

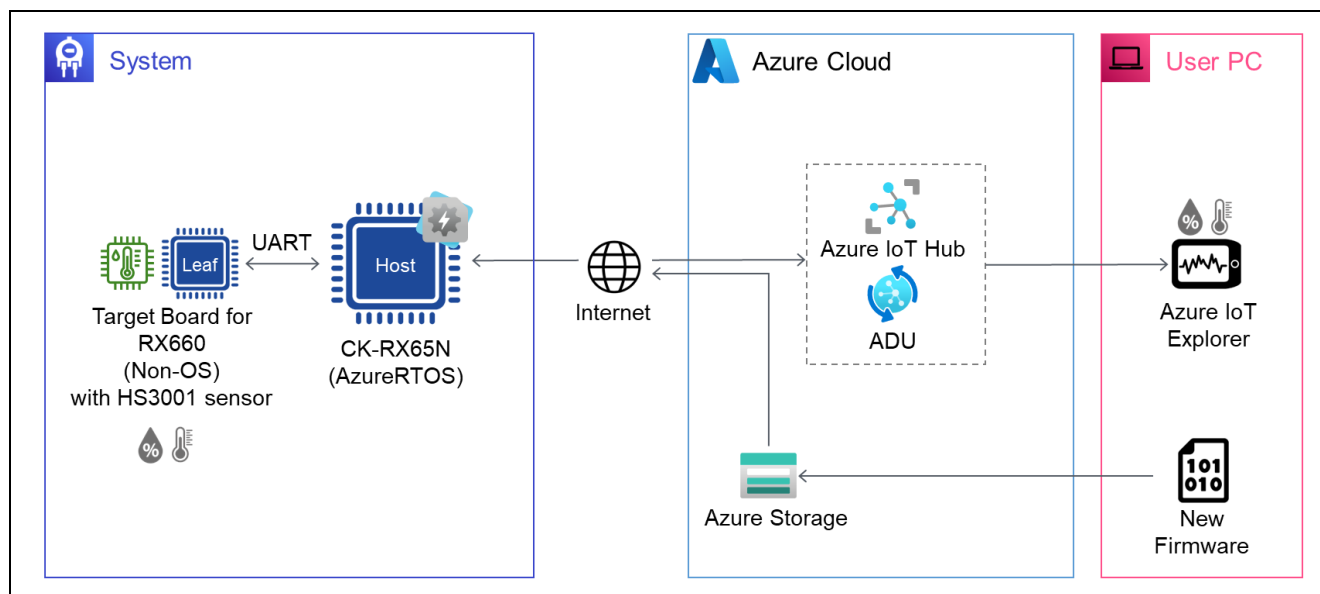
# RX Family

## Sample Code for Secondary OTA Update using Microsoft Azure ADU

### Introduction

This application note uses an RX65N that communicates with Microsoft Azure (hereinafter, Azure) and an RX660 connected to the RX65N via UART as a secondary microcontroller.

This application note describes a demo that uses IoT services provided by Azure to perform an OTA firmware update of the secondary microcontroller.



### Target Device

Host MCU	: RX65N
Secondary MCU	: RX660
Sensor	: HS3001 Relative Humidity and Temperature Sensor (HS3001 sensor)

### Target Board

Host MCU	: CK-RX65N (RTK5CK65N0S04000BE)
Secondary MCU	: Target Board for RX660 (RTK5RX6600C00000BJ)
Sensor	: Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)

### Related Documents

[RX Family How to implement OTA by using Microsoft Azure Services \(R01AN6928\)](#)  
[RX Family Firmware Update Module Using Firmware Integration Technology \(R01AN6850\)](#)  
[RX Family Firmware Update Software Development Guide using AWS/Azure QE for OTA \(R20AN0712\)](#)  
[RX65N Group CK-RX65N v1 User's Manual \(R20UT5100\)](#)  
[RX660 Group Target Board for RX660 User's Manual \(R20UT5068\)](#)

## Contents

1. Overview .....	4
2. Operation Confirmation Condition .....	4
3. Hardware Description .....	5
4. Software Description.....	6
4.1 Sample Program Structure .....	6
4.2 Folder and File Structure .....	7
4.3 Code Size .....	9
5. Demo Operation Description.....	10
6. Demo Setup .....	10
6.1 Hardware Setup.....	10
6.1.1 Overall Structure.....	10
6.1.2 Setup for CK-RX65N .....	11
6.1.3 Setup for TB-RX660 .....	13
6.2 Software Setup .....	17
6.2.1 Advance Preparation .....	17
6.2.2 Terminal Emulator Software Settings.....	17
6.2.3 Creating and Running Initial Firmware for CK-RX65N.....	18
6.2.4 Creating and Running Initial Firmware for TB-RX660.....	21
6.3 Setup for Data Visualization .....	23
6.3.1 Install Azure IoT Explorer .....	23
6.3.2 Configure IoT Plug and Play Settings .....	23
6.3.3 Configure IoT Hub .....	24
6.3.4 Start Receiving Telemetry Data .....	26
7. Procedure for Running the Demo .....	27
7.1 Checking the Initial State of Operation.....	27
7.2 Run OTA Update .....	29
7.2.1 Create Update Firmware .....	29
7.2.1.1 Create Update Firmware for CK-RX65N.....	29
7.2.1.2 Create Update Firmware for TB-RX660.....	29
7.2.2 Create a Manifest File .....	30
7.2.3 Deploying an Update.....	31
7.3 Check Operation during OTA Update Running.....	41
7.4 Check Operation after OTA Update .....	42
7.5 Cleanup of Azure Cloud Resources .....	43
8. Note.....	45
8.1 Software License Information to Use .....	45

8.2	Firmware Update Module Modified Location .....	45
8.3	Limitations/Restrictions of Leaf Version Information .....	45
	Revision History .....	46

Microsoft, Azure are trademarks or registered trademarks of Microsoft Corporation or its affiliates in the United States and/or other countries.

Pmod, PmodUSBUSART is a trademark of Digilent Inc.

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

## 1. Overview

The demonstration shows the operation of the OTA update of the firmware of the RX65N microcontroller connected to the Azure IoT Hub via Ethernet and the secondary microcontroller connected to the RX65N microcontroller via UART (hereinafter, secondary OTA update) using Azure Device Update (ADU), an OTA update service for IoT devices provided by Microsoft.

In this application note, the microcontroller connected to the Azure IoT Hub via Internet is called "Host" and the second microcontroller connected to Host via UART is called "Leaf."

IoT devices are required to fix security vulnerabilities as appropriate and update their functions according to customer requests. By implementing the secondary OTA update in addition to the OTA update of the Host that has been provided in the past, it is possible to realize product development that can respond to vulnerabilities in the Leaf and update flexible services.

## 2. Operation Confirmation Condition

The sample application has been confirmed to operate correctly in the following environment.

**Table 2-1 Demo Operation Confirmation Conditions (RX65N)**

Item	Description
MCU	<a href="#">RX65N (R5F565NEHDFB)</a>
MCU board	<a href="#">CK-RX65N (RTK5CK65N0S04000BE)</a>
Operating voltage	3.3V
RTOS	Azure RTOS 6.2.1_rel-rx-1.2.0
IDE	<a href="#">e<sup>2</sup> studio 2023-07</a> <a href="#">QE for OTA v1.10</a>
C Compiler	<a href="#">C/C++ Compiler Package for RX Family [CC-RX] v3.05.00</a> GCC for Renesas RX 8.3.0.202202
Firmware programming tool	<a href="#">Renesas Flash Programmer V3.12.00</a>

**Table 2-2 Demo Operation Confirmation Conditions (RX660)**

Item	Description
MCU	<a href="#">RX660 (R5F56609BDFP)</a>
MCU board	<a href="#">Target Board for RX660 (RTK5RX6600C00000BJ)</a>
Operating voltage	3.3V
IDE	<a href="#">e<sup>2</sup> studio 2023-07</a>
C Compiler	<a href="#">C/C++ Compiler Package for RX Family [CC-RX] v3.05.00</a> GCC for Renesas RX 8.3.0.202202
Firmware programming tool	<a href="#">Renesas Flash Programmer V3.12.00</a>
MOT file converter	Renesas Image generator (Included in the RX660 project)
USB-UART converter	<a href="#">PmodUSBUSART™</a>

**Table 2-3 Demo Operation Confirmation Conditions (Sensor Board)**

Item	Description
Humidity and temperature sensor board	<a href="#">US082-HS3001EVZ Board</a>

**Table 2-4 Demo Operation Confirmation Conditions (Others)**

Item	Description
<a href="#">Python</a>	3.10.4
<a href="#">Azure IoT Explorer (preview)</a>	0.15.8

### 3. Hardware Description

This system uses CK-RX65N with RX65N as Host and Target Board for RX660 (hereinafter, TB-RX660) with RX660 as Leaf.

In this demonstration, secondary OTA update ([proxy updates](#)) of the Leaf to which the sensor is connected and sensor data acquired by the sensor board can be uploaded to the cloud via UART communication between the microcontrollers.

Figure 3-1 shows the system configuration.

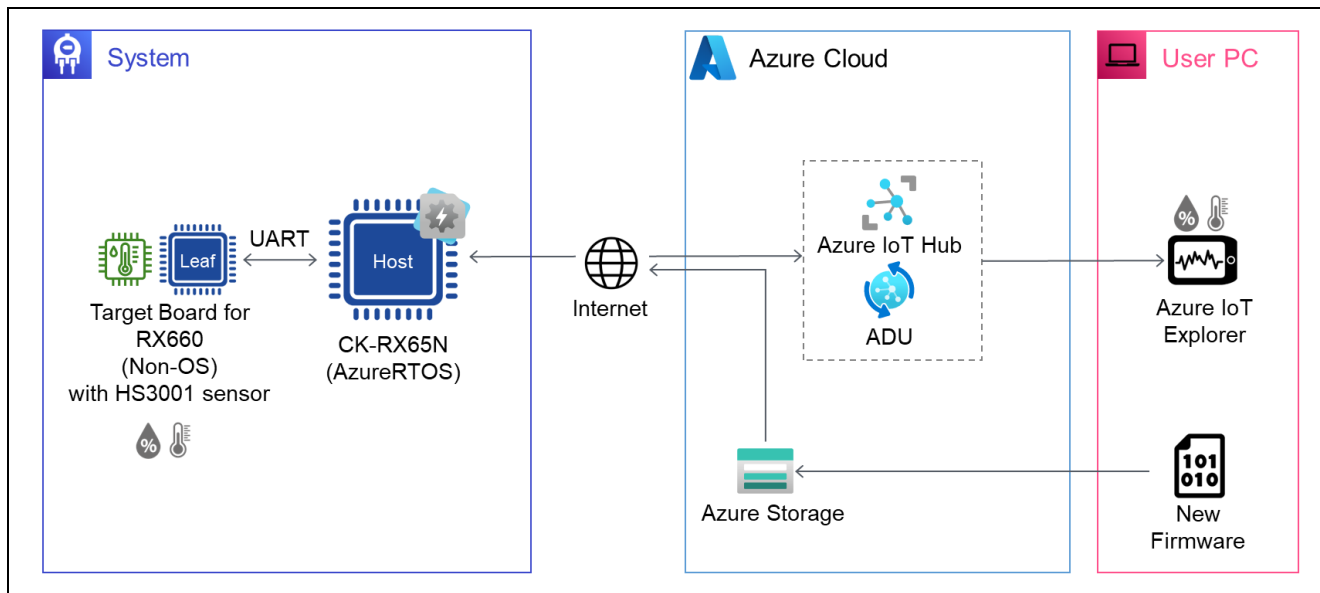


Figure 3-1 System Configuration of Demo

## 4. Software Description

The firmware of the RX65N on the host side is implemented with Azure RTOS.

Therefore, it is possible to perform OTA firmware update and upload data to the cloud via MQTT communication using Azure IoT Hub and Device Update for IoT Hub, which are services for IoT provided in the Azure cloud.

To control the secondary OTA update, the RX65N microcontroller on the Host side uses the [Azure ADU's proxy updates function](#) to send the updated firmware for the RX660 received from Azure to the RX660 microcontroller on the Leaf to achieve the firmware update.

To control the firmware update of the RX660 microcontroller on the Leaf side, use the “[RX Family Firmware Update module Using Firmware Integration Technology Rev.2.00](#)”. Note that the macro values in the source code have been changed to use the Firmware Update Module Rev.2.00 on the RX660, and the operation has been confirmed only with the Happy path in this demo. Please refer to Chapter 8.2 for details.

Regarding uploading sensor data acquired by the sensor board to the cloud and visualization of sensor data, this is achieved by referring to “[RX65N Group Visualization of Sensor Data using RX65N Cloud Kit and Azure RTOS](#)”.

### 4.1 Sample Program Structure

The firmware update mechanism of Leaf in this sample program uses the “Linear Mode Partial Update Method” among the methods provided by the firmware update module. For details of this method, please refer to “Linear Mode Partial Update Method” in Chapter 1.3 “Firmware Update Operation” of “[RX Family Firmware Update module Using Firmware Integration Technology Rev.2.00](#)”.

The memory map of this sample program is shown below.

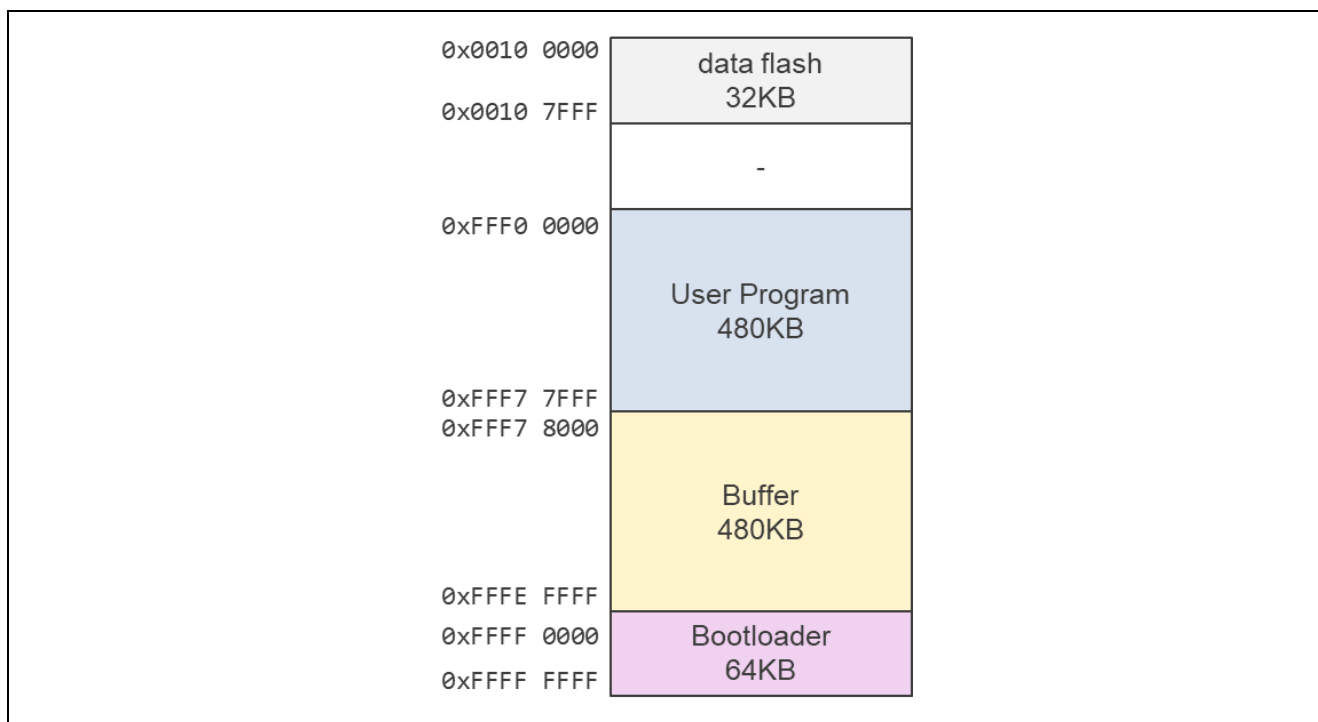


Figure 4-1 The Memory Map of This Sample Program

## 4.2 Folder and File Structure

The following are the folder and file structure.

```
r01an6925xx0100-rx-communication
├─ Demo
│   ├── ccrx
│   │   ├── ck_rx65n_azure_2ndota_demo
│   │   ├── ck_rx65n_demo_bootloader
│   │   ├── rx660_tb_2ndota_demo
│   │   └── rx660_tb_demo_bootloader
│   └── gcc
│       ├── ck_rx65n_azure_2ndota_demo
│       ├── ck_rx65n_demo_bootloader
│       ├── rx660_tb_2ndota_demo
│       └── rx660_tb_demo_bootloader
├─ r01an6925ej0100-rx-communication.pdf
└─ r01an6925jj0100-rx-communication.pdf
```

The ck\_rx65n\_demo\_bootloader folder and ck\_rx65n\_azure\_2ndota\_demo folder contains project files for CK-RX65N.

The rx660\_tb\_demo\_bootloader folder and rx660\_tb\_2ndota\_demo folder contains project files for TB-RX660.

The main file structure of the ck\_rx65n\_azure\_2ndota\_demo folder(CC-RX) is shown below.

```
ck_rx65n_azure_2ndota_demo
├─ .cproject
├─ .project
├─ ck_rx65n_azure_2ndota_demo HardwareDebug.launch
├─ ck_rx65n_azure_2ndota_demo.rcpc
├─ ck_rx65n_azure_2ndota_demo.scfg
├─ libs
├─ model
│   └─ thermostat-4.json
├─ src
│   ├── smc_gen
│   ├── main.c
│   ├── nx_azure_iot_adu_agent_proxy_simulator_driver.c
│   ├── nx_azure_iot_adu_agent_rx_driver.c
│   ├── sample_azure_iot_embedded_sdk_adu.c
│   └─ sample_config.h
└─ tools
    ├── AzureDeviceUpdateScripts
    └─ FlashProjects
```

The main file structure of the rx660\_tb\_2ndota\_demo folder(CC-RX) is shown below.

```
rx660_tb_2ndota_demo
├─.cproject
├─.project
├─rx660_tb_2ndota_demo HardwareDebug.launch
├─rx660_tb_2ndota_demo.rcpc
├─rx660_tb_2ndota_demo.scfg
├─rfp
│   └─rx660_program.rpj
├─RenesasImageGenerator
│   ├──keys
│   └─image-gen.py
└─src
    ├──base64
    ├──cmdresp
    ├──key
    ├──sensor
    ├──smc_gen
    ├──tinycrypt
    └─rx660_tb_2ndota_demo.c
```



### 4.3 Code Size

Code sizes for each project are shown below. The code sizes in the table below are 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 4-1 Code Size Lists (CC-RX)**

Project	ROM	RAM
ck_rx65n_demo_bootloader	35KB	45KB
ck_rx65n_azure_2ndota_demo	570KB	224KB
rx660_tb_demo_bootloader	27KB	13KB
rx660_tb_2ndota_demo	49KB	22KB

#### GCC

Optimization level: Optimize for debug (-Og)

**Table 4-2 Code Size Lists (GCC)**

Project	ROM	RAM
ck_rx65n_demo_bootloader	62KB	47KB
ck_rx65n_azure_2ndota_demo	652KB	234KB
rx660_tb_demo_bootloader	59KB	17KB
rx660_tb_2ndota_demo	91KB	37KB

## 5. Demo Operation Description

- (1) In the initial state of the demo, TB-RX660 only acquires humidity data using the connected HS3001 sensor.
- (2) Using the secondary OTA update mechanism, downloads the updated firmware of TB-RX660 of the Leaf via CK-RX65N from Azure and updates the firmware. At the same time, the firmware update of the CK-RX65N of the Host is also performed.
- (3) After the firmware update, TB-RX660 acquires temperature data in addition to humidity data from the HS3001 sensor.

In the series of steps, the type of sensor data being acquired, and its value can be checked from the log output from both microcontrollers to the PC and Azure IoT Explorer.

## 6. Demo Setup

This section describes the setup required to run the demo in this application note.

Hardware setup such as wiring between CK-RX65N and TB-RX660 and connecting HS3001 sensor, software setup such as creating and writing initial firmware for each microcontroller board, and preparation on the Azure cloud side for executing OTA update and checking sensor data in Azure IoT Explorer are required.

### 6.1 Hardware Setup

#### 6.1.1 Overall Structure

To begin, the overall hardware structure that comprises this demo is shown below. See Figure 6-12 for the actual image after setup. The following sections provide detailed instructions on how to setup each board.

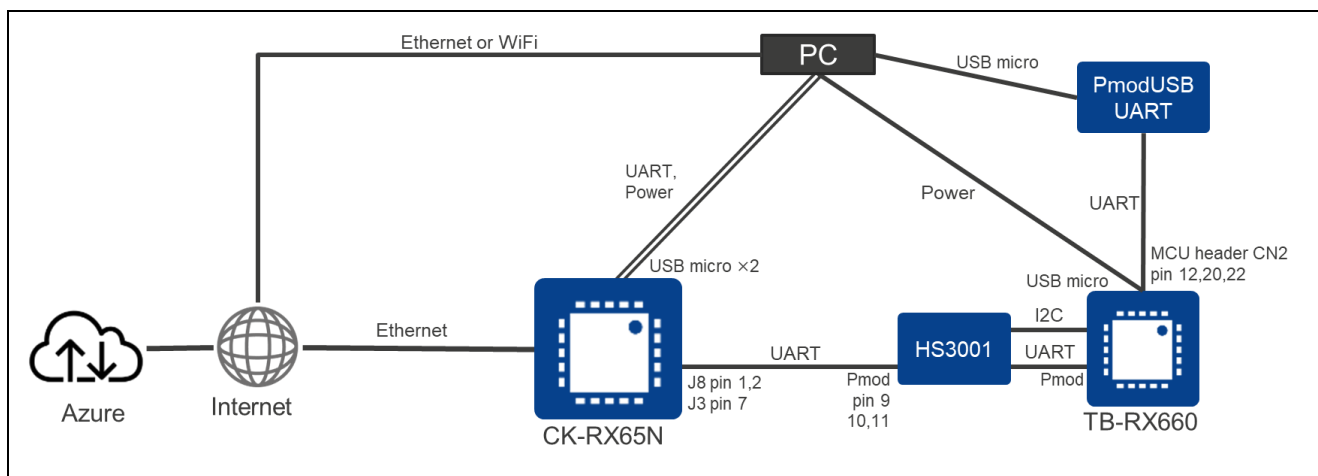


Figure 6-1 Overall Hardware Structure of This Demo

### 6.1.2 Setup for CK-RX65N

The following shows how to setup the CK-RX65N.

