system diagram that communicates with AWS using the RX65N Cloud Kit and the RYZ014A module. As shown in the figure, this document describes the power saving settings of a system that has Amazon FreeRTOS installed on the RX65N and uses the RYZ014A module to communicate with AWS.

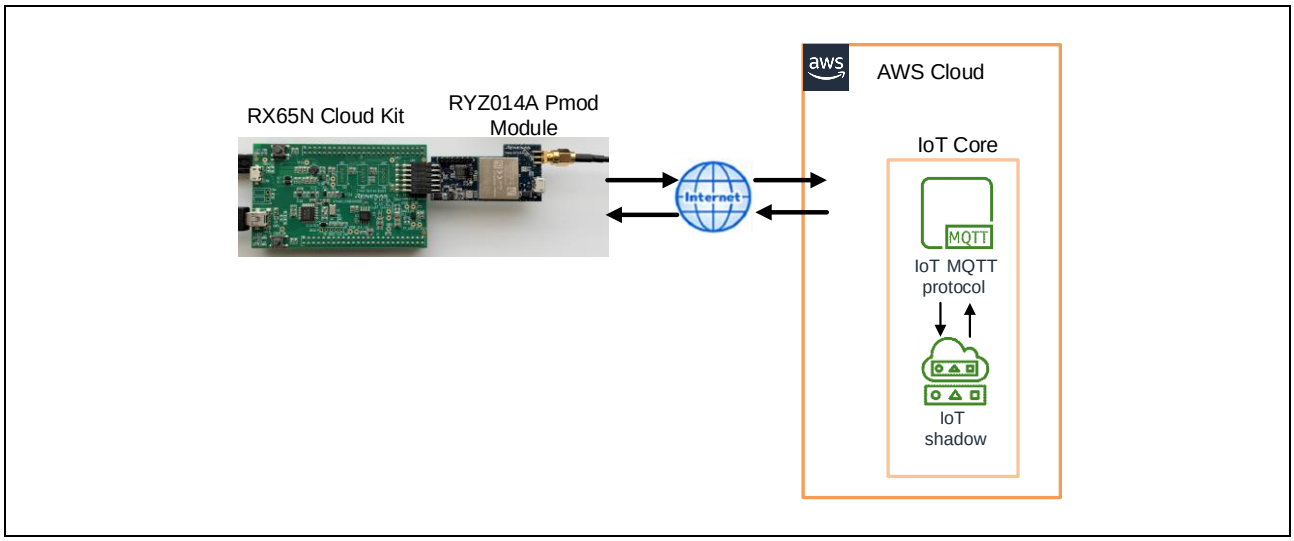

Figure 1-5 Diagram of the System Image

2. Low Power Operation

2.1 Low Power Operation of FreeRTOS

FreeRTOS has a tickless idle mode for low power applications. By using FreeRTOS's tickless idle mode, you can put the RX65N into and out of low power consumption mode during the idle period of the task to save power.

Tickless idle mode is provided by FreeRTOS. The user defines the RX65N low power consumption mode selection and the processing before and after the transition to the low power consumption mode according to the operating status of the application, such as in Figure 2-1.

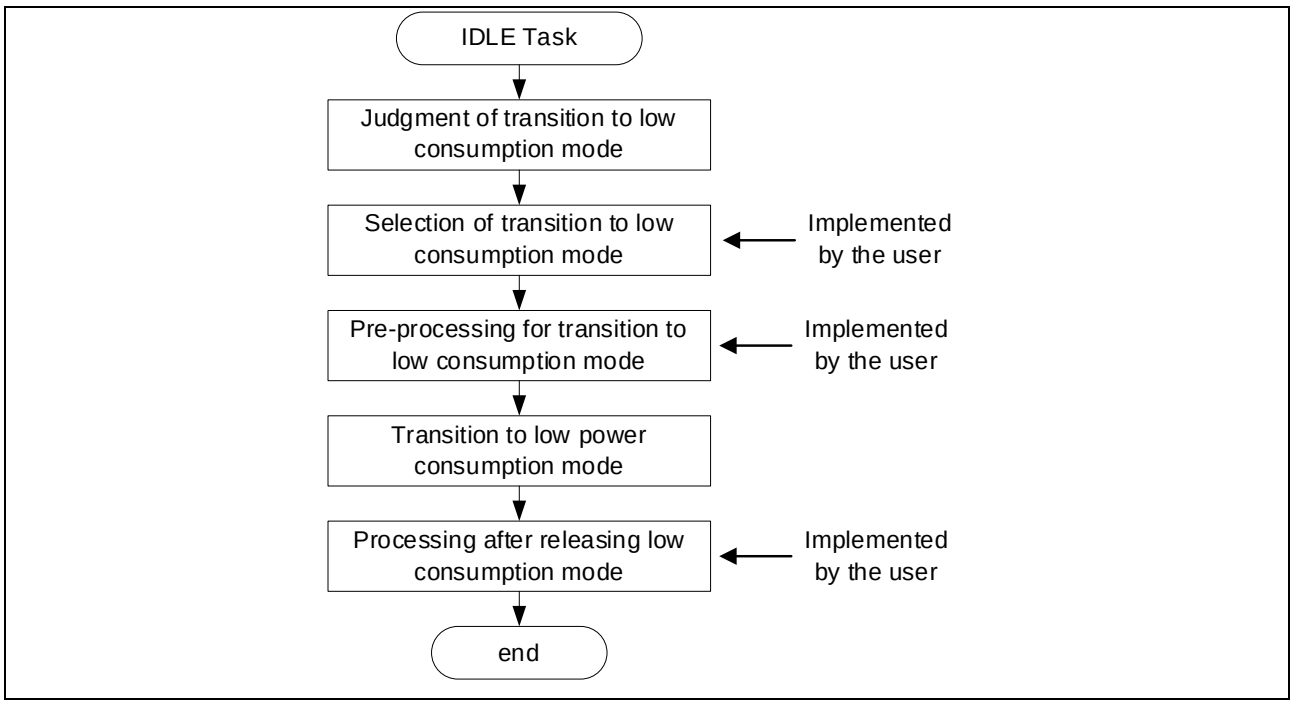

Figure 2-1 Implementation Overview of Tickless Idle Mode

2.2 Low Power Operation of RYZ014A

RYZ014A is a cellular IoT module compatible with LTE Cat-M1 specifications and supports eDRX (extended Discontinuous Reception) and PSM (Power Saving Mode), which are the power saving technologies of Cat-M1.

eDRX is a function that realizes even lower power consumption by expanding the conventional power saving technology, intermittent receiving technology (DRX), and making it possible to set a longer intermittent receiving interval. PSM is a function that achieves even lower power consumption by putting it into hibernation (deep sleep) for a longer period than eDRX. The user can select eDRX or PSM and set the intermittent operation interval for each, and realize low power consumption operation according to the application.

2.3 Low Power Consumption Modes of RX65N

During the FreeRTOS idle period, the RX65N can transition to each mode with low power consumption. Select the mode to shift according to the operating status of the application and the power requirement of the system, and execute each mode transition procedure to shift to the low power consumption mode.

The RX65N reduces power consumption by reducing the functions that operate in each low power consumption mode. Table 2-1 outlines the operating states of each low power consumption mode.

Please refer to the following manual for details on the operating status of each mode.

RX65N Group, RX651 Group User's Manual: Hardware (R01UH0590)
11. Low Power Consumption

Table 2-1 The Operating States in Each Low Power Consumption Mode

Operating States	Normal operation **(RUN)**	Sleep Mode	All-Module Clock Stop Mode	Software Standby Mode	Deep Software Standby Mode
CPU	Operating	Stopped	Stopped	Stopped	Stopped
Clocks	Operating	Operating	Operating	Stopped	Stopped
Flash	Operating	Operating	Stopped (Retained) (*1)	Stopped (Retained) (*1)	Stopped (Undefined) (*2)
RAM	Operating	Operating	Stopped (Retained) (*1)	Stopped (Retained) (*1)	Stopped (Undefined) (*2)
Peripheral module	Operating	Operating	Stopped (Retained) (*1)	Stopped (Retained) (*1)	Stopped (Undefined) (*2)
Supply current (*3)	26mA (ICLK:120MHz)	20mA	9mA	1.6mA	4.9uA (*4)
	High				**Low**

Note 1. "Stopped (Retained)" means that internal register values are retained and internal operations are suspended.

Note 2. "Stopped (Undefined)" means that internal register values are undefined and power is not supplied to the internal circuit.

Note 3. Describe the value of the products with at least 1.5 Mbytes of code flash memory, D version and Typ.

Note 4. The case of power is not supplied to the standby RAM and USB resume detecting unit (USB0 only) and low power consumption function of the power-on reset circuit is enabled.

2.4 Use Case Examples for Each Standby Modes

The usage of the low power consumption function of FreeRTOS, RYZ014A and RX65N is shown in the following three cases by taking the use case of system operation as an example. The program control example and setting method are described according to the low power consumption function used in each case.

Table 2-2 Case 1. System Example

System operation		A system that keeps programs running at all times, such as for communication with AWS and sensors. If you want to reduce power consumption as much as possible.
Using functions	FreeRTOS	Tickless idle mode
	RYZ014A	No power saving setting
	RX65N	Sleep Mode
Control example	Transition to low power consumption mode	Put the RX65N into sleep mode between task operations (during idle task operations) to reduce power consumption.
	Release from low power consumption mode	Release from sleep mode is the tick timer interrupt.
	Operation after release	By using the tickless idle mode, the tick timer value is adjusted and the operation is restored from the continuation of the program.
Setting method	FreeRTOS	Set configUSE_TICKLESS_IDLE using a sample program that supports tickless idle mode.
	RYZ014A	No setting.
	RX65N	Execute the wait () instruction.

Table 2-3 Case 2. System Example

System operation		A system in which there is a period during which the program does not operate due to intervals in communication with AWS and sensors.
Using functions	FreeRTOS	Tickless idle mode
	RYZ014A	Enable “eDRX”
	RX65N	Software Standby Mode
Control example	Setting RYZ014A	Set the operation of eDRX mode and adjust the intermittent receiving interval.
	Transition to low power consumption mode	Suspend the task and put the RX65N into software standby mode at the timing after sensing and AWS communication.
	Release from low power consumption mode	Count the time by the RTC according to the operation interval of the program, and release by the RTC interrupt.
	Operation after release	Resume the suspended task. Adjusts the tick timer value by the tickless idle mode and resumes operation from the continuation of the program.
Setting method	FreeRTOS	Set configUSE_TICKLESS_IDLE using a sample program that supports tickless idle mode.
	RYZ014A	Make settings using the API that enables eDRX.
	RX65N	Set to transition to software standby mode when the sensing and AWS communication tasks are suspended.

Table 2-4 Case 3. System Example

System operation		A system that takes intervals to communicate with AWS and sensors, and the program does not operate for a long period. (A system that operates once an hour, etc.)
Using functions	FreeRTOS	Tickless idle mode
	RYZ014A	Enable “PSM”
	RX65N	Software Standby Mode and Deep Software Standby Mode
Control example	Setting RYZ014A	Set the operation of PSM mode and adjust the time until entering the mode.
	Transition to low power consumption mode - No. 1	Suspend the task and put the RX65N into software standby mode at the timing after sensing and AWS communication.
	Release from low power consumption mode - No. 1	Count the time by the RTC according to the operation interval (short time) of the program, and release by the RTC interrupt.
	Operation after release - No. 1	Resume the suspended task. Adjusts the tick timer value by the tickless idle mode and resumes operation from the continuation of the program.
	Transition to low power consumption mode - No. 2	Repeat No. 1 several times and put the RX65N into deep software standby mode when there is no need for an immediate response such as communication from AWS. Since it will be in the internal reset state after recovery, if it is necessary to store the data, store the data in the standby RAM.
	Release from low power consumption mode - No. 2	Count the time by the RTC according to the operation interval (long time) of the program, and release by the RTC interrupt.
	Operation after release - No. 2	Operation starts from an internal reset. To change to the state before the standby transition, acquire the data from the standby RAM.
Setting method	FreeRTOS	Set configUSE_TICKLESS_IDLE using a sample program that supports tickless idle mode.
	RYZ014A	Make settings using the API that enables PSM.
	RX65N	No. 1: Set to transition to software standby mode when the sensing and AWS communication tasks are in the suspended state. No. 2: Set to transition to deep software standby mode when there is no operation request during the software standby mode period.

3. Configuration of the Sample Programs

3.1 Hardware Configuration

The hardware configuration of this system is shown in the table below.

Table 3-1 Hardware Configurations

Item	Contents	Provider	Description
Board used (RX65N Cloud Kit included items)	Target Board for RX65N	Renesas Electronics Corporation	Evaluation board with RX65N MCU Note
	RX Cloud Option Board		Cloud communication evaluation board that can be connected to AWS Note
Cellular module	RYZ014A Pmod™ Module		Communication board with RYZ014A module
	RYZ014A-EVK		Evaluation Kit with RYZ014A module (Used for current measurement)
SIM	LTE Cat-M1	-	LTE Cat-M1 compatible SIM card Card size of SIM : microSIM
PC	Windows 10	-	Recommendation OS
	Google Chrome	-	Browser to use
Note: Target Board for RX65N and RX65N Cloud Kit Option Board are included in the RX65N Cloud Kit.			

RX65N Cloud Kit Web:

https://www.renesas.com/products/microcontrollers-microprocessors/rx-32-bit-performance-efficiency-mcus/rx65n-cloud-kit-renesas-rx65n-cloud-kit

RYZ014A Web:

https://www.renesas.com/jp/products/interface-connectivity/wireless-communications/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module

RYZ014A Pmod Module:

https://www.renesas.com/products/interface-connectivity/wireless-communications/cellular-iot-modules/rtkyz014a0b00000be-pmod-expansion-board-ryz014a

RYZ014A-EVK:

https://www.renesas.com/products/interface-connectivity/wireless-communications/cellular-iot-modules/ryz014a-evk-lte-cat-m1-cellular-iot-module-evaluation-kit

3.2 Software Configuration

The software configuration of this system is shown in the table below.

Table 3-2 Software Configurations

Item	Contents	Version
Integrated development environment	e² studio	2022-04
Compiler	CC-RX	V3.04
	GCC for Renesas RX	8.3.0.202202
Communication software	Tera Term	Version4.105
Emulator	E2 emulator Lite (on-board)	-
RTOS	Amazon FreeRTOS	Version 202107.00

The software download sites used in this system are shown in the table below.

Note: Please note that the link destination is subject to change.

Table 3-3 Tool Download Site

Contents	Link
e² studio	https://www.renesas.com/kr/software-tool/e-studio
GCC for Renesas RX	https://llvm-gcc-renesas.com/rx-download-toolchains
Tera Term	https://ja.osdn.net/projects/ttssh2

3.3 Tera Term configurations

Set Tera Term as shown in the table below.

Table 3-4 Tera Term Configurations

Item	Setting
Baud Rate	115200
Data length	8bit
Parity	none
Stop Bit	1bit
Flow Control	none

4. Connecting to AWS

The following preparation is necessary to connect RX65N Cloud Kit to AWS.

4.1 AWS Preparation

Refer to the tutorial below and make AWS settings.

- Register the device with AWS IoT
  Link: https://github.com/renesas/amazon-freertos/wiki/Register-device-to-AWS-IoT

Note: Complete the steps up to “Check AWS IoT endpoints.”

The regions used and user permissions are shown below when setting up AWS during the demo.

[About the region used]

This demo is running in the AWS ap-northeast-1 (Asia Pacific (Tokyo)) region.

If you want to run this demo in another region, please make sure that the service used in the demo is available in that region.

[About user permissions]

This demo is run by a user with Administrator Access permissions in AWS Identity and Access Management (IAM). Therefore, there is no particular description regarding the granting of necessary permissions in IAM when using various services.

4.2 Hardware Preparation

1. Install the RYZ014A Pmod Module in the RX65N Cloud Kit.

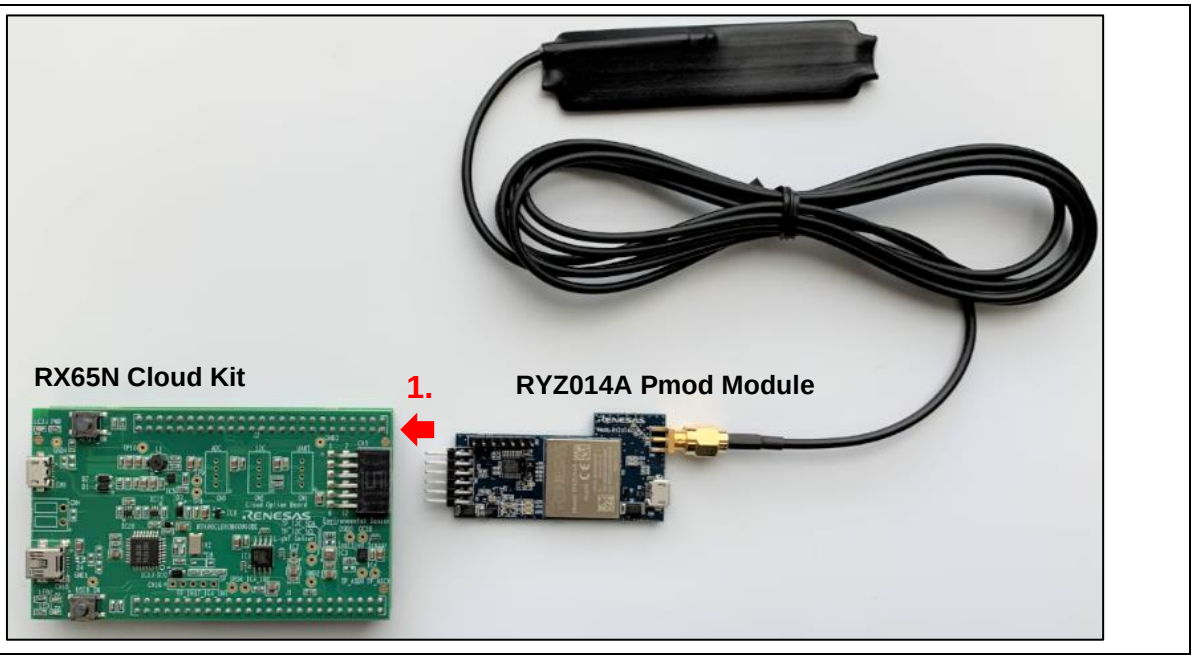

Figure 4-1 RX65N Cloud Kit and RYZ014A Pmod Module

2. Insert the SIM card into RYZ014A Pmod Module.

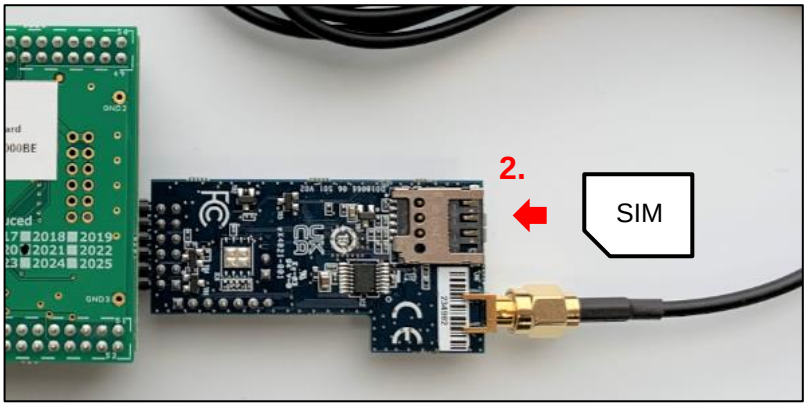

Figure 4-2 Insert the SIM Card into RYZ014A Pmod Module

3. Remove the jumper from the EJ2 pins on the target board (bottom board).
4. Connect the ECN1 connector on the target board (bottom board) to the PC via a USB cable.
5. Connect the CN18 connector on the cloud option board (top board) to the PC via a USB cable.

