RX ファミリ
RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現方法

R01AN5549JJ0102
Rev.1.02
May.28.21

目的

本アプリケーションノートでは、FreeRTOS with IoT Libraries 上で OTA デモアプリケーションを使用する手順を説明します。セキュリティに関する詳細は「**ルネサス MCU ファームウェアアップデートの設計方針　R01AN5548JJ0100**」を参考してください。

動作確認デバイス

- RX65N、RX651 グループ

ハードウェア

1. RX65N-2MB RSK の場合
   - E2 Lite エミュレータと USB シリアルポートを RX65N ボードと PC に接続します。
   - 電源を RX65N に接続します。
2. RX65N Cloud Kit の場合
   - USB シリアルポートを RX65N ボードと PC に接続します。
   - Wi-Fi-Pmod-Expansion-Board

参考ドキュメント

- ルネサス MCU ファームウェアアップデートの設計方針　(R01AN5548JJ0100)

目次

1. AWS の設定

FreeRTOS デモを実行するには、AWS アカウント（AWS IoT と FreeRTOS クラウドサービスにアクセスできる権限を持つ IAM ユーザ）が必要です。

AWS のアカウントと権限の設定方法は、https://docs.aws.amazon.com/freertos/latest/userguide/freertos-account-and-permissions.html を参照してください。

OTA 更新の設定については、https://docs.aws.amazon.com/freertos/latest/userguide/ota-prereqs.html を参照してください。

次に、https://docs.aws.amazon.com/freertos/latest/userguide/get-started-freertos-thing.html の説明に従って、AWS IoT にボードを登録する必要があります。

デモが AWS と通信できるようにするには、2 章の説明に従ってソースコードを設定する必要があります。

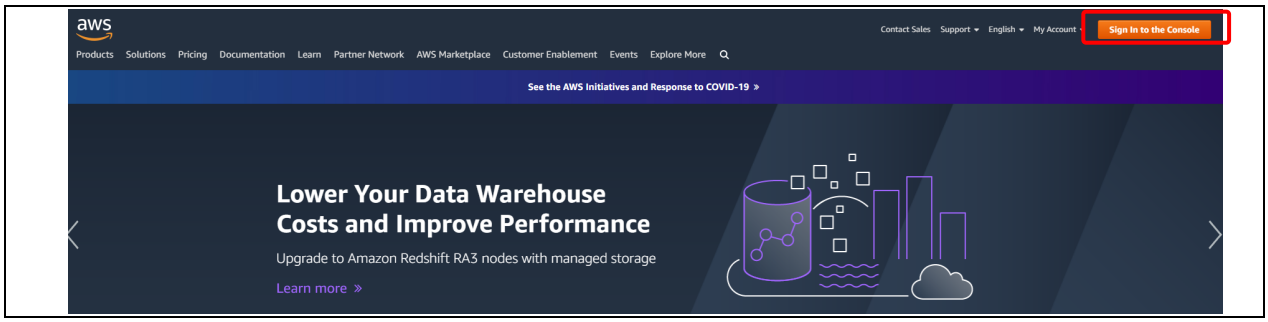
1.1 コンソールへのサインイン

① AWS アカウントの作成が必要です。Set up your AWS Account の説明を参照してください。 これらのセクションで説明されている手順に従って、アカウントとユーザーを作成し、開始してください。

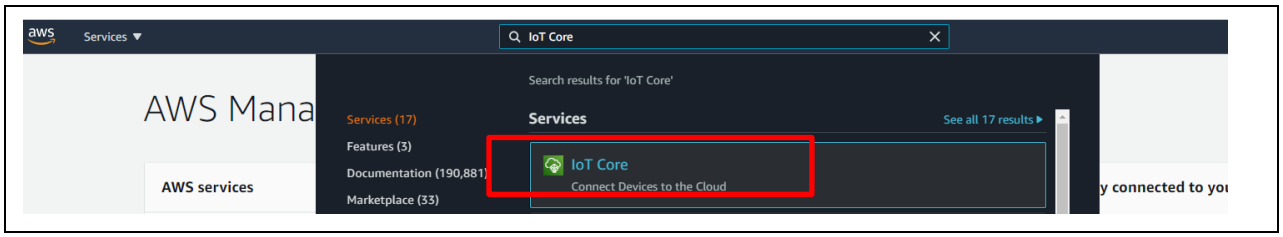
- AWS アカウントにサインアップする。
- ユーザーを作成し、権限を付与する。
- AWS IoT コンソールを開きます。

注意事項に特に注意してください。

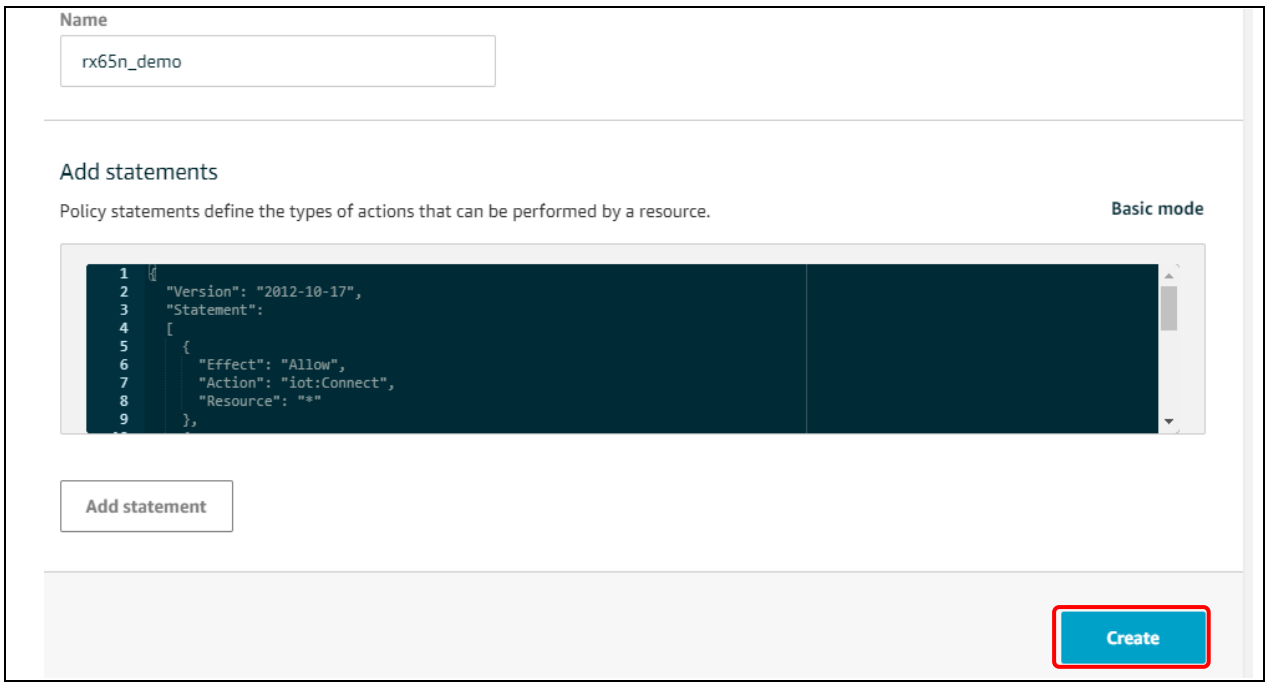
ユーザーが過去にすでにアカウントを作成している場合は、このステップをスキップしてください。



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



② [安全性] → [ポリシー]に移動してポリシーを作成します。
AWS IoT ポリシーは、AWS IoT リソースにアクセスするための権限をデバイスに付与します。AWS Cloud に格納されています。



③ アドバンスモードを選択して次のコードをコピーします。

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

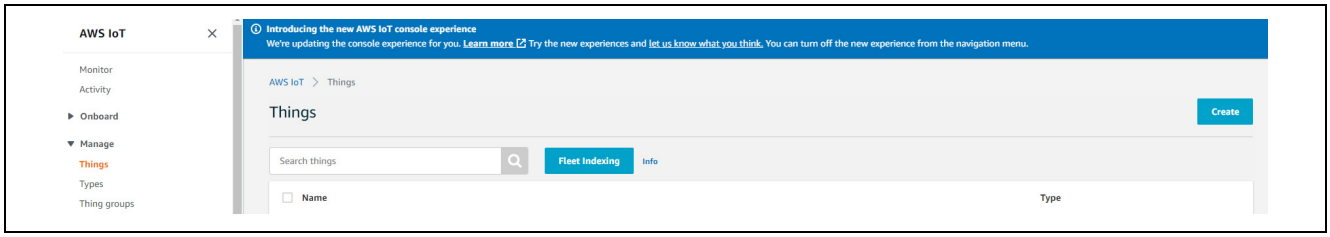
注：このドキュメントの例は、開発環境のみを対象としています。フリート内のすべてのデバイスは、特定のリソースに対して意図されたアクションのみを許可する権限を持つクレデンシャルを持つ必要があります。具体的なパーミッション・ポリシーは、ユースケースによって異なります。ビジネスとセキュリティの要件を最もよく満たす権限ポリシーを特定してください。詳細については、「ポリシーの例」および「セキュリティのベストプラクティス」を参照してください。

④　[管理] → [モノ]に移動して「モノ」を作成します。

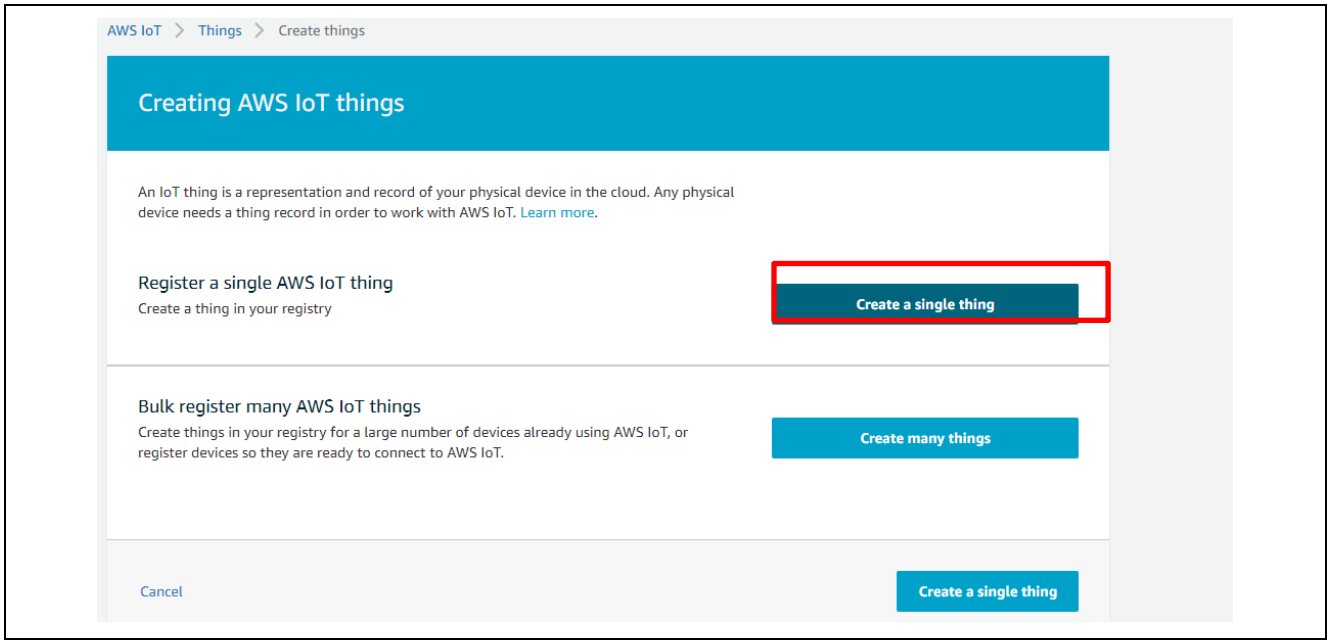
「モノ」とは、AWS IoT に接続されたデバイスや論理エンティティの総称です。物理的なデバイスやセンサー（電球や壁のスイッチなど）は、モノとして扱うことができます。また、アプリケーションのインスタンスといった論理エンティティや、AWS IoT には直接接続されていないものの、AWS IoT に接続されているデバイスに関連する物理エンティティ（たとえば、エンジンセンサーやコントロールパネルを搭載した自動車）もモノです。AWS IoT では、モノのレジストリによってモノの管理を支援しています。

— [単一のモノを作成する]を選択します。
— モノに名前を付けます。
— [証明書の作成]をクリックします。
— 3 つのファイルをダウンロードします。
— 第 1 項の手順②で作成したポリシーを割り当てます。

