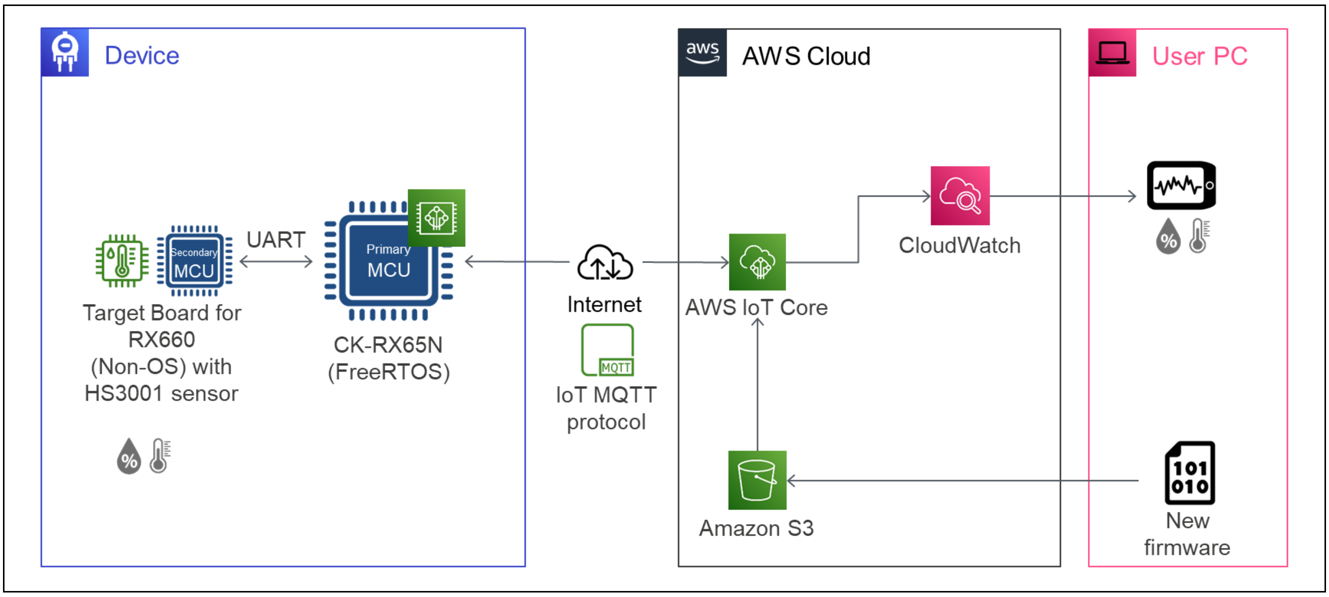
RX65N Group

Sample Code for OTA Update of a Secondary Device by Amazon Web Services with the Use of FreeRTOS

Introduction

This application note is for a system in which an RX65N microcontroller is used as a primary MCU that communicates with Amazon Web Services™ (hereafter, referred to as "AWS") and an RX microcontroller is used as a secondary MCU that receives data measured by sensors. This application note describes a demonstration where AWS services are used to perform an over-the-air (OTA) update of the secondary MCU (hereafter, referred to as "secondary OTA update").



Devices Used in Confirming Operation

- Primary MCU: RX65N
- Secondary MCU: RX660
- Temperature and humidity sensor: HS3001 high-performance relative humidity and temperature sensor

Boards Used in Confirming Operation

- Primary MCU: CK-RX65N (RTK5CK65N0S04000BE)
- Secondary MCU: Target Board for RX660 (RTK5RX6600C00000BJ)
- Temperature and humidity sensor: Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)

Related Documents

This application note refers to and further explains information in the following documents. Updating of a document may lead to changes to the structure of chapters and other items. Take care on this point when referring to the following documents.

RX Family Firmware Update Module Using Firmware Integration Technology (R01AN6850)

RX Family Firmware Update Software Development Guide using AWS/Azure QE for OTA (R20AN0712)

RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.0 or later)

RX65N Group CK-RX65N v1 User's Manual (R20UT5100)

RX660 Group Target Board for RX660 User's Manual (R20UT5068)

(Download the latest versions from the Renesas Electronics Corp. website.)

Technical Updates and Technical News

(Download the latest versions from the Renesas Electronics Corp. website.)

Amazon Web Services, the "Powered by AWS" logo, and any other AWS trademarks used in such materials are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.

FreeRTOS^{TM} and FreeRTOS.org^{TM} are trademarks of Amazon Web Services, Inc.

Pmod is a trademark of Digilent Inc.

All trademarks and registered trademarks are the property of their respective owners.

Contents

1. Overview

This demonstration involves using a secondary OTA update to add a working sensor and confirming addition of the sensor data to be acquired by the display of sensor data in an AWS display in your browser.

IoT devices require the appropriate fixing of security vulnerabilities and updating of functions in response to customer requests. Implementing the secondary OTA update to supplement OTA updating of the primary MCU that has been provided in the past enables product development that supports measures against vulnerabilities in the secondary MCU and the updating of flexible services.

2. Conditions for Confirming Operation

The sample demonstration programs for this application note have been confirmed to operate correctly under the following conditions.

Table 2-1 Conditions for Confirming Demo Operation (RX65N)

Item	Description
MCU	RX65N (R5F565NEHDFB)
Board	CK-RX65N (RTK5CK65N0S04000BE)
Operating voltage	3.3 V
RTOS	FreeRTOS v202210.01-LTS-1.1.3
Integrated development environment (IDE)	e^2 studio 2024-01 (24.1.0) QE for OTA v2.00
C compiler	C/C++ Compiler Package for RX Family [CC-RX] v3.06.00 made by Renesas Electronics Corporation
Flash memory programming tool	Renesas Flash Programmer V3.14.00

Table 2-2 Conditions for Confirming Demo Operation (RX660)

Item	Description
MCU	RX660 (R5F56609BDFP)
Board	Target Board for RX660 (RTK5RX6600C00000BJ)
Operating voltage	3.3 V
Integrated development environment (IDE)	e^2 studio 2024-01 (24.1.0)
C compiler	C/C++ Compiler Package for RX Family [CC-RX] v3.06.00 made by Renesas Electronics Corporation GCC for Renesas RX 8.3.0.202202
Flash memory programming tool	Renesas Flash Programmer V3.14.00
MOT file conversion tool	Renesas Image Generator (included in the RX660 project)
USB-UART converter	Pmod USBUART™

Table 2-3 Condition for Confirming Demo Operation (Sensor)

Item	Description
Temperature and humidity sensor board	US082-HS3001EVZ board

Table 2-4 Condition for Confirming Demo Operation (Others)

Item	Version
Python	3.10.4

QE for OTA is available at https://www.renesas.com/qe-ota/.

Python is available at https://www.python.org/.

3. Description of Hardware

The system consists of an RX65N microcontroller (primary MCU) that provides functionality for controlling communications with AWS and an RX660 microcontroller (secondary MCU) connected to the HS3001 sensor. The two microcontrollers communicate with each other via UARTs.

The system configuration is shown in Figure 3-1.

The CK-RX65N equipped with an RX65N microcontroller is used as the primary MCU.

The Target Board for RX660 (hereafter referred to as “TB-RX660”) equipped with an RX660 microcontroller is used as the secondary MCU.

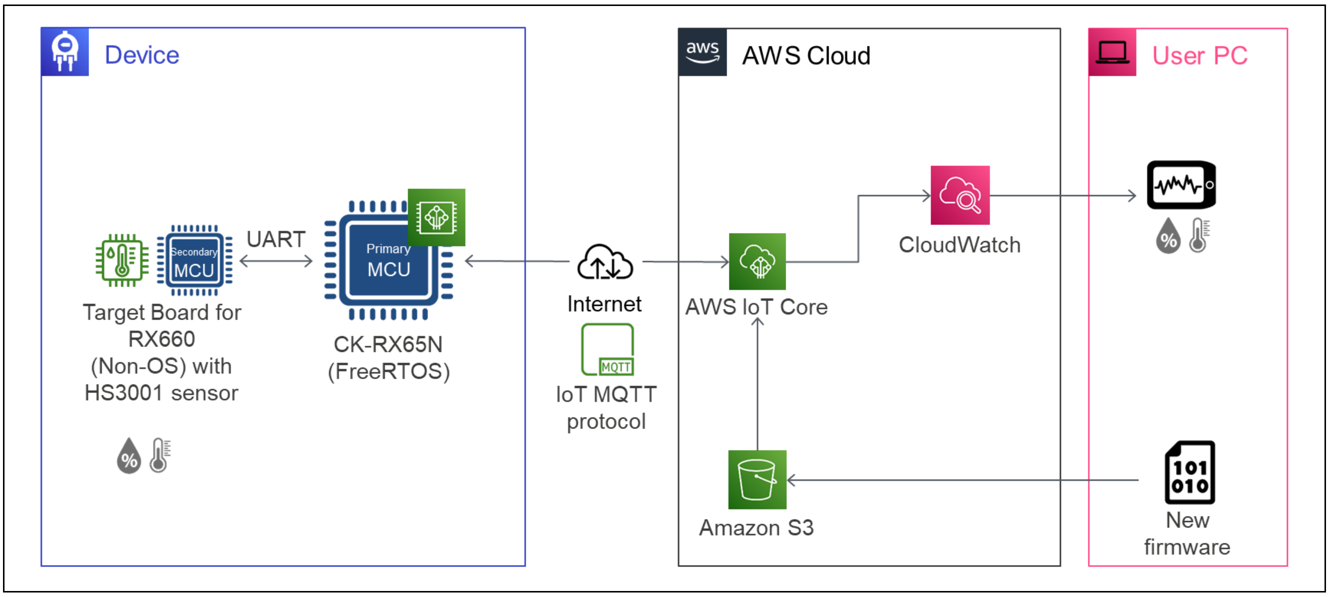


Figure 3-1 System Configuration of This Demo

4. Description of Software

“FreeRTOS™ with IoT Library” is implemented in the RX65N firmware, which utilizes AWS-certified programs. This allows the use of AWS IoT Core and AWS IoT Device Management, which are managed services provided by AWS, to perform OTA firmware updating and data uploading to the cloud via MQTT communications.

The RX65N microcontroller on the primary MCU side uses the AWS IoT Over-the-air Update Library to control OTA updating of the secondary MCU. The update firmware for the secondary MCU, which is received from AWS, is transferred to the secondary MCU, where the firmware update is applied.

The RX microcontroller on the secondary MCU side uses “RX Family Firmware Update Module Using Firmware Integration Technology Rev.2.01” to control firmware updating of the secondary MCU.

4.1 Firmware Update Methods

The mechanism for firmware update in the secondary MCU of this sample program uses “linear mode partial update method” among the methods provided by the firmware update module. For details on this method, refer to “linear mode partial update method” in “1.3 Firmware Update Operation” in “RX Family Firmware Update Module Using Firmware Integration Technology Rev.2.01”.

The memory map of this sample program is shown in the following.

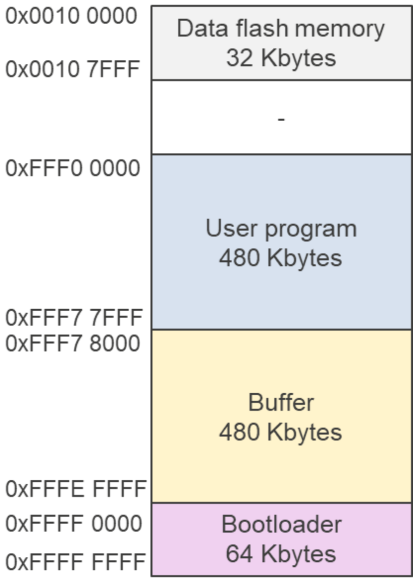


Figure 4-1 Memory Map of the TB-RX660 Sample Program

Operations for the secondary OTA update are summarized in Figure 4-2. The states of the ROM in each phase during updating are shown in Figure 4-3. Note that the red frames in Figure 4-3 indicate the programs under execution at the given times.

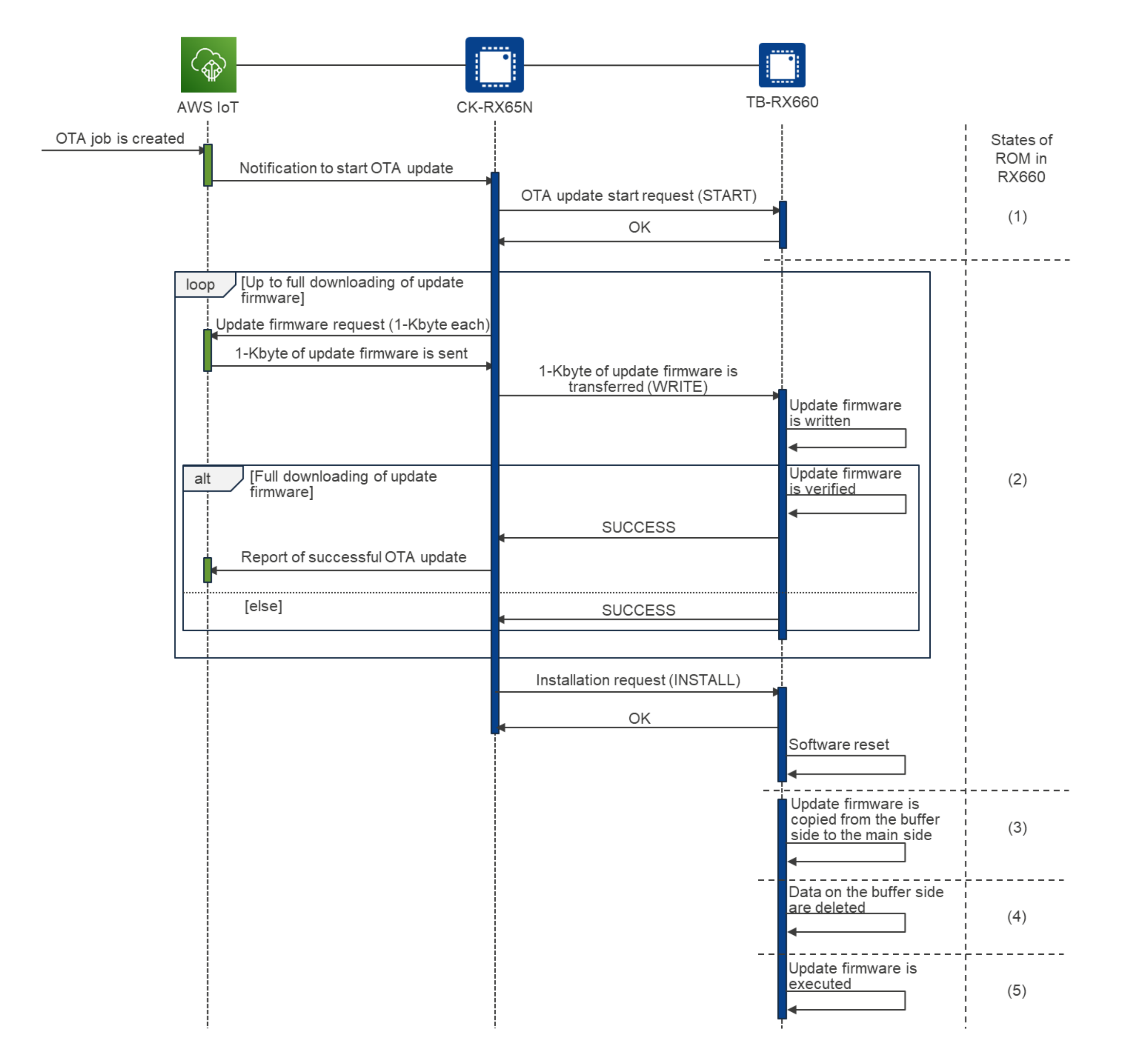
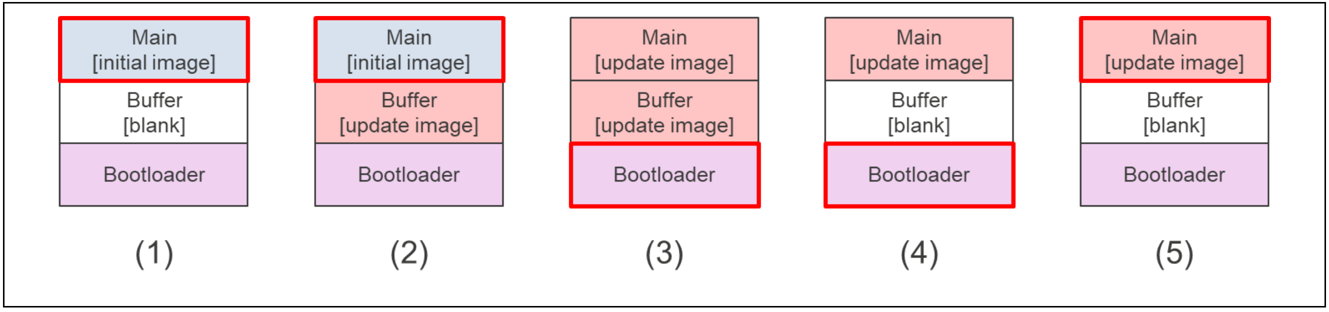

AWS IoT
CK-RX65N
TB-RX660
OTA job is created
Notification to start OTA update
OTA update start request (START)
OK
States of ROM in RX660
(1)
loop
[Up to full downloading of update firmware]
Update firmware request (1-Kbyte each)
1-Kbyte of update firmware is sent
1-Kbyte of update firmware is transferred (WRITE)
Update firmware is written
alt
[Full downloading of update firmware]
Update firmware is verified
(2)
SUCCESS
Report of successful OTA update
[else]
SUCCESS
Installation request (INSTALL)
OK
Software reset
Update firmware is copied from the buffer side to the main side
(3)
Data on the buffer side are deleted
(4)
Update firmware is executed
(5)

Figure 4-2 Overview of Operations for the Secondary OTA Update



(1)	(2)	(3)	(4)	(5)
Main [initial image]	Main [initial image]	Main [update image]	Main [update image]	Main [update image]
Buffer [blank]	Buffer [update image]	Buffer [update image]	Buffer [blank]	Buffer [blank]
Bootloader	Bootloader	Bootloader	Bootloader	Bootloader

Figure 4-3 States of ROM in the Secondary MCU during Updating

4.2 UART Communications between the Microcontrollers

The primary MCU transmits a command to the secondary MCU via UART communications, and the secondary MCU executes the processing directed by the received command and returns the result as a response.

4.2.1 Settings of UART Communications

The UART communications settings are shown in Table 4-1.

Table 4-1 Settings of UART Communications between the Microcontrollers

Item	Setting
Baud rate	1 Mbps or less (speed used in confirming operation)
Data length	8 bits
Parity bit	None
Stop bit	1 bit
Flow control	None

4.2.2 Data Format

The specifications for packet communications of commands or responses between the primary MCU and secondary MCU are described here.

4.2.2.1 Data Format of a Whole Packet

Figure 4-4 shows the data format of a whole packet. A packet consists of a header and a payload.

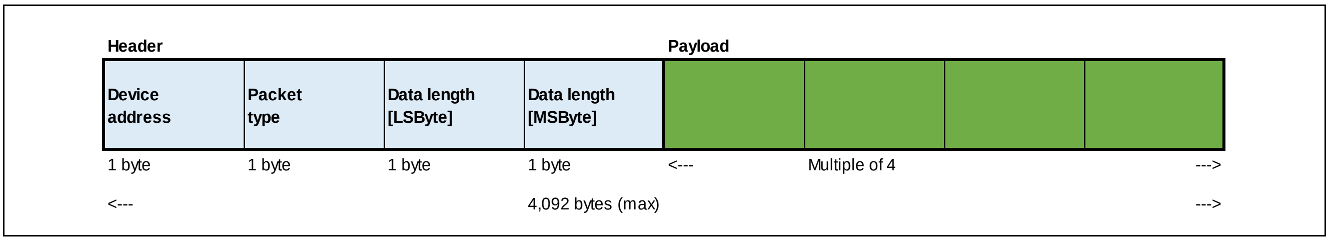


Figure 4-4 Data Format of a Whole Packet

The header specifications for a packet are listed in Table 4-2.

Table 4-2 Header Specifications for a Packet

Item	Description
Device address	Device address Each device refers to the device address to determine whether the address is its own. When the secondary MCU determines that a packet is addressed to itself, it starts detailed analysis of the packet. • 0x00: Device address of the primary MCU • 0x01 to 0xFE: Device address of the secondary MCU • 0xFF: Reserved
Packet type	Packet attribute • b7 and b6: Fixed to 00b • b5: 0: Request, 1: Response • b4: [Request] Fixed to 0 [Response] 0: ACK, 1: NACK* • b3 to b0: Command type 0000b: Common command 0001b: Reserved 0010b: FWUP command 0011b: SENSOR command 0100b to 1111b: Reserved
Data length	Data size of the payload

Note: ACK or NACK indicates whether the data format of the packet was normal or abnormal, respectively.

4.2.2.2 Data Format of a Payload

Figure 4-5 shows the data format of a payload. A payload consists of a command header and command data.

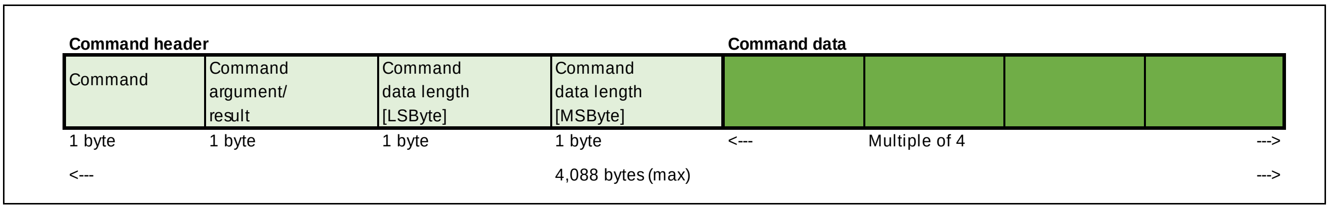


Figure 4-5 Data Format of a Payload

The specifications for a command header are listed in Table 4-3.

Table 4-3 Specifications for a Command Header

Item	Description
Command	Command For details, refer to “0 Command Specifications”.
Command argument/result	In the case of a request, this is an argument for the command. • b7 to b4: Fixed to 0000b • b3: Reserved as 0b. • b2: Reserved as 0b. • b1 and b0: Argument 00: Secondary MCU 0 01: Primary MCU 0 10: Primary MCU 1 11: Primary MCU 2
	In the case of a response, this is the result of executing the command. • b7 to b4: 1111: Failure of command execution 1110: Invalid command header (non-supported command) … 0001: Successful completion of command execution 1 0000: Successful completion of command execution • b3: Reserved as 0b. Same as for a request. • b2: Reserved as 0b. Same as for a request. • b1 and b0: Argument. Same as for a request.
Command data length	Data size of the command data. This should be set to a multiple of 4.

4.2.3 Command Specifications

4.2.3.1 FWUP Commands

These commands are used at the time of updating firmware. Table 4-4 shows a list of the FWUP commands.

Table 4-4 List of FWUP Commands

Command Type	Command	Description
FWUP	START: Firmware update start command	Starts updating of the firmware.
	WRITE: Firmware write command	Writes the update firmware.
	INSTALL: Firmware install command	Installs and executes the update firmware.
	CANCEL: Firmware update cancel command	Aborts updating of the firmware.

(1) START: Firmware update start command

This command requests starting of the firmware update. Target ID can be set to a desired value. This command can be used in the identification code, etc. on the user side.

The secondary MCU sets the result in the command argument/result field and returns the response.

Place the secondary MCU in a state in which the update firmware can be written by using the firmware update module.

