RX Family

R01AN5549EJ0102
Rev.1.02
May.28.21

How to implement FreeRTOS OTA by using Amazon Web Services on RX65N

Objectives

This document helps users to be familiar with the procedures to use OTA demo applications with FreeRTOS IoT libraries on RX65N. More information related to security, please refer **Renesas MCU Firmware Update Design Policy R01AN5548EJ0100**

Operating Environment

The following is a list of devices that are currently supported:

- RX65N, RX651 Groups

Hardware:

1. RX65N-2MB RSK case
   - Connect E2 Lite emulator and USB serial port to RX65N-2MB RSK to PC
   - Connect power source to RX65N-2MB RSK
2. RX65N Cloud Kit case
   - Connect USB serial port to PC x2
   - Wi-Fi-Pmod-Expansion-Board

Reference:

- Renesas MCU Firmware Update Design Policy (R01AN5548EJ0100)

Contents

1 Set up AWS 3
1.1 Sign in the console 3
1.2 Create an Amazon S3 bucket 7
1.3 Create service role for OTA update 10
1.4 Create an OTA user policy and attach the OTA user policy to your IAM user 11
1.5 Register a code-signing certificate on AWS 12
1.6 Grant access to code signing for AWS IoT 13
2 FreeRTOS OTA environment construction 14
2.1 Import, configurate head file and build aws_demos and boot_loader 14
2.2 Install the initial version of firmware 24
2.3 Update the version of your firmware 33
3 Restriction 38
4 Appendices 39
4.1 Confirmed Operation Environment 39
Revision History 40

1 Set up AWS

To run the FreeRTOS demos, user needs an AWS account, an IAM user with permission to access AWS IoT and FreeRTOS cloud services.

To set up AWS account and permission, please refer to
https://docs.aws.amazon.com/freertos/latest/userguide/freertos-account-and-permissions.html.

Set up for the OTA update, please refer to

https://docs.aws.amazon.com/freertos/latest/userguide/ota-prereqs.html

Next, user needs to register the board with AWS IoT as described at
https://docs.aws.amazon.com/freertos/latest/userguide/get-started-freertos-thing.html.

To make the demo communicate with AWS, user needs to configure the source code as described at section 2

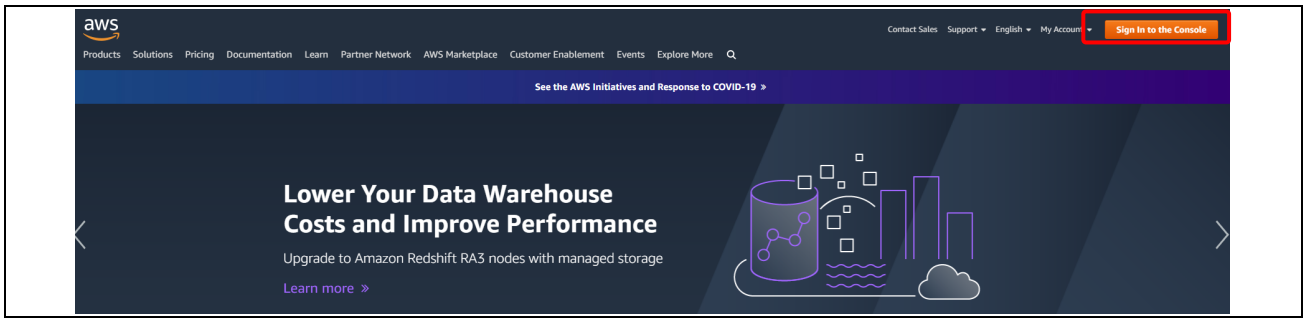
1.1 Sign in the console

① 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:

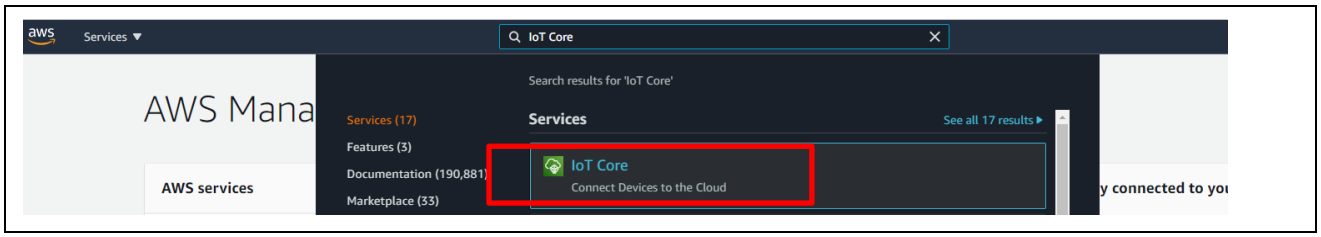
- Sign up for an AWS account.
- Create a user and grant permissions.
- Open the AWS IoT console.

Pay special attention to the Notes.

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

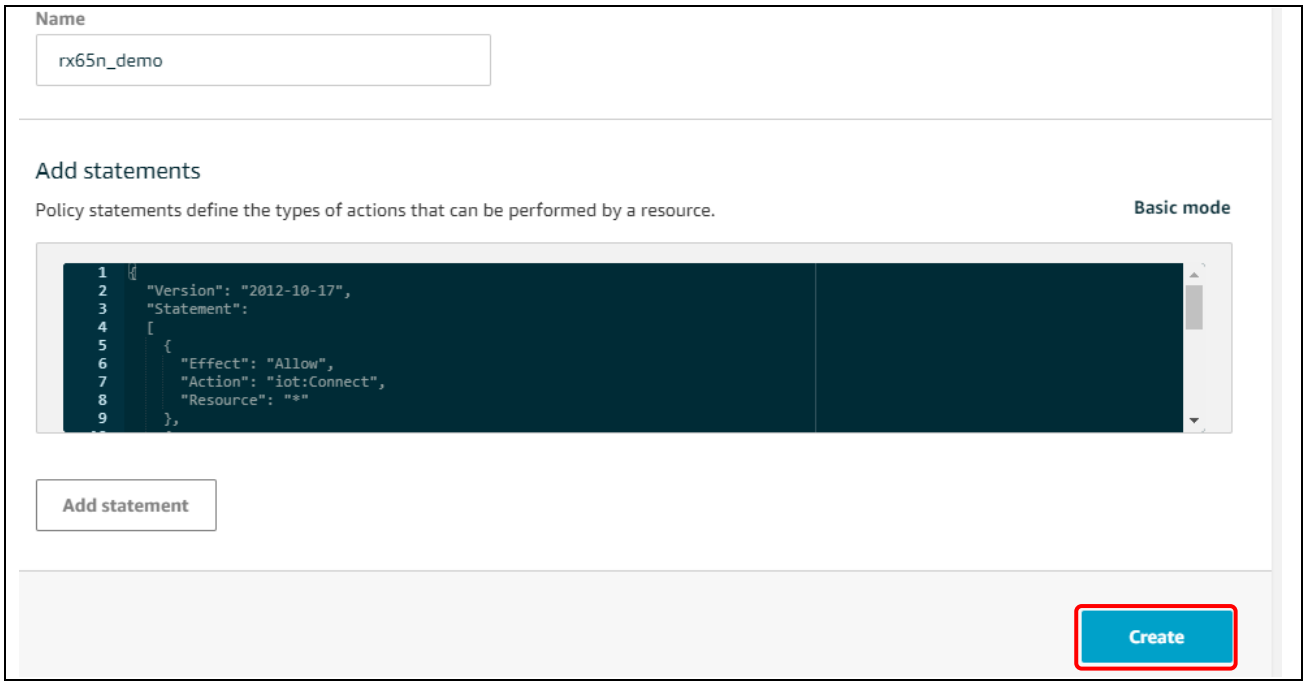


Typing IoT Core in search bar and click **IoT Core**



② Go to Secure→ Policies to create policy

The AWS IoT policy grants device permissions to access AWS IoT resources. It is stored on the AWS Cloud.



③ Choose advance mode and copy the following code

```
{
	"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": "*"
		}
	]
}
```

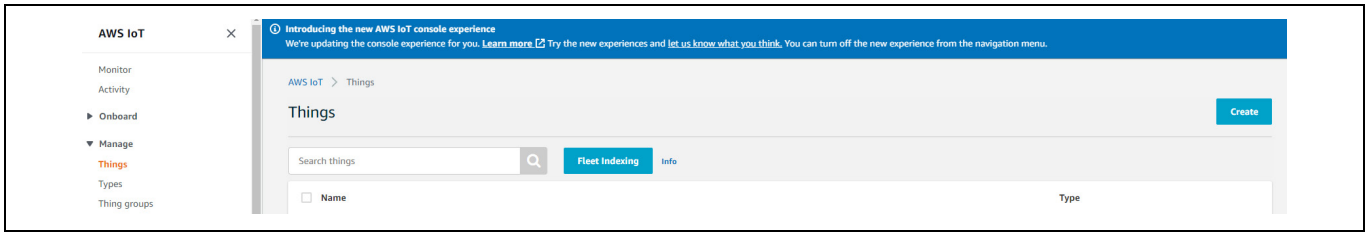
Note: The examples in this document are intended only for dev environments. All devices in your fleet must have credentials with privileges that authorize only intended actions on specific resources. The specific permission policies can vary for your use case. Identify the permission policies that best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

④ Go to Manage→ Things to create Thing

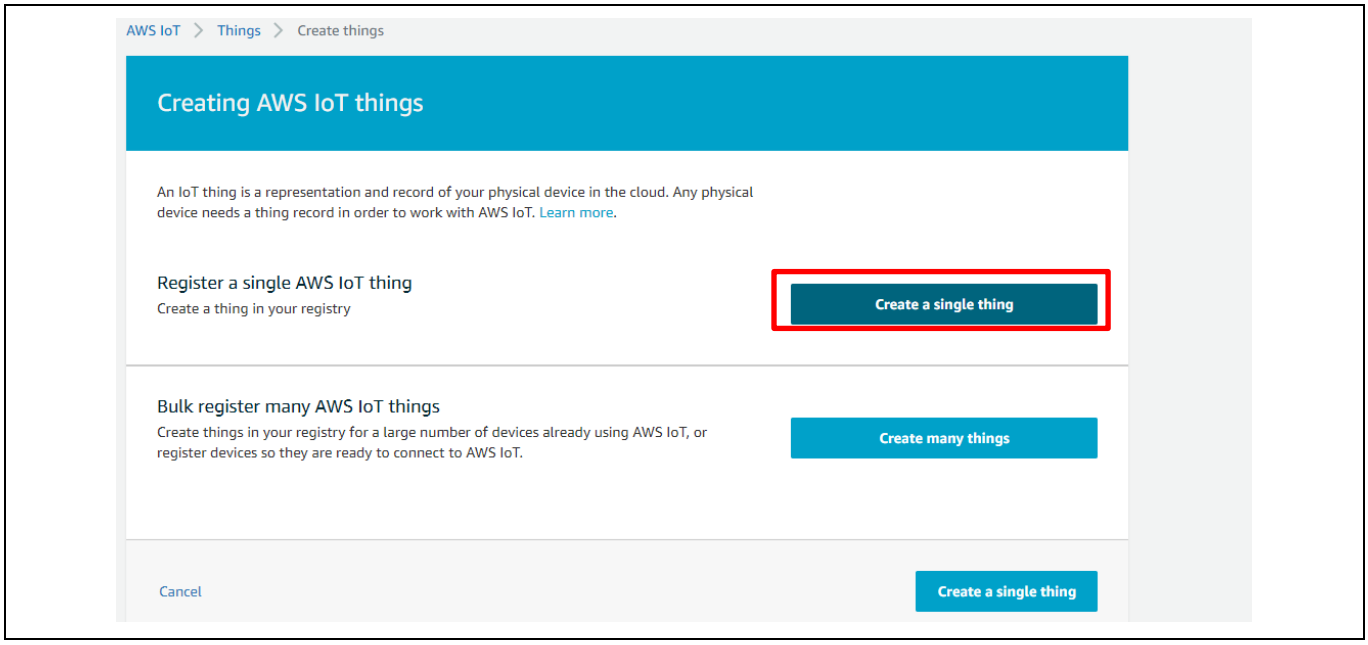
A thing is a representation of a device or logical entity in AWS IoT. It can be a physical device or sensor (for example, a light bulb or a switch on a wall). It can also be a logical entity like an instance of an application or physical entity that does not connect to AWS IoT, but is related to devices that do (for example, a car that has engine sensors or a control panel). AWS IoT provides a thing registry that helps to manage your things.

- Choose Create a single thing
- Give a name for thing
- Click on Create certificate
- Download 3 files
- Attach policy