管理→ モノ→作成

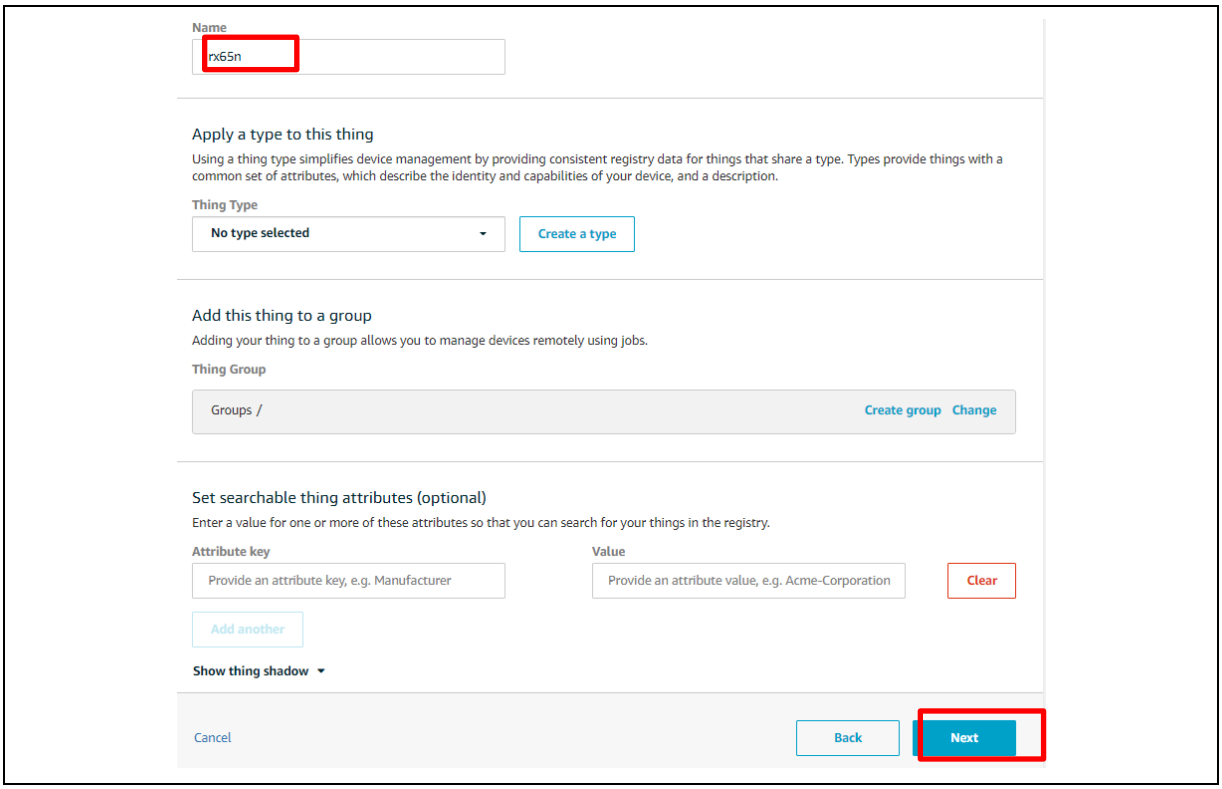


[単一のモノを作成する]を選択します。



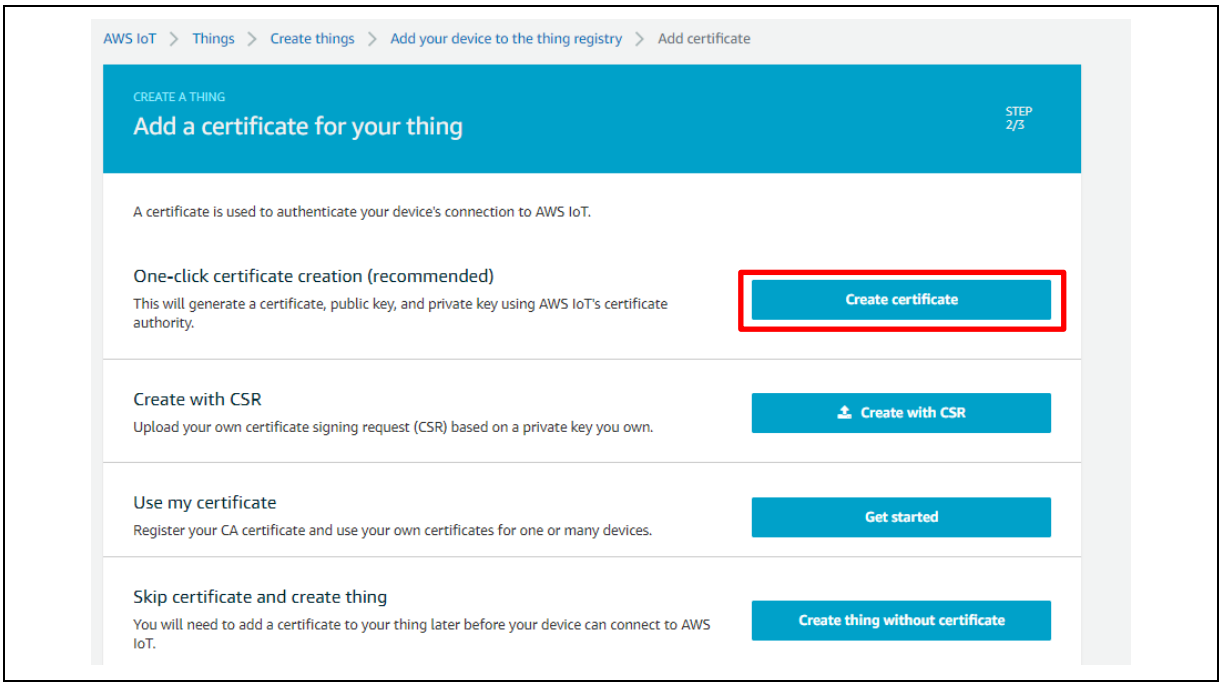
Create a single thing

モノに名前を付けます。



Add name to a single thing

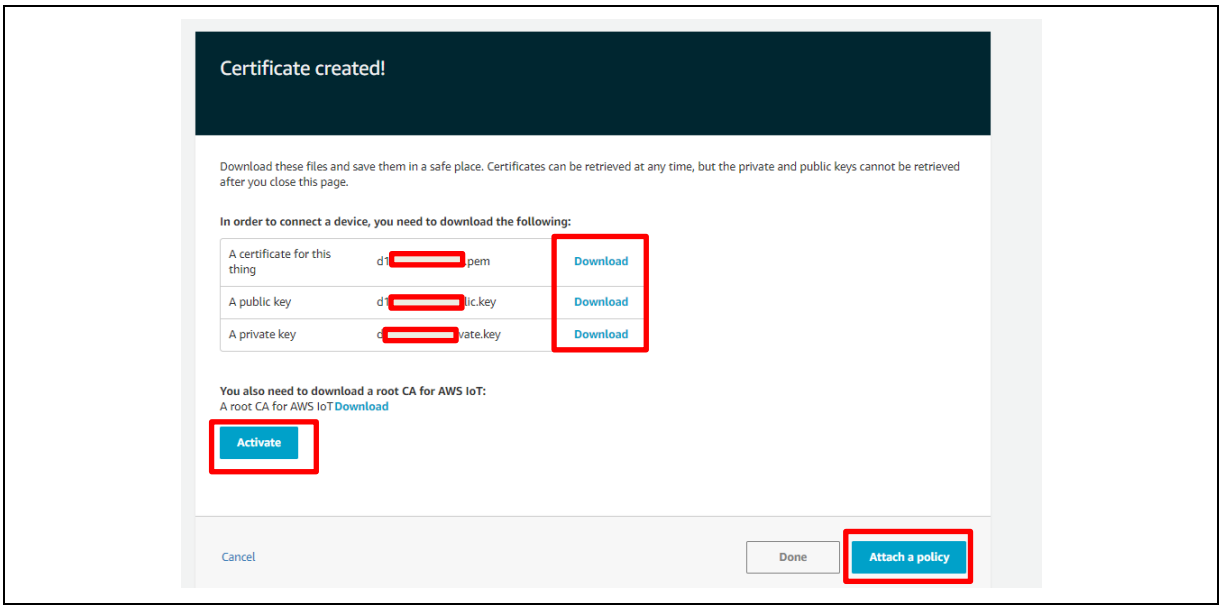
[証明書の作成]をクリックします。



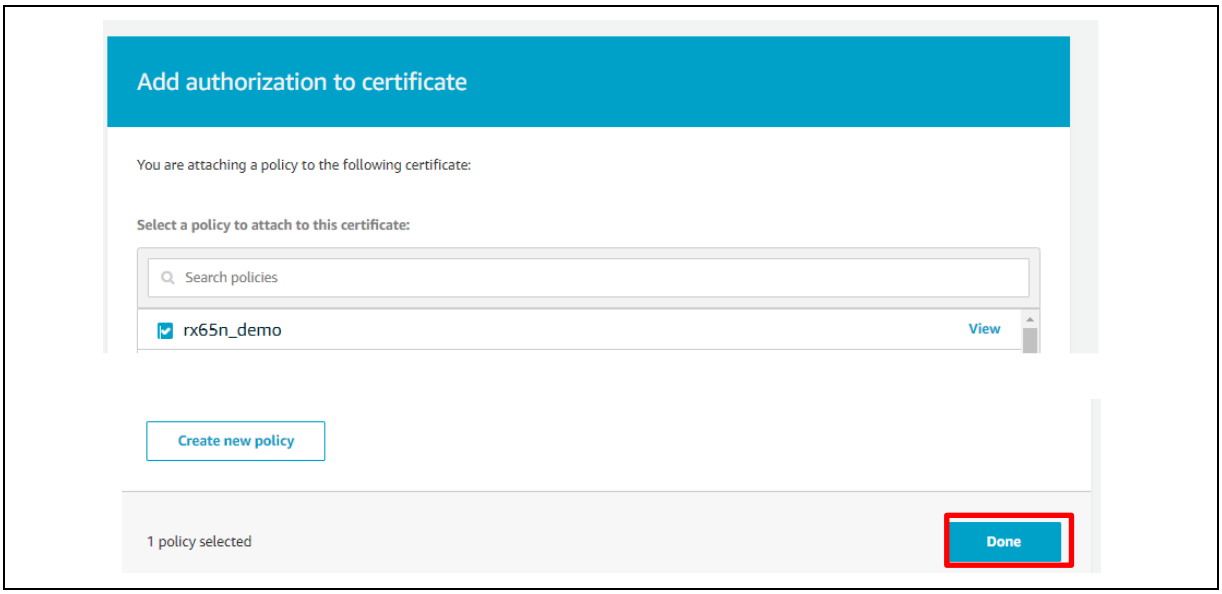
Create a certificate for thing

モノにポリシーを付けます

- 各証明書、鍵の横にあるダウンロードボタンをクリックし、ローカル PC またはホストマシンに保存します。
- Activate ボタンをクリックして、証明書を有効にします。
- ポリシーの添付を選択します。



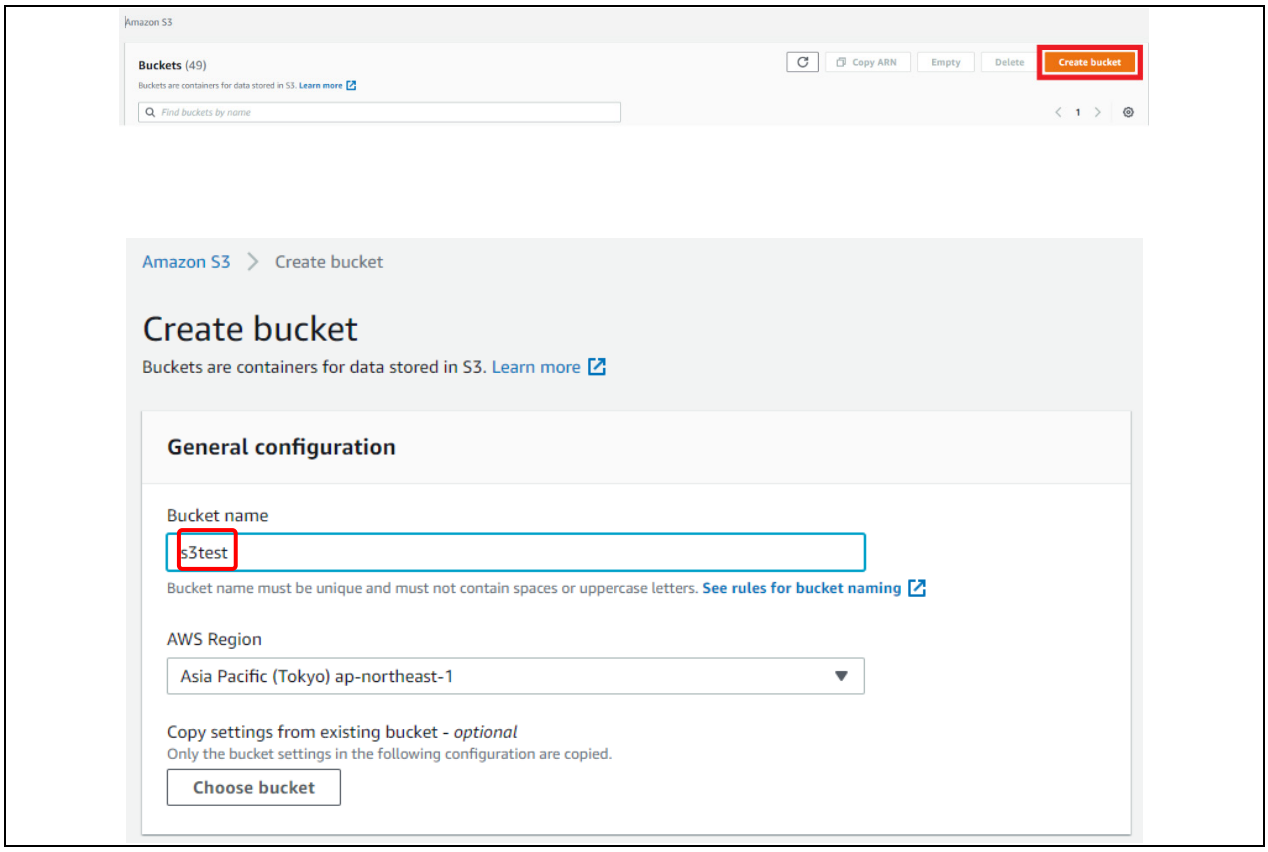
Register policy to thing



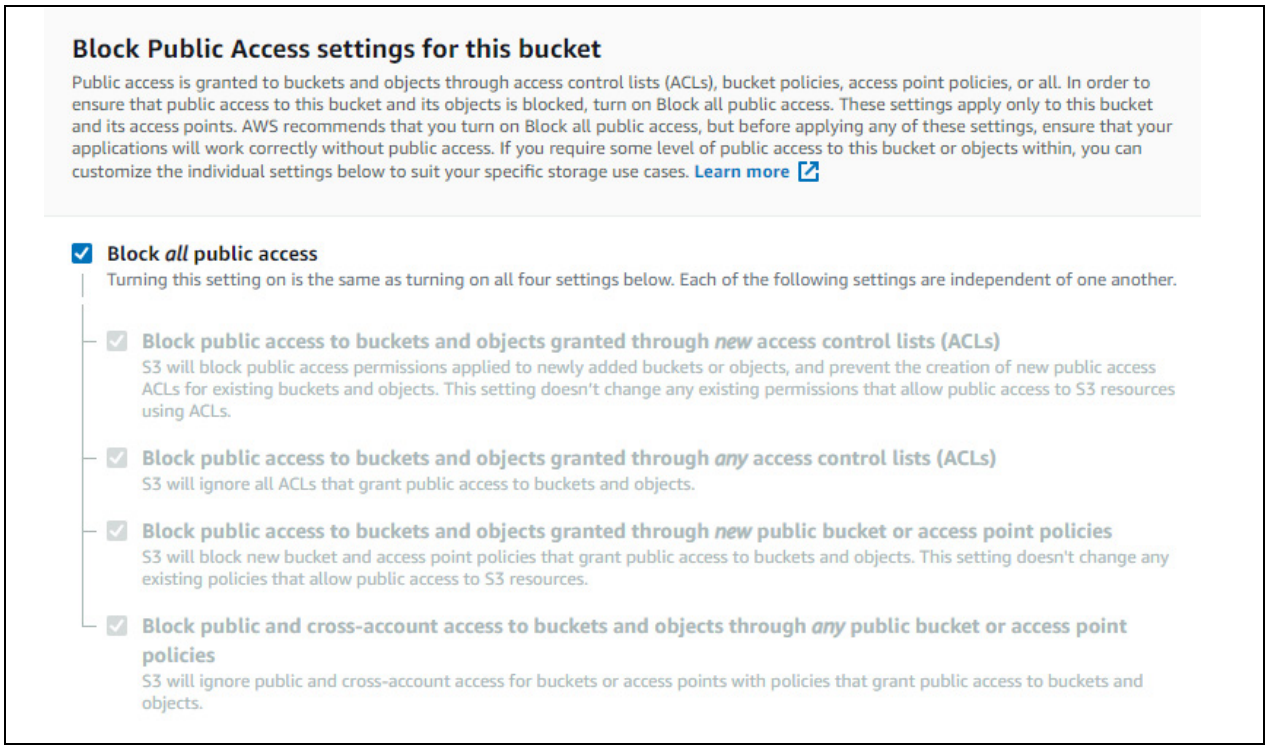
1.2　Amazon S3 バケットの作成

① 各種のサービスから、ファイルをクラウドに保存するための Amazon Simple Storage Service（S3）AWS にアクセスできます。OTA 更新ファイルは、Amazon S3 バケットに保存されます。
https://docs.aws.amazon.com/freertos/latest/userguide/dg-ota-bucket.html を参照してください。

