

RX65N

Connecting AWS Cloud with FreeRTOS Getting Started Guide for CK-RX65N

Contents

1. Overview	3
2. Hardware Description	4
2.1 DataSheet	4
2.2 Schematic.....	4
2.3 Key Components.....	4
2.4 Hardware requirements to run FreeRTOS demo.....	5
2.4.1 Standard Kit Contents	5
2.4.2 User Provided items	5
2.4.3 3rd Party purchasable items.....	5
2.5 Additional Hardware References.....	5
3. Set up your Development Environment	6
3.1 Support IDEs	6
3.2 Toolchains.....	6
3.2.1 CC-RX compiler	6
3.2.2 GCC compiler.....	6
3.3 Establishing a serial connection	6
3.4 Other software required to develop and debug applications for the device	6
3.4.1 To download Tera Term	6
4. Set up your hardware	7
4.1 To connect the RYZ014A.....	7
4.2 To set up the device board.....	9
4.3 To connect for debugging and power supply.....	10
4.4 To connect for receiving debug logs.....	10
5. Setup your AWS account and Permissions	11
5.1 To attach the AmazonFreeRTOSFullAccess policy to your IAM user	11
5.2 To attach the AWSIoTFullAccess policy to your IAM user	11
6. Provision the device with AWS IoT	12
6.1 To create an AWS IoT policy.....	12
6.2 To create an IoT thing, private key, and certificate for your device	15
7. Download FreeRTOS	19
8. Configure FreeRTOS	21
8.1 Header file for AWS IoT	21

8.1.1 Ethernet case	21
8.1.2 Cellular case.....	22
8.2 The endpoint in AWS IoT	24
9. Build the FreeRTOS demo	25
10. Run the FreeRTOS demo project	26
10.1 Flash demo code.....	26
10.2 Receive MQTT messages by AWS IoT	26
11. Debugging	27
11.1 Open e ² studio to debug.....	27
11.2 Tera term.....	28
12. Troubleshooting	29
12.1 The Build errors.....	29
Revision History	30

1. Overview

This tutorial provides instructions of connecting AWS Cloud with FreeRTOS for getting started with CK-RX65N. If you do not have the CK-RX65N, visit the [AWS Partner Device Catalog](#), and purchase one from our partners.

This document explains how to configure AWS IoT Core and FreeRTOS to connect your device to the AWS Cloud.

2. Hardware Description

2.1 DataSheet

Table 2-1 DataSheet

Device	Link
CK-RX65N	https://www.renesas.com/rx/ck-rx65n
PMOD Expansion Board for RYZ014A	https://www.renesas.com/document/dst/ryz014a-lte-category-m1-module-datasheet?language=en&r=1503996

2.2 Schematic

Table 2-2 Schematic

Device	Link
CK-RX65N	https://www.renesas.com/rx/ck-rx65n
PMOD Expansion Board for RYZ014A	https://www.renesas.com/products/interface-connectivity/wireless-communications/cellular-iot-modules/rkyz014a0b00000be-pmod-expansion-board-ryz014a Documentation - PMOD-RYZ014A v01 - Design Package - pmod-ryz014a-schematic.pdf

2.3 Key Components

Table 2-3 Key Components

Device	Link
CK-RX65N	https://www.renesas.com/rx/ck-rx65n
PMOD Expansion Board for RYZ014A	https://www.renesas.com/products/interface-connectivity/wireless-communications/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module

2.4 Hardware requirements to run FreeRTOS demo

2.4.1 Standard Kit Contents

Table 2-4 Standard Kit Contents

Device	Link
CK-RX65N	https://www.renesas.com/rx/ck-rx65n
PMOD Expansion Board for RYZ014A (This is packaged in CK-RX65N)	https://www.renesas.com/products/interface-connectivity/wireless-communications/cellular-iot-modules/rkyz014a0b00000be-pmod-expansion-board-ryz014a

2.4.2 User Provided items

1. USB cable (A male – USB-Micro-B male) x2
These cables packaged in the CK-RX65N are used to connect the PC to the CK-RX65N to debug and receive debug logs.
2. LTE Antenna
3. MVNO SIM Card (MicroAi SIM)

Note: It is not necessary to prepare an emulator because it is attached on CK-RX65N.

2.4.3 3rd Party purchasable items

1. USB cable (A male – USB-Micro-B male) x1
It is used to supply supplemental power for the RYZ014A.

2.5 Additional Hardware References

See the Documentation section in the following CK-RX65N's page.

<https://www.renesas.com/rx/ck-rx65n>

3. Set up your Development Environment

3.1 Support IDEs

We support e² studio as IDE.

1. Go to the [Renesas e² studio installer](#) download page and download the offline installer.
2. You are directed to a Renesas Login page.
If you have an account with Renesas, enter your username and password and then choose **Login**.
If you do not have an account, choose **Register now**, and follow the first registration steps.
You should receive an email with a link to activate your Renesas account. Follow this link to complete your registration with Renesas, and then login to Renesas.
3. After you log in, download the e² studio installer to your computer.
4. Open the installer and follow the steps to completion.

For more information, see the [e² studio](#) on the Renesas website.

Note: Host machine running Windows 8.1 or 10. Linux and MacOS are not supported.

3.2 Toolchains

3.2.1 CC-RX compiler

1. Go to the [C/C++ Compiler Package for RX Family](#), and download the V3.04.00 package.
<https://www.renesas.com/document/ucm/rx-compiler-cc-rx-v30400-e-studio?language=en&r=1169511>
2. Open the executable and install the compiler.
For more information, see the [C/C++ Compiler Package for RX Family](#) on the Renesas website.

Note: The compiler is available free for evaluation version only and valid for 60 days. On the 61st day, you need to get a License Key. For more information, see [Evaluation Software Tools](#).

3.2.2 GCC compiler

1. Go to the [Open Source Tools for RENESAS](#), and download the GCC for Renesas 8.3.0.202104-GNURX Windows Toolchain (ELF).
2. Open the executable and install the compiler.

3.3 Establishing a serial connection

Serial communication is used for communication. Configure the serial port settings as follow:

- Baud rate: 115200bps
- Data: 8 bits
- Parity: none
- Stop: 1 bit
- Flow control: none

3.4 Other software required to develop and debug applications for the device

3.4.1 To download Tera Term

Go to <https://ttssh2.osdn.jp/index.html.en> to download the software.

4. Set up your hardware

In this section, provide instructions for setting up the platform's hardware.

Note that administrator privileges are required to install the drivers.