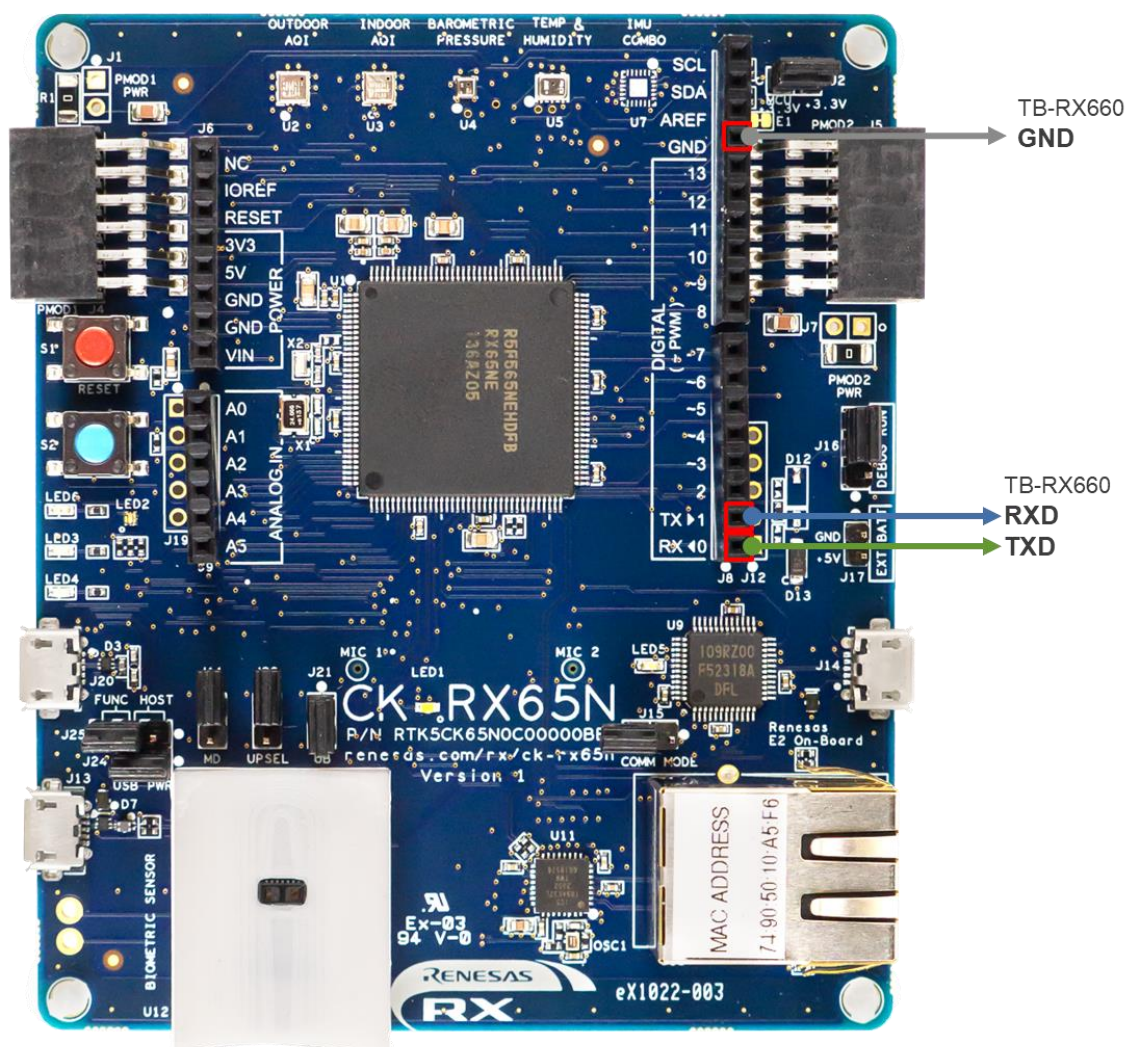
#### (1) Cable Connection for UART Communication with TB-RX660

TXD, RXD and GND for UART communication with TB-RX660 are allocated to the following pins on the J8 and J3 connectors of CK-RX65N. Connect the pins on the TB-RX660 side as shown in 6.1.3(3), with the corresponding UART signals as shown in the table below.

**Table 6-1 UART Connection between CK-RX65N and TB-RX660**

CK-RX65N	(Note1)	HS3001 Pmod I/F		TB-RX660 Pmod I/F
J8 Pin 1: D0/RX (RXD7)	<-->	Pin 9: TXD (Note1)	<-->	Pin 9: TXD9
J8 Pin 2: D1/TX (TXD7)	<-->	Pin 10: RXD (Note1)	<-->	Pin 10: RXD9
J3 Pin 7: GND	<-->	Pin 11: GND	<-->	Pin 11: GND

Note1: Since both pins of the HS3001 sensor board are directly connected, it can handle the input/output signals of the Pmod I/F of the TB-RX660.



**Figure 6-2 Pin Positions in CK-RX65N Used for UART Communication between Microcontrollers**

## (2) Cable Connection for Log Output to PC

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

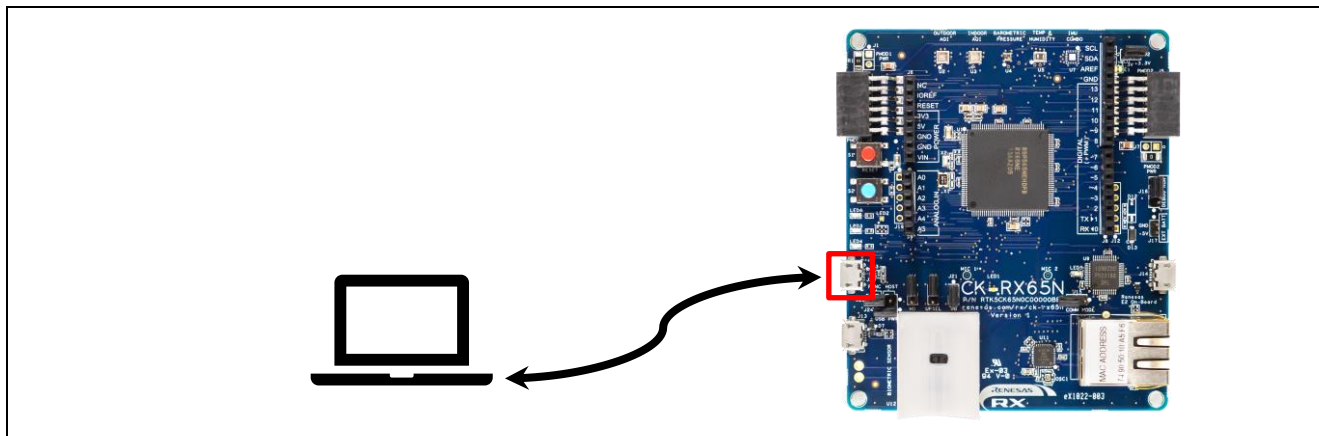


Figure 6-3 Connection for Log Output to PC

## (3) Power Supply and Connection with Debugger

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

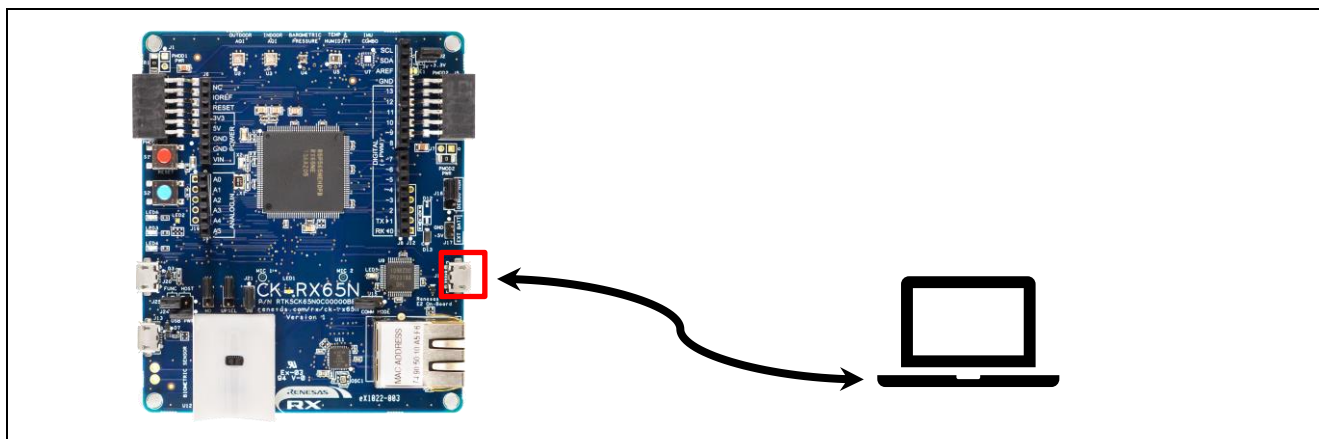


Figure 6-4 Connection for Power Supply and Debugger

## (4) LAN Cable Connection for Internet

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

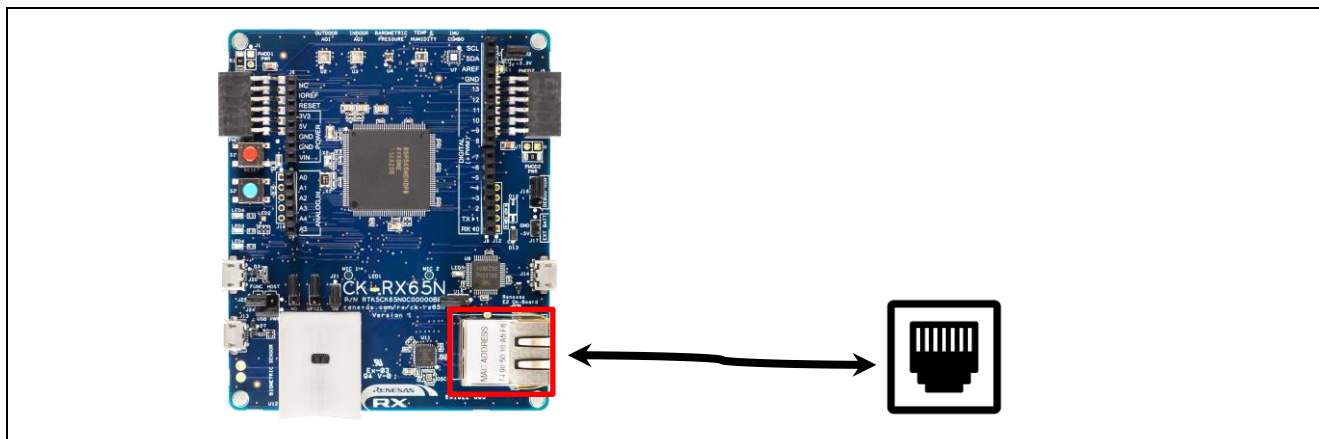


Figure 6-5 Wired Internet connection with Ethernet



**(5) Short-circuit Jumper J16 to DEBUG Side**

To set CK-RX65N to debug mode, short jumper J16 to the DEBUG side (pin 1-2).

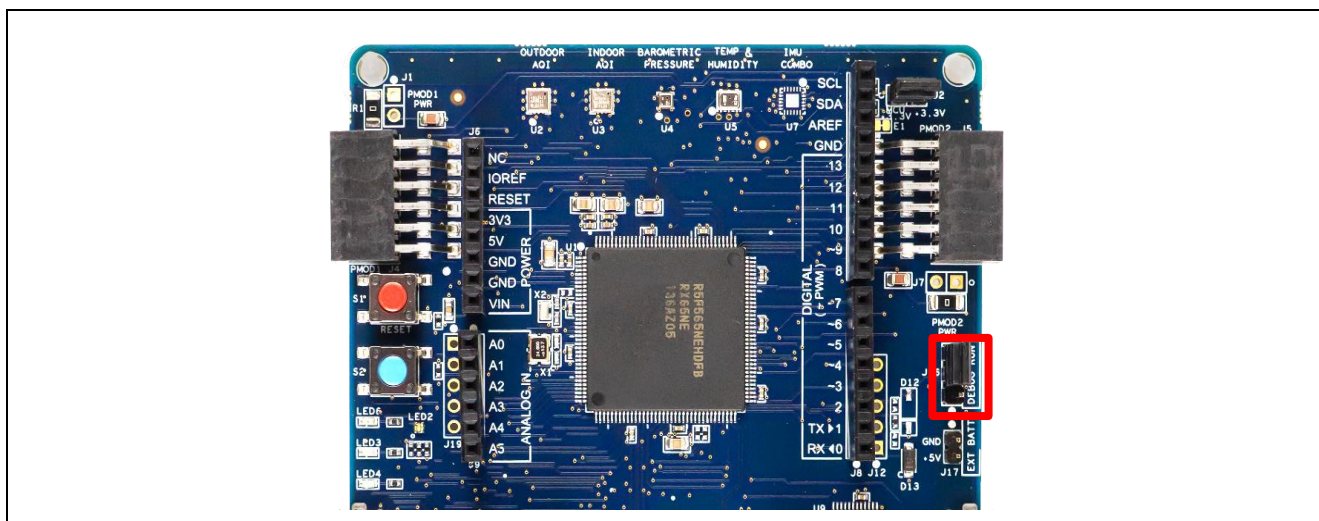


Figure 6-6 Location of Jumper J16

**6.1.3 Setup for TB-RX660**

The following shows how to setup the CK-RX65N.

**(1) Advance Preparation**

Since the board in the shipped condition does not have headers attached to the through-holes, please make the following advance preparations.

- Refer to “5.13 Emulator Reset Header” in the [TB-RX660 User's Manual](#) and attach header pins.
- Refer to “5.14 Power-Supply Selection Header” in the TB-RX660 User's Manual to enable 3.3V voltage supply. In this demonstration, the RX660 is operated at 3.3V.
- Attach CN2 connector to TB-RX660.

**(2) Connection with HS3001 Board**

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

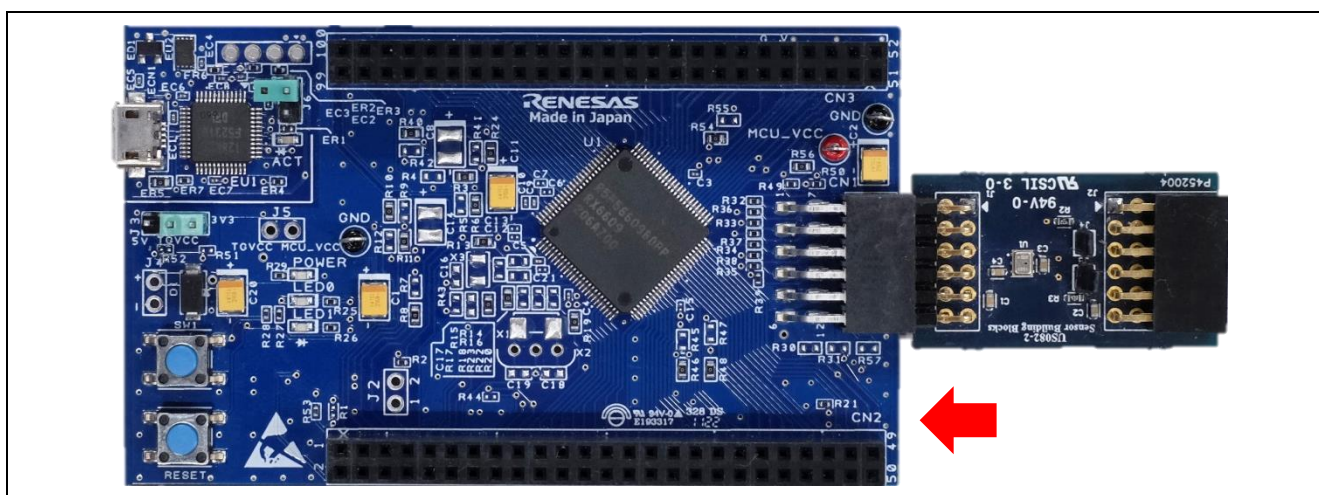
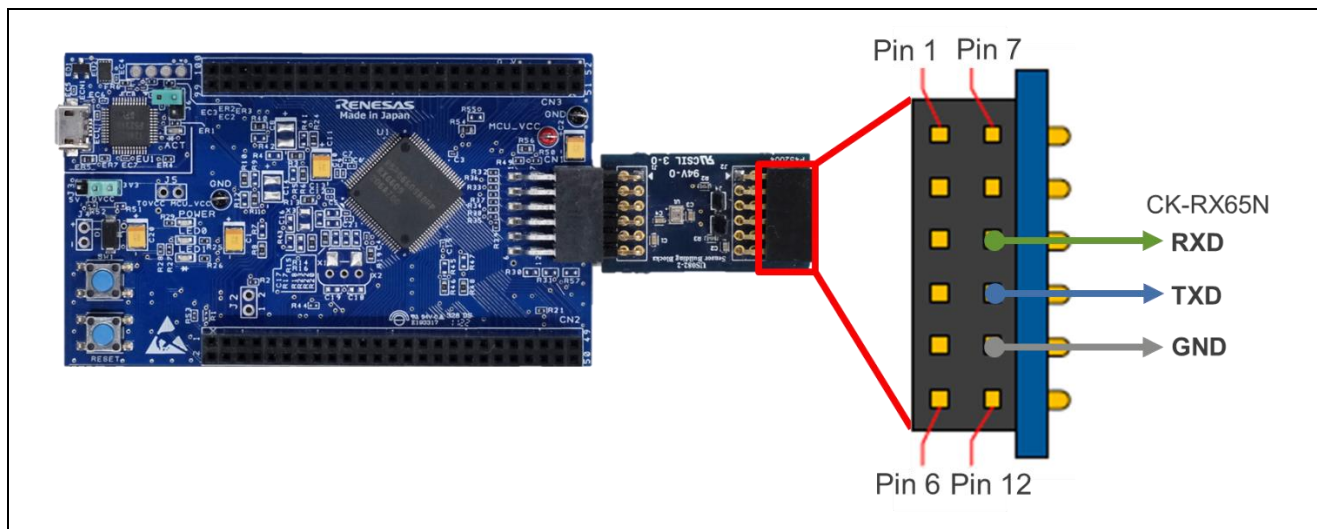


Figure 6-7 Connection between Pmod Connector of TB-RX660 and Sensor Board

**(3) Cable Connection for UART Communication with CK-RX65N**

TXD, RXD and GND for UART communication with CK-RX65N are allocated to the following pins on the Pmod connector of TB-RX660. Connect the pins on the CK-RX65N side as shown in 6.1.2(1), with the corresponding UART signals as shown in the Table 6-1.



**Figure 6-8 Pin Positions Used for UART Communication between Microcontrollers**

**(4) Cable Connection for Log Output to PC**

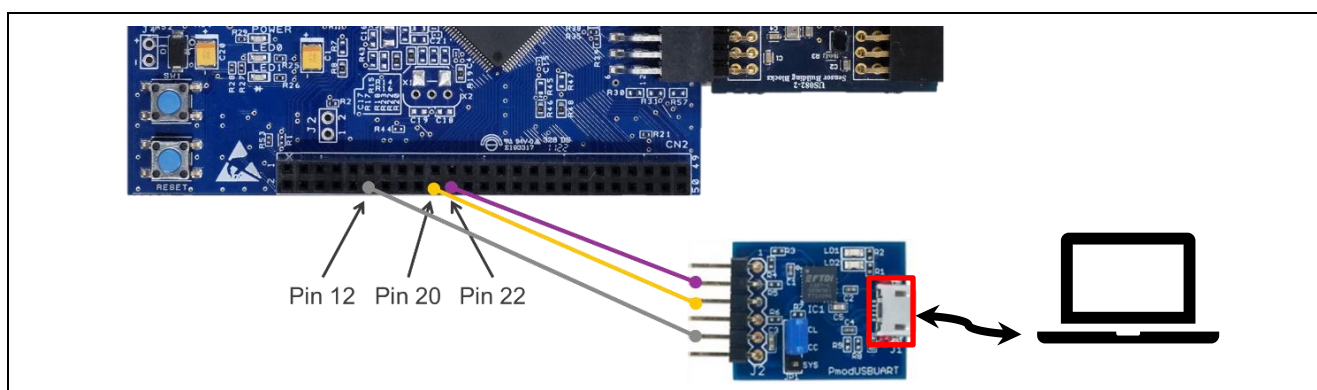
TXD, RXD, and GND for log output to PC are allocated to the following pins on the MCU header CN2 of TB-RX660. Connect the pins of the PmodUSBUART converter with the corresponding UART signals as shown in the table and figure below.

Jumper JP1 on PmodUSBUART should be shorted on the VCC-LCL side. This will make the UART I/F voltage 3.3V with power from the micro USB side.

In addition, connect the micro USB Type-B connector of the PmodUSBUART converter to the PC with a USB cable.

**Table 6-2 Connection of TB-RX660 and PmodUSBUART Converter**

TB-RX660 MCU Header CN2		PmodUSBUART 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 Cable Connection of Serial Communication for Log Output to PC for TB-RX660**



**(5) Cable Connection for Power Supply**

Connect the PC to the micro USB Type-B connector of the RL78/G22 FPB with a USB cable.

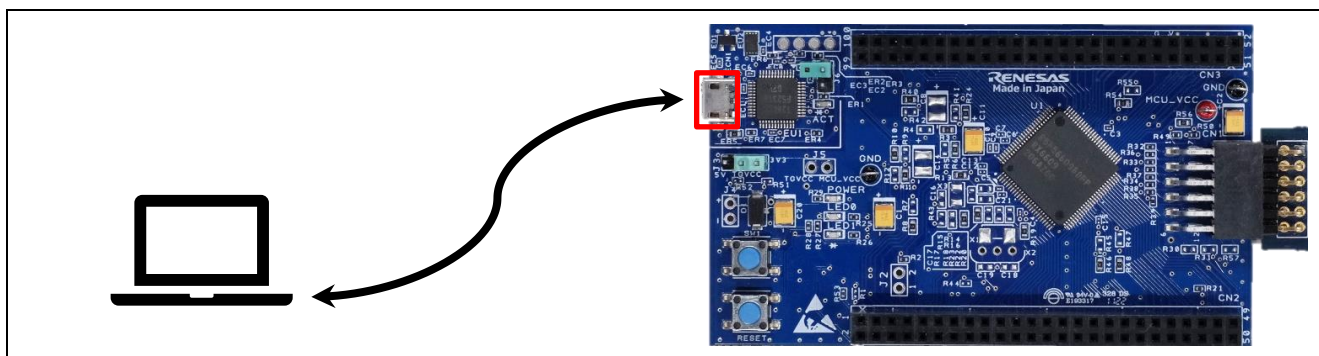


Figure 6-10 USB Connection for Power Supply and Emulator

**(6) Open the Emulator Reset Header (J6)**

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

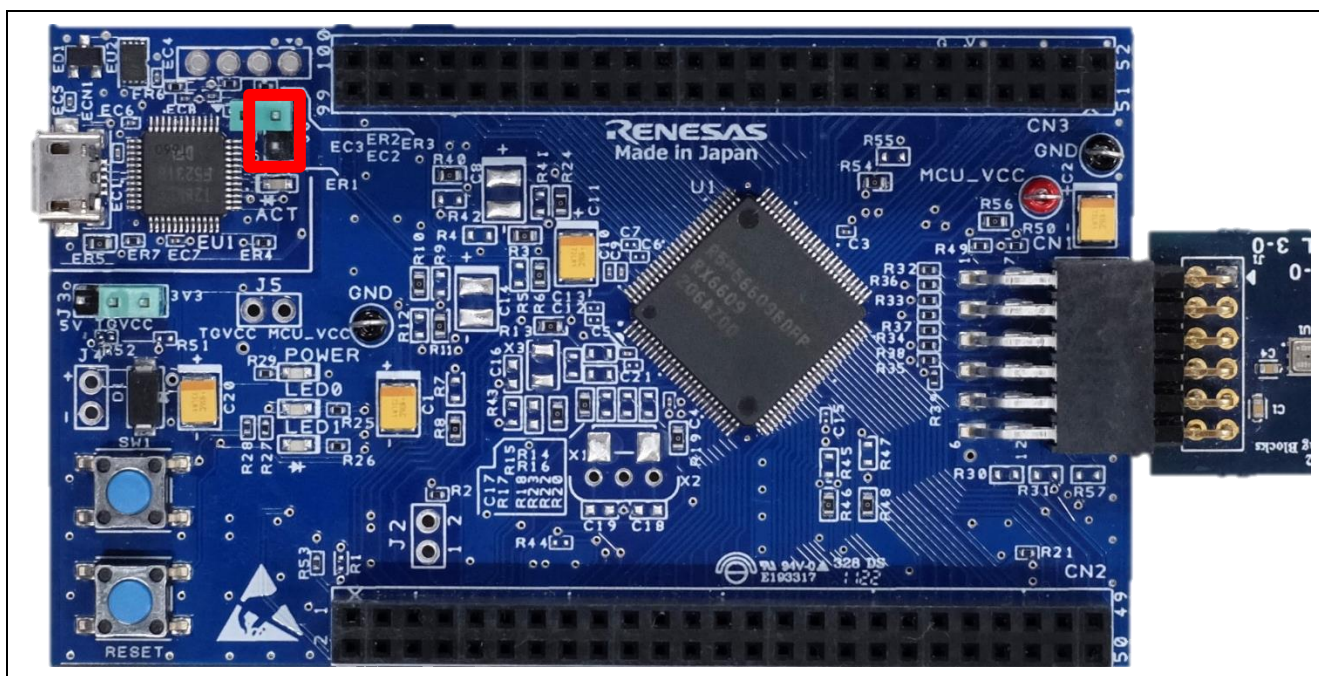


Figure 6-11 Position of the Emulator Reset Header (J6)

That completes the hardware setup for the demonstration. Figure 6-12 shows an overall image of the demonstration configuration.