Select **Manager→ Thing→Create** to create a thing

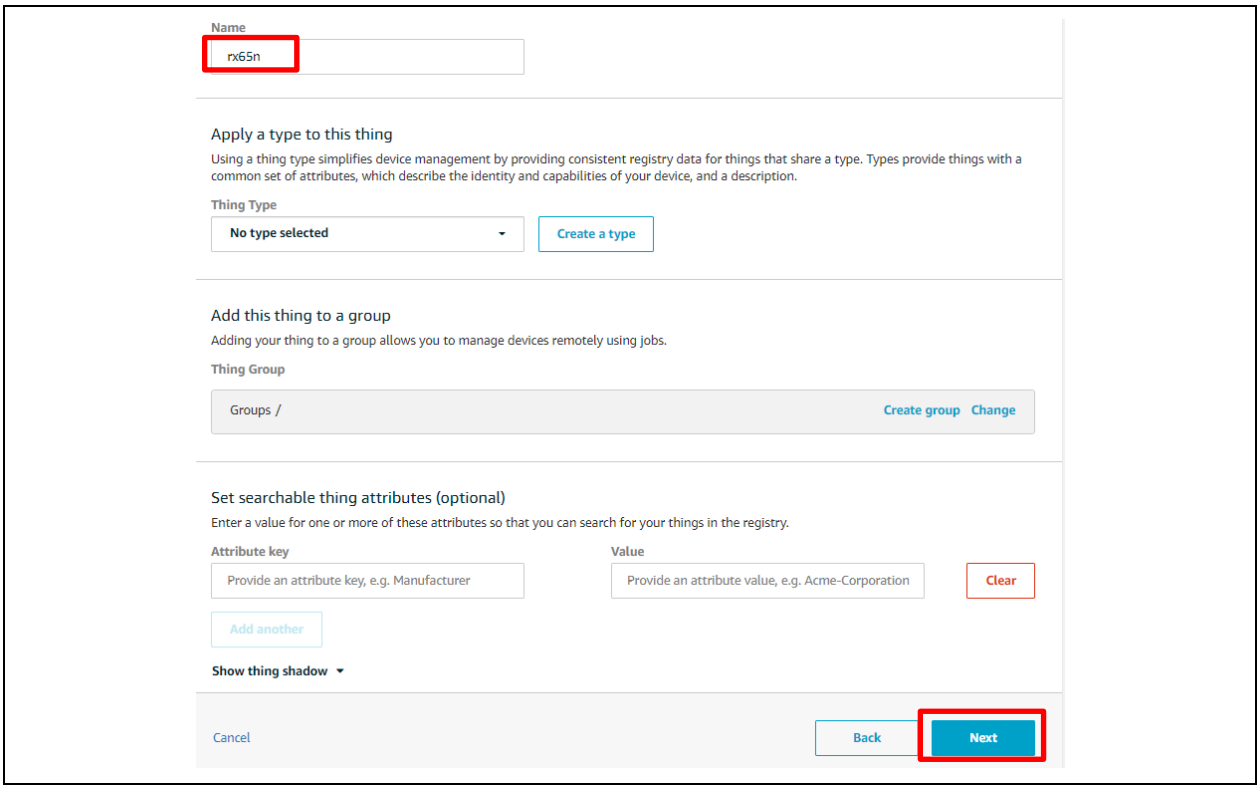


Select the **Create a single thing**



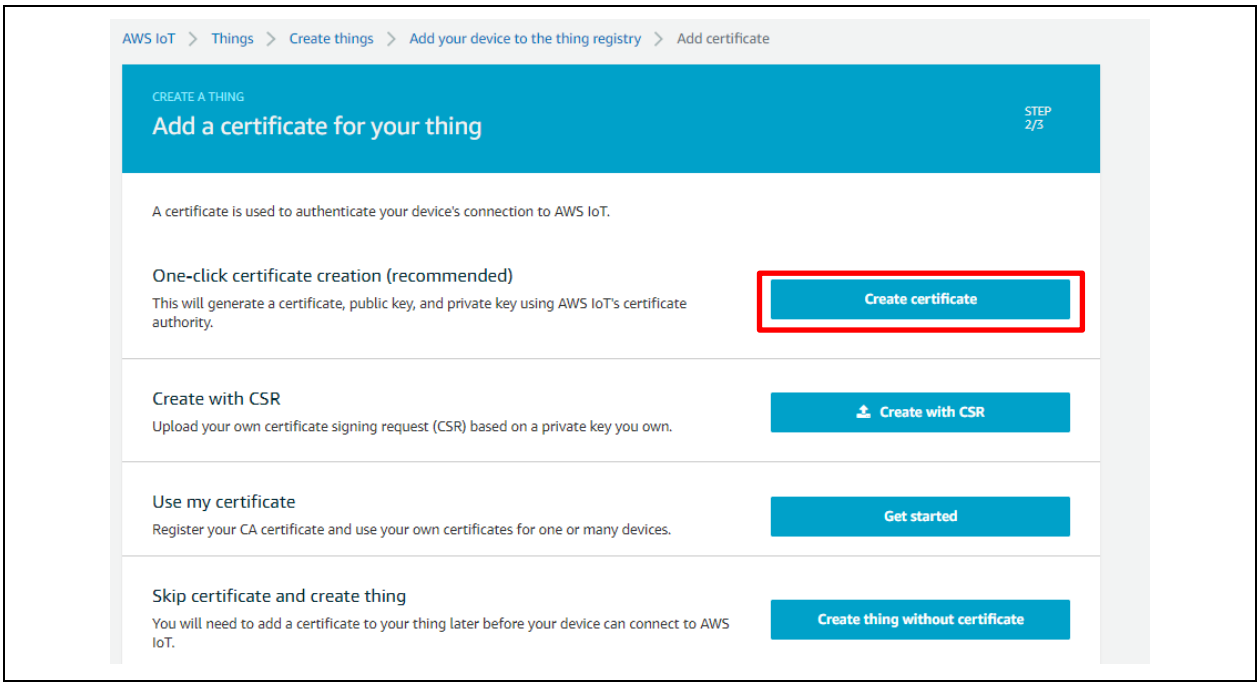
Create a single thing

Add name to thing and **Next**



Add name to a single thing

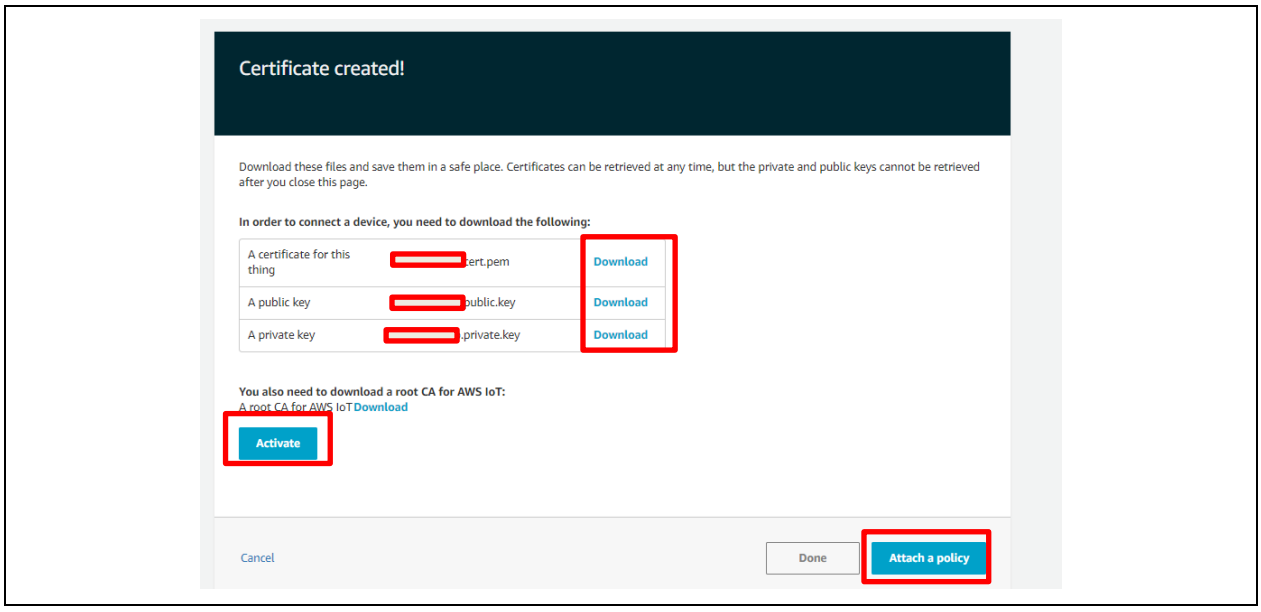
Add a certificate for thing



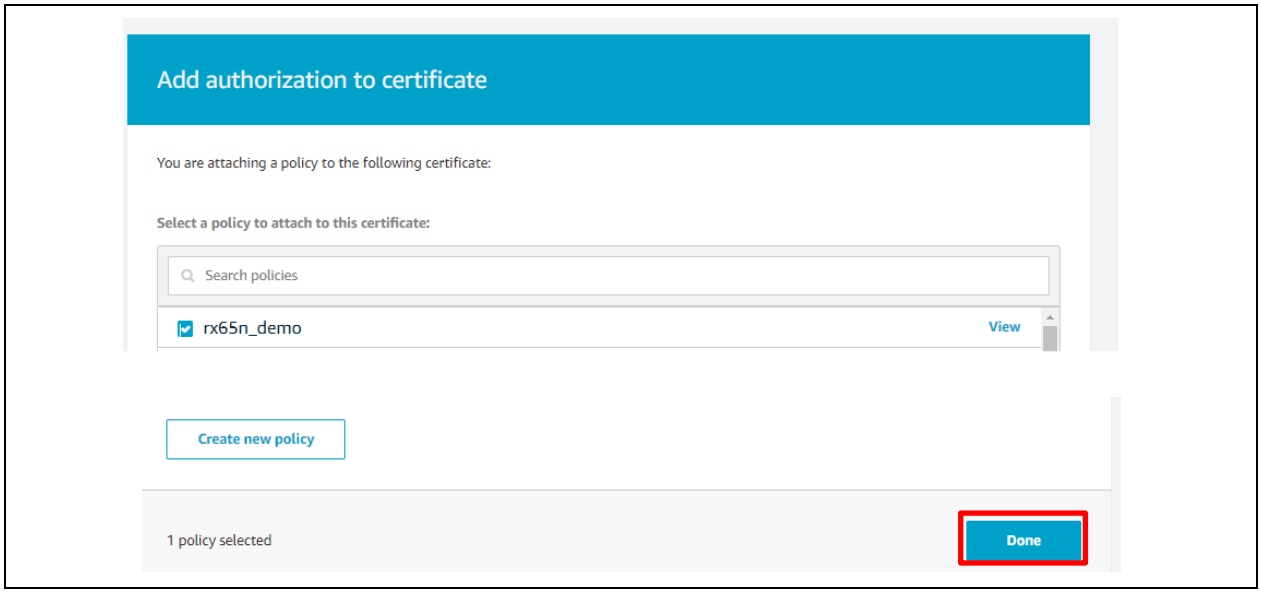
Create a certificate for thing

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**



Register policy to thing

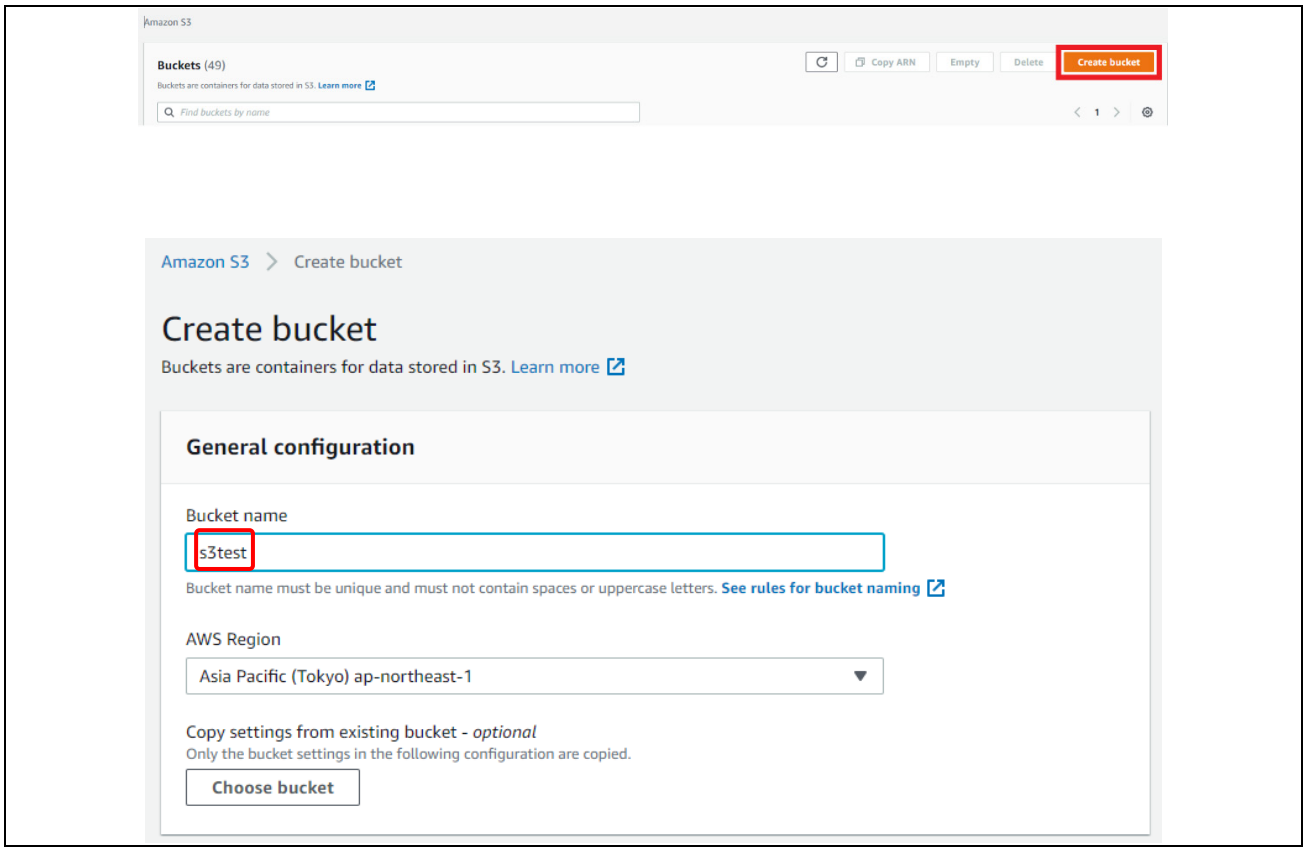


1.2 Create an Amazon S3 bucket

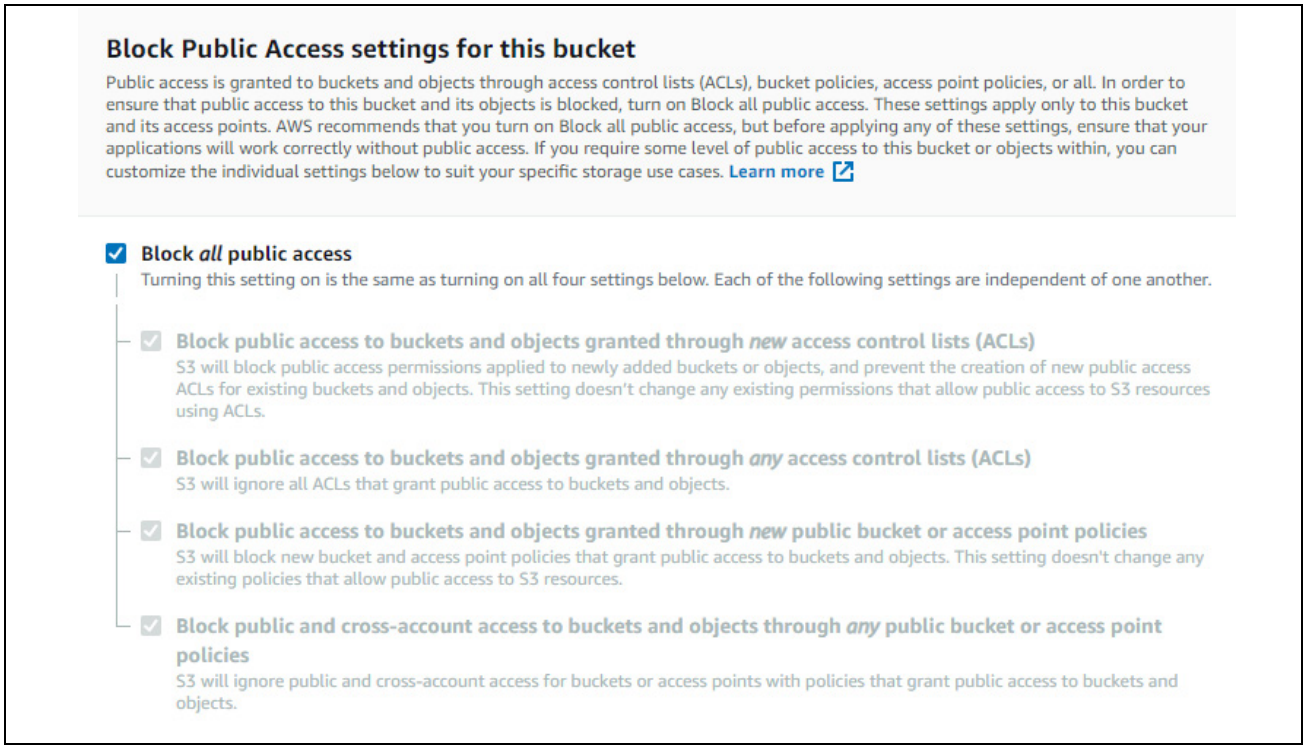
① Amazon Simple Storage Service (S3) AWS Service that enables to store files in the cloud that can be accessed by you or other services. OTA update files are stored in Amazon S3 buckets.

Please refer https://docs.aws.amazon.com/freertos/latest/userguide/dg-ota-bucket.html

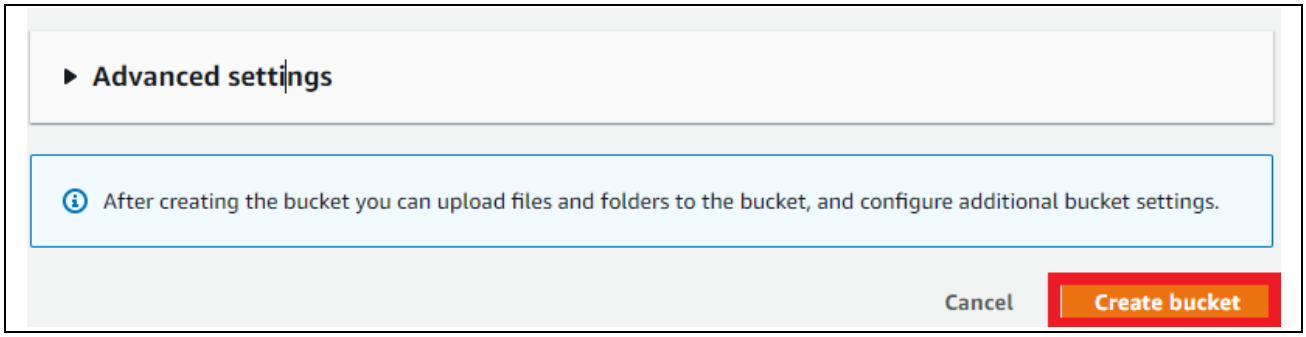
② Choose **Create bucket**, type name


Amazon S3
Buckets (49)
Buckets are containers for data stored in S3. Learn more
Find buckets by name
Copy ARN
Empty
Delete
Create bucket
Amazon S3 > Create bucket
Create bucket
Buckets are containers for data stored in S3. Learn more
General configuration
Bucket name
s3test
Bucket name must be unique and must not contain spaces or uppercase letters. See rules for bucket naming
AWS Region
Asia Pacific (Tokyo) ap-northeast-1
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket

③ Select **Block *all* public access**



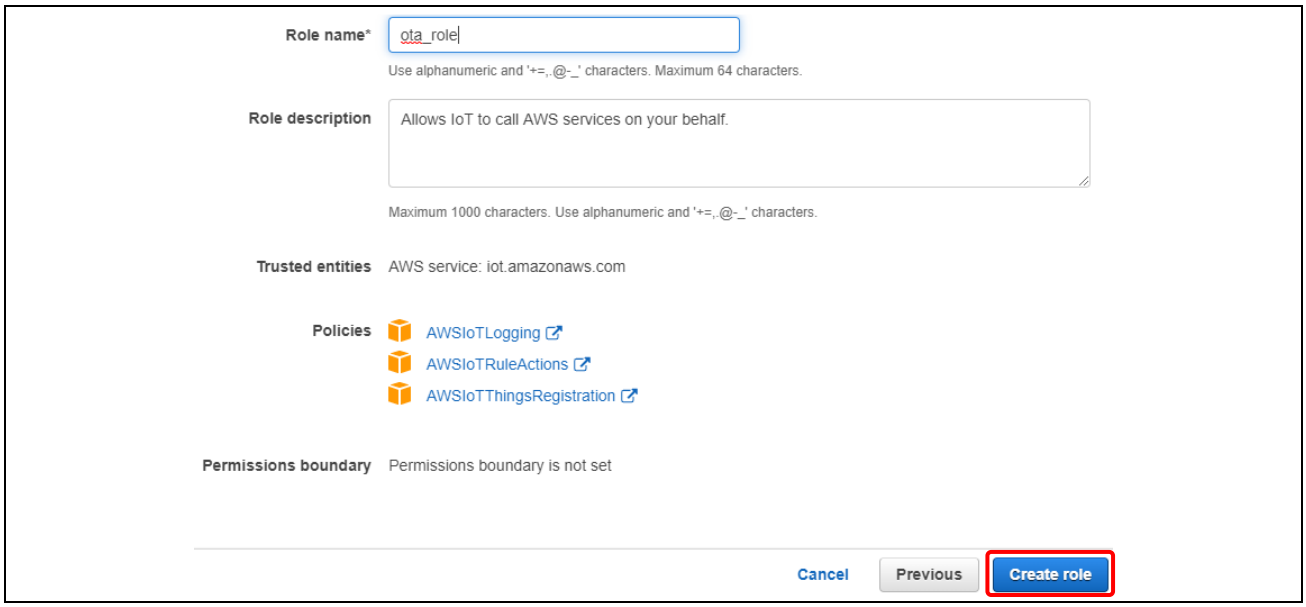
④ Choose **Create bucket**.