4.1 To connect the RYZ014A

1. Connect the LTE antenna to the RYZ014A. The antenna is packaged with the purchase of RYZ014A.

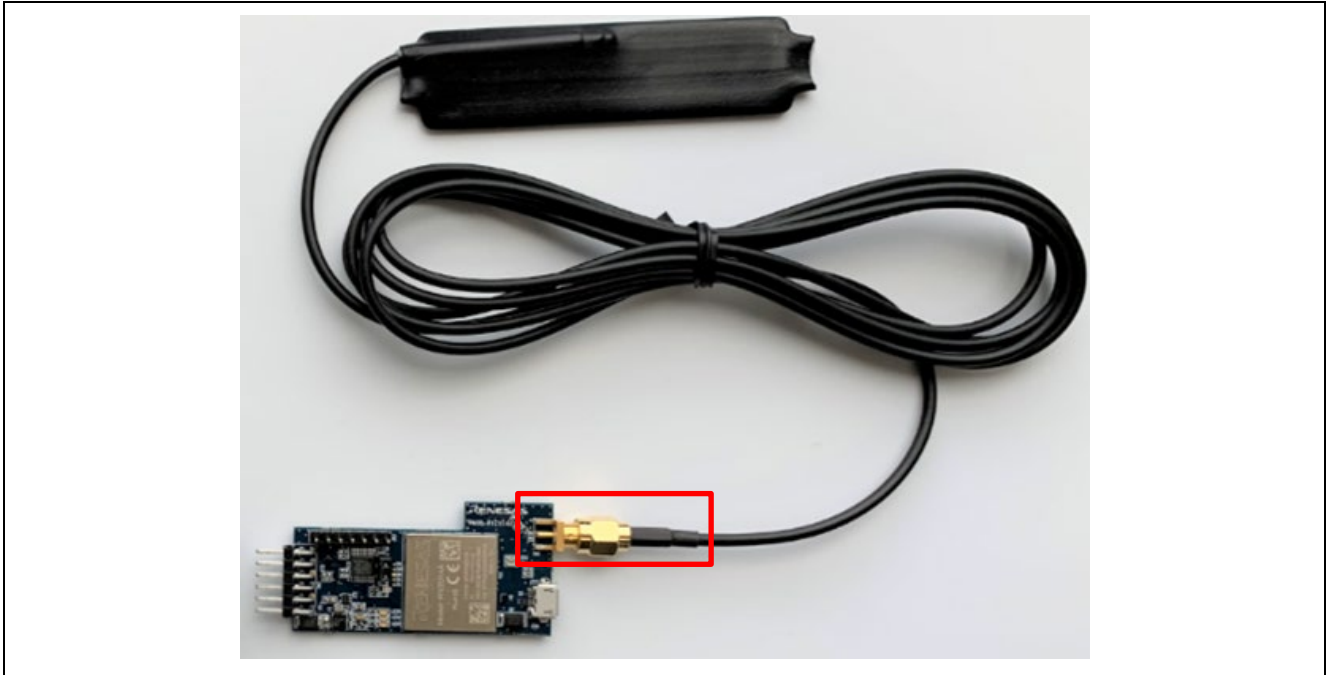


Figure 4-1 Connect the LTE antenna to the RYZ014A

2. The SIM card is required to be activated in advance (refer to your SIM card's guide), then insert it to the RYZ014A.

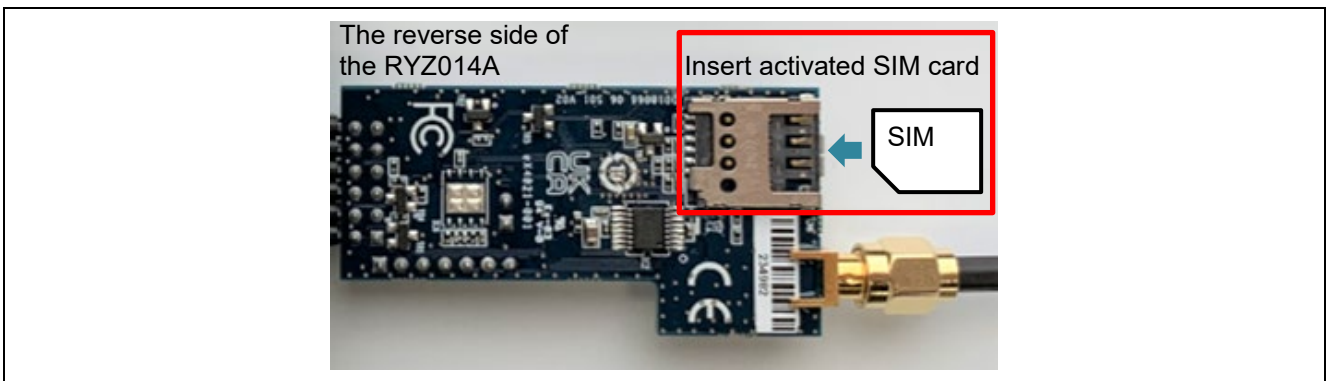


Figure 4-2 Insert activated SIM card to the CK-RX65N

3. Connect the RYZ014A to the CK-RX65N. The RYZ014A connects to PMOD1.

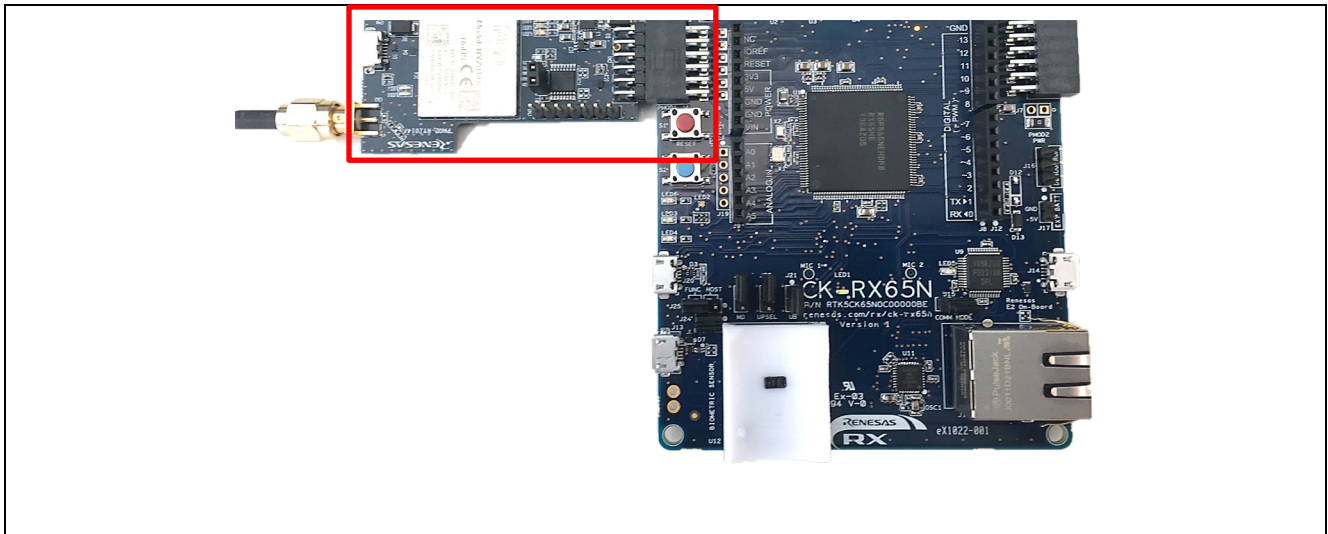


Figure 4-3 Connect the RYZ014A to the CK-RX65N

4. Connect a USB cable to the RYZ014A for supplemental power supply.

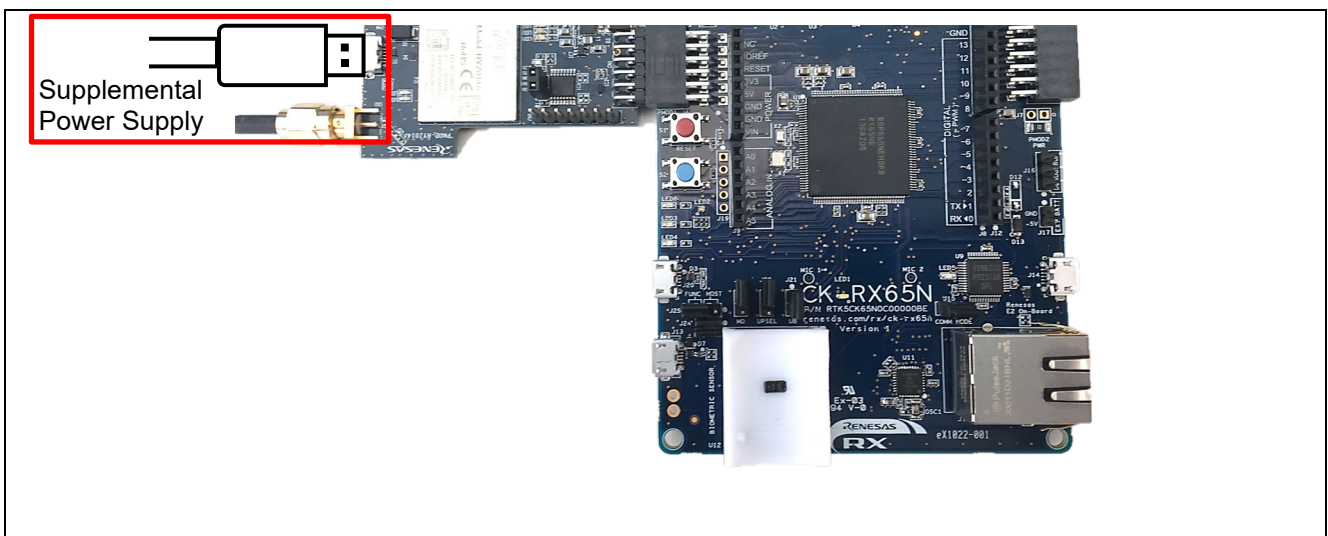


Figure 4-4 Connect a USB cable to the RYZ014A for supplemental power supply

4.2 To set up the device board

Set up jumpers of device board as follows:

- J2 Short for invalidating MCU current measurement point
- J11 Open for single chip mode as MCU boot option
- J16 1-2 Short for debugging

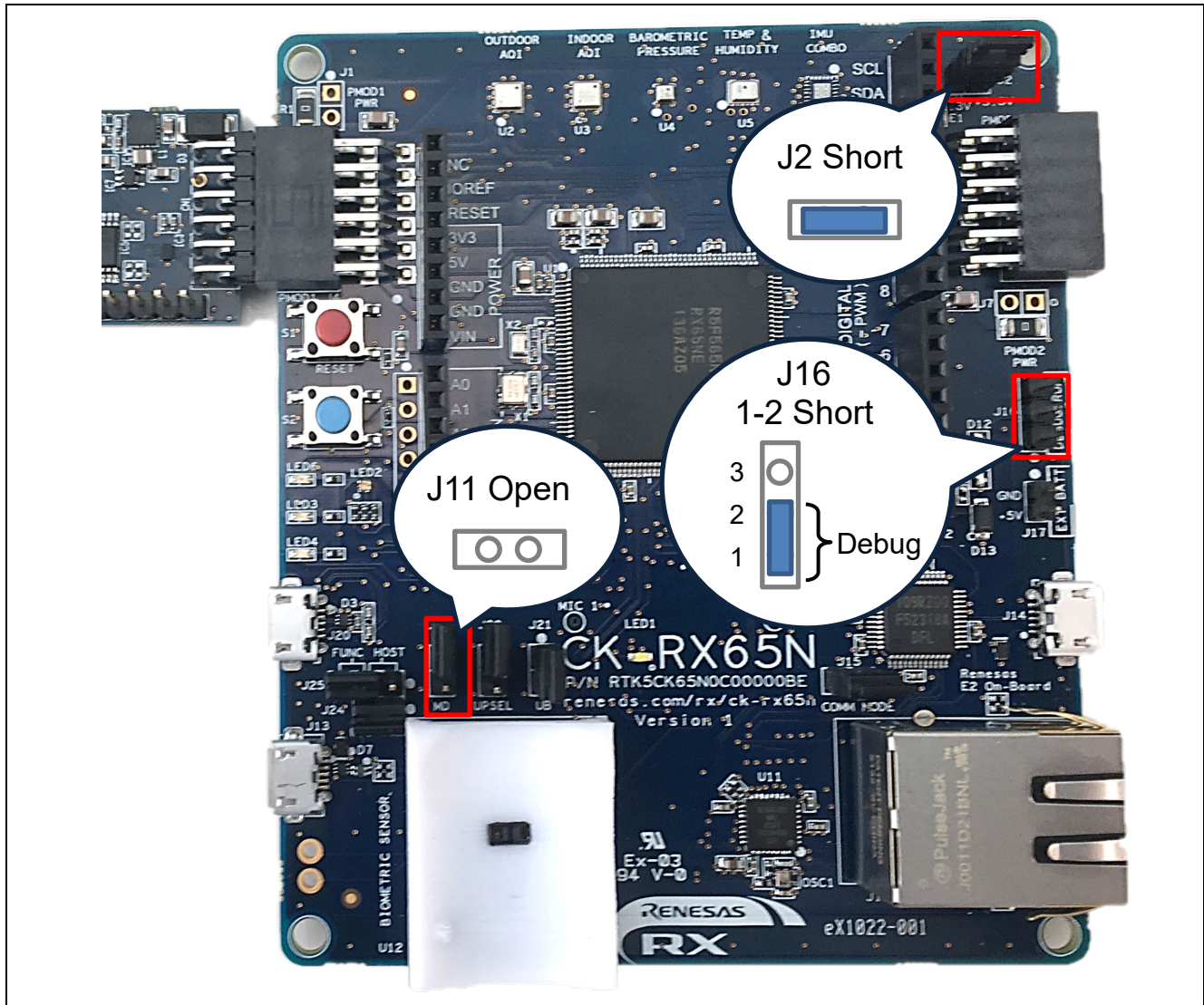


Figure 4-5 Set up the CK-RX65N

4.3 To connect for debugging and power supply

For debugging, connect an USB cable between the debug connector (J14) on the CK-RX65N and your PC with installed e² studio. This connector also has a role of power supply.