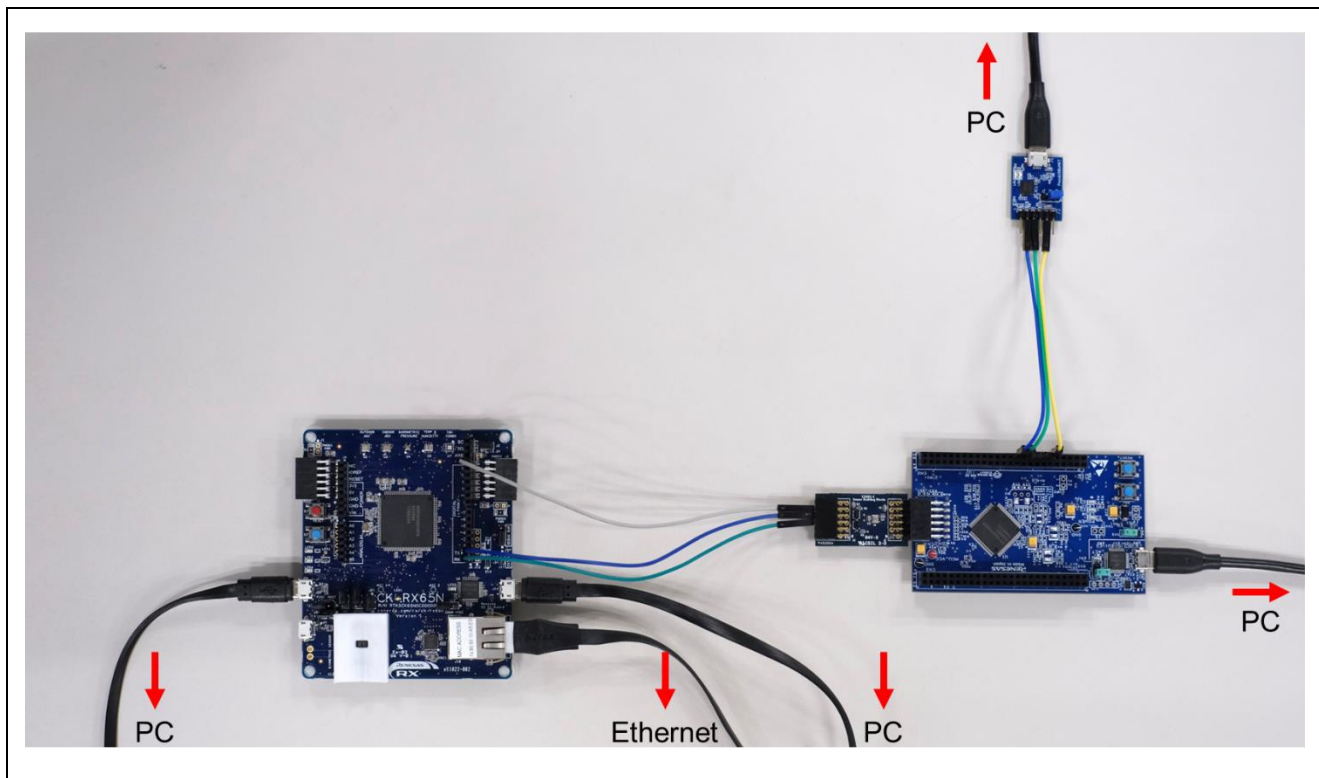


Figure 6-12 Overall Image of the Demo Configuration



## 6.2 Software Setup

The following procedure is performed with the sample project for the CC-RX compiler provided in this application note, but the procedure is the same for the project for GCC.

### 6.2.1 Advance Preparation

Install the software to be used in the demonstration. Please refer to Table 2-4 for the version of each software that has been confirmed to work.

#### (1) Install QE for OTA

Open [Renesas Views] → [Renesas QE] from e<sup>2</sup> studio menu bar and check if QE for OTA is installed. If OTA Main (QE), OTA Manage IoT Device (QE) is available, it has already been installed.

If not, refer to chapter “2.1 Install QE for OTA” in “[RX Family Firmware Update Software Development Guide using AWS/Azure QE for OTA](#)” and install QE for OTA.

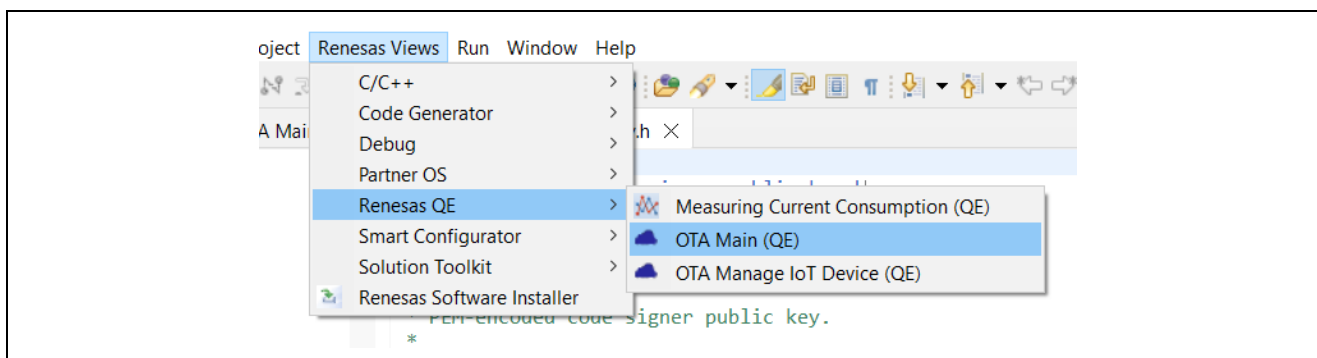


Figure 6-13 Check if QE for OTA is installed

#### (2) Install Python Runtime Environment

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

In addition, we will use the pycryptodome library in Python. After installing Python, run the following pip command to install it.

```
> pip install pycryptodome
```

#### (3) Install Renesas Flash Programmer

Renesas Flash Programmer is available at [Renesas Flash Programmer \(Programming GUI\) | Renesas](#).

### 6.2.2 Terminal Emulator Software Settings

Terminal emulator software (e.g., [Tera Term](#)) is required to generate log output using serial communication. The serial port settings are shown below.

Table 6-3 Serial Port Settings

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

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

Create initial firmware for CK-RX65N using QE for OTA and run debugging in e<sup>2</sup> studio. The procedure is shown below.

#### (1) Import Projects

Import the **ck\_rx65n\_demo\_bootloader** project, a bootloader for the CK-RX65N, and the **ck\_rx65n\_azure\_2ndota\_demo** project, a user program for the CK-RX65N, provided as sample code in this application note, into e<sup>2</sup> studio.

When importing, please uncheck the option "Copy projects into workspace".

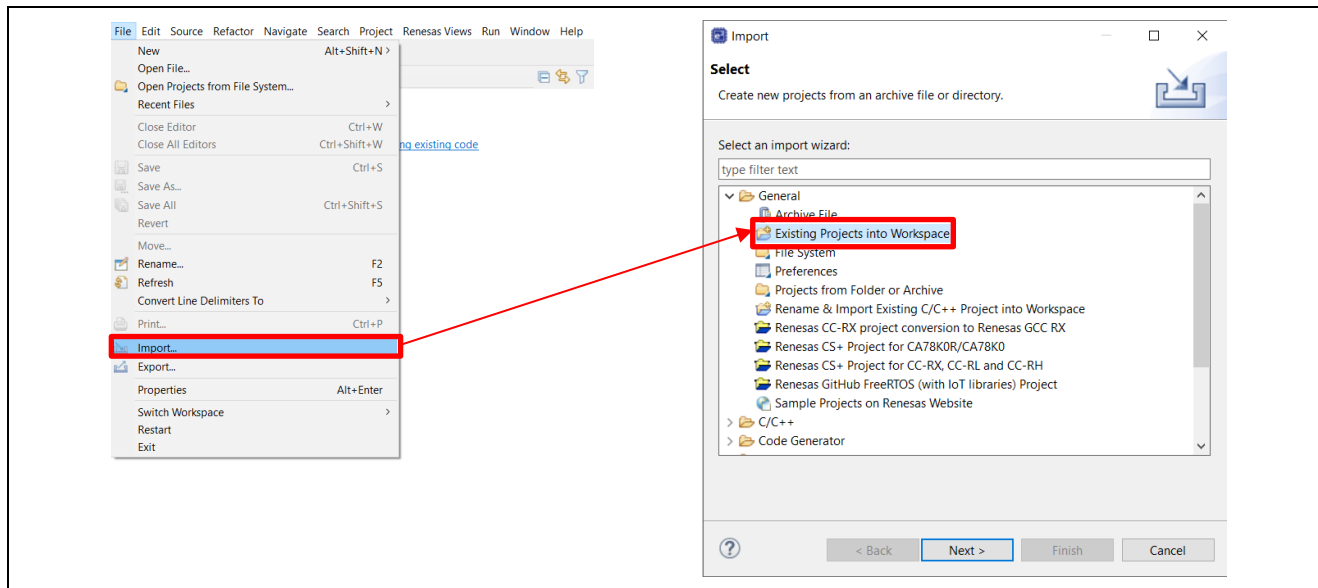


Figure 6-14 Importing Projects (1)

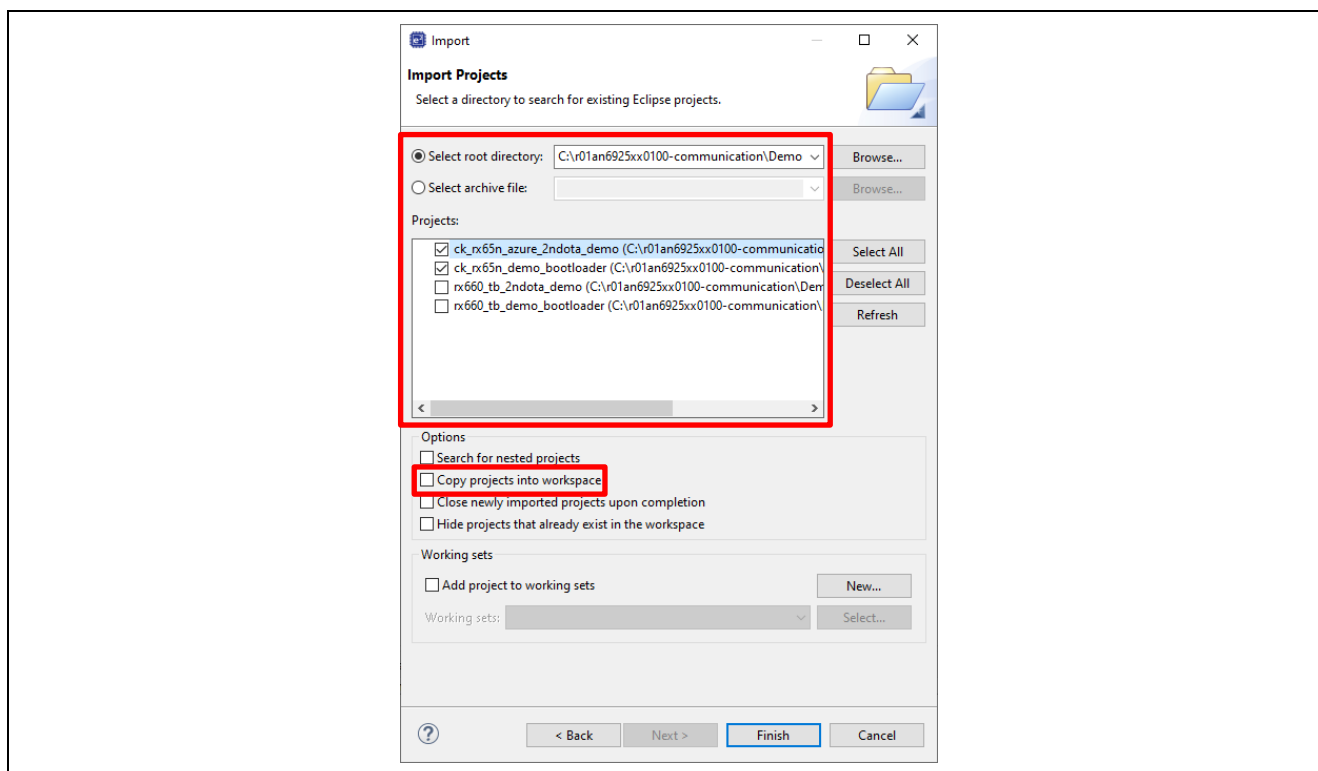
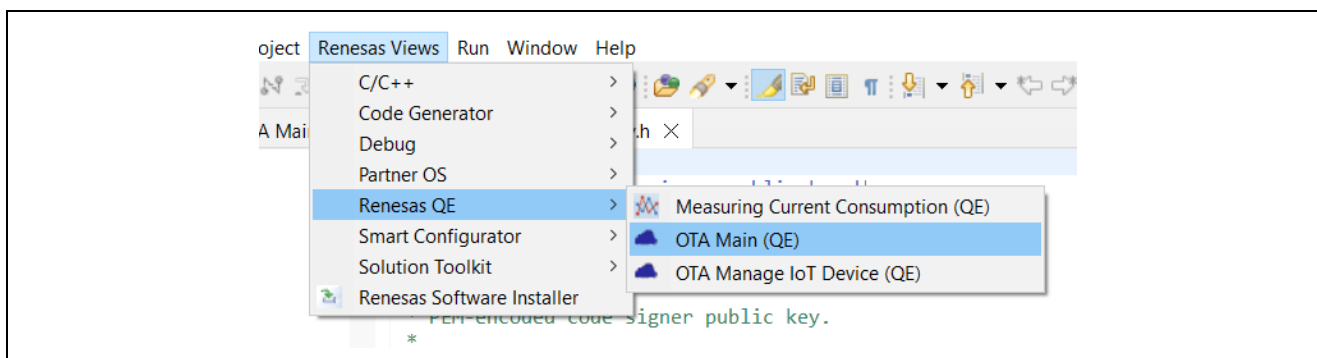


Figure 6-15 Importing Projects (2)

**(2) Open QE for OTA Screen**

From the e<sup>2</sup> studio menu bar, select [Renesas Views] → [Renesas QE] → [OTA Main (QE)].



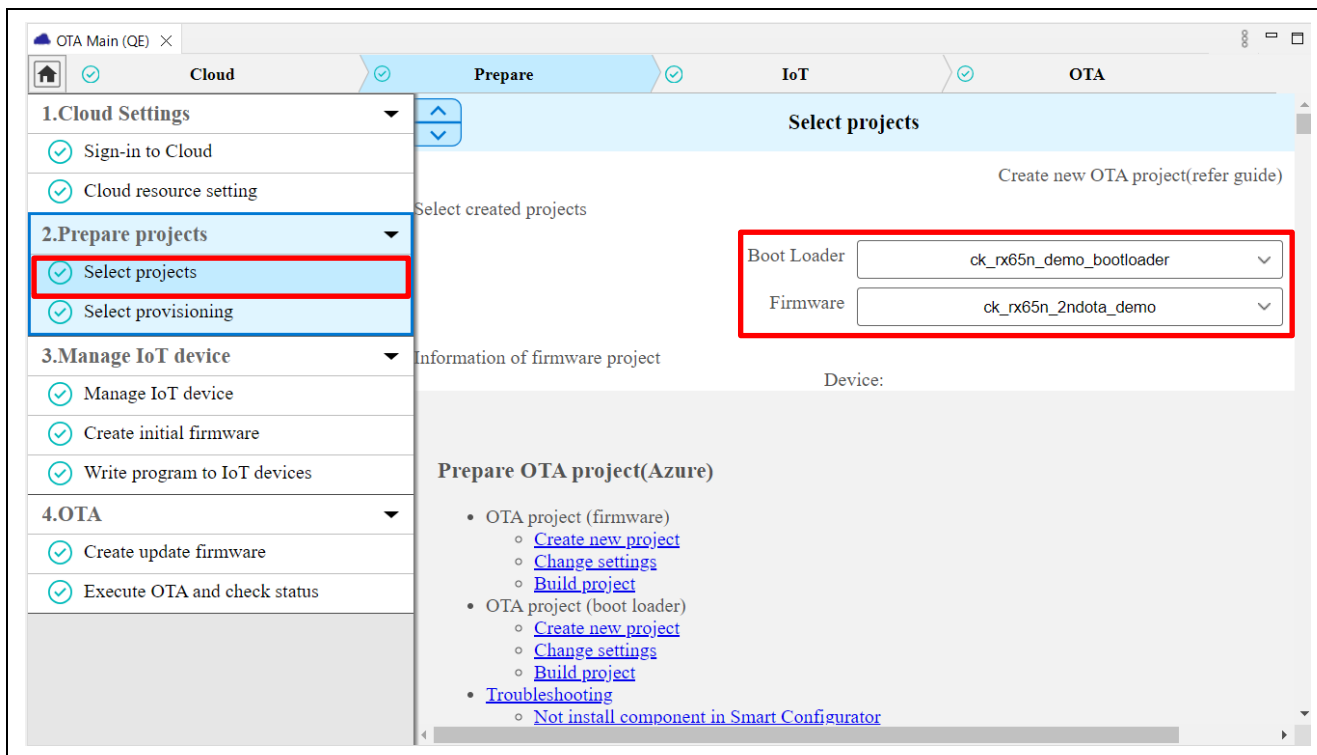
**Figure 6-16 Open QE for OTA Screen**

**(3) [QE for OTA] 1. Cloud Settings**

Execute the steps for Azure in chapter “4.3 Cloud Settings” of the “[RX Family Firmware Update Software Development Guide using AWS/Azure QE for OTA](#)”. This will complete the configuration on the Azure Cloud side.

**(4) [QE for OTA] 2. Prepare projects – Select projects**

Select the ck\_rx65n\_demo\_bootloader project and ck\_rx65n\_azure\_2ndota\_demo project that you just imported into e<sup>2</sup> studio. There is no need to import a new OTA project.



**Figure 6-17 Selecting Projects**

**(5) [QE for OTA] 2. Prepare projects – Select provisioning**

Select "Source code includes credentials (symmetric keys)" as the provisioning method.



**Figure 6-18 Selecting a Provisioning Method**

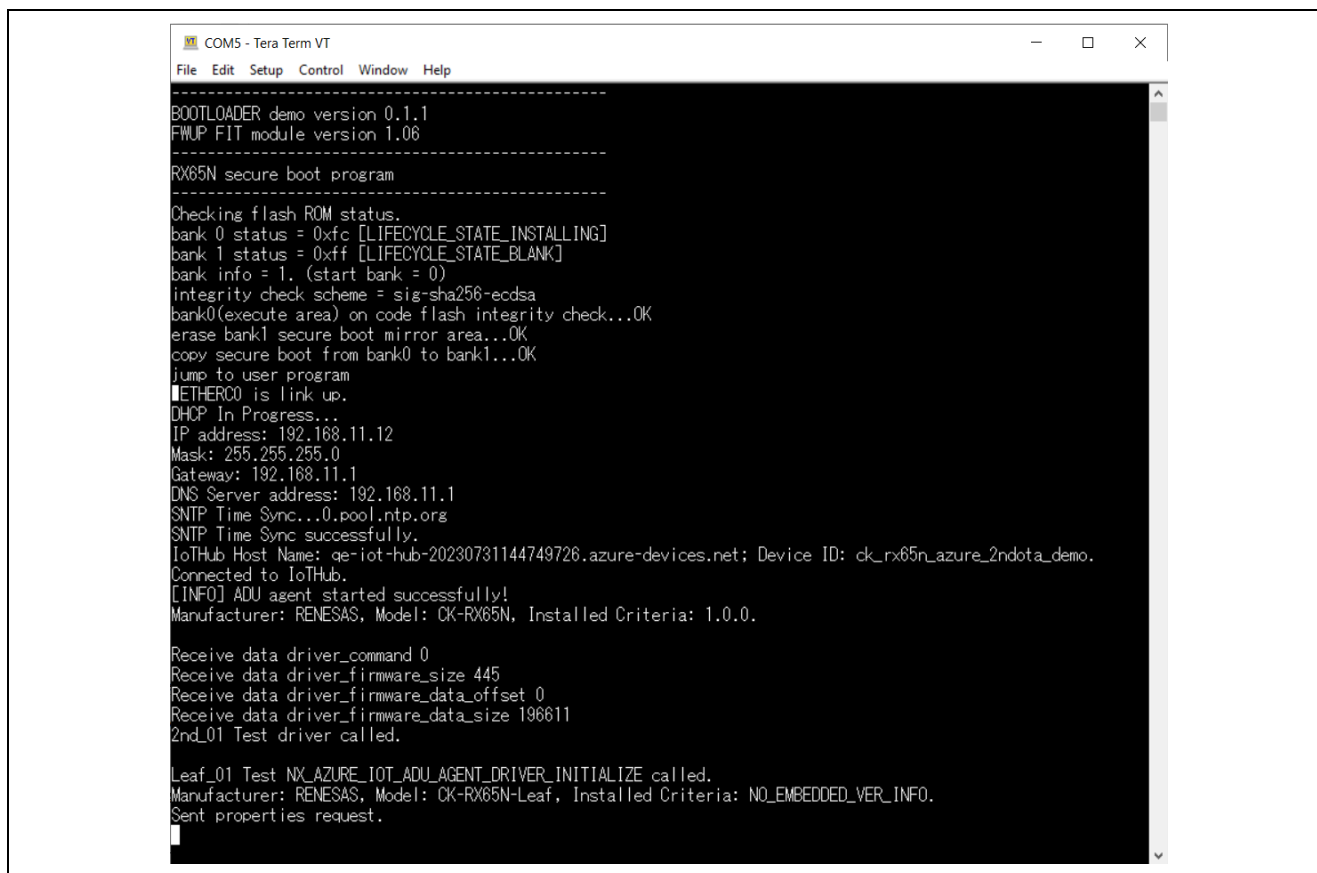
**(6) [QE for OTA] 3. Manage IoT device**

Execute steps for Azure in chapter “4.5 Manage IoT Device” of the [“RX Family Firmware Update Software Development Guide using AWS/Azure QE for OTA”](#). This will complete the registration of the device to the Azure cloud and the writing of the initial firmware for the CK-RX65N.

**(7) Operation Check**

Short-circuit jumper J16 in Figure 6-6 to the RUN side (pin 2-3).

Launch the terminal emulator software, and if the log is output as shown in Figure 6-19, the CK-RX65N is ready.



**Figure 6-19 CK-RX65N Log Output Window**

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

Create the initial firmware for the TB-RX660 and write it with the Renesas Flash Programmer. The procedure is shown below.

#### (1) Import Projects

As we did earlier with the project for the CK-RX65N, import the **rx660\_tb\_demo\_bootloader** project, a bootloader for the TB-RX660, and the **rx660\_tb\_2ndota\_demo** project, a user program for the TB-RX660, provided as sample code in this application note into e<sup>2</sup> studio.

#### (2) Build Projects

Build the rx660\_tb\_demo\_bootloader project and the rx660\_tb\_2ndota\_demo project, and create a MOT file. The MOT file is created in the HardwareDebug folder directly under the project folder.