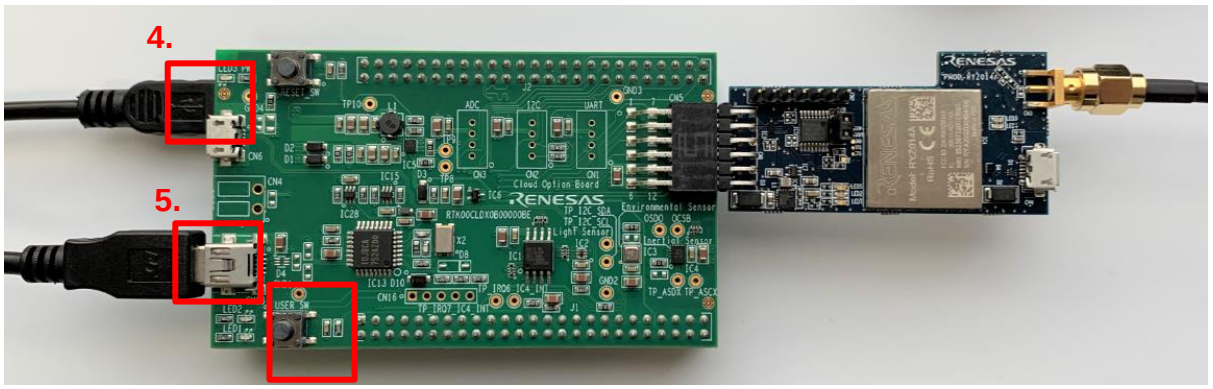

Figure 4-3 RX65N Cloud Kit (top board)

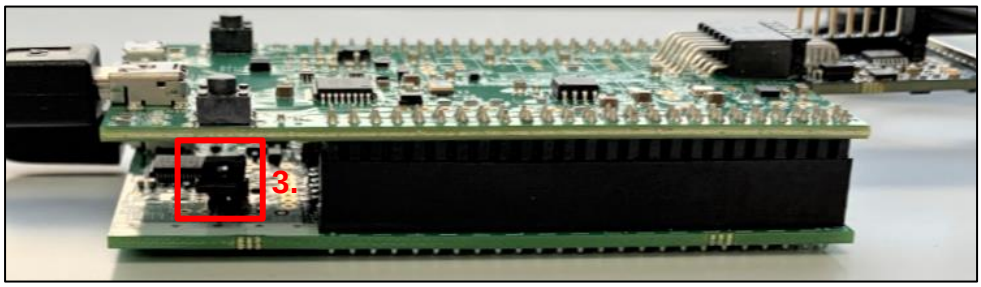

Figure 4-4 RX65N Cloud Kit (bottom board)

4.3 Software Preparation

Follow the steps below to prepare the software for the demo program.

1. Extract the project files from the archive and copy them to a suitable location. (In the description below,the root folder containing the project files is designated as ${base_folder}.)

   Note: After extracting the project files from the archive, copy them to a location with a short file path, such as the root folder on the C: drive. If the file path is too long, a build error may result.

2. Launch e² studio and specify a workspace directory.

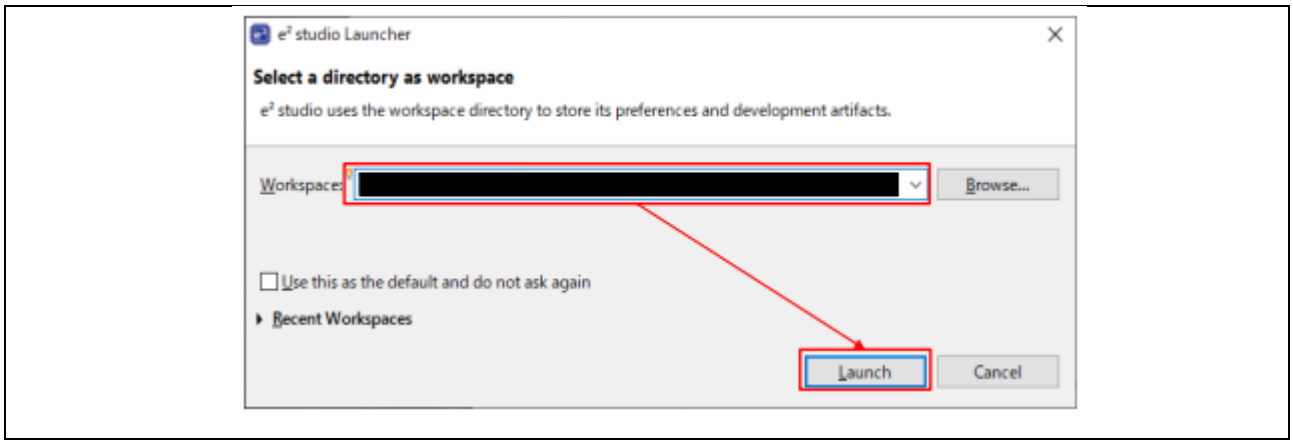

Figure 4-5 Workspace Selection Menu

3. Select **File → Import….**

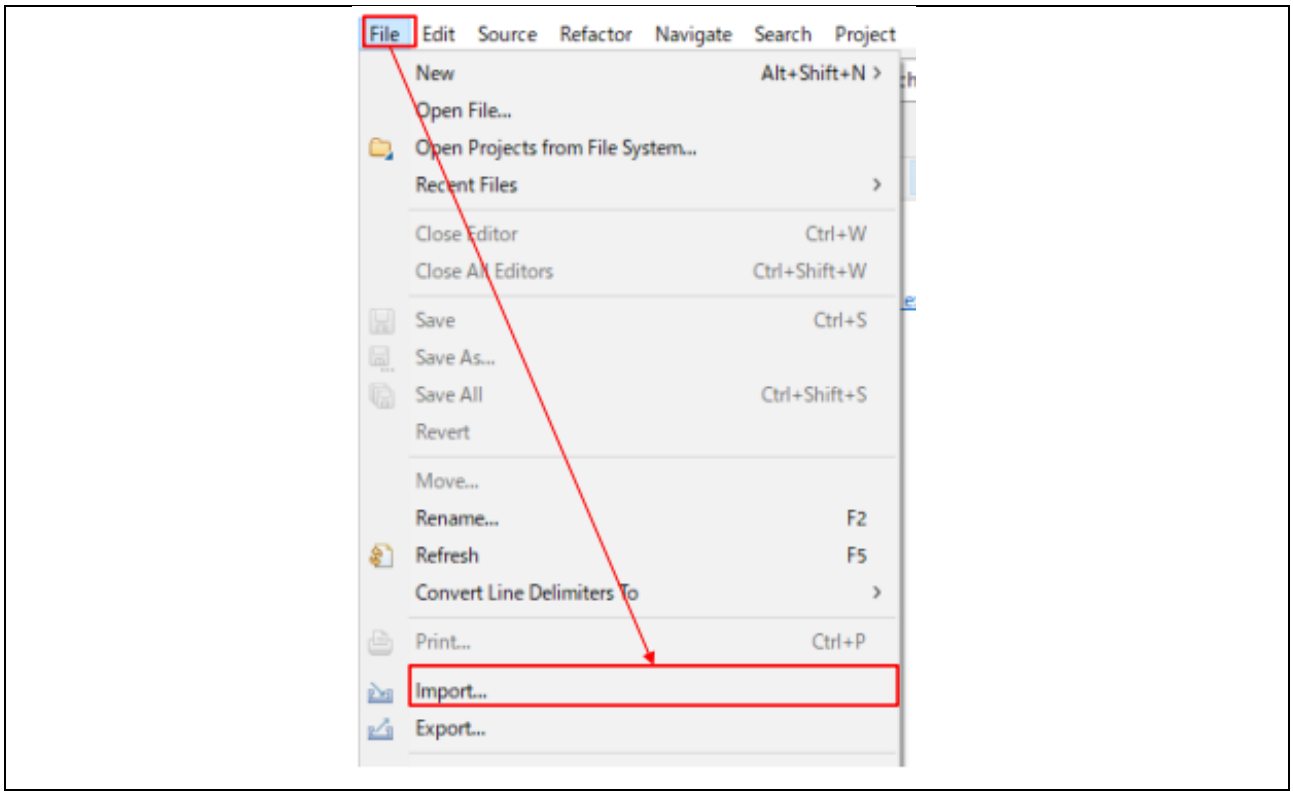

Figure 4-6 File → Import…

4. Click **General → Existing Projects into Workspace → Next >**.

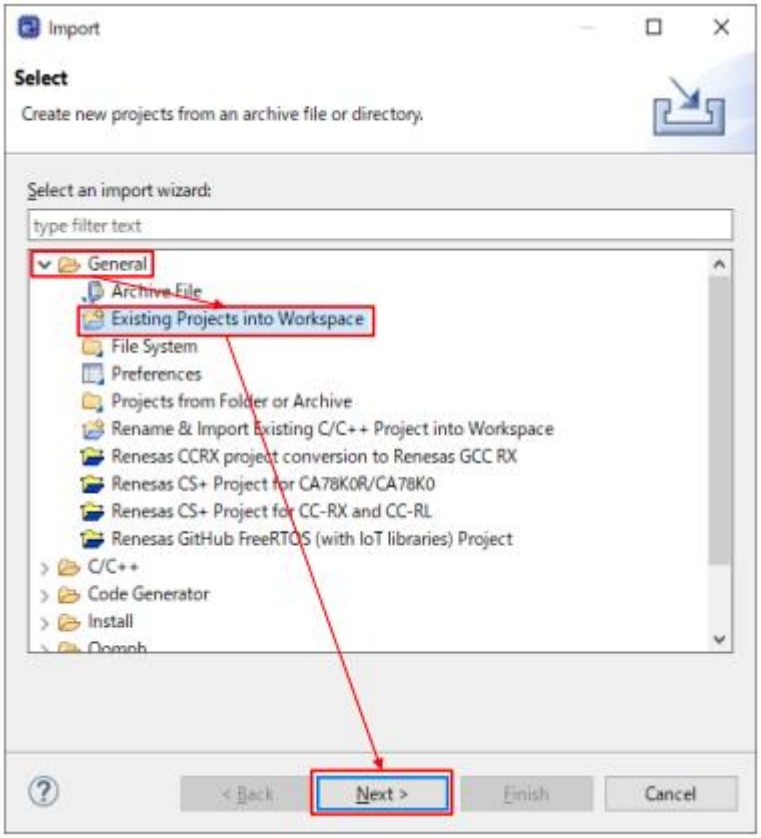

Figure 4-7 General → Existing Projects into Workspace → Next >

5. Click **Browse**…, then specify the root directory as follows.

- ${base_folder}¥projects¥renesas¥rx65n-cloud-kit-uart-sx-ulpgn¥e2studio¥aws_demos
  (When using CC-RX as the compiler)
- ${base_folder}¥projects¥renesas¥rx65n-cloud-kit-uart-sx-ulpgn¥e2studio-gcc¥aws_demos
  (When using GCC for Renesas RX as the compiler)

Finally, click **Finish**.

Note: Make sure **Copy projects into workspace** is unchecked.

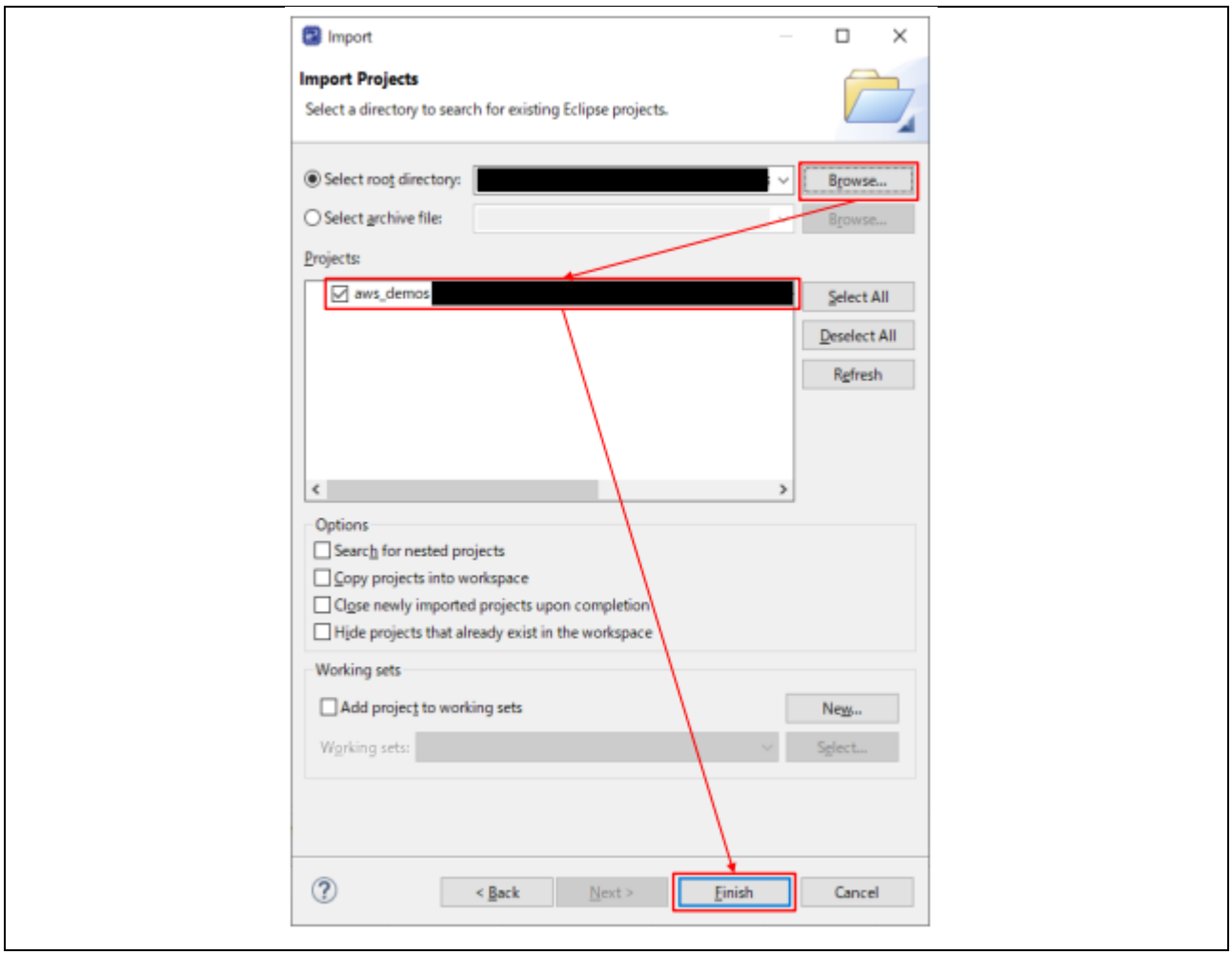

Figure 4-8 Click Browse…, then specify the root directory → Finish

6. Define the following two macros in ${base_folder}¥demos¥include¥aws_clientcredential.h.

- clientcredentialMQTT_BROKER_ENDPOINT
  The name of the endpoint confirmed as described in 4.1, AWS Preparation.

- clientcredentialIOT_THING_NAME
  The name of the thing registered as described in 4.1, AWS Preparation.

(Makes sure to enclose the above macro definitions in quotes (" ") as shown in the figure below.)

Note: clientcredentialWIFI_SSID and clientcredentialWIFI_PASSWORD describe a dummy string for program operation. Be careful not to delete the string.