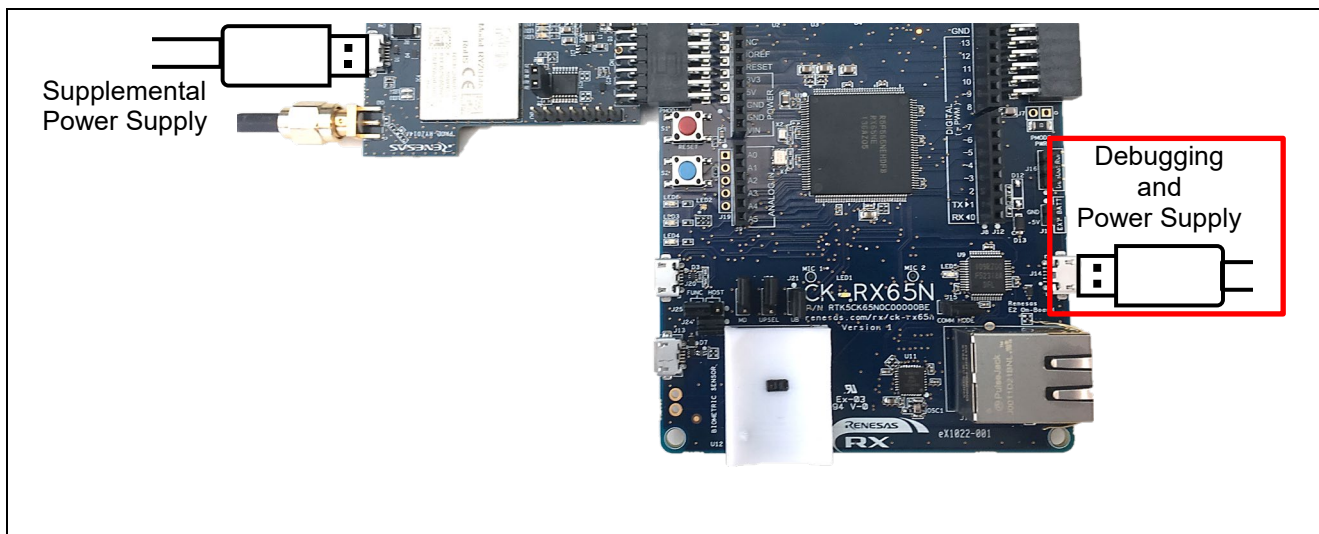


Figure 4-6 Connect an USB cable for debugging

4.4 To connect for receiving debug logs

For receiving debug logs, connect an USB cable between the USB-serial connector (J20) on the CK-RX65N and your PC to receive logs.

Note that debug logs mean serial output data which is coded in aws demo and Renesas driver software. It's not directly related to debugging on e² studio indicated in section 4.3.

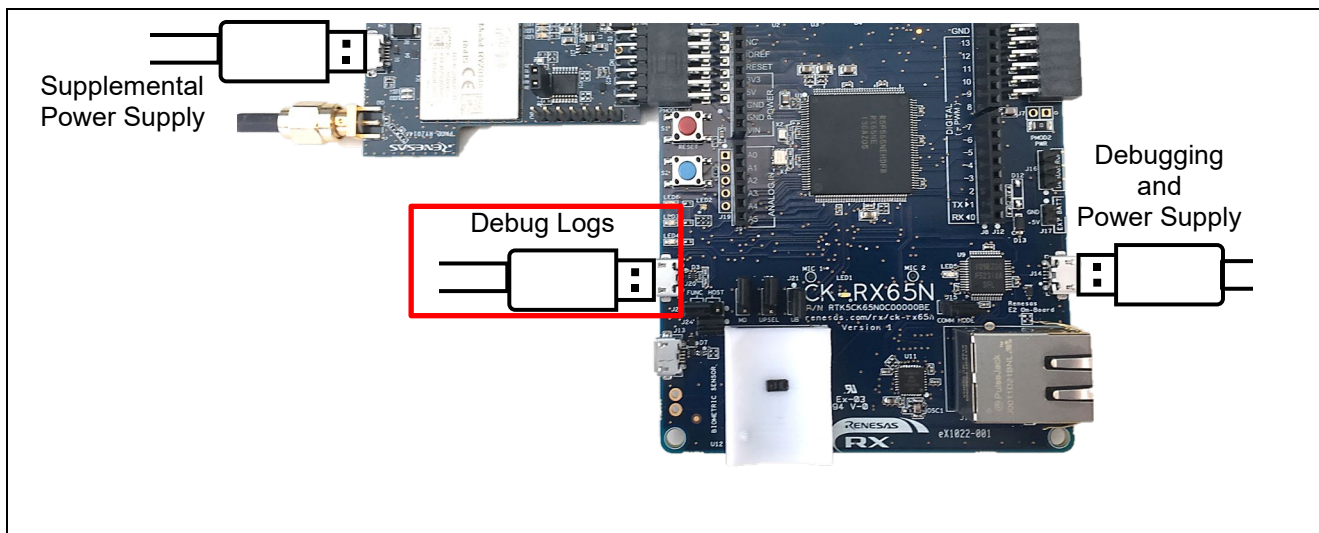


Figure 4-7 Connect an USB cable for receiving debug logs

5. Setup your AWS account and Permissions

To create an AWS account, see [Create and Activate an AWS Account](#)

User needs to create AWS account. Refer to the instructions at [Set up your AWS Account](#). Follow the steps outlined in these sections to create your account and a user and get started:

- Sign up for an AWS account.
- Create a user and grant permissions.
- Open the AWS IoT console.

Pay special attention to the Notes.

If user created AWS account already in the past, please skip this step.

To add an IAM user to your AWS account, see [IAM User Guide](#). To grant your IAM user account access to AWS IoT and FreeRTOS, attach the following IAM policies to your IAM user account:

- AmazonFreeRTOSFullAccess
- AWSIoTFullAccess

Note:

The policy examples in this document are intended only for dev environments. All devices in your fleet must have credentials with privileges that authorize only intended actions on specific resources. The specific permission policies can vary for your use case. Identify the permission policies that best meet your business and security requirements. For more information, refer to [Example policies](#) and [Security best practices](#).

For more information about IAM and user accounts, see [IAM User Guide](#).

For more information about policies, see [IAM Permissions and Policies](#).

5.1 To attach the AmazonFreeRTOSFullAccess policy to your IAM user

1. Browse to the [IAM console](#), and from the navigation pane, choose **Users**.
2. Enter your user name in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter AmazonFreeRTOSFullAccess, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

5.2 To attach the AWSIoTFullAccess policy to your IAM user

1. Browse to the [IAM console](#), and from the navigation pane, choose **Users**.
2. Enter your username in the search text box, and then choose it from the list.
3. Choose **Add permissions**.
4. Choose **Attach existing policies directly**.
5. In the search box, enter AWSIoTFullAccess, choose it from the list, and then choose **Next: Review**.
6. Choose **Add permissions**.

6. Provision the device with AWS IoT

Refer to [Registering MCU board](#).

6.1 To create an AWS IoT policy

Follow 1→6 under the heading **To create an AWS IoT policy**.

1. Note that the AWS region for your account can also be found in the drop-down between the account name and Support drop-downs in the top menu bar.