#### (3) Create Initial Firmware

The initial firmware for TB-RX660 is created by combining the created MOT files of rx660\_tb\_demo\_bootloader and rx660\_tb\_2ndota\_demo. To combine MOT files, use the Renesas Image Generator. Renesas Image Generator is a tool included in the “[RX Family Firmware Update module Using Firmware Integration Technology Rev.2.00](#)”. For details, please refer to the “Renesas Image Generator” chapter in the application note linked 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_2ndota_demo.csv -vt ecdsa
```

#### (4) Write Initial Firmware

Using Renesas Flash Programmer, write the initial firmware **initial\_firm.mot** created above 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 **initial\_firm.mot** and click Start.

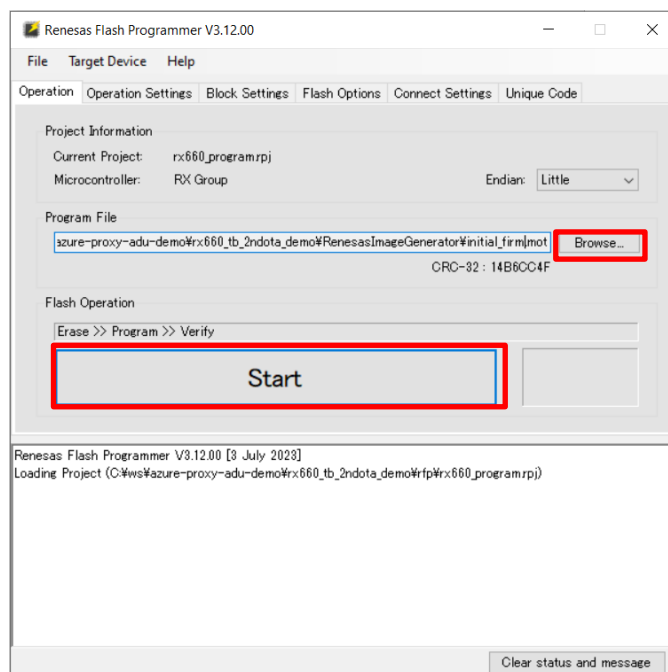


Figure 6-20 Writing Initial Firmware with Renesas Flash Programmer

#### (5) Operation Check

Short-circuit the emulator reset header (J6) of the TB-RX660.

Launch the terminal emulator software, and if the log is output as shown in Figure 6-21, the TB-RX660 is ready.

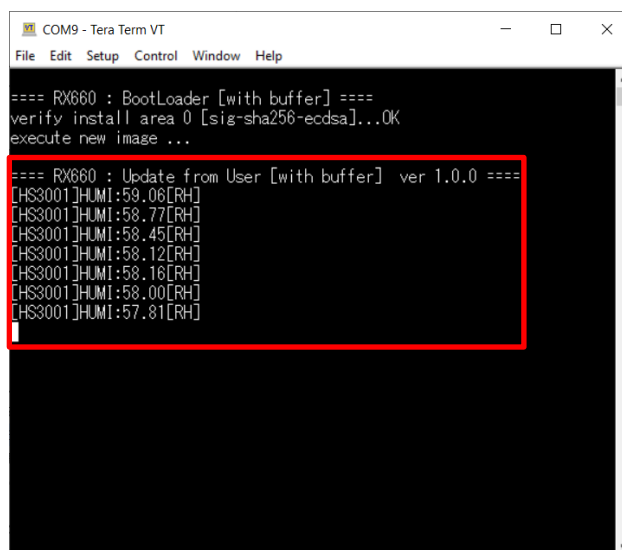


Figure 6-21 TB-RX660 Log Output Window

## 6.3 Setup for Data Visualization

### 6.3.1 Install Azure IoT Explorer

See the link below for instructions on how to install and use Azure IoT Explorer.

[Install and use Azure IoT explorer - Azure IoT | Microsoft Learn](#)

This application note only describes how to check sensor data sent by Host.

Please refer to the following link to obtain Azure IoT Explorer.

[Releases · Azure/azure-iot-explorer \(github.com\)](#)

### 6.3.2 Configure IoT Plug and Play Settings

Launch Azure IoT Explorer.

Select "IoT Plug and Play Settings" from the left menu, then select Add -> Local folder from the top menu.

Click "Pick a folder" and select the ck\_rx65n\_azure\_2ndota\_demo/model folder.

Click "Save" to save the settings.

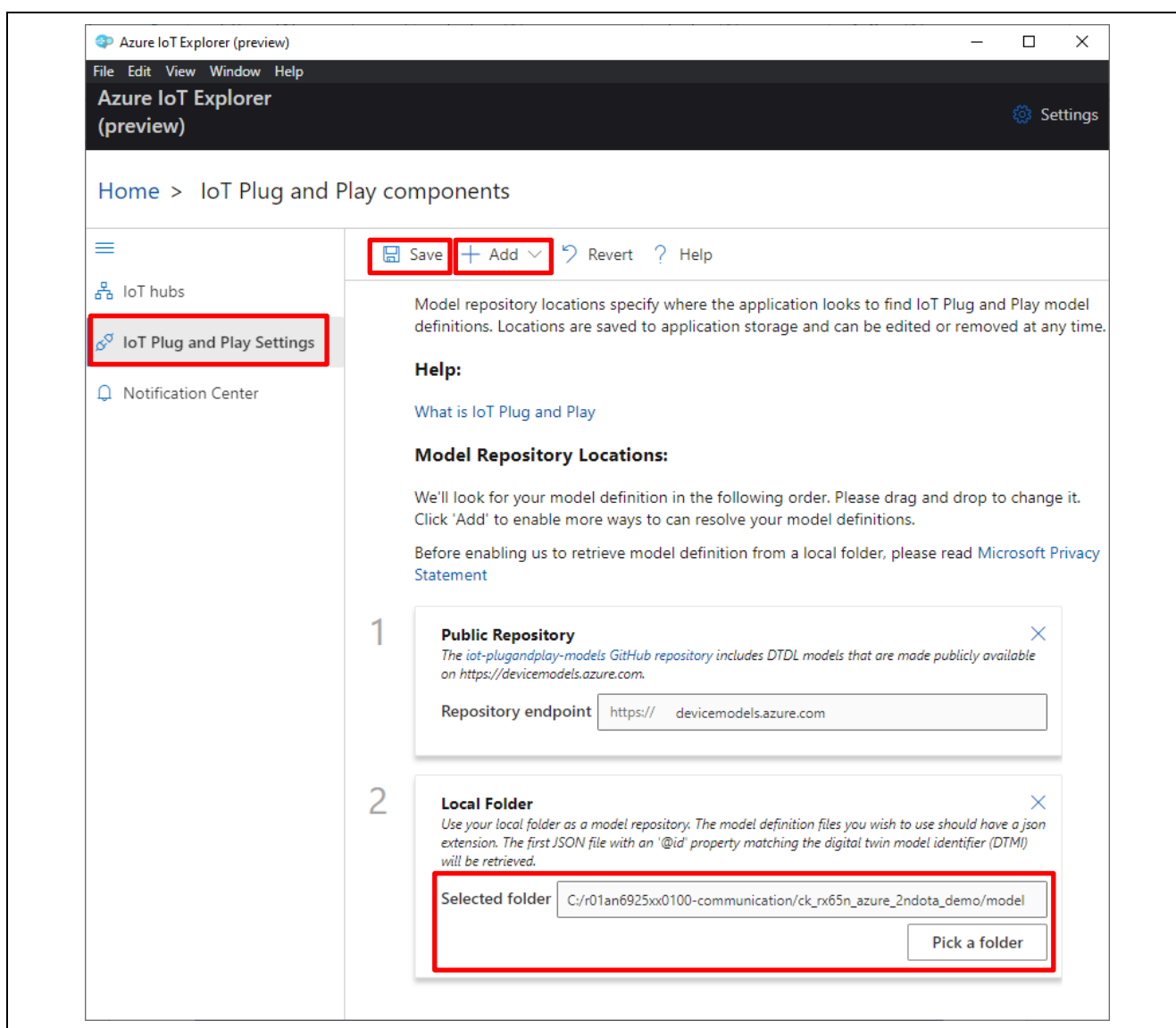


Figure 6-22 Configure IoT Plug and Play Settings

### 6.3.3 Configure IoT Hub

Open the Azure portal in a browser and obtain the primary connection string from the IoT Hub you created.

In the Azure portal, select in the following order.

Azure IoT Hub -> Shared access policies -> iothubowner -> Primary connection string

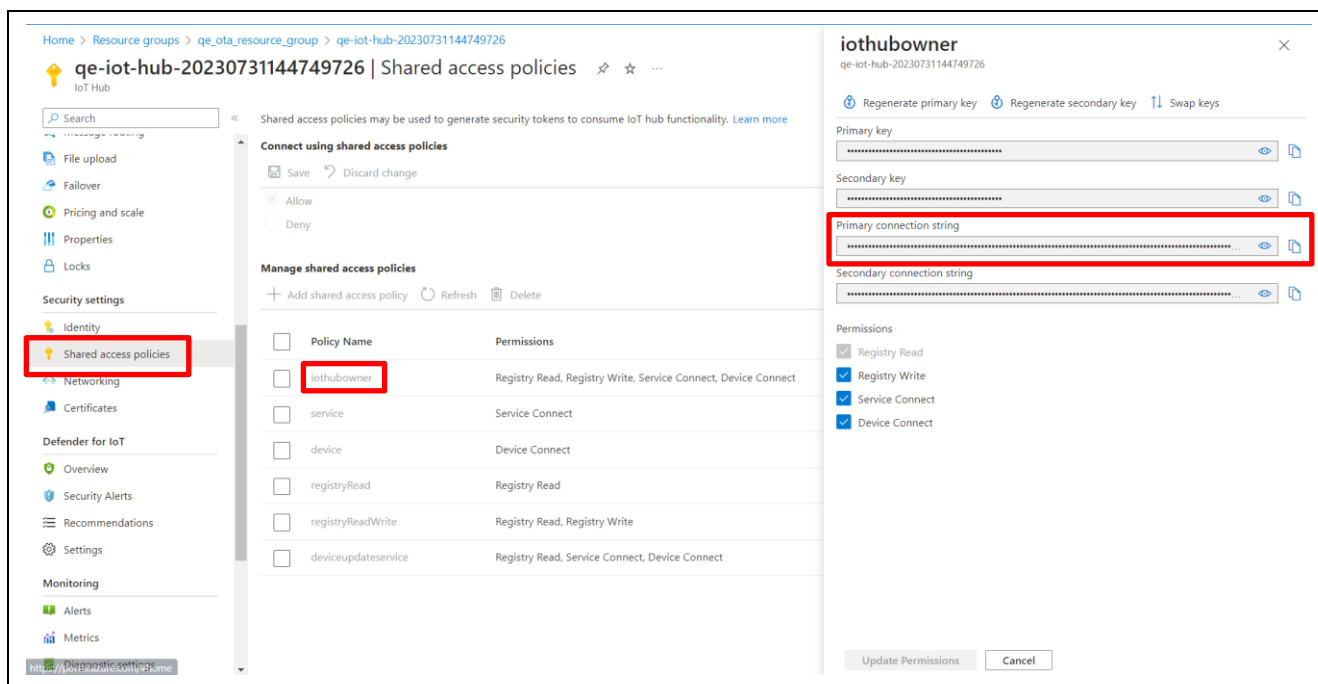


Figure 6-23 Obtaining the Primary Connection String

Select "IoT hubs" from the left side menu of Azure IoT Explorer.

Click "Connect via IoT Hub connection string".

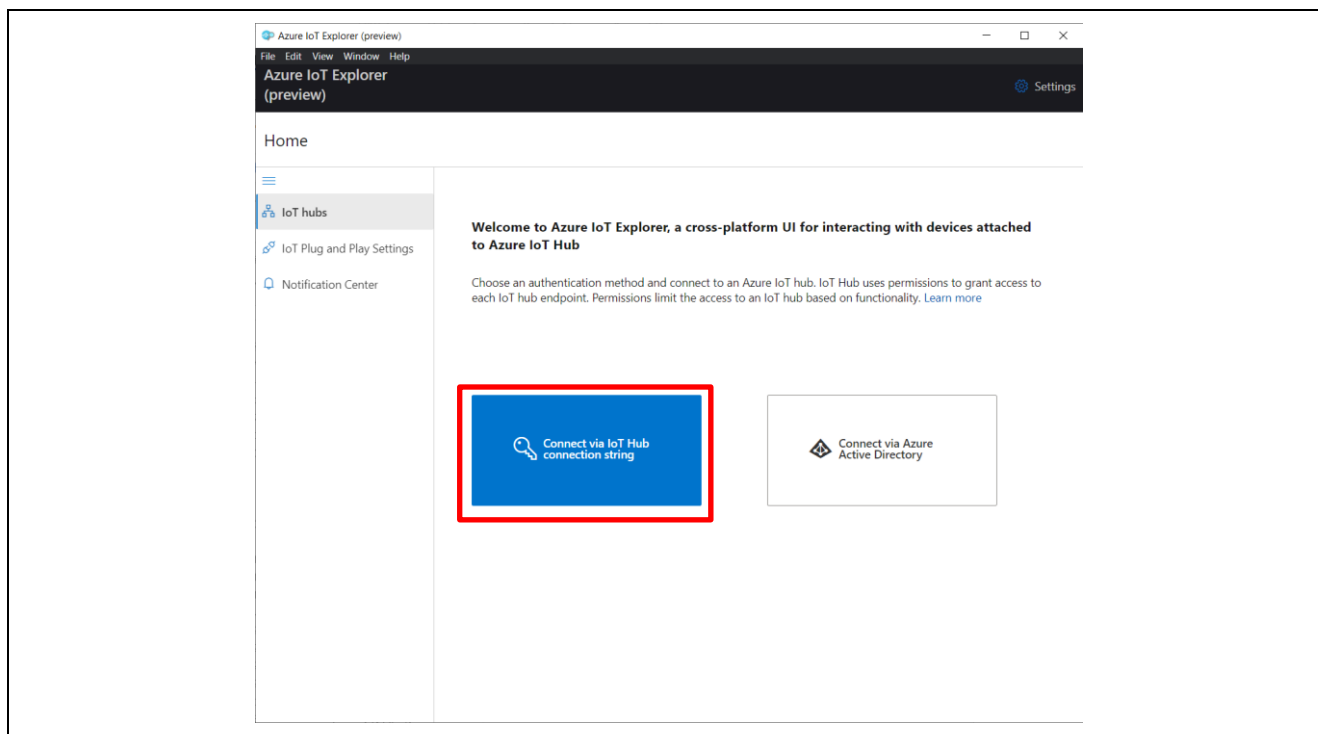


Figure 6-24 Selecting a Connection Method with the IoT Hub



Click “Add connection”.

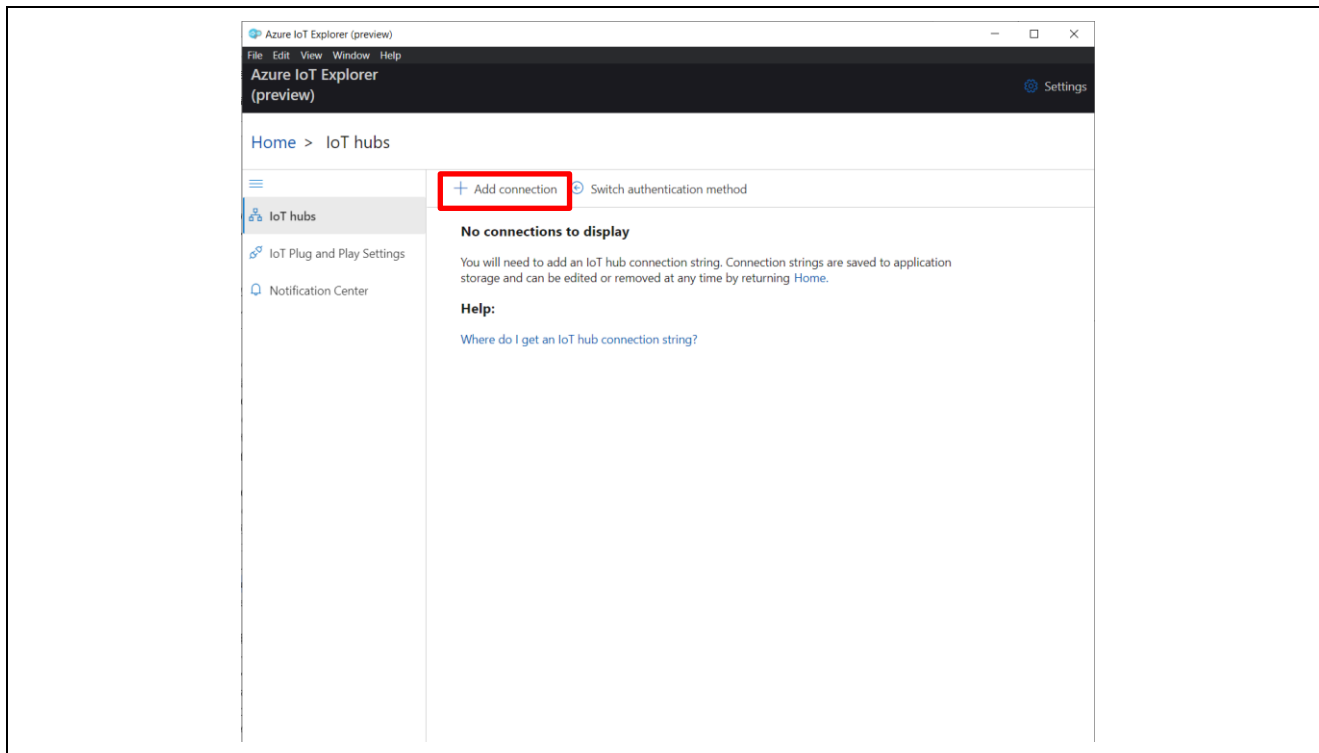


Figure 6-25 Connection settings with IoT Hub

Enter the primary connection string you obtained in “Connection string”.

Click “Save”.

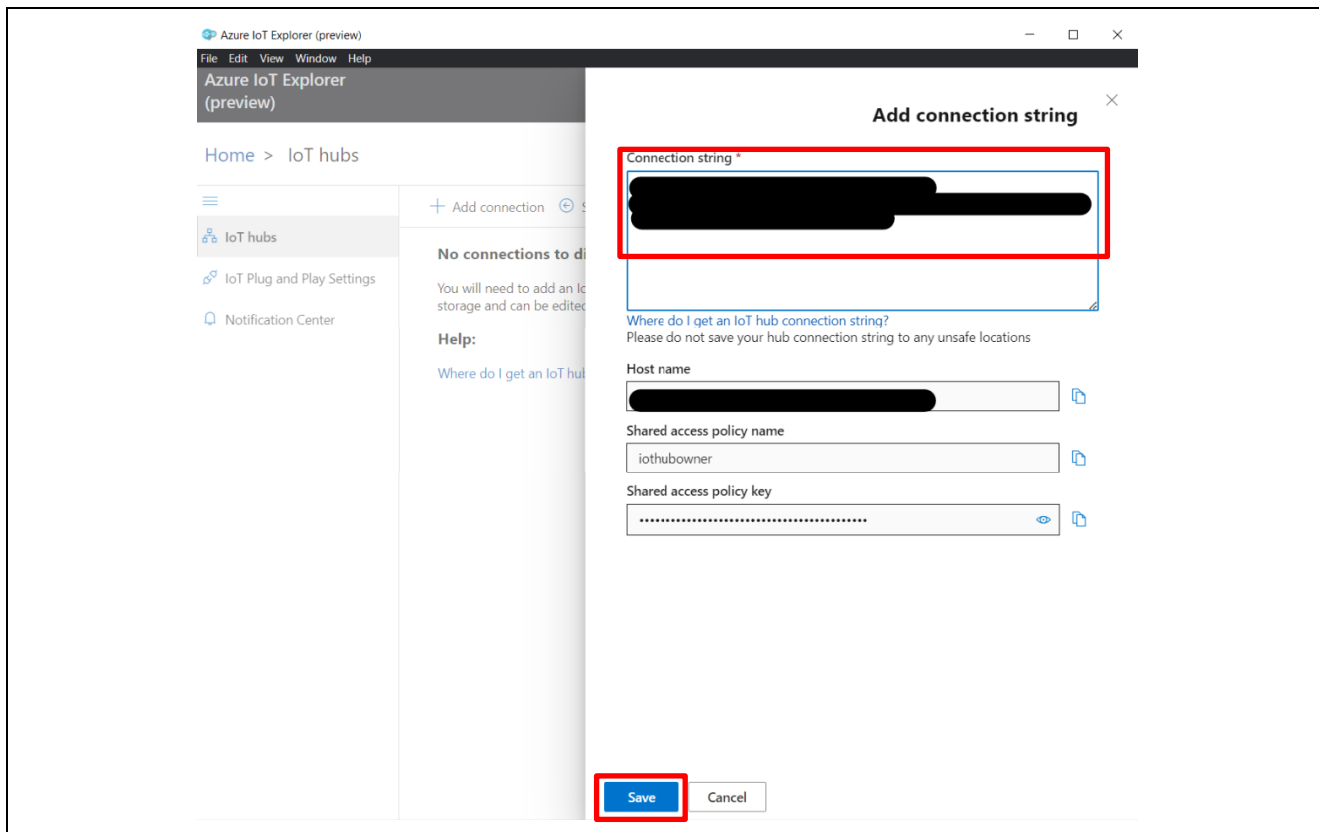


Figure 6-26 Enter Primary Connection String

A list of devices registered in Azure IoT hub will be displayed.

Click the link of the Device ID.

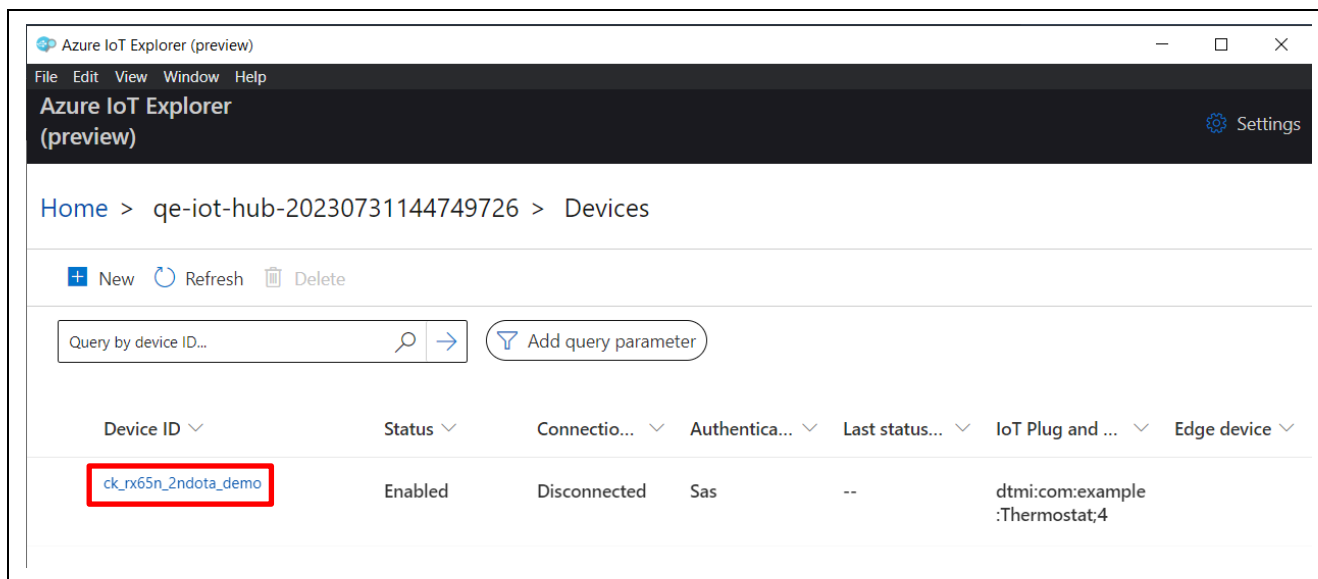


Figure 6-27 IoT Hub Setup Complete.

### 6.3.4 Start Receiving Telemetry Data

Select “Telemetry” from the left menu.

Click “Start” to start displaying the Telemetry data sent from the Host.

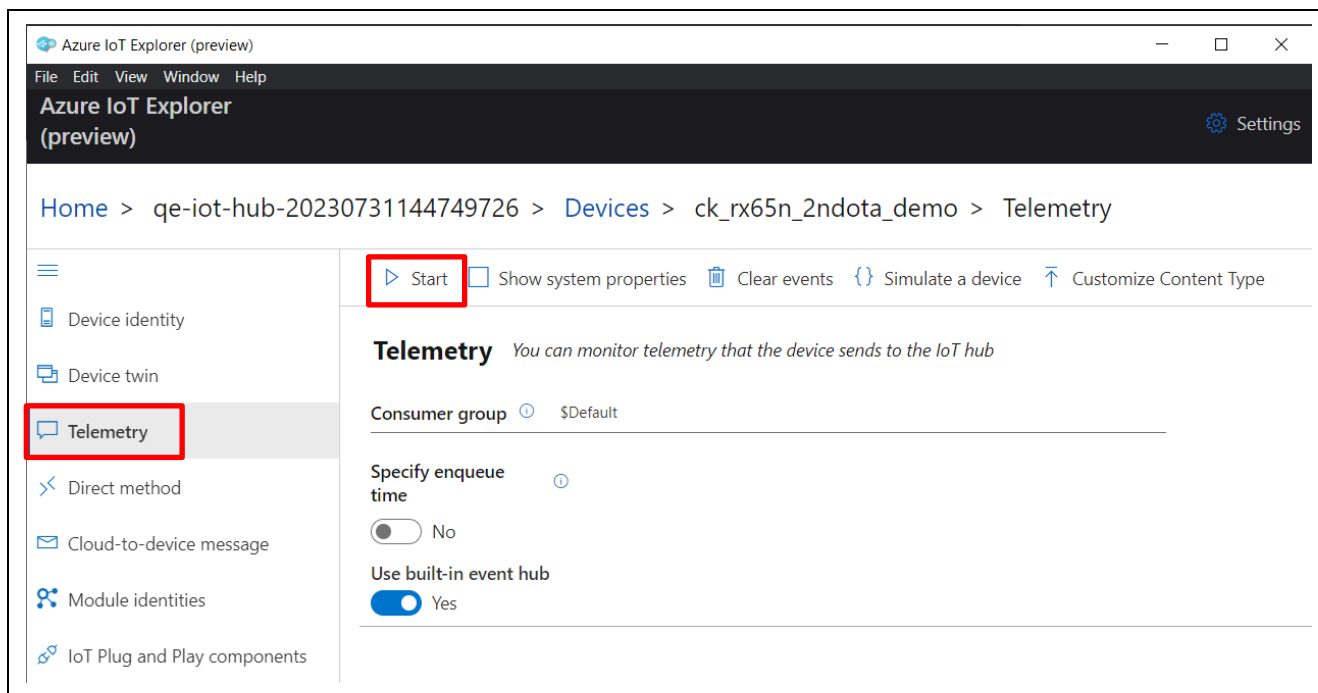


Figure 6-28 Start Receiving Telemetry Data

## 7. Procedure for Running the Demo

The detailed procedure for running the demo is described below.

### 7.1 Checking the Initial State of Operation

With the setup for the demo in chapter 6 completed, press the reset switch (RESET) on the TB-RX660 to execute a hardware reset. Similarly, press the reset switch (S1) on the CK-RX65N to execute a hardware reset.

Check the logs from each microcontroller using terminal emulator software.

Figure 7-1 shows the log window of the CK-RX65N, confirming that the humidity data from the HS3001 sensor is displayed.

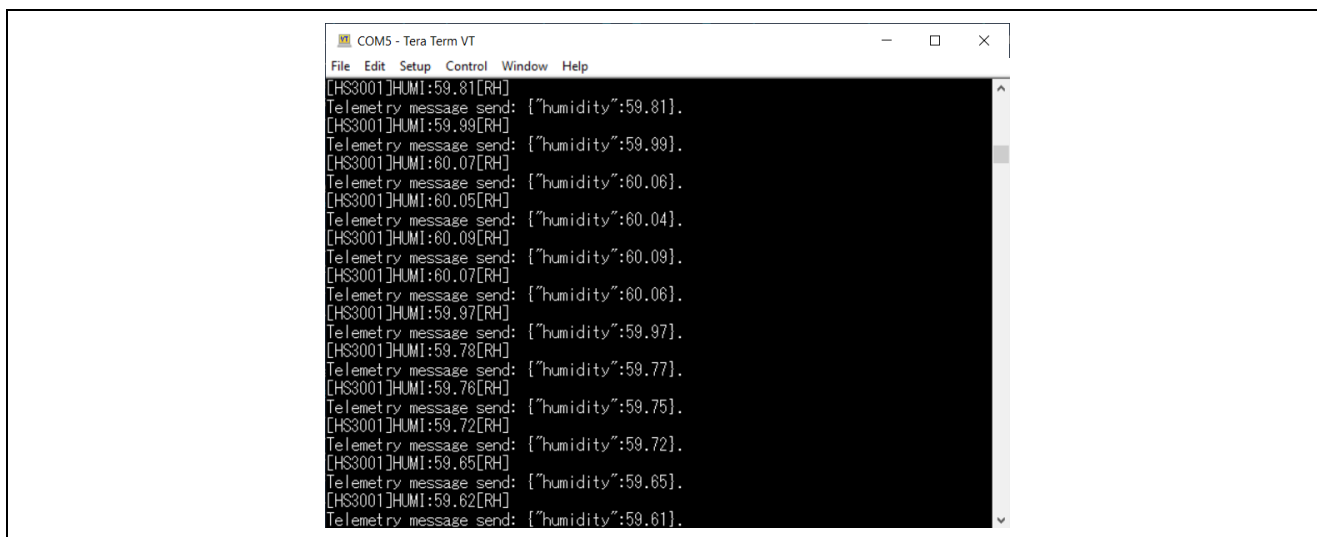


Figure 7-1 CK-RX65N Log Output Window

Next, Figure 7-2 shows the log window of TB-RX660, confirming that only the humidity data from the HS3001 sensor is displayed.

In addition, LED0 of TB-RX660 blinks in the initial state.

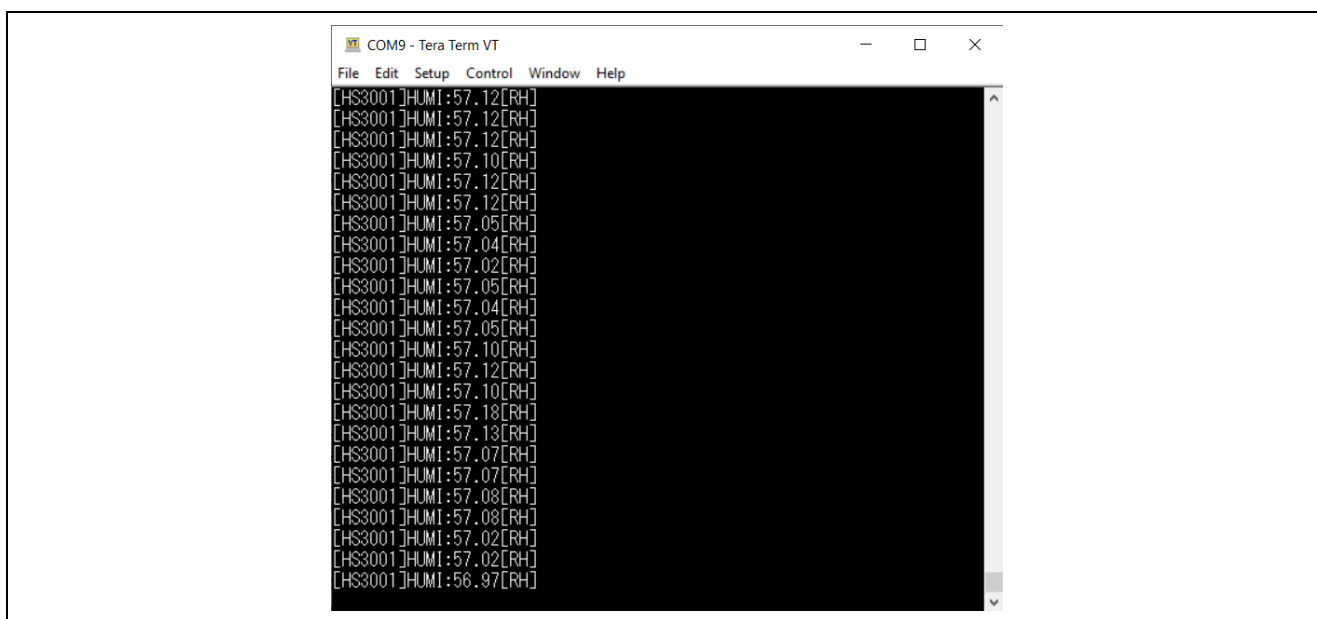


Figure 7-2 TB-RX660 Log Output Window

Finally, Figure 7-3 shows the Azure IoT Explorer window, confirming that the humidity data from the HS3001 sensor is displayed.

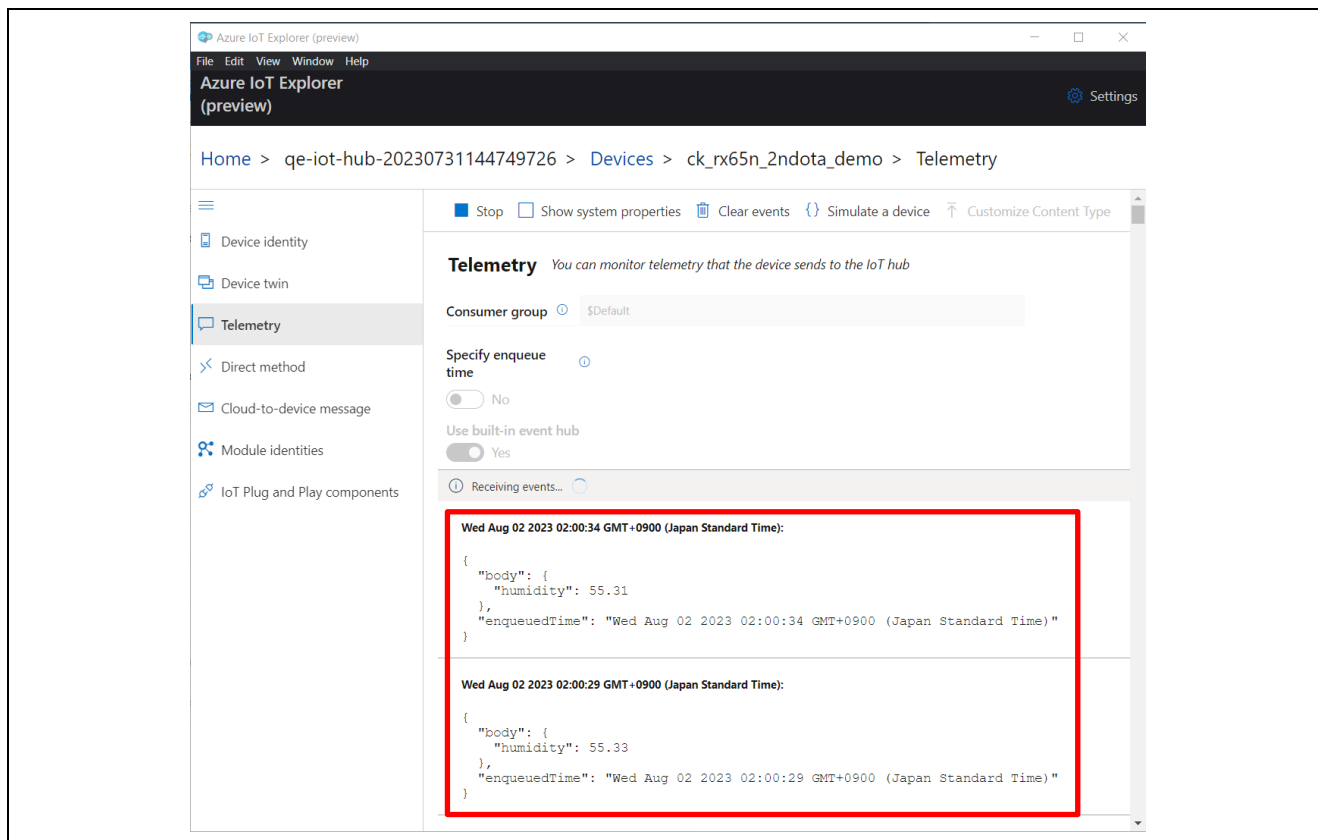


Figure 7-3 Azure IoT Explorer Telemetry Display before Secondary OTA Update

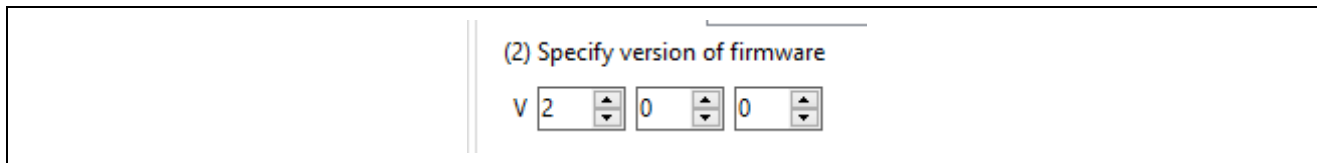
This is the initial state before the secondary OTA update is executed.

## 7.2 Run OTA Update

### 7.2.1 Create Update Firmware

#### 7.2.1.1 Create Update Firmware for CK-RX65N

Execute the Azure procedure in chapter “4.6.1 Create Update Firmware” of the [“RX Family Firmware Update Software Development Guide using AWS/Azure QE for OTA”](#). The version of the update firmware should be “2.0.0”.



(2) Specify version of firmware

v 2 0 0

Figure 7-4 Update Firmware Version Settings

This will create update firmware for the CK-RX65N.

Copy the “<IoT device name>.rsu” file in the “ck\_rx65n\_azure\_2ndota\_demo/QE-OTA/apl/v2.0.0/<IoT device name>” folder to the “ck\_rx65n\_azure\_2ndota\_demo\_tools/AzureDeviceUpdateScripts” folder and rename the file to “host\_update\_firm\_2.0.0.rsu”.

#### 7.2.1.2 Create Update Firmware for TB-RX660

##### (1) Changes in the Source Code of the ck\_rx65n\_azure\_2ndota\_demo Project

Change the definition of the “MEASURE\_TEMPERATURE” macro from 0 to 1 of “rx660\_tb\_2ndota\_demo/src/rx660\_tb\_2ndota\_demo.c”.

You can also change the firmware version displayed in the log output by setting DEMO\_VER\_MAJOR, MINOR, and BUILD to any value.

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

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

##### (2) Create Update Firmware (MOT file format)

Build the rx660\_tb\_2ndota\_demo project and create the MOT file.

### (3) Create Update Firmware (RSU file format)

Convert the created rx660\_tb\_2ndota\_demo MOT file to update firmware in RSU format using Renesas Image Generator.

Run the following command in the “rx660\_tb\_2ndota\_demo/RenasasImageGenerator” folder to create the update firmware **leaf\_update\_firm\_2.0.0.rsu**.