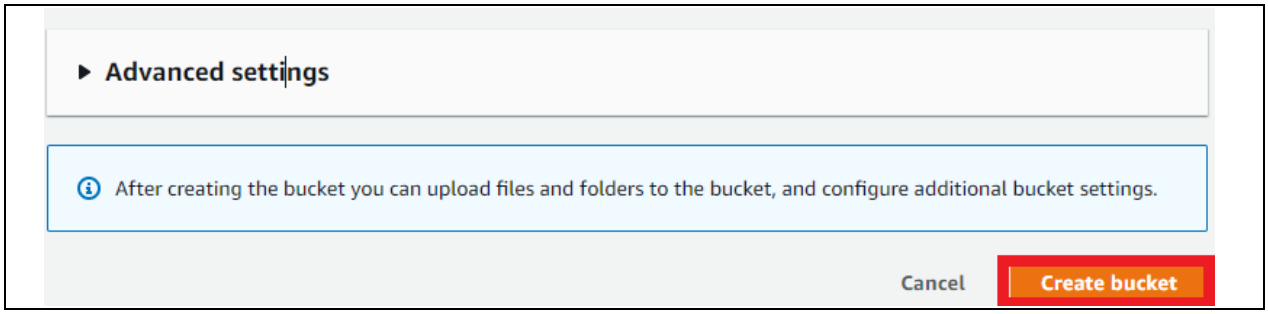
② Bucket name を入力し、Create bucket を押します。



③ **Block *all* public access** 選択。

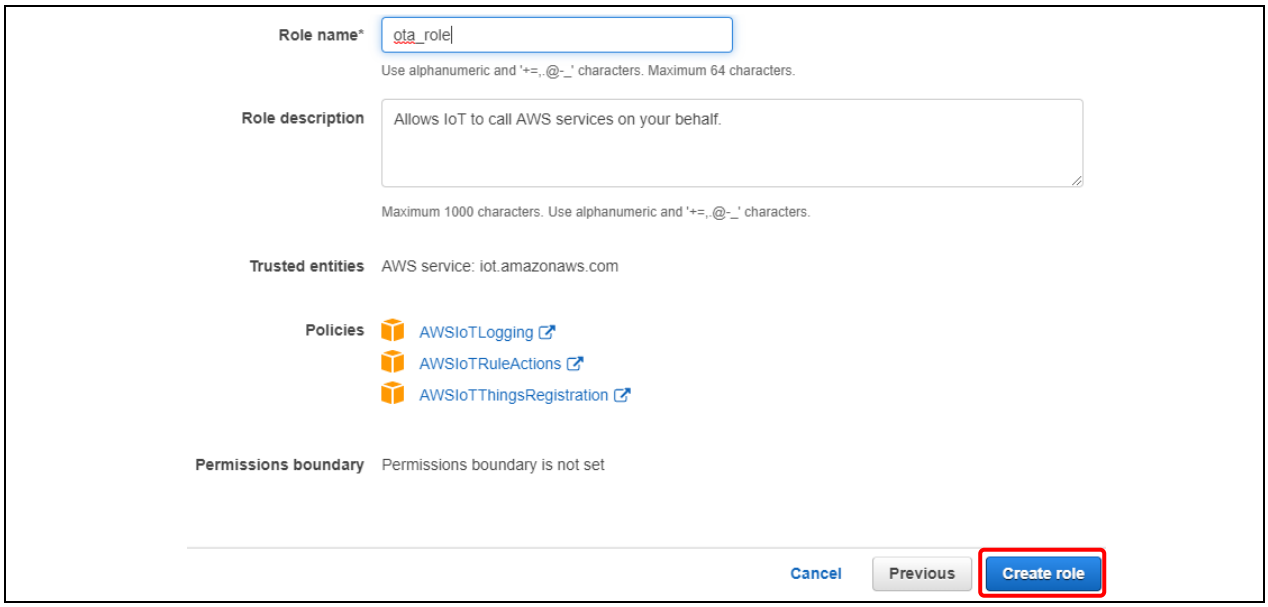


④　**Create bucket** 選択



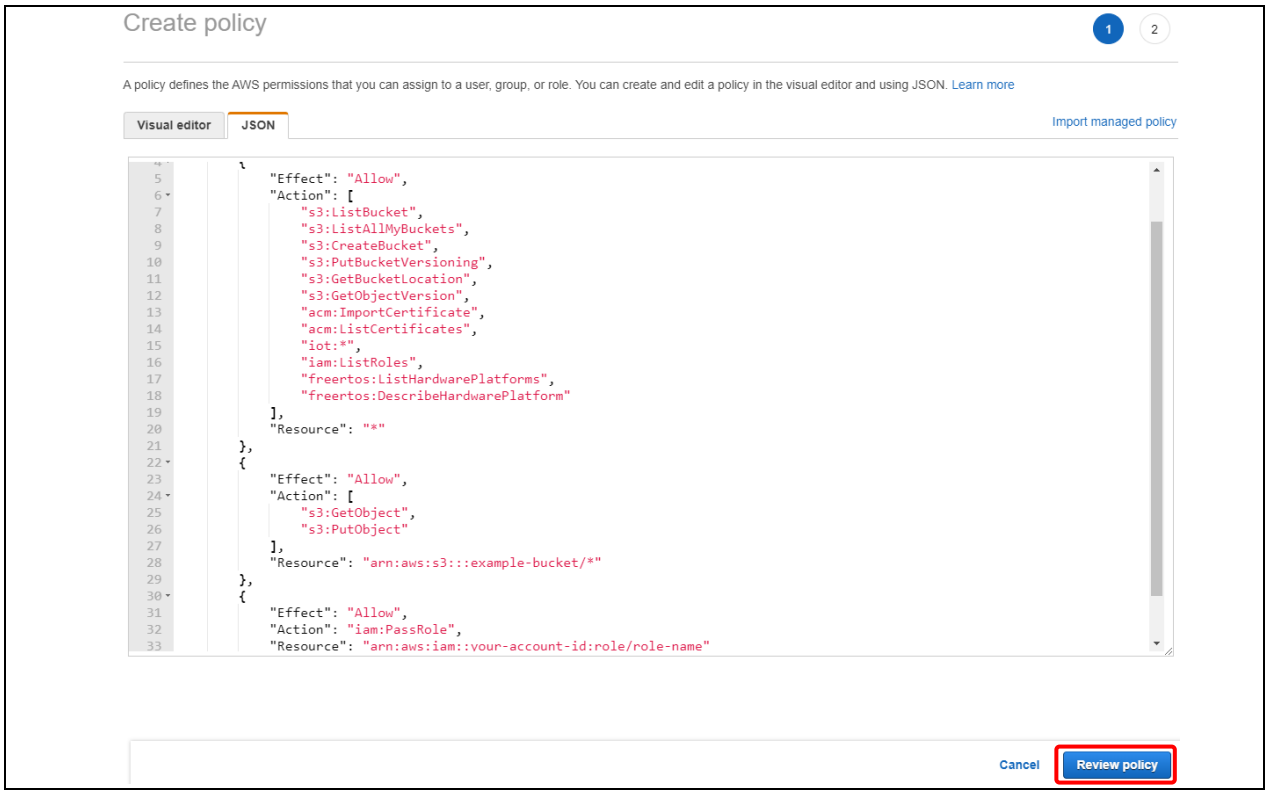
1.3　OTA 更新用サービスロールの作成

Identity Access Management（IAM）は、AWS リソースへのアクセスを安全に管理するのに役立ちます。
https://docs.aws.amazon.com/freertos/latest/userguide/create-service-role.html を参照してください。

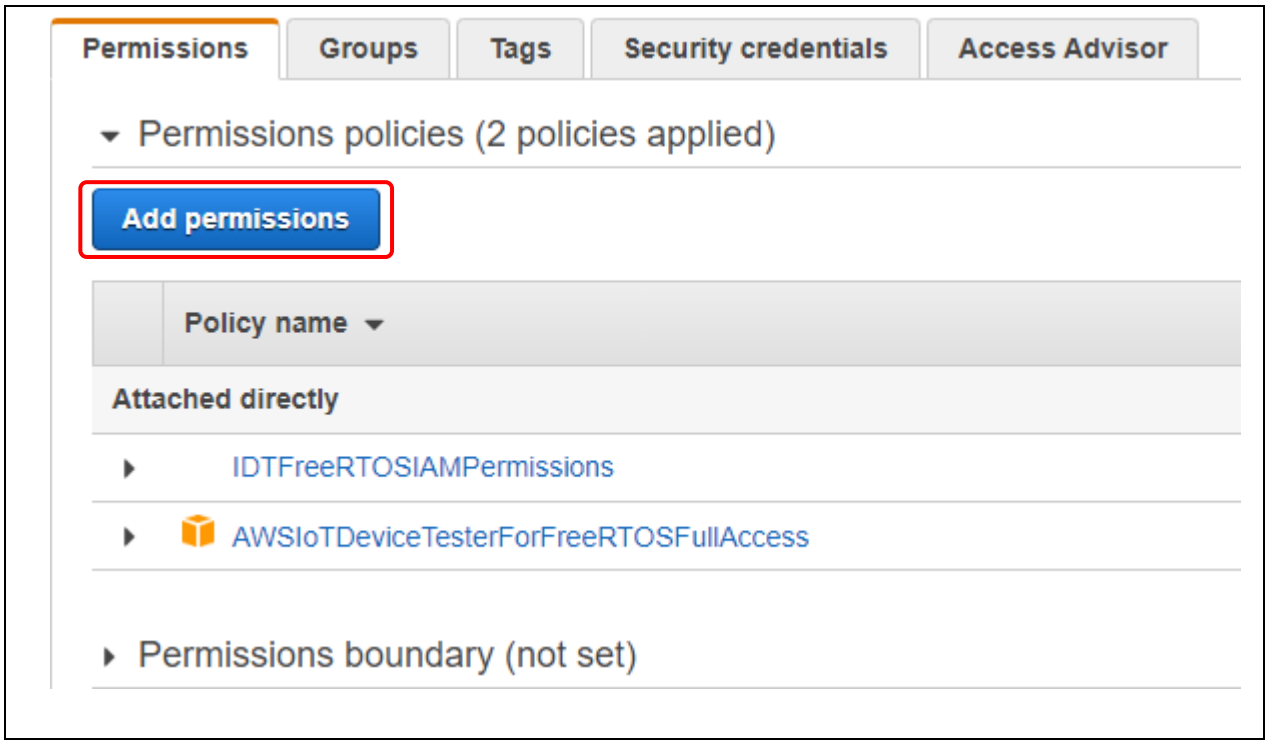


1.4　OTA 更新用ユーザポリシーの作成と IAM ユーザの割り当て

① OTA 更新用ユーザポリシーを作成します。
https://docs.aws.amazon.com/freertos/latest/userguide/create-ota-user-policy.html を参照してください。



② OTA ユーザポリシーを IAM ユーザに割り当てます。

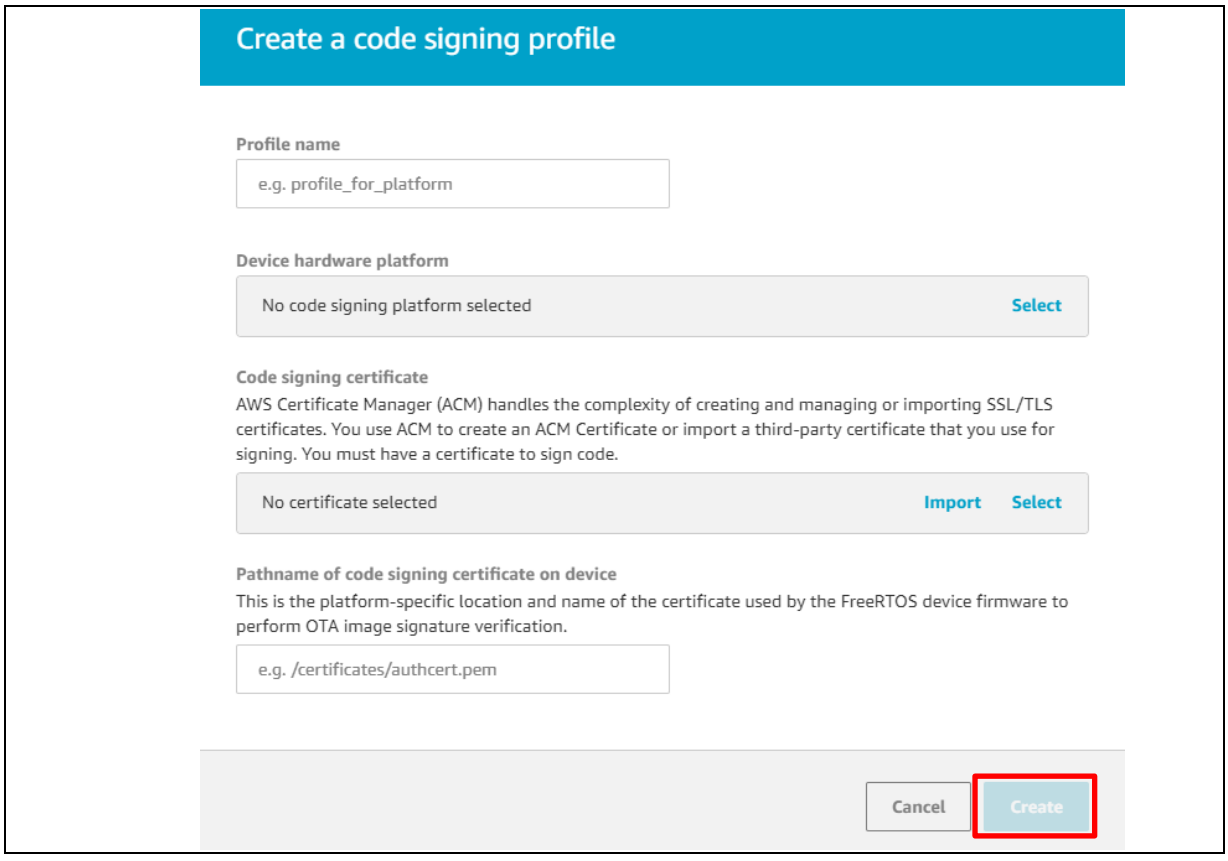


1.5　AWS にコード署名証明書の登録

AWS にコード署名証明書を登録します。

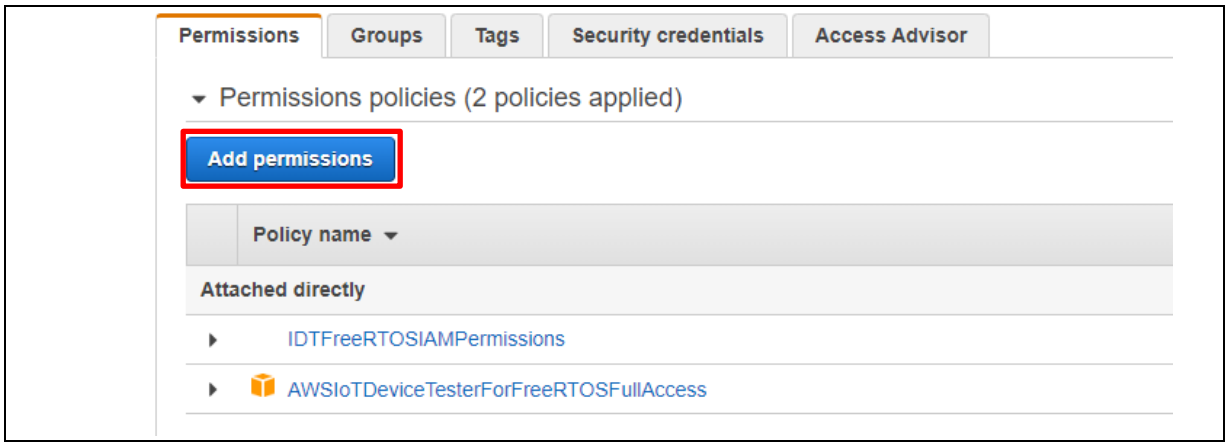
- 「ルネサス MCU ファームウェアアップデートの設計方針」7.3 OpenSSL での ECDSA+SHA256 用の鍵ペア生成方法を参照し、鍵と証明書を生成してください。
- [IoT Core] → [管理] → [ジョブ] → [作成] → [Amazon FreeRTOS 無線通信経由（OTA）の更新ジョブの作成] → [Select a job] で更新するデバイスを選択 → [新しいファームウェアイメージに署名します] で [Select] を選択し、作成してあるいずれかのモノを選択 → [次へ] → [コード署名プロファイル] で [作成] を選択します。

✓ プロファイル名：任意
✓ デバイスハードウェアプラットフォーム： Windows Simulator
✓ コード署名証明書：
✓ 証明書を選択：secp256r1.crt を指定
✓ 証明書のプライベートキーを選択：secp256r1.privatekey を指定
✓ 証明書チェーンを選択（オプション）：ca.crt
✓ デバイスのコード署名証明書のパス名：任意



1.6　AWS IoT のコード署名へのアクセス権付与

AWS IoT のコード署名へのアクセス権を付与します。
https://docs.aws.amazon.com/freertos/latest/userguide/code-sign-policy.html を参照してください。



2. FreeRTOS OTA 環境構築

最初に、ユーザは Amazon FreeRTOS パッケージのバージョンを選択できます。選択されたバージョンが自動的に GitHub からダウンロードされ、プロジェクトにインポートされます。そのため、ユーザは Amazon FreeRTOS の設定とアプリケーションコードの作成に集中できます。