Table 4-5 Data Format of the FWUP/START Command

Command	Packet	Device Address	Packet Type	Data Length		Command	Command Argument/ Result	Command Data Length		Command Data			
				LSB	MSB			LSB	MSB	0	1	2	3
FWUP/ START	Request	0xXX	0x02	0x08	0x00	0x02	0x00	0x04	0x00	Target ID [little endian]			
	ACK response		0x22	0x04			*	0x00					

Note: The possible result values for a response are listed below.
0x00: Successful operation
0xFF: Failed operation

(2) WRITE: Firmware write command

This command transmits an offset address based on the start address of binary data having an offset value of 0 and its firmware data, and requests writing of the firmware data.

The secondary MCU executes the write processing. Signing verification will be executed at the end of the last block. The secondary MCU sets the result in the command argument/result field and returns the response.

Table 4-6 Data Format of the FWUP/WRITE Command

Command	Packet	Device Address	Packet Type	Data Length		Command	Command Argument/ Result	Command Data Length		Command Data			
				LSB	MSB			LSB	MSB	0	1	2	3
FWUP/ WRITE	Request	0xXX	0x02	8 + (length of write data)		0x04	0x00	4 + (length of write data)		Offset address [little endian]: 4 bytes Write data [big endian]: N bytes*2			
	ACK response		0x22	0x04	0x00		*1	0x00	0x00				

Notes 1. The possible result values in the case of a response are listed below.
0x00: Successful operation
0xFF: Failed operation
2. Data are written in the same order as the transmitted binary data.

(3) INSTALL: Firmware install command

This command requests the installation and execution of the written update firmware.

The secondary MCU sets the result in the command argument/result field and returns the response, then executes the update firmware.

Table 4-7 Data Format of the FWUP/INSTALL Command

Command	Packet	Device Address	Packet Type	Data Length		Command	Command Argument/ Result	Command Data Length	
				LSB	MSB			LSB	MSB
FWUP/ INSTALL	Request	0xXX	0x02	0x04	0x00	0x06	0x00	0x00	0x00
	ACK response		0x22				*		

Note: The possible result values in the case of a response are listed below.
0x00: Successful operation
0xFF: Failed operation

(4) CANCEL: Firmware update cancel command

This command requests that updating of the firmware be aborted.

The secondary MCU aborts updating and erases the written update firmware. The secondary MCU sets the result in the command argument/result field and returns the response.

Table 4-8 Data Format of the FWUP/CANCEL Command

Command	Packet	Device Address	Packet Type	Data Length		Command	Command Argument/ Result	Command Data Length	
				LSB	MSB			LSB	MSB
FWUP/ CANCEL	Request	0xXX	0x02	0x04	0x00	0xF0	0x00	0x00	0x00
	ACK response		0x22				*		

Note: The possible result values in the case of a response are listed below.
0x00: Successful operation
0xFF: Failed operation

4.2.3.2 SENSOR Command

This command is used for controlling a sensor. Table 4-9 shows a list of the SENSOR commands.

Table 4-9 List of SENSOR Commands

Command Type	Command	Description
SENSOR	GET_HS300X: HS300x measured data acquisition command	Acquires the data measured by the HS300x sensor.

(1) GET_HS300X: HS300x measured data acquisition command

This command requests transmission of the data measured by the HS300x sensor, which is connected to the secondary MCU.

The secondary MCU sets the result in the command argument/result field and returns the response.

The primary MCU refers to the command argument/result field and confirms the validity of the data. If the data are stale, the former state of the data is indicated.

Table 4-10 Data Format of the SENSOR/GET_HS300X Command

Command	Packet	Device Address	Packet Type	Data Length		Command	Command Argument/ Result	Command Data Length		Command Data			
				LSB	MSB			LSB	MSB	0	1	2	3
SENSOR/ GET_HS300X	Request	0xXX	0x03	0x04	0x00	0x01	0x00	0x00	0x00				
	ACK response		0x23	0x0C			*	0x08		Humidity [little endian]: 4 bytes [float type] Temperature [little endian]: 4 bytes [float type]			

Note: The possible result values in the case of a response are listed below.
0x00: Valid data
0x01: Stale data
0xFX: Failed operation

4.3 Folder/File Structure

Figure 4-6 shows the folder/file structure.

```
r01an6220xx0200-rx-2nd-ota-apl
├─Demo
│    ├─ccrx
│    │    ├─afr-v202210.01-LTS-rx-1.1.3
│    │    ├─ck_rx65n_2ndota_demo
│    │    ├─ck_rx65n_demo_bootloader
│    │    ├─rx660_tb_2ndota_demo
│    │    └─rx660_tb_demo_bootloader
│    └─gcc
│         ├─rx660_tb_2ndota_demo
│         └─rx660_tb_demo_bootloader
├─r01an6220ej0200-rx-2nd-ota-apl.pdf
└─r01an6220jj0200-rx-2nd-ota-apl.pdf
```

Figure 4-6 Folder/File Structure

The ck_rx65n_2ndota_demo folder and ck_rx65n_demo_bootloader folder contain project files for the CK-RX65N.

The rx660_tb_2ndota_demo folder and rx660_tb_demo_bootloader folder contain project files for the TB-RX660.

The projects for the CK-RX65N only support the CC-RX compiler and the projects for the TB-RX660 support the CC-RX and GCC compilers.

4.4 Code Size

The ROM and RAM sizes for projects included in the sample code of this application note are listed in the tables below. The values in the tables were confirmed under the following conditions.

- CC-RX
  - Compiler
    Optimization level (-optimize): Level 2: Performs whole module optimization
    Optimization type (-speed/-size): Optimizes with emphasis on code size
  - Linker
    Optimization type (-nooptimize/-optimize): All
  - Library Generator
    Optimization level (-optimize): Level 2: Performs whole module optimization
    Optimization type (-speed/-size): Optimizes with emphasis on code size

Table 4-11 Code Size (CC-RX)

Project	ROM	RAM
ck_rx65n_demo_bootloader	33 Kbytes	9 Kbytes
ck_rx65n_2ndota_demo	368 Kbytes	345 Kbytes
rx660_tb_demo_bootloader	27 Kbytes	13 Kbytes
rx660_tb_2ndota_demo	49 Kbytes	22 Kbytes

- GCC
  Optimization level: Optimize for debug (-Og)

Table 4-12 Code Size (GCC)

Project	ROM	RAM
rx660_tb_demo_bootloader	60 Kbytes	17 Kbytes
rx660_tb_2ndota_demo	90 Kbytes	41 Kbytes

5. Operations of the Demonstration

(1) In the initial state of the demonstration, the TB-RX660 only acquires humidity data by using the HS3001 sensor.

(2) The secondary OTA update mechanism is used to download the update firmware for the TB-RX660 from AWS via the CK-RX65N and then update the firmware.

(3) After the firmware updating is complete, the TB-RX660 acquires temperature data in addition to humidity data from the HS3001 sensor.

In this sequence, the type of sensor data from which data are being acquired and the values can be checked from the log output from both microcontrollers to your PC and from the dashboard on AWS.

6. Setting up the Demonstration

This section describes the setting up required to run the demonstration covered by this application note.

The necessary steps are setting up the hardware, such as the wiring of the CK-RX65N and TB-RX660 and connection of the HS3001 sensor, setting up the software, such as creating and writing the initial firmware for each microcontroller board, and the preparation on the AWS cloud side for execution of the OTA update and display of the sensor data from AWS.

6.1 Setting up the Hardware

Firstly, the overall hardware structure for this demonstration is shown below. For an actual image after setup is complete, see Figure 6-12. The methods for setting up each of the boards are described in detail in the subsequent subsections.

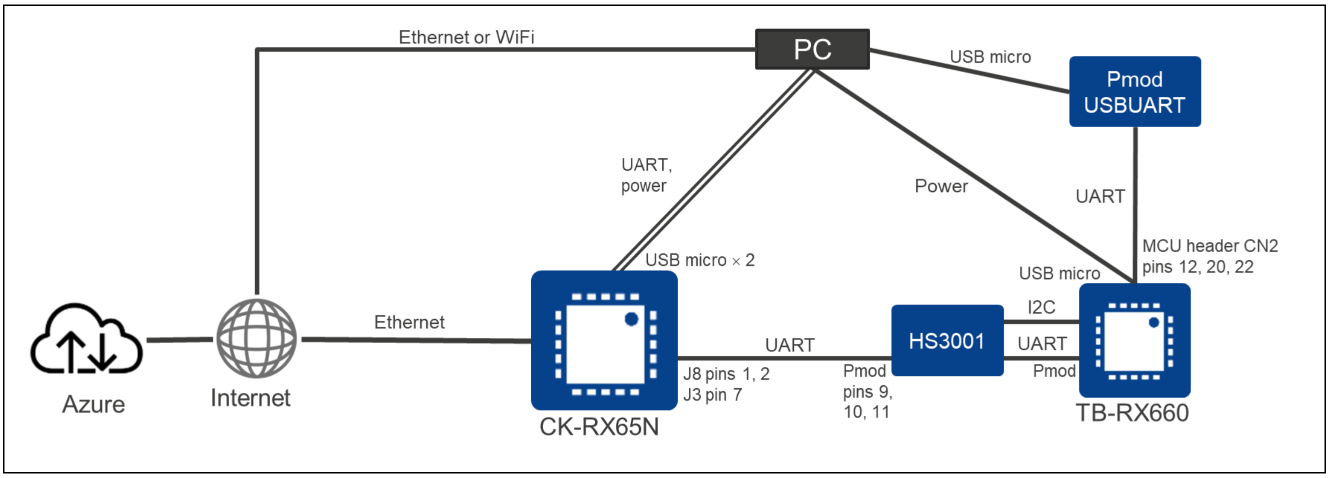


Figure 6-1 Overall Hardware Structure for This Demo

6.1.1 Setting up the CK-RX65N

The procedure for setting up the CK-RX65N is described in the following passages.

(1) Connecting the cable for UART communications with the TB-RX660

TXD, RXD, and GND for UART communications with the TB-RX660 are allocated to the following pins on the J8 and J3 connectors of the CK-RX65N. Connect the pins on the TB-RX660 side as described in 6.1.2(3), with the corresponding UART signals listed in the table below.

Table 6-1 UART Connection Method between the Microcontrollers (CK-RX65N ↔ HS3001 Sensor Board ↔ TB-RX660)

CK-RX65N	*1	HS3001 Sensor Board Pmod I/F		TB-RX660 Pmod I/F
J8 Pin 1: D0/RX (RXD7)	↔	Pin 9: TXD*	↔	Pin 9: TXD9
J8 Pin 2: D1/TX (TXD7)	↔	Pin 10: RXD*	↔	Pin 10: RXD9
J3 Pin 7: GND	↔	Pin 11: GND	↔	Pin 11: GND

Note: Since the two Pmod pins on both sides of the HS3001 sensor board are directly connected to the corresponding pins on the other boards, the I/O signals of the TB-RX660 Pmod interface can be sent to the CK-RX65N via the HS3001 sensor board.

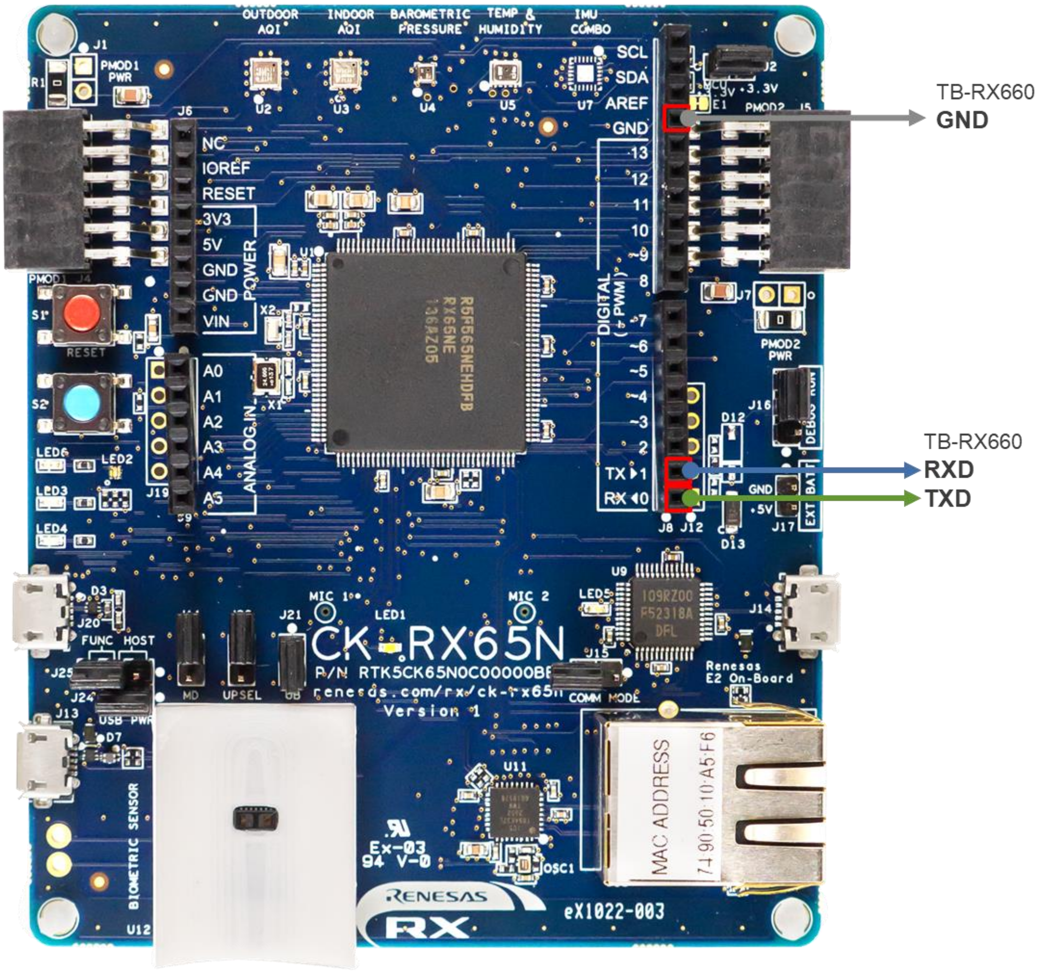


Figure 6-2 Locations of Pins on the CK-RX65N Used for UART Communications between the Microcontrollers

(2) Connecting the cable for log output to the PC

Connect the PC to the USB serial connector (micro USB Type-B) on the CK-RX65N with a USB cable.

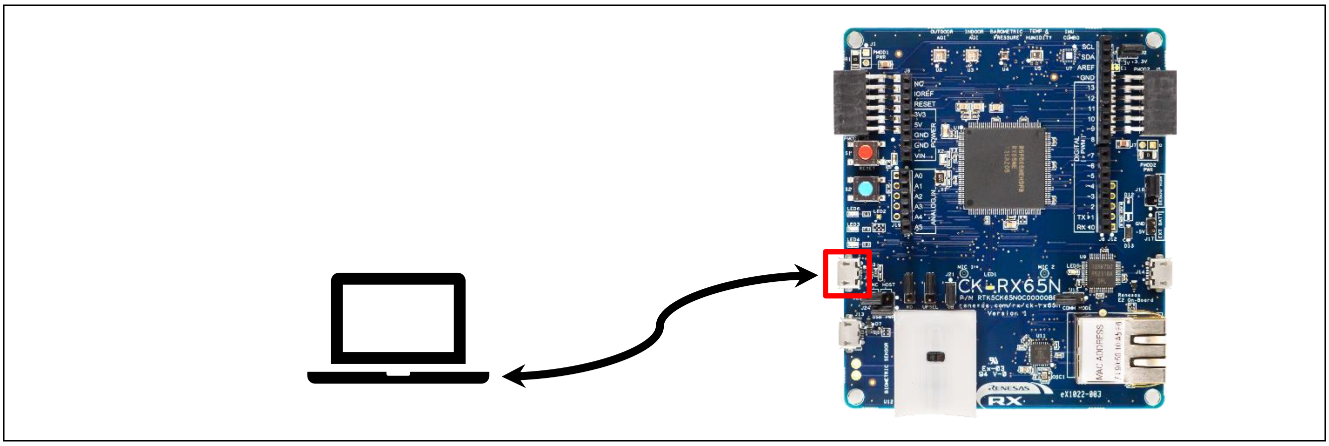


Figure 6-3 USB Connection for Log Output to the PC

(3) Connecting the power supply and debugger

Connect the PC to the E2OB Debugger connector (micro USB Type-B) on the CK-RX65N with a USB cable.

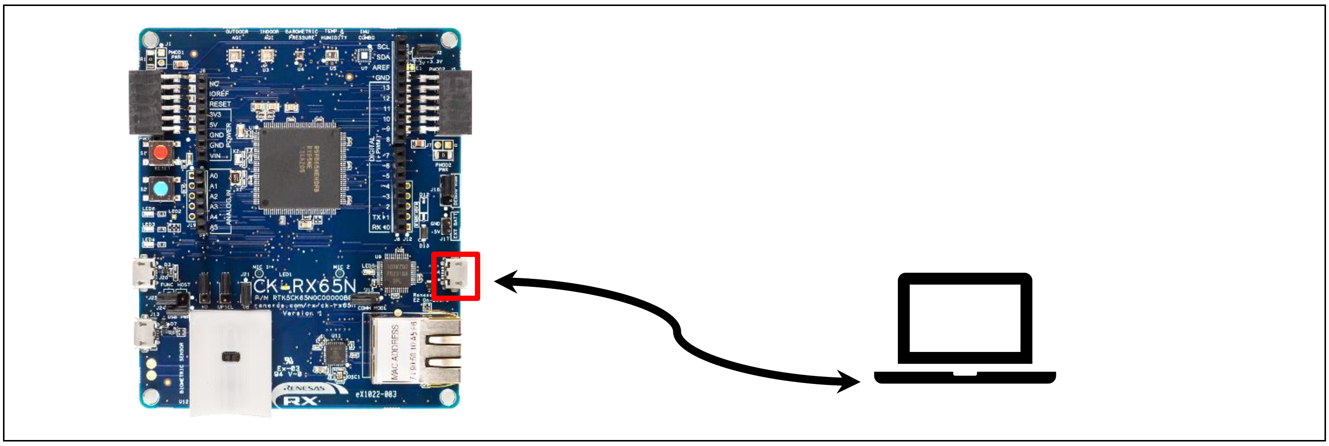


Figure 6-4 USB Connection of the Power Supply and Emulator

(4) Connecting the LAN cable for Internet connection

Connect the LAN cable connected with the Internet to the Ethernet connector on the CK-RX65N.

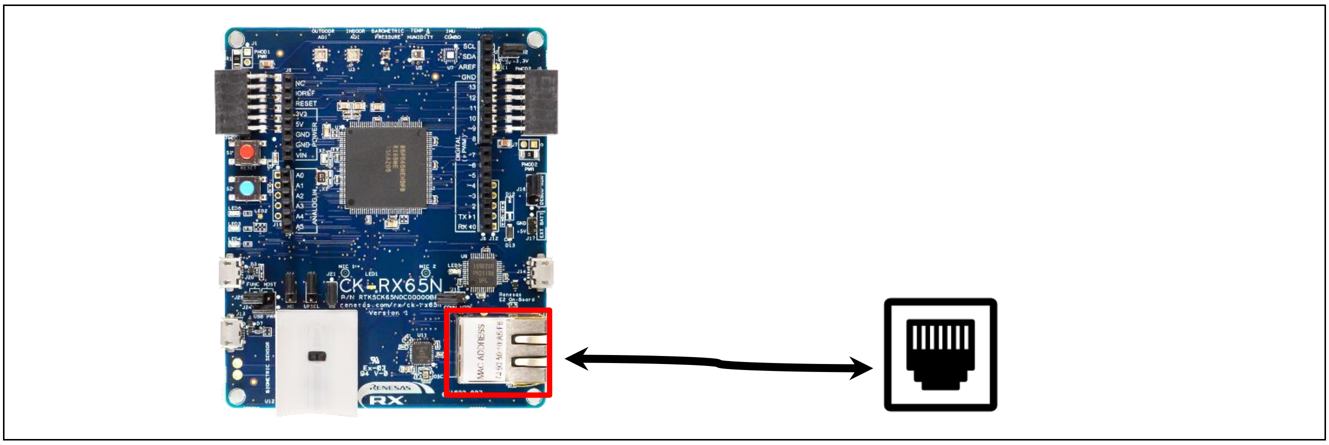


Figure 6-5 Wired Internet Connection with the Ethernet

(5) Closing jumper block J16 on the DEBUG side

To set the CK-RX65N to debug mode, close jumper block J16 on the DEBUG side (pins 1-2).

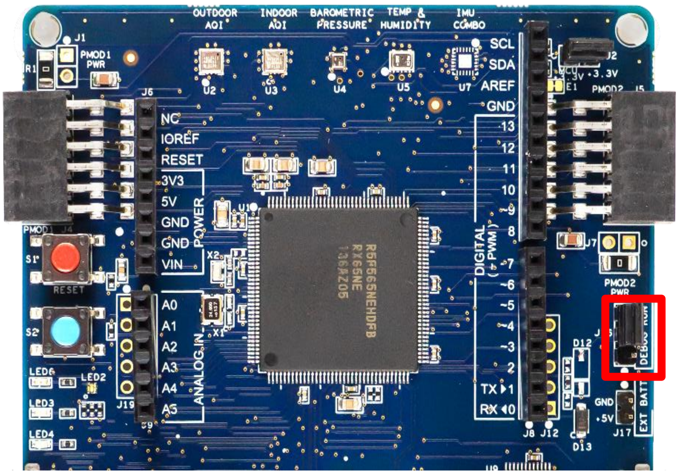


Figure 6-6 Location of Jumper Block J16

6.1.2 Setting up the TB-RX660

The procedure for setting up the TB-RX660 is described in the following passages.

(1) Advance preparation

Since headers are not inserted in the through holes on the board as shipped, make the following preparations in advance.

- Refer to “5.13 Emulator Reset Header” in the TB-RX660 User's Manual and mount the pin header.
- Refer to “5.14 Power-Supply Selection Header” in the TB-RX660 User's Manual and take the necessary steps to supply a voltage of 3.3 V. In this demonstration, the operating voltage for the RX660 is 3.3 V.
- Mount a CN2 connector on the TB-RX660.

(2) Connecting the HS3001 board

Connect the HS3001 board to the Pmod connector on the TB-RX660.

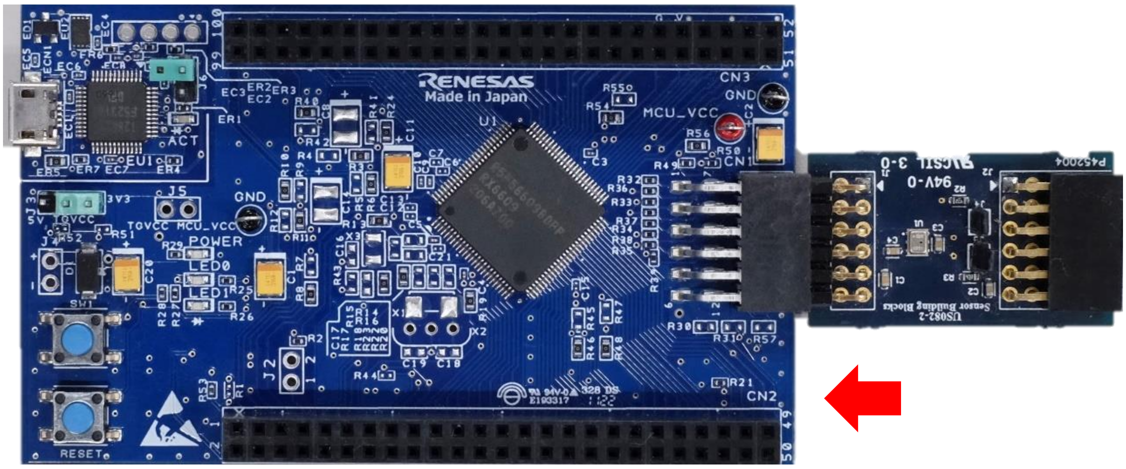


Figure 6-7 Connecting the Sensor Board to the Pmod Connector on the TB-RX660

(3) Connecting the cable for UART communications with the CK-RX65N

TXD, RXD, and GND for UART communications with the CK-RX65N are allocated to the following pins on the Pmod connector of the TB-RX660. Connect the pins on the CK-RX65N as described in 6.1.1(1), with the corresponding UART signals listed in Table 6-1.