1.3 Create service role for OTA update

⑤ Identity Access Management (IAM) helps you securely control access to AWS resources

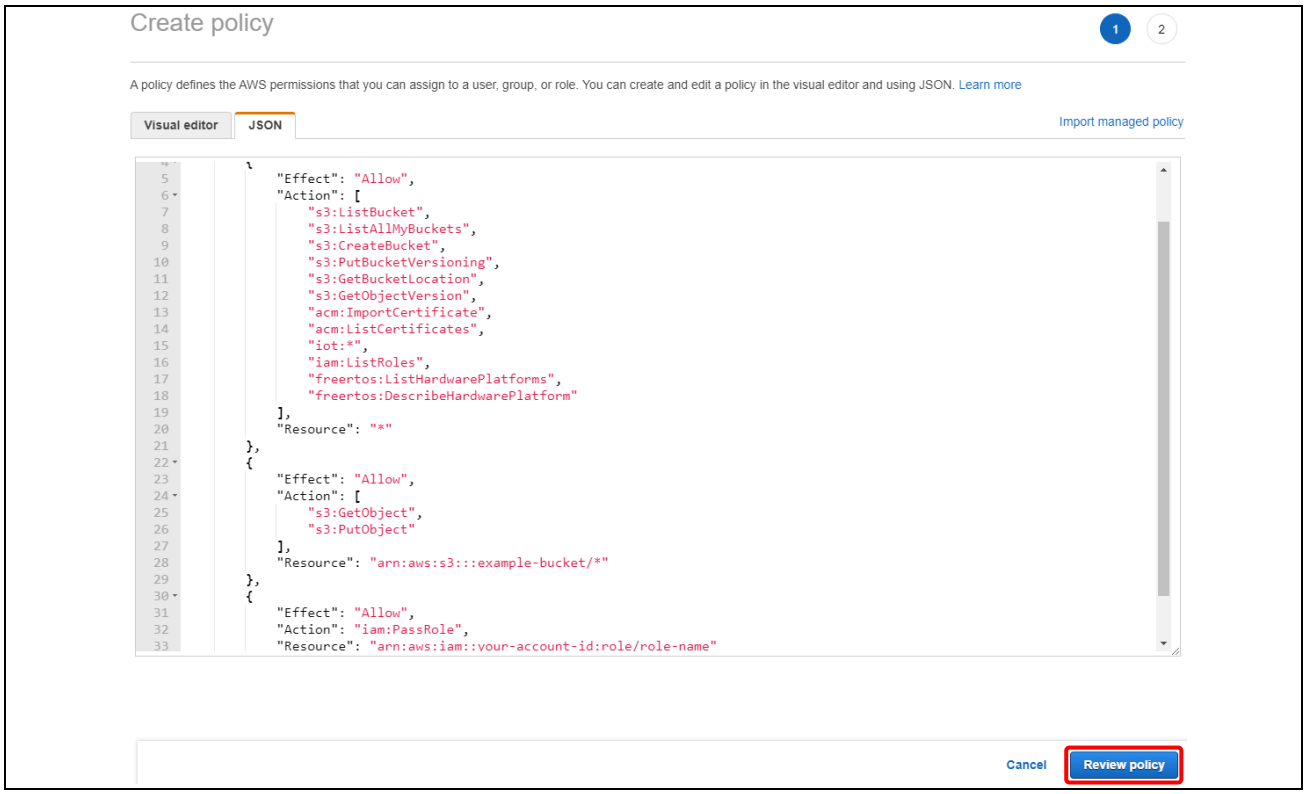
Please refer https://docs.aws.amazon.com/freertos/latest/userguide/create-service-role.html



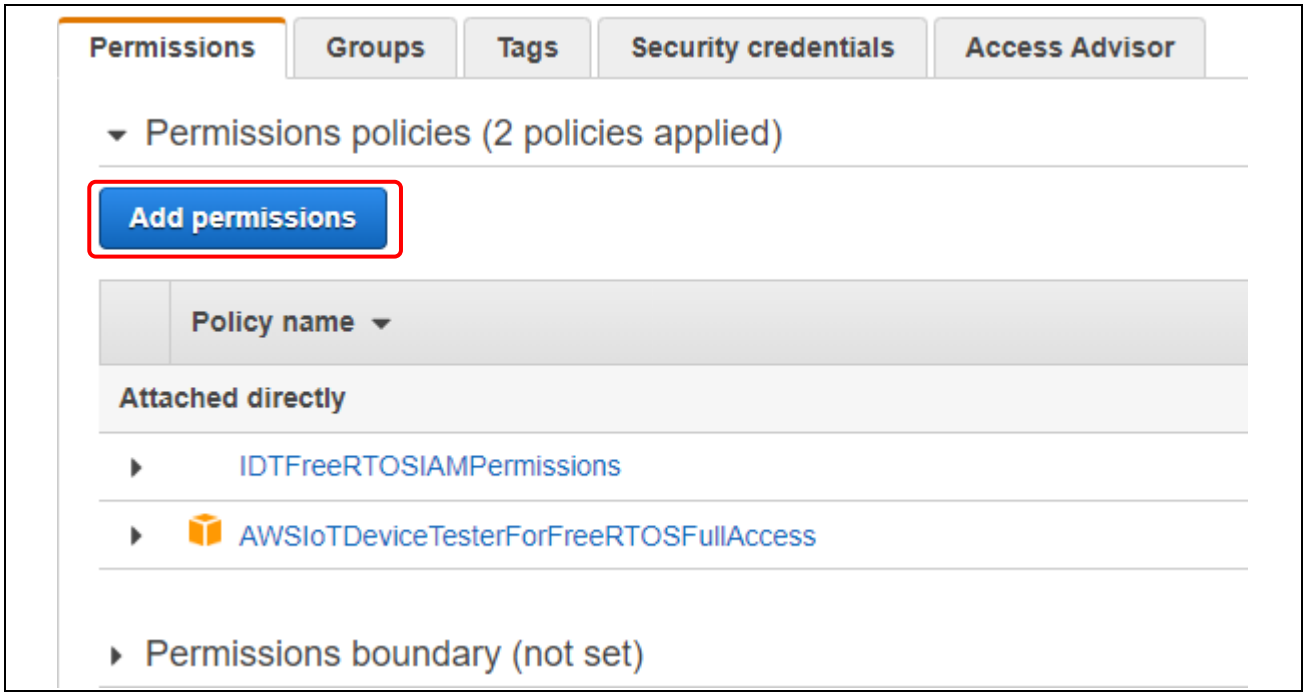
1.4 Create an OTA user policy and attach the OTA user policy to your IAM user

① Create an OTA user policy and attach the OTA user policy to your IAM user

Please refer https://docs.aws.amazon.com/freertos/latest/userguide/create-ota-user-policy.html



② Attach the OTA user policy to your IAM user



1.5 Register a code-signing certificate on AWS

Register a code-signing certificate on AWS

- Please refer Renesas MCU Firmware Update Design Policy section 7.3 Generating ECDSA-SHA256 Key Pairs with OpenSSL to create keys and certification.
- Go to IoT Core→ Manage → Jobs → **Create** → Create Update Job → Select **Devices to Update** under Select a job → choose **Select** under Sign New Firmware Image and **choose any thing** create before→**Next**→ Choose **Create** under Code Signing Profile

✓ Profile Name: Anything is OK

✓ Device Hardware Platform: **Windows Simulator**

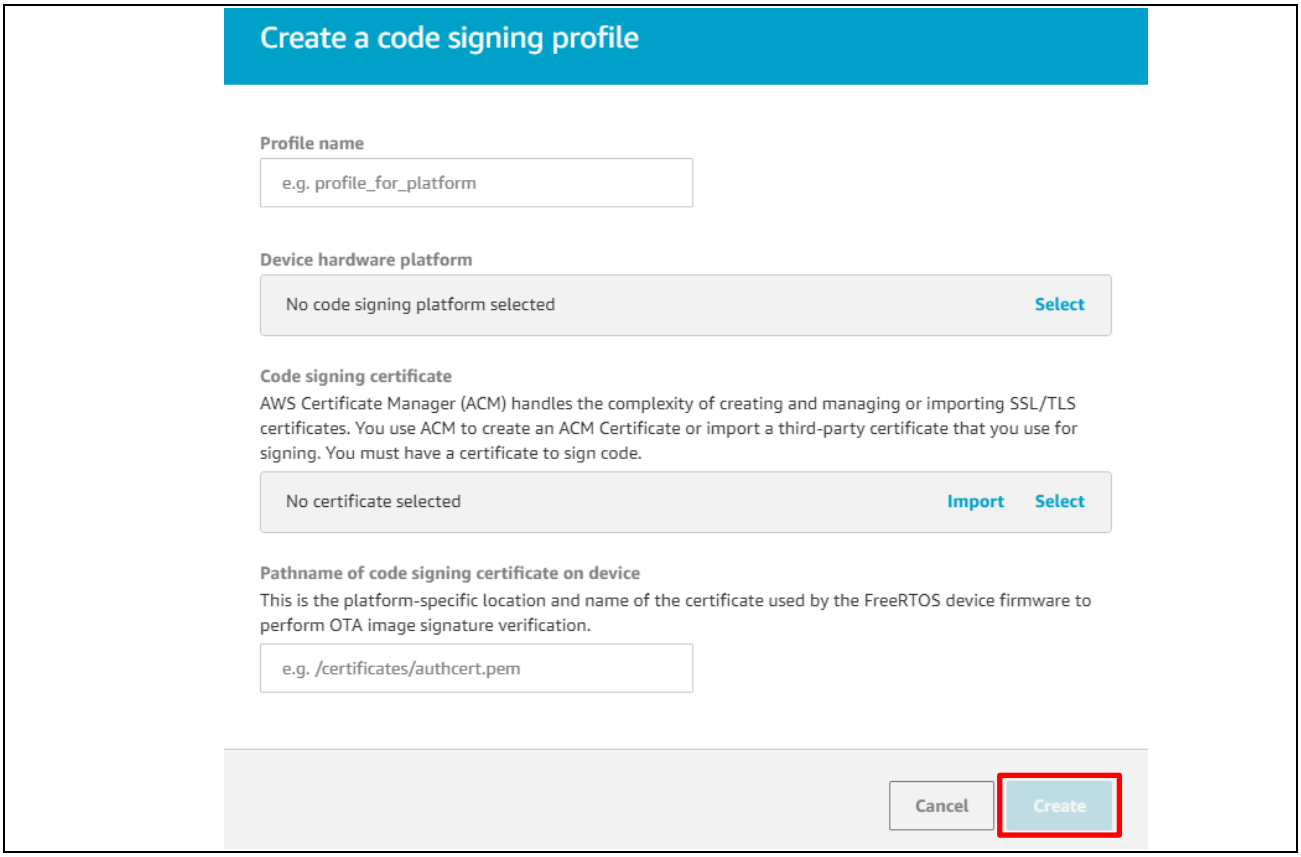
✓ Code-signing certificate:

✓ Select Certificate: Specify **secp256r1.crt**

✓ Select certificate private key: specify **secp256r1.privatekey**

✓ Select Certificate Chain (Optional): **ca.crt**

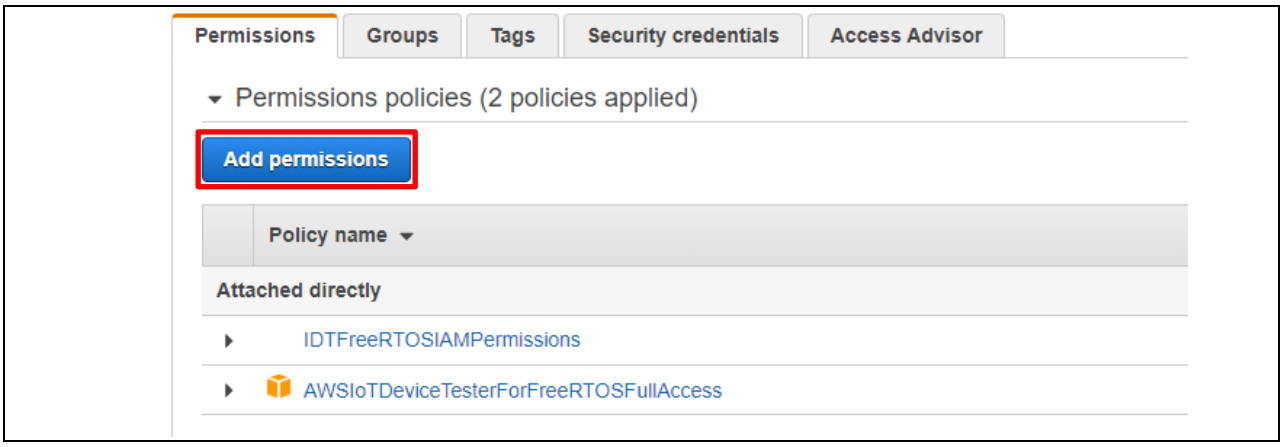
✓ Device code-signing certificate pathname: Anything is OK



1.6 Grant access to code signing for AWS IoT

Grant access to code signing for AWS IoT

Please refer https://docs.aws.amazon.com/freertos/latest/userguide/code-sign-policy.html



2 FreeRTOS OTA environment construction

At the beginning, user would be able to choose the version of Amazon FreeRTOS package, and the selected version will be downloaded from GitHub and imported automatically into the project. This makes it easier for the user, so that the user can focus only on Amazon FreeRTOS configuration and writing application code.

Note: If you want to start over from the beginning due to an operation error in 2.2 and 2.3, execute "⑥ Erase RX65N-RSK" in 2.2 and then start over.

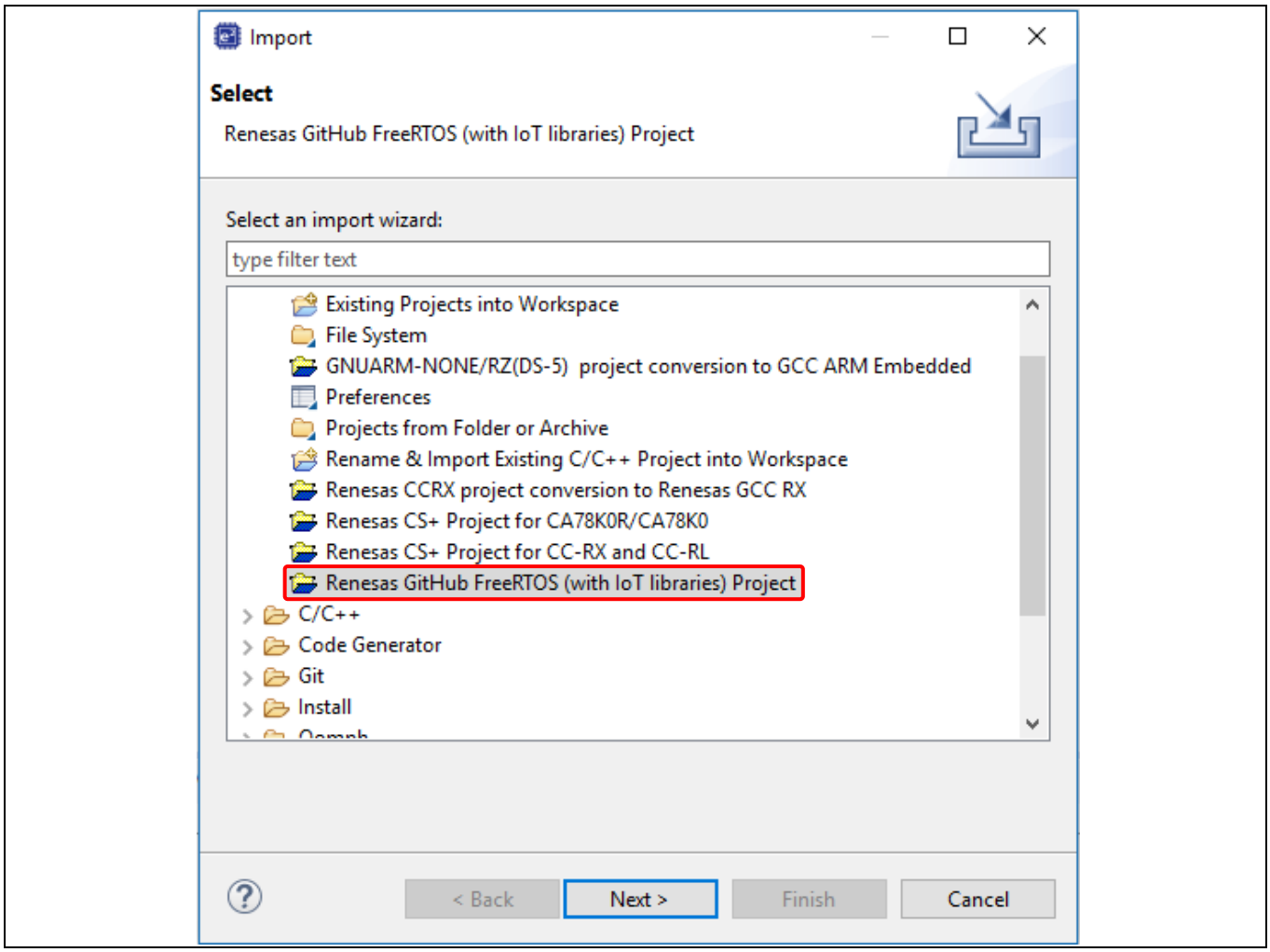
2.1 Import, configurate head file and build aws_demos and boot_loader