（注：2.2 および 2.3 で操作の間違い等で最初からやり直したい場合は、2.2 の「⑥RX65N-RSK のフラッシュ ROM を消去します。」を実行してからやり直してください）

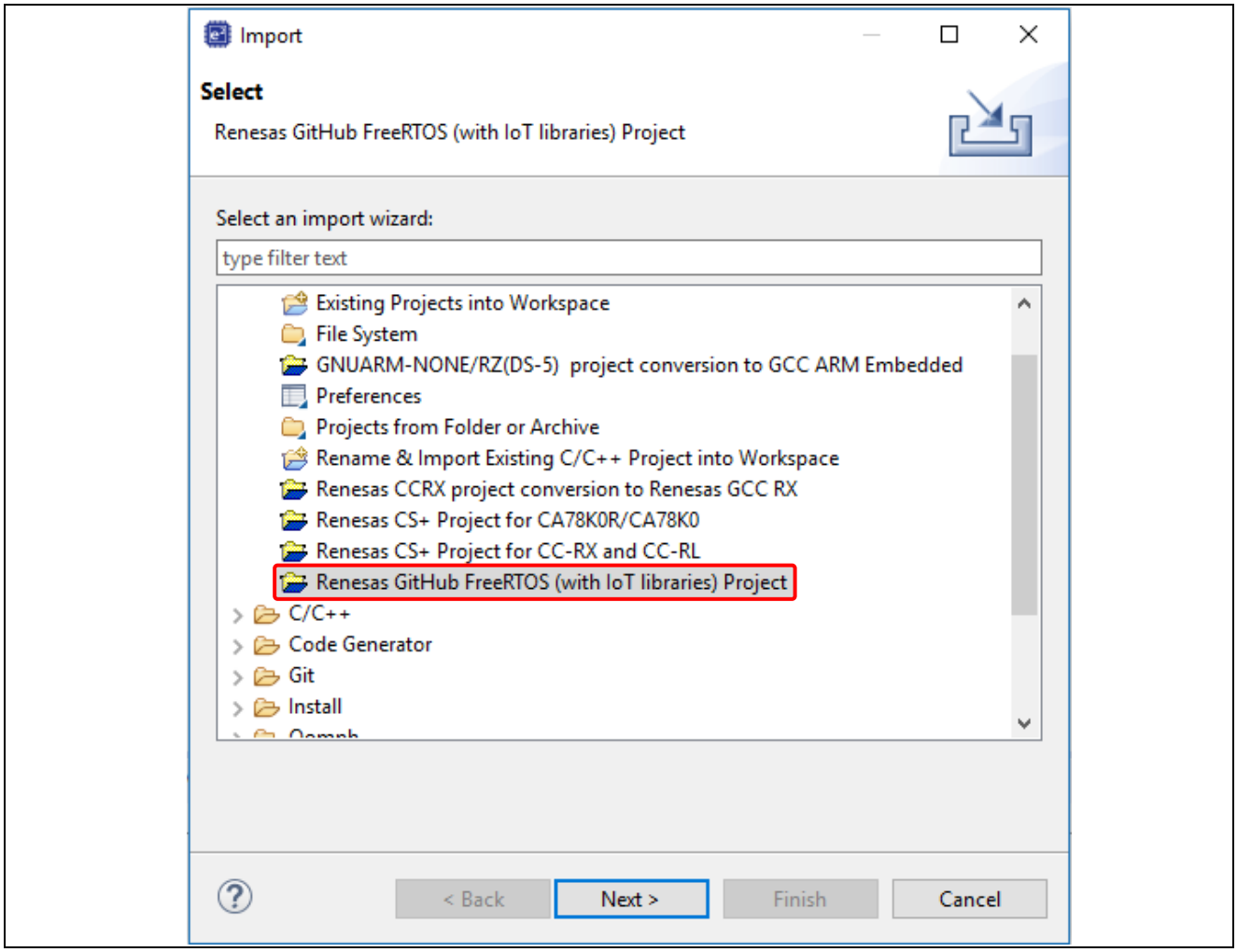
2.1 ヘッダファイルのインポートと設定、および aws_demos と boold_loader の構築

Amazon FreeRTOS プロジェクトのインポート方法を下図に示します。

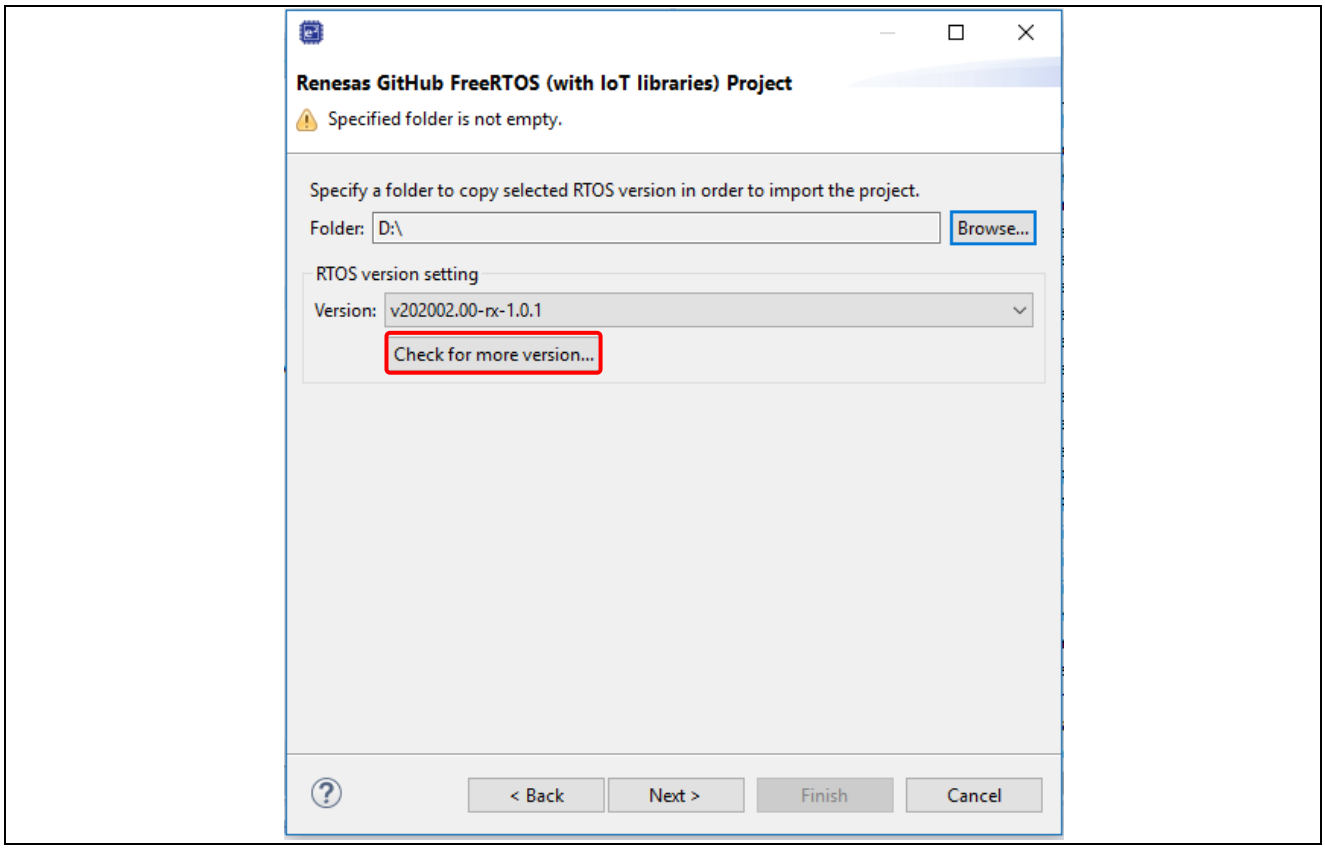
① e² studio を起動します。

② [ファイル] → [インポート...]を選択します。

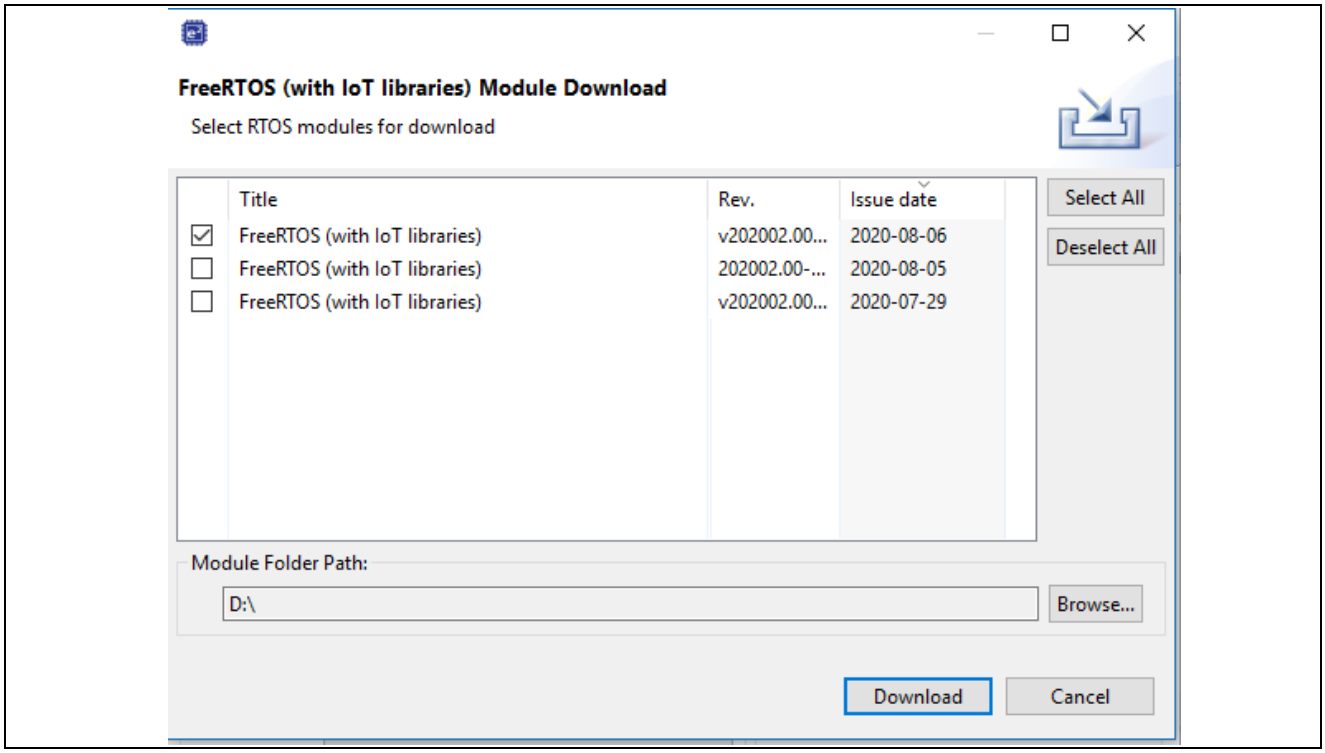
③「Renesas GitHub FreeRTOS(with IoT libraries) Project」を選択します。



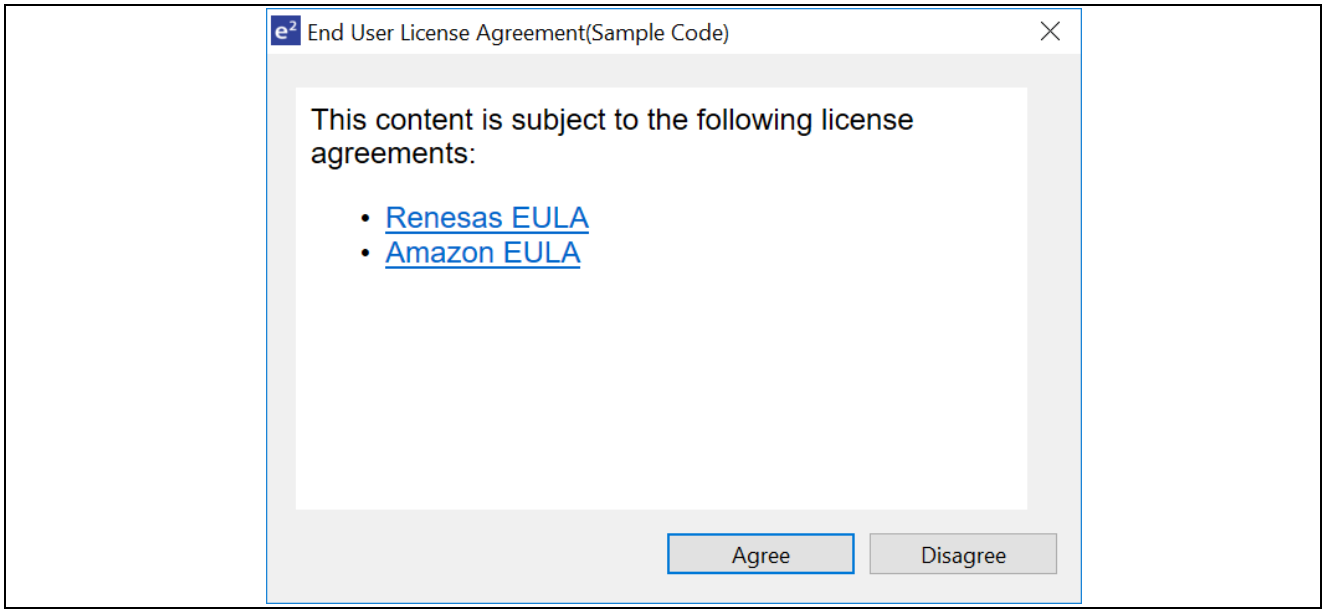
④ [Check for more version…]を押し、”FreeRTOS(with IoT libraries”ダイアログを表示させます。



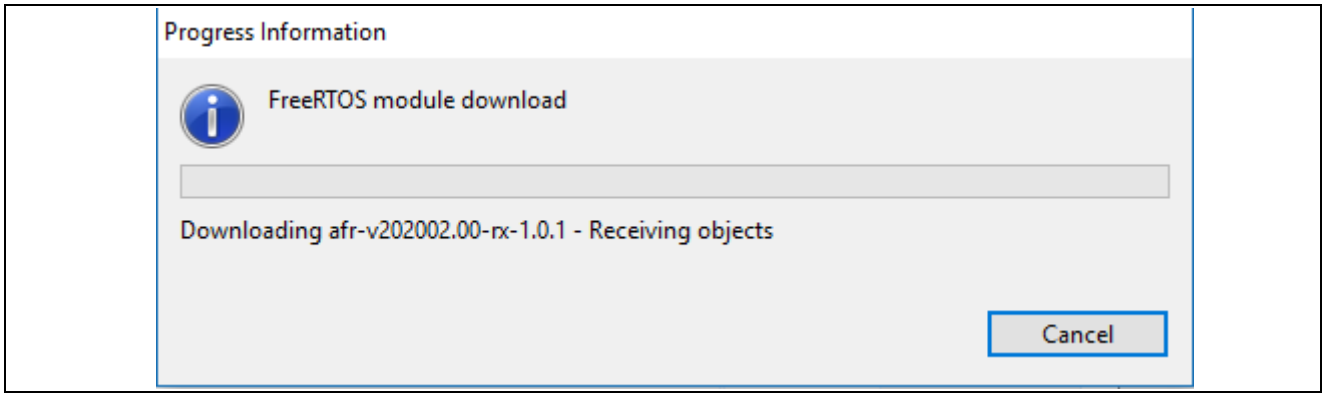
⑤ 最新バージョンを選択します（最新バージョンが表示されない場合、e² studio のワークスペースを新規に作成してください）。