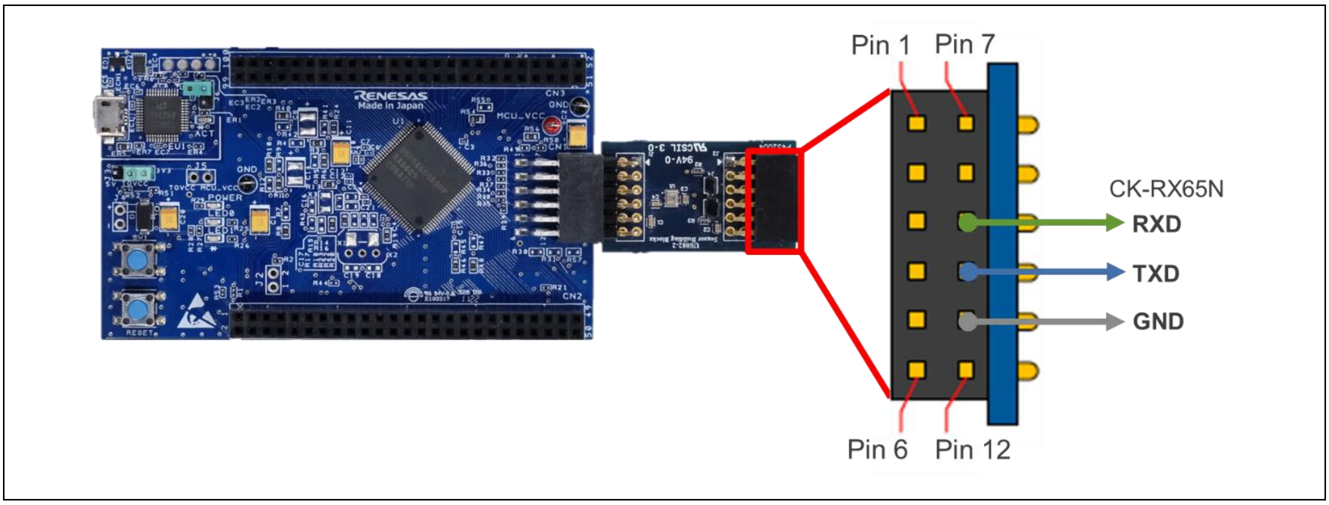


Figure 6-8 Locations of Pins on the TB-RX660 Used for UART Communications between the Microcontrollers

(4) Connecting the cable for serial communications to be used in log output to the PC

TXD, RXD, and GND for log output to the PC through a serial connection are allocated to the following pins on the MCU header CN2 of the TB-RX660. Connect the pins on the Pmod USB-to-UART converter as shown in the following table and figure, with the corresponding UART signals listed in the table below.

Close jumper block JP1 of the Pmod USB-to-UART converter on the VCC-LCL side. This makes the UART interface voltage become 3.3 V due to the power supply being from the micro-USB side.

Also, connect the PC and the micro USB Type-B connector on the Pmod USB-to-UART converter with a USB cable.

Table 6-2 Connection Method between the TB-RX660 and Pmod USB-to-UART Converter

TB-RX660 MCU Header CN2		Pmod USB-to-UART Converter Pmod I/F
Pin 12: GND	↔	Pin 5: GND
Pin 20: RXD1 (MCU P30)	↔	Pin 3: TXD
Pin 22: TXD2 (MCU P26)	↔	Pin 2: RXD

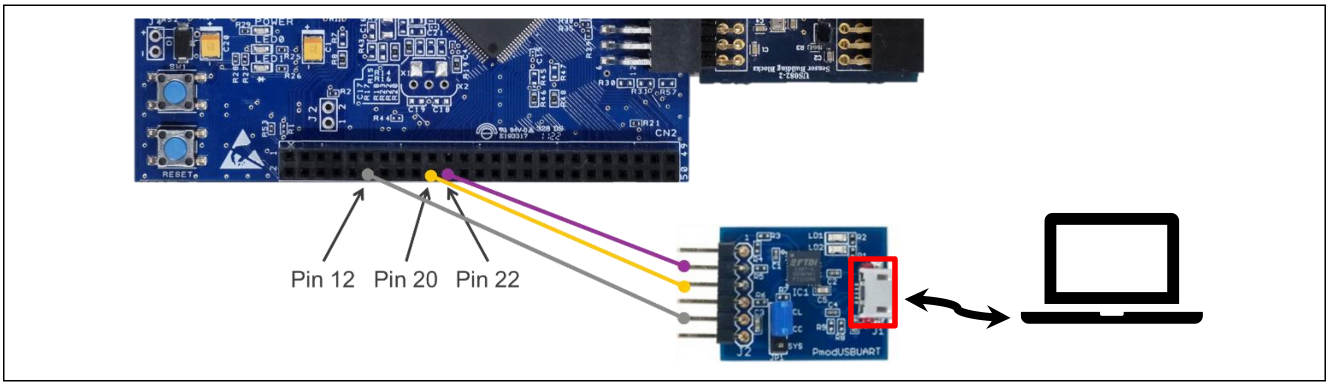


Figure 6-9 Connecting TB-RX660 with the Serial Communications Cable for Log Output to the PC

(5) Connecting the cable for power supply

Connect the PC and the micro USB Type-B connector on the TB-RX660 with a USB cable.

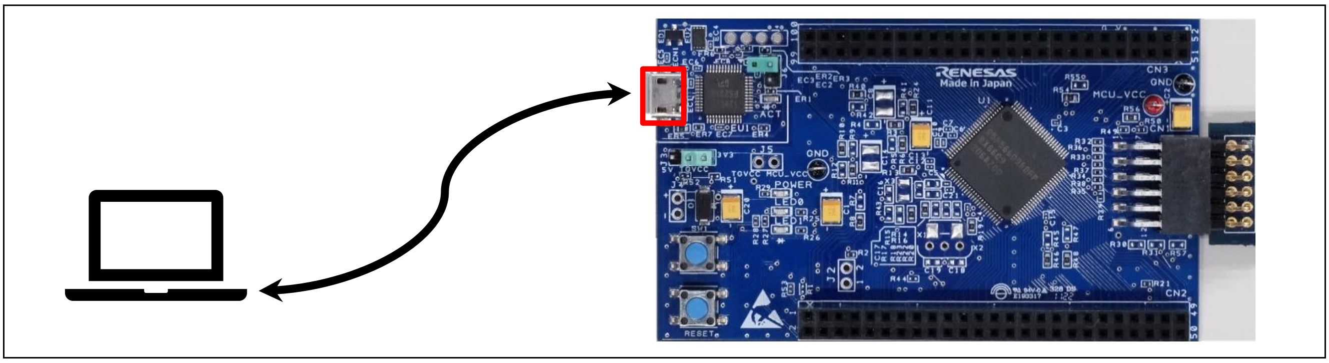


Figure 6-10 Connecting the USB Cable for the Power Supply and Emulator

(6) Opening the emulator reset header (J6)

Open the emulator reset header (J6) on the TB-RX660.

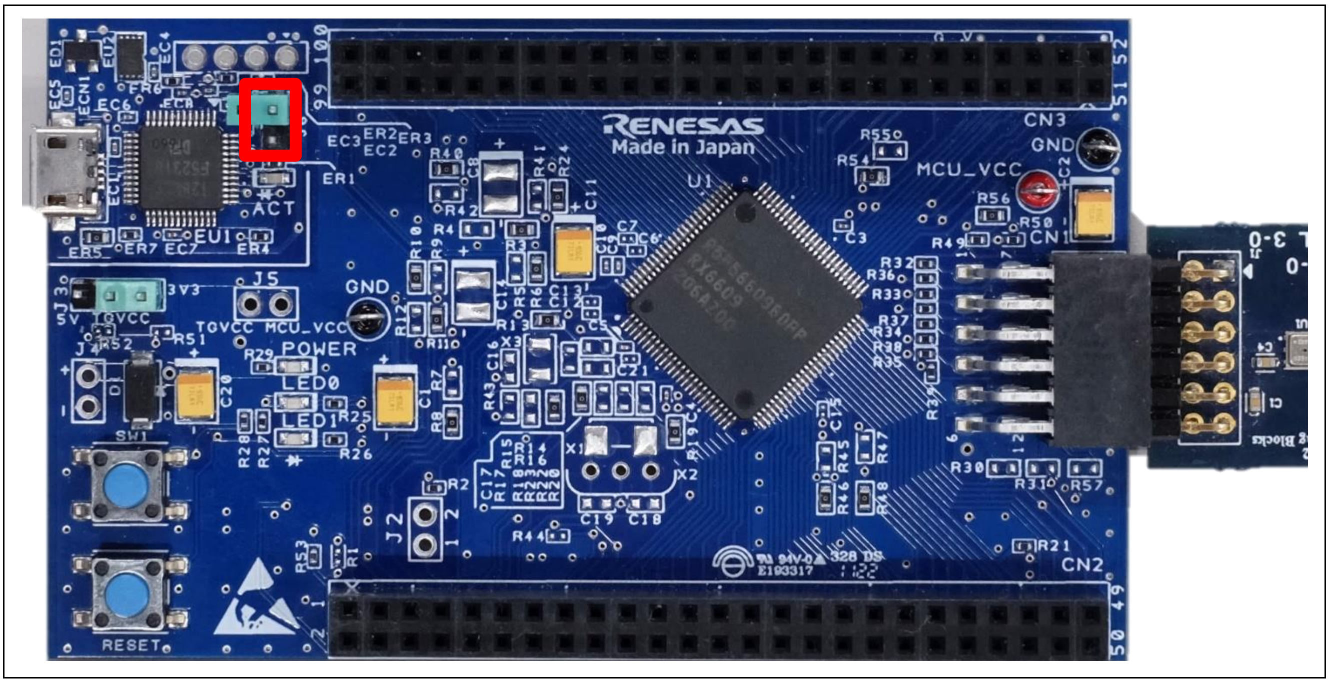


Figure 6-11 Location of the Emulator Reset Header (J6)

The hardware setup for the demonstration is now completed. Figure 6-12 is an image of the overall configuration for the demonstration.

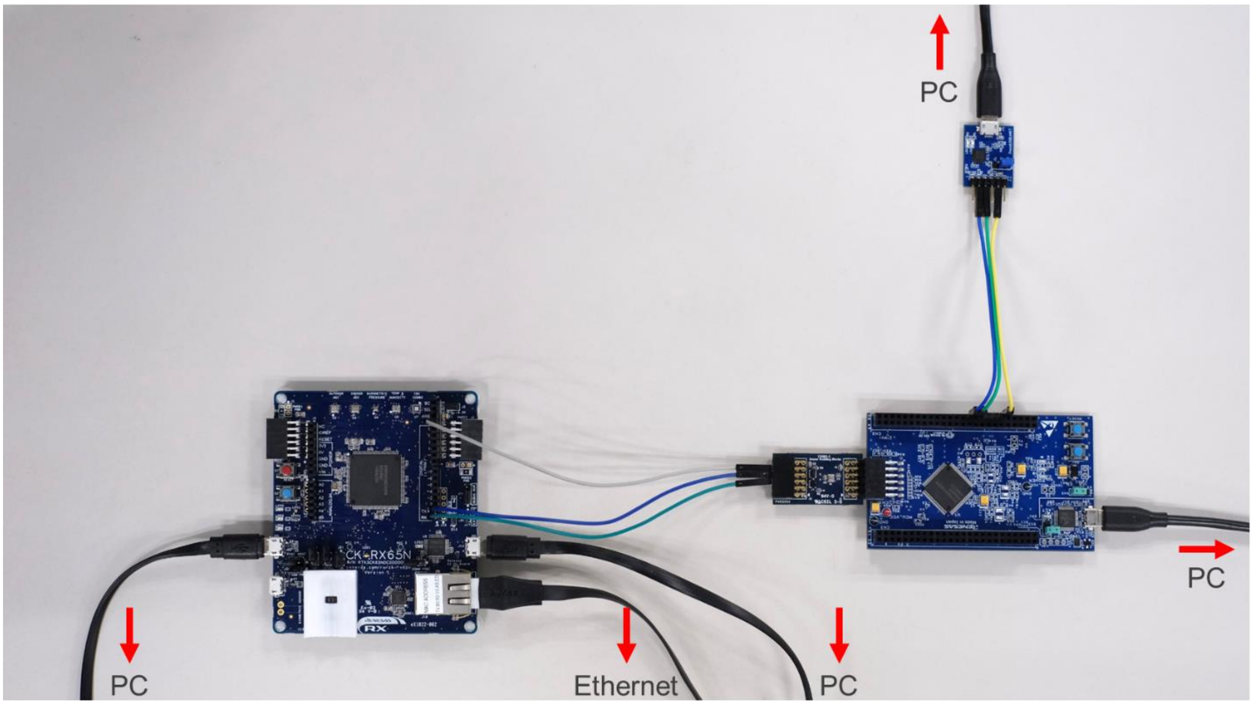


Figure 6-12 Image of the Overall Configuration for the Demo

6.2 Setting up the Software

6.2.1 Advance Preparation

For each of the software versions used in confirming operation, see Table 2-4.

(1) Installing QE for OTA

From the e^2 studio menu bar, select [Renesas Views] → [Renesas QE] to check whether QE for OTA is installed. If [OTA Main (QE)] and [OTA Manage IoT Device (QE)] are displayed, installation is completed.

If they are not displayed, refer to “2.1 Install QE for OTA” in “RX Family Firmware Update Software Development Guide using AWS/Azure QE for OTA (R20AN0712)” and install QE for OTA.

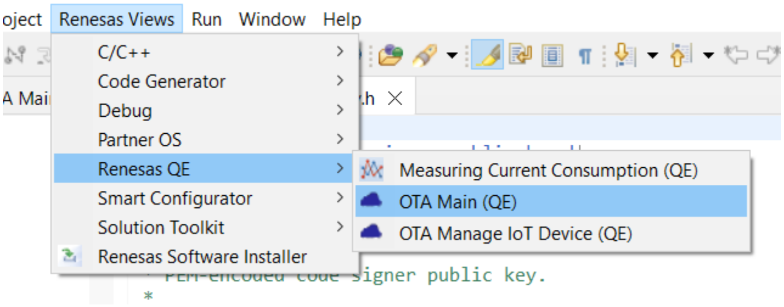


Figure 6-13 Confirming the Installation of QE for OTA

(2) Installing the Python execution environment

Python can be downloaded from https://www.python.org/.

The PyCryptodome library of Python is also to be used. After installing Python, execute the follow pip command to install it.

```
> pip install pycryptodome
```

(3) Installing the Renesas Flash Programmer

The Renesas Flash Programmer can be downloaded from Renesas Flash Programmer (Programming GUI) | Renesas.

6.2.2 Setting the Terminal Software

The terminal software (e.g., Tera Term) is required to generate log output using serial communication. The serial port settings are shown in the following.

Table 6-3 Serial Port Settings

Item	Setting
Baud rate	115,200 bps
Data length	8 bits
Parity bit	None
Stop bit	1 bit
Flow control	None

6.2.3 Creating and Running the Initial Firmware for the CK-RX65N

Create initial firmware for the CK-RX65N by using QE for OTA and execute debugging in e^2 studio. The procedure is described below.

(1) Importing projects

Import the “ck_rx65n_demo_bootloader” project, a bootloader for the CK-RX65N, and the “ck_rx65n_2ndota_demo” project, a user program, into e^2 studio.

When importing projects, uncheck [Copy projects into workspace] in the [Options] field.

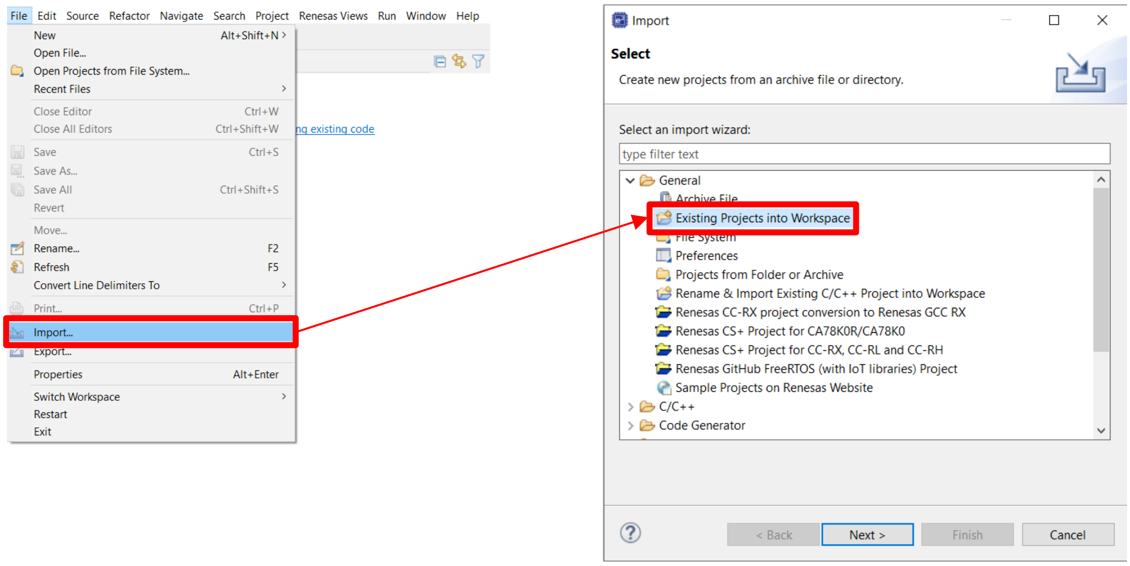
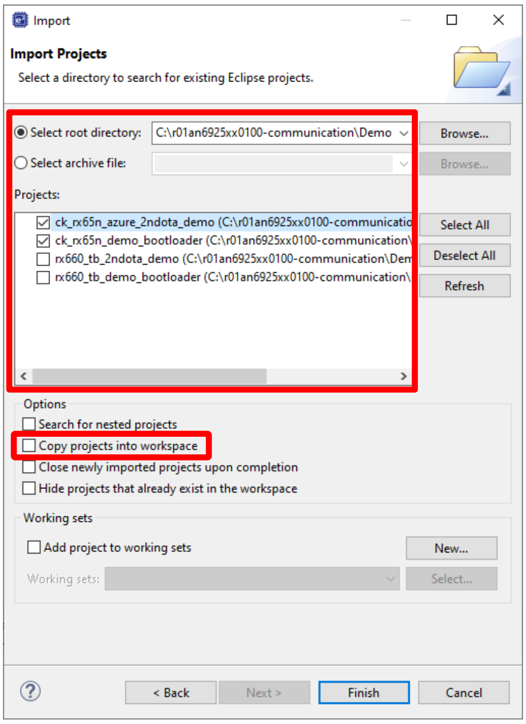

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
ng existing code
Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
General
Archive File
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
Rename & Import Existing C/C++ Project into Workspace
Renesas CC-RX project conversion to Renesas GCC RX
Renesas CS+ Project for CA78K0R/CA78K0
Renesas CS+ Project for CC-RX, CC-RL and CC-RH
Renesas GitHub FreeRTOS (with IoT libraries) Project
Sample Projects on Renesas Website
C/C++
Code Generator
< Back
Next >
Finish
Cancel

Figure 6-14 Importing Projects (1)


Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\r01an6925xx0100-communication\Demo

○ Select archive file:

Projects:

☑ ck_rx65n_azure_2ndota_demo (C:\r01an6925xx0100-communicatio
☑ ck_rx65n_demo_bootloader (C:\r01an6925xx0100-communication\
☐ rx660_tb_2ndota_demo (C:\r01an6925xx0100-communication\Dem
☐ rx660_tb_demo_bootloader (C:\r01an6925xx0100-communication\

Options

☐ Search for nested projects
☐ Copy projects into workspace
☐ Close newly imported projects upon completion
☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

Figure 6-15 Importing Projects (2)

(2) Opening the QE for OTA window

From the e² studio menu bar, select [Renesas Views] → [Renesas QE] → [OTA Main (QE)].

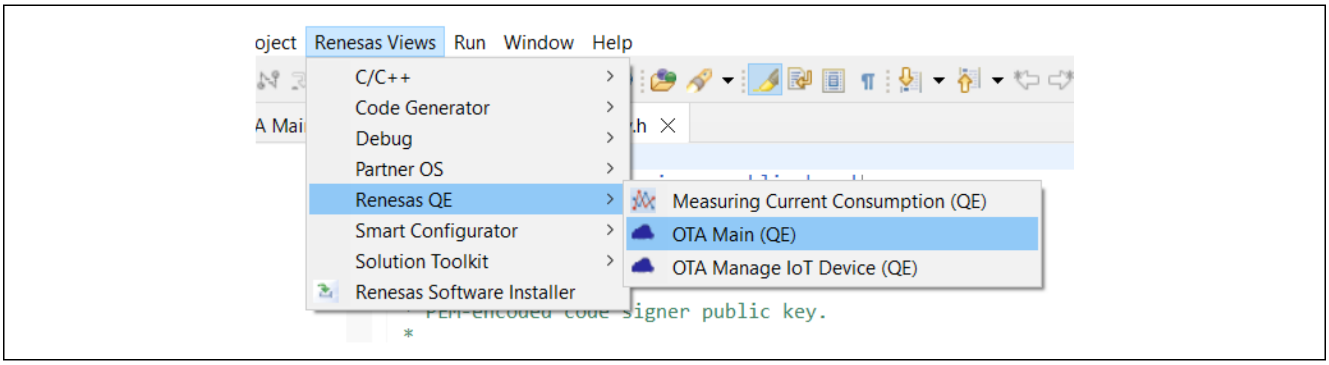
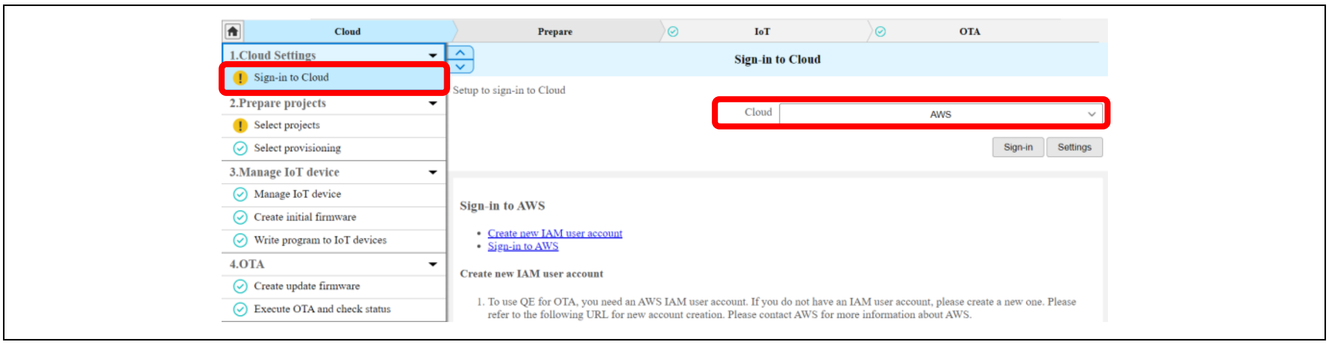


Figure 6-16 Opening the QE for OTA Window

(3) <QE for OTA> [1. Cloud Settings] → [Sign-in to Cloud]

From here, follow the steps displayed in the GUI window of QE for OTA.