The figure below shows how to import Amazon FreeRTOS project:

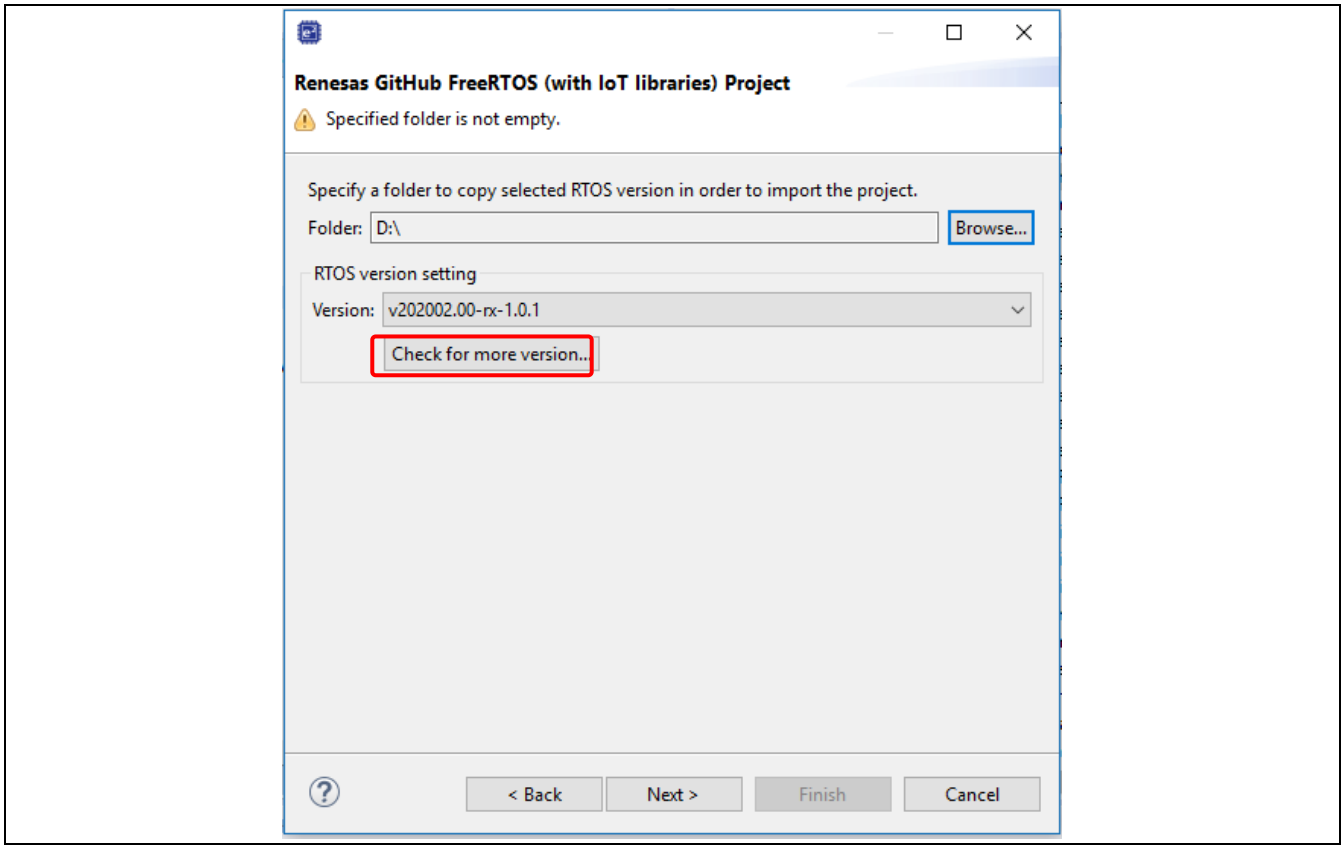
① Launch e² studio

② Select [File] → [Import…]

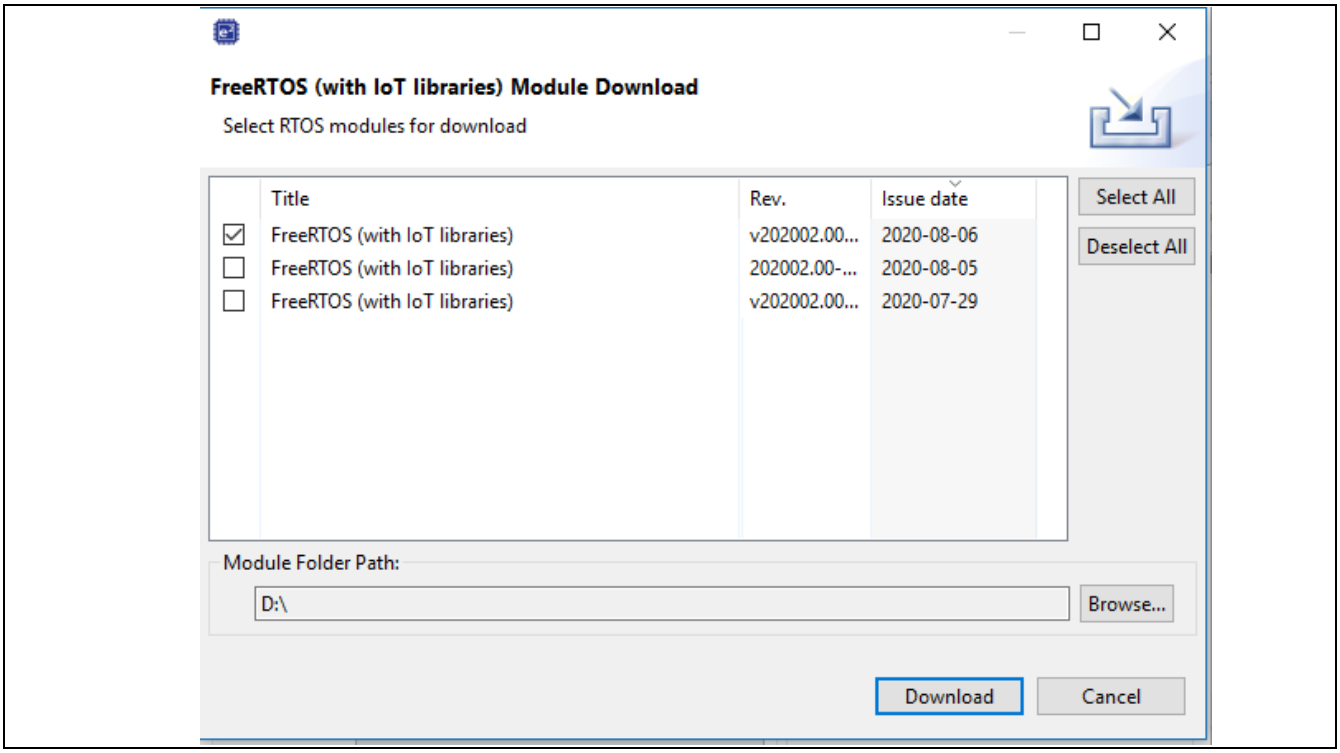
③ Select “Renesas GitHub FreeRTOS (with IoT libraries) Project”



④ Select “Check for more version…” to show the download dialog

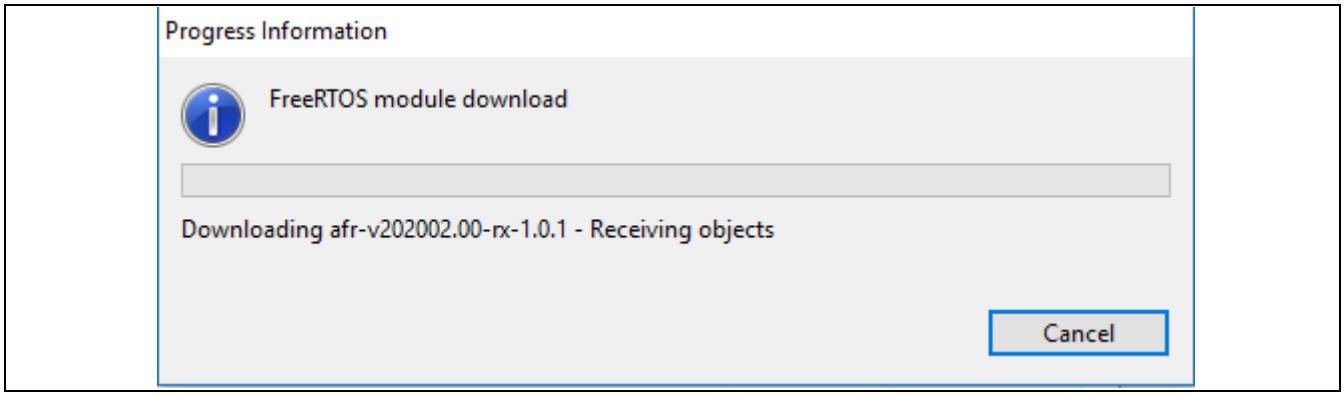


⑤ Choose the lastest package

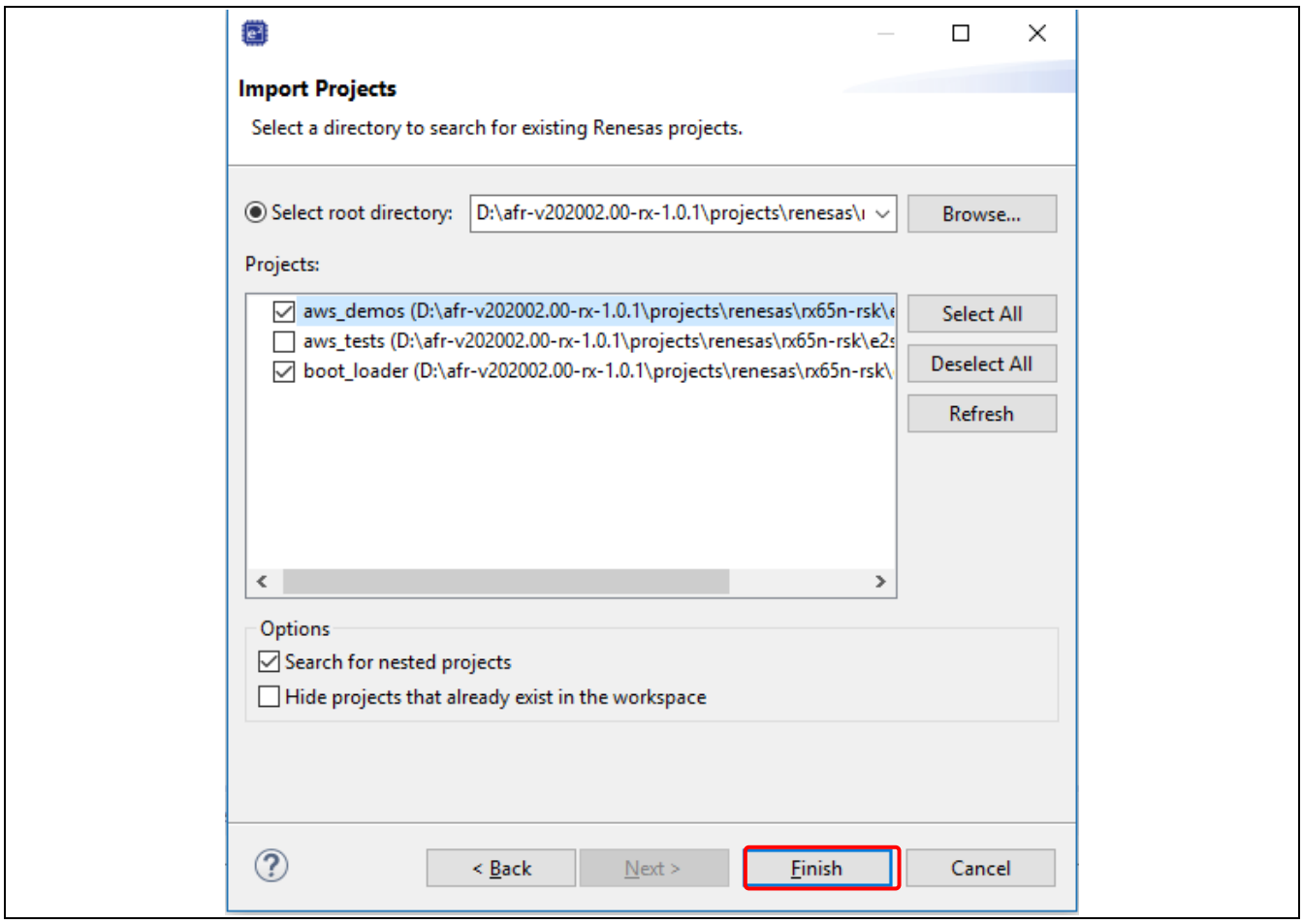


⑥ Agree the end user license agreement.

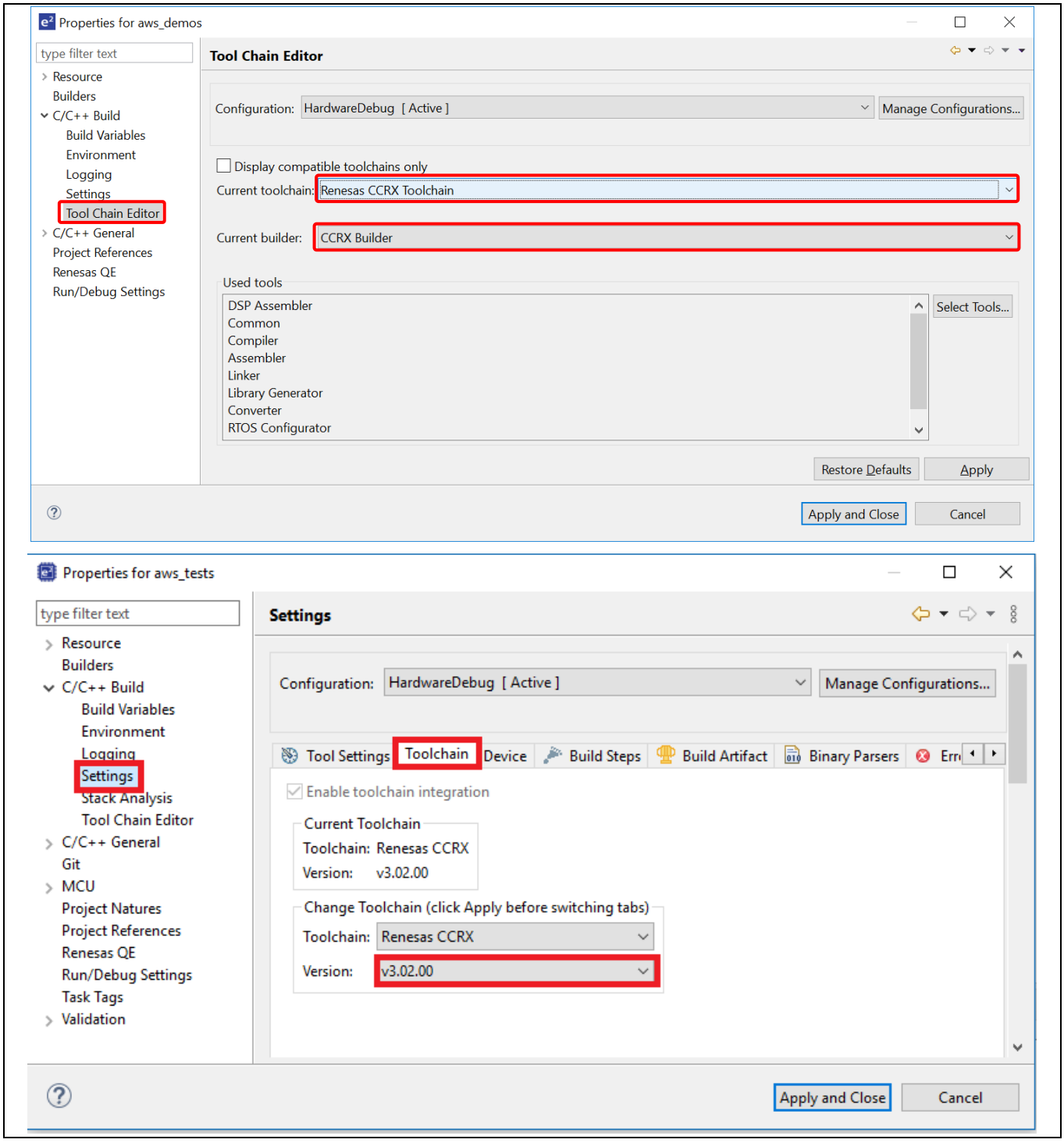
⑦ Wait for downloading completed.