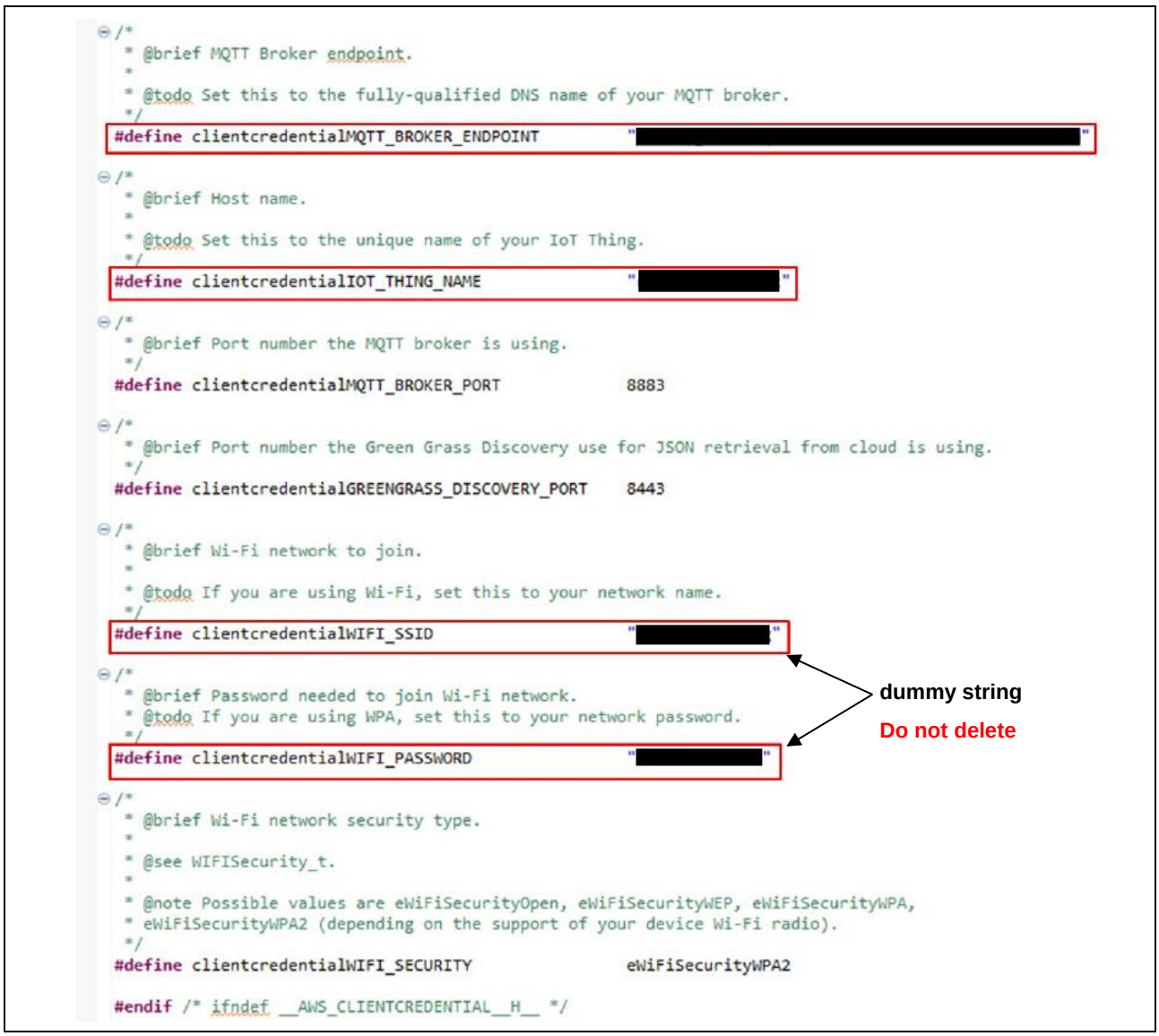

Figure 4-9 aws_clientcredential.h

7. Double-click ${base_folder}¥tools¥certificate_configuration¥CertificateConfigurator.html.

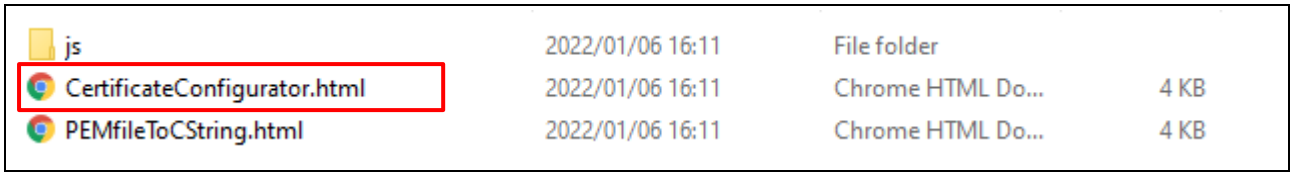

Figure 4-10 Opening CertificateConfigurator.html

8. Specify the thing certificate and private key files downloaded as described in 4.1, AWS Preparation, then click **Generate and save aws_clientcredential_key.h**.
   - xxxxxx-certificate.pem.crt (thing certificate)
   - xxxxxx-private.pem.key (private key)

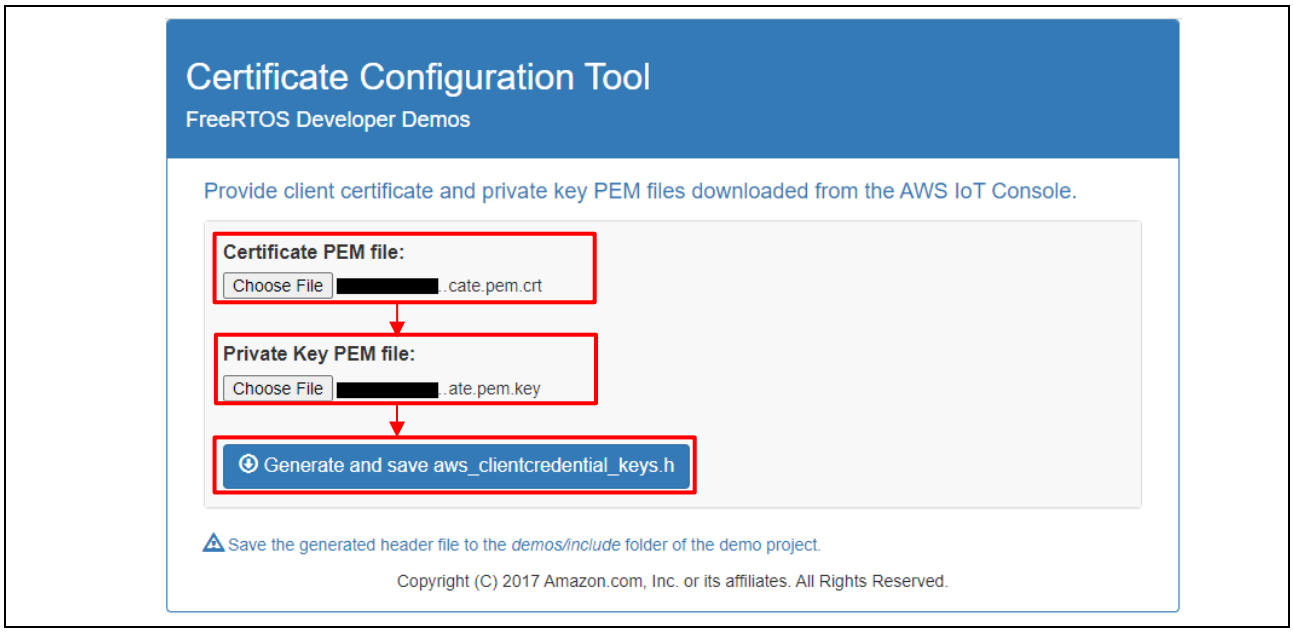

Figure 4-11 Certificate Configuration Tool

9. Overwrite the file ${base_folder}¥demos¥common¥include¥aws_clientcredential_key.h with the newly generated aws_clientcredential_key.h file.

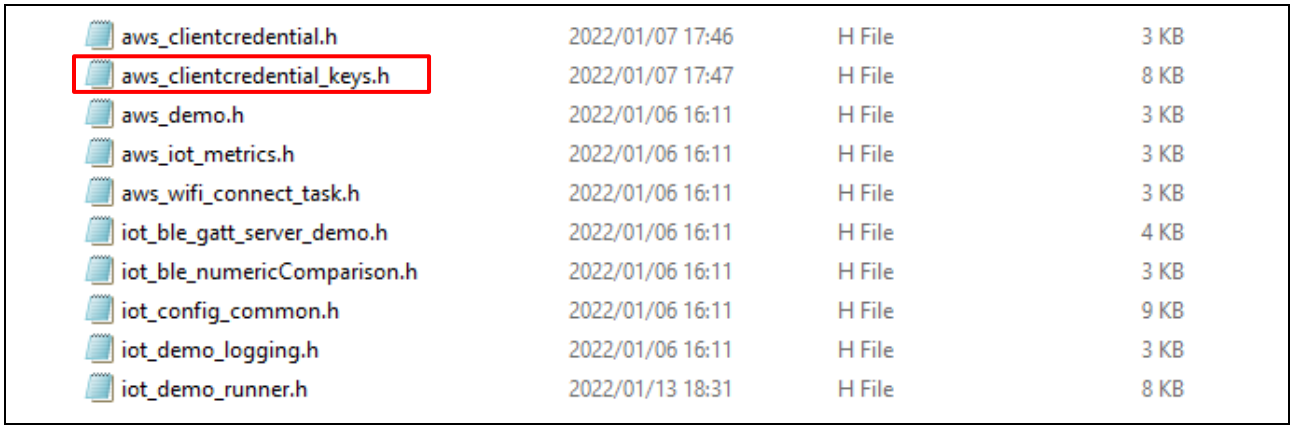

Figure 4-12 Overwriting aws_clientcredential_key.h

10. Define the following three macros in ${base_folder}¥vendors¥renesas¥boards¥rx65n-cloud-kit-cellular(*)¥aws_demos¥src¥smc_gen¥r_config¥r_cellular_config.h.

- CELLULAR_CFG_AP_NAME → The name of SIM access point.
- CELLULAR_CFG_AP_USERID → The user ID of SIM access point.
- CELLULAR_CFG_AP_PASSWORD → The password of SIM access point.

(*) (When using GCC for Renesas RX as the compiler) rx65n-cloud-kit-cellular-gcc

- Check with the SIM contractor for the value to be entered.
- For the above macro, do not use "" as shown in the figure below, but enter the character string as it is. If you do not have a user ID and password set, you do not need to enter them.

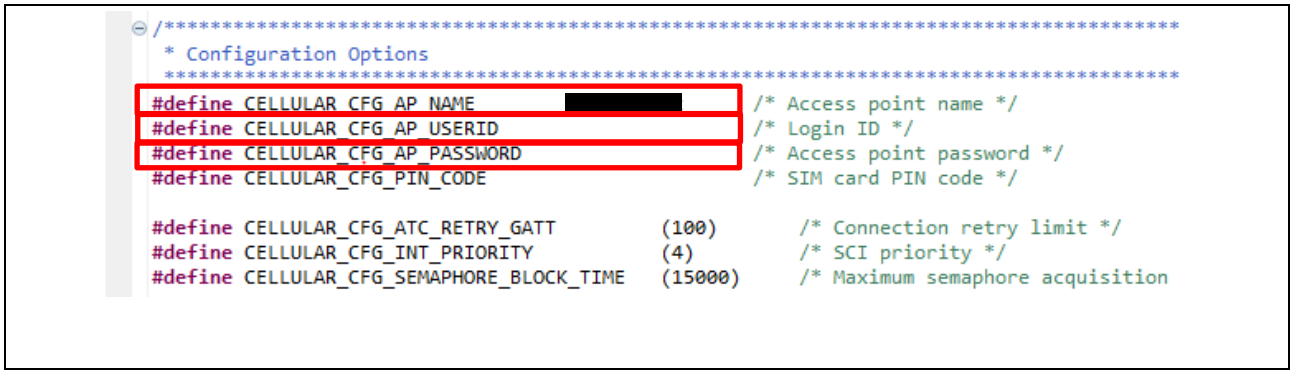

Figure 4-13 r_cellular_config.h

11. Select Project → Build All and confirm that 0 errors are reported.

Note: Make sure to clean the project before building it for the first time. If a demo build error occurs after the initial build, clean the project again and then build it.

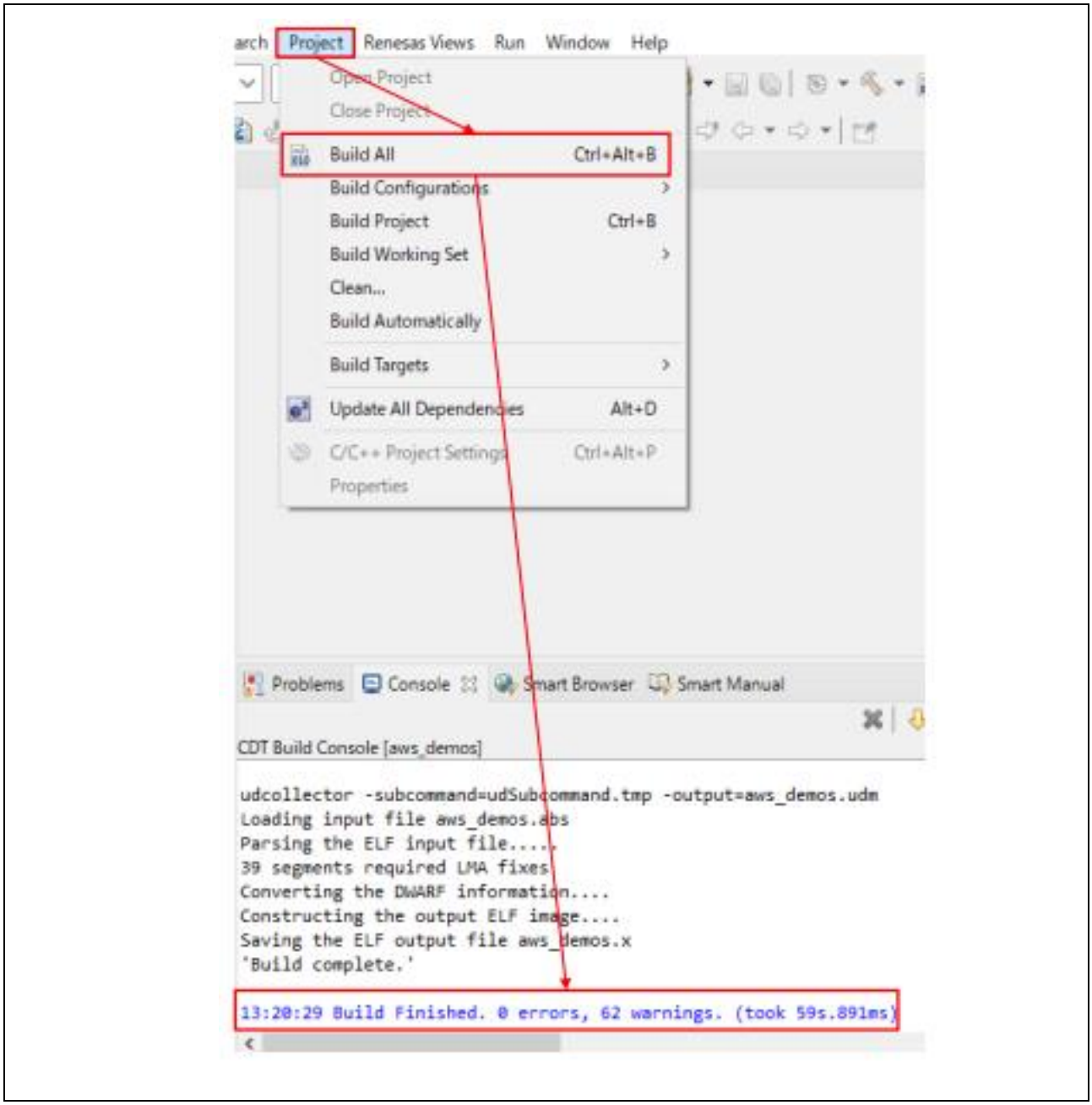

Figure 4-14 Project → Build All → 0 errors

To check the connection to AWS, perform the steps described in section 2, Confirming the Connection to AWS, in the reference document Troubleshooting when Using Amazon Web Services.

5. Sample Project

5.1 Specifications of the Operation

The connection diagram of the device used in the operation of the sample project is shown below.

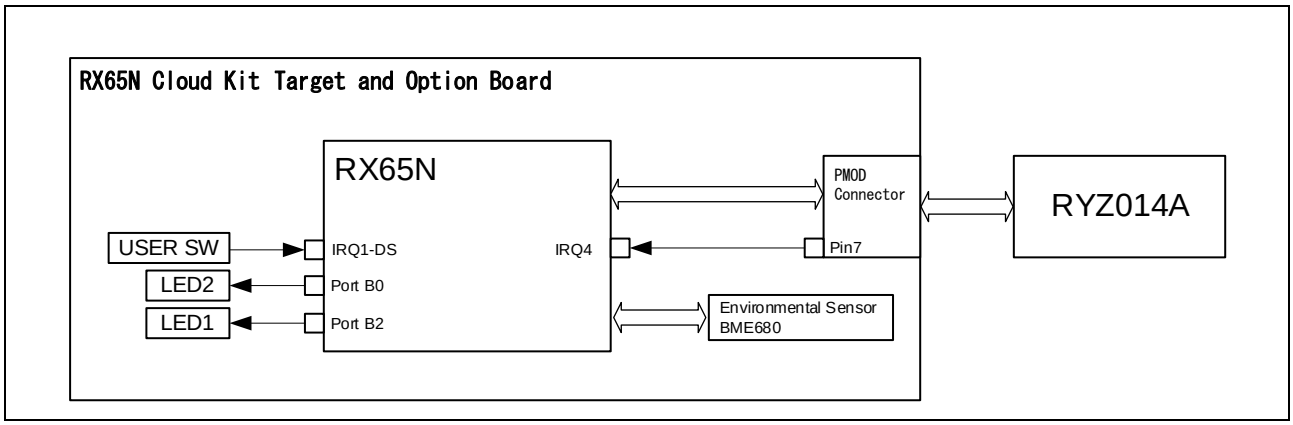

Figure 5-1 Connection Diagram

The sample project runs on Amazon FreeRTOS on the RX65N. It connects to AWS IoT Core via RYZ014A, acquires temperature data from the environment sensor of RX65N Cloud Kit, and uploads it to AWS together with the status of LED1. You can switch LED1 on / off by operating from AWS.

LED2 is lit when the RX65N is in software standby or deep software standby mode. Software standby mode is released by GPIO pin interrupts (IRQ1-DS:USER SW, IRQ4:Operating from AWS). Deep software standby mode is released by GPIO pin interrupts (IRQ1-DS:USER SW).

* The RX65N Cloud Kit does not have an external clock oscillator as shipped from the factory, so RTC does not work. The sample project uses USER SW interrupts instead of RTC interrupts to return from the standby mode.

The positions of USER SW, LED1 and LED2 of RX65N Cloud Kit are as follows.

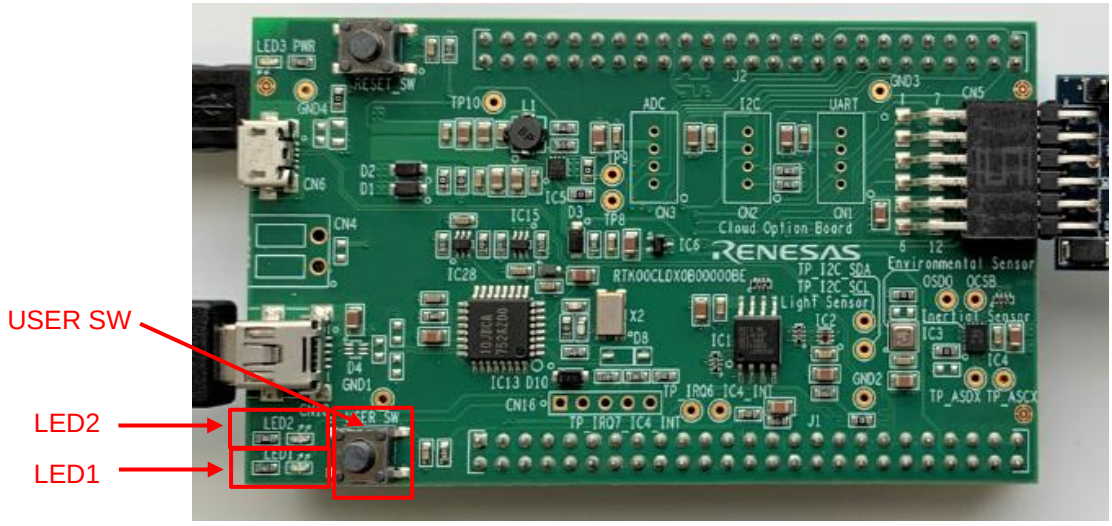

Figure 5-2 RX65N Cloud Kit LED1/2, USERSW

5.2 Flowchart of the Operation

Figure 5-3 shows the flowchart for sending data to AWS.