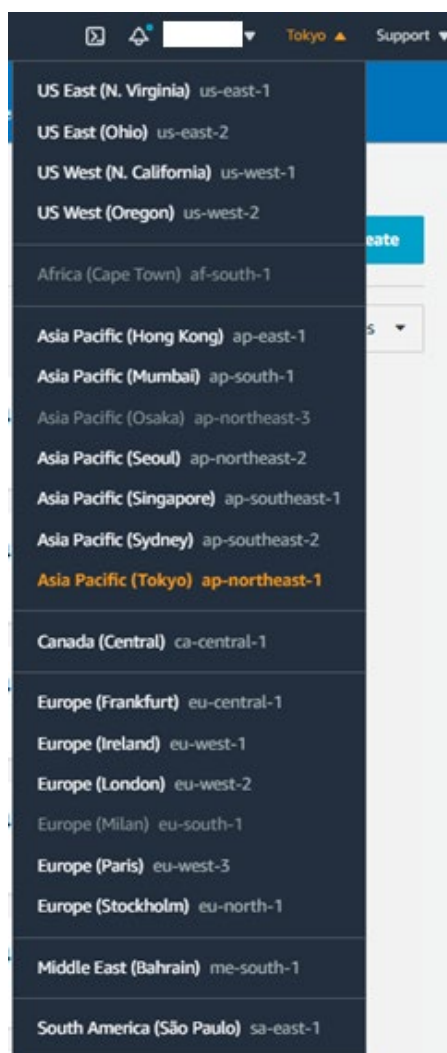


Figure 6-1 AWS region

2. Type IoT Core in search bar and click **IoT Core**.

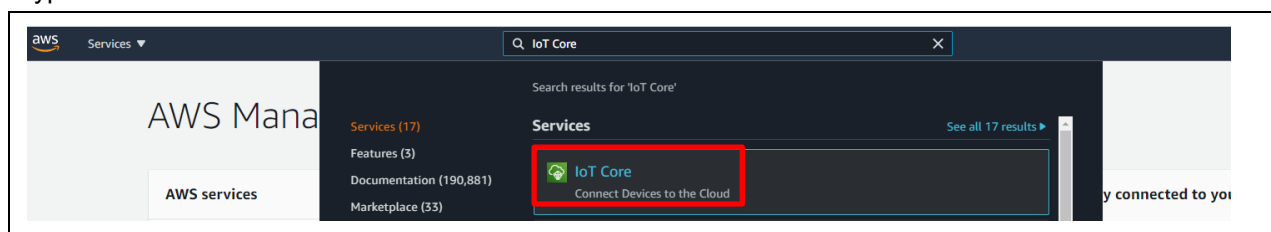


Figure 6-2 AWS IoT Core Selection

3. Go to Secure→Policies.
Click on **Create** to create a policy



Figure 6-3 Create policy

4. In the **Name** field, enter a name for the policy.
Then, change to **Advanced mode**

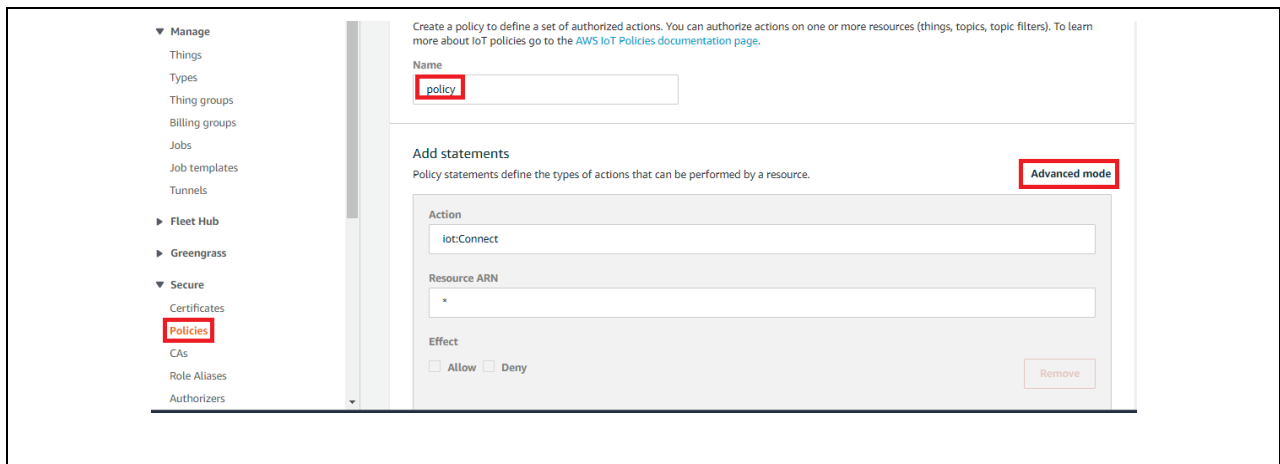


Figure 6-4 Give a policy name

5. Add following text to **Advanced mode**.

```
{
  "Version": "2012-10-17",
  "Statement":
  [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "*"
    }
  ]
}
```

Figure 6-5 Add statements for policy

6. Create a policy.

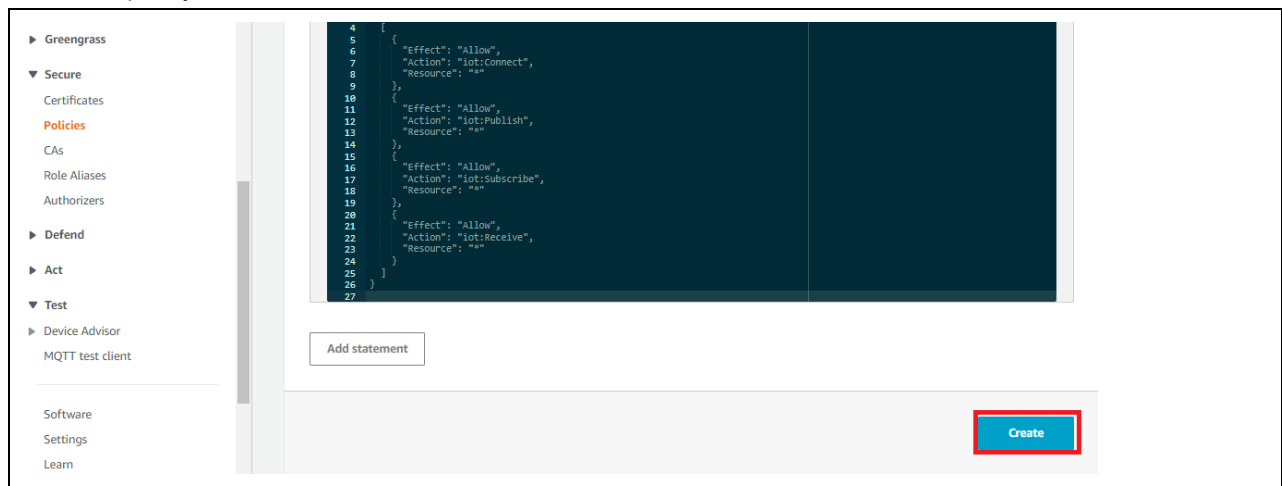


Figure 6-6 Create a policy

Note:

The examples in this document are intended only for dev environments. All devices in your fleet must have credentials with privileges that authorize only intended actions on specific resources. The specific permission policies can vary for your use case. Identify the permission policies that best meet your business and security requirements. For more information, refer to [Example policies](#) and [Security Best practices](#).

6.2 To create an IoT thing, private key, and certificate for your device

Follow steps 1→6 under the heading **To create an IoT thing, private key, and certificate for your device**.

1. Create a Thing

Select **Manage→ Things→Create** to create a thing

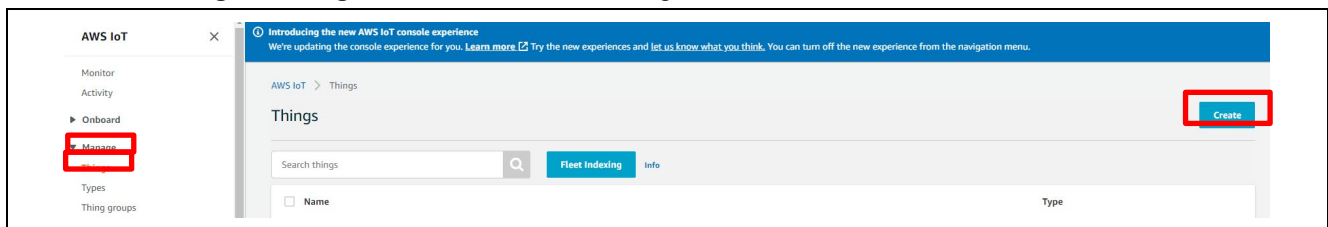
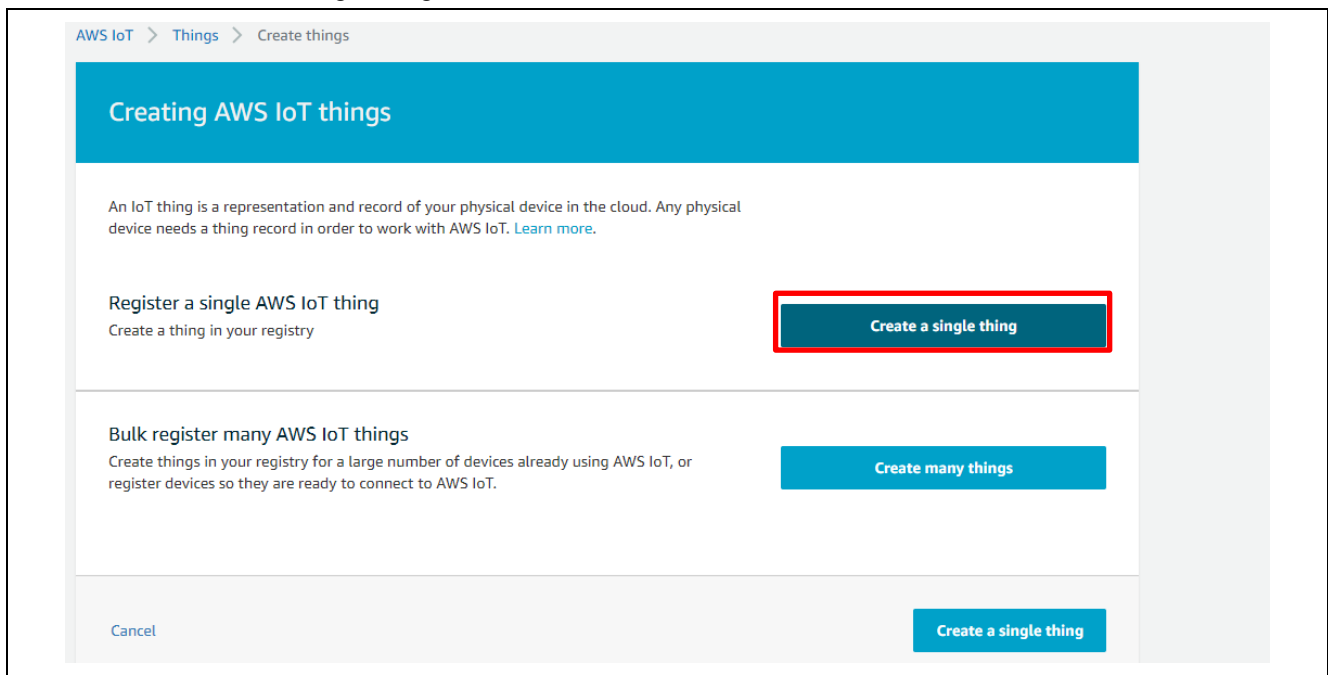


Figure 6-7 Create a thing

2. Select the Create a single thing



AWS IoT > Things > Create things

Creating AWS IoT things

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. [Learn more.](#)