Start by selecting “AWS” for [Cloud] and sign in. An AWS resource is generated in the region selected at the time of login.



Setup to sign-in to Cloud

Cloud: AWS

Sign-in to AWS

- Create new IAM user account
- Sign-in to AWS

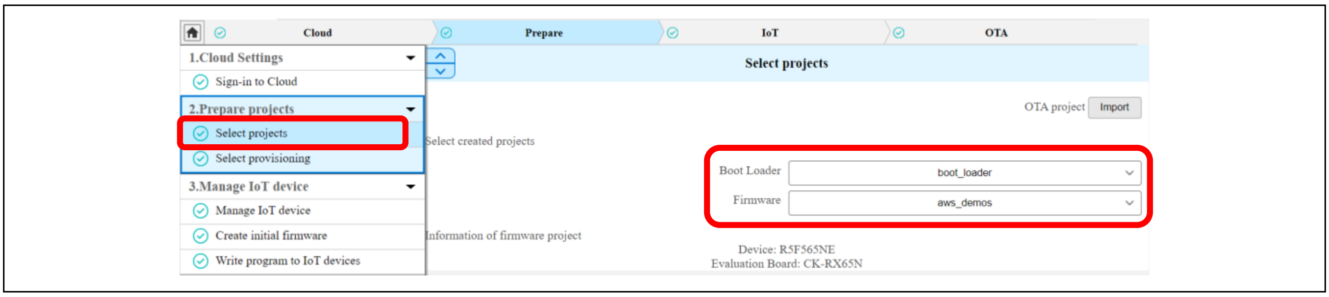
Create new IAM user account

1. To use QE for OTA, you need an AWS IAM user account. If you do not have an IAM user account, please create a new one. Please refer to the following URL for new account creation. Please contact AWS for more information about AWS.

Figure 6-17 Signing in to AWS with QE for OTA

(4) <QE for OTA> [2. Prepare projects] → [Select projects]

Select the ck_rx65n_demo_bootloader project and the ck_rx65n_2ndota_demo project that were imported into e² studio earlier.



Select created projects

Boot Loader: boot_loader

Firmware: aws_demos

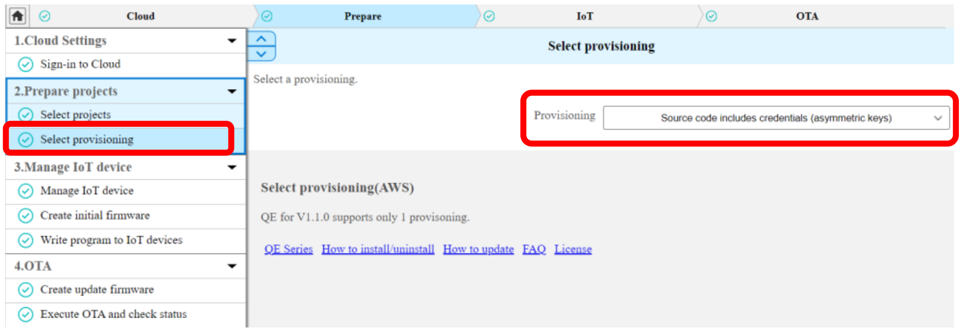
Information of firmware project

Device: R5F565NE
Evaluation Board: CK-RX65N

Figure 6-18 Selecting Projects

(5) <QE for OTA> [2. Prepare projects] → [Select provisioning]

Select “Source code includes credentials (asymmetric keys)” as the provisioning method.



Select provisioning

Select a provisioning.

Provisioning: Source code includes credentials (asymmetric keys)

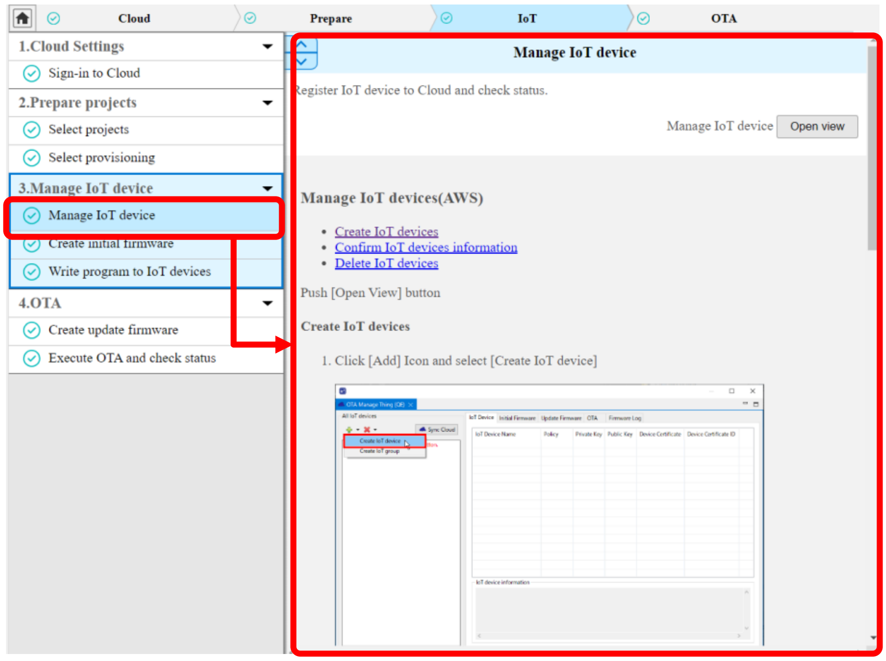
Select provisioning(AWS)

QE for V1.1.0 supports only 1 provisoning.

Figure 6-19 Selecting the Provisioning Method

(6) <QE for OTA> [3. Manage IoT device] → [Manage IoT device]

Create an IoT device following the procedure of QE for OTA.



Manage IoT device

Register IoT device to Cloud and check status.

Manage IoT device

Manage IoT devices(AWS)

- Create IoT devices
- Confirm IoT devices information
- Delete IoT devices

Push [Open View] button

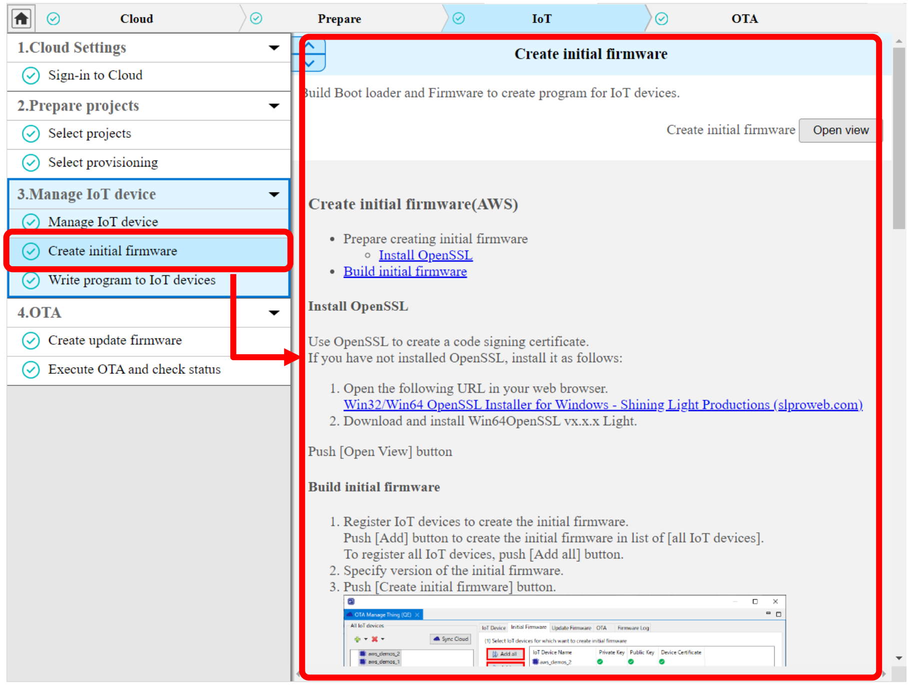
Create IoT devices

1. Click [Add] Icon and select [Create IoT device]

Figure 6-20 Creating an IoT Device

(7) <QE for OTA> [3. Manage IoT device] → [Create initial firmware]

Create initial firmware following the procedure of QE for OTA.



Create initial firmware

Build Boot loader and Firmware to create program for IoT devices.

Create initial firmware [Open view]

Create initial firmware(AWS)

- Prepare creating initial firmware
  - Install OpenSSL
- Build initial firmware

Install OpenSSL

Use OpenSSL to create a code signing certificate.
If you have not installed OpenSSL, install it as follows:

1. Open the following URL in your web browser.
   Win32/Win64 OpenSSL Installer for Windows - Shining Light Productions (slproweb.com)
2. Download and install Win64OpenSSL vx.x.x Light.

Push [Open View] button

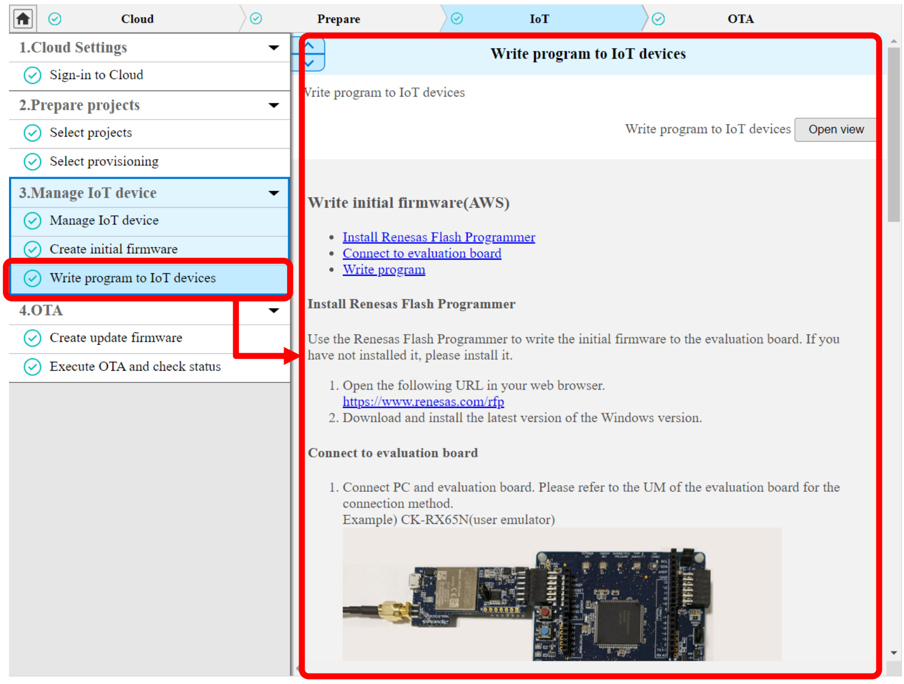
Build initial firmware

1. Register IoT devices to create the initial firmware.
   Push [Add] button to create the initial firmware in list of [all IoT devices].
   To register all IoT devices, push [Add all] button.
2. Specify version of the initial firmware.
3. Push [Create initial firmware] button.

Figure 6-21 Creating the Initial Firmware

(8) <QE for OTA> [3. Manage IoT device] → [Write program to IoT devices]

Write the initial firmware to the CK-RX65N following the procedure of QE for OTA.



Write program to IoT devices

Write program to IoT devices

Write program to IoT devices [Open view]

Write initial firmware(AWS)

- Install Renesas Flash Programmer
- Connect to evaluation board
- Write program

Install Renesas Flash Programmer

Use the Renesas Flash Programmer to write the initial firmware to the evaluation board. If you have not installed it, please install it.

1. Open the following URL in your web browser.
   https://www.renesas.com/rfp
2. Download and install the latest version of the Windows version.

Connect to evaluation board

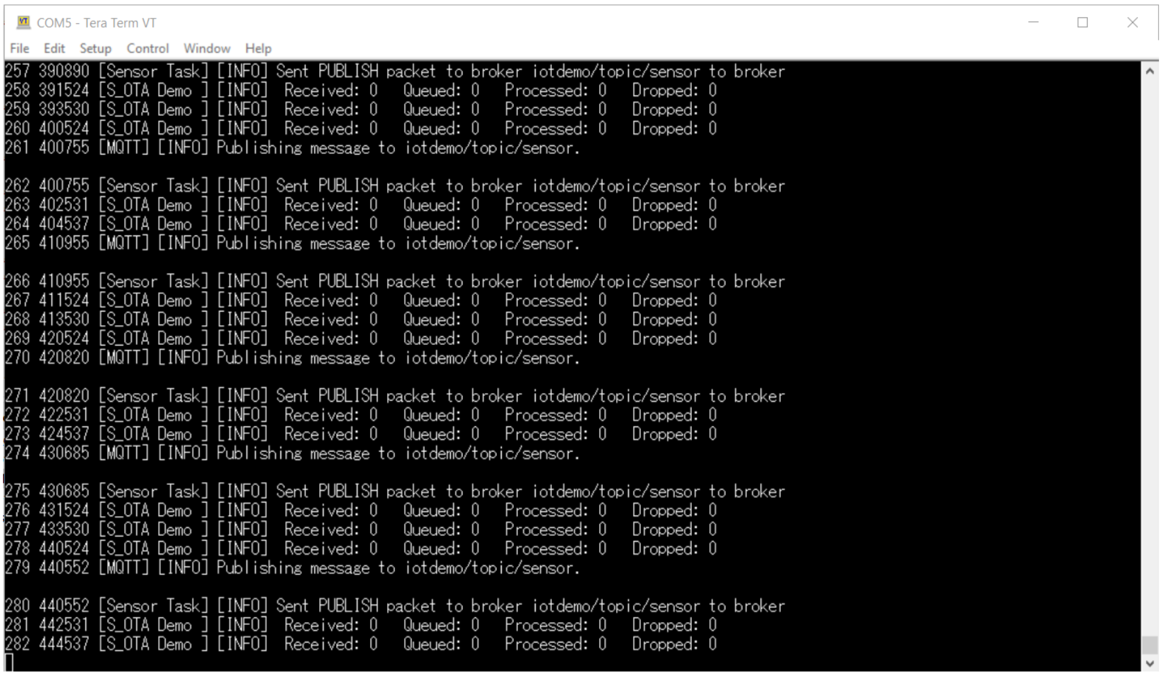
1. Connect PC and evaluation board. Please refer to the UM of the evaluation board for the connection method.
   Example) CK-RX65N(user emulator)

Figure 6-22 Writing the Initial Firmware

(9) Checking operation

Close jumper block J16 in Figure 6-6 on the RUN side (pins 2-3).

Launch the terminal software, and if the log is output as shown in Figure 6-23, the CK-RX65N is ready to run.



```
257 390890 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
258 391524 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
259 393530 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
260 400524 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
261 400755 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

262 400755 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
263 402531 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
264 404537 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
265 410955 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

266 410955 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
267 411524 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
268 413530 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
269 420524 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
270 420820 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

271 420820 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
272 422531 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
273 424537 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
274 430685 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

275 430685 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
276 431524 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
277 433530 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
278 440524 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
279 440552 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

280 440552 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
281 442531 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
282 444537 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
```

Figure 6-23 Log Screen of the CK-RX65N

6.2.4 Creating and Running the Initial Firmware for the TB-RX660

Create initial firmware for the TB-RX660 and write it to the microcontroller by using the Renesas Flash Programmer. The procedure is described below.

(1) Importing projects

Similarly to the procedure for importing projects for the CK-RX65N described in the previous subsection, import the “rx660_tb_demo_bootloader” project, a bootloader for the TB-RX660, and the “rx660_tb_2ndota_demo” project, a user program, into e^2 studio. These projects are provided as sample code with this application note.

(2) Building projects

Build the rx660_tb_demo_bootloader project and the rx660_tb_2ndota_demo project and create a MOT file for each. The MOT files are created in the HardwareDebug folder directly under the project folder.

(3) Creating the initial firmware

The initial firmware for the TB-RX660 is created by combining the created MOT files of the rx660_tb_demo_bootloader and rx660_tb_2ndota_demo projects. The Renesas Image Generator is a tool for use in combining MOT files and is included with the “RX Family Firmware Update Module Using Firmware Integration Technology Rev.2.01” Application Note. For details, refer to the “Renesas Image Generator” section in the application note at the link above.

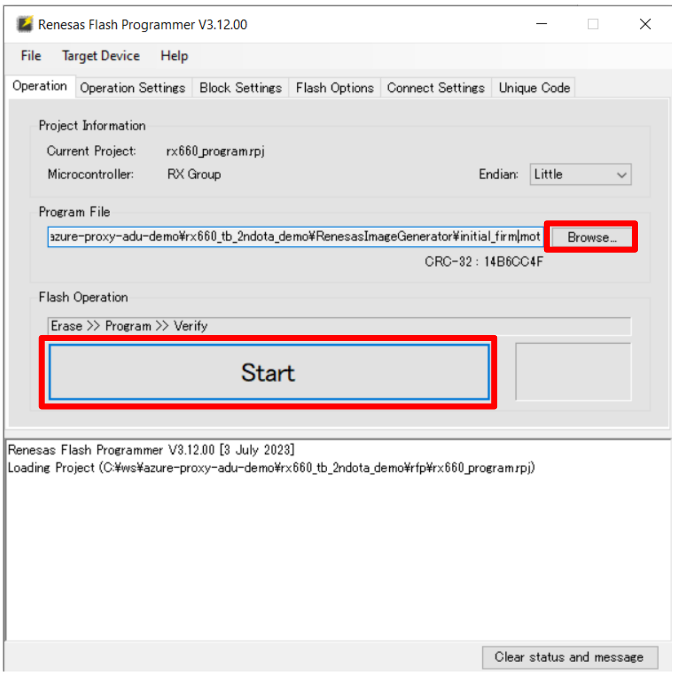
Execute the following command in the rx660_tb_2ndota_demo\RenesasImageGenerator folder to create the initial firmware “initial_firm.mot”.

```
> python .\image-gen.py -iup ..\HardwareDebug\rx660_tb_2ndota_demo.mot -
ibp ..\..\rx660_tb_demo_bootloader\HardwareDebug\rx660_tb_demo_bootloader.mot -
o .\initial_firm -key .\keys\secp256r1.privatekey -
ip .\RX660_Linear_Half_ImageGenerator_PRM.csv -vt ecdsa
```

(4) Writing the initial firmware

Using the Renesas Flash Programmer, write the initial firmware “initial_firm.mot” that was created in the previous step to the TB-RX660.

Open the Renesas Flash Programmer project “rx660_program.rpj” in the rx660_tb_2ndota_demo\rfp folder, specify the initial firmware for the RX660 “initial_firm.mot” that was just created in the [Program File] field, and click on [Start].



Project Information

Current Project: rx660_program.rpj

Microcontroller: RX Group

Endian: Little

Program File

azure-proxy-adu-demo¥rx660_tb_2ndota_demo¥RenesasImageGenerator¥initial_firm.mot

CRC-32 : 14B6CC4F

Flash Operation

Erase >> Program >> Verify

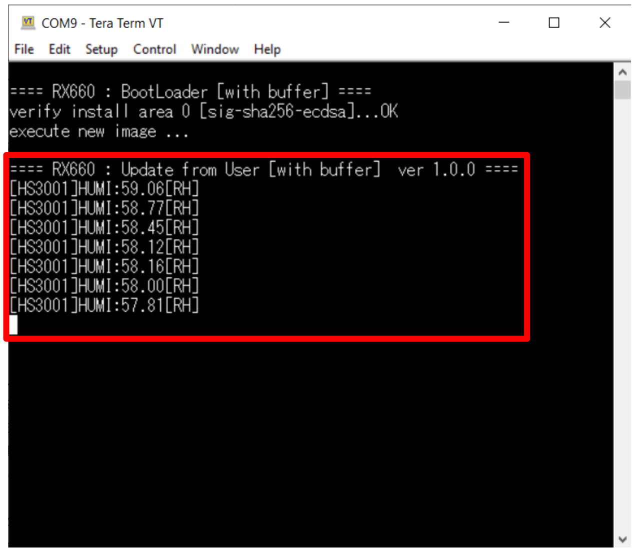
```
Renesas Flash Programmer V3.12.00 [3 July 2023]
Loading Project (C:¥ws¥azure-proxy-adu-demo¥rx660_tb_2ndota_demo¥rfp¥rx660_program.rpj)
```

Figure 6-24 Writing the Initial Firmware with the Renesas Flash Programmer

(5) Checking operation

Close the emulator reset header (J6) on the TB-RX660.

Launch the terminal software, and if the values for humidity are output in the log as shown in Figure 6-25, the TB-RX660 is ready to run.



```
==== RX660 : BootLoader [with buffer] ====
verify install area 0 [sig-sha256-ecdsa]...OK
execute new image ...

==== RX660 : Update from User [with buffer]  ver 1.0.0 ====
[HS3001]HUMI:59.06[RH]
[HS3001]HUMI:58.77[RH]
[HS3001]HUMI:58.45[RH]
[HS3001]HUMI:58.12[RH]
[HS3001]HUMI:58.16[RH]
[HS3001]HUMI:58.00[RH]
[HS3001]HUMI:57.81[RH]
```

Figure 6-25 Log Screen of the TB-RX660

6.3 Preparations for Using the AWS Cloud

Start by logging in to the AWS Management Console.

Manage AWS Resources - AWS Management Console - AWS (amazon.com)

Confirm the region displayed in the upper-right corner of the management console screen and select the same region as that set at the time of logging in to QE for OTA.

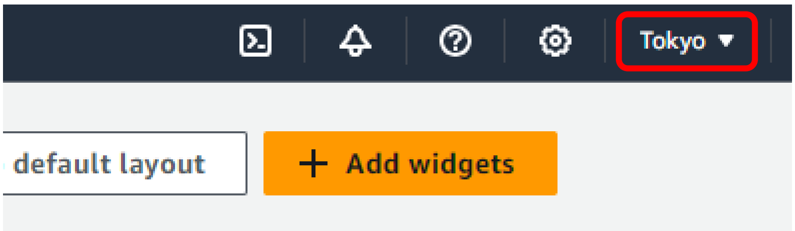


Figure 6-26 Confirming the Region

6.3.1 Settings for the OTA Update

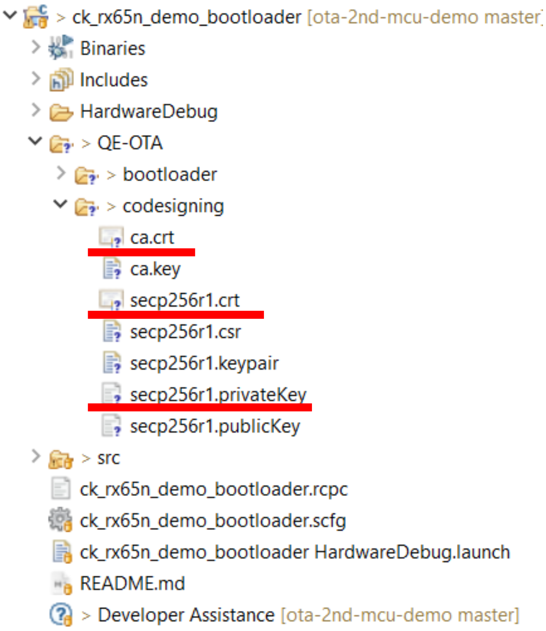
Refer to the "RX Family How to implement FreeRTOS OTA using Amazon Web Services in RX65N (for v202210.01-LTS-rx-1.1.0 or later)" Application Note and make the necessary settings.

(1) Create an Amazon S3 bucket according to the procedure described in "3.4 Creating an Amazon S3 bucket" in the above application note. The bucket name set here will be used when running the demonstration.

(2) Create a service role according to the procedure described in "3.5 Allocating OTA execution permission to IAM users" in the above application note. The service role name set here will be used when running the demonstration.