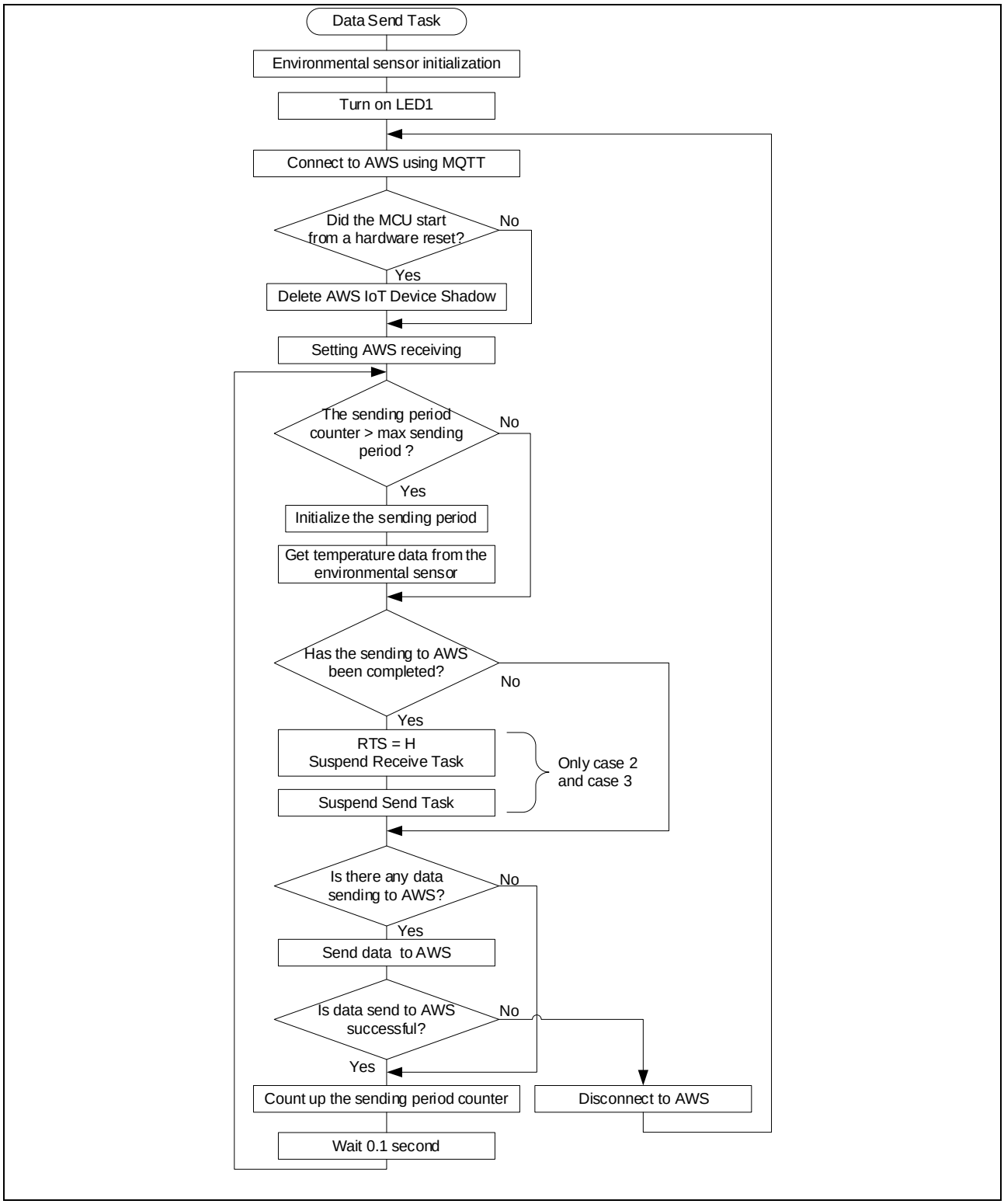

Figure 5-3 Flowchart for Sending Data to AWS

Figure 5-4 shows the operation flowchart when receiving an LED control command from AWS.

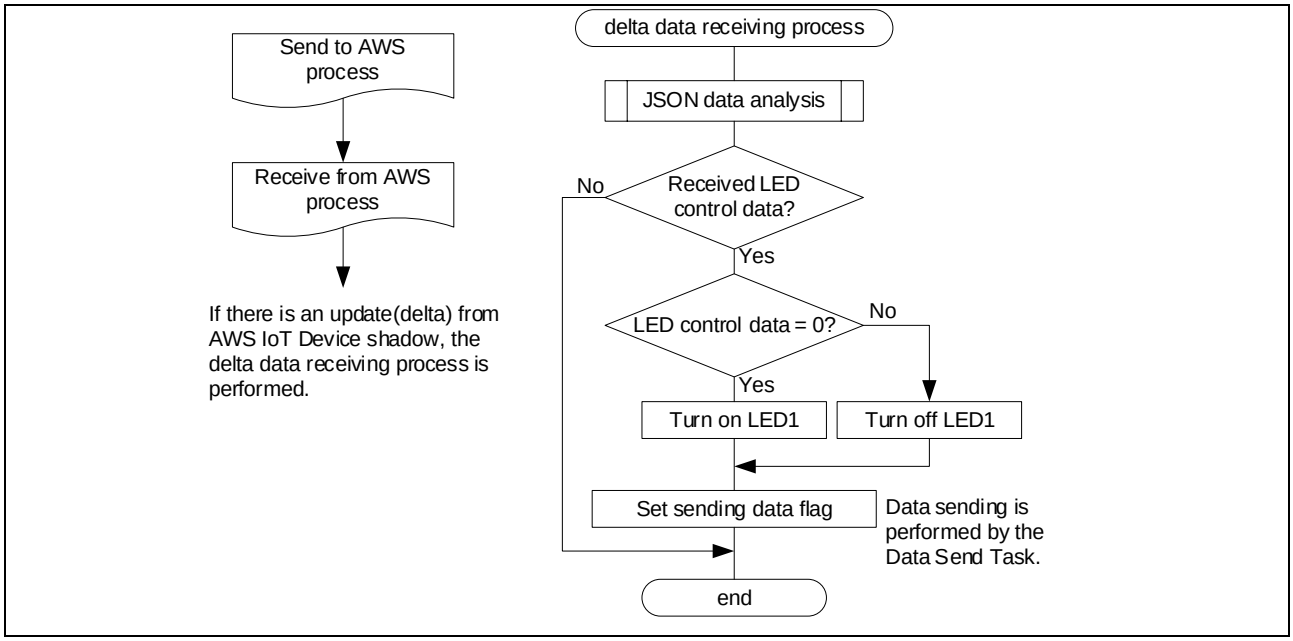

Figure 5-4 Receiving Flowchart of AWS IoT Device Shadow

Figure 5-5 shows the Tickless Idle Mode processing flowchart for the IDLE task. Performs the process of shifting the RX65N to the low power consumption mode in the low power consumption mode setting.

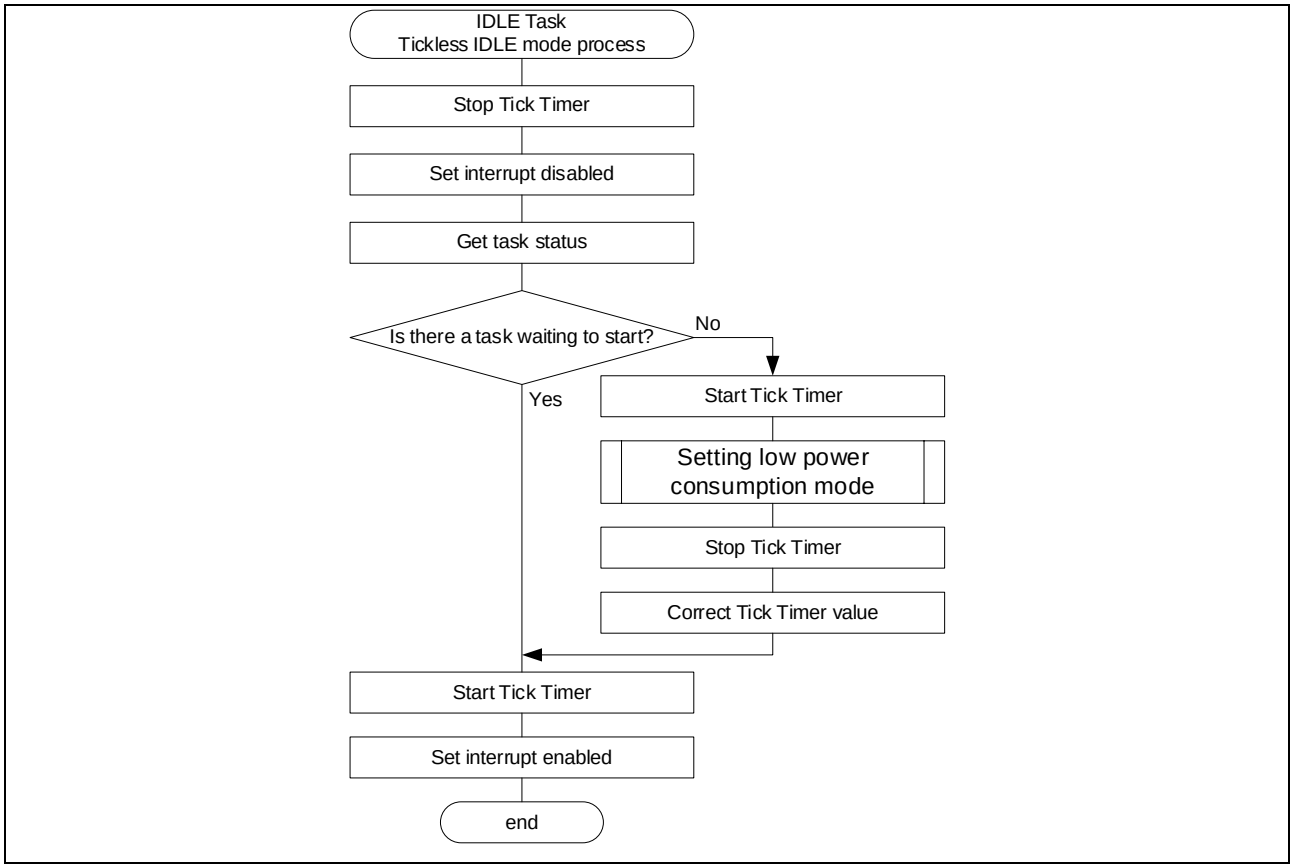

Figure 5-5 Flowchart of Tickless Idle Mode Process for IDLE Task

Figure 5-6 shows the flowchart for the low power consumption mode setting.

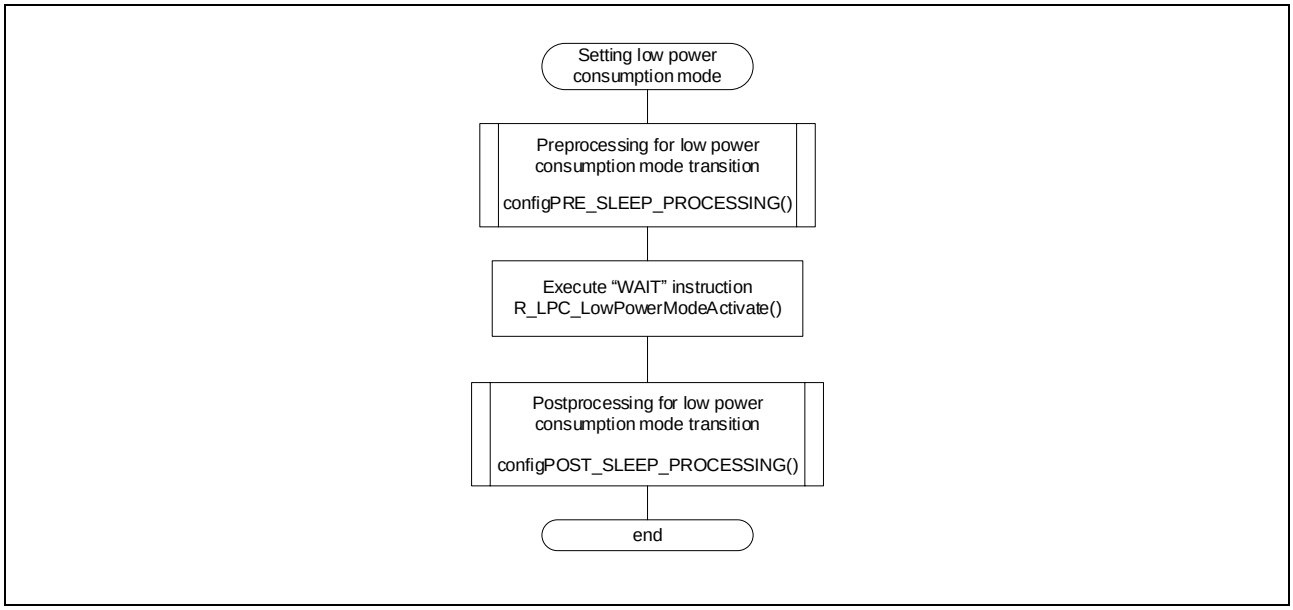

Figure 5-6 Flowchart for Low Power Consumption Mode Setting

The processing and flowchart of the low power consumption mode transition in each mode of the sample project are shown below.

Table 5-1 The Processing of the Low Power Consumption Mode Transition

	Case 1. Sleep Mode	Case 2. Software Standby Mode	Case 3. Software Standby Mode ↓ Deep Software Standby Mode
configPRE_SLEEP_PROCESSING()	No processing	Figure 5-7	Figure 5-9
configPOST_SLEEP_PROCESSING()	No processing	Figure 5-8	No processing

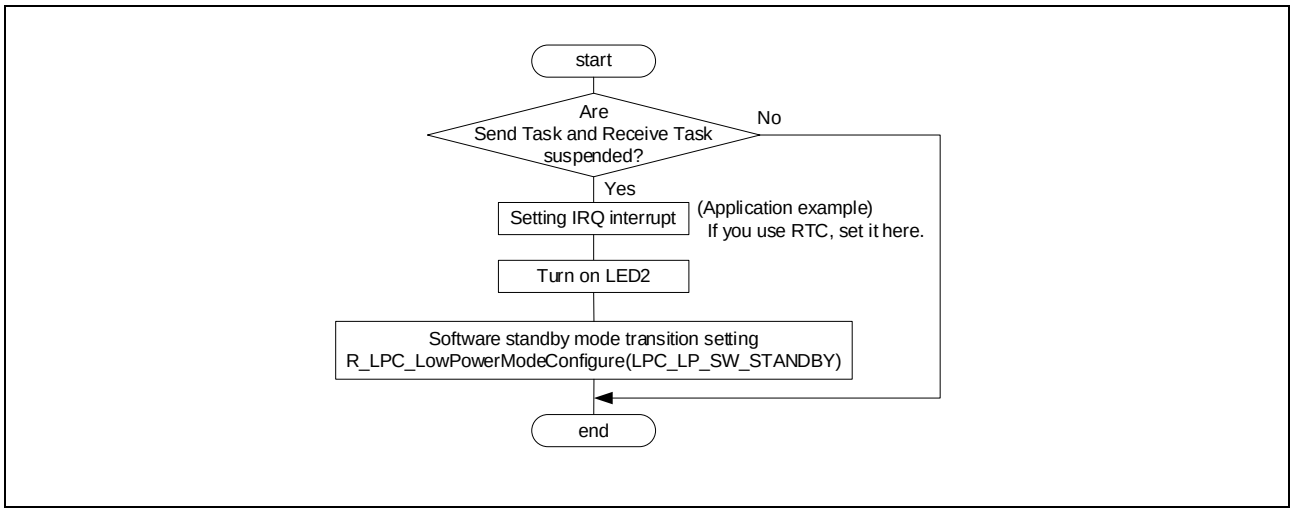

Figure 5-7 Case 2. The Flowchart of configPRE_SLEEP_PROCESSING()

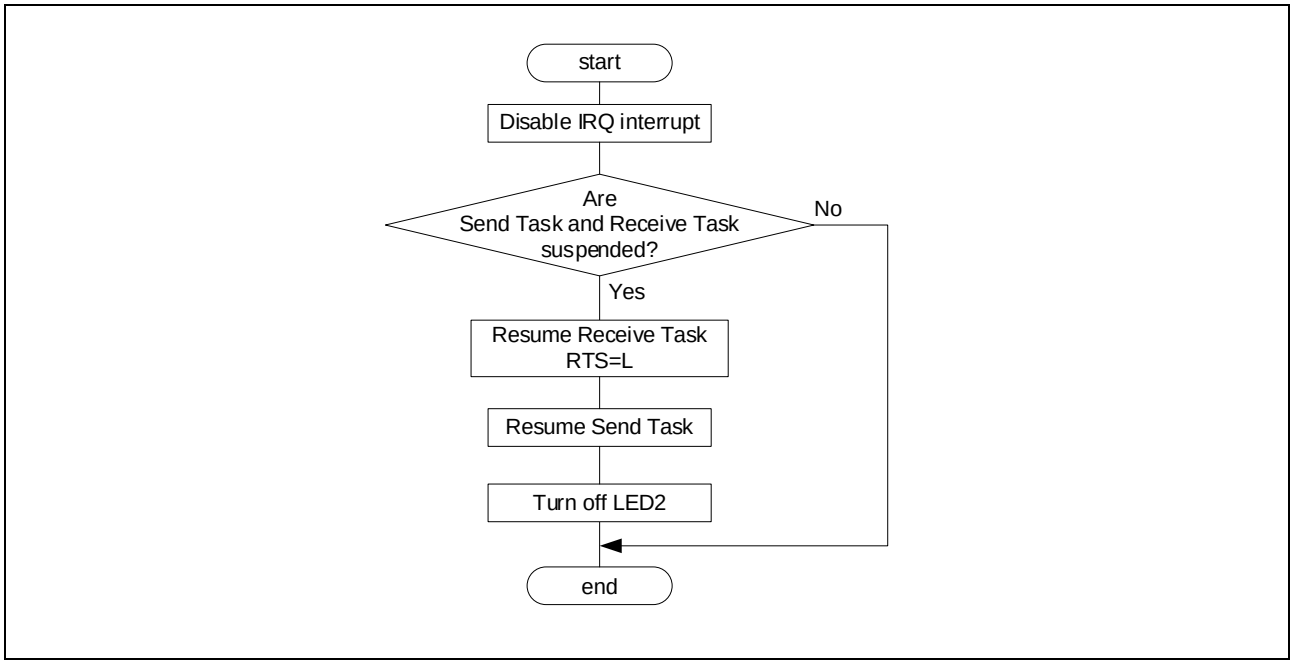

Figure 5-8 Case 2. The Flowchart of configPOST_SLEEP_PROCESSING()