Register a single AWS IoT thing
Create a thing in your registry

Create a single thing

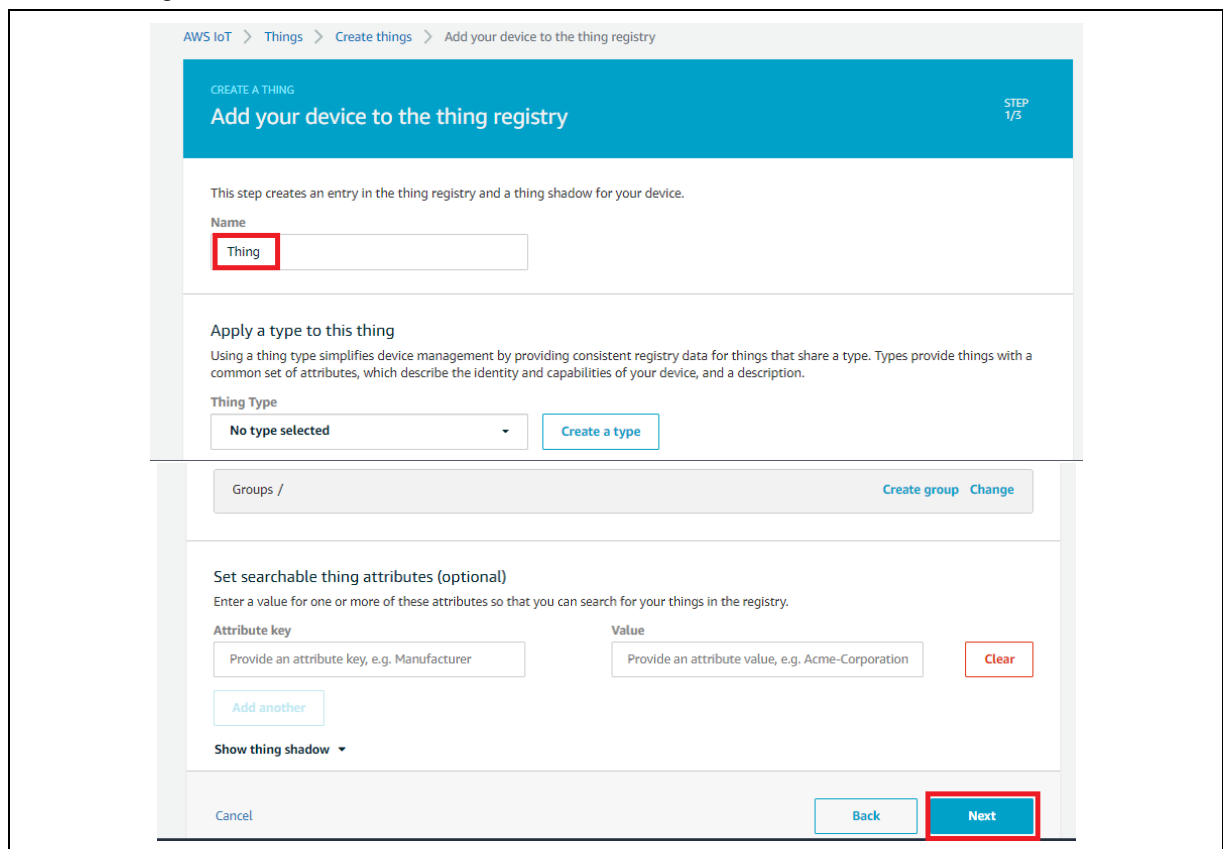
Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

Cancel **Create a single thing**

Figure 6-8 Create a single thing

3. Add name to thing and **Next**



AWS IoT > Things > Create things > Add your device to the thing registry

CREATE A THING

Add your device to the thing registry

STEP 1/3

This step creates an entry in the thing registry and a thing shadow for your device.

Name

Thing

Apply a type to this thing
Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected **Create a type**

Groups / **Create group** **Change**

Set searchable thing attributes (optional)
Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key **Value**

Provide an attribute key, e.g. Manufacturer Provide an attribute value, e.g. Acme-Corporation **Clear**

Add another

Show thing shadow ▼

Cancel **Back** **Next**

Figure 6-9 Add name to a single thing

4. Add a certificate for thing

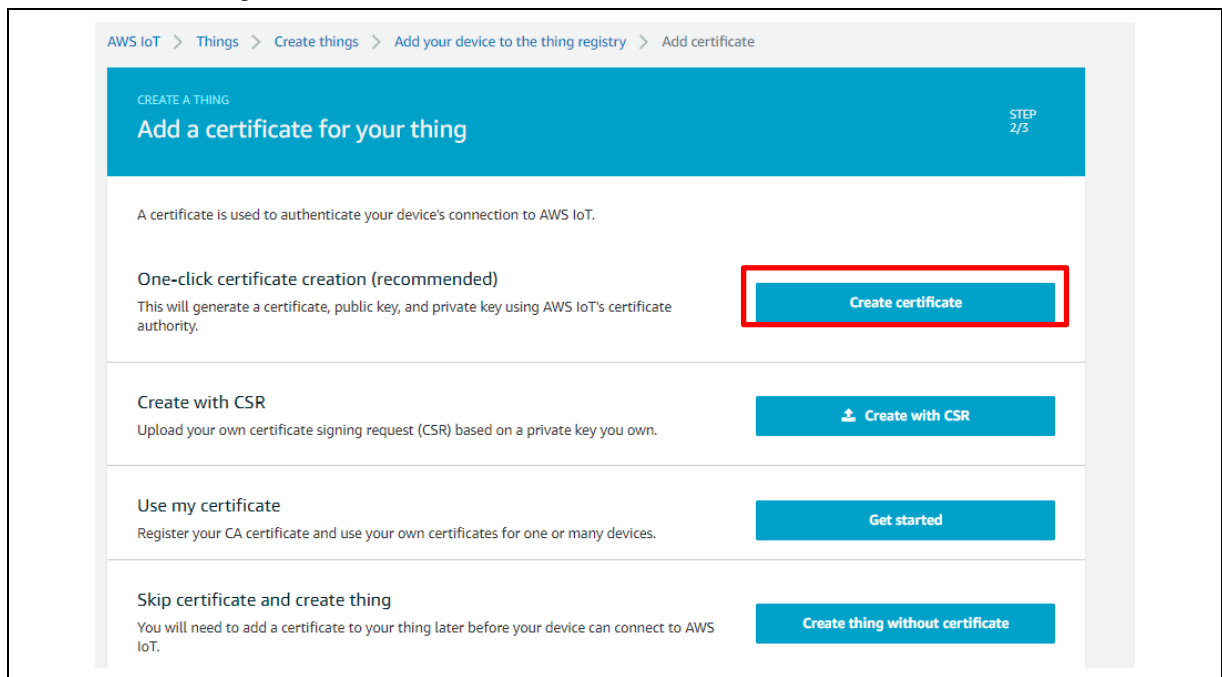


Figure 6-10 Create a certificate for thing

5. Attach a policy to thing:

- Click the **Download** button next to each of the certificates, keys and save in local PC or host machine.
- Click the **Activate** button to activate the certificate.
- Select **Attach a policy** and choose the policy you created in 6.

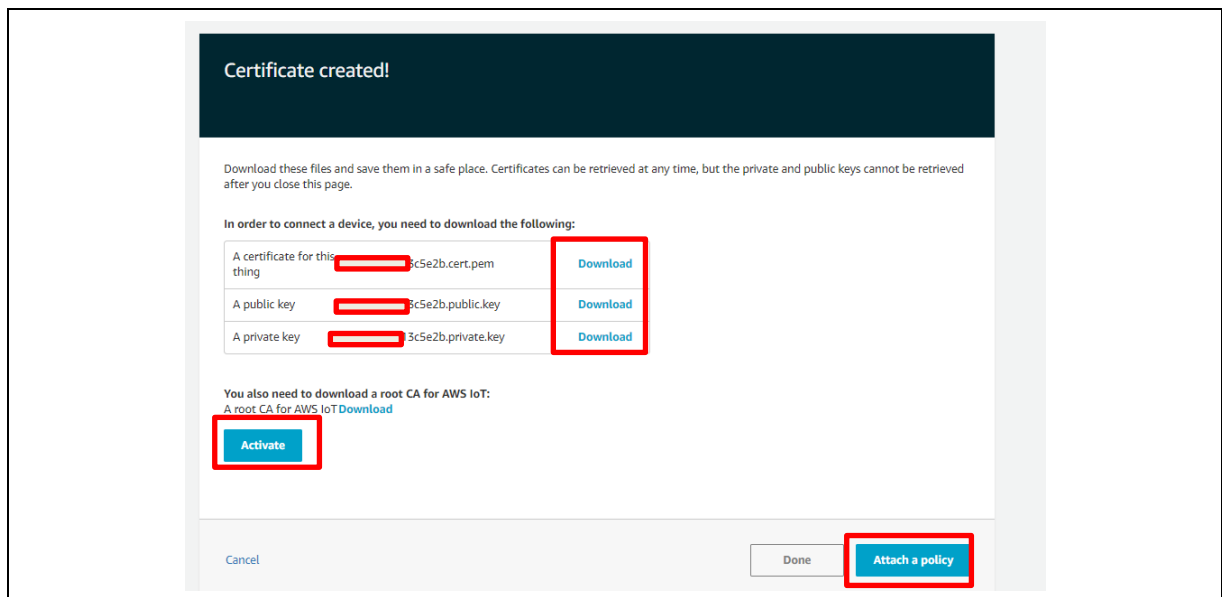
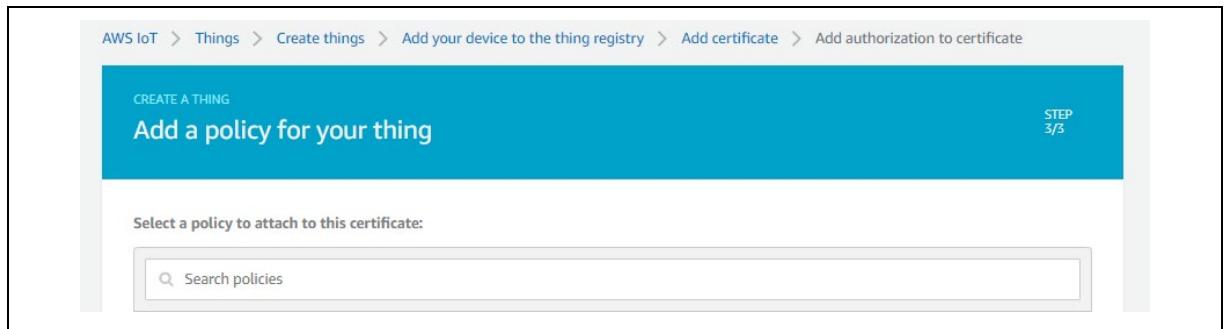


Figure 6-11 Attach a policy

6. Attach policy to thing

**Figure 6-12 Attach policy to thing**

Refer to [Developer-mode key provisioning](#) for an example.