(3) Register a code signing certificate according to the procedure described in (5) to (9) in "5.2 Updating the firmware" in the above application note. The code signing certificate to be registered here is the certificate created when the initial firmware for the CK-RX65N was created by using QE for OTA in 6.2.3(7).

The certificate is created in "ck_rx65n_demo_bootloader/QE-OTA/codesigning". Specify secp256r1.crt for the certificate, secp256r1.privateKey for the private key, and ca.crt for the certificate chain.



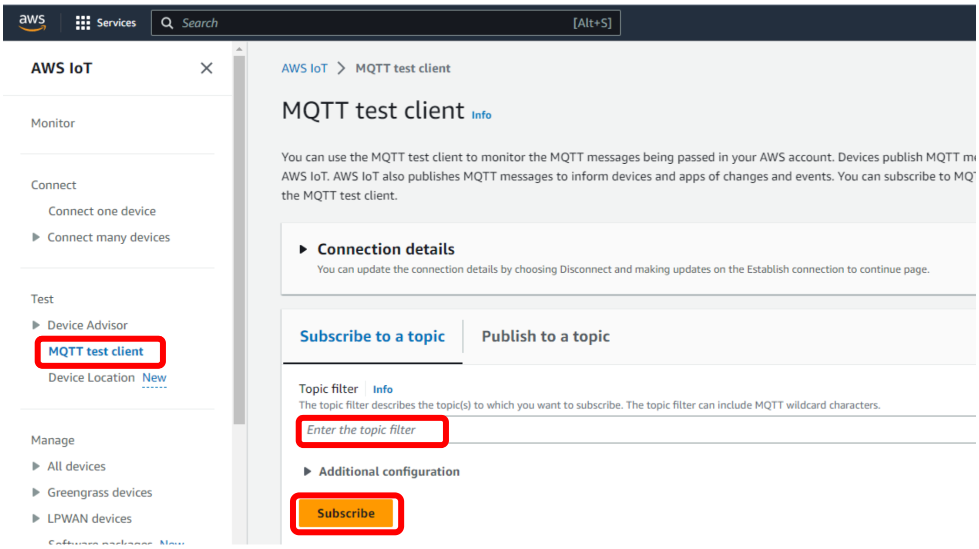
```
> ck_rx65n_demo_bootloader [ota-2nd-mcu-demo master]
    Binaries
    Includes
    HardwareDebug
    > QE-OTA
        > bootloader
        > codesigning
            ca.crt
            ca.key
            secp256r1.crt
            secp256r1.csr
            secp256r1.keypair
            secp256r1.privateKey
            secp256r1.publicKey
    > src
    ck_rx65n_demo_bootloader.rcpc
    ck_rx65n_demo_bootloader.scfg
    ck_rx65n_demo_bootloader HardwareDebug.launch
    README.md
    > Developer Assistance [ota-2nd-mcu-demo master]
```

Figure 6-27 Location Where the Code Signing Certificate is Created

6.3.2 Settings for Displaying the Sensor Data

To display the received sensor data in a graphical format, set up Amazon CloudWatch and AWS IoT Core through the following steps.

Note: If you do not need to display the data in a graphical format, but only need to confirm in your browser that the data have been received by AWS, you can omit the entire procedure in 6.3.2. In this case, as shown in Figure 6-28, you can subscribe to “`iotdemo/topic/sensor`” in [MQTT test client] of AWS IoT to confirm in a text format that the sensor data are being received as expected.



You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT m AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQ the MQTT test client.

You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

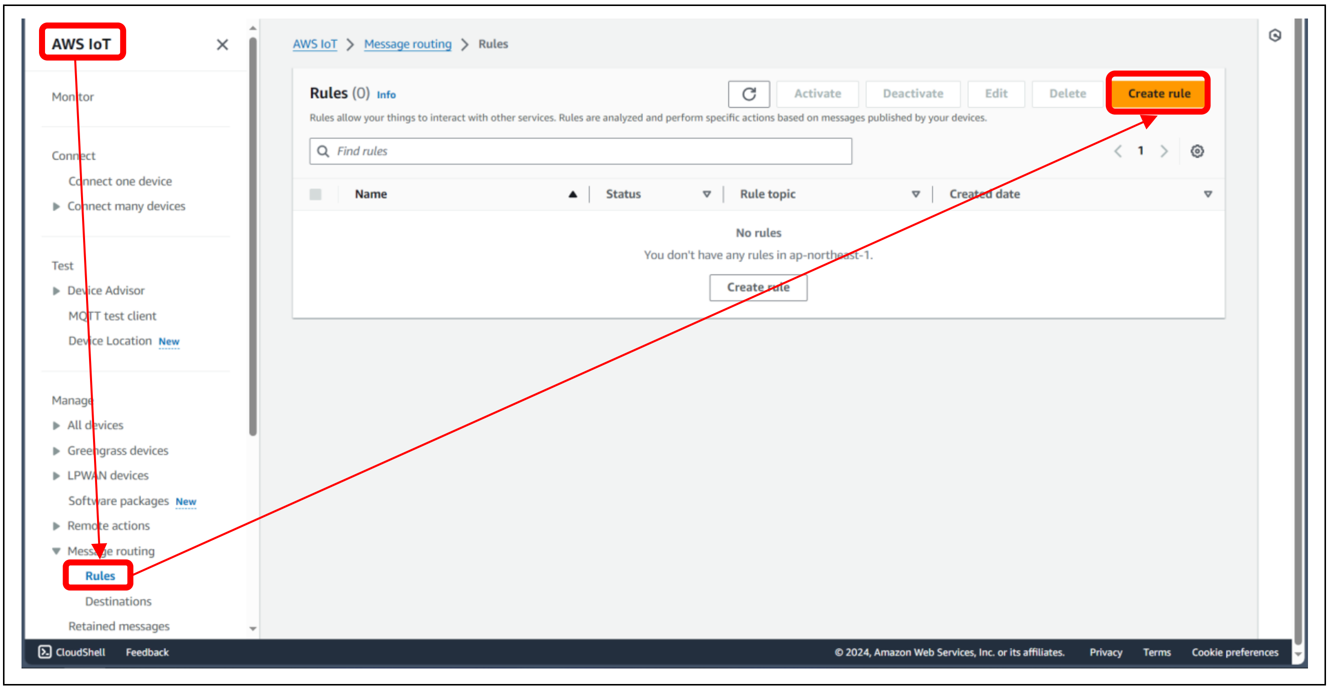
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

Figure 6-28 Confirming Data Reception by the MQTT Test Client

6.3.2.1 Setting up Amazon CloudWatch

(1) Creating rules in AWS IoT

Click on [AWS IoT] → [Rules] → [Create rule].



Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

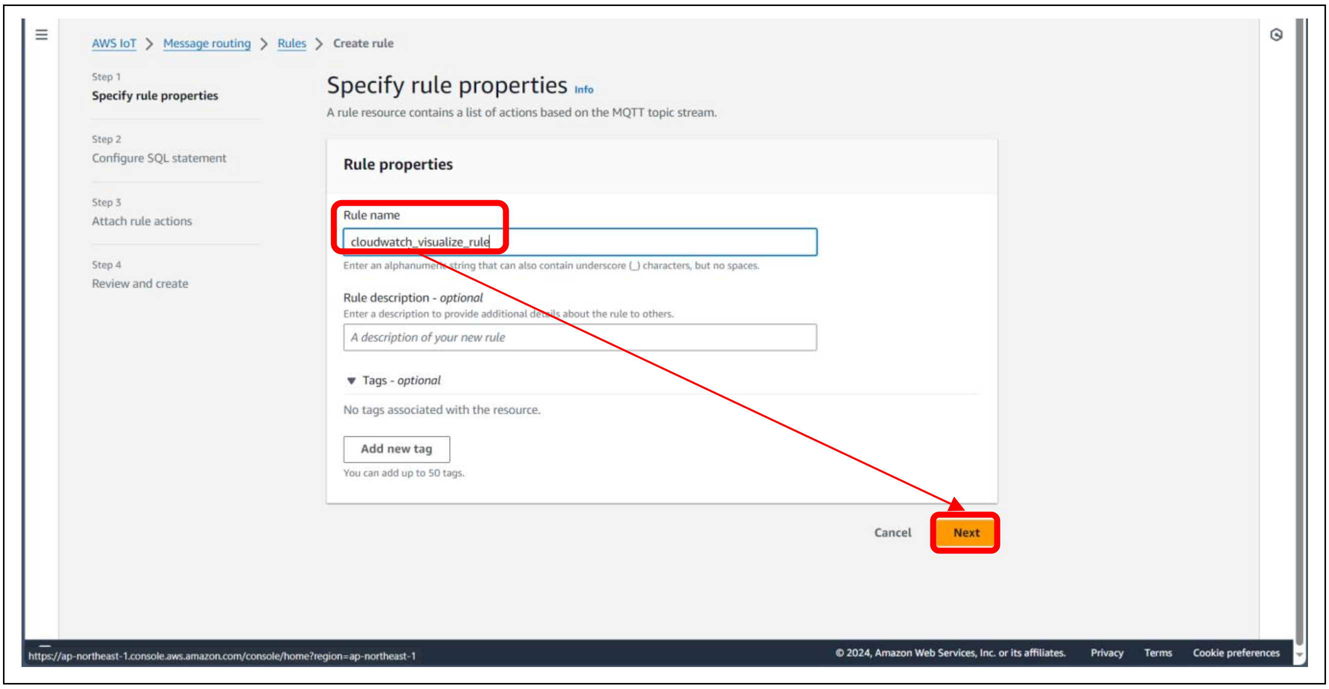
No rules

You don't have any rules in ap-northeast-1.

© 2024, Amazon Web Services, Inc. or its affiliates.

(2) Specifying the rule properties

Enter a rule name in [Rule name] and click on [Next].



A rule resource contains a list of actions based on the MQTT topic stream.

Rule name

cloudwatch_visualize_rule

Enter an alphanumeric string that can also contain underscore (_) characters, but no spaces.

Rule description - *optional*

Enter a description to provide additional details about the rule to others.

A description of your new rule

No tags associated with the resource.

You can add up to 50 tags.

https://ap-northeast-1.console.aws.amazon.com/console/home?region=ap-northeast-1

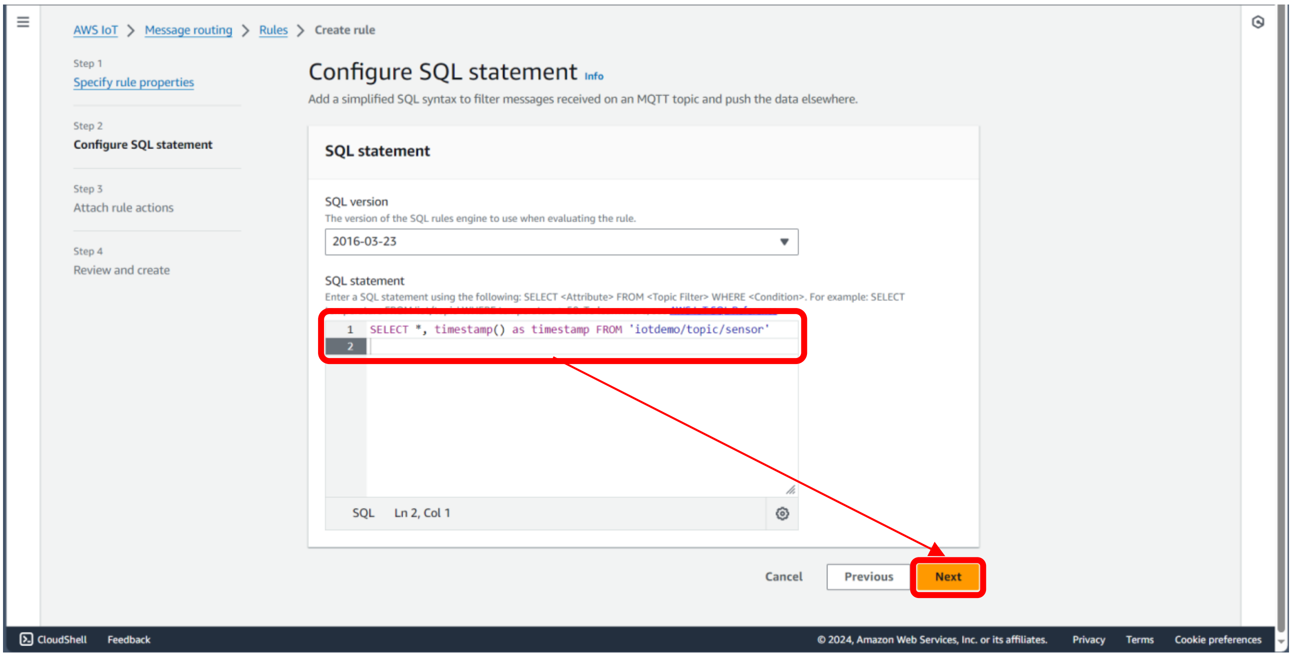
© 2024, Amazon Web Services, Inc. or its affiliates.

(3) Setting the SQL statement

Enter the SQL statement by entering code like the following in the text editor field for [SQL statement]. Be sure to add a new line character at the end.

```
SELECT *, timestamp() as timestamp FROM 'iotdemo/topic/sensor'
```

(A new line has to be entered at the end of the line above.)



AWS IoT > Message routing > Rules > Create rule

Step 1 Specify rule properties

Step 2 Configure SQL statement

Step 3 Attach rule actions

Step 4 Review and create

Configure SQL statement Info

Add a simplified SQL syntax to filter messages received on an MQTT topic and push the data elsewhere.

SQL statement

SQL version: 2016-03-23

The version of the SQL rules engine to use when evaluating the rule.

SQL statement

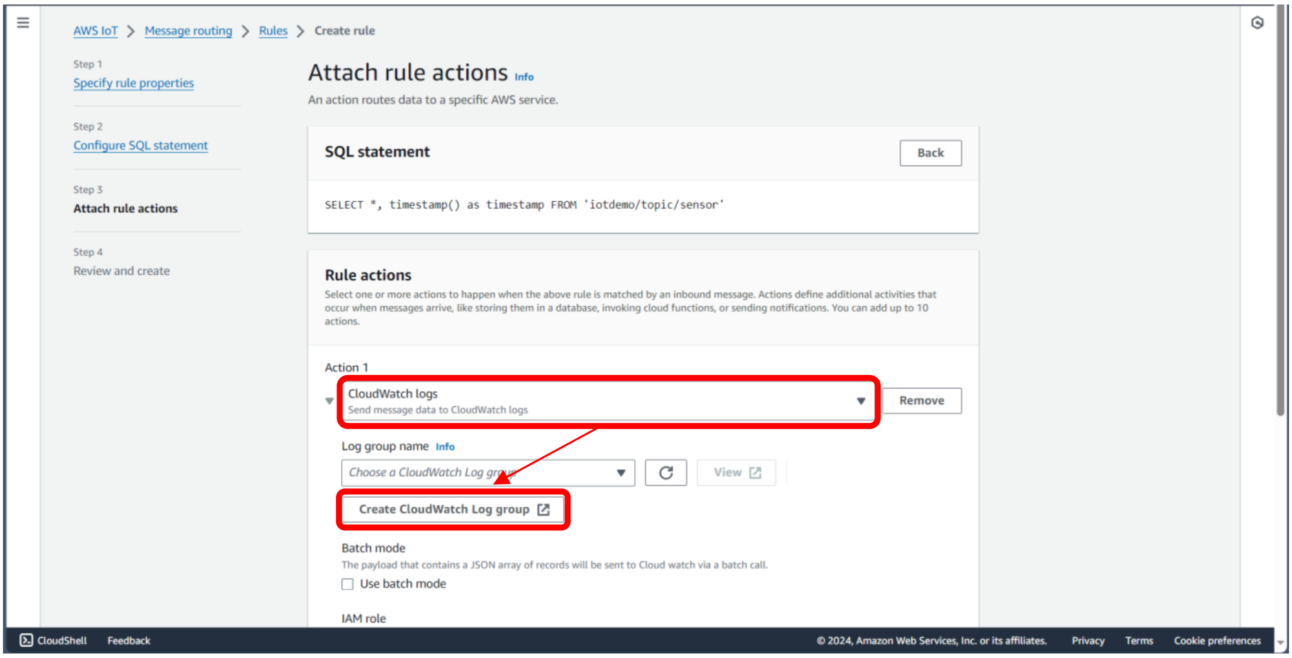
Enter a SQL statement using the following: SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT

```
SELECT *, timestamp() as timestamp FROM 'iotdemo/topic/sensor'
```

SQL Ln 2, Col 1

(4) Selecting rule actions in the [Attach rule actions] step

Select “CloudWatch logs” for [Action 1] and click on [Create CloudWatch Log group].



AWS IoT > Message routing > Rules > Create rule

Step 1 Specify rule properties

Step 2 Configure SQL statement

Step 3 Attach rule actions

Step 4 Review and create

Attach rule actions Info

An action routes data to a specific AWS service.

SQL statement

```
SELECT *, timestamp() as timestamp FROM 'iotdemo/topic/sensor'
```

Rule actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

Action 1: CloudWatch logs — Send message data to CloudWatch logs

Log group name Info: Choose a CloudWatch Log group

Batch mode

The payload that contains a JSON array of records will be sent to Cloud watch via a batch call.

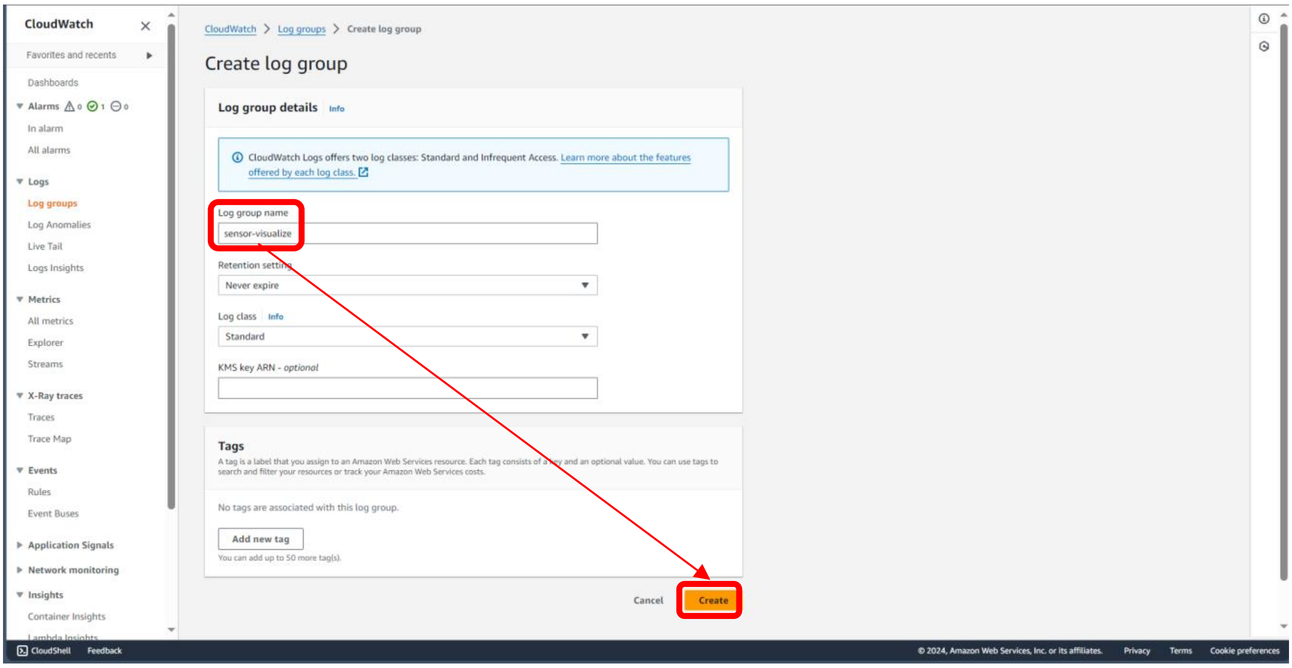
□ Use batch mode

IAM role

Back
Remove
View
Create CloudWatch Log group
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

(5) Creating a log group

Enter a log group name and click on [Create].



Create log group

Log group details Info

CloudWatch Logs offers two log classes: Standard and Infrequent Access. Learn more about the features offered by each log class.

Log group name: sensor-visualize

Retention setting: Never expire

Log class Info: Standard

KMS key ARN - *optional*

Tags

A tag is a label that you assign to an Amazon Web Services resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your Amazon Web Services costs.

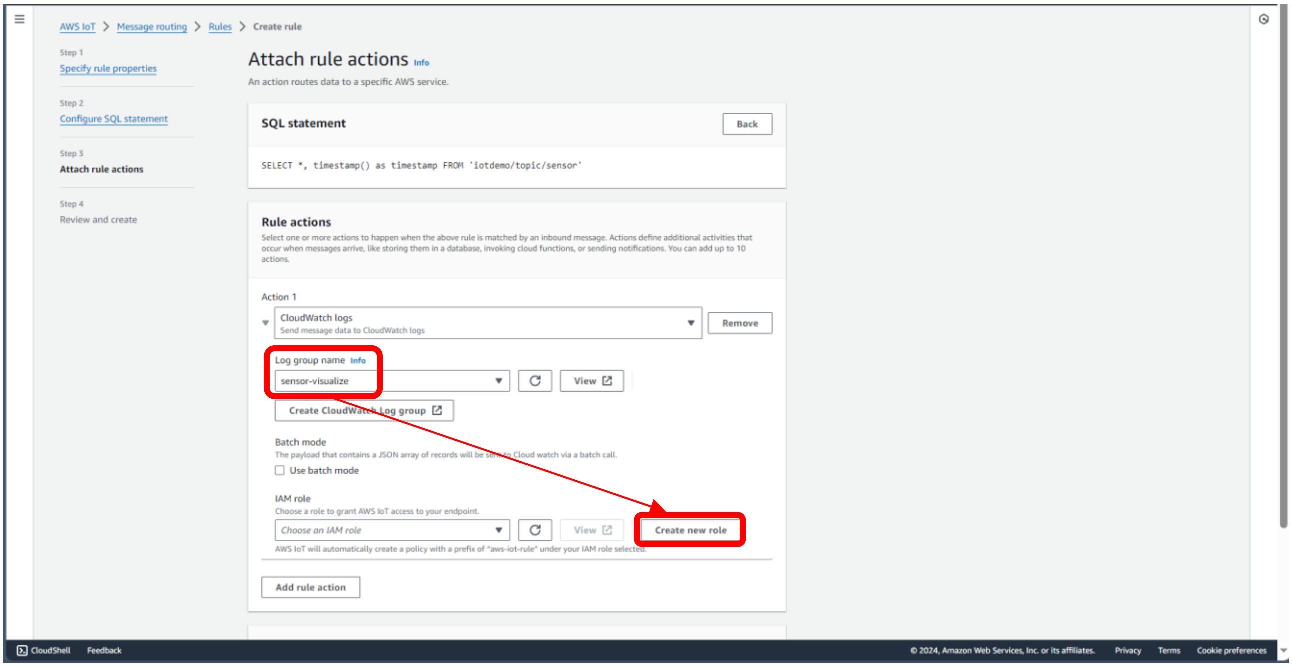
No tags are associated with this log group.

Add new tag

You can add up to 50 more tag(s).

(6) Creating a new role

Select the created log group in [Log group name] and click on [Create new role].



Attach rule actions Info

An action routes data to a specific AWS service.

SQL statement

```
SELECT *, timestamp() as timestamp FROM 'iotdemo/topic/sensor'
```

Rule actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

Action 1: CloudWatch logs — Send message data to CloudWatch logs

Log group name Info: sensor-visualize

Batch mode

The payload that contains a JSON array of records will be sent to Cloud watch via a batch call.