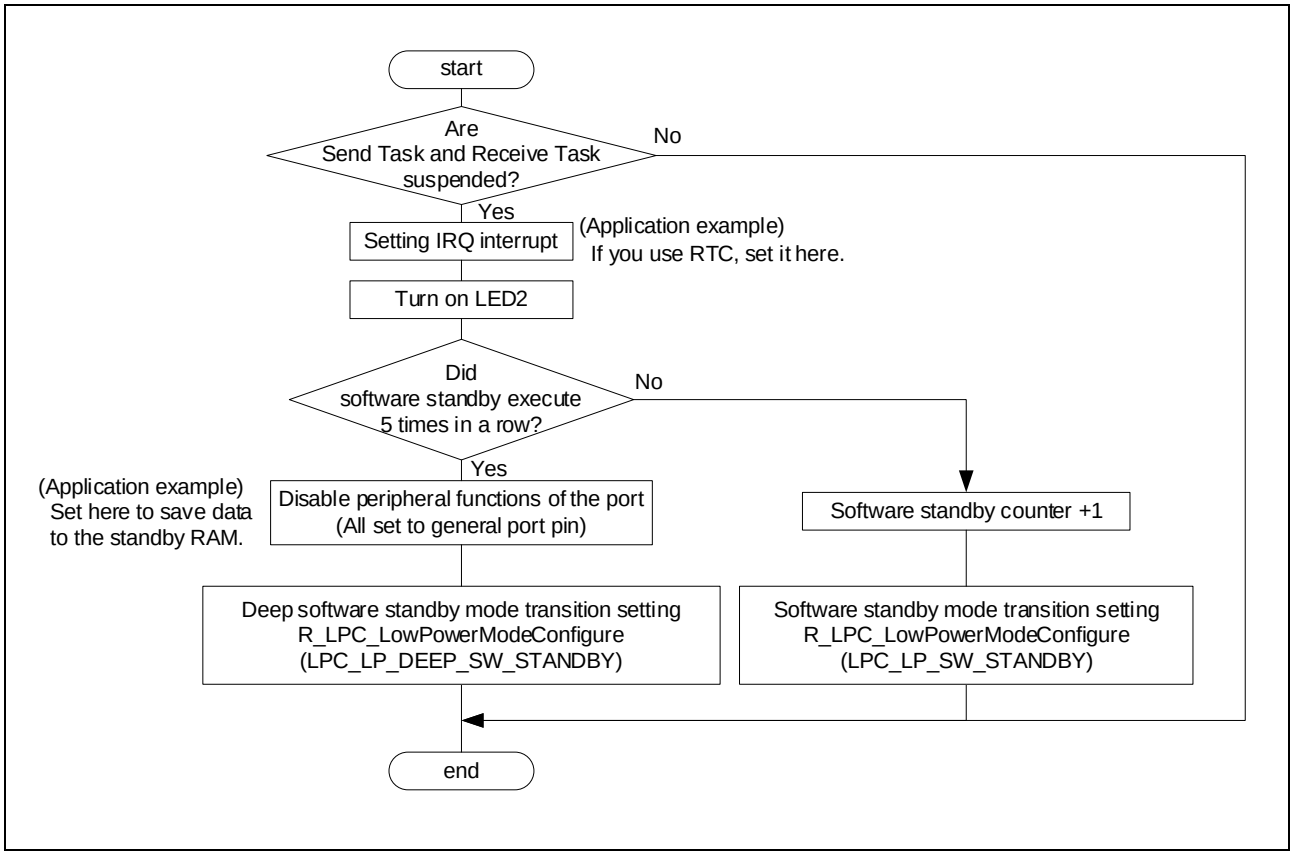

Figure 5-9 Case 3. The Flowchart of configPRE_SLEEP_PROCESSING()

5.3 Explanation of the Programs

5.3.1 Structure of the Folders

The folder structure of this sample project is shown below.

```
r01an6271xx0110-rtos-lowpower
├─── amazon-freertos
│      ├─── CHANGELOG.md
│      ├─── CMakeLists.txt
│      ├─── CODE_OF_CONDUCT.md
│      ├─── CONTRIBUTING.md
│      ├─── LICENSE
│      ├─── PreLoad.cmake
│      ├─── README.md
│      ├─── checksums.json
│      ├─── demos
│      ├─── directories.txt
│      ├─── doc
│      ├─── freertos_kernel
│      ├─── libraries
│      ├─── projects
│      ├─── tools
│      └─── vendors
├─── amazon-freertos-ota
├─── r01an6271ej0110-rtos-lowpower.pdf
└─── r01an6271jj0110-rtos-lowpower.pdf
```

5.3.2 Structure of the Files

The major files and folders used in this sample project are shown below.

Table 5-2 List of Files

Path	File Name and Folder Name	Overview
${application}¥	renesas_demo.c	Demo operation processing of the Sample Project
	renesas_demo.h	Operation setting definition file of the Sample Project
	renesas_sensors.c	On-board sensor control of RX65N Cloud Kit
	renesas_sensors.h	Definition file of communication processing with the on-board sensor of RX65N Cloud Kit
	renesas_code¥renesas_demo_lowpwr.c	Low power consumption transition processing
	sensorlib¥*.c, *.h	RX65N Cloud Kit installed sensor individual communication processing
${smc_gen}¥	r_config¥r_riic_rx_config.h	Configuration file of r_iic_rx
	r_config¥r_riic_rx_pin_config.h	Configuration file of r_iic_rx
	r_config¥r_cellular_config.h	Configuration file of r_cellular
	r_config¥r_irq_rx_config.h	Configuration file of r_irq_rx
${ports}¥	wifi¥iot_wifi.c	Setting RYZ014A eDRX and PSM mode when network initializing
${driver}¥	r_lpc_rx¥*.c, *.h	LPC FITModule (RX65N Low Power Consumption module)
	r_cellular¥*.c, *.h	RYZ014A FITModule
	r_irq_rx¥*.c, *.h	IRQ FITModule (Used for the interrupt to wake RX65N up from the low power mode)
	r_iic_rx¥*.c, *.h	RIIC FITModule (Used for communication with sensors)

The file and folder paths in the table are as follows.

${application}: ¥vendors¥renesas¥boards¥${board}¥aws_demos¥application_code

${smc_gen}: ¥vendors¥renesas¥boards¥${board}¥aws_demos¥src¥smc_gen

${ports}: ¥vendors¥renesas¥boards¥${board}¥ports

${driver}: ¥vendors¥renesas¥rx_driver_package¥v133

${board}:(When using CC-RX as the compiler) rx65n-cloud-kit-cellular

(When using GCC for Renesas RX as the compiler) rx65n-cloud-kit-cellular-gcc

5.3.3 List of the Constants

The constants used in this sample project are shown below.

Table 5-3 List of Constants

Name of Constants	Setting value	Contents
configUSE_TICKLESS_IDLE	1	Enable Tickless Idle Mode of FreeRTOS
R_LPC_SLEEP	0	Low power consumption mode: Sleep
R_LPC_SOFTSTBY	1	Low power consumption mode: Software standby
R_LPC_DEEPSTBY	2	Low power consumption mode: Deep software standby
R_CELLULAR_LPC_OFF	0	RYZ014A Low power consumption mode: Off
R_CELLULAR_LPC_ON	1	RYZ014A Low power consumption mode: On
R_DEMO_SHADOW_UPDATE_SEC	5	Data transmission interval to AWS (seconds)
R_DEMO_DEEP_COUNT	5	Number of software standby transitions before transitioning to deep software standby in the Case 3.

The constants with different settings for each case are shown below.

Table 5-4 Constants Setting for Case 1.

Name of Constants	Setting value	Contents
R_DEMO_LPCMODE	R_LPC_SLEEP	RX65N Sleep Mode
R_DEMO_CELLULAR_PSM	R_CELLULAR_LPC_OFF	RYZ014A PSM Mode Off
R_DEMO_CELLULAR_EDRX	R_CELLULAR_LPC_OFF	RYZ014A eDRX Mode Off

Table 5-5 Constants Setting for Case 2.

Name of Constants	Setting value	Contents
R_DEMO_LPCMODE	R_LPC_SOFTSTBY	RX65N Software Standby
R_DEMO_CELLULAR_PSM	R_CELLULAR_LPC_OFF	RYZ014A PSM Mode Off
R_DEMO_CELLULAR_EDRX	R_CELLULAR_LPC_ON	RYZ014A eDRX Mode On

Table 5-6 Constants Setting for Case3.

Name of Constants	Setting value	Contents
R_DEMO_LPCMODE	R_LPC_DEEPSTBY	RX65N Deep Software Standby
R_DEMO_CELLULAR_PSM	R_CELLULAR_LPC_ON	RYZ014A PSM Mode On
R_DEMO_CELLULAR_EDRX	R_CELLULAR_LPC_OFF	RYZ014A eDRX Mode Off

5.3.4 List of the Variables

The variables used in this sample project are shown below.

Table 5-7 List of Variables

Type	Name of Variables	Contents
uint8_t	shadow_init_flg	AWS IoT Device shadow initializing flag 0: Delete device shadow and recreate 1: Not delete device shadow
uint32_t	softstby_counter	Software standby transition count counter Used in case 3.
static uint32_t	ulCurrentPowerOnState	Status of LED1 0: LED1 On 1: LED1 Off
static bool	stateChanged	LED1 status change flag false: No change true: There is a change (Send the changed state to AWS)

5.3.5 List of the Functions

The functions used in this sample project are shown below.

Table 5-8 List of the Functions

Name of Functions	File Name	Overview
RunRenesasDemo	renesas_demo.c	Main processing functions for sensing and communication with AWS
sleep_pre	renesas_demo_lowpwr.c	Preprocessing function for low power consumption transition * 1
sleep_post	renesas_demo_lowpwr.c	Post-processing function for low power consumption transition * 1
R_LPC_OperatingModeSet	LPC FIT module	Configure the Operating Power Control modes (at initializing)
R_LPC_LowPowerModeConfigure		Configure the low power consumption modes when the WAIT instruction is executed
R_LPC_LowPowerModeActivate		Activate the Low Power Consumption mode
R_IRQ_Open	IRQ FIT module	initialize the associated IRQ registers and enable interrupts (IRQ1 and IRQ4 are used)
R_IRQ_Close		disable the IRQ (Called at interrupt processing)

* 1 Define and use the following macros.

```
#define configPRE_SLEEP_PROCESSING( x ) {extern void sleep_pre(TickType_t time); sleep_pre(x);}
#define configPOST_SLEEP_PROCESSING( x )  {extern void sleep_post(TickType_t time); sleep_post(x);}
```

5.3.6 RYZ014A Cellular FIT module changes

In the current RYZ014A Cellular FIT module (R01AN6324) Rev.1.06, it is necessary to change the processing related to the low consumption mode for the sample project operation. The changes made in the sample project are shown below. These changes will be supported by the revision of the FIT module, so please check the source code with your version.

Table 5-9 RYZ014A Cellular FIT module changes

Path	File Name	Overview	Reason
${driver}¥	r_cellular_setpsm.c	Change AT+CMER command parameter “ind” to 0	To suppress the notification from RYZ014A and react only to the notification of AWS operation when RX65N is in the low power consumption mode.
	r_cellular_setedrx.c	Added sending the following command AT+SQNRICFG AT+CMER AT+SQNIPSCFG	To enable notification of AWS operations when in eDRX mode

${driver}: ¥vendors¥renesas¥rx_driver_package¥v133¥r_cellular¥src¥api

(1) r_cellular_setpsm.c

Change the atc_cmer () argument to 0 in the cellular_psm_config () function.

```
static e_cellular_err_t cellular_psm_config(st_cellular_ctrl_t * const p_ctrl, const e_cellular_psm_mode_t mode)
{
    e_cellular_err_t ret = CELLULAR_SUCCESS;

    if (CELLULAR_PSM_MODE_ACTIVE == mode)
    {
        ret = atc_sqnricfg(p_ctrl, CELLULAR_PSM_MODE_ACTIVE);

        if (CELLULAR_SUCCESS == ret)
        {
//            ret = atc_cmer(p_ctrl, CELLULAR_PSM_MODE_ACTIVE);
            ret = atc_cmer(p_ctrl, (e_cellular_psm_mode_t)0);   // modified
        }
        if (CELLULAR_SUCCESS == ret)
        {
            ret = atc_sqnipscfg(p_ctrl, CELLULAR_PSM_MODE_ACTIVE);
        }
    }
```

(2) r_cellular_setedrx.c

1. Create a new cellular_edrx_config () function.

Function declaration

```
static e_cellular_err_t cellular_edrx_config(st_cellular_ctrl_t * const p_ctrl, const e_cellular_edrx_mode_t mode);
```

Function description

```
static e_cellular_err_t cellular_edrx_config(st_cellular_ctrl_t * const p_ctrl, const e_cellular_edrx_mode_t mode)
{
    e_cellular_err_t ret = CELLULAR_SUCCESS;
    e_cellular_psm_mode_t mode_psm;

    if (CELLULAR_EDRX_MODE_ACTIVE == mode)
    {
        mode_psm = CELLULAR_PSM_MODE_ACTIVE;
        ret = atc_sqnricfg(p_ctrl, mode_psm);

        if (CELLULAR_SUCCESS == ret)
        {
            ret = atc_cmer(p_ctrl, (e_cellular_psm_mode_t)0);
        }
        if (CELLULAR_SUCCESS == ret)
        {
            ret = atc_sqnipscfg(p_ctrl, mode_psm);
        }
    }

    return ret;
}
```

2. Added cellular_edrx_config () function call in R_CELLULAR_SetEDRX () function.

```
    if (CELLULAR_SUCCESS == ret)
    {
        semaphore_ret = cellular_take_semaphore(p_ctrl->at_semaphore);
        if (CELLULAR_SEMAPHORE_SUCCESS == semaphore_ret)
        {
            ret = cellular_edrx_config(p_ctrl, mode);       // modified
            if (CELLULAR_SUCCESS == ret)                    // modified
            {
                ret = atc_sqnedrx(p_ctrl, mode, edrx, ptw);
            }
            cellular_give_semaphore(p_ctrl->at_semaphore);
        }
        else
        {
            ret = CELLULAR_ERR_OTHER_ATCOMMAND_RUNNING;
        }
    }
```

5.4 Results of the Operation

5.4.1 Current Measurement Environment

The following shows how to measure the current consumption using the evaluation board.

Please Use RYZ014A EVK (not PMOD version) when measuring current. This is because in the PMOD version, the currents of other mounted parts on the PMOD board are seen as offsets.

Table 5-10 Current Measurement Environment

Measurement Target	Evaluation Board	Measurement Terminal	Modify of the Hardware
RX65N	RX65N Cloud Kit	JP2 on the lower side of the Cloud Kit (Target Board)	Needed Refer to Figure 5-10
RYZ014A	RYZ014A-EVK	ST3	No need Refer to Figure 5-11

As shown in Figure 5-10, cut SS6 of RX65N Target Board and connect an ammeter to JP2.

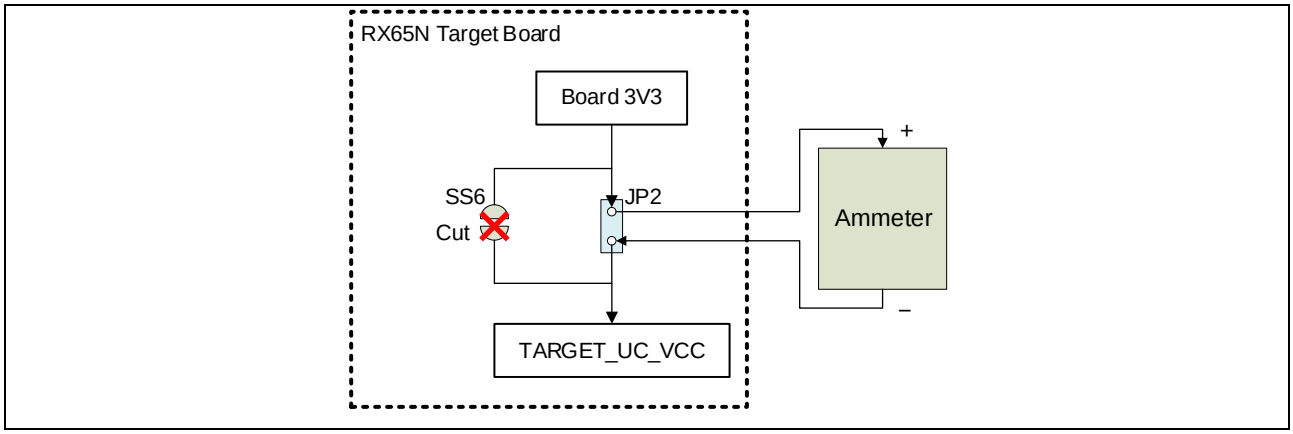

Figure 5-10 Ammeter Connection for RX65N

As shown in Figure 5-11, connect an ammeter to ST3 of RYZ014A-EVK.

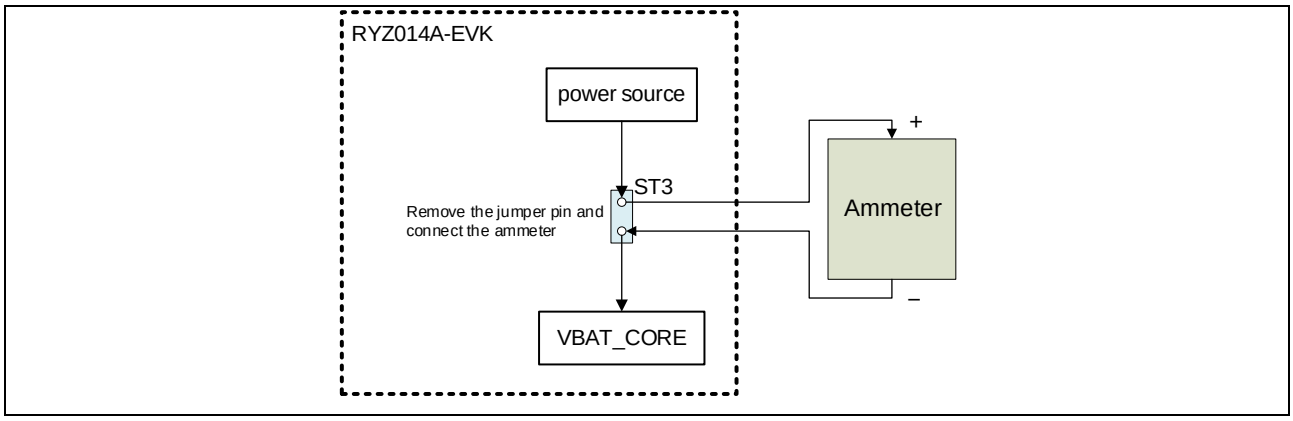

Figure 5-11 Ammeter Connection for RYZ014A

Note: When using RYZ014A-EVK, change r_cellular_config.h to the following setting.

```
#define CELLULAR_CFG_RESET_SIGNAL_LOGIC        (0)
```

5.4.2 Operation of Sample Project

5.4.2.1 Communication to AWS

In the sample project, the operation of each case can be performed by setting Table 5-4, Table 5-5 and Table 5-6. The data sent from the sample project to AWS is shown below.