7. Download FreeRTOS

1. Launch e² studio from the Start menu.
2. On the **Select a directory as a workspace** window, browse to the folder that you want to work in, and choose **Launch**.
3. The first time you open e² studio, the **Toolchain Registry** window opens. Choose **Renesas Toolchains** and confirm that **CC-RX v3.04.00** or **GCC for Renesas 8.3.0.202104**.
4. GNURX is selected. Choose **Register**, and then choose **OK**.
5. If you are opening e² studio for the first time, the **Code Generator Registration** window appears. Choose **OK**.
6. The **Code Generator COM component register** window appears. Under **Please restart e² studio to use Code Generator**, choose **OK**.
7. The **Restart e² studio** window appears. Choose **OK**.
8. e² studio restarts. On the **Select a directory as a workspace** window, choose **Launch**.
9. On the e² studio welcome screen, choose the **Go to the e² studio workbench** arrow icon.
10. Right-click the **Project Explorer** window and choose **Import**.
11. In the import wizard, choose **General**, **Renesas GitHub FreeRTOS (with IoT libraries) Project**, and the choose **Next**.

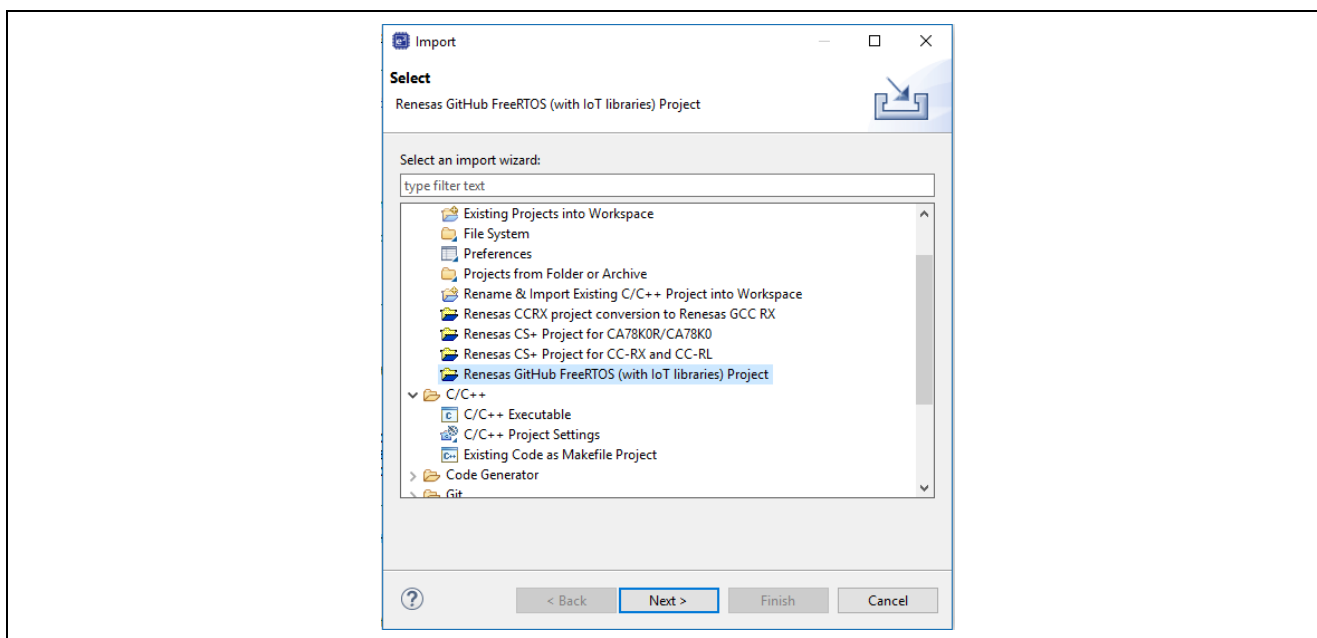


Figure 7-1 General, Renesas GitHub FreeRTOS (with IoT libraries) Project

12. Choose **Browse** to specify a folder to copy downloaded RTOS content in order to import project.

13. In RTOS version setting, choose **Check for more version...** to see a list of all supported RTOS version. On the **FreeRTOS (with IoT libraries) Module Download** window, select the FreeRTOS version (recommended: **v202107.00-rx-1.0.0**) you want to work on by clicking the checkbox, then choose **Download**.

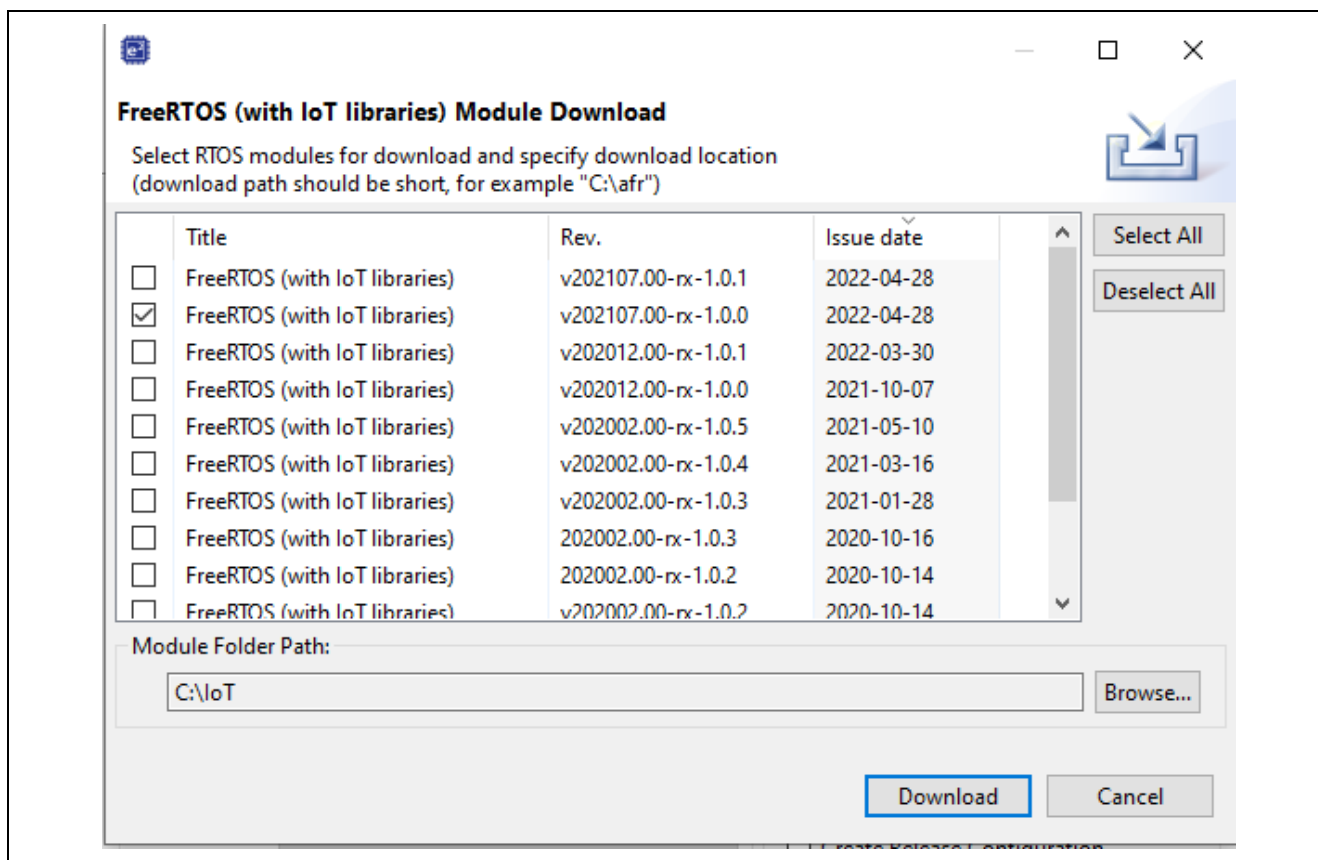


Figure 7-2 FreeRTOS (with IoT libraries) Module Download

14. Once download is completed, choose Next in the Renesas GitHub FreeRTOS (with IoT libraries) Project window.
15. If you are *not* using an empty folder, the Copy Resources warning message appears. Choose Yes.
16. Choose the project aws_demos according to the following table, then choose Finish:

Table 7-1 The path of aws_demos project

Communication	Compiler	Path of project
Ethernet	CC-RX	\${FOLDER_DIR}/projects/renesas/ck-rx65n/e2studio/aws_demos
	GCC for RX	\${FOLDER_DIR}/projects/renesas/ck-rx65n/e2studio-gcc/aws_demos
Cellular	CC-RX	\${FOLDER_DIR}/projects/renesas/ck-rx65n-ryz014a/e2studio/aws_demos
	GCC for RX	\${FOLDER_DIR}/projects/renesas/ck-rx65n-ryz014a/e2studio-gcc/aws_demos

8. Configure FreeRTOS

Follow the instructions under the heading [Configuring the FreeRTOS demos](#) for more details.

8.1 Header file for AWS IoT

8.1.1 Ethernet case

— AWS client credential settings

Check `aws_demos/demos/include/aws_clientcredential.h` and confirm 2 settings:

- `clientcredentialMQTT_BROKER_ENDPOINT`
- `clientcredentialIOT_THING_NAME`

For “`clientcredentialIOT_THING_NAME`”, input name of the thing you created in 6.2.

```

* @todo Set this to the fully-qualified DNS name of your MQTT broker.
*/
#define clientcredentialMQTT_BROKER_ENDPOINT ""

/*
* @brief Host name.
*
* @todo Set this to the unique name of your IoT Thing.
* Please note that for convenience of demonstration only we
* are using a #define here. In production scenarios the thing
* name can be something unique to the device that can be read
* by software, such as a production serial number, rather
* than a hard coded constant.
*/
#define clientcredentialIOT_THING_NAME ""

/*
* @brief Port number the MQTT broker is using.
*/
#define clientcredentialMQTT_BROKER_PORT 8883

/*
* @brief Port number the Green Grass Discovery use for JSON retrieval from cloud is using.
*/
#define clientcredentialGREENGRASS_DISCOVERY_PORT 8443

/*
* @brief Wi-Fi network to join.
*
* @todo If you are using Wi-Fi, set this to your network name.
*/
#define clientcredentialWIFI_SSID ""

/*
* @brief Password needed to join Wi-Fi network.
* @todo If you are using WPA, set this to your network password.
*/
#define clientcredentialWIFI_PASSWORD ""
/*

```

Figure 8-1 Ethernet `aws_clientcredential.h`

8.1.2 Cellular case

— AWS client credential settings

Check `aws_demos/demos/include/aws_clientcredential.h` and confirm 4 settings:

- `clientcredentialMQTT_BROKER_ENDPOINT`
- `clientcredentialIOT_THING_NAME`
- `clientcredentialWIFI_SSID`
- `clientcredentialWIFI_PASSWORD`

For “`clientcredentialIOT_THING_NAME`”, input name of the thing you created in 6.2.