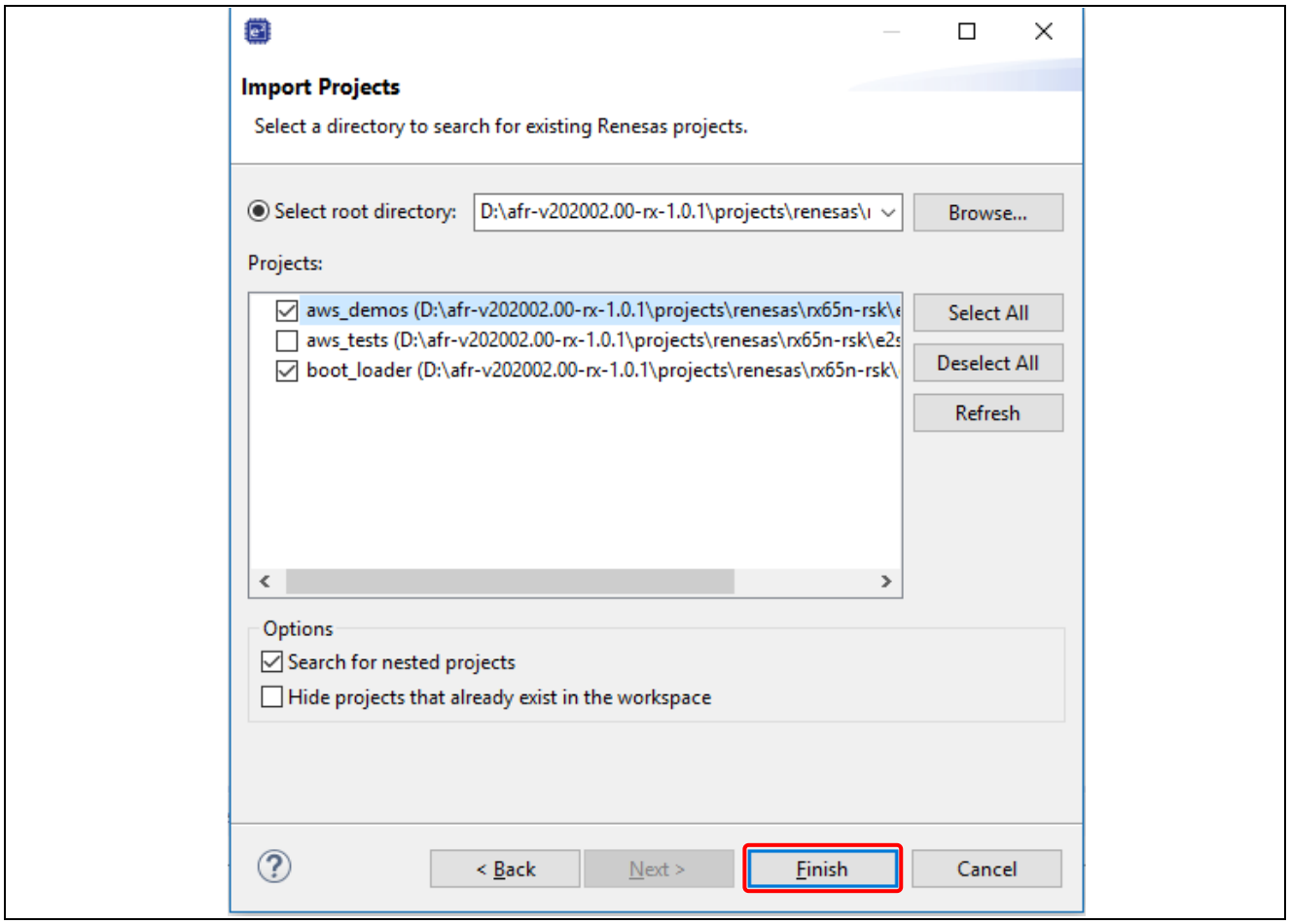
⑥ エンドユーザライセンス契約に同意します。



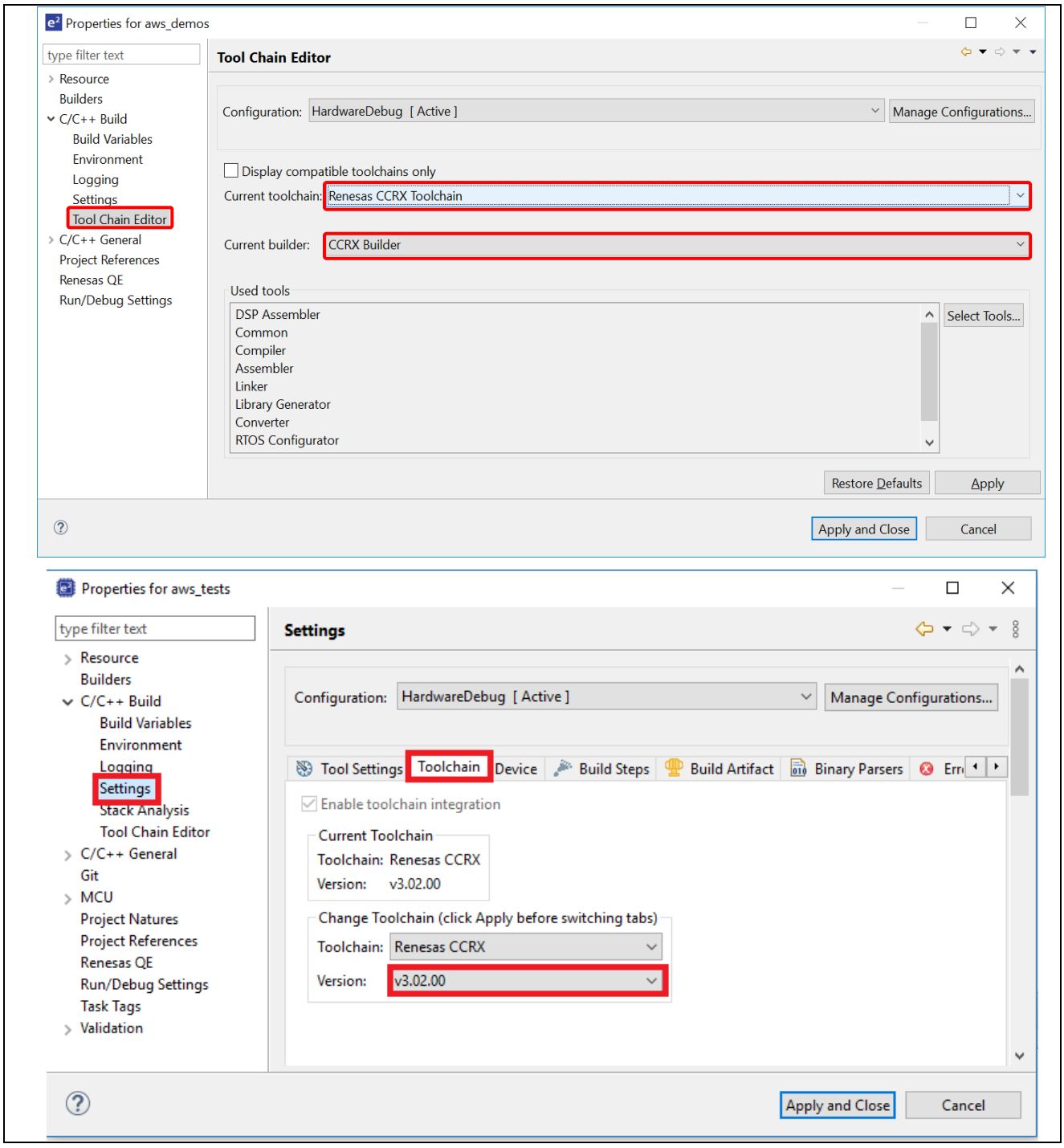
⑦ ダウンロードが完了するまで待ちます。



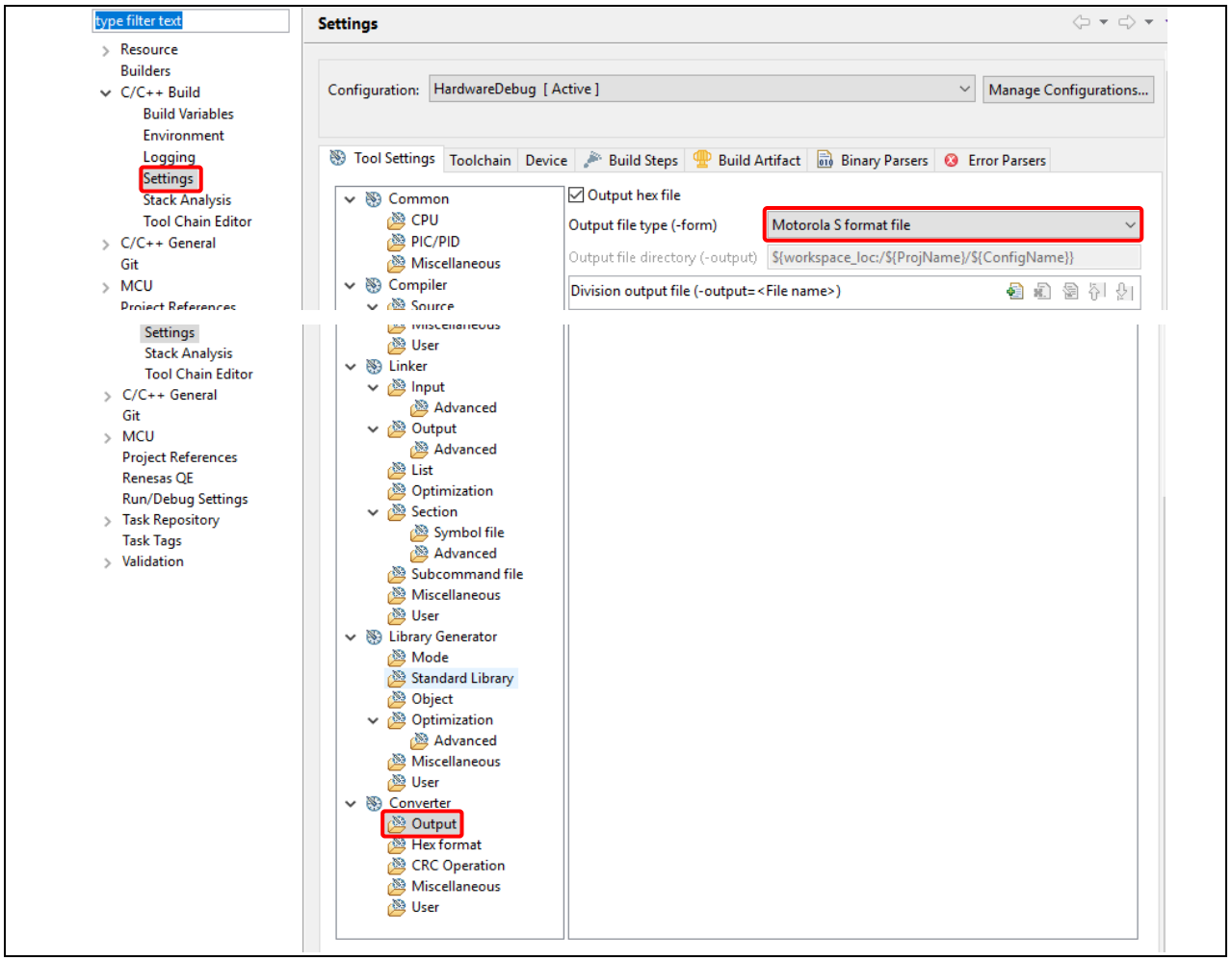
⑧ インポートするプロジェクトを選択します。”aws_demos”と”boot_loader”プロジェクトを選択します。



⑨ 両方のプロジェクトの[プロジェクト]→[プロパティ]→C/C++ビルド→ツールチェーン・エディタを開き、ツールチェーンとビルダーを選択して、ツールチェーンバージョンを指定します。



⑩ motorola S format ファイルの出力を設定します。

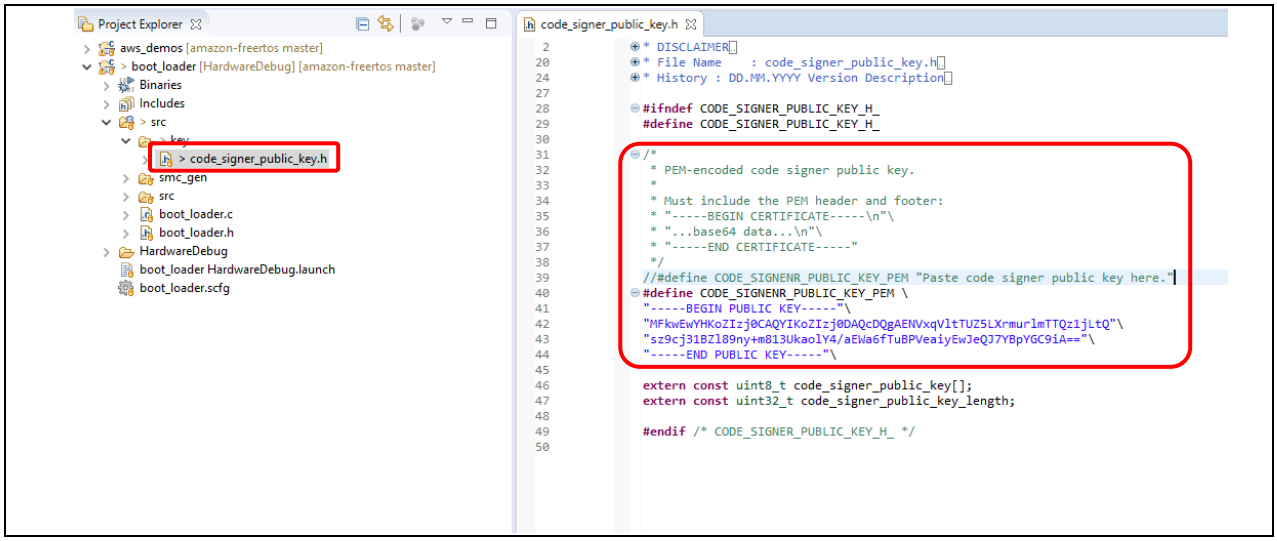


⑪ 公開鍵を入力します。

boot_loader プロジェクトで以下のファイル（projects¥renesas¥rx65n-rsk¥e2studio¥boot_loader¥src¥key¥code_signer_public_key.h）を開き公開鍵を入力します。

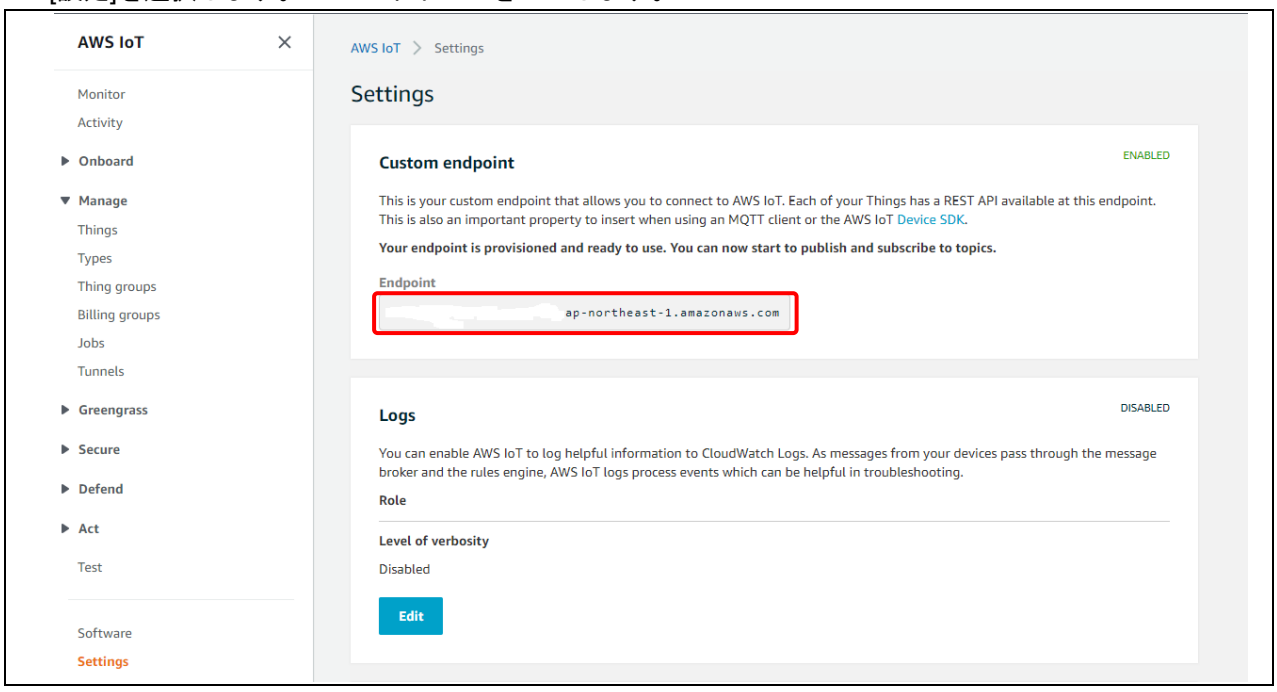
「ルネサス MCU ファームウェアアップデートの設計方針」7.3 OpenSSL での ECDSA+SHA256 用の鍵ペア生成方法 を参照して、公開鍵を生成してください。