Table 5-11 Device Shadow Property List

Property	Status	Operation
"temperature"	For example: 24.55	Environment sensor temperature data (degree C)
"powerOn"	0	LED1 Lighting state
	1	LED1 Off state

LED1 can change its state by operating Device Shadow.

The procedure for operating Device Shadow is shown below.

1. Execute **4.** Connecting to AWS

2. Run the sample project.

3. On the AWS Management Console select **Services → All services → IoT → IoT Core**, then click **Test → MQTT test client → Publish to a topic**, enter the following code in the topic name.

   Note: For the Things name “xxxx”, enter the name of the thing registered as described in **4.1 AWS Preparation** .

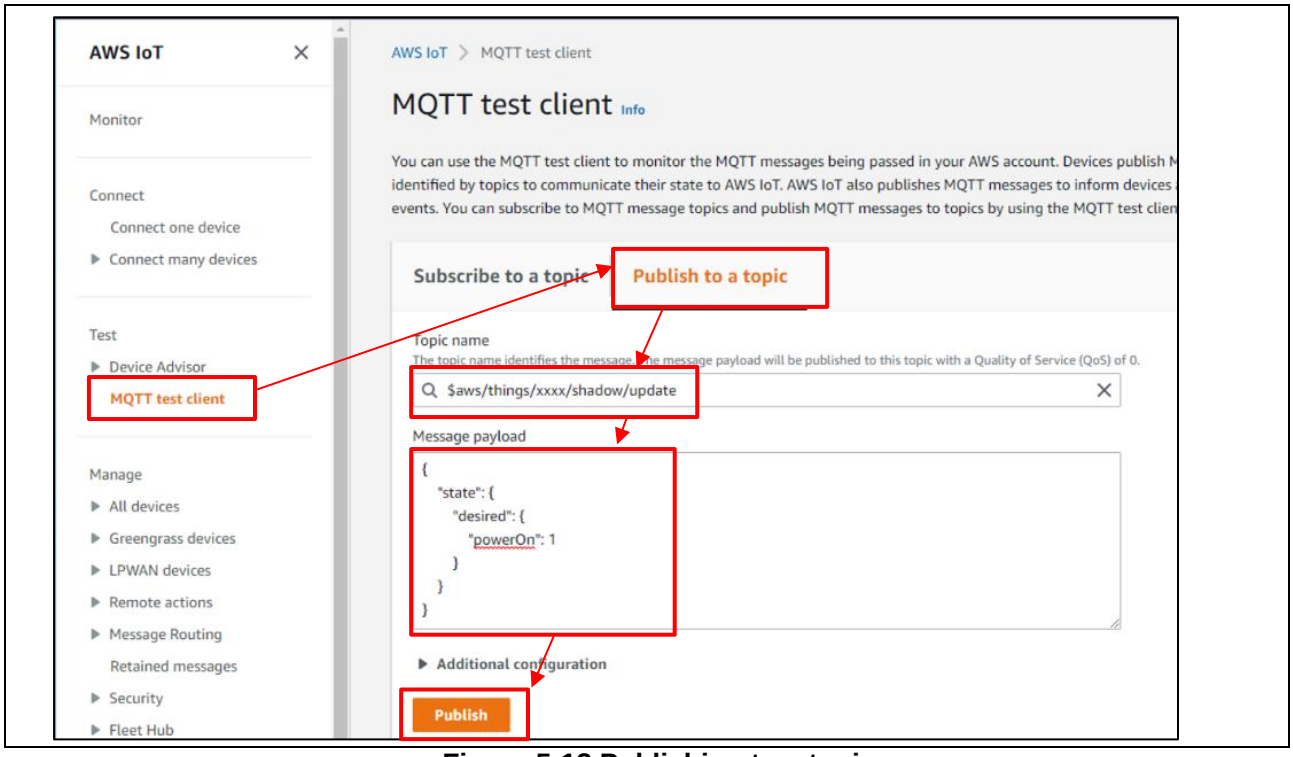

Figure 5-12 Publishing to a topic

```
$aws/things/xxxx/shadow/update
```

Figure 5-13 Topic Name

```
{
    "state": {
        "desired": {
            "powerOn": 1
        }
    }
}
```

Figure 5-14 Message Payload

4. Click **Subscribe**, make sure that LED1 of the RX65N Cloud Kit goes off.

5.4.2.2 Confirmation of transmitted data to AWS

"How to check the data uploaded to AWS" is shown below.

Select **Services** → **All services** → **IoT** → **IoT Core**, then click **Thing**.

The name of the item registered in **4.1 AWS Preparation** is displayed, so click it.

You can check the uploaded data by clicking **Device Shadow** on the transition destination screen.

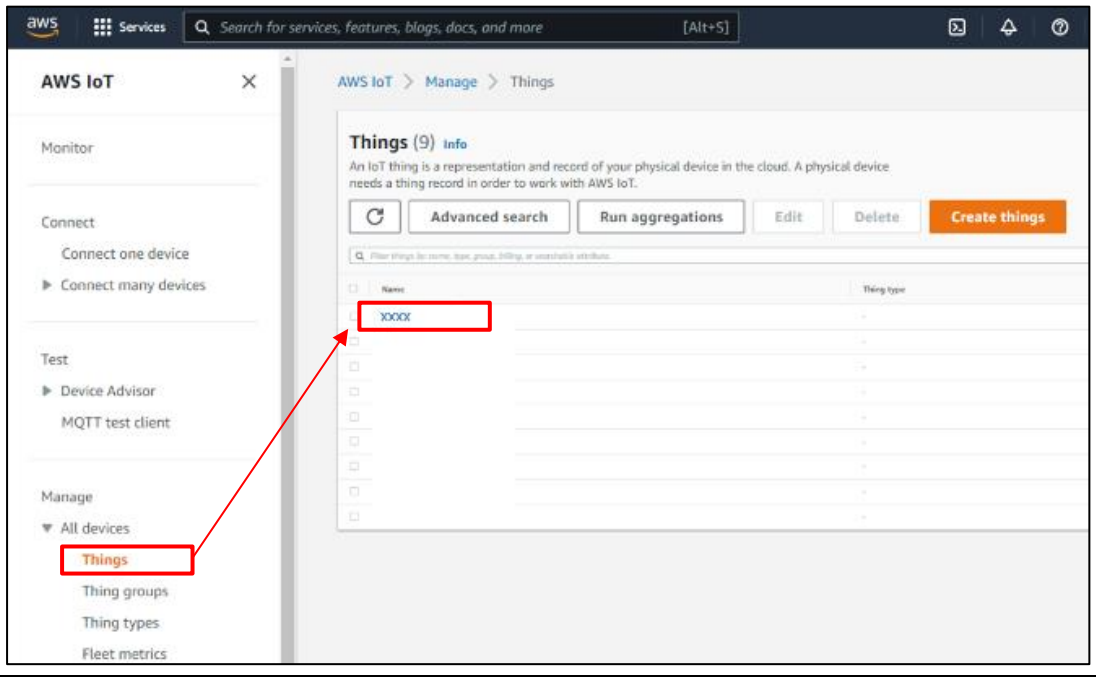

Figure 5-15 Selecting the Thing on AWS IoT Core

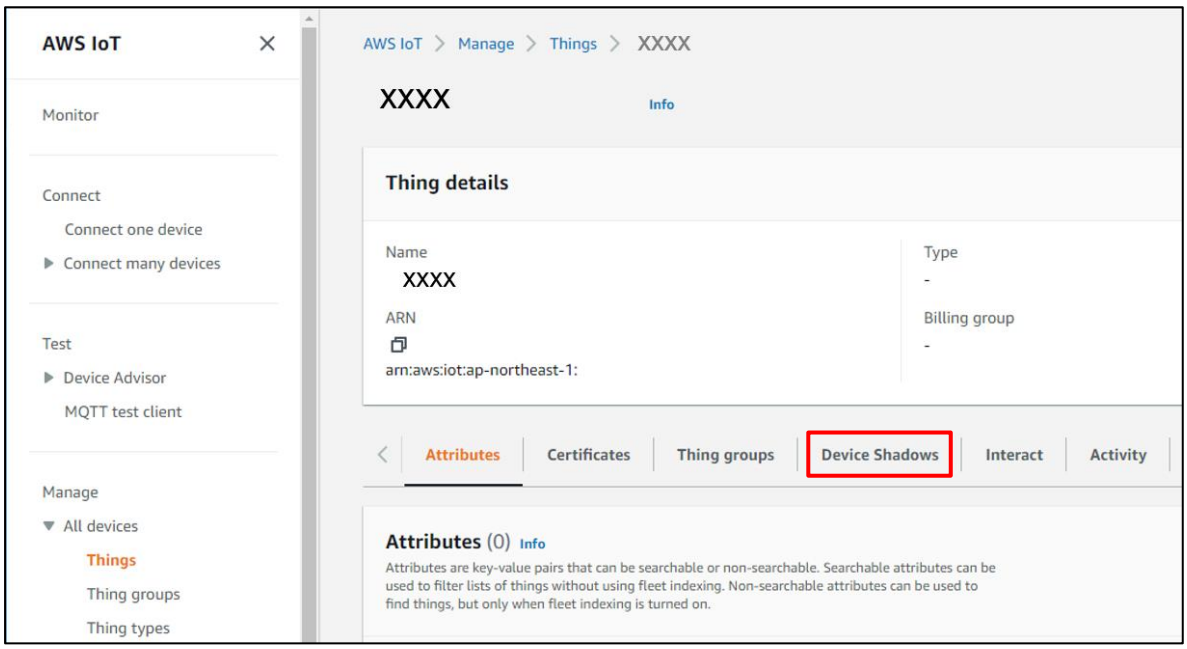

Figure 5-16 Selecting Device Shadow

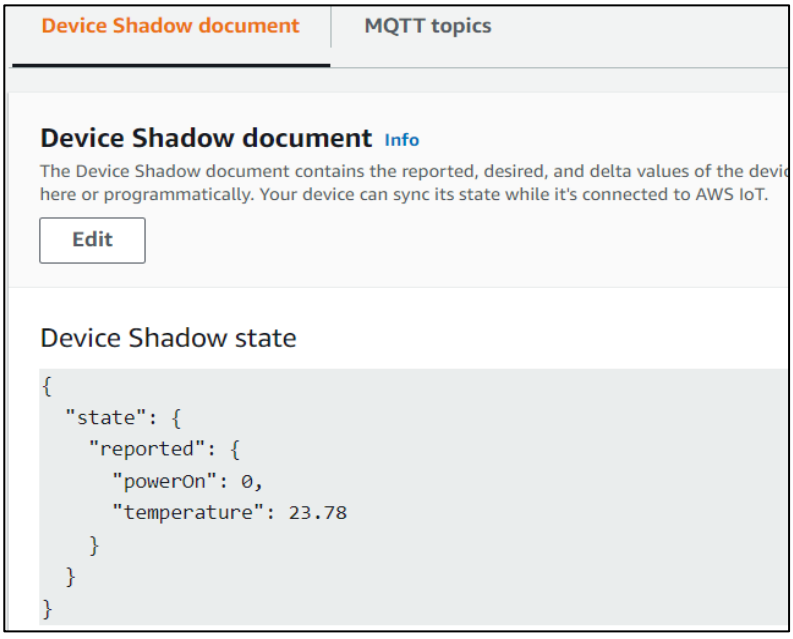

Figure 5-17 Status of Device Shadow

5.4.2.3 Execution of Sample Project

The operation contents for each case of the sample project are explained below.

To change the AWS communication interval in Case 1, change Table 5-3 R_DEMO_SHADOW_UPDATE_SEC.

To return from the software standby mode and deep software standby mode of Case 2 and Case 3, press the USER SW (see Figure 5-2 RX65N Cloud Kit LED1/2, USERSW for the position) at the top of the RX65N Cloud Kit. It is also possible to return from the software standby mode of Case 2 by changing the status from AWS as in 5.4.2.1.You can see the operating status of RX65N by checking LED2. (LED2 lights up during software standby mode and deep software standby mode, and turns off during program operation)

Note: The status change notifications from AWS will be accepted when RYZ014A is in the Paging. If you use a lot of the software standby mode return operation by changing the state, you can reduce the delay of return by setting a long Paging period of RYZ014A eDRX. The eDRX settings for RYZ014A are set in Table 5-2 iot_wifi.c. Refer to the application note “RYZ014A Cellular FIT module (R01AN6324)” for the set value.

Note: Please note that if the RX65N is in software standby mode or deep software standby mode and you do not press the USER SW for 20 minutes or more, the connection with AWS will be disconnected. If the connection is lost, the program will reconnect.

In Case 3, after the software standby transition is performed for the number of times in Table 5-3 R_DEMO_DEEP_COUNT, if there is no notification of the status change from AWS IoT Device Shadow, the transition to deep software standby mode is performed. If you want to change the number of software standby modes, set R_DEMO_DEEP_COUNT.

The operation image of each case is shown below.

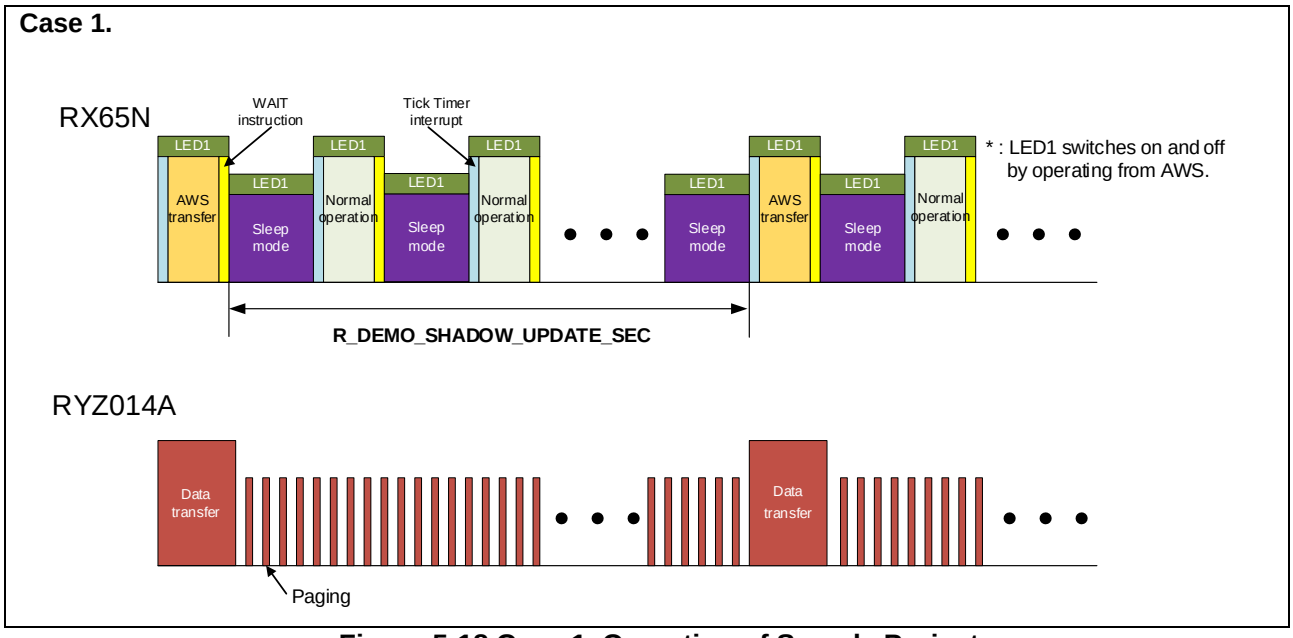

Figure 5-18 Case 1. Operation of Sample Project

Case 2.

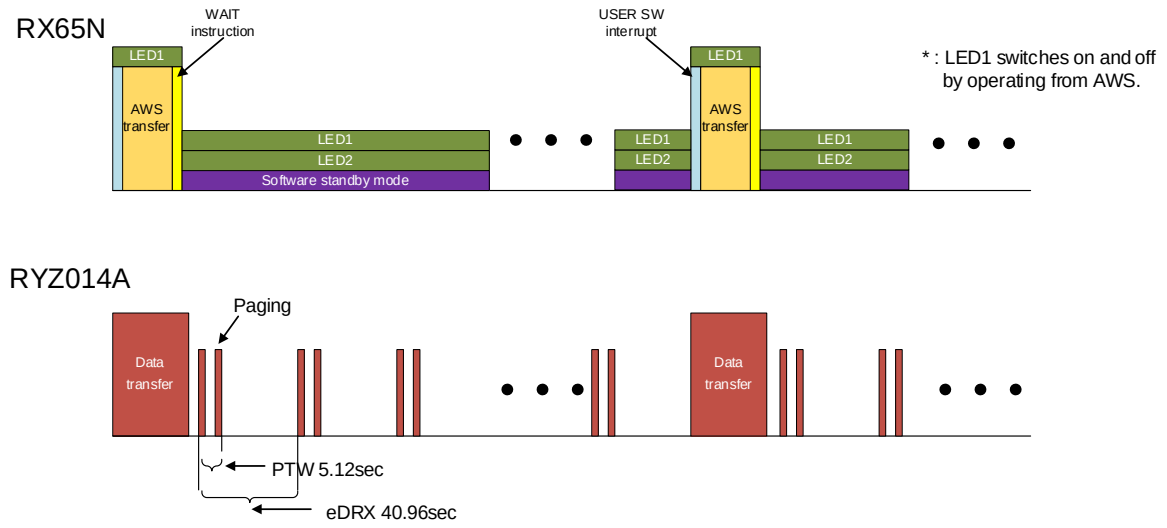

Figure 5-19 Case 2. Operation of Sample Project

Case 3.