Specify “`dummy`” for the following `#define` constants: `clientcredentialWIFI_SSID` and `clientcredentialWIFI_PASSWORD`.

```
/*
 * @brief MQTT Broker endpoint.
 *
 * @todo Set this to the fully-qualified DNS name of your MQTT broker.
 */
#define clientcredentialMQTT_BROKER_ENDPOINT ""

/*
 * @brief Host name.
 *
 * @todo Set this to the unique name of your IoT Thing.
 * Please note that for convenience of demonstration only we
 * are using a #define here. In production scenarios the thing
 * name can be something unique to the device that can be read
 * by software, such as a production serial number, rather
 * than a hard coded constant.
 */
#define clientcredentialIOT_THING_NAME ""

/*
 * @brief Port number the MQTT broker is using.
 */
#define clientcredentialMQTT_BROKER_PORT 8883

/*
 * @brief Port number the Green Grass Discovery use for JSON retrieval from cloud is using.
 */
#define clientcredentialGREENGRASS_DISCOVERY_PORT 8443

/*
 * @brief Wi-Fi network to join.
 *
 * @todo If you are using Wi-Fi, set this to your network name.
 */
#define clientcredentialWIFI_SSID "dummy"

/*
 * @brief Password needed to join Wi-Fi network.
 * @todo If you are using WPA, set this to your network password.
 */
#define clientcredentialWIFI_PASSWORD "dummy"
```

Figure 8-2 Cellular `aws_clientcredential.h`

— SIM Card settings

Set up the 4 macros related to LTE in [aws_demos/vendors/renesas/boards/ck-rx65n-ryz014a/aws_demos/src/smc_gen/r_config/r_cellular_config.h](#) in the demo project.

The settings will vary depending on the SIM you are using, so please check the SIM information.

- CELLULAR_CFG_AP_NAME : Set the access point name of the SIM
- CELLULAR_CFG_AP_USERID : Set the user name of the SIM
- CELLULAR_CFG_AP_PASSWORD : Set sim card password
- CELLULAR_CFG_PIN_CODE : Set SIM card PIN code

```

Configuration Options
#define CELLULAR_CFG_AP_NAME      ibasis.iot      /* Access point name */
#define CELLULAR_CFG_AP_USERID    /* Login ID */
#define CELLULAR_CFG_AP_PASSWORD  /* Access point password */
#define CELLULAR_CFG_PIN_CODE     /* SIM card PIN code */

```

Figure 8-3 Cellular r_cellular_config.h

Here is the example of the SIM information and the settings associated with it.

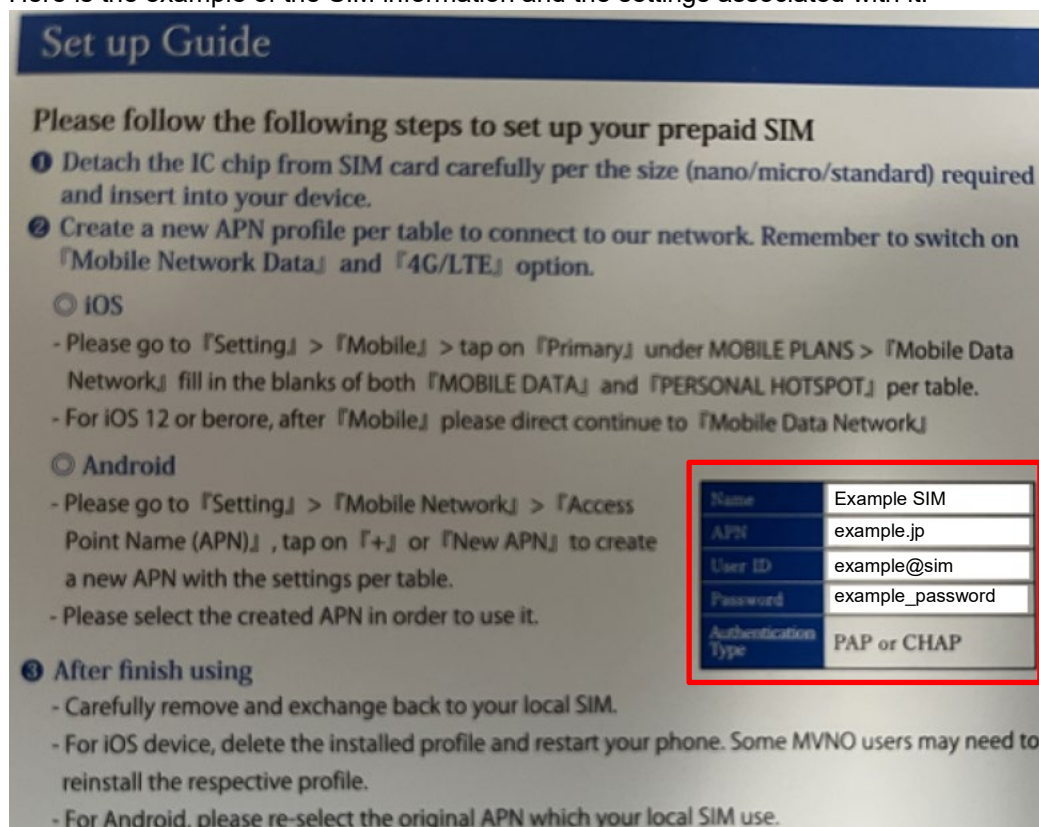


Figure 8-4 The manual of SIM card for an example

In the case of above manual, edit r_cellular_config.h such as the followings (note that the value of macro CELLULAR_CFG_PIN_CODE is empty when information about pin code is nothing):

```

#define CELLULAR_CFG_AP_NAME      example.jp      /* Access point name */
#define CELLULAR_CFG_AP_USERID    example@sim    /* Login ID */
#define CELLULAR_CFG_AP_PASSWORD  example_password /* Access point password */
#define CELLULAR_CFG_PIN_CODE     /* SIM card PIN code */

```

8.2 The endpoint in AWS IoT

To find the endpoint for your account, use the AWS IoT console at console.aws.amazon.com/iot. In the left panel, choose Settings. The endpoint is listed under Custom endpoint as following snapshot:

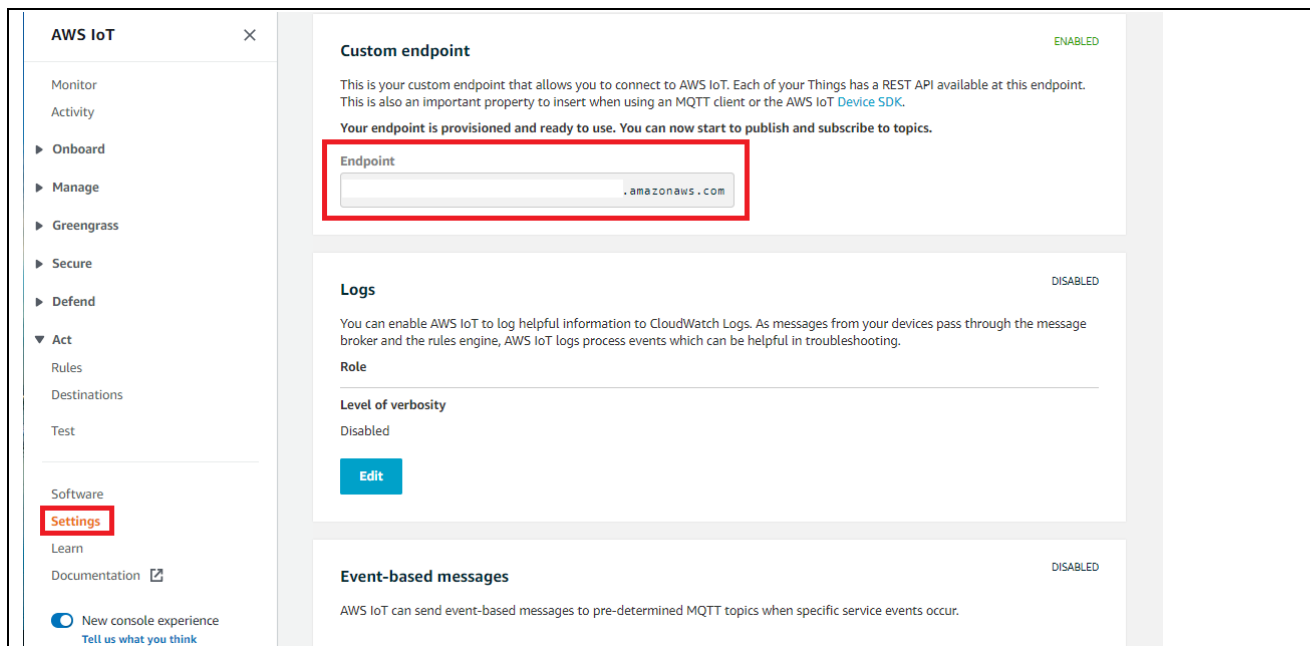


Figure 8-5 The endpoint in AWS IoT

9. Build the FreeRTOS demo

Go to e² studio, from **Project** menu, choose **Build All**



10. Run the FreeRTOS demo project

10.1 Flash demo code

Step-by-step instructions to flash and run the FreeRTOS demo code:

1. Confirm that you have connected your computer to the USB-to-serial port on CK-RX65N.
2. From the top menu, choose **Run, Debug Configurations....**
3. Expand **Renesas GDB Hardware Debugging** and choose **aws_demos HardwareDebug**.
4. Choose the **Debugger** tab, and then choose the **Connection Settings** tab. Confirm that your connection settings are correct.
5. Choose **Debug** to download the code to your board and begin debugging.
You might be prompted by a firewall warning for `e2-server-gdb.exe`. Check Private networks, such as my home or work network, and then choose Allow access.
6. e² studio might ask to change to **Renesas Debug Perspective**. Choose **Yes**.
7. After the code is downloaded to the board, choose **Resume** to run the code up to the first line of the main function. Choose **Resume** again to run the rest of the code.