```
> python .\image-gen.py -iup ..\HardwareDebug\rx660_tb_2ndota_demo.mot -
o ..\..\ck_rx65n_azure_2ndota_demo\tools\AzureDeviceUpdateScripts\leaf_update_firm_2.0.
0 -key .\keys\secp256r1.privatekey -
ip .\RX660_Linear_Half_ImageGenerator_PRM_2ndota_demo.csv -vt ecdsa
```

#### 7.2.2 Create a Manifest File

Launch Windows PowerShell and run the following command in the “ck\_rx65n\_azure\_2ndota\_demo\_tools\AzureDeviceUpdateScripts” folder to create the manifest file.

```
.\CreateRX65NRSKUpdate.ps1
```

You will be asked to enter Version, HostPath, and LeafPath, each of which should be entered as follows:

- Version: 2.0.0  
Manifest file version. It is independent of each firmware version.
- HostPath: host\_update\_firm\_2.0.0.rsu  
The RSU file created in 7.2.1.1.
- LeafPath: leaf\_update\_firm\_2.0.0.rsu  
The RSU file created in 7.2.1.2(3)

```
# .\CreateCKRX65NUpdate.ps1

cmdlet CreateCKRX65NUpdate.ps1 at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
Version: 2.0.0
HostPath: host_update_firm_2.0.0.rsu
LeafPath: leaf_update_firm_2.0.0.rsu
Preparing update RENESAS.CK-RX65N.2.0.0 ...
Preparing child update RENESAS.CK-RX65N-Leaf.2.0.0 ...
Preparing child update manifest RENESAS.CK-RX65N-Leaf.2.0.0 ...
Saving child manifest files and payload to .\RENESAS.CK-RX65N.2.0.0 ...