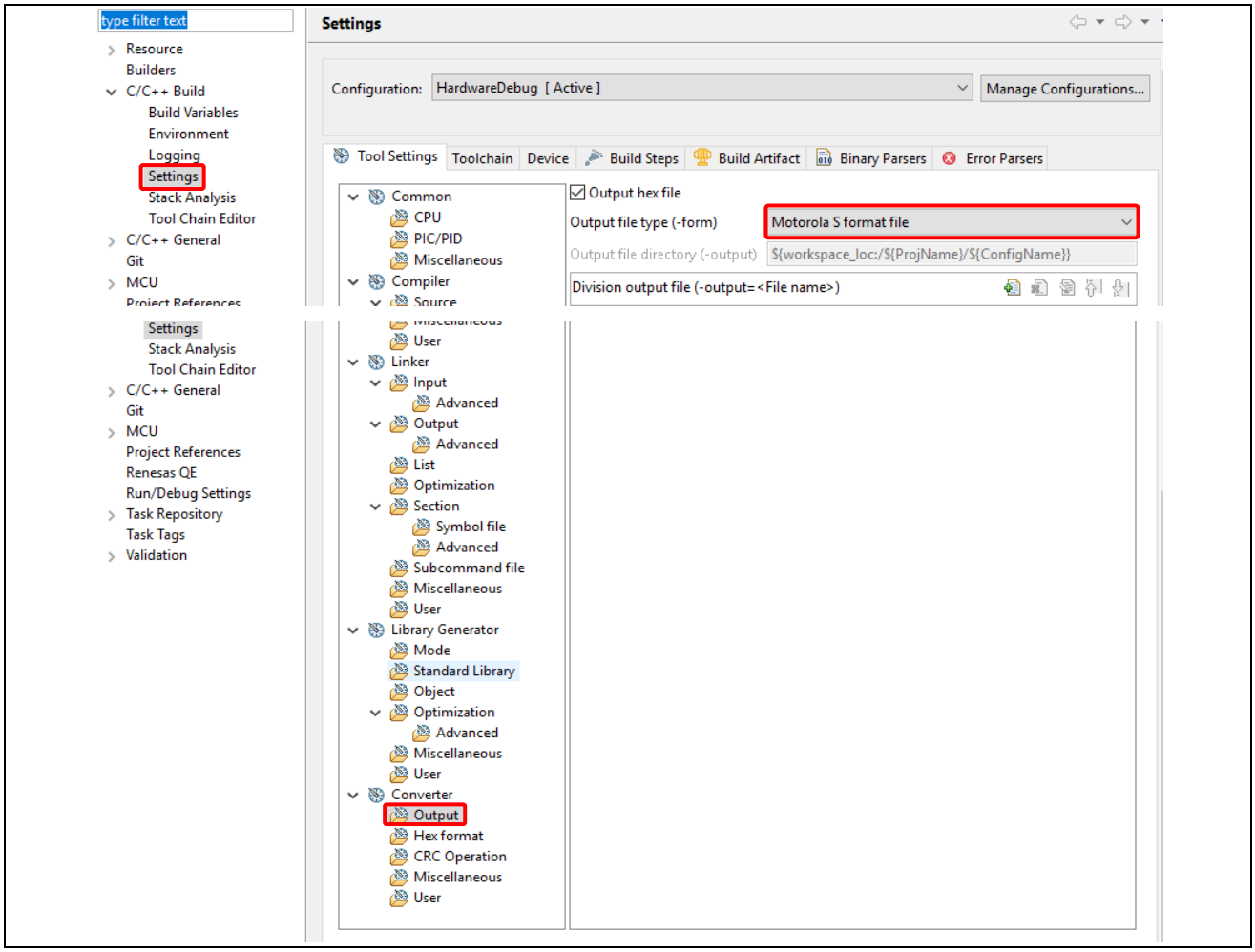
⑧ Select the project to import. Choose aws_demos and booloader project.



⑨ Open project [project] → [properties] → C/C++ Build → Tool Chain Editor for both projects, select toolchain and builder, then specify toolchain version.



⑩ Check output hex file.

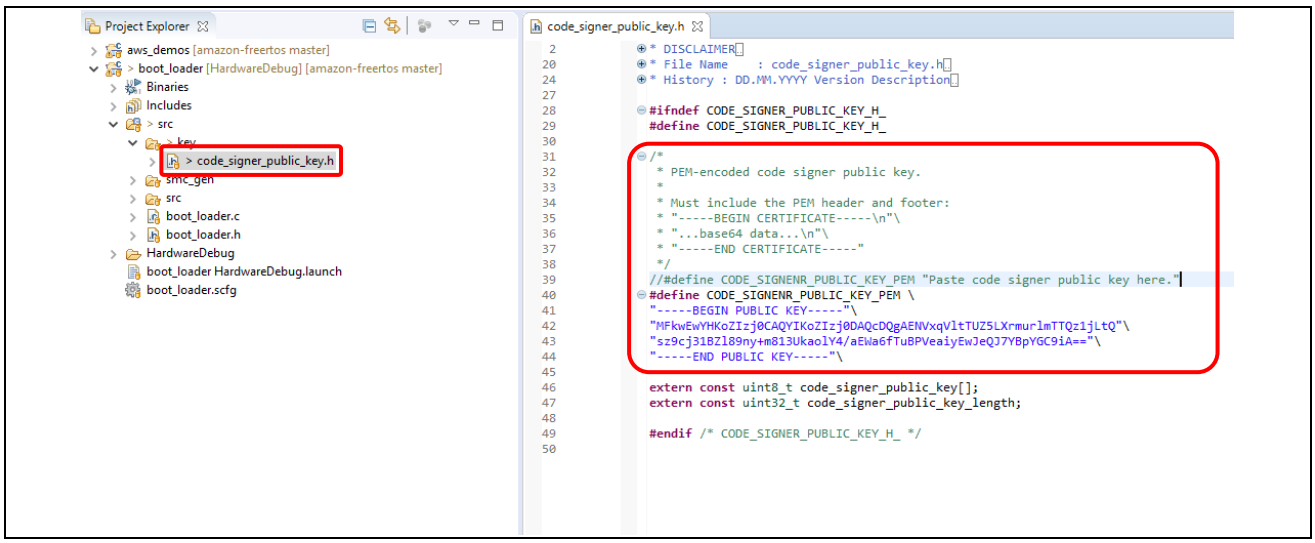


⑪ Input **public key**

In bootloader project, open **projects\renesas\rx65n-rsk\e2studio\boot_loader\src\key\code_signer_public_key.h** and input **public key**.

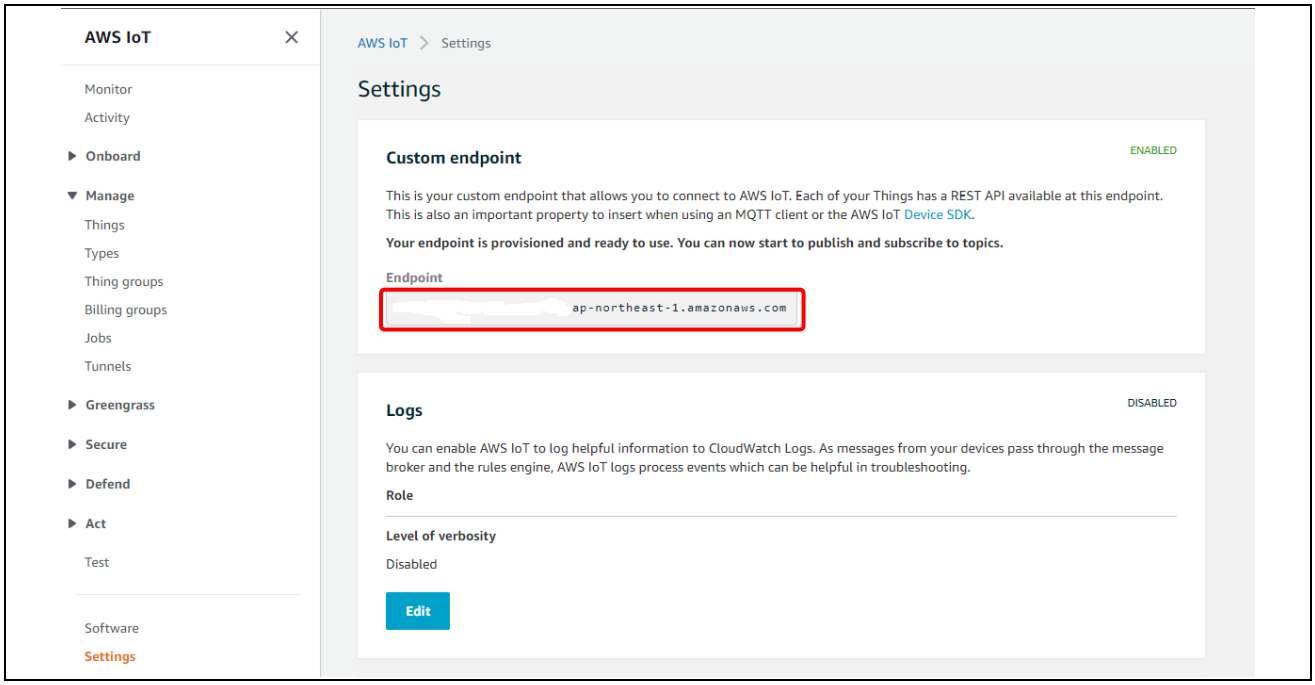
Please refer “How to implement FreeRTOS OTA by using Amazon Web Services on RX65N” section “7.3 Generating ECDSA-SHA256 Key Pairs with OpenSSL” to create public key.

Then **build**  to create **boot_loader.mot.**

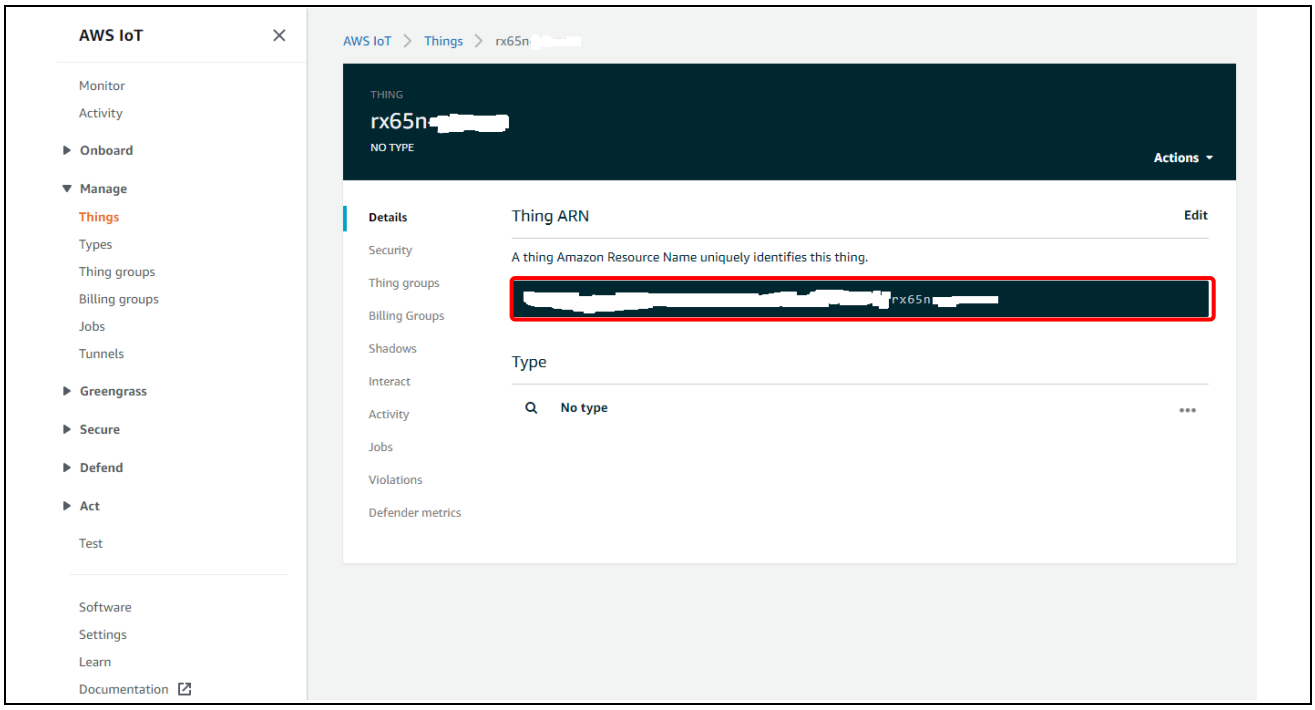


⑫ Open AWS IoT console

- Browse to the AWS IoT console.
- Choose **Setting**. Make a note of Endpoint. "Your AWS IoT endpoint"



- Choose **Manager→Things**. Make a note of AWS IoT thing name. "The AWS IoT thing name of your board"

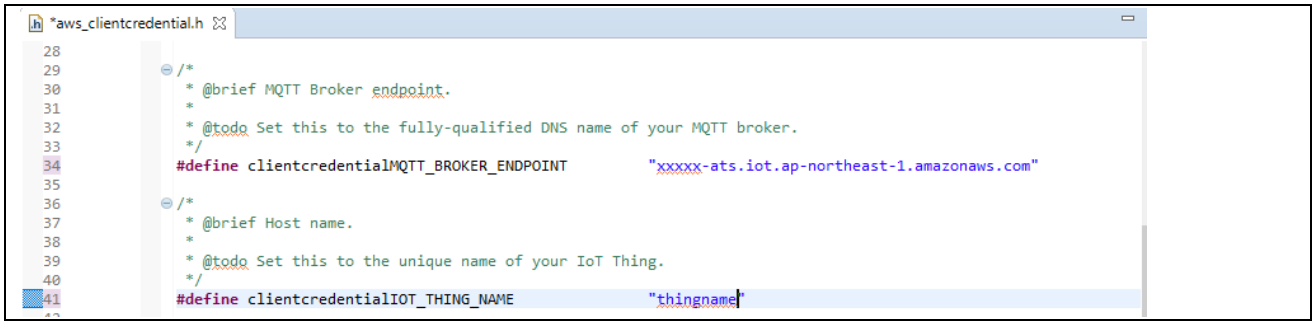


⑬ Open aws_demos project

- Open /demos/include/aws_clientcredential.h, specify values below

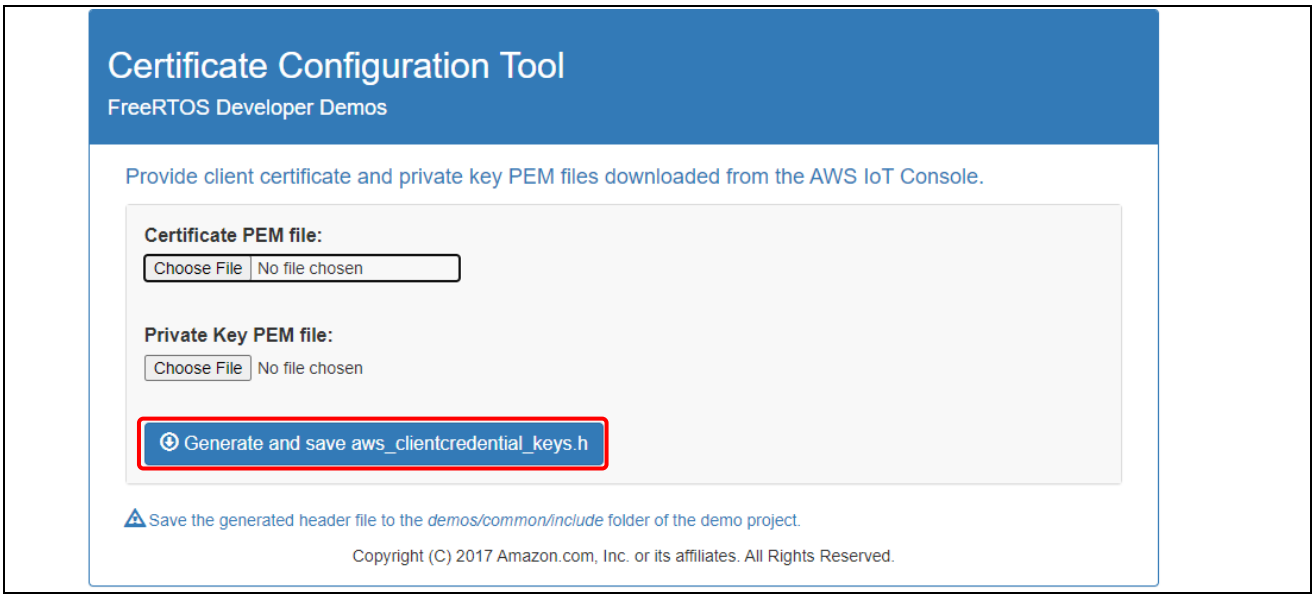
  #define clientcredentialMQTT_BROKER_ENDPOINT = "Your AWS IoT endpoint"

  #define clientcredentialIOT_THING_NAME "The AWS IoT thing name of your board"



⑭ Open Certificate Configuration Tool

- Move to the FreeRTOS path downloaded in 2.1 step ⑤
- Open tools→certificate_configuration→ CertificateConfigurator.html
- Import certificate PEM file and Private Key PEM file which were downloaded on 1.1 step ④
- Generate aws_clientcredential_keys.h