完了したら、  を押してビルドし、boot loader の boot_loader.mot ファイルを生成してください。

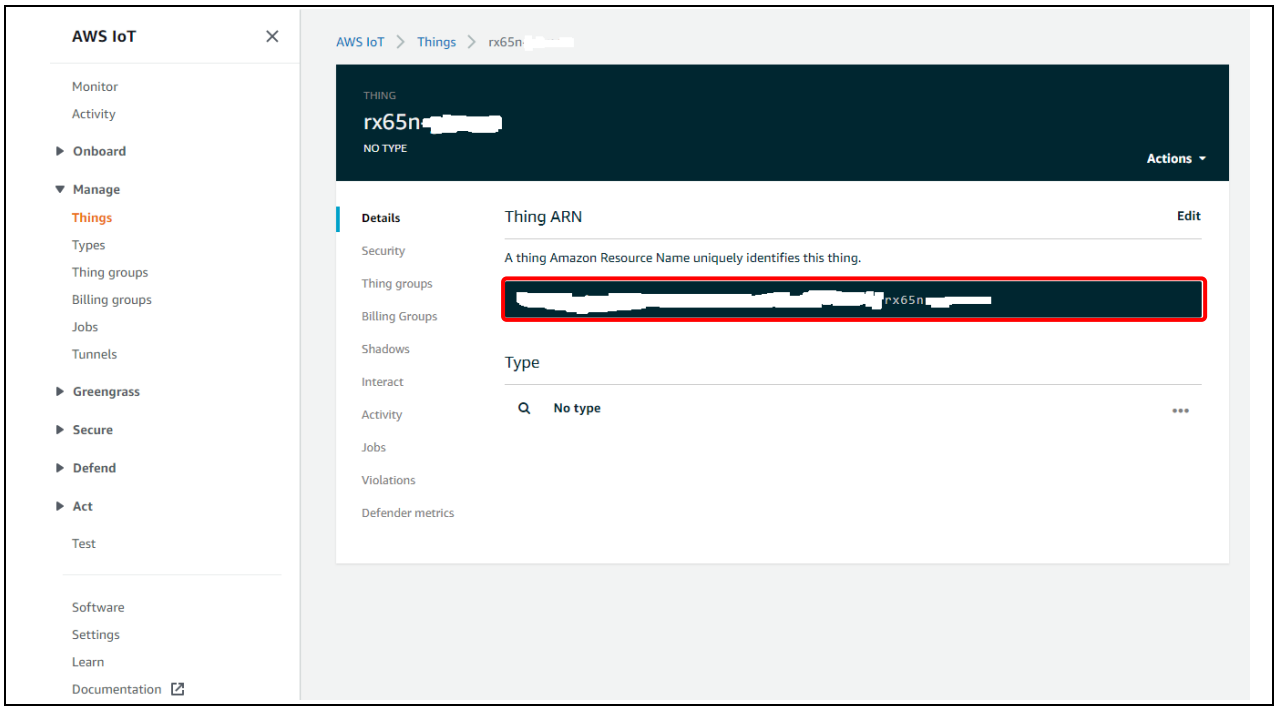


⑫ AWS IoT コンソールを開きます。

— AWS IoT コンソールに移動します。
— [設定]を選択します。エンドポイントをメモします。

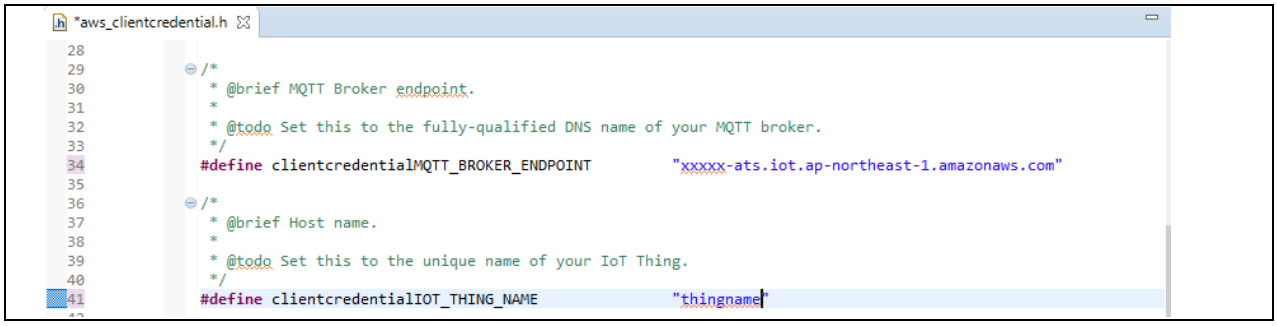


— [管理] → [モノ]を選択します。AWS IoT のモノの名前をメモします。



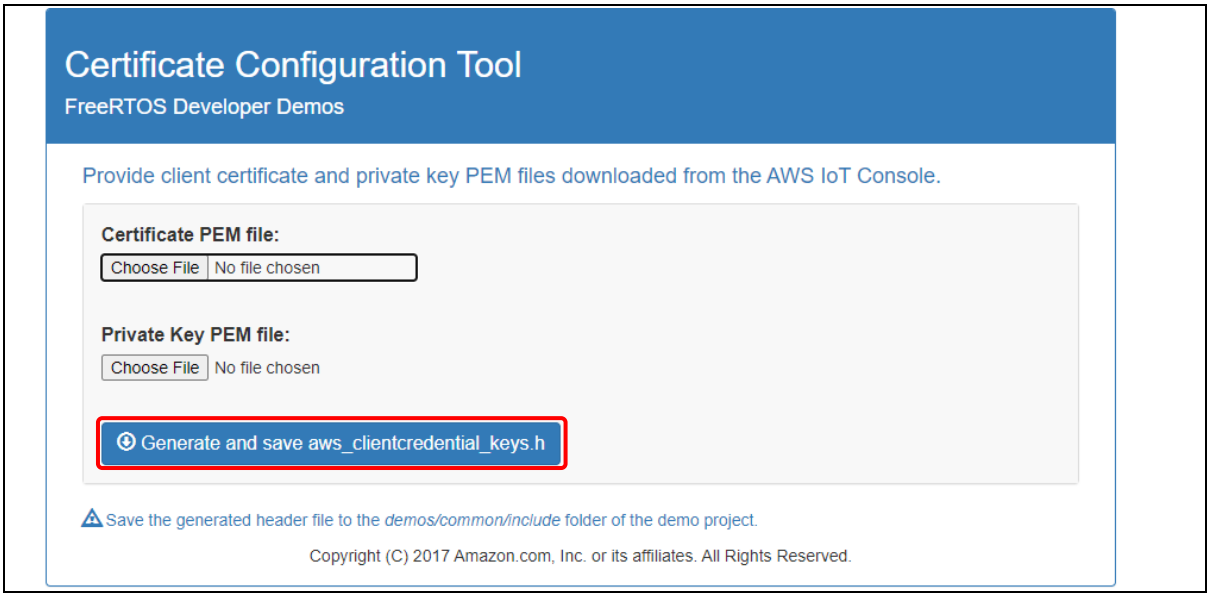
⑬ aws_demo プロジェクトを開きます。

— /demos/include/aws_clientcredential.h を開いて次の値を指定します。
#define clientcredentialMQTT_BROKER_ENDPOINT "Your AWS IoT endpoint";
#define clientcredentialIOT_THING_NAME "The AWS IoT thing name of your board"



⑭ Certificate Configuration Tool を開きます。

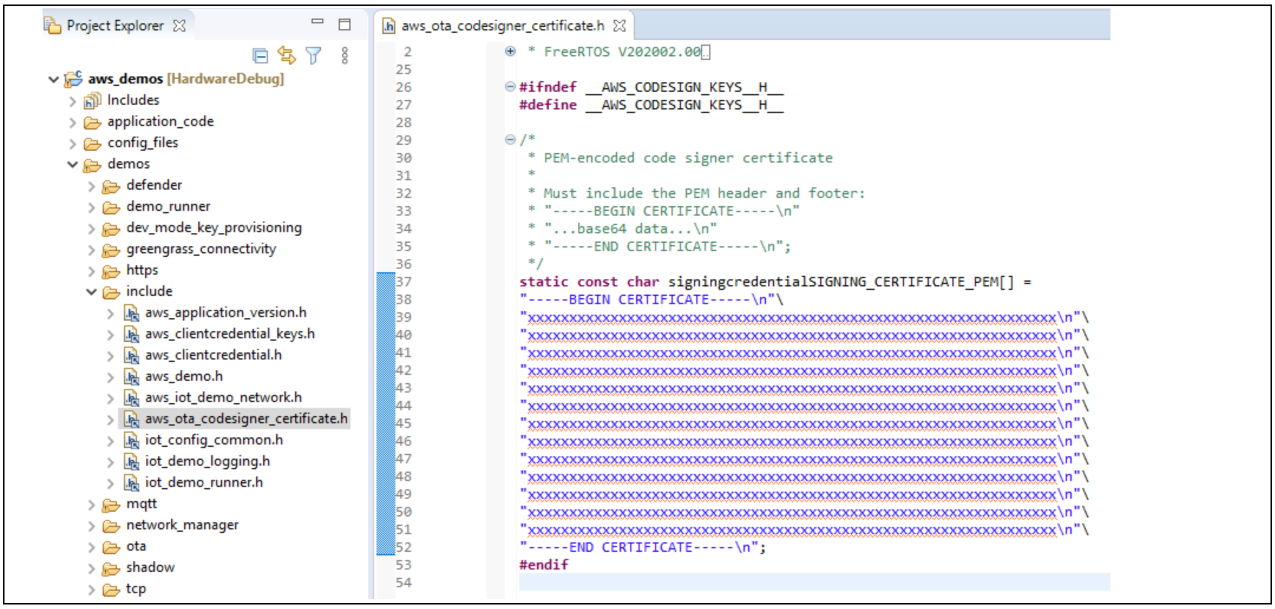
— 2.1 ⑤でダウンロードした FreeRTOS が格納されたパスに移動します。
— tools → certificate_configuration → CertificateConfigurator.html を開きます。
— 1.1 の④でダウンロードした証明書の PEM ファイルとプライベートキーの PEM ファイルをインポートします。
— aws_clientcredential_keys.h を生成します。



⑮ 再度 aws_demo プロジェクトを開きます。

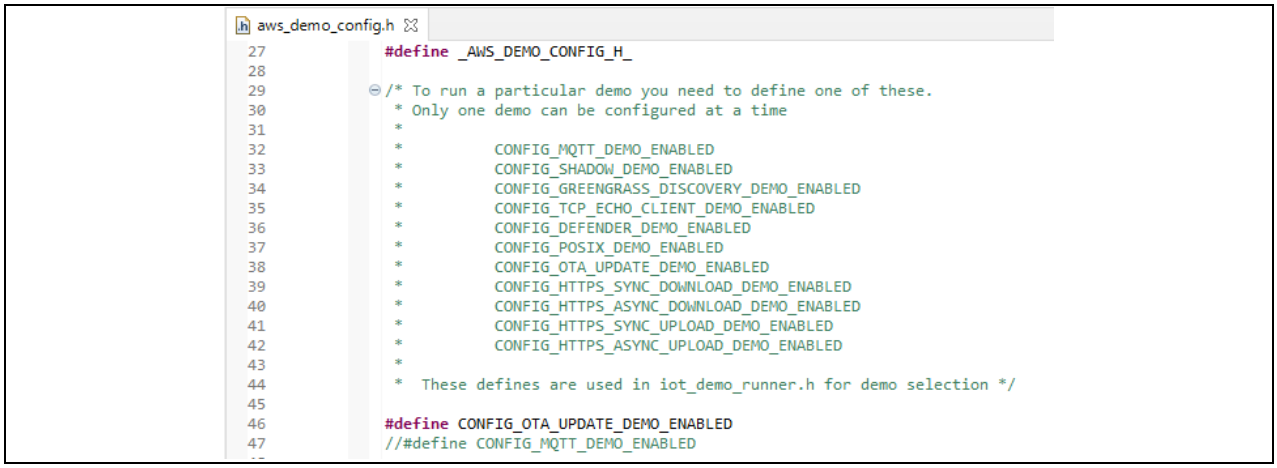
— ⑭で生成した aws_clientcredential_keys.h を/demos/include/にあるファイルと置き換えます。

— /demos/include/aws_ota_codesigner_certificate.h を開いて次の値を指定します。
signingcredentialSIGNING_CERTIFICATE_PEM [] = "xxxx";
"xxxx"には、secp256r1.crt の値を指定します。
secp256r1.crt の値をコピーするときに、各行の最後に“¥”が必要です。忘れずに追加してください。
secp256r1.crt については、「ルネサス MCU ファームウェアアップデートの設計方針」7.3 OpenSSL での ECDSA+SHA256 用の鍵ペア生成方法 を参照してください。