Preparing parent update RENESAS.CK-RX65N.2.0.0 ...
Generating an import manifest RENESAS.CK-RX65N.2.0.0 ...
Saving parent manifest file and payload(s) to .\RENESAS.CK-RX65N.2.0.0 ...
```

Figure 7-5 Create a Manifest File

When execution is complete, a folder “RENESAS.CK-RX65N.2.0.0” will be created and a manifest file and updated firmware will be generated in the folder.

### 7.2.3 Deploying an Update

#### (1) Add ADUGroup Tag to the Device

Open the Azure portal in a browser and select “Devices” from the left menu of the IoT Hub.

Check the Device ID created with QE for OTA in 6.2.3(6) and click “Assign tags”.

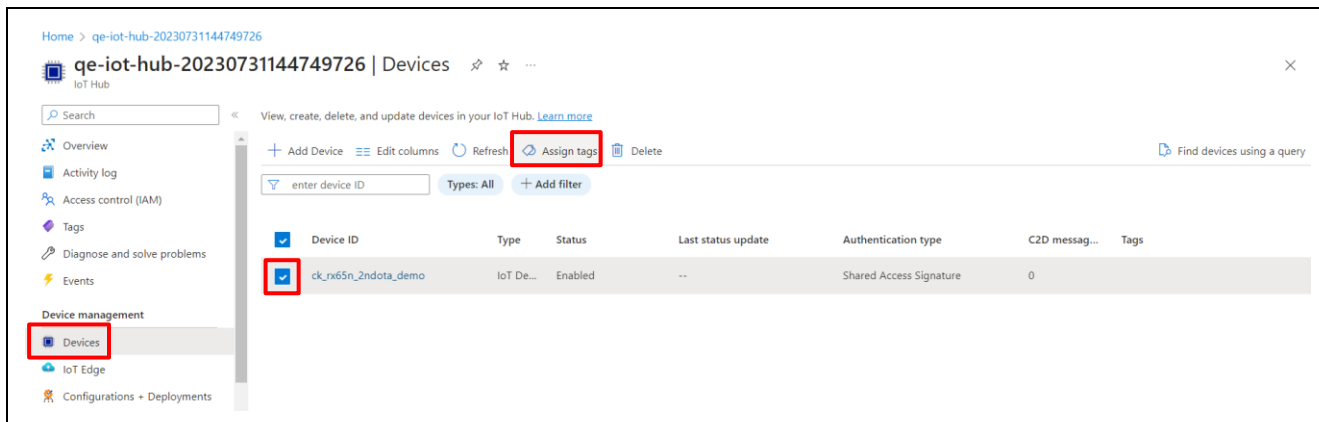


Figure 7-6 Select Device and Add Tags

Enter “ADUGroup” for Name and “OTADemoGroup” for Value and click “Save”.

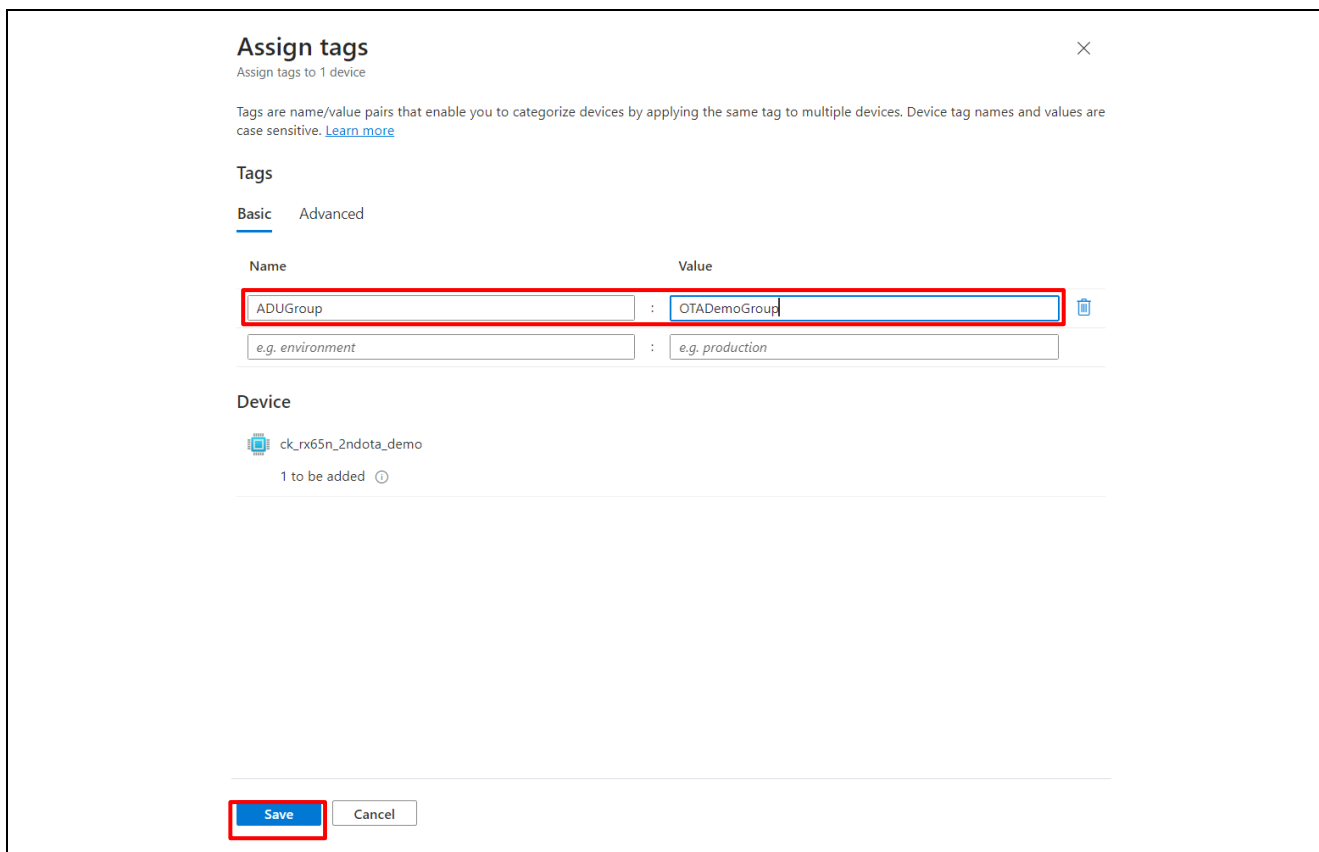


Figure 7-7 Add ADUGroup Tag

## (2) Add Permission to Update

Return to the IoT Hub page and select “Updates” from the left side menu.

Click the gear icon in the upper right corner and click “View account details”.

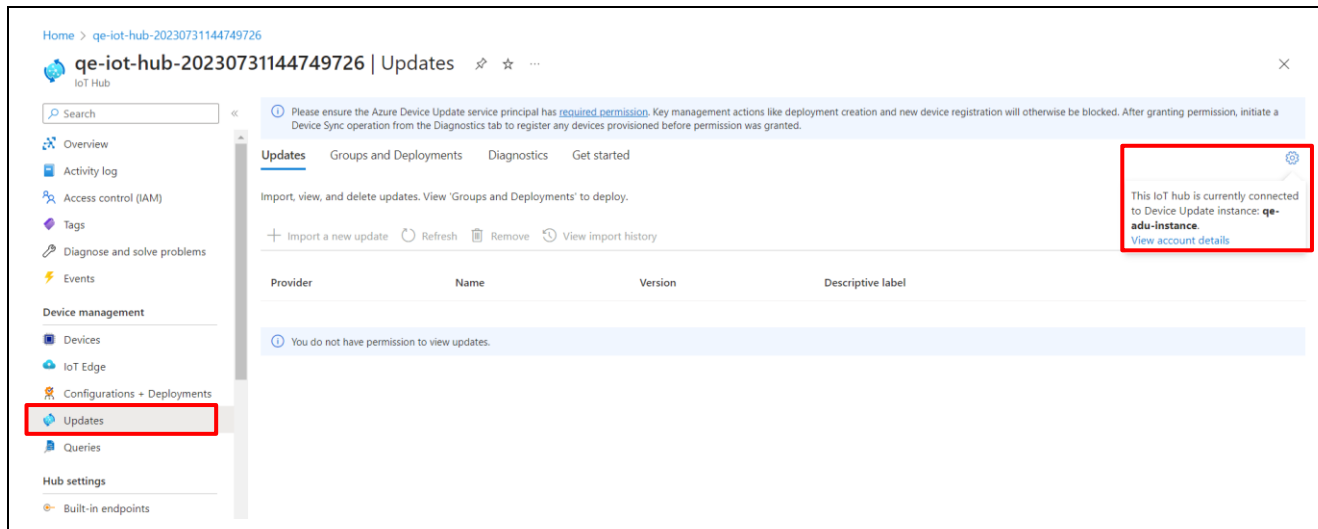


Figure 7-8 Go to the Device Update Account Page from the Updates Page

The Device update account page will open, click “Access control(IAM)” from the left menu, and select “Add role assignment” from the “Add” tab.

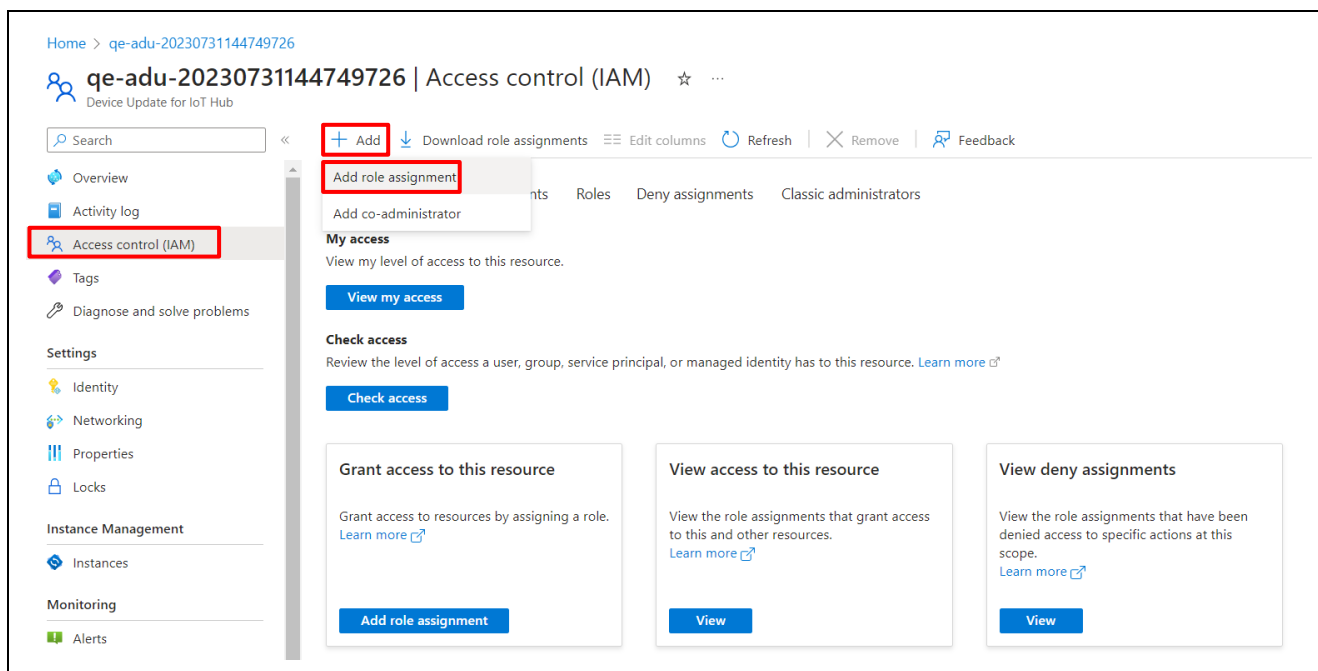


Figure 7-9 Add Roles from the IAM Page



Select “Device Update Administrator” from the list of roles and click “Next”.

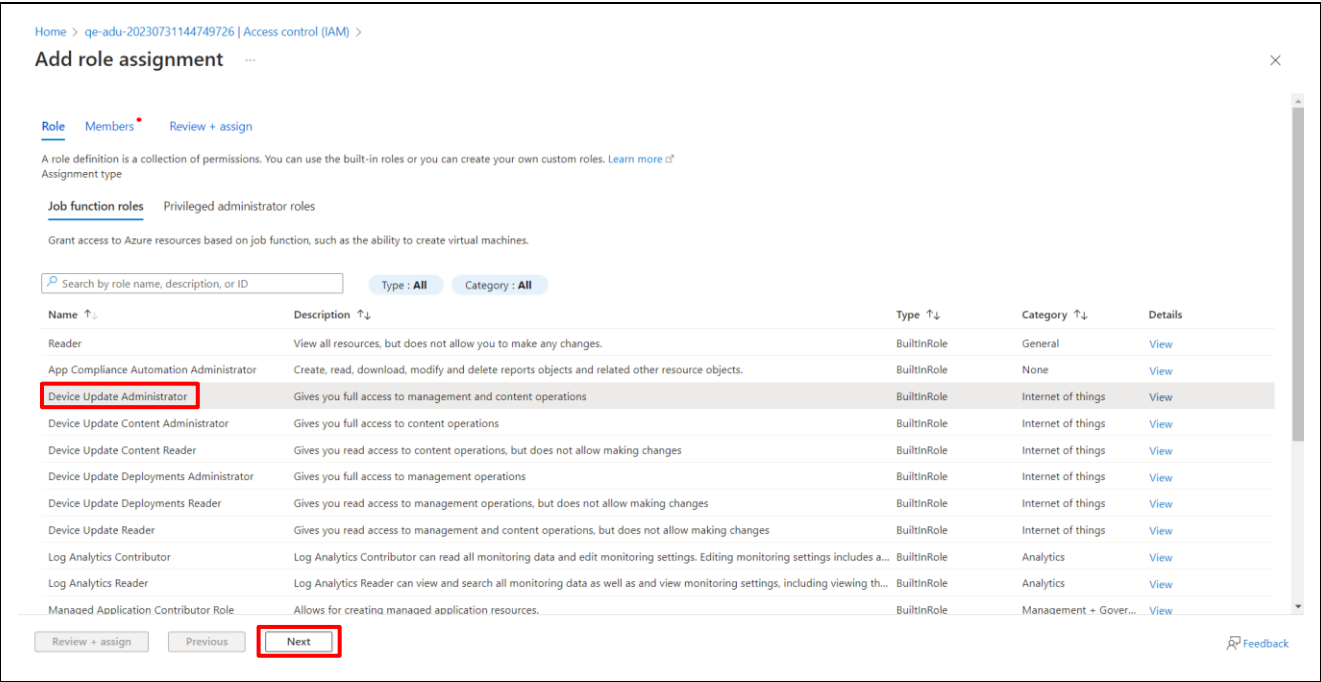


Figure 7-10 Select the Device Update Administrator Role from the List of Roles

Click “Select members,” search for your account name and click on it.

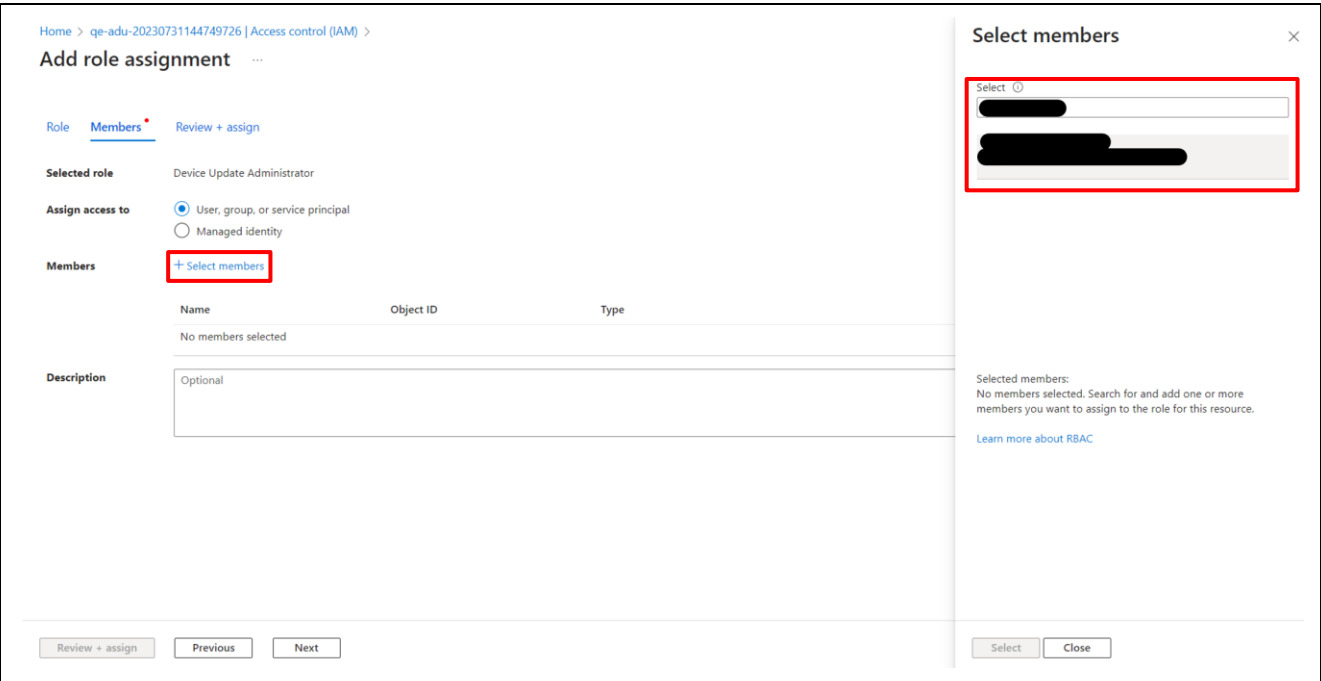


Figure 7-11 Select the Member to Assign the Role (1)

Confirm that your account is selected and click “Select”.

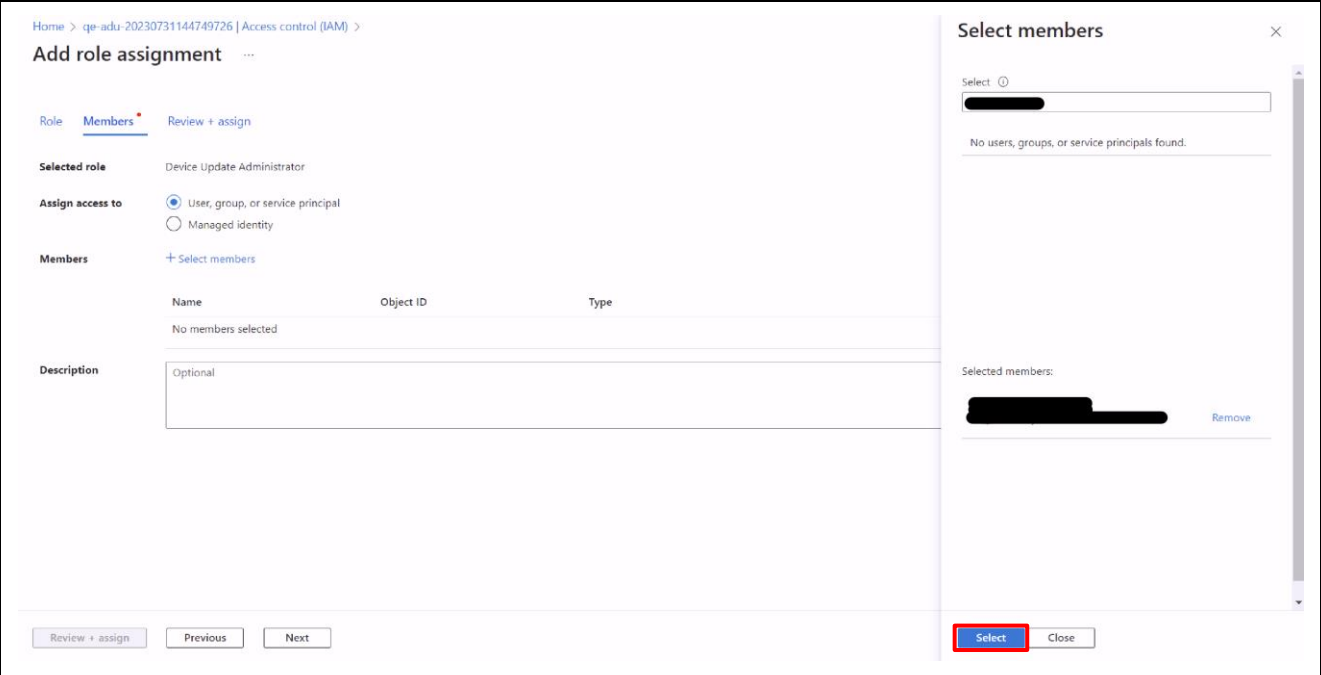


Figure 7-12 Select the Member to Assign the Role (2)

Confirm that your account is listed in “Members” and click “Review + assign”.

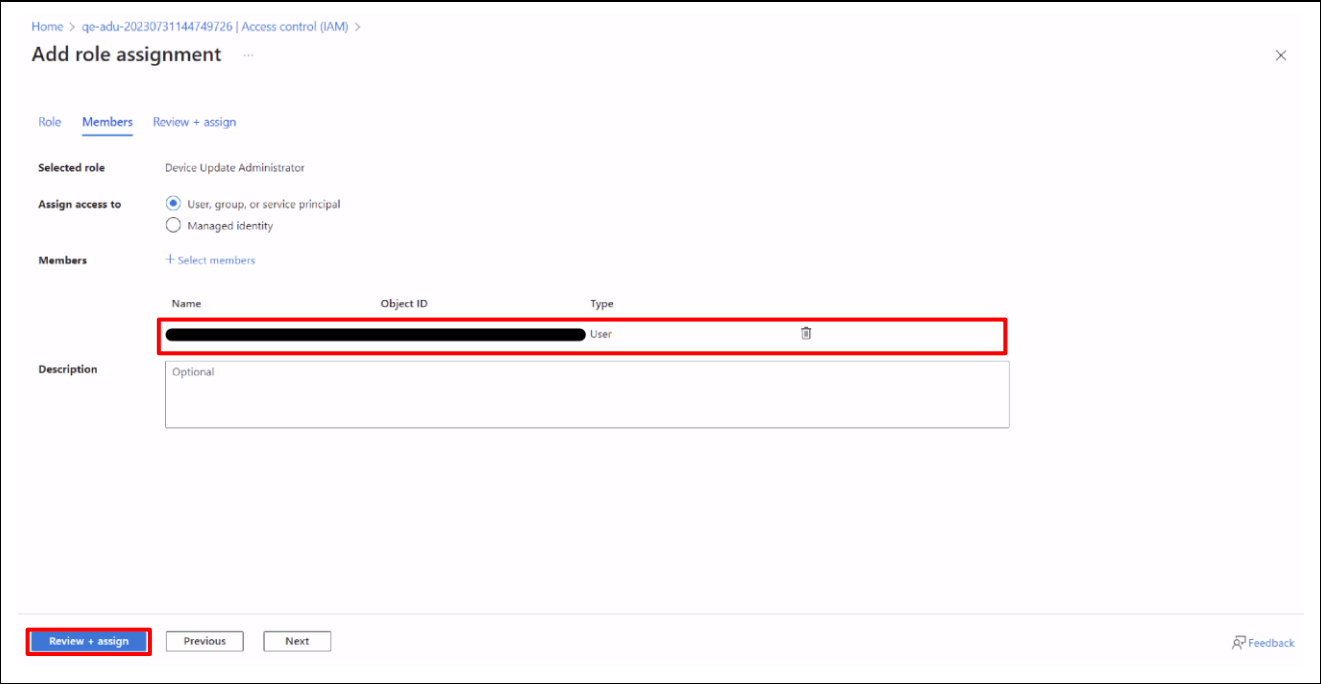


Figure 7-13 Assign the Device Update Administrator Role

### (3) Import a New Update

Return to the IoT Hub page and select “Updates” from the left menu.

Click “Import a new update”.

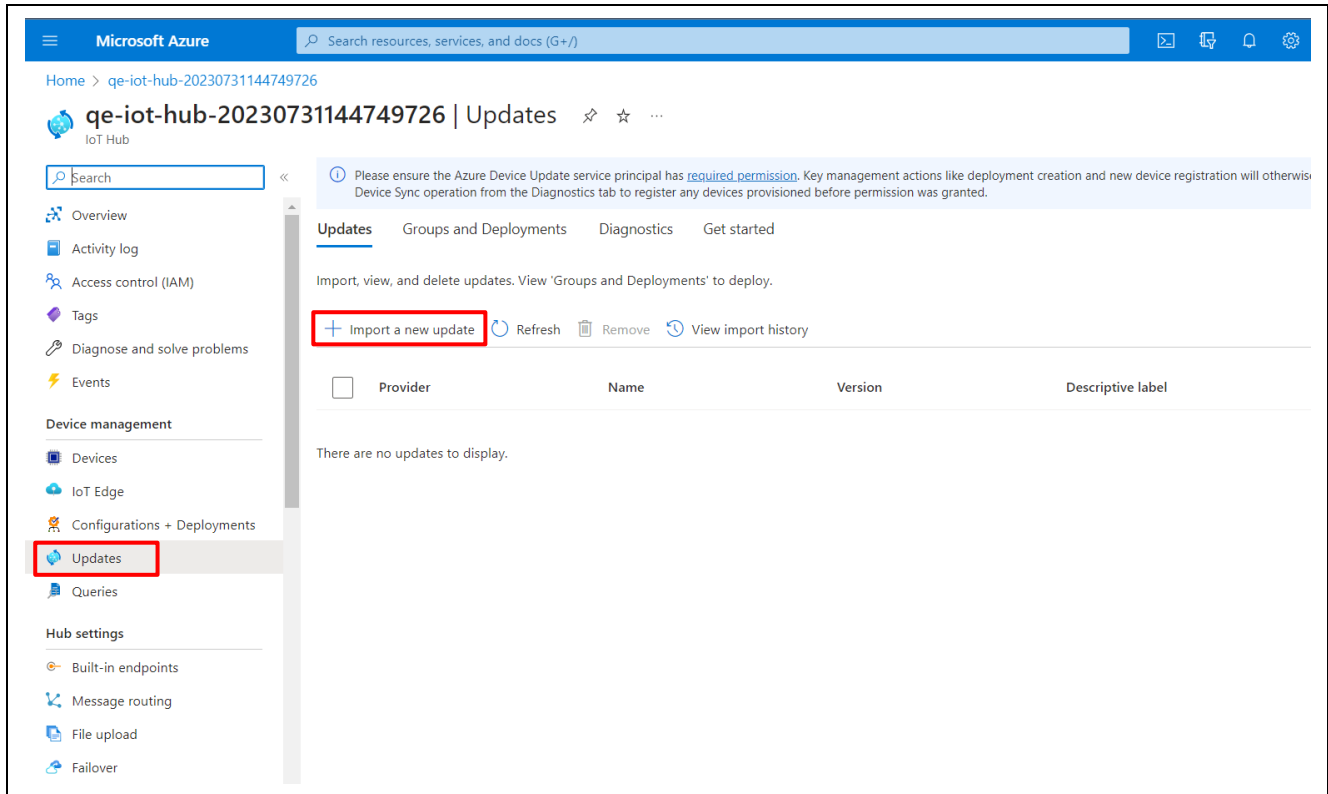


Figure 7-14 Import a New Update

Click “Select from storage container”.

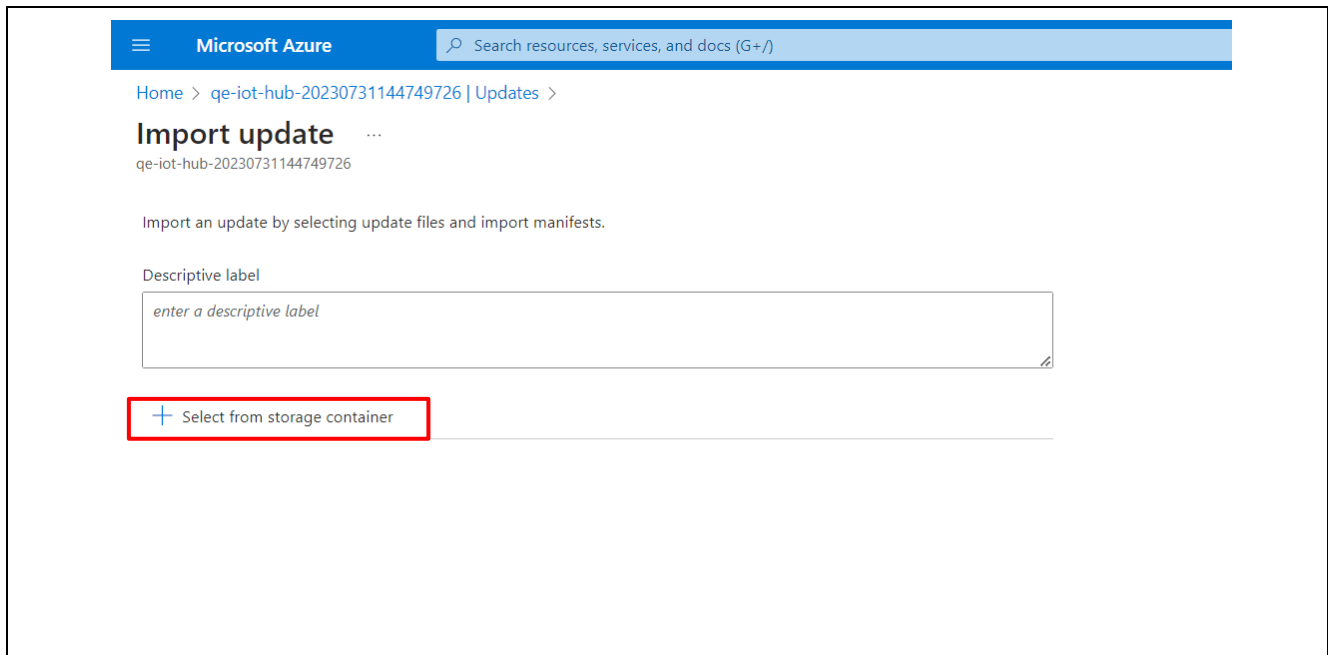


Figure 7-15 Select from Storage Container

Select the storage account created with QE for OTA in 6.2.3(3) and select the container on the Containers page that appears.

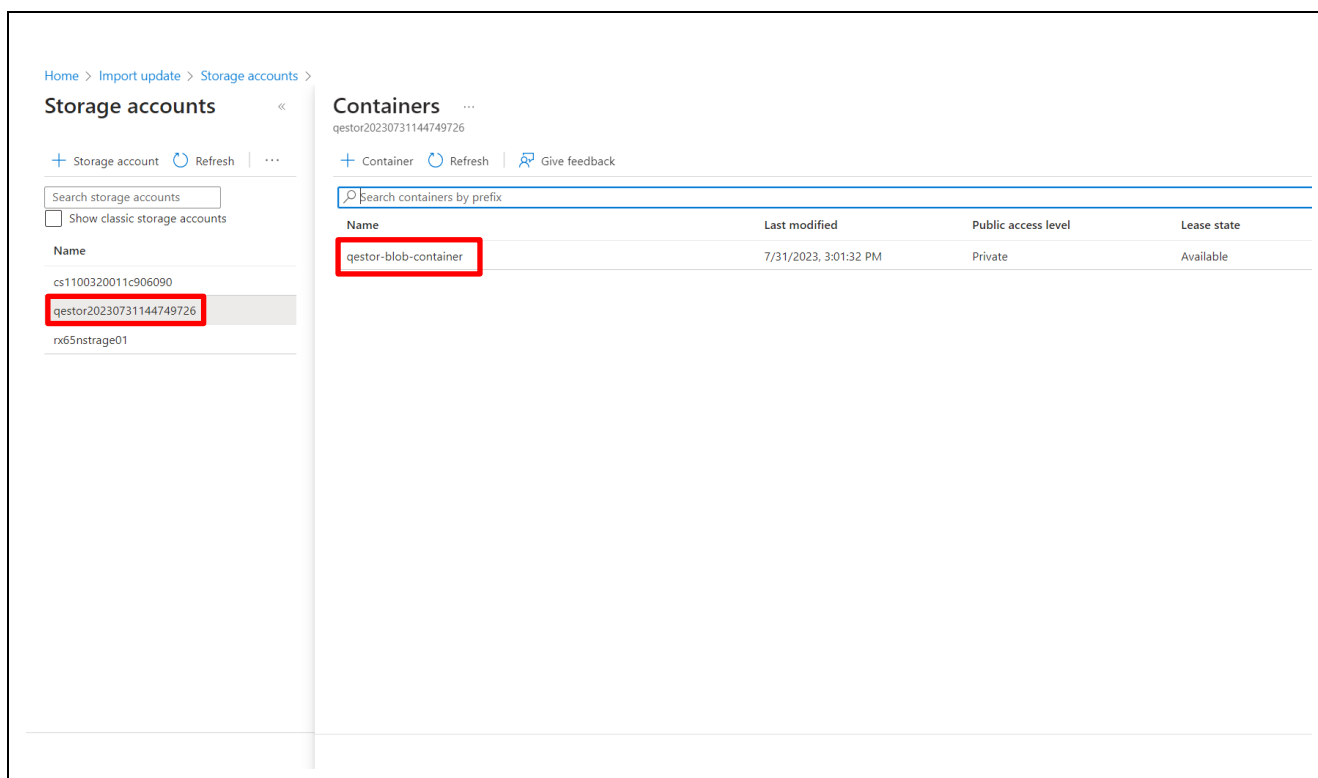


Figure 7-16 Select the Container

Click "Upload" on the Container page that appears, and upload the set of files in the "RENESAS.CK-RX65N.2.0.0" folder created in section 7.2.2.

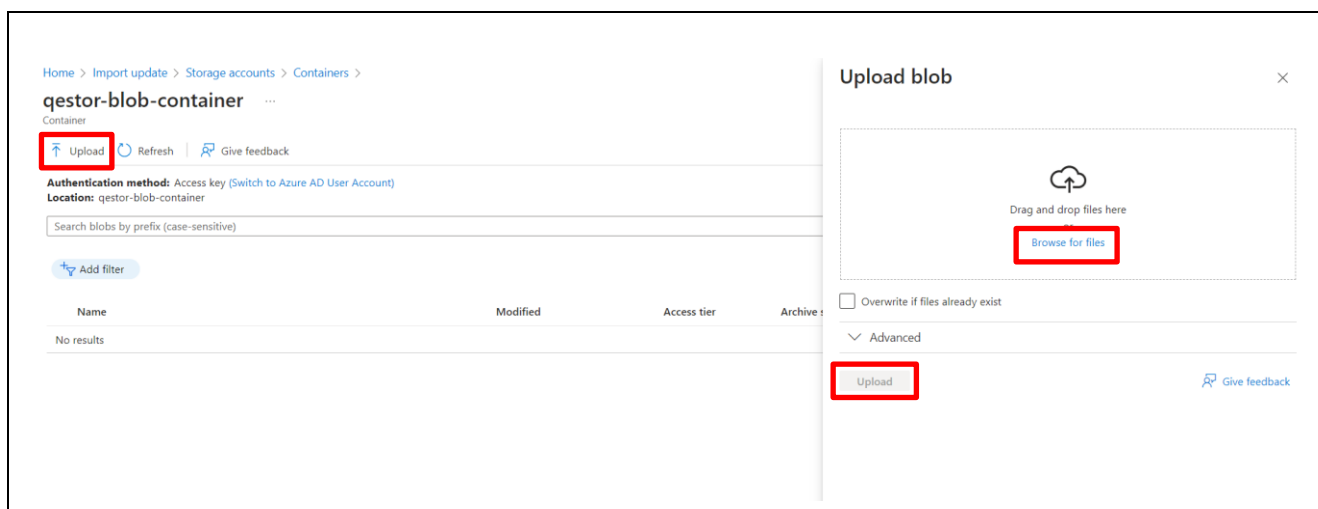


Figure 7-17 Upload Manifest File and Update Firmware