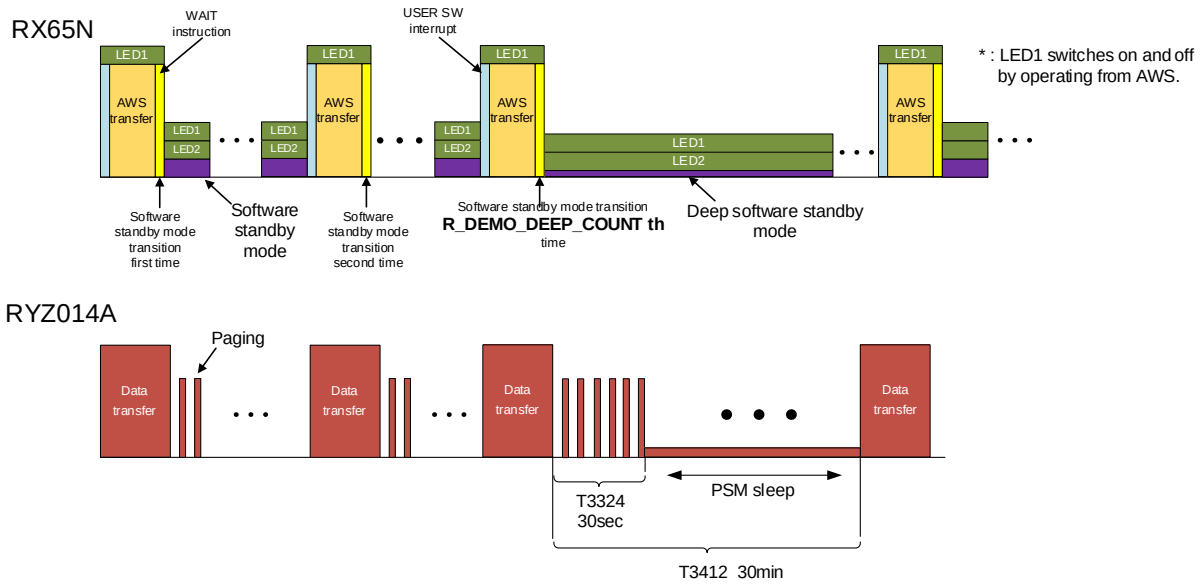

Figure 5-20 Case 3. Operation of Sample Project

5.5 How to add low power consumption code

This section describes the procedure for enabling the low power consumption function and the procedure for adding the code of the sample project to the standard FreeRTOS project, such as when creating a FreeRTOS project by e^2 studio.

5.5.1 Adding low consumption functions

The files and folders related to low power consumption are shown below.

Table 5-12 List of Files related to low power consumption

Path	File Name and Folder Name	Overview
${kernel}¥	port.c	The low power consumption operation code of FreeRTOS is described
${driver}¥	r_lpc_rx¥*.c, *.h	LPC FITModule
	r_irq_rx¥*.c, *.h	IRQ FITModule (Used for the interrupt to wake RX65N up from the low power mode)
${board}¥	ports¥wifi¥iot_wifi.c	Setting RYZ014A eDRX and PSM mode when network initializing
	aws_demos¥config_files¥FreeRTOSConfig.h	Enable setting the low power consumption function of FreeRTOS
${application}¥	renesas_code¥renesas_demo_lowpwr.c, .h	Low power consumption transition processing

The file and folder paths in the table are as follows.

(When using CC-RX as the compiler)

${kernel}: ¥freertos_kernel¥portable¥Renesas¥RX600v2

${board}: ¥vendors¥renesas¥boards¥rx65n-cloud-kit-cellular

(When using GCC for Renesas RX as the compiler)

${kernel}: ¥freertos_kernel¥portable¥GCC¥RX600v2

${board}: ¥vendors¥renesas¥boards¥rx65n-cloud-kit-cellular-gcc

${driver}: ¥vendors¥renesas¥rx_driver_package¥v133

${application}: ${board}¥aws_demos¥application_code

Note: The above file paths are the paths of the sample project. Please note that the paths may be different for other projects.

Note: The sample project's port.c uses CMT ch0 for the FreeRTOS system timer. The FreeRTOS system timer can change the CMT channel with the BSP_CFG_RTOS_SYSTEM_TIMER described in RX Family Board Support Package Module Using Firmware Integration Technology 3.2.5 RTOS. When changing the CMT channel, change the CMT channel in port.c according to the channel to be used.

5.5.2 Setting low power consumption functions

To enable the low consumption function, set it to the state shown in Figure 5-21 in FreeRTOSConfig.h.

To disable the low consumption function, set it to the state shown in Figure 5-22.

```
/* configUSE_TICKLESS_IDLE = 1: enable FreeRTOS's built in tickless idle functionality. */
#define configUSE_TICKLESS_IDLE    1

/* Tickless Idle configuration. */
#define configEXPECTED_IDLE_TIME_BEFORE_SLEEP   2

/* If configUSE_TICKLESS_IDLE is 1, uncomment the following function.
 * The sleep_pre(TickType_t time) and sleep_post(TickType_t time) functions
 * should be implemented by the user. */
#define configPRE_SLEEP_PROCESSING( x ) {extern void sleep_pre(TickType_t time); sleep_pre(x);}
#define configPOST_SLEEP_PROCESSING( x )  {extern void sleep_post(TickType_t time); sleep_post(x);}
```

Figure 5-21 Enable low power function

```
/* configUSE_TICKLESS_IDLE = 1: enable FreeRTOS's built in tickless idle functionality. */
#define configUSE_TICKLESS_IDLE    0

/* Tickless Idle configuration. */
#define configEXPECTED_IDLE_TIME_BEFORE_SLEEP   2

/* If configUSE_TICKLESS_IDLE is 1, uncomment the following function.
 * The sleep_pre(TickType_t time) and sleep_post(TickType_t time) functions
 * should be implemented by the user. */
#define configPRE_SLEEP_PROCESSING( x ) {/*extern void sleep_pre(TickType_t time); sleep_pre(x);*/}
#define configPOST_SLEEP_PROCESSING( x )  {/*extern void sleep_post(TickType_t time); sleep_post(x);*/}
```

Figure 5-22 Disable low power function

The RX65N's low power mode transition is set using the sleep_pre () and sleep_post () functions.
Please describe the necessary processing in the function according to the system. In the sample project, it is described in renesas_demo_lowpwr.c according to the usage of the Case 1 to Case 3.

5.5.3 Adding the code of the sample project

The files and folders for operating the sample project are shown below. When uploading the sensing value using the sensor installed in the RX65N Cloud Kit, add the following file to the project.

Table 5-13 List of Files for operating the sample project

Path	File Name and Folder Name	Overview
${application}¥	renesas_demo.c	Demo operation processing of the Sample Project
	renesas_demo.h	Operation setting definition file of the Sample Project
	renesas_sensors.c	On-board sensor control of RX65N Cloud Kit
	renesas_sensors.h	Definition file of communication processing with the on-board sensor of RX65N Cloud Kit
	renesas_code¥renesas_demo_lowpwr.c	Low power consumption transition processing
	sensorlib¥*.c, *.h	RX65N Cloud Kit installed sensor individual communication processing
${smc_gen}¥	r_config¥r_riic_rx_config.h	Configuration file of r_iic_rx
	r_config¥r_riic_rx_pin_config.h	Configuration file of r_iic_rx
${driver}¥	r_iic_rx¥*.c, *.h	RIIC FITModule (Used for communication with sensors)

The file and folder paths in the table are as follows.

${application}: ¥vendors¥renesas¥boards¥${board}¥aws_demos¥application_code

${smc_gen}: ¥vendors¥renesas¥boards¥${board}¥aws_demos¥src¥smc_gen

${driver}: ¥vendors¥renesas¥rx_driver_package¥v133

${board}:(When using CC-RX as the compiler) rx65n-cloud-kit-cellular

(When using GCC for Renesas RX as the compiler) rx65n-cloud-kit-cellular-gcc

6. Sample Program for Low Consumption Operation & OTA Update Using CK-RX65N Board

6.1 Overview of This Sample Program

This section provides an overview of the sample program for low consumption operation & OTA update using CK-RX65N.

6.1.1 Specifications of the Operation

The connection diagram of the device used in the operation of the sample program is shown below.

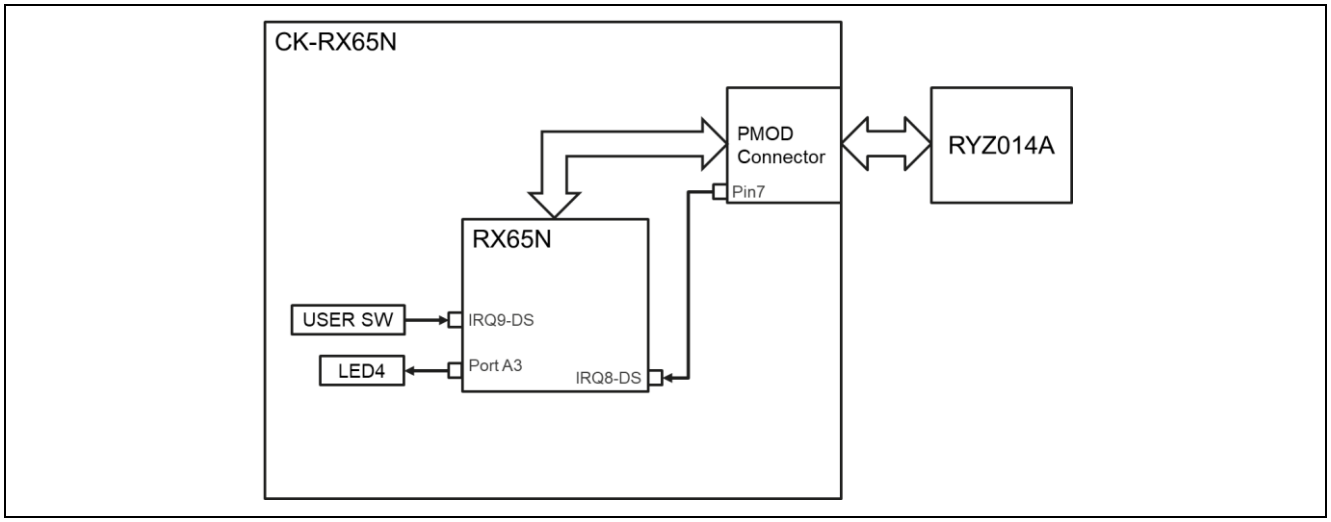

Figure 6-1 Connection Diagram

The sample program runs on Amazon FreeRTOS on the RX65N. It connects to AWS IoT Core via RYZ014A and waits for an OTA job from AWS.

LED4 is lit when the RX65N is in software standby or deep software standby mode. Software standby or deep software standby mode is released by interrupts (IRQ9-DS:USER SW, IRQ8-DS: RYZ014A RING0 signal operated from AWS).

The positions of USER SW and LED4 of CK-RX65N are as follows.

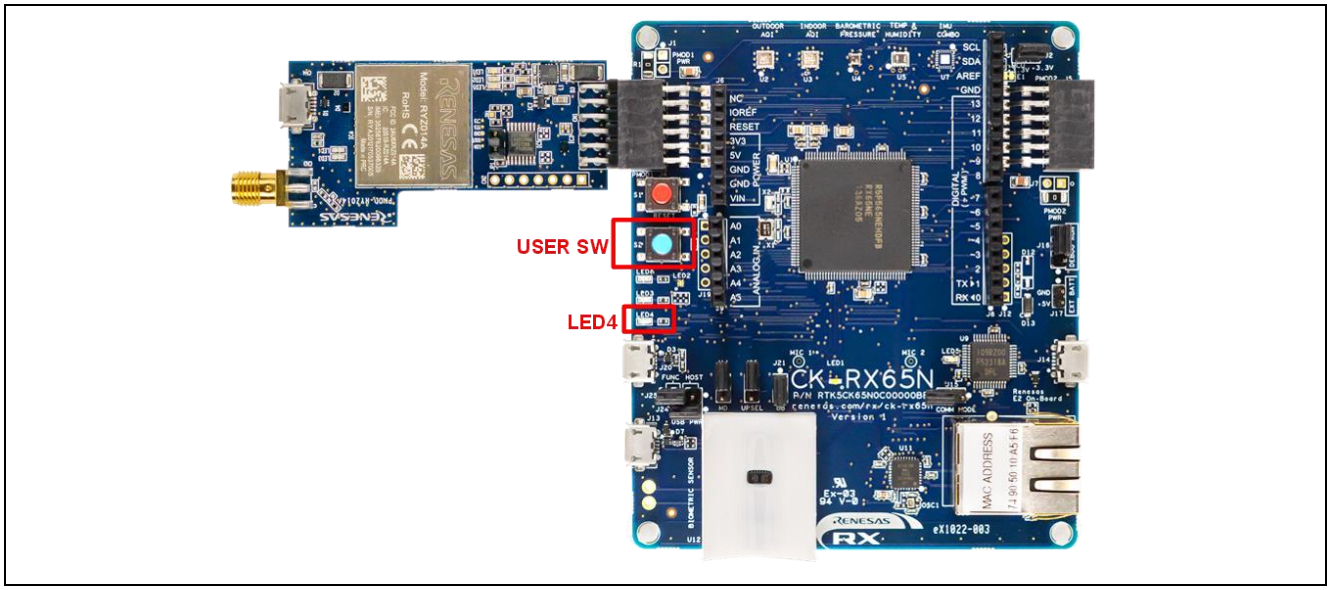

Figure 6-2 CK-RX65N USER SW, LED4

6.1.2 Diagram of the System

The system diagram for this sample program is similar to Figure 1-5, but uses the CK-RX65N instead of the RX65N Cloud Kit.

6.1.3 Flowchart of the Operation

The flowchart of the overall operation and the low power consumption mode transition in each mode of the sample project are shown below.

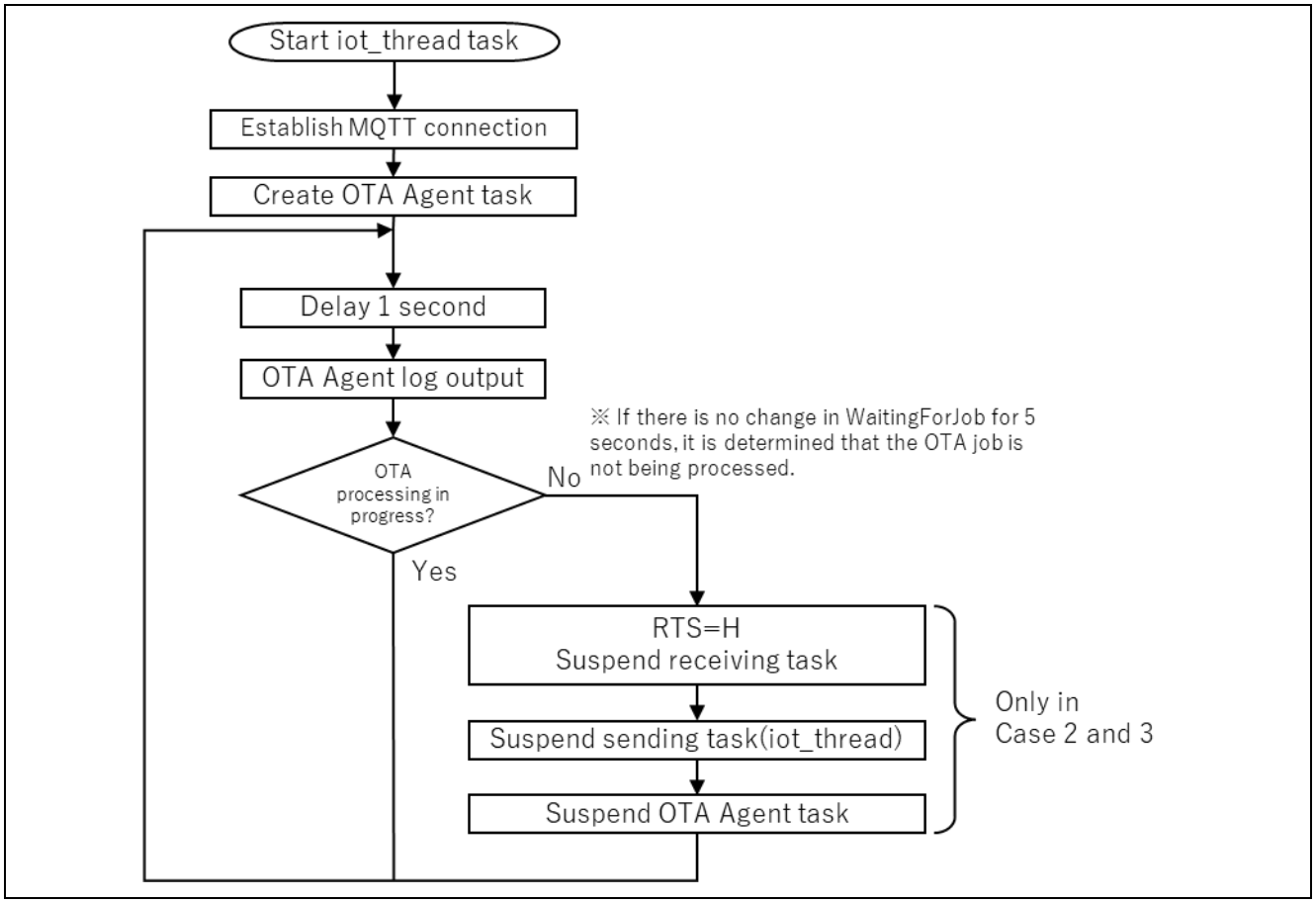

Figure 6-3 Flowchart of OTA Job Waiting Operation

The flowchart for the Tickless idle mode process for IDLE task is similar to Figure 5-5 and the flowchart for low power consumption mode setting is similar to Figure 5-6.

6.1.4 Configuration of the Sample Programs

6.1.4.1 Hardware Configuration

The hardware configuration of this system is shown in the table below.

Table 6-1 Hardware Configurations

Item	Content	Provider	Description
Board used	CK-RX65N	Renesas Electronics Corporation	Evaluation board with RX65N MCU Note
Cellular module	RYZ014A Pmod Module		Communication board with RYZ014A module Note
SIM	LTE Cat-M1	-	LTE Cat-M1 compatible SIM card Card size of SIM : microSIM
PC	Windows 10	-	Recommendation OS
	Google Chrome	-	Browser to use
Note: CK-RX65N and RYZ014A Pmod Module are included in the CK-RX65N			

CK-RX65N Web:

https://www.renesas.com/jp/ja/products/microcontrollers-microprocessors/rx-32-bit-performance-efficiency-mcus/ck-rx65n-cloud-kit-based-rx65n-mcu-group

6.1.4.2 Software Configuration

The software configuration of this system is shown in the table below.