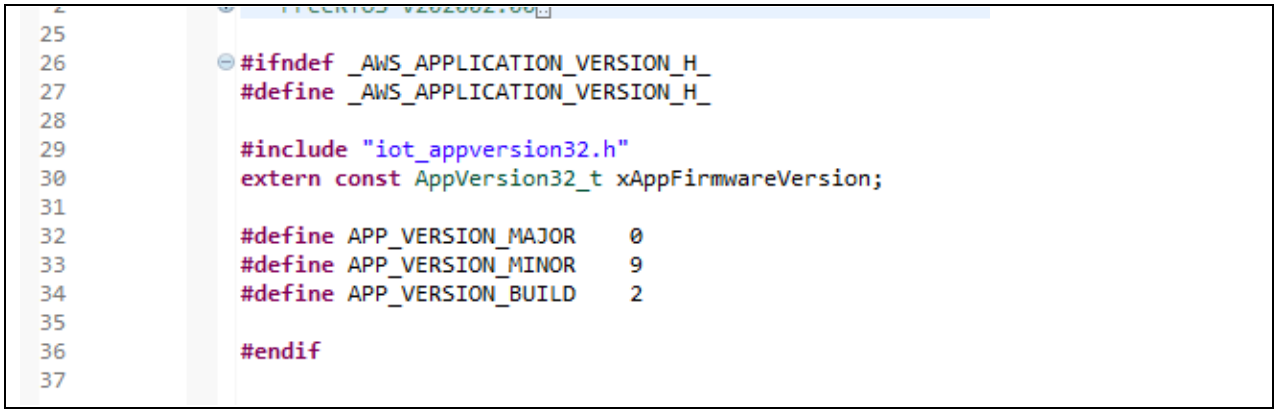


2.2　ファームウェアの初期バージョンのインストール

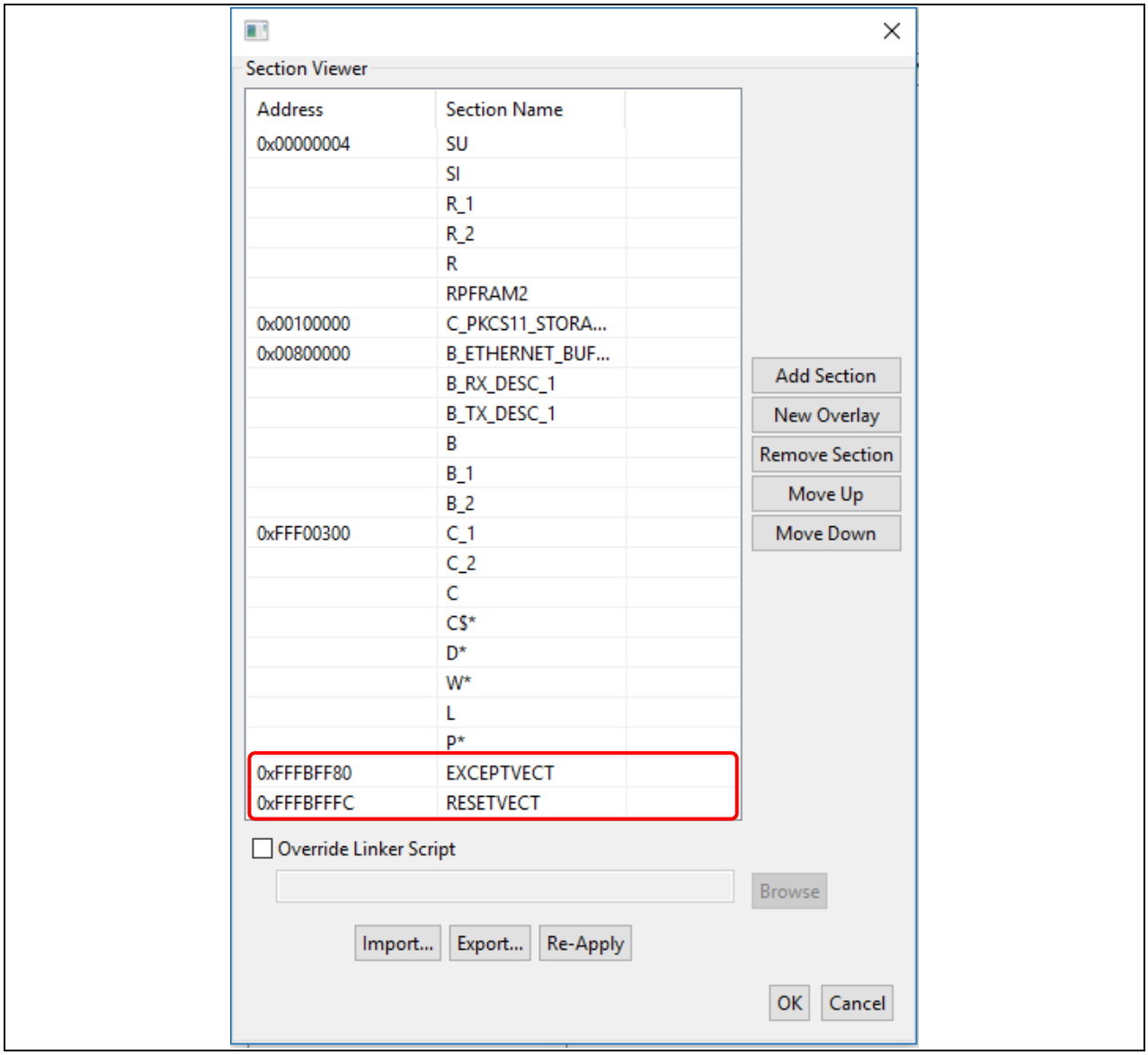
① amazon-freertos/vendors/renesas/boards/board/aws_demos/config_files/aws_demo_config.h を開き、#define CONFIG_MQTT_DEMO_ENABLED をコメントアウトして、CONFIG_OTA_UPDATE_DEMO_ENABLED を定義します。



② amazon-freertos/demos/include/aws_application_version.h を開き、ファームウェアの初期バージョンを 0.9.2 に設定します。



③ [プロジェクト]→[プロパティ]→C/C++ビルド→設定→[ツール設定]タブ→Linker→セクション→[…]で Section Viewer を開き、セクションを以下のように設定します。



④ aws_demos.mot ファイルをビルド して作成します。

⑤ Renesas Secure Flash Programmer から userprog.mot を作成します。

userprog.mot は、aws_demos.mot と boot_loader.mot を組み合わせたファイルです。このファイルを RX65N-RSK に読み込むことで、初期ファームウェアをインストールできます。

Renesas Secure Flash Programmer release 1.0.1 からダウンロードし Renesas Secure Flash Programmer.exe を実行します（一緒に置かれているファイルも必要ですのでダウンロードしてください）。

— [Initial Firm]タブを選択して、下図のようにパラメータを設定します。

・Private Key Path：secp256r1.privatekey へのパス

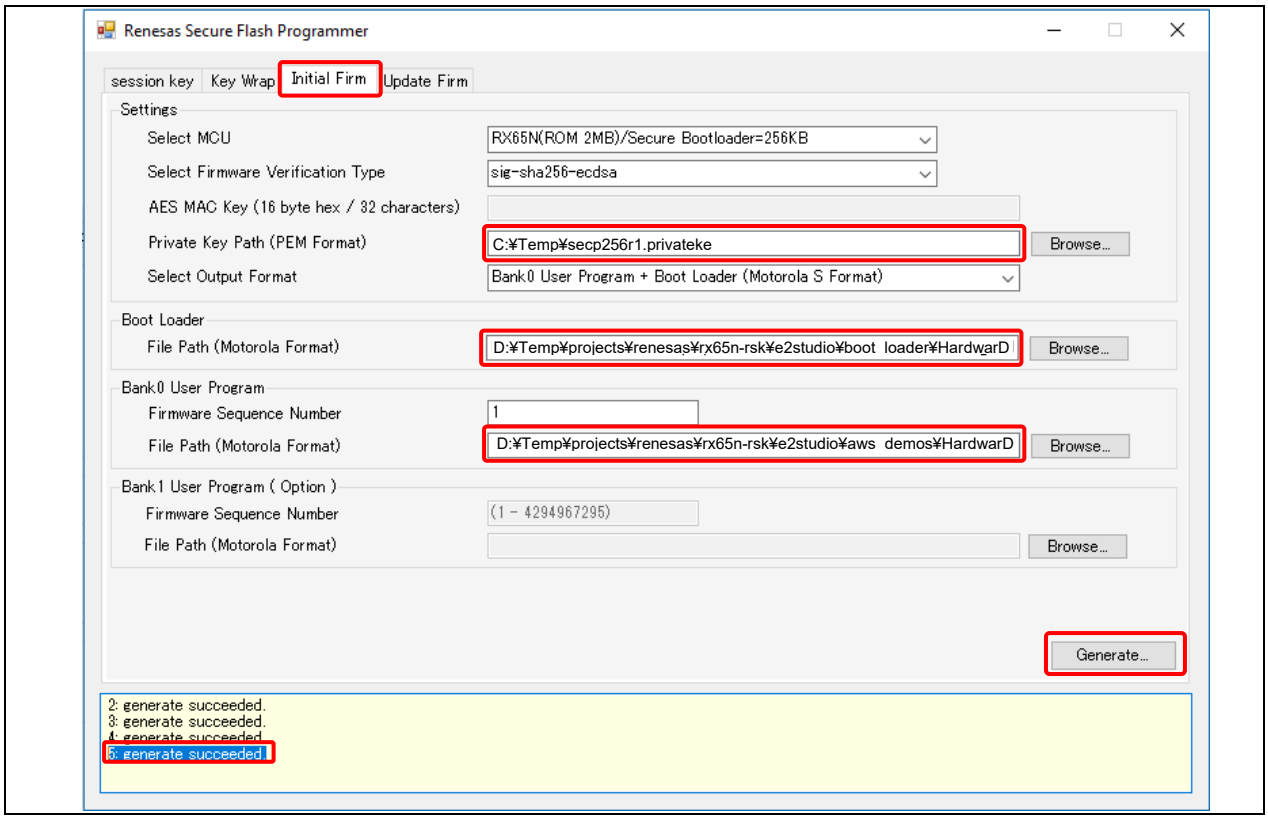
・Boot Loader File Path：boot_loader.mot へのパス

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

・Bank 0 User Program File Path：aws_demos.mot へのパス

（¥projects¥renesas¥rx65n-rsk¥e2studio¥aws_demos¥HardwareDebug）

— [Generate]をクリックすると、userprog.mot が生成され init_firmware フォルダに保存されます。generate succeeded と表示されたことを確認してください。

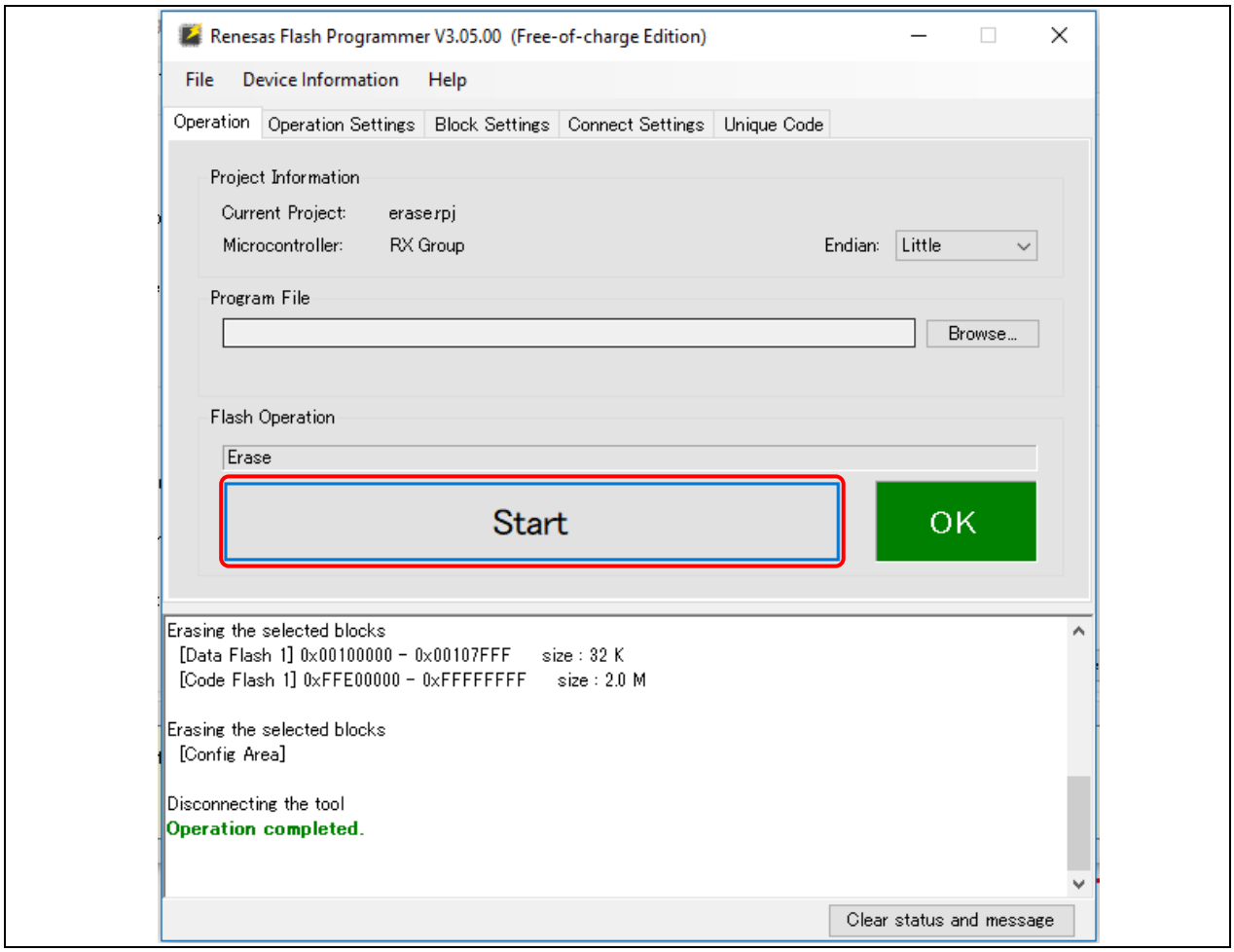


⑥ RX65N-RSK のフラッシュ ROM を消去します。

— 以下から最新の Renesas Flash Programmer をダウンロードしてください。
https://www.renesas.com/rfp

— Renesas Flash Programmer で以下のプロジェクトを開いてください。
¥vendors¥renesas¥rx_mcu_boards¥boards¥rx65n-rsk¥aws_demos¥flash_project¥erase_from_bank¥erase.rpj