⑮ Open aws_demos project

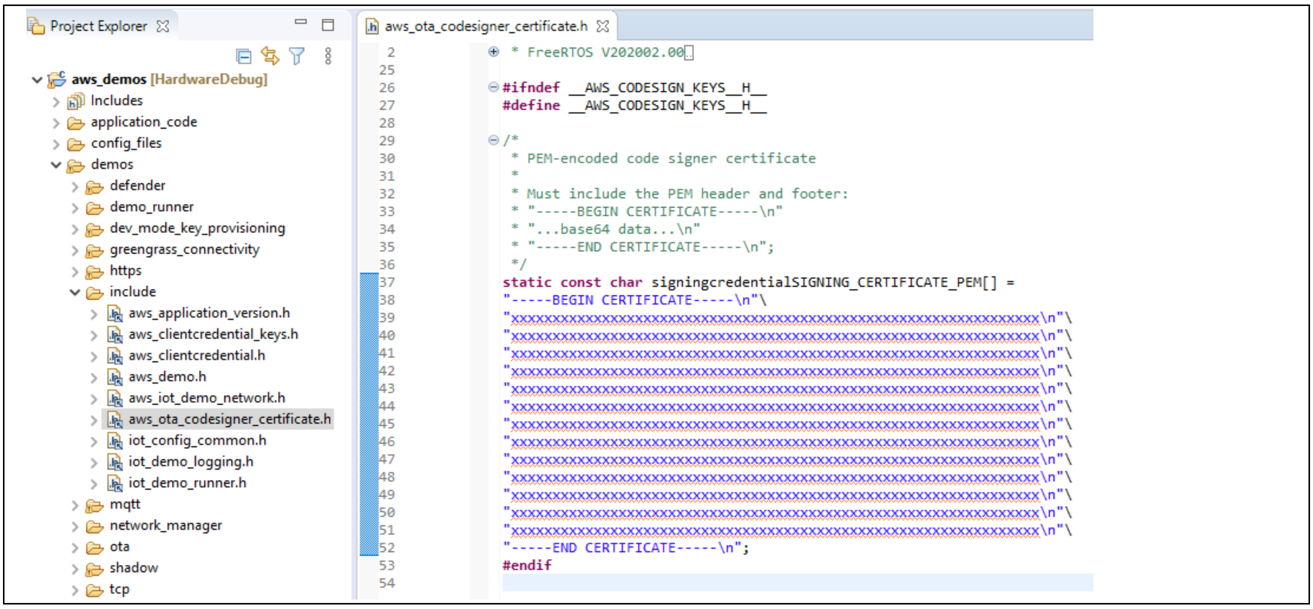
- Replace the aws_clientcredential_keys.h generated in ⑭ with the file in /demos/include/
- Open /demos/include/ aws_ota_codesigner_certificate.h, specify values below

signingcredentialSIGNING_CERTIFICATE_PEM [] = "xxxx";

"xxxx" is value from **secp256r1.crt.** Remember the "\" after each line of certification

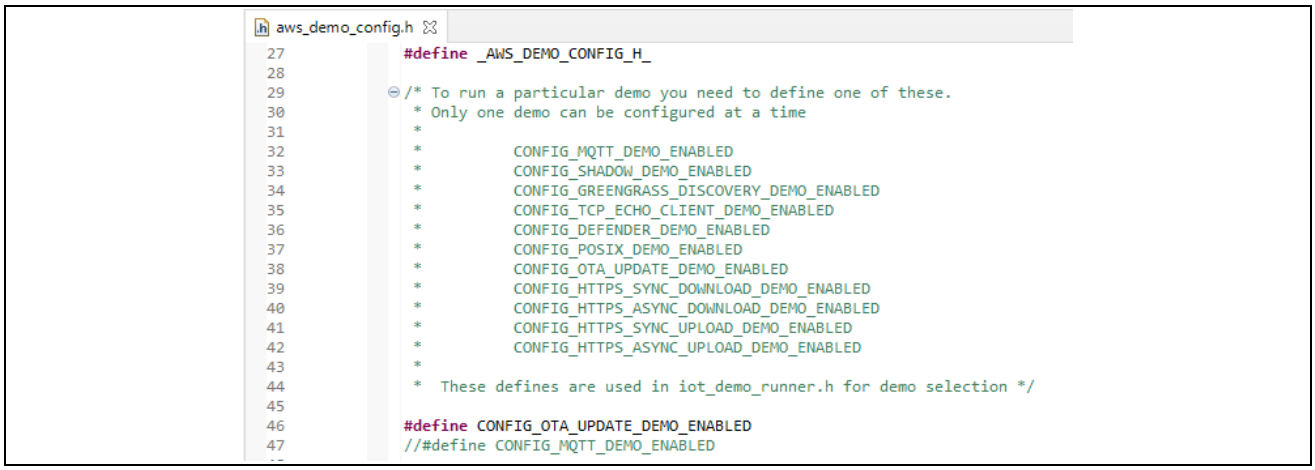
For creating **secp256r1.crt** please refer

"How to implement FreeRTOS OTA by using Amazon Web Services on RX65N" section "7.3 Generating ECDSA-SHA256 Key Pairs with OpenSSL".



2.2 Install the initial version of firmware

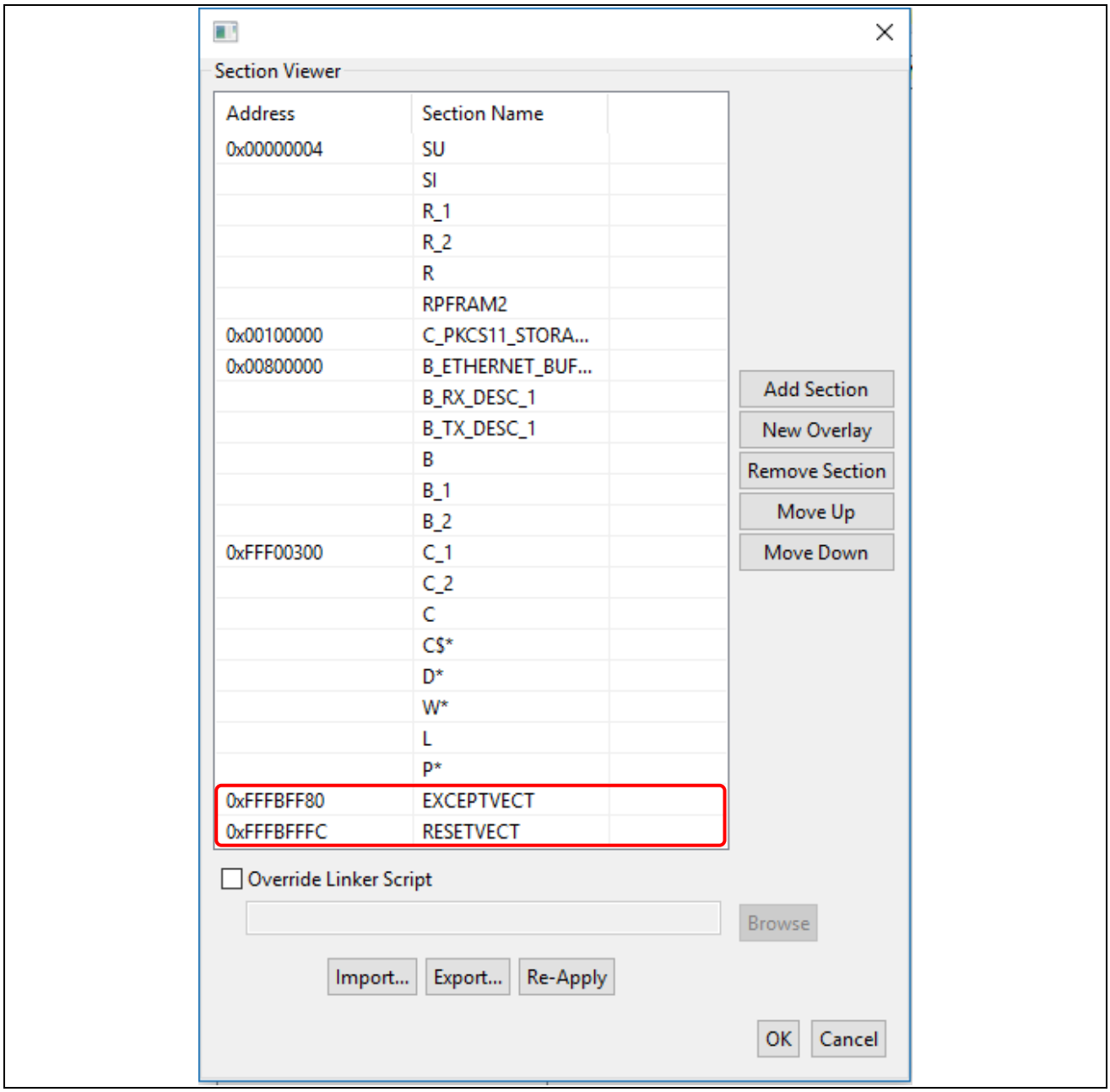
① Open amazon-freertos/vendors/renesas/boards/board/aws_demos/config_files/aws_demo_config.h, comment out #define CONFIG_MQTT_DEMO_ENABLED, and define CONFIG_OTA_UPDATE_DEMO_ENABLED.



② Open amazon-freertos/demos/include/ aws_application_version.h, set initial version of firmware to 0.9.2

```
25
26  #ifndef _AWS_APPLICATION_VERSION_H_
27  #define _AWS_APPLICATION_VERSION_H_
28
29  #include "iot_appversion32.h"
30  extern const AppVersion32_t xAppFirmwareVersion;
31
32  #define APP_VERSION_MAJOR    0
33  #define APP_VERSION_MINOR    9
34  #define APP_VERSION_BUILD    2
35
36  #endif
37
```

③ Open Section Viewer by selecting [Project]-> [Properties]-> C / C ++ Build-> Settings-> [Tool Settings] tab-> Linker-> Section-> […] and change section of aws_demos as following picture:



④ **Build** to create **aws_demos.mot** file

⑤ Create userprog.mot from Renesas Secure Flash Programmer

userprog.mot is a combination of aws_demos.mot and boot_loader.mot. Users can flash this file to RX65N-RSK to install initial firmware.

- Download Renesas Secure Flash Programmer release 1.0.1 and open **Renesas Secure Flash Programmer.exe**. Also downloads other files.
- Choose Initial Firm tab and then set parameters as following picture.

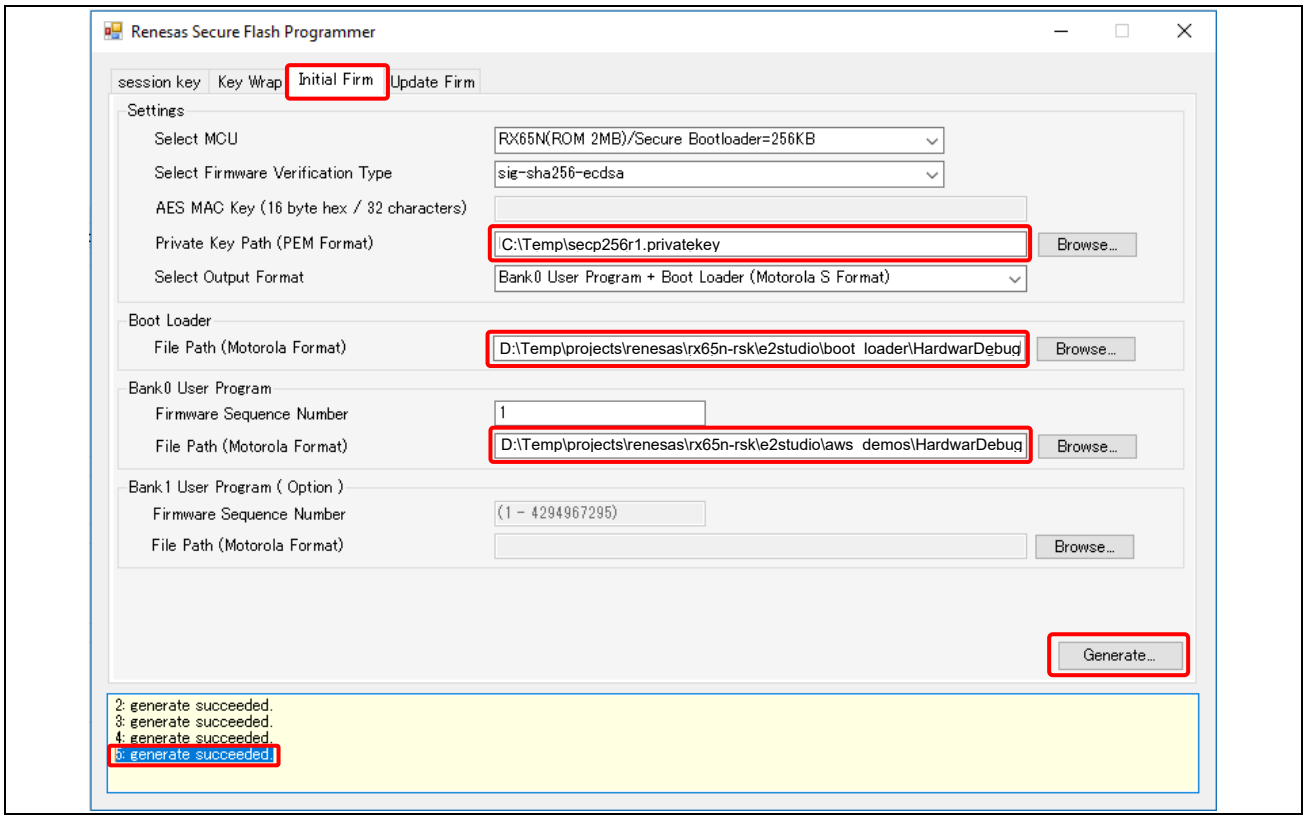
  Private Key Path: location to secp256r1.privatekey

  Boot Loader File Path: location to boot_loader.mot

  (projects\renesas\rx65n-rsk\e2studio\boot_loader\HardwareDebug)

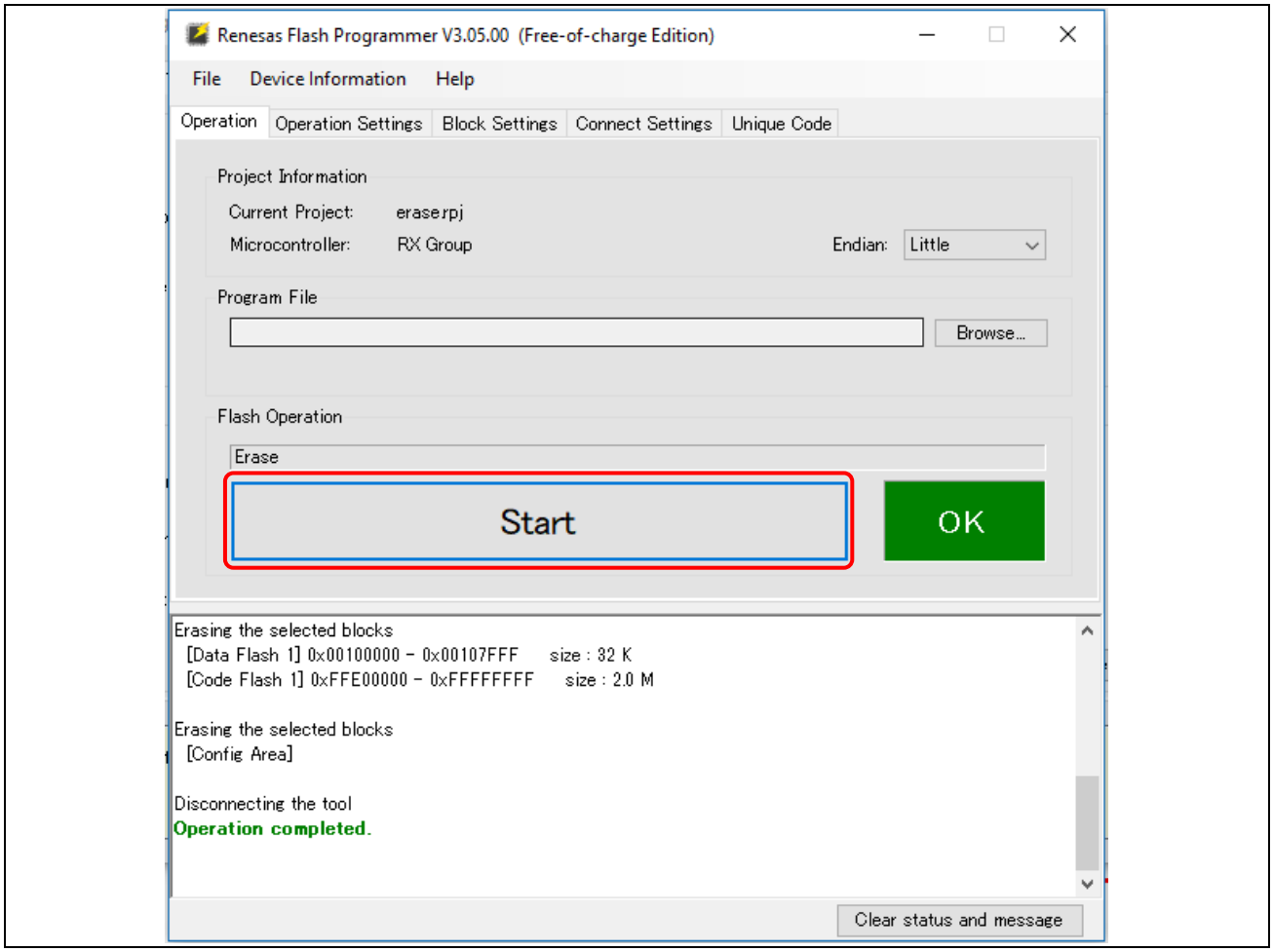
  Bank 0 User Program File Path: location to aws_demos.mot