Table 6-2 Software Configurations

Item	Content	Version
Integrated development environment	e^2 studio	2022-04
Compiler	CC-RX	V3.04
Communication software	Tera Term	Version4.105
Emulator	E2 emulator Lite (on-board)	-
RTOS	Amazon FreeRTOS	Version 202107.00-rx-1.0.1

6.1.4.3 Tera Term configurations

Set Tera Term as shown in the table below.

Table 6-3 Tera Term Configurations

Item	Setting
Baud Rate	115200
Data length	8bit
Parity	none
Stop Bit	1bit
Flow Control	none

6.1.4.4 Current Measurement Environment

Please refer to 6.1 MCU and Pmod Current Measurement in the CK-RX65N User's Manual for information on how to measure the RX65N MCU current consumption using the CK-RX65N evaluation board.

CK-RX65N v1 – User's Manual:

https://www.renesas.com/us/en/document/mat/ck-rx65n-v1-users-manual

6.2 OTA Sample Project

6.2.1 Explanation of the Programs

6.2.1.1 Structure of the Folders

The folder structure of this sample project is shown below.

```
r01an6271xx0110-rtos-lowpower
 ├──── amazon-freertos
 ├──── amazon-freertos-ota
 │      ├──── CHANGELOG.md
 │      ├──── CMakeLists.txt
 │      ├──── CODE_OF_CONDUCT.md
 │      ├──── CONTRIBUTING.md
 │      ├──── LICENSE
 │      ├──── PreLoad.cmake
 │      ├──── README.md
 │      ├──── checksums.json
 │      ├──── demos
 │      ├──── directories.txt
 │      ├──── doc
 │      ├──── freertos_kernel
 │      ├──── libraries
 │      ├──── projects
 │      ├──── tools
 │      └──── vendors
 ├──── r01an6271ej0110-rtos-lowpower.pdf
 └──── r01an6271jj0110-rtos-lowpower.pdf
```

6.2.1.2 Structure of the Files

The major files and folders used in this sample project are shown below.

Table 6-4 List of Files

Path	File Name and Folder Name	Overview
${application}¥	renesas_demo.h	Operation setting definition file of the Sample Project
	renesas_code¥renesas_demo_lowpwr.c	Low power consumption transition processing
${demos}	ota¥ota_demo_core_mqtt¥ota_demo_core_mqtt.c	Control OTA processing

The file and folder paths in the table are as follows.

${application}: ¥vendors¥renesas¥boards¥$ck-rx65n-ryz014a¥aws_demos¥application_code

${demos}: ¥demos

6.2.1.3 List of the Constants

The constants used in this sample project are shown below.

Table 6-5 List of Constants

Name of Constants	Setting value	Contents
configUSE_TICKLESS_IDLE	1	Enable Tickless Idle Mode of FreeRTOS
R_LPC_SLEEP	0	Low power consumption mode: Sleep
R_LPC_SOFTSTBY	1	Low power consumption mode: Software standby
R_LPC_DEEPSTBY	2	Low power consumption mode: Deep software standby
R_CELLULAR_LPC_OFF	0	RYZ014A Low power consumption mode: Off
R_CELLULAR_LPC_ON	1	RYZ014A Low power consumption mode: On

The setting of constants for each case is similar to Table 5-4, Table 5-5, and Table 5-6.

6.2.1.4 RYZ014A Cellular FIT module changes

The OTA sample project has the same modifications to the RYZ014A Cellular FIT module as described in "5.3.6 RYZ014A Cellular FIT module changes".

6.2.2 How to build the project and run the demo

This section describes how to prepare the AWS side, CK-RX65N board and software required to run the demo.

6.2.2.1 AWS Preparation

The procedure is similar to "4.1 AWS Preparation". If you have already done these steps, you can use the settings as they are.

Also, please set up the environment for OTA execution by referring to "1.2 Create an Amazon S3 bucket" to "1.6 Grant access to code signing for AWS IoT" in " RX Family How to implement FreeRTOS OTA by using Amazon Web Services on RX65N".

6.2.2.2 Hardware Preparation

Please refer to "4. Set up your hardware" in "RX65N Connecting AWS Cloud with FreeRTOS Getting Started Guide for CK-RX65N" for configuration.

Also, in order to catch the USER SW and the RYZ014A RING0 signals with the DS interrupt source pin, which can be used as the triggers for release from deep software standby mode, please wire as follows.

PMOD-RYZ014A Pmod #7(RING0) – CK-RX65N J9 connector 1 pin(**Purple line in the figure below**)

CK-RX65N S2(USER SW) – CK-RX65N J9 connector 2 pin(**Green line in the figure below**)

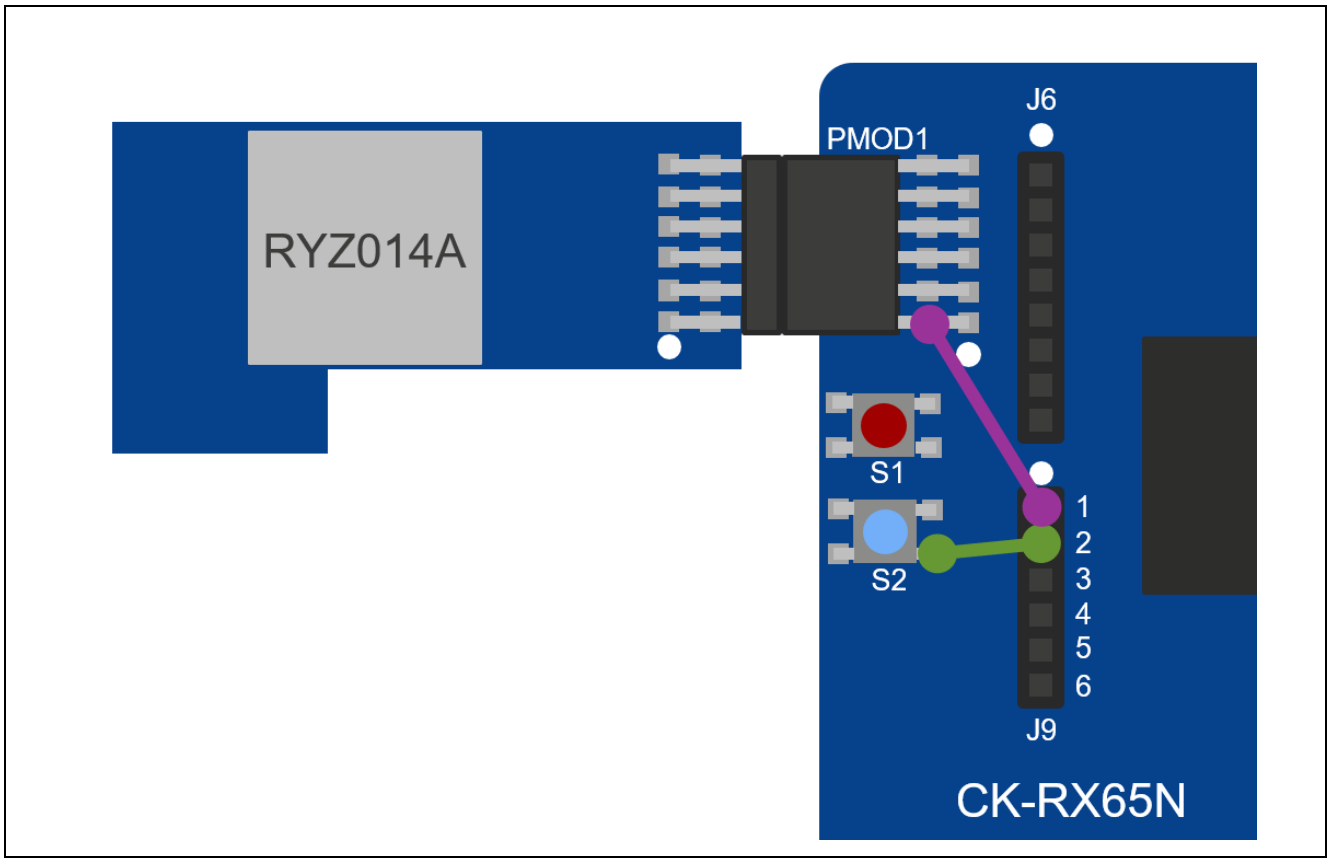

Figure 6-4 Wiring diagram for catching the USER SW and the RYZ014A RING0 signals with the DS interrupt source pin

6.2.2.3 Software Preparation

Please refer to "4.3 Software Preparation" from 1. to 5. Please specify the following root directory when importing projects.

${base_folder}¥amazon-freertos-ota¥projects¥renesas¥ck-rx65n-ryz014a¥e2studio

Select two projects, aws_demos and boot_loader, and click "Finish".

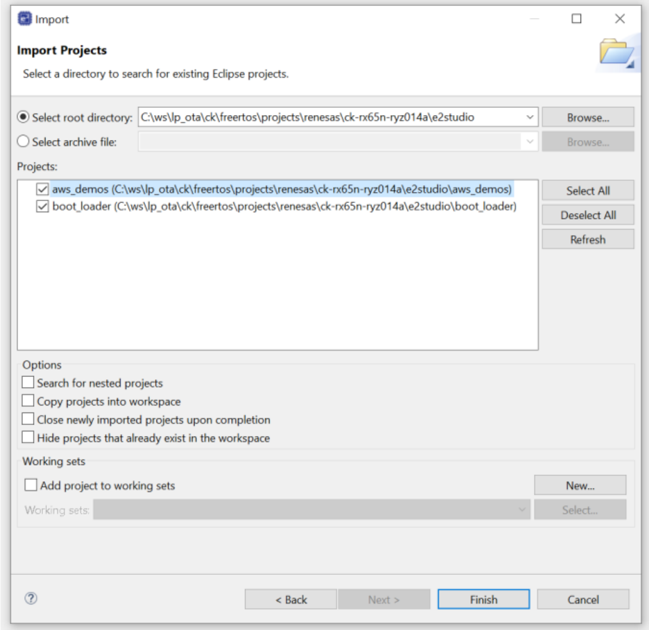

Figure 6-5 Importing projects

For the imported boot_loader project, perform step ⑪ "Input public key" of "2.1 Import, configurate head file and build aws_demos and boot_loader" in " RX Family How to implement FreeRTOS OTA by using Amazon Web Services on RX65N".

Next, perform steps from 6. to 10. in "4.3 Software Preparation" for the imported aws_demos project.

In addition, the configuration for OTA updates needs to be set to `otapalconfigCODE_SIGNING_CERTIFICATE` macro in the following file with the secp256r1.crt created in "6.2.2.1 AWS Preparation" in the following format.

${base_folder}¥vendors¥renesas¥boards¥ck-rx65n-ryz014a¥aws_demos¥config_files¥ota_demo_config.h

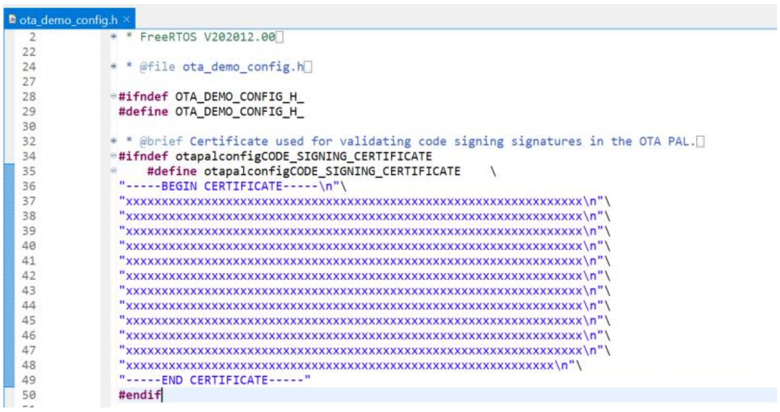

Figure 6-6 format for otapalconfigCODE_SIGNING_CERTIFICATE macro

Perform 11. in "4.3 Software Preparation" to build.

6.2.2.4 Firmware Programming

Perform steps from ⑤ to ⑧ of "2.2 Install the initial version of firmware" in "RX Family How to implement FreeRTOS OTA by using Amazon Web Services on RX65N", write the firmware to the RX65N, and confirm that the log from the CK-RX65N is output in Tera Term.

6.2.3 Results of the Operation

6.2.3.1 Operation of Sample Project

The sample project runs by default with "Table 5-5 Constants Setting for Case 2."

By changing to the settings in Table 5-4 and Table 5-6, the operation can be performed in each case.

The sample project can be OTA updated by executing an OTA job from AWS. Please refer to "2.3 Update the version of your firmware" in "RX Family How to implement FreeRTOS OTA by using Amazon Web Services on RX65N" for the procedure.

Note: The definition of the APP_VERSION_BUILD macro has been moved to the following file.

${base_folder}¥vendors¥renesas¥boards¥ck-rx65n-ryz014a¥aws_demos¥config_files¥ota_demo_config.h

ota_demo_config.h

```
 2  * FreeRTOS V202012.00
22
24  * @file ota_demo_config.h
27
28 #ifndef OTA_DEMO_CONFIG_H_
29 #define OTA_DEMO_CONFIG_H_
30
32  * @brief Certificate used for validating code signing signatures in the OTA PAL.
34 #ifndef otapalconfigCODE_SIGNING_CERTIFICATE
35     #define otapalconfigCODE_SIGNING_CERTIFICATE    \
36 "-----BEGIN CERTIFICATE-----\n"\
37 "MIICNzCCAd0CFEaV68Rv4FvfrMBmhoMyMbIM6efsMAoGCCqGSM49BAMCMIGdMQsw\n"\
38 "CQYDVQQGEwJKUDEOMAwGA1UECAwFVG9reW8xEDAOBgNVBAcMB0tvZGFpcmExEDAO\n"\
39 "BgNVBAoMB1JlbmVzYXMxGjAYBgNVBAsMEVNvZnR3YXJlIERldiBVbml0MQ8wDQYD\n"\
40 "VQQDDAZOYWtha2kxLTArBgkqhkiG9w0BCQEWHmhpcm95dWtpLm5ha2FraS5yaEBy\n"\
41 "ZW5lc2FzLmNvbTAeFw0yMTA3MDkwODA0MjlaFw0zMTA3MDcwODA0MjlaMIGdMQsw\n"\
42 "CQYDVQQGEwJKUDEOMAwGA1UECAwFVG9reW8xEDAOBgNVBAcMB0tvZGFpcmExEDAO\n"\
43 "BgNVBAoMB1JlbmVzYXMxGjAYBgNVBAsMEVNvZnR3YXJlIERldiBVbml0MQ8wDQYD\n"\
44 "VQQDDAZOYWtha2kxLTArBgkqhkiG9w0BCQEWHmhpcm95dWtpLm5ha2FraS5yaEBy\n"\
45 "ZW5lc2FzLmNvbTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABLwyr3uQuy+YLyxa\n"\
46 "HMQEvu8nKyJVwbnyTGvCpNoMIZlXjNmH07eda5fs1dyRJjosLBOp2p/zgwQMZ1QH\n"\
47 "E0x9WlowCgYIKoZIzj0EAwIDSAAwRQIgBIJo+mPelcLB6RUo1VVm8X/1+phzm6m6\n"\
48 "sNUe6nrFYEkCIQC9+8pmmRKAYH+zhdTnHEIKzmRkDDX/4RcZY9Dhh249NA==\n"\
49 "-----END CERTIFICATE-----"
50 #endif
51
53  * @brief Major version of the firmware.
58 #ifndef APP_VERSION_MAJOR
59     #define APP_VERSION_MAJOR    0
60 #endif
61
63  * @brief Minor version of the firmware.
68 #ifndef APP_VERSION_MINOR
69     #define APP_VERSION_MINOR    9
70 #endif
71
73  * @brief Build version of the firmware.
78 #ifndef APP_VERSION_BUILD
79     #define APP_VERSION_BUILD    2
80 #endif
```

Figure 6-7 Position of the APP_VERSION_BUILD macro

6.2.3.2 Execution of Sample Project

The operation contents for each case of the sample project are explained below.

Case 1 maintains an MQTT connection with AWS and logs out OTAgent telemetry every second.

In Cases 2 and 3, after OTAAgent is started, it checks with AWS for OTA job and if there is no job, the RX65N moves to software standby mode(Case 2) or deep software standby mode(Case 3). To return from the software standby mode and deep software standby mode of Case 2 and Case 3, press the USER SW. (see Figure 6-2 CK-RX65N USER SW, LED4) It is also possible to return by executing an OTA job from AWS as in 6.2.3.1.

You can see the operating status of RX65N by checking LED4. (LED4 lights up during software standby mode and deep software standby mode, and turns off during program operation)

7. Trouble Shooting

(1) Q: I measured the current, but the measurement result is higher than expected.

A1: It is possible that the current was measured with the emulator connected.
Make sure that the emulator is not connected to the board for current measurement.

A2: If you write a program to measure the current consumption in the debug mode of e2studio, the current consumption cannot be measured correctly.
Use writer software such as Renesas Flash Programmer to write a program for measuring current consumption.

A3: If the RX65N port mode register (PORTn.PMR) is set to be used as "1: Peripheral module", current may flow to the peripheral module.
Set it to be used as "0: General-purpose I / O port".

A4: In deep software standby mode, the current consumption differs depending on whether power is supplied to the standby RAM and the USM resume detecting unit. Please set DEEPCUT [1: 0] of the deep standby control register (DPSBYCR) according to your environment.

(2) Q: The connection with AWS is cut off during intermittent operation.

A1: If the RX65N is in software standby mode or deep software standby mode, set the RTS pin of RYZ014A to H to transition to standby mode.

8. Reference Documents

The documents related to this document are shown below. Please also refer to it.

- RYZ014 Module System Integration Guide (R19AN0074)
- RYZ014 Modules User's Manual: AT Command (R11UZ0093)
- RYZ014 Power Consumption Measurements on RYZ014-Based Modules (R19AN0073)
- RX65N Group, RX651 Group User’s Manual: Hardware (R01UH0590)
- RX65N Group Target Board for RX65N User’s Manual (R20UT4167)
- RX Family Cloud Option Board User’s Manual (R12UM0039)
- RX Family Troubleshooting when Using Amazon Web Services (R20AN0624)
- RX Family How to implement FreeRTOS OTA by using Amazon Web Services on RX65N(R01AN5549)
- RX65N Connecting AWS Cloud with FreeRTOS Getting Started Guide for CK-RX65N(R01QS0065)

9. Related Website and Support

AWS Amazon FreeRTOS forum: http://forums.aws.amazon.com

Renesas Amazon FreeRTOS GitHub: https://github.com/renesas/amazon-freertos

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jun.30.22	-	First edition
1.10	Aug.31.22	43 -	Added low consumption operation & OTA update sample program using CK-RX65N board.
		-	Changed some text expressions.

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

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

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

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

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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