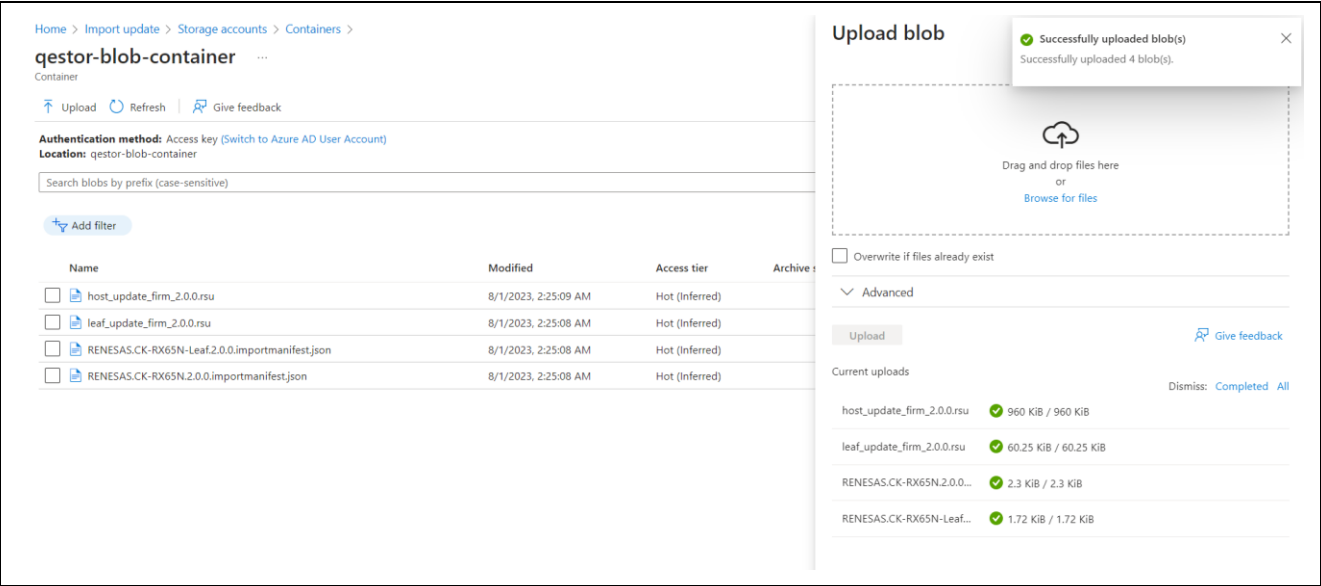


Figure 7-18 Manifest File and Update Firmware Successfully Uploaded

Check all uploaded files and click “Select”.

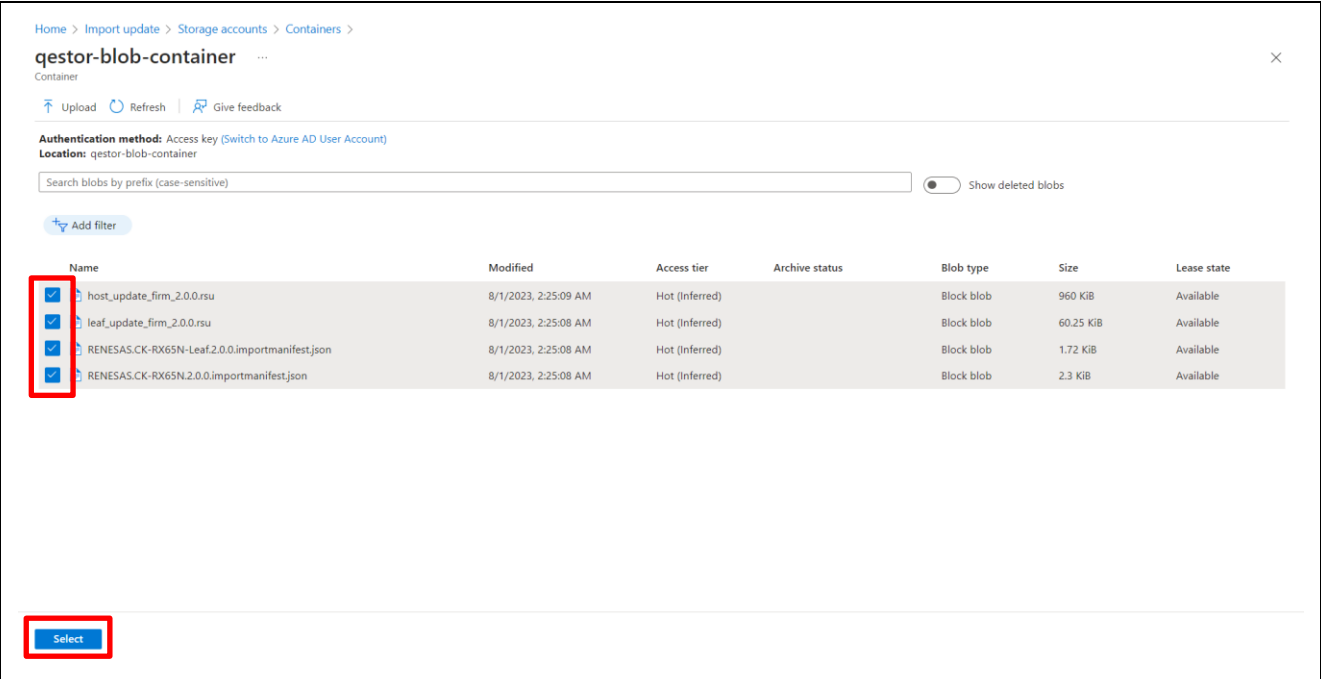
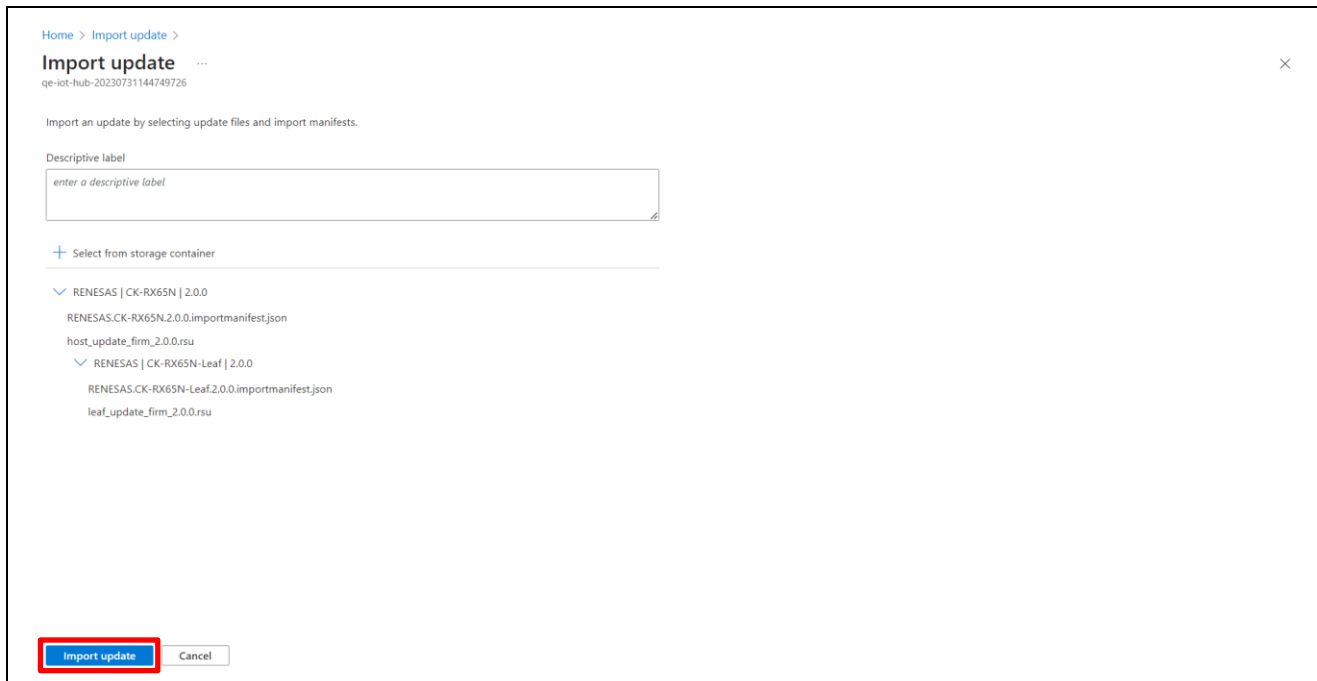


Figure 7-19 Select Manifest File and Update Firmware

Click “Import update”.



Home > Import update >

### Import update

qe-iot-hub-20230731144749726

Import an update by selecting update files and import manifests.

Descriptive label

enter a descriptive label

+ Select from storage container

✓ RENESAS | CK-RX65N | 2.0.0

RENESAS.CK-RX65N.2.0.0.importmanifest.json

host\_update\_firm\_2.0.0.rsu

✓ RENESAS | CK-RX65N-Leaf | 2.0.0

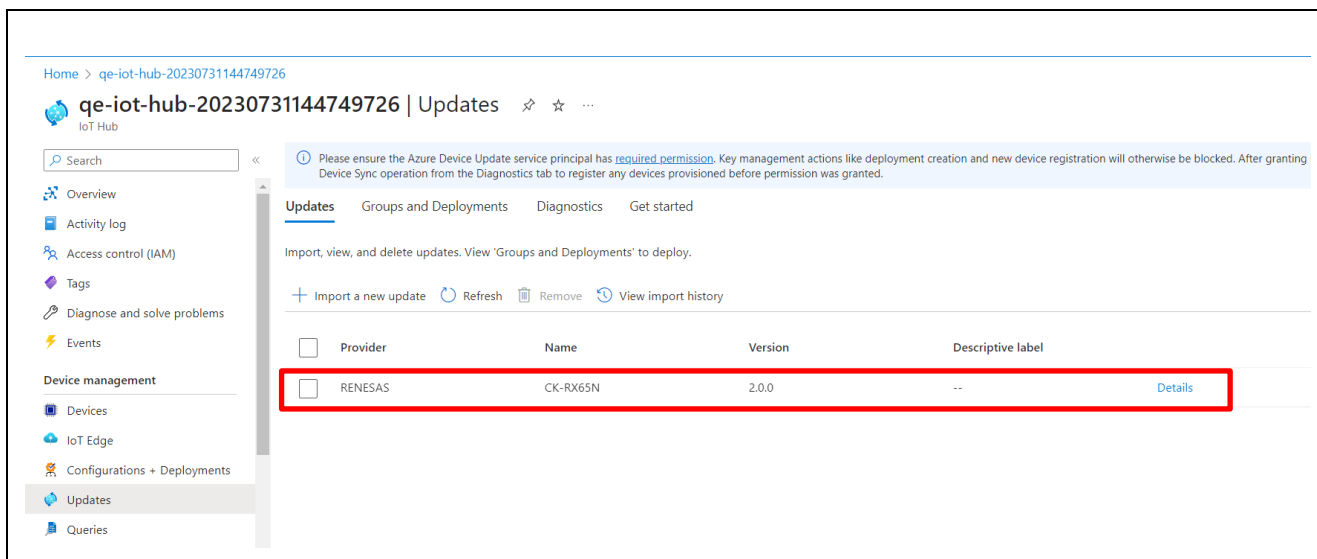
RENESAS.CK-RX65N-Leaf.2.0.0.importmanifest.json

leaf\_update\_firm\_2.0.0.rsu

**Import update** Cancel

Figure 7-20 Import Update Firmware

Once the import is complete, you will see the update information. It may take several minutes for the process to complete.



Home > qe-iot-hub-20230731144749726

### qe-iot-hub-20230731144749726 | Updates

IoT Hub

Please ensure the Azure Device Update service principal has [required permission](#). Key management actions like deployment creation and new device registration will otherwise be blocked. After granting Device Sync operation from the Diagnostics tab to register any devices provisioned before permission was granted.

Updates Groups and Deployments Diagnostics Get started

Import, view, and delete updates. View 'Groups and Deployments' to deploy.

+ Import a new update Refresh Remove View import history

<input type="checkbox"/>	Provider	Name	Version	Descriptive label	
<input type="checkbox"/>	RENASAS	CK-RX65N	2.0.0	--	<a href="#">Details</a>

Figure 7-21 Successful Import of the Update

#### (4) Deploy an Update

Select the “Groups and Deployments” tab and click on the “OTADemoGroup” group created in 7.2.3(1). If the group does not appear, press the RESET button on the CK-RX65N, reconnect the device to the IoT Hub, and then click “Refresh” on the page.

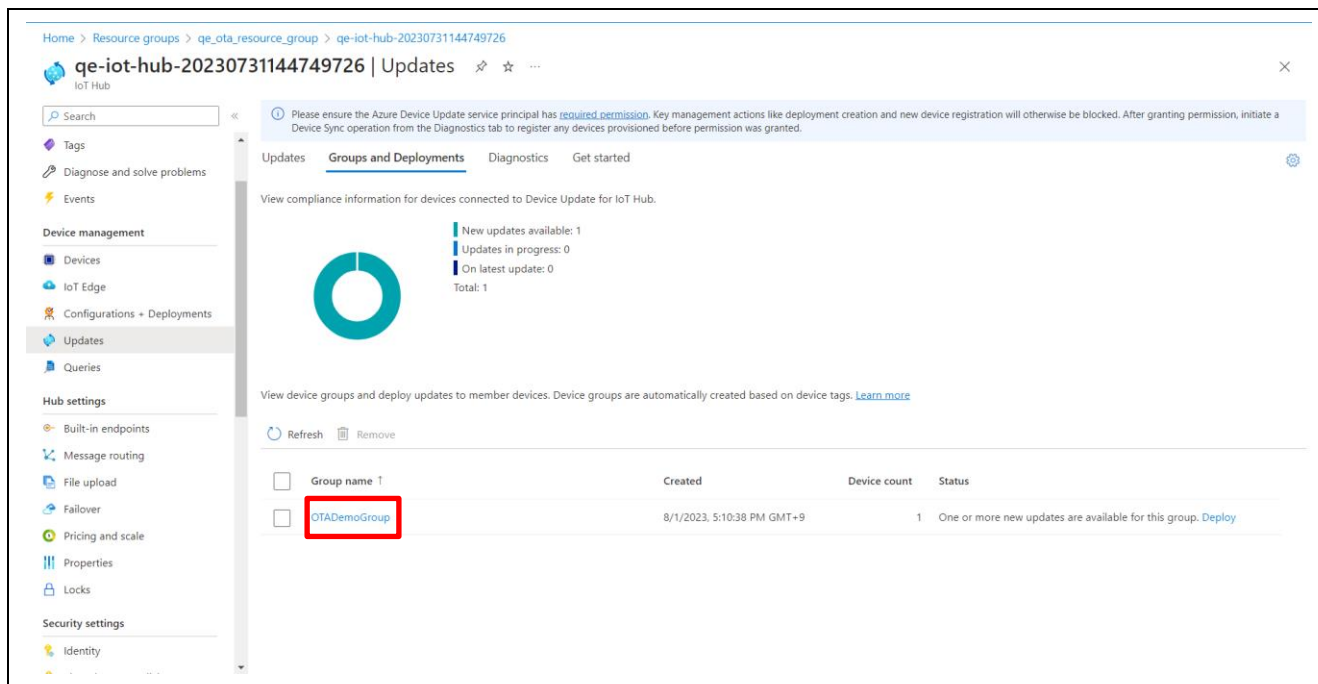


Figure 7-22 Select a Group on the Groups and Deployments Page

Click “Deploy”.

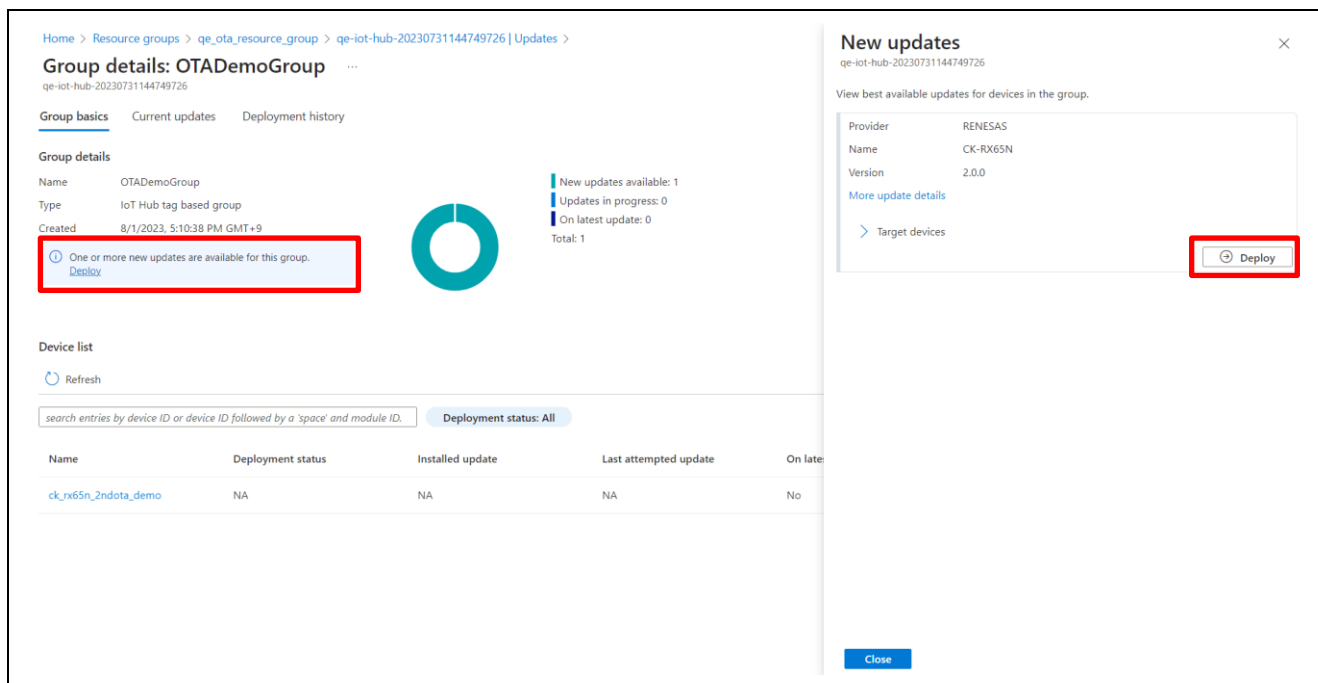


Figure 7-23 Deploy the Update

Click “Create” to deploy the update.

Create deployment

qe-iot-hub-20230731144749726

Create a new deployment targeting devices in this group.

Update properties

Provider

RENESAS

Name

CK-RX65N

Version

2.0.0

Group name

OTADemoGroup

Specify when this deployment should start. \*

☒ Start immediately.

☐ Start at a scheduled date and time.

☐ Rollback all applicable devices in the group to a different update if rollback criteria is met

Create

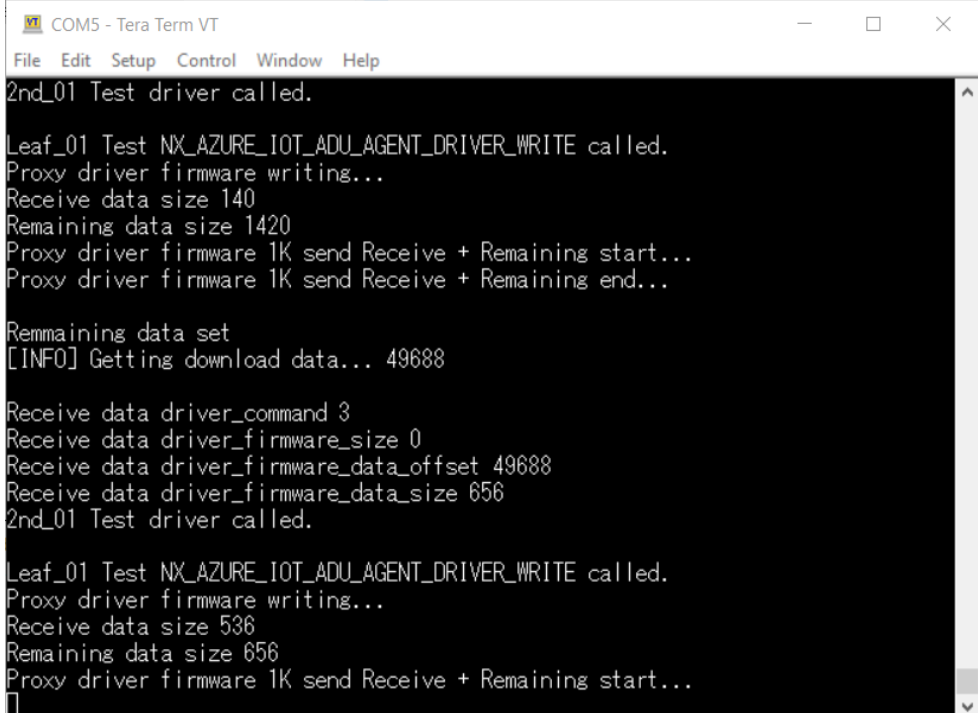
Cancel

Figure 7-24 Create Deployment



### 7.3 Check Operation during OTA Update Running

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