☐ Use batch mode

IAM role

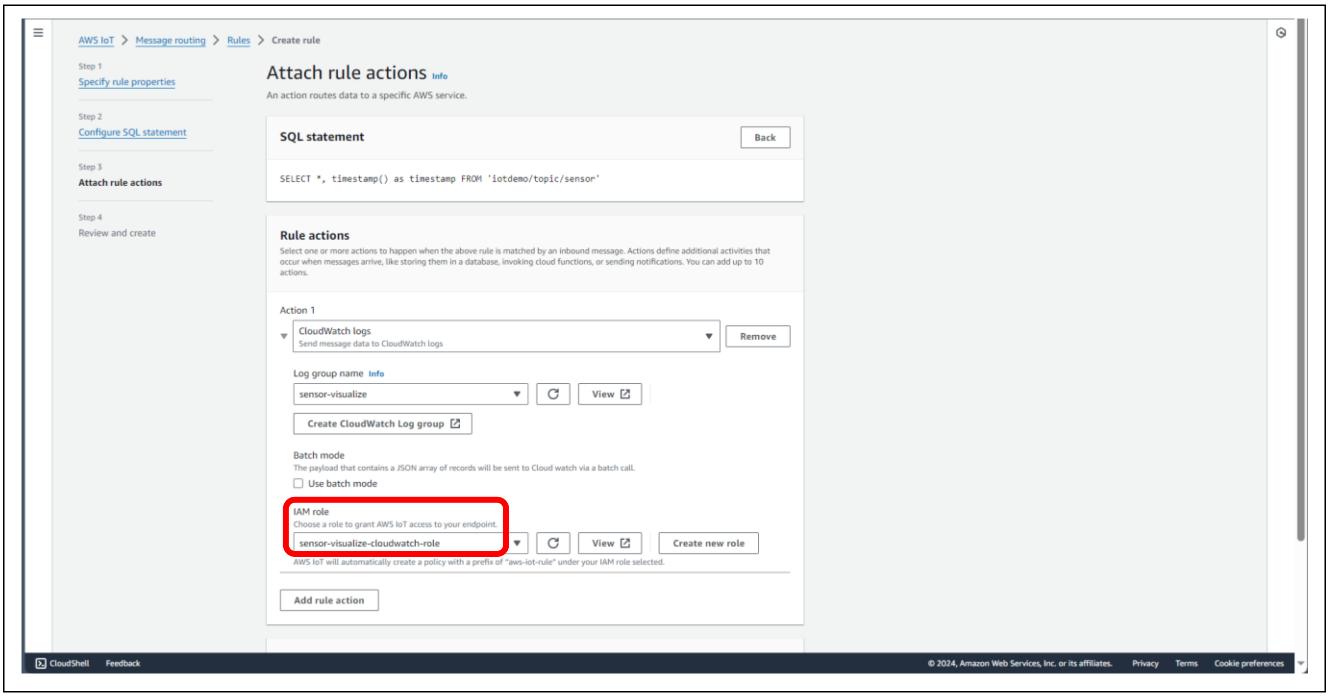
Choose a role to grant AWS IoT access to your endpoint.

Choose an IAM role

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

(7) Selecting the created IAM role

Select the created role in [IAM role].



AWS IoT > Message routing > Rules > Create rule

Step 1 Specify rule properties
Step 2 Configure SQL statement
Step 3 Attach rule actions
Step 4 Review and create

Attach rule actions Info

An action routes data to a specific AWS service.

SQL statement

```
SELECT *, timestamp() as timestamp FROM 'iotdemo/topic/sensor'
```

Rule actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

Action 1: CloudWatch logs — Send message data to CloudWatch logs

Log group name: sensor-visualize

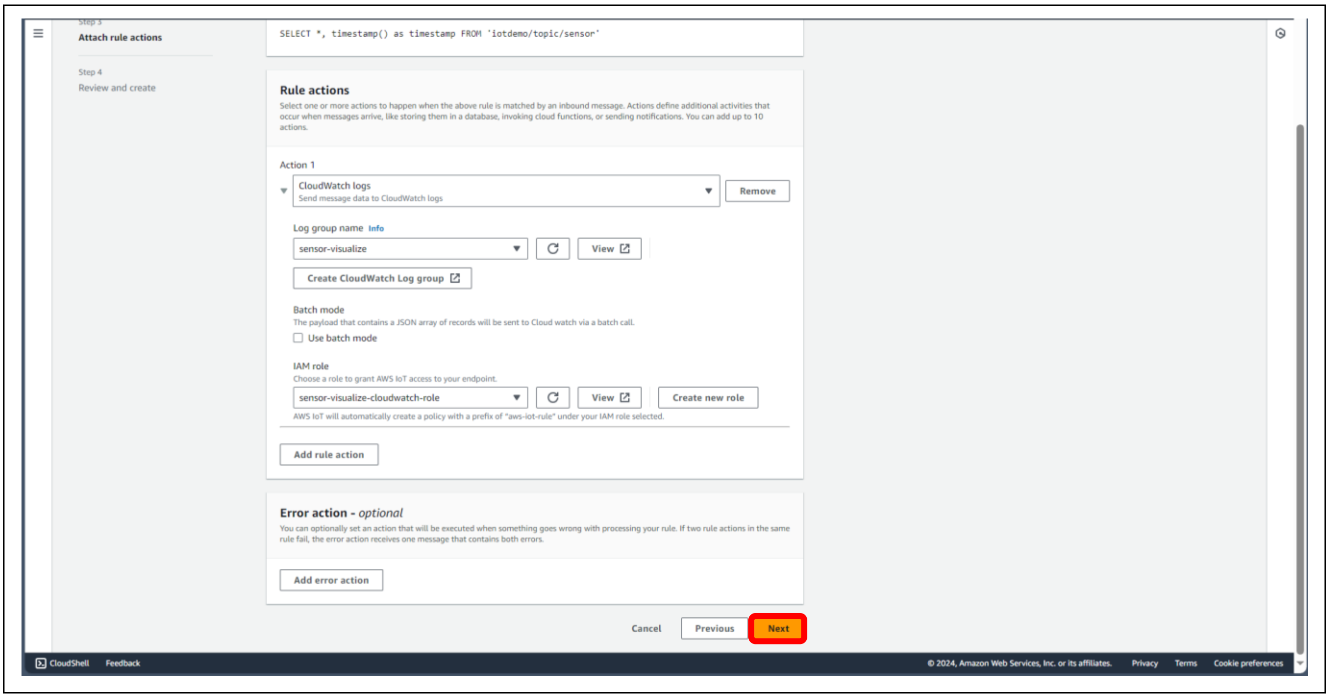
Batch mode: The payload that contains a JSON array of records will be sent to Cloud watch via a batch call.
□ Use batch mode

IAM role: Choose a role to grant AWS IoT access to your endpoint.
sensor-visualize-cloudwatch-role

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

(8) Confirming successful creation of the rule

Click on [Next] and then click on [Create] on the subsequent page. Finally, confirm that the created rule is displayed in the list of rules.



Step 3 Attach rule actions
Step 4 Review and create

```
SELECT *, timestamp() as timestamp FROM 'iotdemo/topic/sensor'
```

Rule actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. You can add up to 10 actions.

Action 1: CloudWatch logs — Send message data to CloudWatch logs

Log group name: sensor-visualize

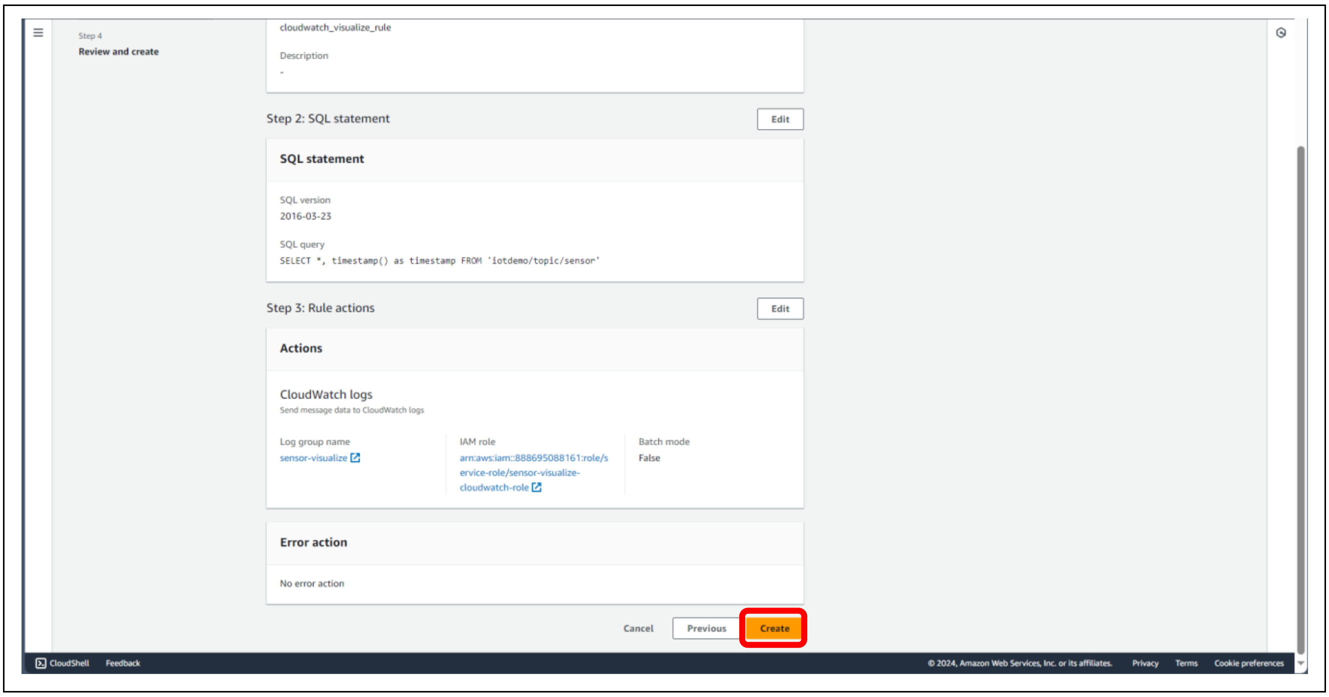
Batch mode: The payload that contains a JSON array of records will be sent to Cloud watch via a batch call.
□ Use batch mode

IAM role: Choose a role to grant AWS IoT access to your endpoint.
sensor-visualize-cloudwatch-role

AWS IoT will automatically create a policy with a prefix of "aws-iot-rule" under your IAM role selected.

Error action - *optional*

You can optionally set an action that will be executed when something goes wrong with processing your rule. If two rule actions in the same rule fail, the error action receives one message that contains both errors.



Step 4
Review and create

cloudwatch_visualize_rule

Description
-

Step 2: SQL statement

SQL statement

SQL version
2016-03-23

SQL query

```
SELECT *, timestamp() as timestamp FROM 'iotdemo/topic/sensor'
```

Step 3: Rule actions

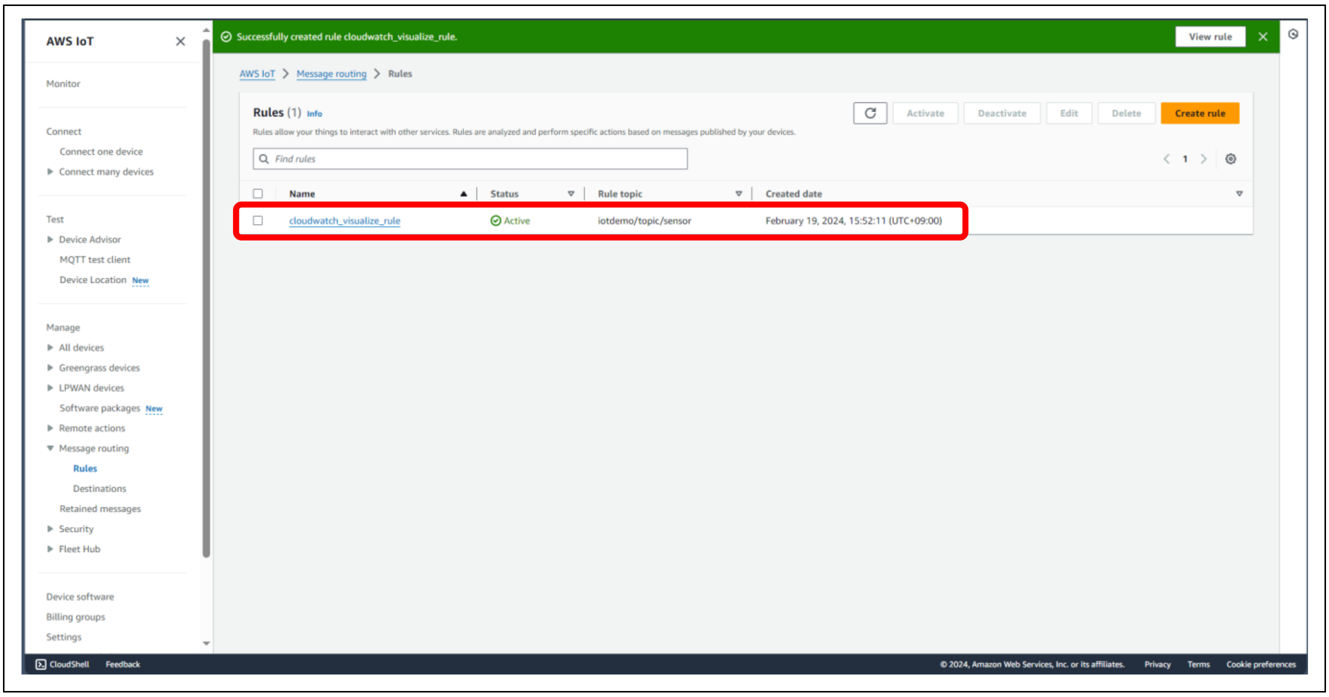
Actions

CloudWatch logs
Send message data to CloudWatch logs

Log group name	IAM role	Batch mode
sensor-visualize	arn:aws:iam::888695088161:role/service-role/sensor-visualize-cloudwatch-role	False

Error action

No error action



Successfully created rule cloudwatch_visualize_rule.

AWS IoT > Message routing > Rules

Rules (1) Info

Rules allow your things to interact with other services. Rules are analyzed and perform specific actions based on messages published by your devices.

□	Name	Status	Rule topic	Created date
□	cloudwatch_visualize_rule	✓ Active	iotdemo/topic/sensor	February 19, 2024, 15:52:11 (UTC+09:00)

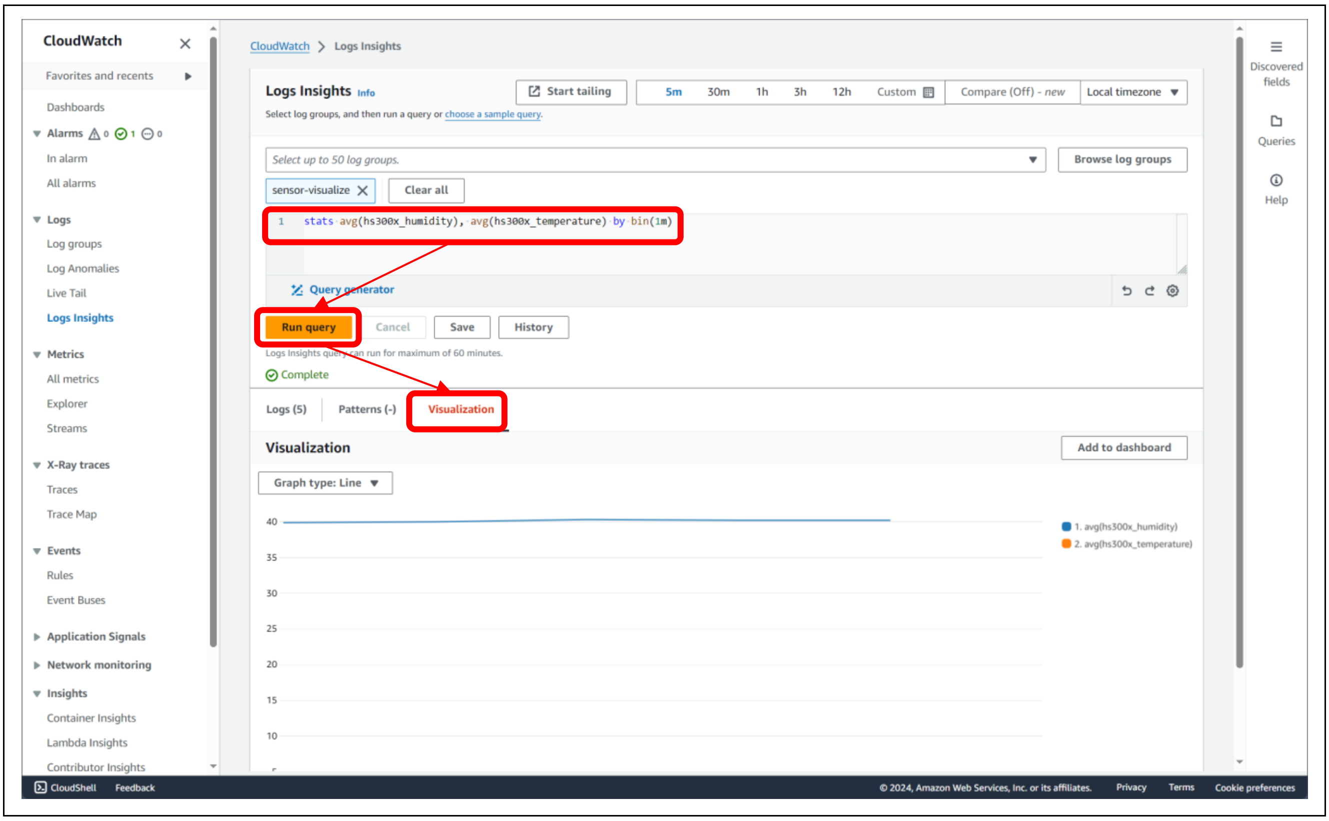
(9) Checking the graphical display in Amazon CloudWatch

Display the screen of Amazon CloudWatch and click on [Logs Insights] on the menu at left.

Select the group that was created in 6.3.2.1(5) as the log group, enter the following query, and click on [Run query].

```
stats avg(hs300x_humidity), avg(hs300x_temperature) by bin(1m)
```

A graph is displayed on the [Visualization] tabbed page.



Select log groups, and then run a query or choose a sample query.

sensor-visualize

```
stats avg(hs300x_humidity), avg(hs300x_temperature) by bin(1m)
```

Logs Insights query can run for maximum of 60 minutes.

Complete

7. Procedure for Running the Demonstration

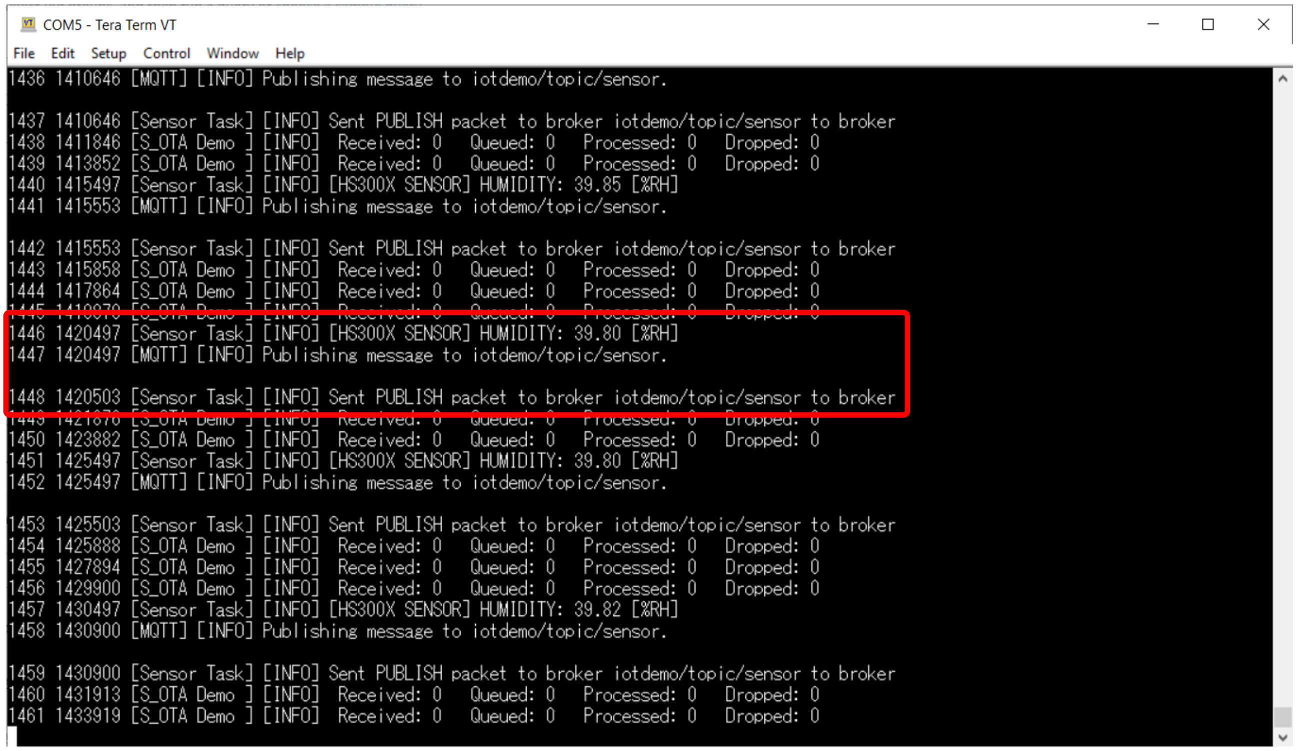
The procedure for running the demonstration is described below.

7.1 Checking the Initial State of Operation

With the setup for the demonstration described in section 6 completed, press the reset switch (RESET) on the TB-RX660 to apply a hardware reset. Similarly, press the reset switch (S1) on the CK-RX65N to apply a hardware reset.

Check the logs from each of the microcontrollers by using terminal software.

Figure 7-1 shows the log screen of the CK-RX65N. Confirm that humidity data from the HS3001 sensor are being output. Below that, you can also see the log of sensor data being sent to AWS via MQTT communications.

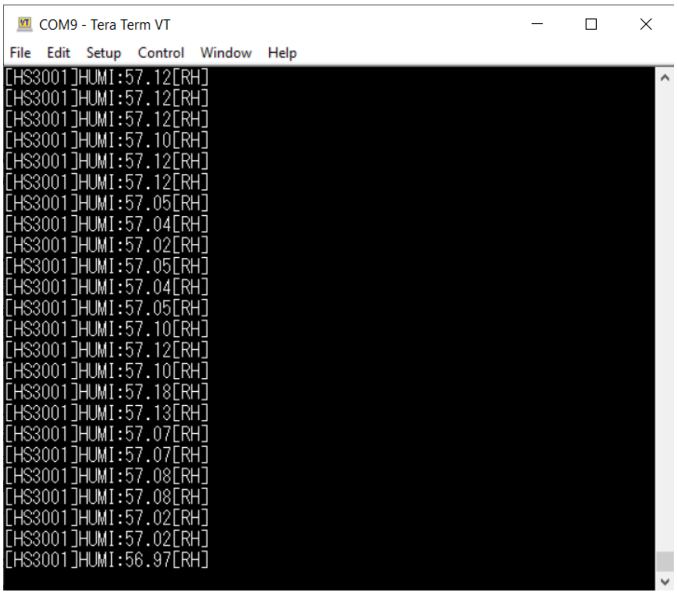


```
1436 1410646 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

1437 1410646 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
1438 1411846 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1439 1413852 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1440 1415497 [Sensor Task] [INFO] [HS300X SENSOR] HUMIDITY: 39.85 [%RH]
1441 1415553 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

1442 1415553 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
1443 1415858 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1444 1417864 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1445 1419870 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1446 1420497 [Sensor Task] [INFO] [HS300X SENSOR] HUMIDITY: 39.80 [%RH]
1447 1420497 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

1448 1420503 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
1449 1421876 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1450 1423882 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1451 1425497 [Sensor Task] [INFO] [HS300X SENSOR] HUMIDITY: 39.80 [%RH]
1452 1425497 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

1453 1425503 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
1454 1425888 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1455 1427894 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1456 1429900 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1457 1430497 [Sensor Task] [INFO] [HS300X SENSOR] HUMIDITY: 39.82 [%RH]
1458 1430900 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

1459 1430900 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
1460 1431913 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
1461 1433919 [S_OTA Demo ] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
```

Figure 7-1 Log Screen of the CK-RX65N

Next, Figure 7-2 shows the log screen of the TB-RX660. Confirm that only the humidity data from the HS3001 sensor are displayed.