  (projects\renesas\rx65n-rsk\e2studio\aws_demos\HardwareDebug)
- Create a folder named **init_firmware,** generate **userprog.mot,** and save to **init_firmware** folder and check **generate succeeded**



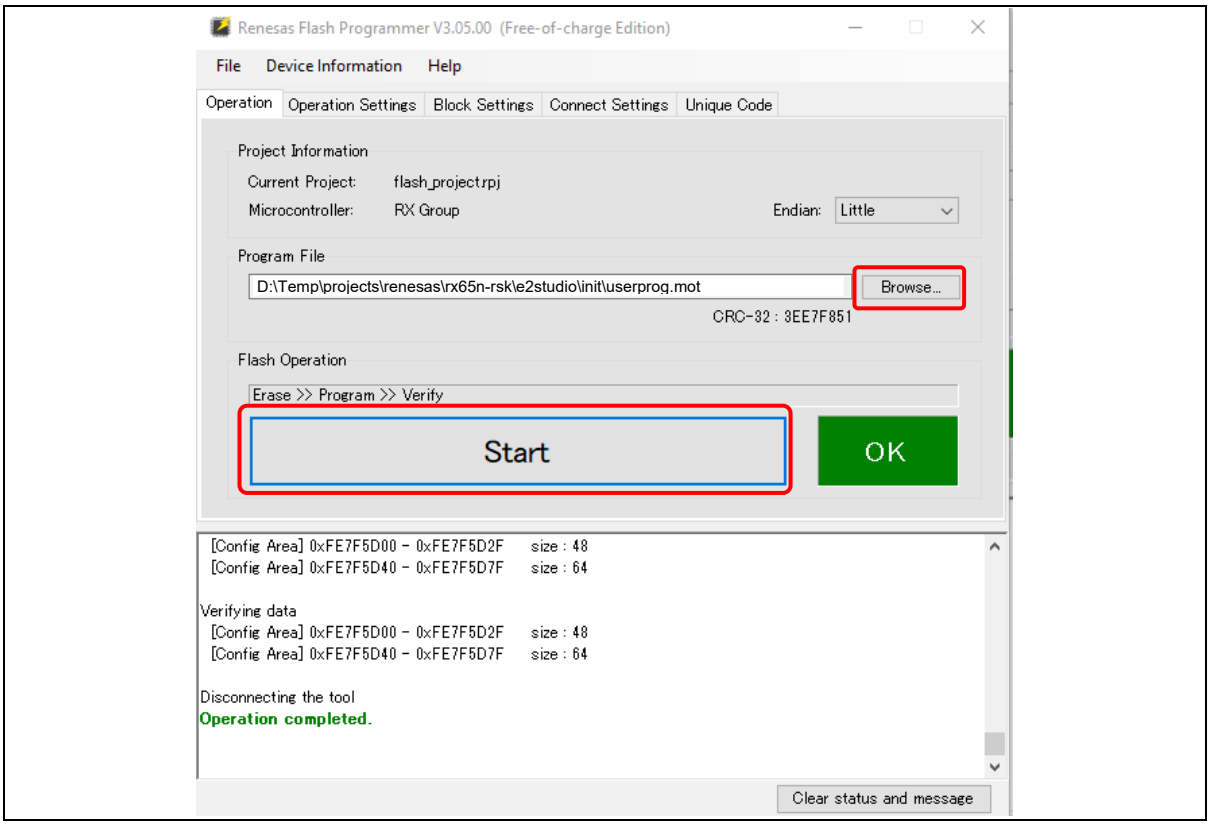
⑥ Erase RX65N-RSK

- Please download Renesas Flash Programmer (Programming GUI) from https://www.renesas.com/us/en/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html to get lastest version
- Open vendors\renesas\rx_mcu_boards\boards\rx65n-rsk\aws_demos\flash_project\erase_from_bank\ erase.rpj to erase data on bank
- Hit **Start** to erase flash ROM



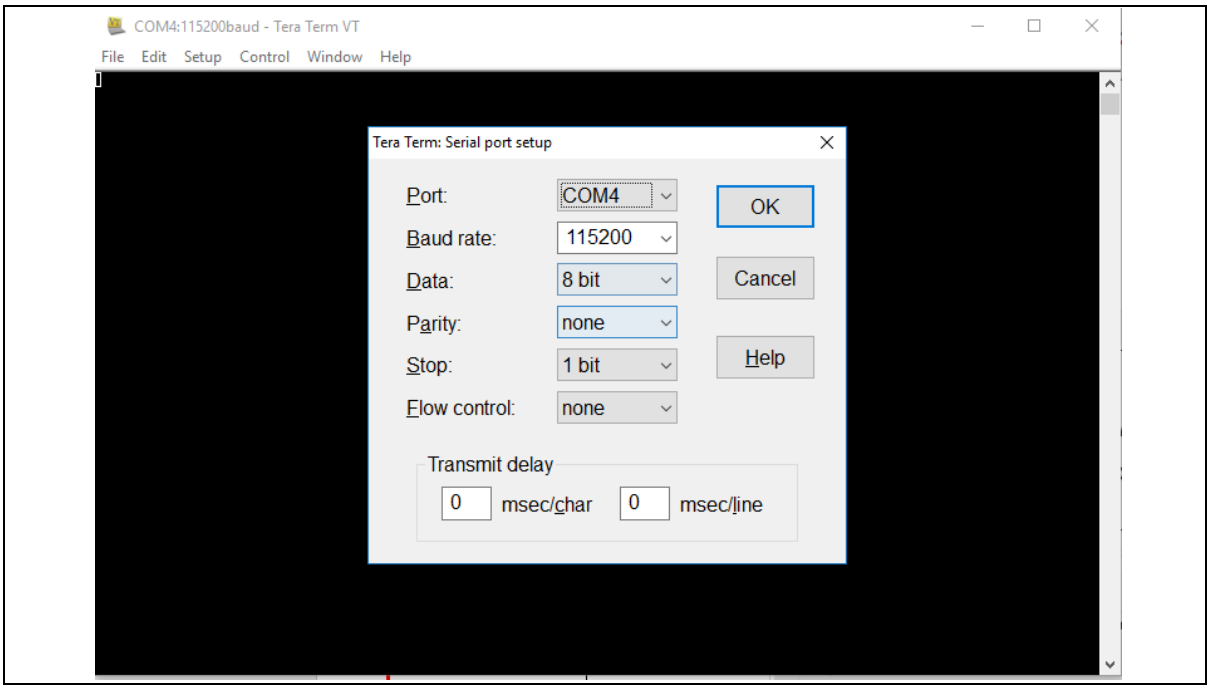
⑦ Flash initial firmware on RX65N-RSK

- Create a new project with a Renesas Flash Programmer. (Ex: flash_project.rpj)
- Start flashing userprog.mot which was saved in **init_firmware** folder.
- Browse to **init_firmware** folder, select userprog.mot and hit **Start**



⑧ Open Tera Term to see something like the following on initial firmware

If do not have Tera Term on PC, please download from https://ttssh2.osdn.jp/index.html.en and set up as following picture. Make sure that plugin USB Serial port to PC.



Version 0.9.2 (initial version) was installed to RX65N-RSK. The RX65N-RSK board is now listening for OTA updates.

```
------------------------------------------------
RX65N secure boot program
------------------------------------------------
Checking flash ROM status.
bank 0 status = 0xff [LIFECYCLE_STATE_BLANK]
bank 1 status = 0xfc [LIFECYCLE_STATE_INSTALLING]
bank info = 1. (start bank = 0)
start installing user program.
copy secure boot (part1) from bank0 to bank1...OK
copy secure boot (part2) from bank0 to bank1...OK
update LIFECYCLE_STATE from [LIFECYCLE_STATE_INSTALLING] to [LIFECYCLE_STATE_VALID]
bank1(temporary area) block0 erase (to update LIFECYCLE_STATE)...OK
bank1(temporary area) block0 write (to update LIFECYCLE_STATE)...OK
swap bank...
------------------------------------------------
RX65N secure boot program
------------------------------------------------
Checking flash ROM status.
bank 0 status = 0xf8 [LIFECYCLE_STATE_VALID]
bank 1 status = 0xff [LIFECYCLE_STATE_BLANK]
bank info = 0. (start bank = 1)
integrity check scheme = sig-sha256-ecdsa
bank0(execute area) on code flash integrity check...OK
jump to user program
```

□0 1 [ETHER_RECEI] Deferred Interrupt Handler Task started

1 1 [ETHER_RECEI] Network buffers: 3 lowest 3

2 1 [ETHER_RECEI] Heap: current 234192 lowest 234192

3 1 [ETHER_RECEI] Queue space: lowest 8

4 1 [IP-task] InitializeNetwork returns OK

5 1 [IP-task] xNetworkInterfaceInitialise returns 0

6 101 [ETHER_RECEI] Heap: current 234592 lowest 233392

7 2102 [ETHER_RECEI] prvEMACHandlerTask: PHY LS now 1

8 3001 [IP-task] xNetworkInterfaceInitialise returns 1

9 3092 [ETHER_RECEI] Network buffers: 2 lowest 2

10 3092 [ETHER_RECEI] Queue space: lowest 7

11 3092 [ETHER_RECEI] Heap: current 233320 lowest 233320

12 3193 [ETHER_RECEI] Heap: current 233816 lowest 233120

13 3593 [IP-task] vDHCPProcess: offer c0a80a09ip

14 3597 [ETHER_RECEI] Heap: current 233200 lowest 233000

15 3597 [IP-task] vDHCPProcess: offer c0a80a09ip

16 3597 [IP-task] IP Address: 192.168.10.9

17 3597 [IP-task] Subnet Mask: 255.255.255.0

18 3597 [IP-task] Gateway Address: 192.168.10.1

19 3597 [IP-task] DNS Server Address: 192.168.10.1

20 3600 [Tmr Svc] The network is up and running

21 3622 [Tmr Svc] Write certificate...

22 3697 [ETHER_RECEI] Heap: current 232320 lowest 230904

23 4497 [ETHER_RECEI] Heap: current 226344 lowest 225944

24 5317 [iot_thread] [INFO][DEMO][5317] ---------STARTING DEMO---------

25 5317 [iot_thread] [INFO][INIT][5317] SDK successfully initialized.

26 5317 [iot_thread] [INFO][DEMO][5317] Successfully initialized the demo. Network type for the demo: 4

27 5317 [iot_thread] [INFO][MQTT][5317] MQTT library successfully initialized.

28 5317 [iot_thread] [INFO][DEMO][5317] OTA demo version 0.9.2

29 5317 [iot_thread] [INFO][DEMO][5317] Connecting to broker...

30 5317 [iot_thread] [INFO][DEMO][5317] MQTT demo client identifier is rx65n (length 5).

31 5325 [ETHER_RECEI] Heap: current 206944 lowest 206504

32 5325 [ETHER_RECEI] Heap: current 206440 lowest 206440

33 5325 [ETHER_RECEI] Heap: current 206240 lowest 206240

38 5334 [ETHER_RECEI] Heap: current 190288 lowest 190288

39 5334 [ETHER_RECEI] Heap: current 190088 lowest 190088

40 5361 [ETHER_RECEI] Heap: current 158512 lowest 158168

41 5363 [ETHER_RECEI] Heap: current 158032 lowest 158032

42 5364 [ETHER_RECEI] Network buffers: 1 lowest 1

43 5364 [ETHER_RECEI] Heap: current 156856 lowest 156856

44 5364 [ETHER_RECEI] Heap: current 156656 lowest 156656

46 5374 [ETHER_RECEI] Heap: current 153016 lowest 152040

47 5492 [ETHER_RECEI] Heap: current 141464 lowest 139016

48 5751 [ETHER_RECEI] Heap: current 140160 lowest 138680

49 5917 [ETHER_RECEI] Heap: current 138280 lowest 138168

59 7361 [iot_thread] [INFO][MQTT][7361] Establishing new MQTT connection.

62 7428 [iot_thread] [INFO][MQTT][7428] (MQTT connection 81cfc8, CONNECT operation 81d0e8) Wait complete with result SUCCESS.

63 7428 [iot_thread] [INFO][MQTT][7428] New MQTT connection 4e8c established.

64 7430 [iot_thread] [OTA_AgentInit_internal] OTA Task is Ready.

65 7430 [OTA Agent T] [prvOTAAgentTask] Called handler. Current State [Ready] Event [Start] New state [RequestingJob]

66 7431 [OTA Agent T] [INFO][MQTT][7431] (MQTT connection 81cfc8) SUBSCRIBE operation scheduled.

67 7431 [OTA Agent T] [INFO][MQTT][7431] (MQTT connection 81cfc8, SUBSCRIBE operation 818c48) Waiting for operation completion.

68 7436 [ETHER_RECEI] Heap: current 128248 lowest 127992

69 7480 [OTA Agent T] [INFO][MQTT][7480] (MQTT connection 81cfc8, SUBSCRIBE operation 818c48) Wait complete with result SUCCESS.

70 7480 [OTA Agent T] [prvSubscribeToJobNotificationTopics] OK: $aws/things/rx65n-gr-rose/jobs/$next/get/accepted

71 7481 [OTA Agent T] [INFO][MQTT][7481] (MQTT connection 81cfc8) SUBSCRIBE operation scheduled.

72 7481 [OTA Agent T] [INFO][MQTT][7481] (MQTT connection 81cfc8, SUBSCRIBE operation 818c48) Waiting for operation completion.

73 7530 [OTA Agent T] [INFO][MQTT][7530] (MQTT connection 81cfc8, SUBSCRIBE operation 818c48) Wait complete with result SUCCESS.

74 7530 [OTA Agent T] [prvSubscribeToJobNotificationTopics] OK: $aws/things/rx65n-gr-rose/jobs/notify-next

75 7530 [OTA Agent T] [prvRequestJob_Mqtt] Request #0

76 7532 [OTA Agent T] [INFO][MQTT][7532] (MQTT connection 81cfc8) MQTT PUBLISH operation queued.

77 7532 [OTA Agent T] [INFO][MQTT][7532] (MQTT connection 81cfc8, PUBLISH operation 818b80) Waiting for operation completion.

78 7552 [OTA Agent T] [INFO][MQTT][7552] (MQTT connection 81cfc8, PUBLISH operation 818b80) Wait complete with result SUCCESS.