```

COM5 - Tera Term VT
File Edit Setup Control Window Help
2nd_01 Test driver called.

Leaf_01 Test NX_AZURE_IOT_ADU_AGENT_DRIVER_WRITE called.
Proxy driver firmware writing...
Receive data size 140
Remaining data size 1420
Proxy driver firmware 1K send Receive + Remaining start...
Proxy driver firmware 1K send Receive + Remaining end...

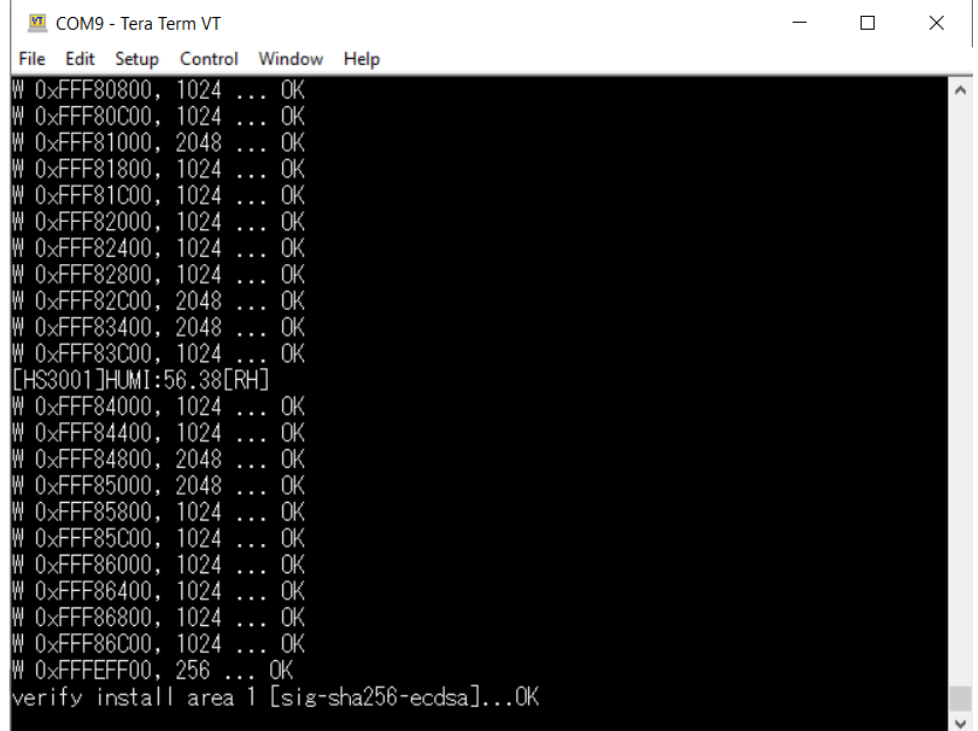
Remmaining data set
[INFO] Getting download data... 49688

Receive data driver_command 3
Receive data driver_firmware_size 0
Receive data driver_firmware_data_offset 49688
Receive data driver_firmware_data_size 656
2nd_01 Test driver called.

Leaf_01 Test NX_AZURE_IOT_ADU_AGENT_DRIVER_WRITE called.
Proxy driver firmware writing...
Receive data size 536
Remaining data size 656
Proxy driver firmware 1K send Receive + Remaining start...

```

Figure 7-25 CK-RX65N Log during Update Running



```

COM9 - Tera Term VT
File Edit Setup Control Window Help
W 0xFFF80800, 1024 ... OK
W 0xFFF80C00, 1024 ... OK
W 0xFFF81000, 2048 ... OK
W 0xFFF81800, 1024 ... OK
W 0xFFF81C00, 1024 ... OK
W 0xFFF82000, 1024 ... OK
W 0xFFF82400, 1024 ... OK
W 0xFFF82800, 1024 ... OK
W 0xFFF82C00, 2048 ... OK
W 0xFFF83400, 2048 ... OK
W 0xFFF83C00, 1024 ... OK
[CHS3001]HUMI:56.38[RH]
W 0xFFF84000, 1024 ... OK
W 0xFFF84400, 1024 ... OK
W 0xFFF84800, 2048 ... OK
W 0xFFF85000, 2048 ... OK
W 0xFFF85800, 1024 ... OK
W 0xFFF85C00, 1024 ... OK
W 0xFFF86000, 1024 ... OK
W 0xFFF86400, 1024 ... OK
W 0xFFF86800, 1024 ... OK
W 0xFFF86C00, 1024 ... OK
W 0xFFFEFF00, 256 ... OK
verify install area 1 [sig-sha256-ecdsa]...OK

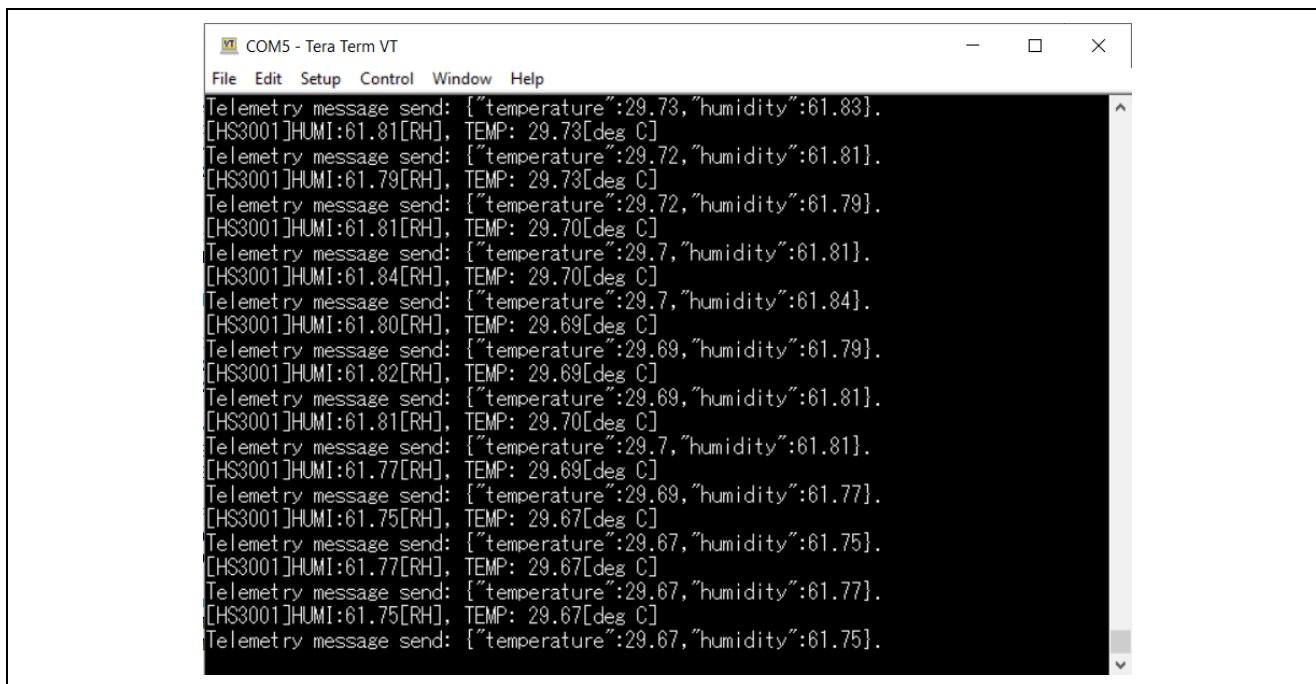
```

Figure 7-26 TB-RX660 Log during Update Running

## 7.4 Check Operation after OTA Update

Figure 7-27 shows the log window of the CK-RX65N after the update.

In addition to the humidity data acquired by the HS3001 sensor, temperature data can be seen on the log window.

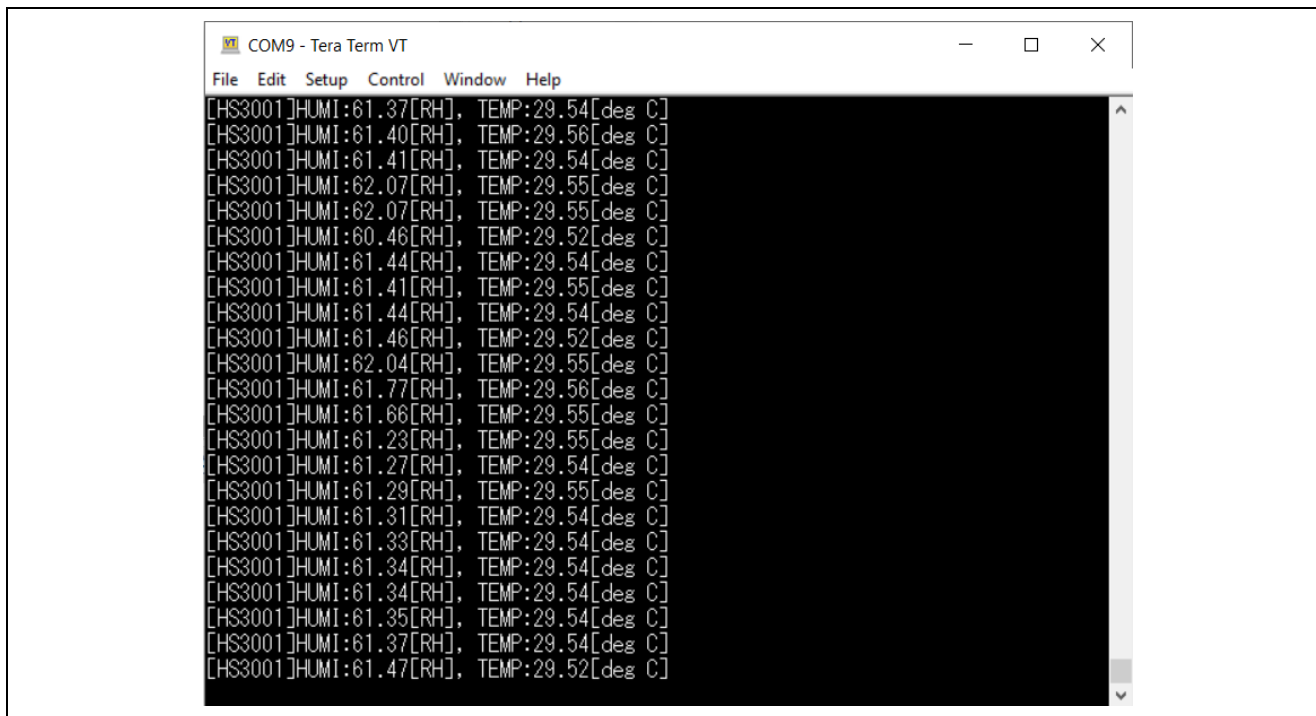


```
COM5 - Tera Term VT
File Edit Setup Control Window Help
Telemetry message send: {"temperature":29.73,"humidity":61.83}.
[HS3001]HUMI:61.81[RH], TEMP: 29.73[deg C]
Telemetry message send: {"temperature":29.72,"humidity":61.81}.
[HS3001]HUMI:61.79[RH], TEMP: 29.73[deg C]
Telemetry message send: {"temperature":29.72,"humidity":61.79}.
[HS3001]HUMI:61.81[RH], TEMP: 29.70[deg C]
Telemetry message send: {"temperature":29.7,"humidity":61.81}.
[HS3001]HUMI:61.84[RH], TEMP: 29.70[deg C]
Telemetry message send: {"temperature":29.7,"humidity":61.84}.
[HS3001]HUMI:61.80[RH], TEMP: 29.69[deg C]
Telemetry message send: {"temperature":29.69,"humidity":61.79}.
[HS3001]HUMI:61.82[RH], TEMP: 29.69[deg C]
Telemetry message send: {"temperature":29.69,"humidity":61.81}.
[HS3001]HUMI:61.81[RH], TEMP: 29.70[deg C]
Telemetry message send: {"temperature":29.7,"humidity":61.81}.
[HS3001]HUMI:61.77[RH], TEMP: 29.69[deg C]
Telemetry message send: {"temperature":29.69,"humidity":61.77}.
[HS3001]HUMI:61.75[RH], TEMP: 29.67[deg C]
Telemetry message send: {"temperature":29.67,"humidity":61.75}.
[HS3001]HUMI:61.77[RH], TEMP: 29.67[deg C]
Telemetry message send: {"temperature":29.67,"humidity":61.77}.
[HS3001]HUMI:61.75[RH], TEMP: 29.67[deg C]
Telemetry message send: {"temperature":29.67,"humidity":61.75}.
```

Figure 7-27 CK-RX65N Log Window after Firmware Update

Next, Figure 7-28 shows the log window of the TB-RX660 after the update. After a successful TB-RX660 firmware update, the HS3001 sensor will acquire humidity and temperature measurement data.

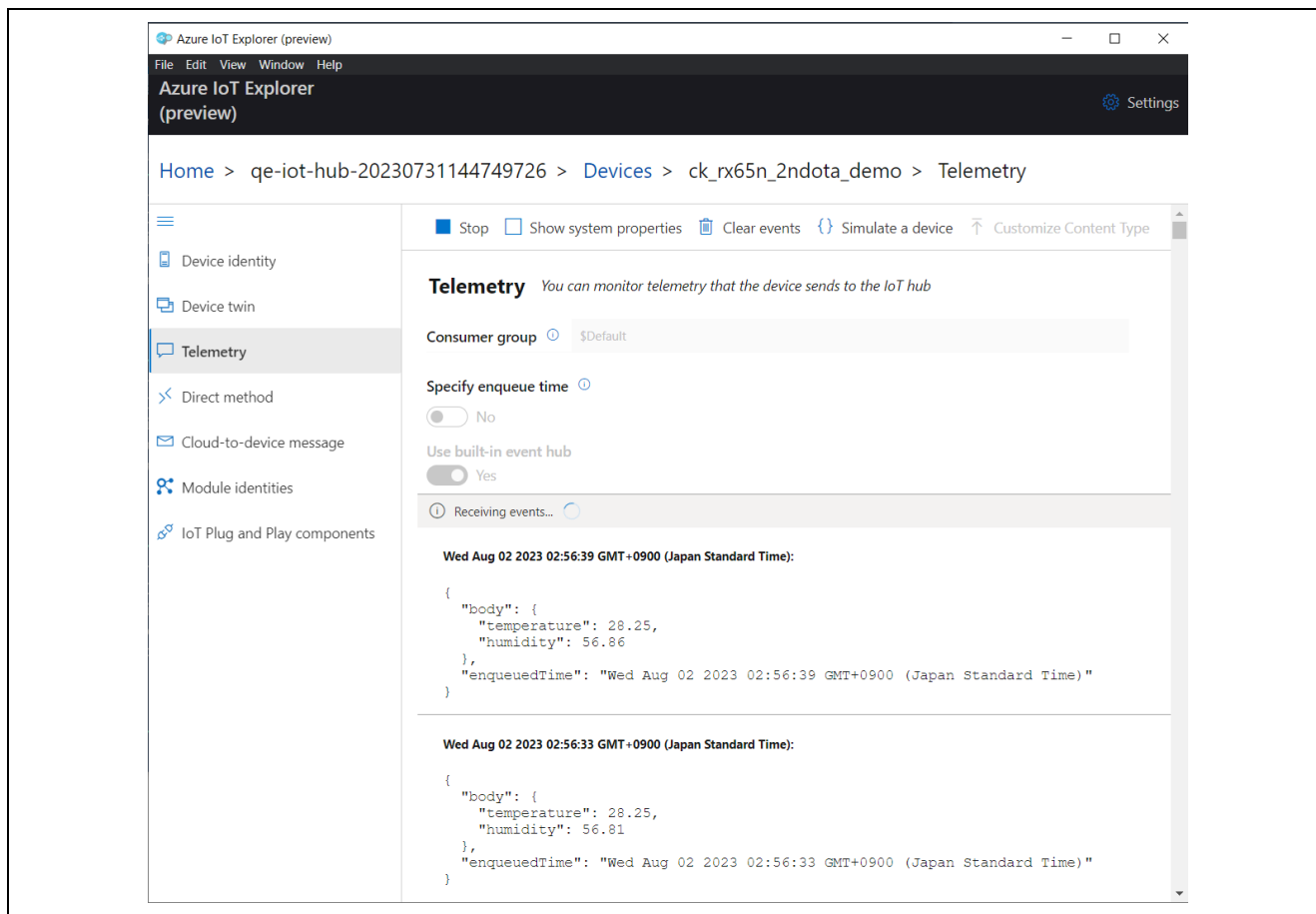
Also, LED1 will blink in addition to LED0, which blinks initially.



```
COM9 - Tera Term VT
File Edit Setup Control Window Help
[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 7-28 TB-RX660 Log Window after Firmware Update

Finally, Figure 7-29 shows the Azure IoT Explorer window, where you can see the humidity and temperature measurement data acquired from the HS3001 sensor.



**Figure 7-29 Telemetry Display in Azure IoT Explorer after Secondary OTA Update**

That is all for the demo operation.

## 7.5 Cleanup of Azure Cloud Resources

Delete the Azure cloud resource created by the above demo operation.

Open the Resource groups service page from the Azure portal. Click the resource group created in the demo and click "Delete resource group" on the screen that appears.

Depending on your usage of Microsoft Azure, you may be charged by the cloud resources created in the demo. To avoid unintended charges, we recommend that you delete the resources on the cloud you created after the demo is done.

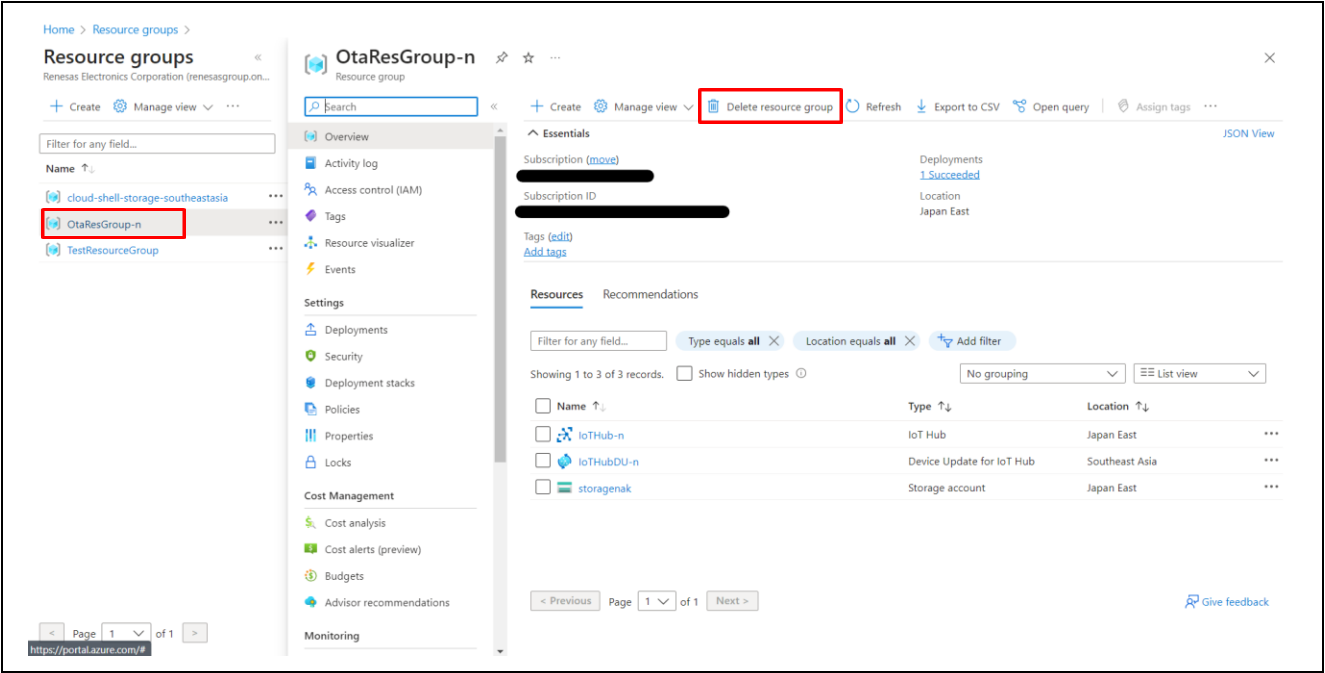


Figure 7-30 Delete Azure Cloud Resources

## 8. Note

### 8.1 Software License Information to Use

The software license information to use is shown below.

- TinyCrypt Cryptographic Library  
License <https://github.com/intel/tinycrypt/blob/master/LICENSE>
- Azure RTOS  
— License <https://github.com/azure-rtos/threadx/blob/master/LICENSED-HARDWARE.txt>

### 8.2 Firmware Update Module Modified Location

In `r_fwup.c` of the firmware update module used in the `rx660_tb_demo_bootloader` project, change `FWUP_COPY_BUF_SIZE` at line 37 to `256U` to match the RX660 flash write unit.

```

32
34      + Macro definitions
36      #define FWUP_READ_BUF_SIZE                (128U)
37      #define FWUP_COPY_BUF_SIZE                (256U) // modified for RX660
38      #define FWUP_VERIFY_BUF_SIZE             (128U)
39      #define FWUP_WRITE_HEADER_BUF_SIZE       (128U)
40      #define FWUP_IMAGE_FLAG                  (0xFEU)
41      #define FWUP_HASH_BYTES                  (32U)

```

Figure 8-1 Firmware Update Module Modified Location

### 8.3 Limitations/Restrictions of Leaf Version Information

The RSU file name generated in 7.2.1.2(3) has the following limitations:

File name: "2.0.0" in `leaf_update_firm_2.0.0.rsu` indicates the Leaf firmware version, and each digit has an input limit of 0~9.

Do not enter more than two digits.

Incorrect use case:

`leaf_update_firm_10.0.0.rsu` / `leaf_update_firm_1.10.0.rsu` / `leaf_update_firm_1.0.10.rsu`

The output file name can be handled by changing the following command.

```

> python .\image-gen.py -iup ..\HardwareDebug\rx660_tb_2ndota_demo.mot -
o ..\..\ck_rx65n_azure_2ndota_demo\tools\AzureDeviceUpdateScripts\leaf_update_firm_2.0.0 -
key .\keys\secp256r1.privatekey -ip .\RX660_Linear_Half_ImageGenerator_PRM_2ndota_demo.csv -vt
ecdsa

```

Figure 8-2 Renesas Image Generator Command Changes

**Revision History**

Rev.	Date	Description	
		Page	Summary
1.00	Aug. 31, 2023	—	First edition issued.

# 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](http://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/](http://www.renesas.com/contact/).