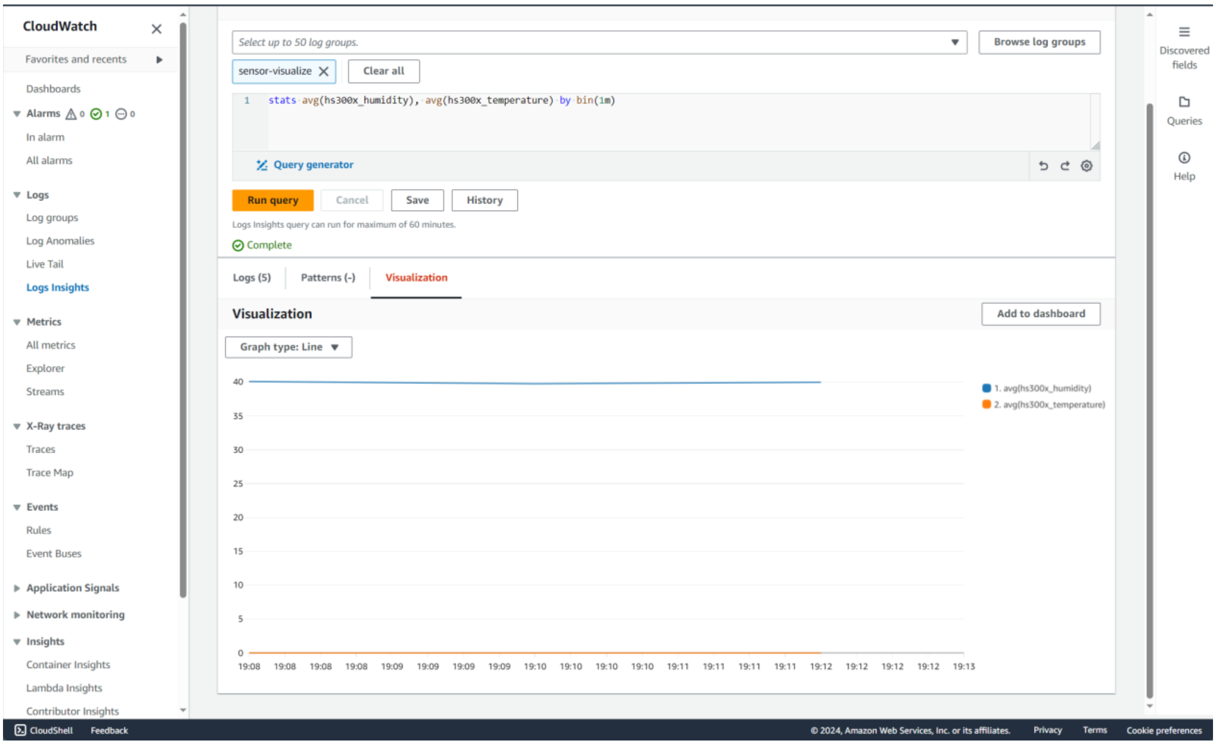
In the initial state, LED0 of the TB-RX660 is blinking.



```
[HS3001]HUMI:57.12[RH]
[HS3001]HUMI:57.12[RH]
[HS3001]HUMI:57.12[RH]
[HS3001]HUMI:57.10[RH]
[HS3001]HUMI:57.12[RH]
[HS3001]HUMI:57.12[RH]
[HS3001]HUMI:57.05[RH]
[HS3001]HUMI:57.04[RH]
[HS3001]HUMI:57.02[RH]
[HS3001]HUMI:57.05[RH]
[HS3001]HUMI:57.04[RH]
[HS3001]HUMI:57.05[RH]
[HS3001]HUMI:57.10[RH]
[HS3001]HUMI:57.12[RH]
[HS3001]HUMI:57.10[RH]
[HS3001]HUMI:57.18[RH]
[HS3001]HUMI:57.13[RH]
[HS3001]HUMI:57.07[RH]
[HS3001]HUMI:57.07[RH]
[HS3001]HUMI:57.08[RH]
[HS3001]HUMI:57.08[RH]
[HS3001]HUMI:57.02[RH]
[HS3001]HUMI:57.02[RH]
[HS3001]HUMI:56.97[RH]
```

Figure 7-2 Log Screen of the TB-RX660

Finally, Figure 7-3 shows the display for Amazon CloudWatch. Confirm that the humidity data acquired from the HS3001 sensor are displayed as a graph.



```
stats avg(hs300x_humidity), avg(hs300x_temperature) by bin(1m)
```

Figure 7-3 Graphical Display of Amazon CloudWatch before the Secondary OTA Update

This is the initial state before the secondary OTA update is run.

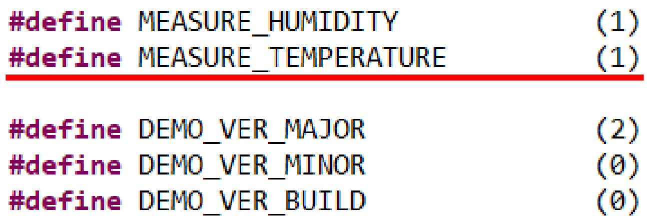
7.2 Executing the OTA Update of the TB-RX660

7.2.1 Creating the Update Firmware

(1) Changing the source code of the rx660_tb_2ndota_demo project

Set the MEASURE_TEMPERATURE macro of rx660_tb_2ndota_demo/src/rx660_tb_2ndota_demo.c to 1.

The version values displayed in the log at execution can be changed by changing DEMO_VER_MAJOR, DEMO_VER_MINOR, and DEMO_VER_BUILD, which are displayed below the above macro.



```
#define MEASURE_HUMIDITY             (1)
#define MEASURE_TEMPERATURE          (1)

#define DEMO_VER_MAJOR               (2)
#define DEMO_VER_MINOR               (0)
#define DEMO_VER_BUILD               (0)
```

(2) Creating the update firmware (MOT file format)

Build the rx660_tb_2ndota_demo project and create a MOT file.

(3) Creating the update firmware (RSU file format)

Convert the created rx660_tb_2ndota_demo MOT file into update firmware in the RSU format by using the Renesas Image Generator.

Run the following command in the rx660_tb_2ndota_demo\RenesasImageGenerator folder to create the RSU-format update firmware “update_firm.rsu”.

```
> python .\image-gen.py -iup ..\HardwareDebug\rx660_tb_2ndota_demo.mot -o .\update_firm
-key .\keys\secp256r1.privatekey -ip .\RX660_Linear_Half_ImageGenerator_PRM.csv -vt
ecdsa
```

7.2.2 Creating an OTA Job in AWS

(1) Sign in to the AWS Management Console and select [Services] in the upper-left corner, then select [Internet of Things] → [IoT Core].

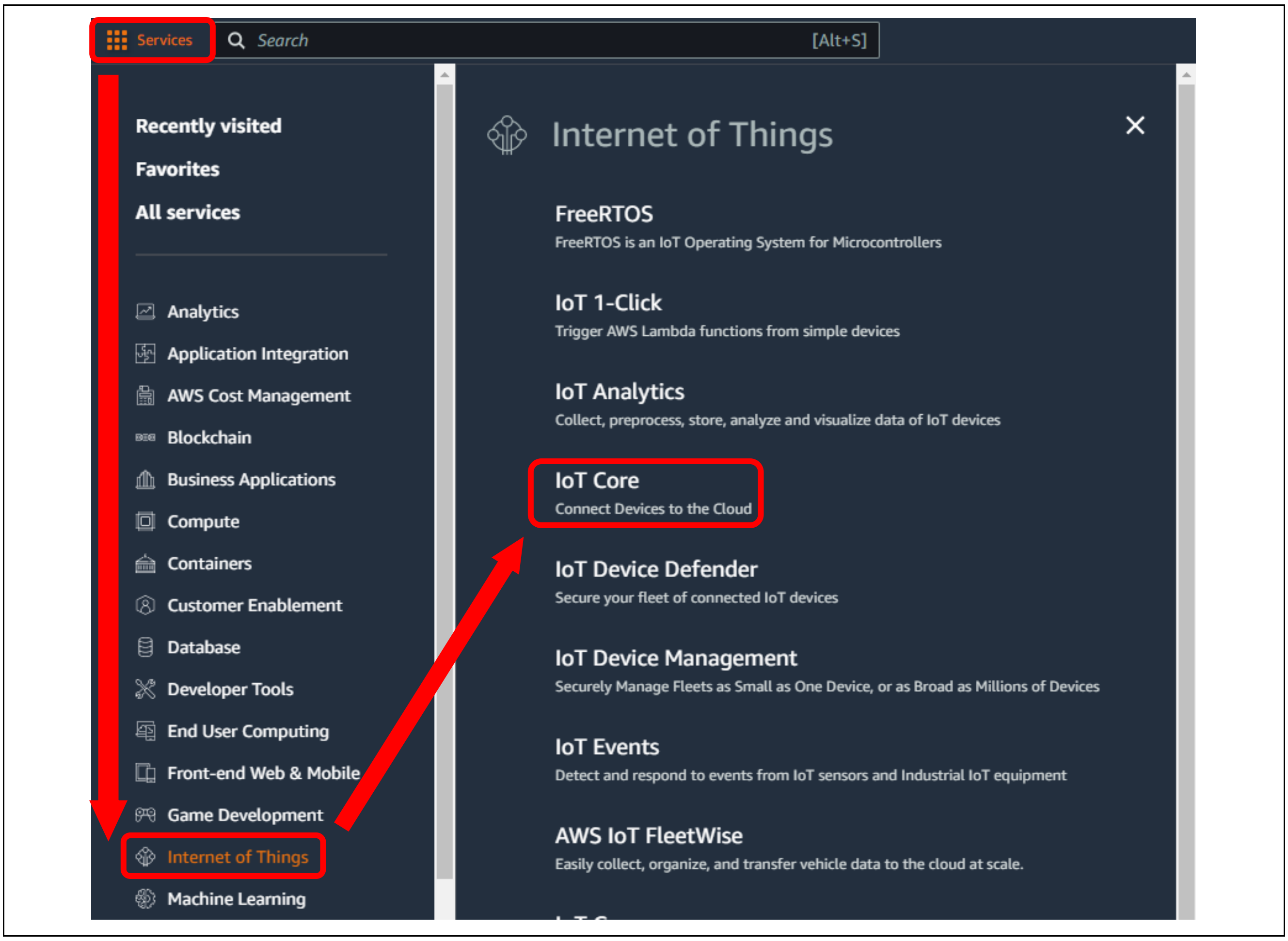
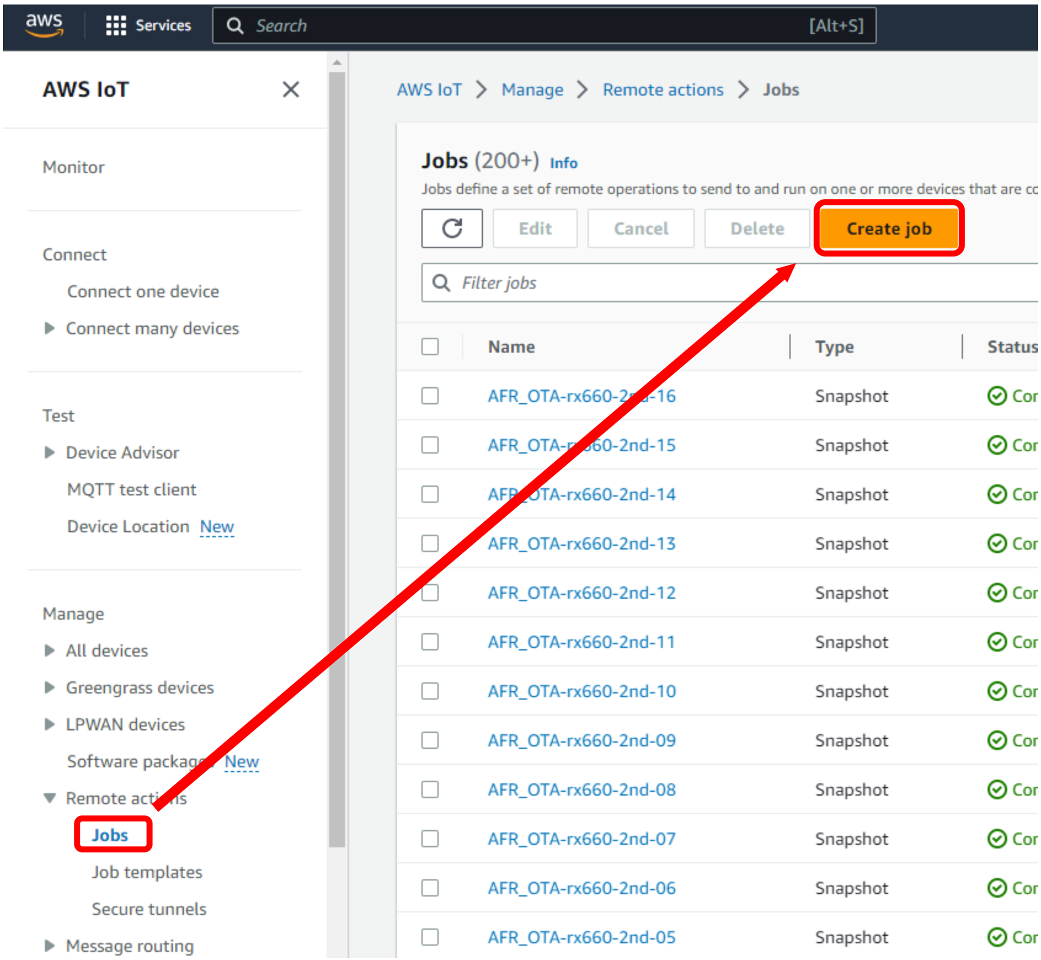

Services
Search
[Alt+S]
Recently visited
Favorites
All services
Analytics
Application Integration
AWS Cost Management
Blockchain
Business Applications
Compute
Containers
Customer Enablement
Database
Developer Tools
End User Computing
Front-end Web & Mobile
Game Development
Internet of Things
Machine Learning
Internet of Things
FreeRTOS
FreeRTOS is an IoT Operating System for Microcontrollers
IoT 1-Click
Trigger AWS Lambda functions from simple devices
IoT Analytics
Collect, preprocess, store, analyze and visualize data of IoT devices
IoT Core
Connect Devices to the Cloud
IoT Device Defender
Secure your fleet of connected IoT devices
IoT Device Management
Securely Manage Fleets as Small as One Device, or as Broad as Millions of Devices
IoT Events
Detect and respond to events from IoT sensors and Industrial IoT equipment
AWS IoT FleetWise
Easily collect, organize, and transfer vehicle data to the cloud at scale.

Figure 7-4 Window of Services of AWS

(2) Select [Remote action] → [Jobs] from the menu at left in AWS IoT Core and click on [Create job].

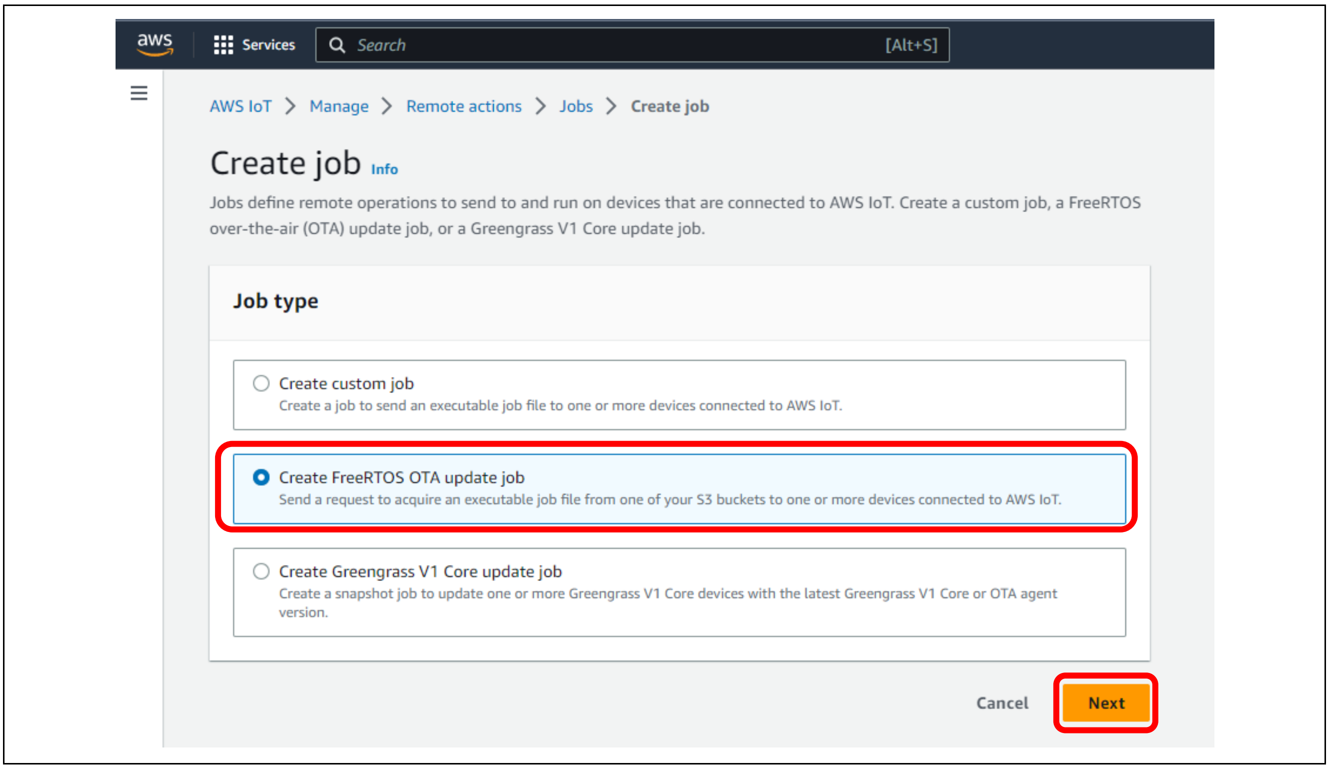


Jobs define a set of remote operations to send to and run on one or more devices that are co

□	Name	Type	Status
□	AFR_OTA-rx660-2nd-16	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-15	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-14	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-13	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-12	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-11	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-10	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-09	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-08	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-07	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-06	Snapshot	✓ Cor
□	AFR_OTA-rx660-2nd-05	Snapshot	✓ Cor

Figure 7-5 Window of AWS IoT Core

(3) On the [Create job] page, select “Create FreeRTOS OTA update job” and click on [Next].



Create job Info

Jobs define remote operations to send to and run on devices that are connected to AWS IoT. Create a custom job, a FreeRTOS over-the-air (OTA) update job, or a Greengrass V1 Core update job.

Job type

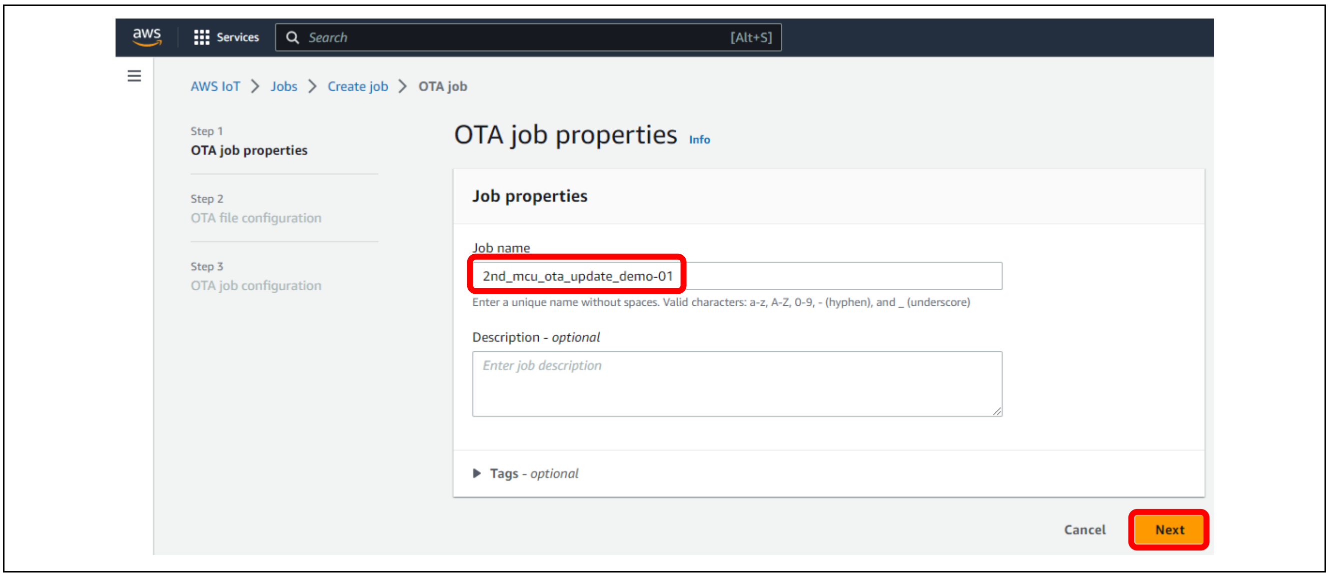
○ Create custom job
Create a job to send an executable job file to one or more devices connected to AWS IoT.

● Create FreeRTOS OTA update job
Send a request to acquire an executable job file from one of your S3 buckets to one or more devices connected to AWS IoT.

○ Create Greengrass V1 Core update job
Create a snapshot job to update one or more Greengrass V1 Core devices with the latest Greengrass V1 Core or OTA agent version.

Figure 7-6 Page for Creating a Job

(4) On the [OTA job properties] page, enter a job name in [Job name] and click on [Next].