10.2 Receive MQTT messages by AWS IoT

Step-by-step instructions to verify that the FreeRTOS demo works successfully and MQTT messages are being received by AWS IoT.

You can use the MQTT client in the AWS IoT console to monitor the messages that your device sends to the AWS Cloud.

To subscribe to the MQTT topic with the AWS IoT MQTT client

1. Sign in to the [AWS IoT console](#).
2. In the navigation pane, choose **Test** and then MQTT Test Client to open the MQTT client.
3. In **Subscription topic**, enter `iotdemo/#`, and then choose **Subscribe to topic**.
4. Successful demo run looks like following the picture.

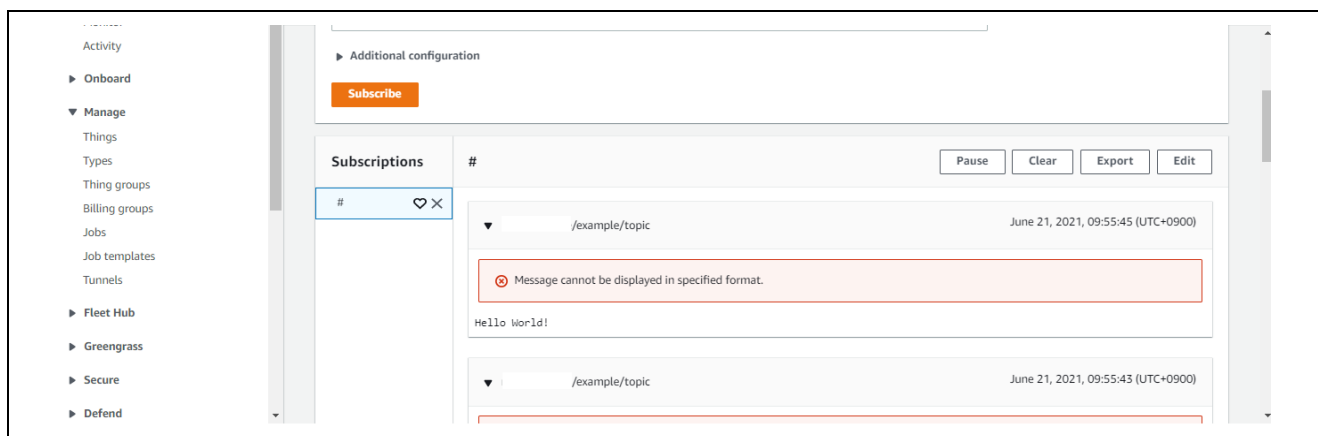


Figure 10-1 Successful demo run

For the latest projects released by Renesas, see the [renesas](#) fork of the [amazon-freertos](#) repository on [GitHub](#).

11. Debugging

11.1 Open e² studio to debug

Make sure that debug configuration is same as the following setting.

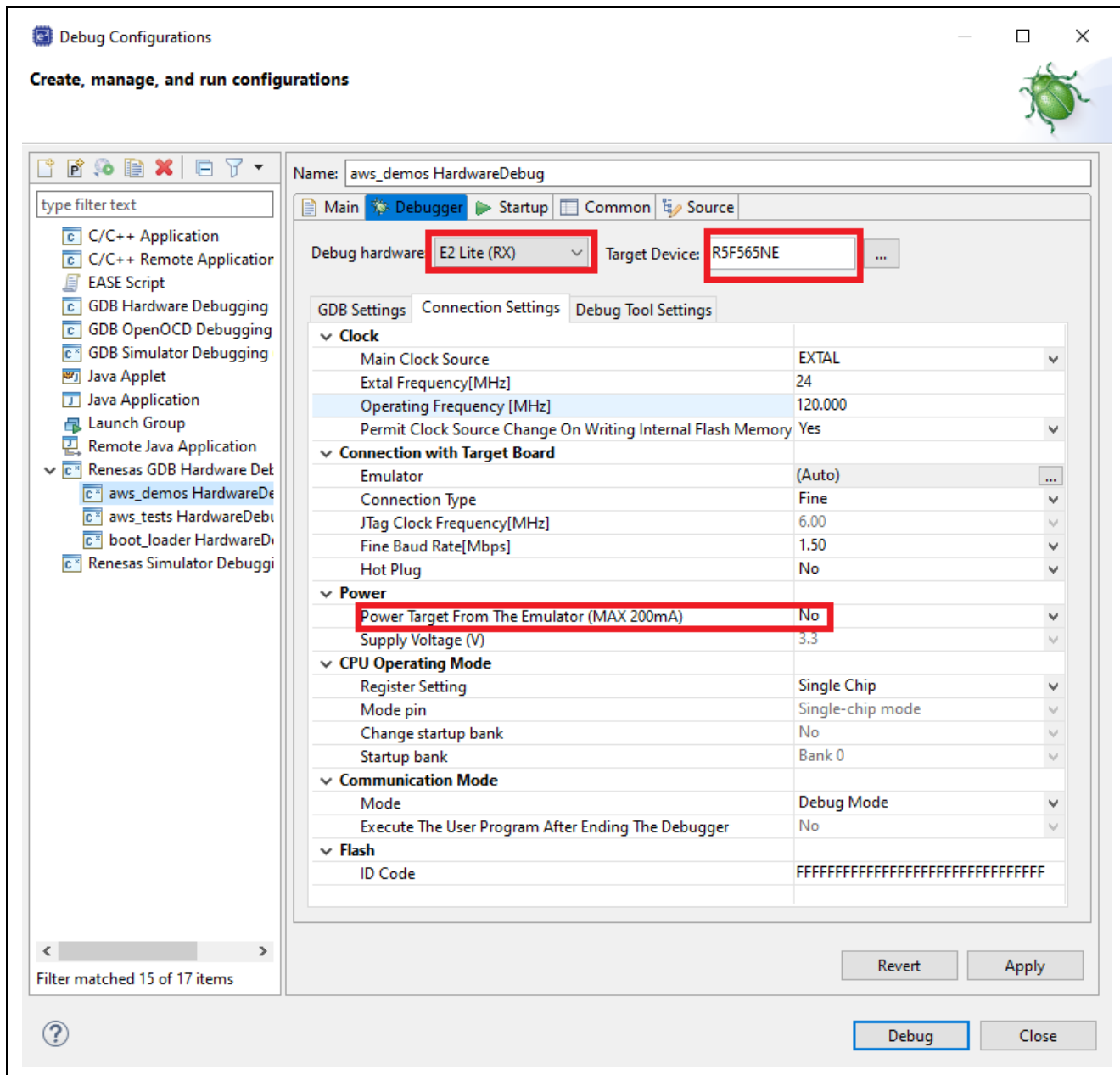


Figure 12-1 e² studio debug setting

11.2 Tera term

Open tera term to check port, baud rate, Data, Parity, Stop and Flow control.

- Baud rate: 115200bps
- Data: 8 bit
- Parity: none
- Stop: 1 bit
- Flow control: none

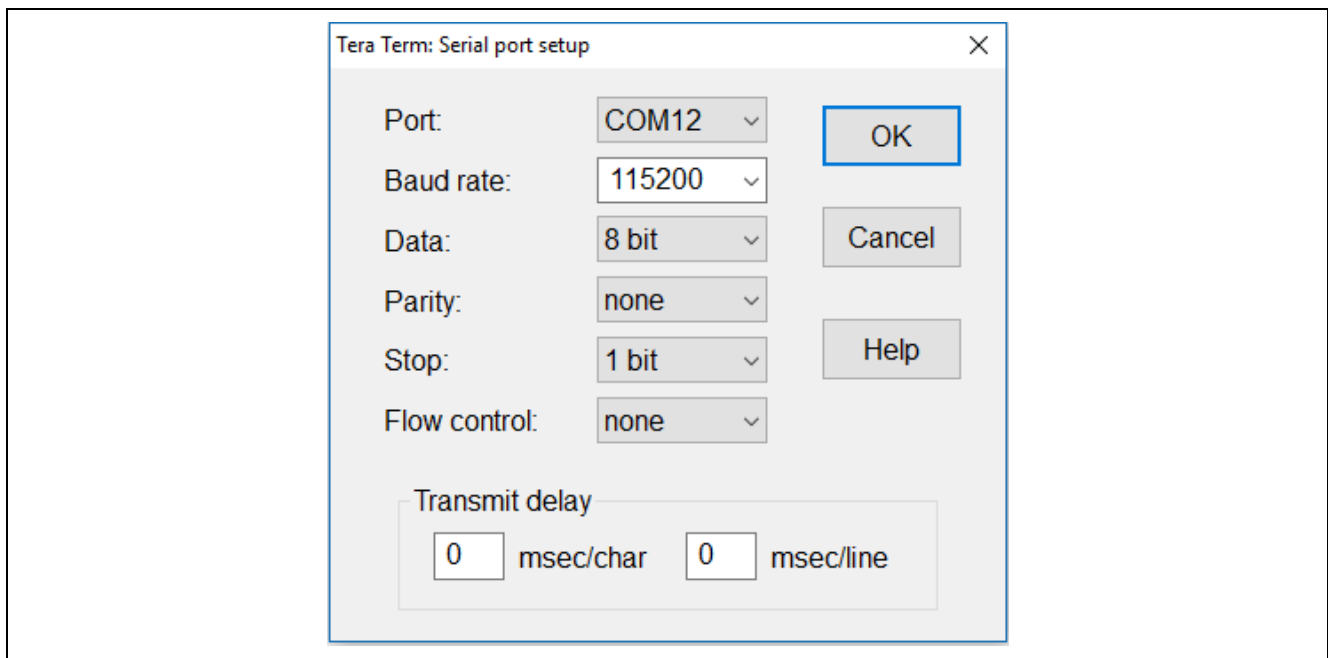


Figure 12-2 Tera term setting

12. Troubleshooting

12.1 The Build errors

Make sure that [v202107.00-rx-1.0.0](#) is located to C: or D: drive or etc. Windows has a path length limitation of 260 characters. The path structure of FreeRTOS is many levels deep, so if you are using Windows, keep your file paths under the 260-character limit. The build will be passed if file paths under the 260-character.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May.31.22	-	First published

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/.