79 7552 [OTA Agent T] [prvOTAAgentTask] Called handler. Current State [RequestingJob] Event [RequestJobDocument] New state [WaitingForJob]

80 7552 [OTA Agent T] [prvParseJSONbyModel] Extracted parameter [clientToken: 0:rx65n-gr-rose]

81 7552 [OTA Agent T] [prvParseJSONbyModel] parameter not present: execution

82 7552 [OTA Agent T] [prvParseJSONbyModel] parameter not present: jobId

83 7552 [OTA Agent T] [prvParseJSONbyModel] parameter not present: jobDocument

84 7552 [OTA Agent T] [prvParseJSONbyModel] parameter not present: afr_ota

85 7552 [OTA Agent T] [prvParseJSONbyModel] parameter not present: protocols

86 7552 [OTA Agent T] [prvParseJSONbyModel] parameter not present: files

87 7552 [OTA Agent T] [prvParseJSONbyModel] parameter not present: filepath

99 7651 [ETHER_RECEI] Heap: current 129720 lowest 127304

100 8430 [iot_thread] [INFO][DEMO][8430] State: Ready Received: 1 Queued: 0 Processed: 0 Dropped: 0

101 9430 [iot_thread] [INFO][DEMO][9430] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

102 10430 [iot_thread] [INFO][DEMO][10430] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

103 11430 [iot_thread] [INFO][DEMO][11430] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

104 12430 [iot_thread] [INFO][DEMO][12430] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

105 13430 [iot_thread] [INFO][DEMO][13430] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

106 14430 [iot_thread] [INFO][DEMO][14430] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

107 15430 [iot_thread] [INFO][DEMO][15430] State: WaitingForJob Received: 1 Queued: 0 Processed: 0 Dropped: 0

2.3 Update the version of your firmware

① Open demos/include/aws_application_version.h and increment the APP_VERSION_BUILD token value (increase to 0.9.3)

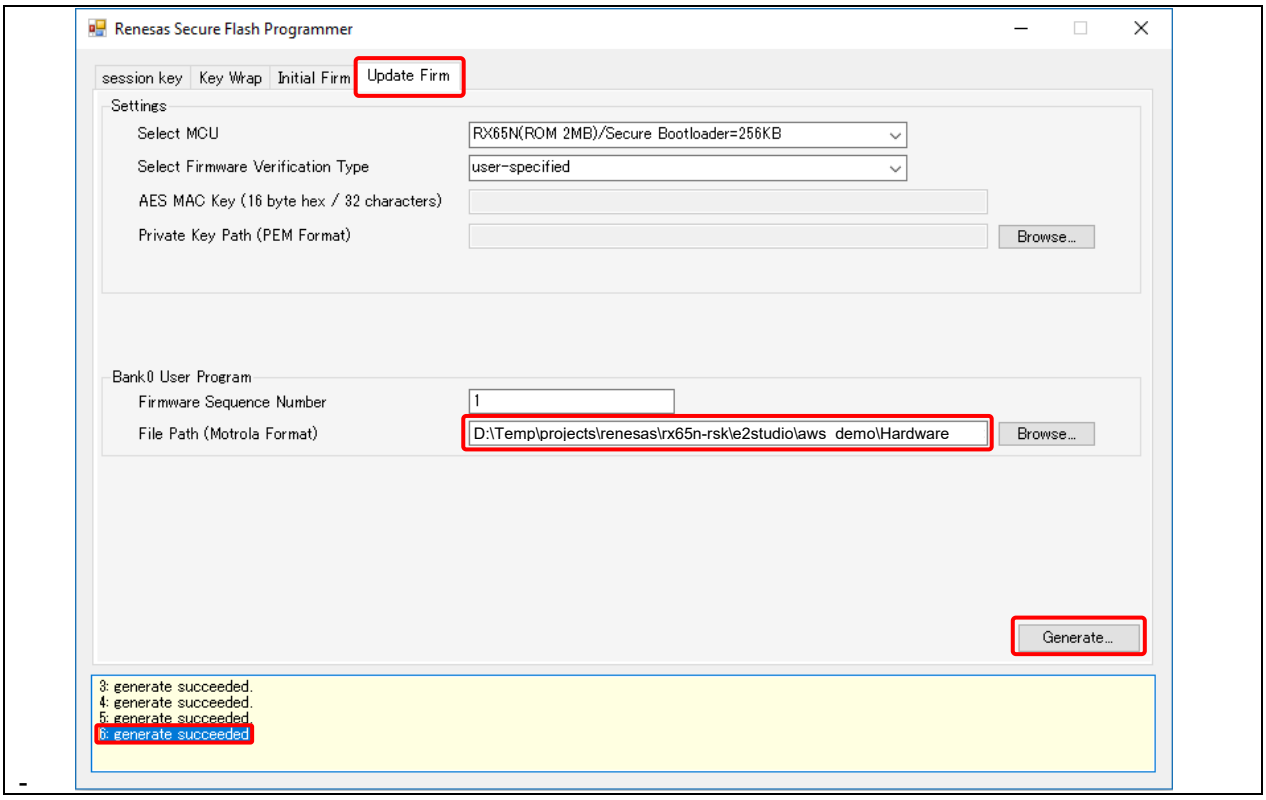
② Rebuild  the project

③ Create userprog.rsu from Renesas Secure Flash Programmer for Update the version of your firmware

- Open Amazon-FreeRTOS-Tools\Renesas Secure Flash Programmer.exe
- Choose Update Firm tab and then set parameters as following picture.

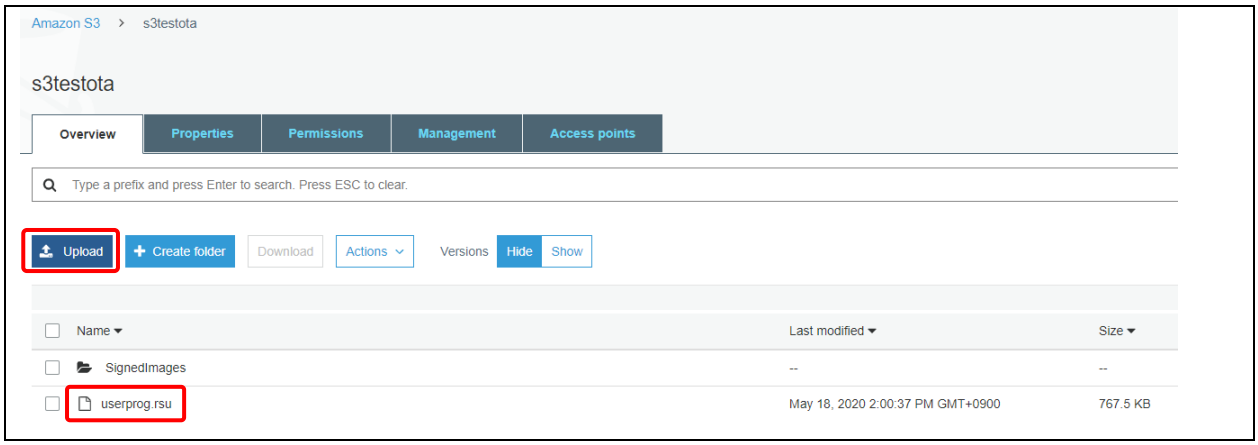
  File Path: location to aws_demos.mot

  (projects\renesas\rx65n-rsk\e2studio\aws_demos\HardwareDebug)
- Create a folder named **update_firmware,** generate **userprog. rsu** and save to **update_firmware** folder and check **generate succeeded**



④ Upload firmware update into the Amazon S3 bucket as described in 1.2 Create an Amazon S3 bucket to store update

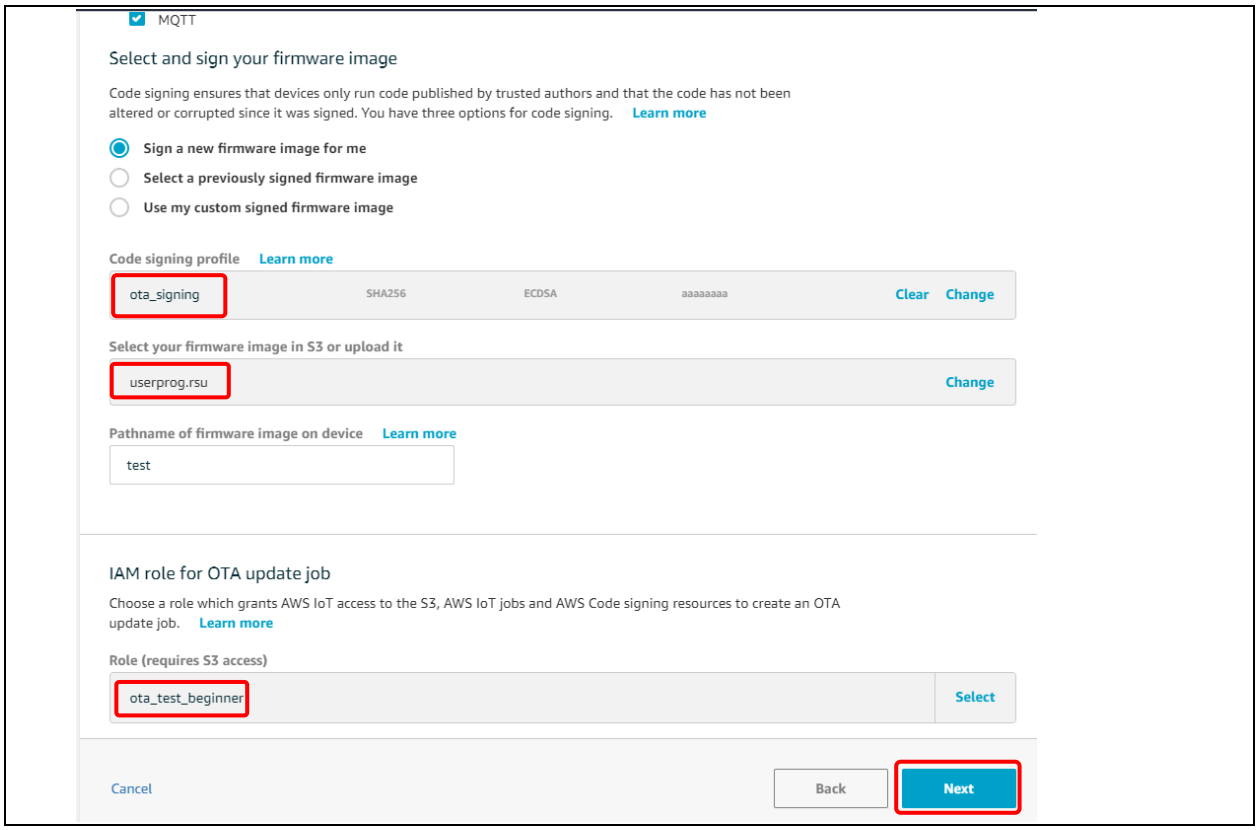
Upload **userprog.rsu** to Amazon S3 bucket



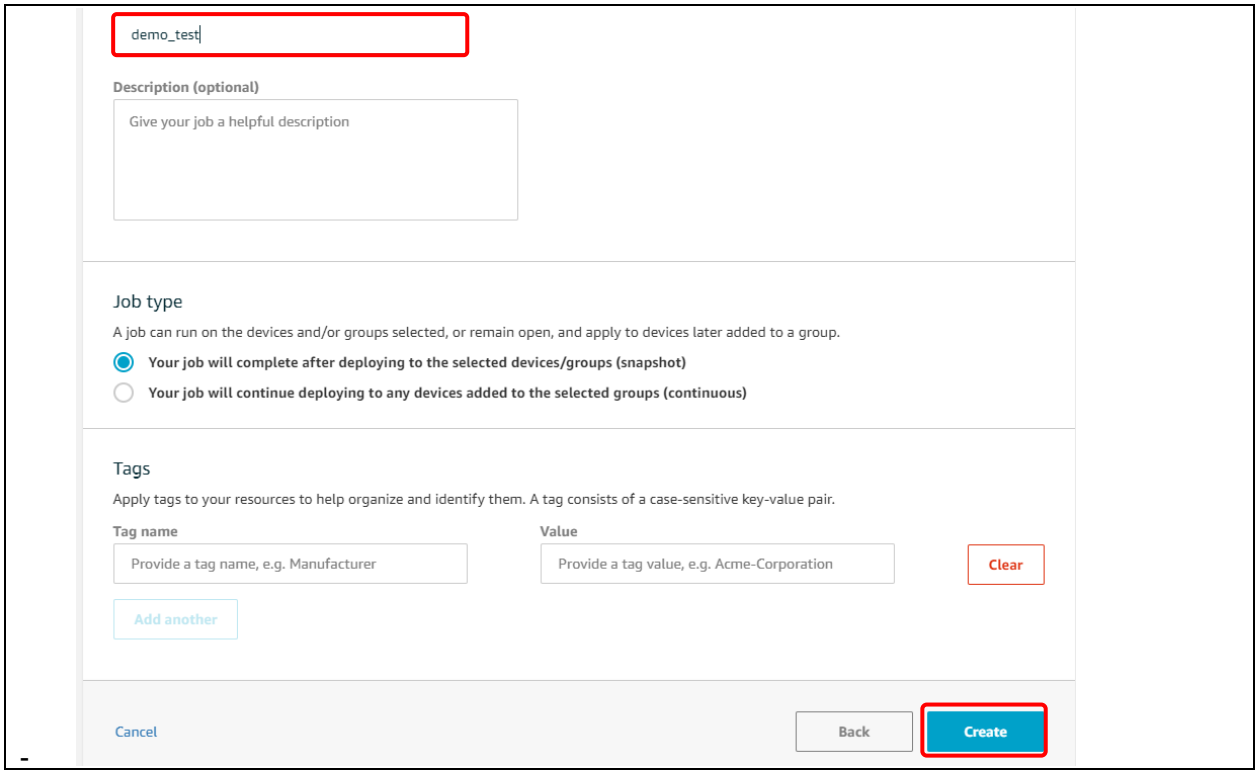
⑤ Create Job to update firmware on RX65N-RSK

AWS IoT Jobs is a service that notifies one or more connected devices of a pending “Job”. A Job can be used to manage fleet of devices, update firmware and security certificates on devices, or perform administrative tasks such as restarting devices and performing diagnostics.

- Go to AWS IoT→ Manage→ Jobs→ Create→ Create OTA Update job→ Choose thing name→ Next
- Create a FreeRTOS OTA update job as below:
  - ➢ Select Code signing profile created in previous section
  - ➢ Select firmware image from S3
  - ➢ Choose IAM role created in previous section
- Click Next

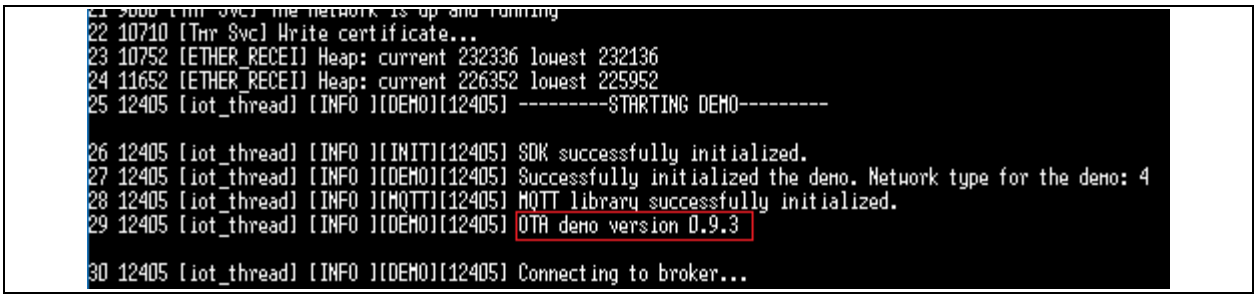


⑥ Give ID and hit Create

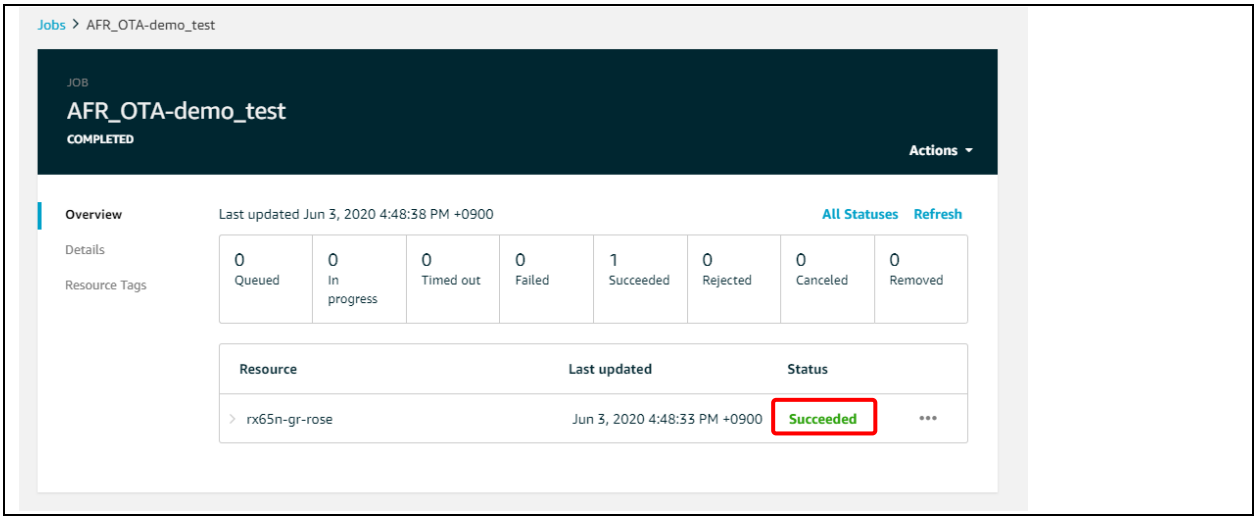


⑦ Reopen Tera Term to see update firmware

OTA demo version is 0.9.3 was updated successfully.



⑧ Check Job status to be “Succeeded” or not.



3 Restriction

This section describes restriction for this application note.

- FreeRTOS OTA programs with big endian operate abnormally.
  Build and operate programs with little endian.

4 Appendices

4.1 Confirmed Operation Environment

This section describes confirmed operation environment for this application note.

Table 4.1 Confirmed Operation Environment (R01AN5549xx0102)

Integrated development environment	e² studio 7.8.0 e² studio 2020-10
C compiler	CC-RX Compiler v3.02.00 GCC 8.3.0.202004
Board used	RSKRX65N-2MB (Part Number: RTK50565Nxxxxxxxxx) RX65N Cloud Kit (Part Number: RTK5RX65Nxxxxxxxxx)
Debuggers	E2 emulator E2 emulator Lite
Software	Amazon FreeRTOS Package v202002.00-rx-1.0.5 Renesas Flash Programmer V3.06.01 Renesas Secure Flash Programmer.exe (mot-file-converter) v1.0.1 Tera Term Version 4.87
Endian	Little endian

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug. 31, 2020	-	First release.
1.01	Oct. 30. 2020	-	Chapter division.
1.02	May. 28. 2021	-	Newly support GCC.
		3	Add more detailed steps at section 1.1 to sign in.
		7-9	Change images to verify “Create an Amazon S3 bucket” step.
		12	Correct wrong name to secp256r1.privatekey at section 1.5.
		27	Add image at section 2.2.
		38	Add section of restriction. Add restriction related to big endian.
		39	Add section of confirmed operation environment. Add the follow confirmed operation environment for R01AN5549xx0102: - Update CC-RX to v3.02.00. - Update GCC to 8.3.0.202004. - Add RX65N Cloud Kit. - Update Amazon FreeRTOS Package to v202002.00-rx-1.0.5. - Update Renesas Secure Flash Programmer.exe to v1.0.1.

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/.