OTA job properties Info

Job properties

Job name: 2nd_mcu_ota_update_demo-01

Enter a unique name without spaces. Valid characters: a-z, A-Z, 0-9, - (hyphen), and _ (underscore)

Description - *optional*: Enter job description

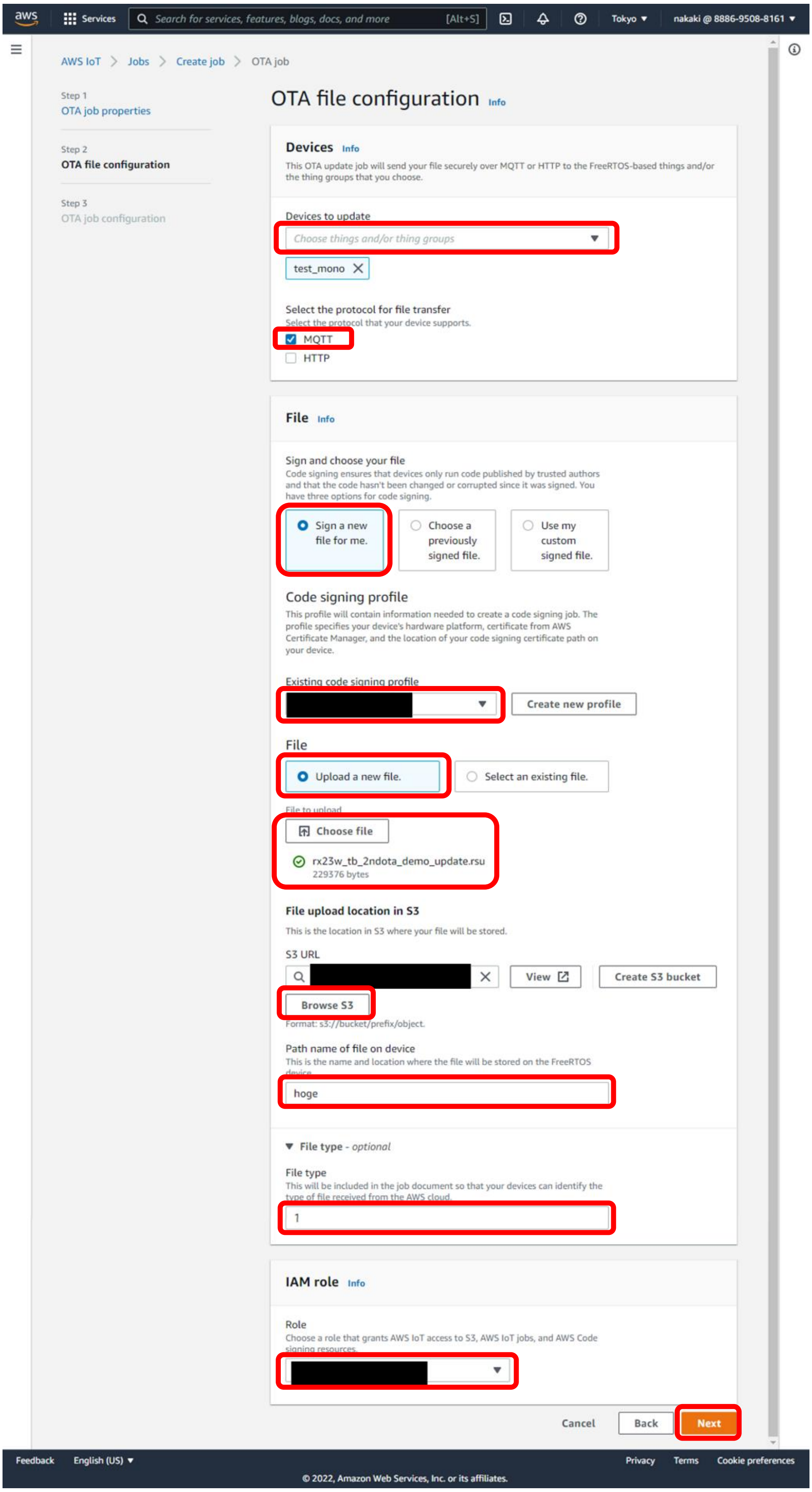
Figure 7-7 Page for Entering the Properties of the OTA Job

(5) Enter the various items indicated below on the [OTA file configuration] page.

1. In [Devices to update], select the IoT device name that was set in QE for OTA in 6.2.3(6).
2. In [Select the protocol for file transfer], select “MQTT”.
3. In [Sign and choose your file], select “Sign a new file for me”.
4. In [Code signing profile], select the code signing profile that was created in 6.3.1(3).

   Note: The code signing certificate profile specified here is not used for code signing verification of the firmware of the secondary MCU, so any profile can be specified. The code signing will be written in the RSU file of the update firmware when it is created by the Renesas Image Generator.
5. In [File], select “Upload a new file.”.
6. In [File to upload], click on [Choose file] and select the firmware (.rsu format) that was created in 7.2.1 for use in updating the TB-RX660.
7. In [S3 URL], click on [Browse S3] and select the Amazon S3 bucket that was set in 6.3.1(1).
8. In [Path name of file on device], enter a desired string of characters.
9. In [File type], enter “1”.
10. In [Role], select the service role for the OTA update that was set in 6.3.1(2).

After entering the above, click on [Next].


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Tokyo ▾
nakaki @ 8886-9508-8161 ▾
AWS IoT > Jobs > Create job > OTA job
Step 1
OTA job properties
Step 2
OTA file configuration
Step 3
OTA job configuration

OTA file configuration Info

Devices Info

This OTA update job will send your file securely over MQTT or HTTP to the FreeRTOS-based things and/or the thing groups that you choose.

Devices to update: Choose things and/or thing groups ▾

test_mono ✕

Select the protocol for file transfer

Select the protocol that your device supports.

☑ MQTT

□ HTTP

File Info

Sign and choose your file

Code signing ensures that devices only run code published by trusted authors and that the code hasn't been changed or corrupted since it was signed. You have three options for code signing.

● Sign a new file for me. ○ Choose a previously signed file. ○ Use my custom signed file.

Code signing profile

This profile will contain information needed to create a code signing job. The profile specifies your device's hardware platform, certificate from AWS Certificate Manager, and the location of your code signing certificate path on your device.

Existing code signing profile: [redacted] ▾ Create new profile

File

● Upload a new file. ○ Select an existing file.

File to upload

Choose file

rx23w_tb_2ndota_demo_update.rsu
229376 bytes

File upload location in S3

This is the location in S3 where your file will be stored.

S3 URL: [redacted] ✕ View Create S3 bucket

Browse S3

Format: s3://bucket/prefix/object.

Path name of file on device

This is the name and location where the file will be stored on the FreeRTOS device.

hoge

▼ File type - *optional*

File type

This will be included in the job document so that your devices can identify the type of file received from the AWS cloud.

1

IAM role Info

Role

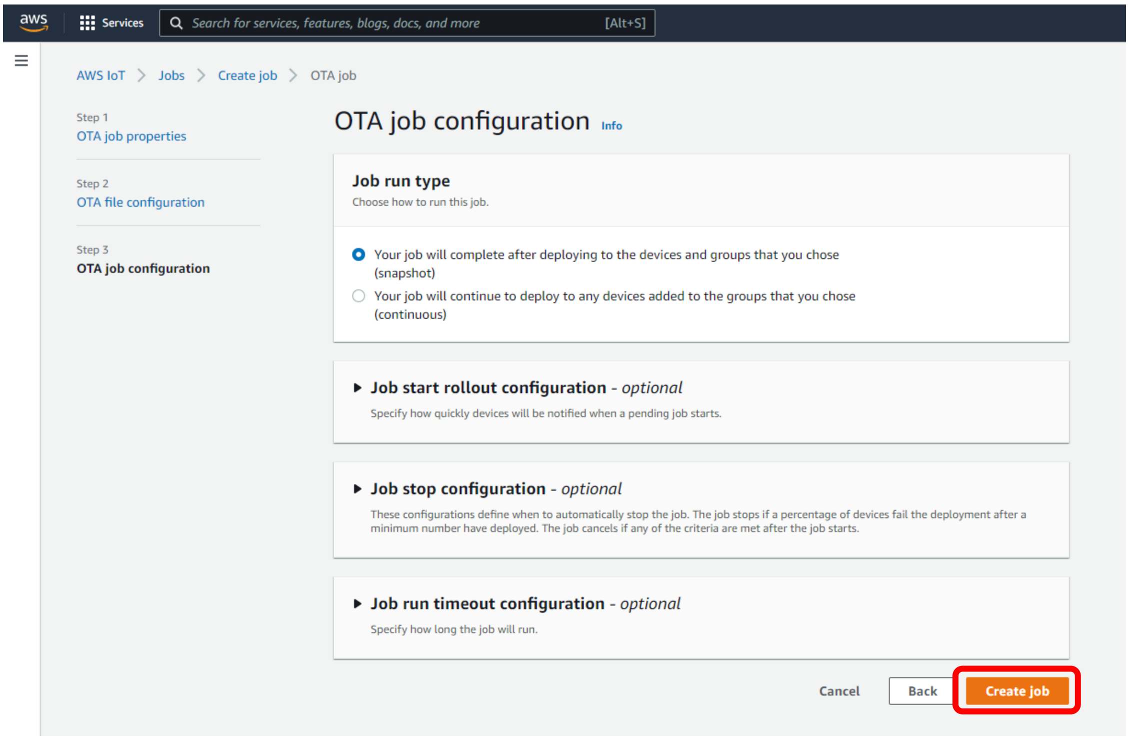
Choose a role that grants AWS IoT access to S3, AWS IoT jobs, and AWS Code signing resources.

[redacted] ▾

Cancel Back Next

Figure 7-8 Page for Setting up the OTA File

(6) Just click on [Create job] on the [OTA job configuration] page as it is not necessary to make any changes.



aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
AWS IoT > Jobs > Create job > OTA job
Step 1
OTA job properties
Step 2
OTA file configuration
Step 3
OTA job configuration

OTA job configuration Info

Job run type

Choose how to run this job.

● Your job will complete after deploying to the devices and groups that you chose (snapshot)

○ Your job will continue to deploy to any devices added to the groups that you chose (continuous)

▸ Job start rollout configuration - *optional*

Specify how quickly devices will be notified when a pending job starts.

▸ Job stop configuration - *optional*

These configurations define when to automatically stop the job. The job stops if a percentage of devices fail the deployment after a minimum number have deployed. The job cancels if any of the criteria are met after the job starts.

▸ Job run timeout configuration - *optional*

Specify how long the job will run.

Figure 7-9 Page for Setting up the OTA Job

An OTA job for the secondary OTA update is created by following the above steps, and the OTA job is delivered to the specified device.

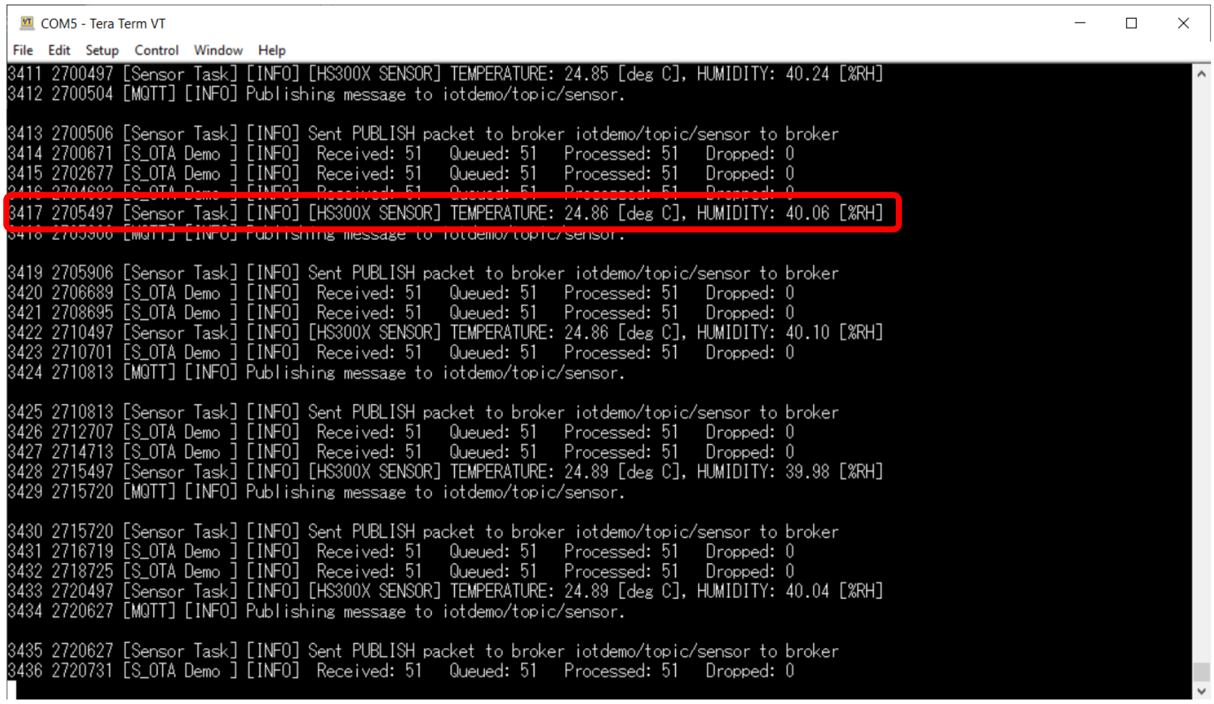
7.2.3 Checking Operation during Execution of the Secondary OTA Update

The OTA update starts within a few seconds after creation of the job. Both the CK-RX65N and TB-RX660 will output logs of the progress of the secondary OTA update.

7.3 Checking Operation after the OTA Update

Figure 7-10 shows the log screen of the CK-RX65N after the update.

You can see that the temperature data are newly displayed in addition to the humidity data acquired from the HS3001 sensor.

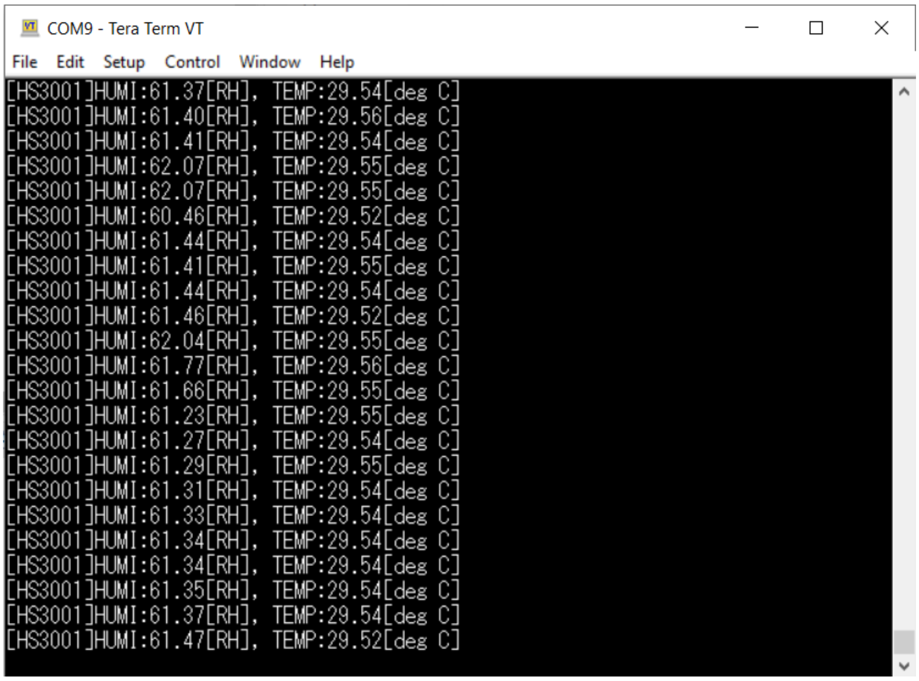


```
3411 2700497 [Sensor Task] [INFO] [HS300X SENSOR] TEMPERATURE: 24.85 [deg C], HUMIDITY: 40.24 [%RH]
3412 2700504 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

3413 2700506 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
3414 2700671 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3415 2702677 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3416 2704683 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3417 2705497 [Sensor Task] [INFO] [HS300X SENSOR] TEMPERATURE: 24.86 [deg C], HUMIDITY: 40.06 [%RH]
3418 2705500 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

3419 2705906 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
3420 2706689 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3421 2708695 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3422 2710497 [Sensor Task] [INFO] [HS300X SENSOR] TEMPERATURE: 24.86 [deg C], HUMIDITY: 40.10 [%RH]
3423 2710701 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3424 2710813 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

3425 2710813 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
3426 2712707 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3427 2714713 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3428 2715497 [Sensor Task] [INFO] [HS300X SENSOR] TEMPERATURE: 24.89 [deg C], HUMIDITY: 39.98 [%RH]
3429 2715720 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

3430 2715720 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
3431 2716719 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3432 2718725 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
3433 2720497 [Sensor Task] [INFO] [HS300X SENSOR] TEMPERATURE: 24.89 [deg C], HUMIDITY: 40.04 [%RH]
3434 2720627 [MQTT] [INFO] Publishing message to iotdemo/topic/sensor.

3435 2720627 [Sensor Task] [INFO] Sent PUBLISH packet to broker iotdemo/topic/sensor to broker
3436 2720731 [S_OTA Demo ] [INFO]  Received: 51   Queued: 51   Processed: 51   Dropped: 0
```

Figure 7-10 Log Screen of the CK-RX65N after the Firmware Update

Next, Figure 7-11 shows the log output for the TB-RX660 after the update. If the firmware update of the TB-RX660 was successful, measured humidity and temperature data are acquired from the HS3001 sensor.

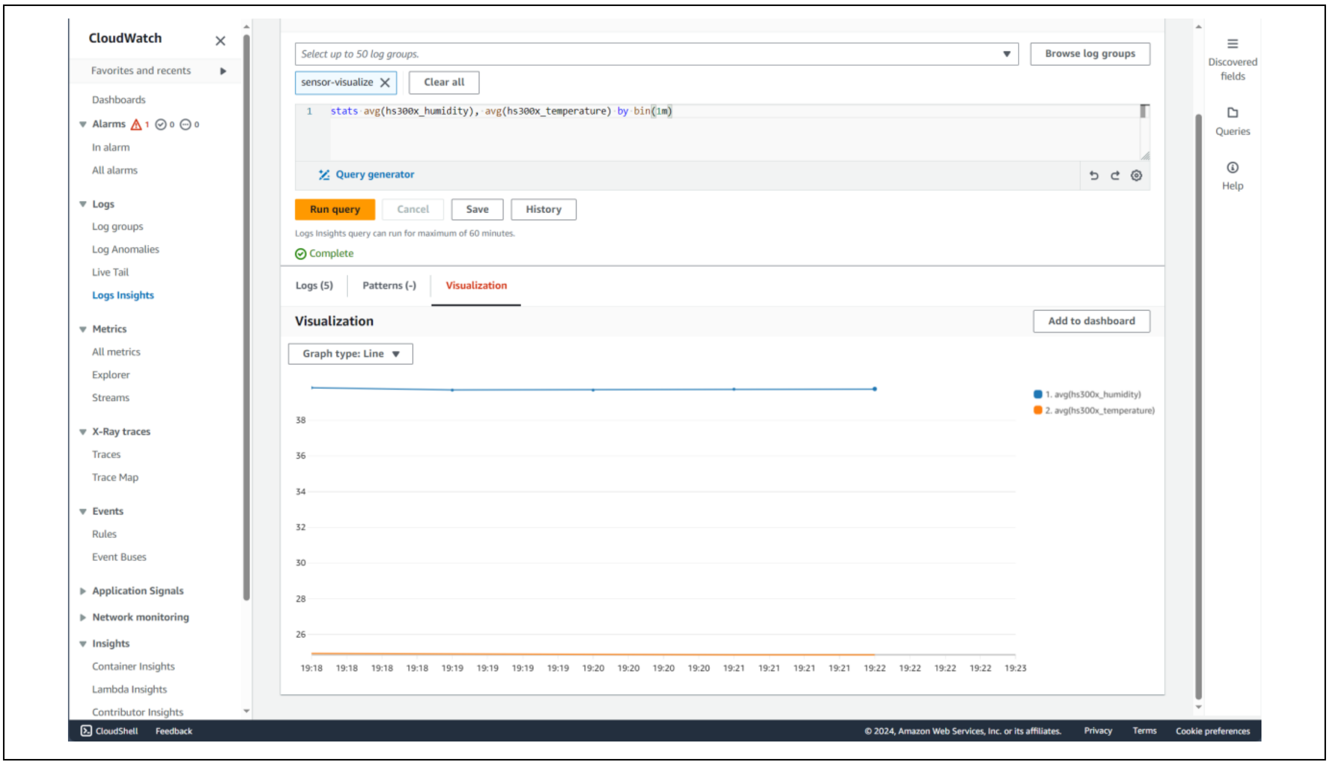
In addition to LED0 that was blinking in the initial state, LED1 will now also be blinking.



```
[HS3001]HUMI:61.37[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.40[RH], TEMP:29.56[deg C]
[HS3001]HUMI:61.41[RH], TEMP:29.54[deg C]
[HS3001]HUMI:62.07[RH], TEMP:29.55[deg C]
[HS3001]HUMI:62.07[RH], TEMP:29.55[deg C]
[HS3001]HUMI:60.46[RH], TEMP:29.52[deg C]
[HS3001]HUMI:61.44[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.41[RH], TEMP:29.55[deg C]
[HS3001]HUMI:61.44[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.46[RH], TEMP:29.52[deg C]
[HS3001]HUMI:62.04[RH], TEMP:29.55[deg C]
[HS3001]HUMI:61.77[RH], TEMP:29.56[deg C]
[HS3001]HUMI:61.66[RH], TEMP:29.55[deg C]
[HS3001]HUMI:61.23[RH], TEMP:29.55[deg C]
[HS3001]HUMI:61.27[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.29[RH], TEMP:29.55[deg C]
[HS3001]HUMI:61.31[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.33[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.34[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.34[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.35[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.37[RH], TEMP:29.54[deg C]
[HS3001]HUMI:61.47[RH], TEMP:29.52[deg C]
```

Figure 7-11 Log Screen of the TB-RX660 after the Firmware Update

Finally, Figure 7-12 shows the display for Amazon CloudWatch. Confirm that the measured temperature and humidity data acquired from the HS3001 sensor are displayed as a graph.



sensor-visualize

```
stats avg(hs300x_humidity), avg(hs300x_temperature) by bin(1m)
```

Logs Insights query can run for maximum of 60 minutes.

Complete

Graph type: Line

Figure 7-12 Graphical Display of Amazon CloudWatch after the Secondary OTA Update

Operations for the demonstration are completed at this point.

8. Precautions

8.1 License Information on the Open-Source Software in Use

The following open-source software is used.

- TinyCrypt Cryptographic Library
  - URL: https://01.org/tinycrypt
  - License: https://github.com/intel/tinycrypt/blob/master/LICENSE

- FreeRTOS
  - URL: https://www.freertos.org/
  - License: FreeRTOS open source licensing, FreeRTOS license description, FreeRTOS license terms and OpenRTOS commercial licensing options.

8.2 Region and User Privileges of AWS for the Demonstration

Regarding the setup of AWS for running the demonstration, notes on the region of use and user privileges are given below.

<Region of use>

This demonstration is provided in the ap-northeast-1 (Asia Pacific (Tokyo)) region of AWS.

If you want to run this demonstration in another region, confirm that the services used in the demonstration are available in that region beforehand.

<User privileges>

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

8.3 Fees for Using AWS

A charge may apply to the cloud resources created and used in the demonstration depending on how AWS is used. To avoid inadvertently incurring charges, deleting the resources created in the cloud after running the demonstration is recommended.

Revision History

Rev.	Date	Description	
		Page	Summary
1.01	Jan. 24, 2022	—	First edition issued.
1.10	Mar. 31, 2022	—	Supported AWS IoT Over-the-air Update Library v3.0.0.
		5 - 8	Added .settings folder to the folder structure of each project.
		5 - 8	Revised package and folder structure for RX65N project.
		9	Updated IDE environment to e2studio 2022-01, Toolchain to CC-RX V3.04.00 and FreeRTOS for RX65N project to Version 2021.07.
		9	Updated code size.
		19 - 20	Updated initial firmware creation method due to RX65N project changes.
		47	Updated screenshot of output log due to RX65N project changes.
		51 - 55	Changed the method of executing an update.
		56	Updated screenshot of output log due to RX65N project changes.
2.00	Mar. 31, 2024	—	The used boards were changed to the CK-RX65N and TB-RX660.
		—	The FreeRTOS package for the projects for the RX65N was updated.
		—	The version of the firmware update module (FWUP) using Firmware Integration Technology (FIT) for the projects for the RX660 was updated.
		—	The entire application note was revised due to changes in the used boards and projects.

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/.