— [Operation]タブを選択し、[Start]をクリックしてフラッシュ ROM を消去します。

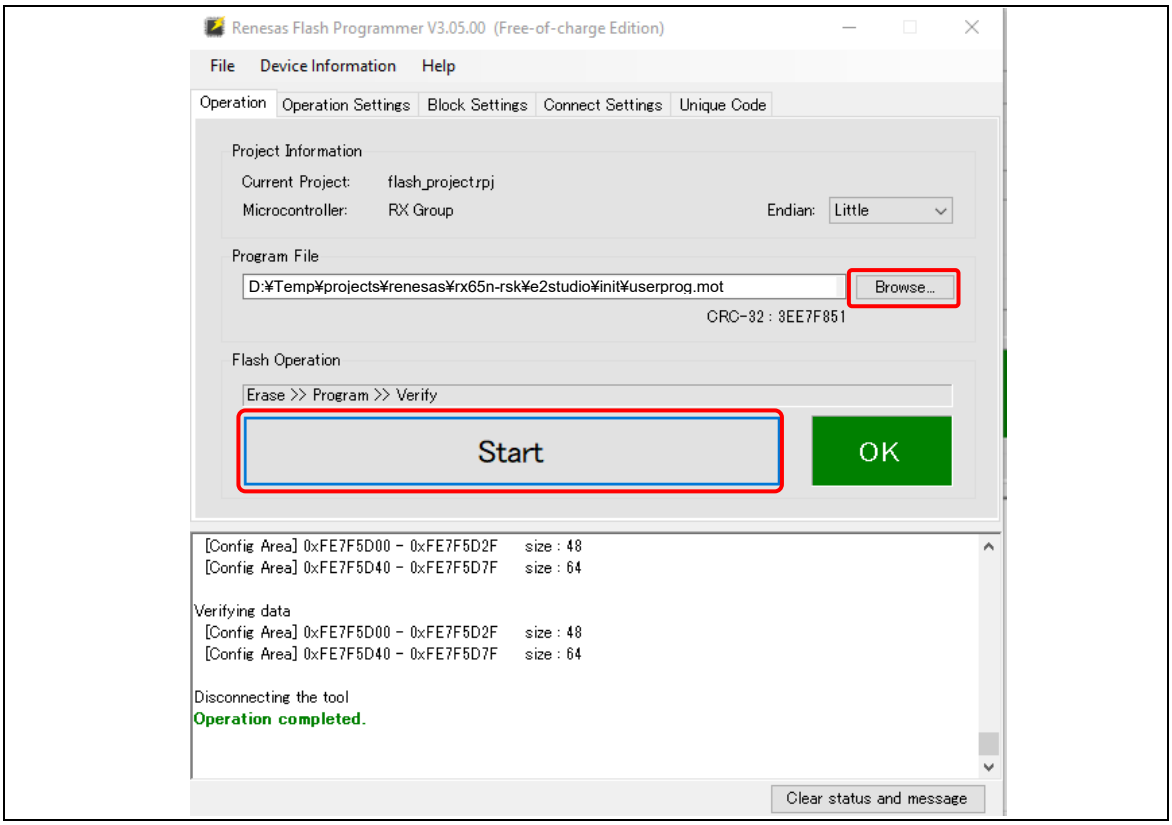


⑦ RX65N-RSK に初期ファームウェアを書き込みます。

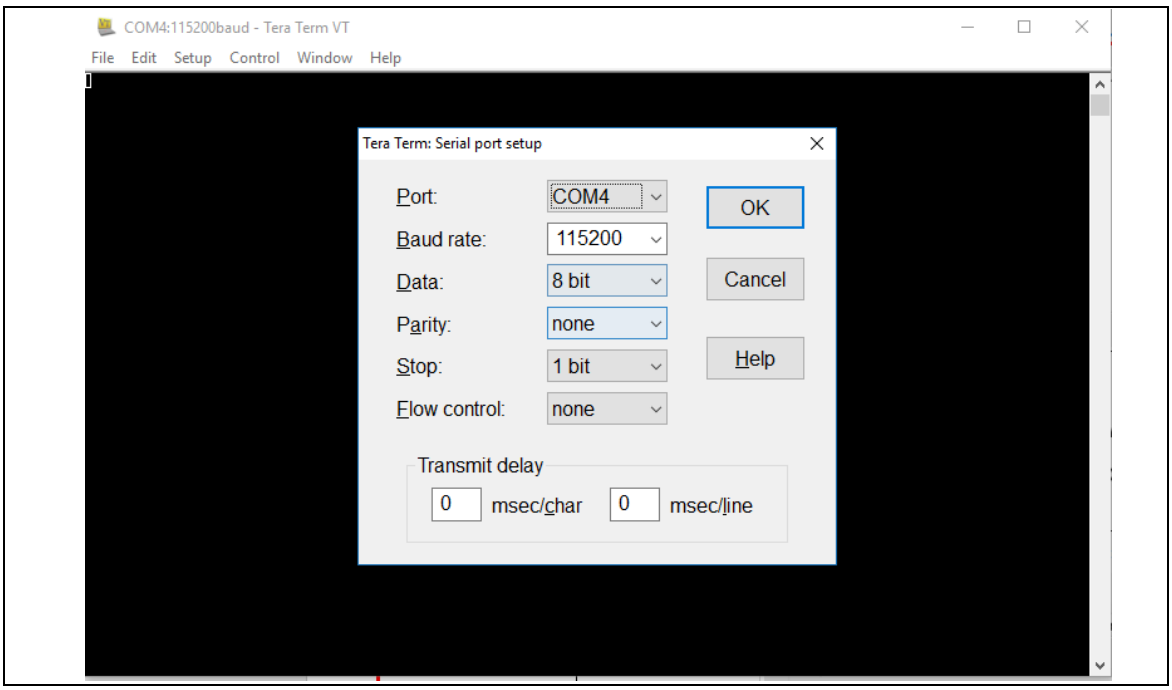
— Renesas Flash Programmer で新しいプロジェクトを作成してください。(例：flash_project.rpj)

— [Operation]タブを選択し、Program File に init_firmware フォルダに保存されている userprog.mot を設定してください。

— [Start]をクリックします。



⑧ Tera Term を開きます。
Tera Term の設定は以下になります。Tera Term をインストールしていない場合は、次からダウンロードしてください。https://ttssh2.osdn.jp/index.html.en



RX65N-RSK にはバージョン 0.9.2（初期バージョン）がインストールされています。これで、RX65N-RSK ボードは OTA 更新を受信できるようになりました。

```
------------------------------------------------
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-gr-rose (length 13).
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
```

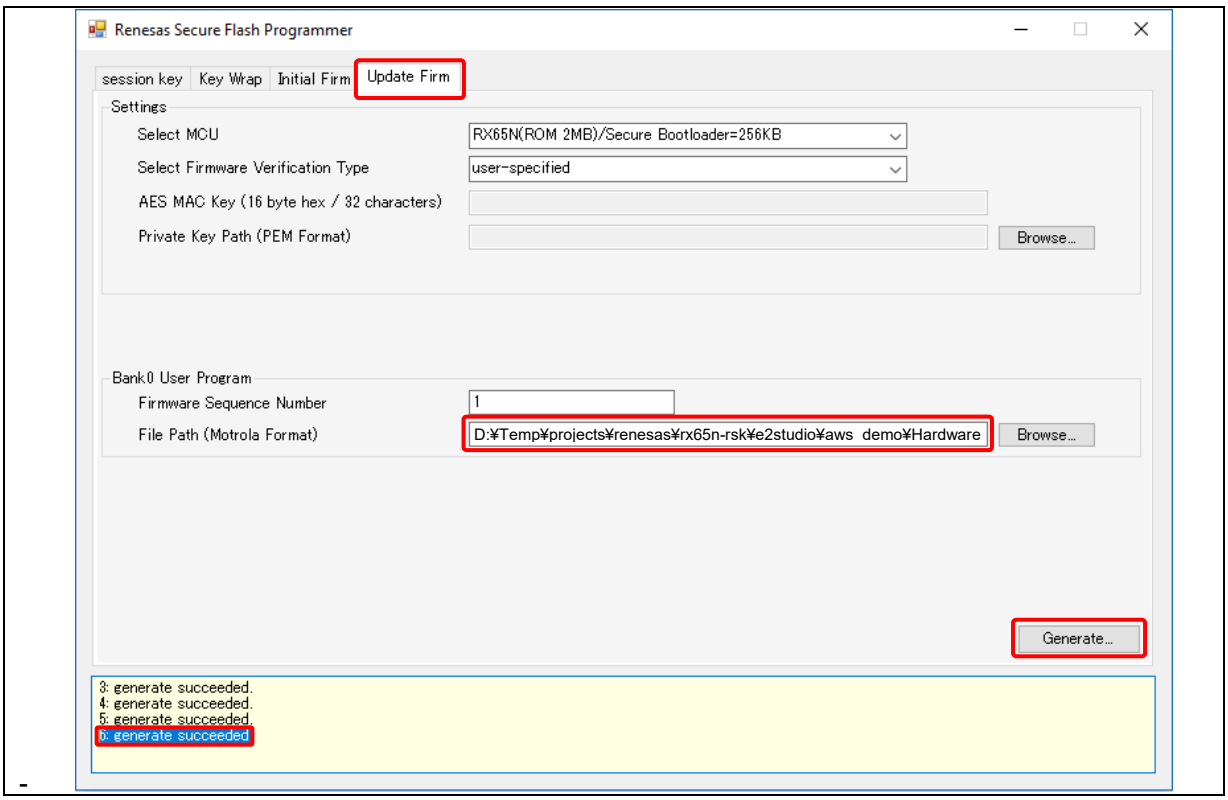
2.3　ファームウェアのバージョン更新

① demos/include/aws_application_version.h を開き、APP_VERSION_BUILD トークン値を 0.9.3 に増やします。

② プロジェクトを再びビルドします。

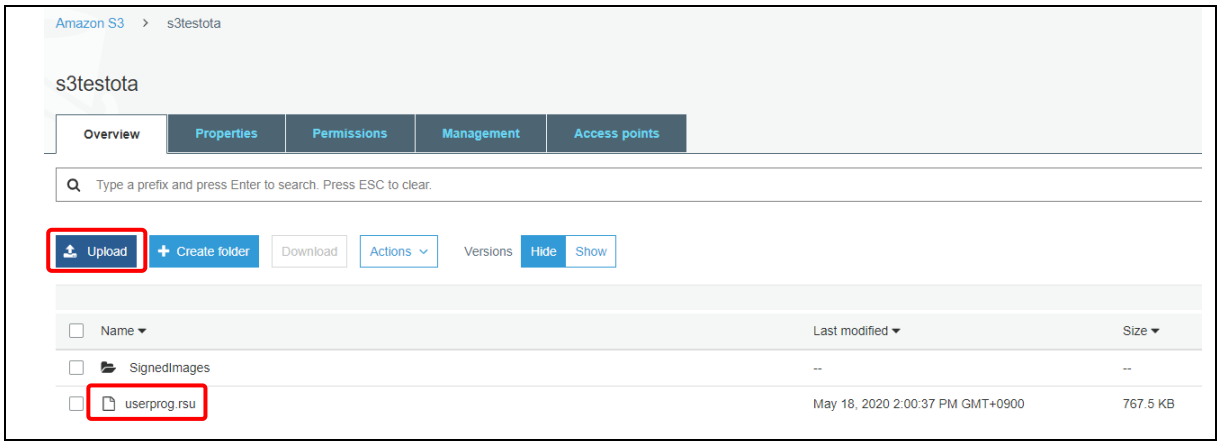
③ Renesas Secure Flash Programmer でファームウェアのバージョン更新用の userprog.rsu を作成します。

— Amazon-Freertos-Tools¥Renesas Secure Flash Programmer.exe を実行します。
— [Update Firm]タブを選択して、下図のようにパラメータを設定します。
・Bank 0 User Program File Path：aws_demos.mot へのパス
（¥projects¥renesas¥rx65n-rsk¥e2studio¥aws_demos¥HardwareDebug）
— [Generate]をクリックすると、userprog.rsu を生成され update_firmware フォルダに保存します。
generate succeeded と表示されたことを確認してください。



④ 1.2 Amazon S3 バケットの作成の説明に従って、ファームウェアの更新を Amazon S3 バケットにアップロードし、更新を保存します。

userprog.rsu を Amazon S3 バゲットにアップロードします。



⑤ RX65N-RSK のファームウェアを更新するためのジョブを作成します。

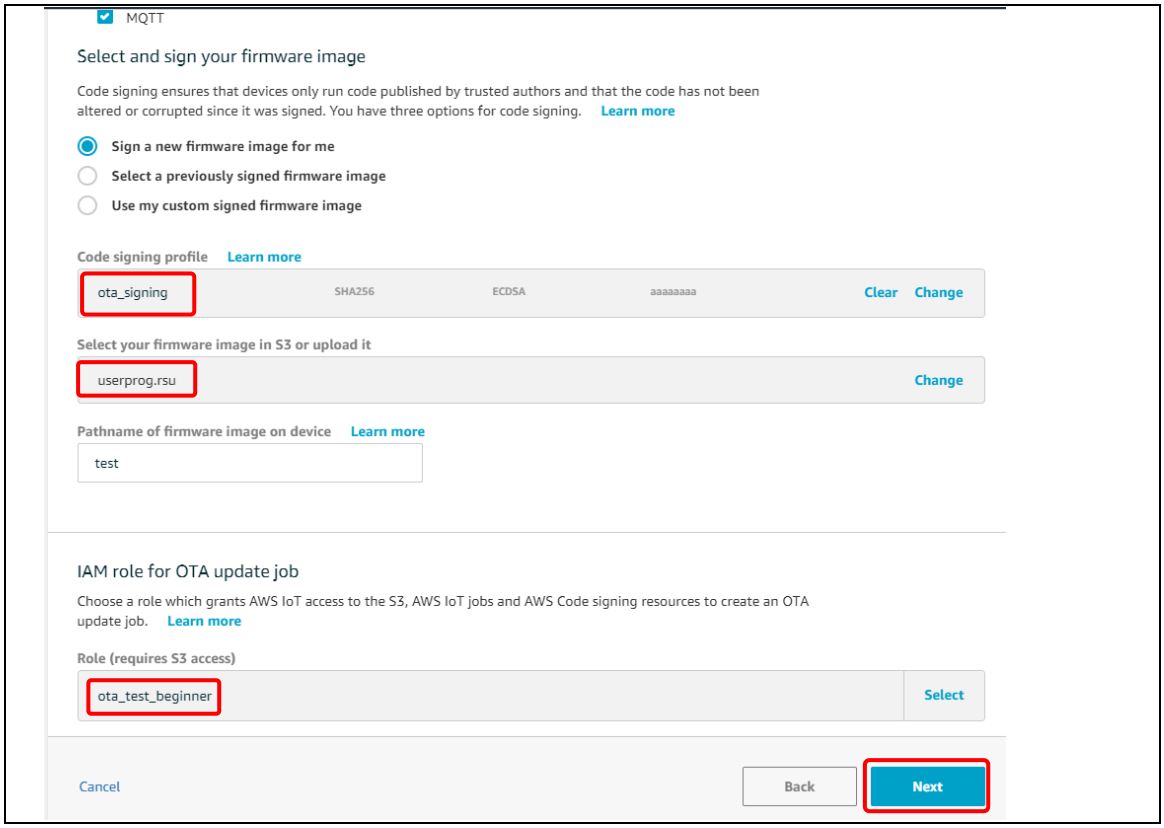
AWS IoT Jobs は、1 台または複数台の接続済みデバイスに待機中の「ジョブ」があることを通知します。ジョブを使用することで、大量のデバイスを管理したり、デバイス上のファームウェアやセキュリティ証明書を更新したり、デバイスの再起動や診断などの管理タスクを実行したりできます。

— [AWS IoT] → [管理] → [ジョブ] → [作成] → [OTA 更新ジョブの作成] → モノの名前を選択 → [次へ]を選択します。

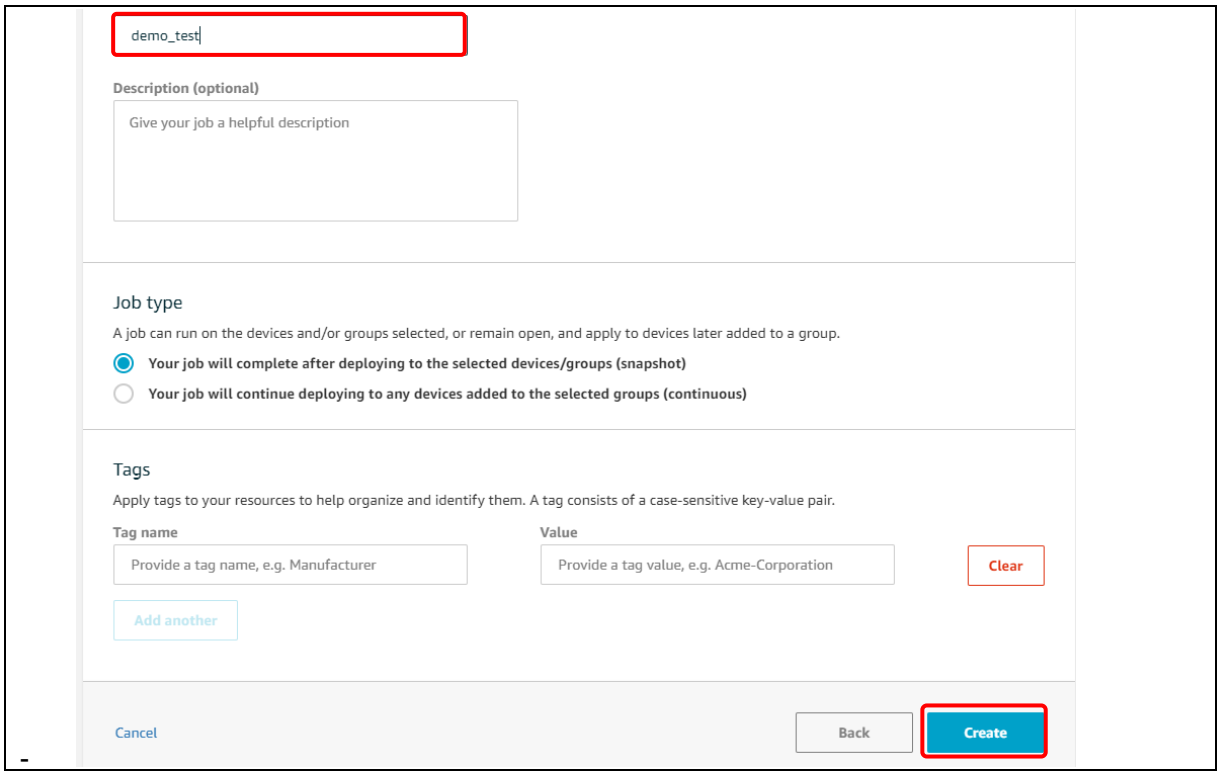
— FreeRTOS OTA 更新ジョブを次のように作成します。

前の章で作成したコード署名プロファイルを選択します。
S3 からファームウェアイメージを選択します。
前の章で作成した IAM ロールを選択します。

— [Next]をクリックします。

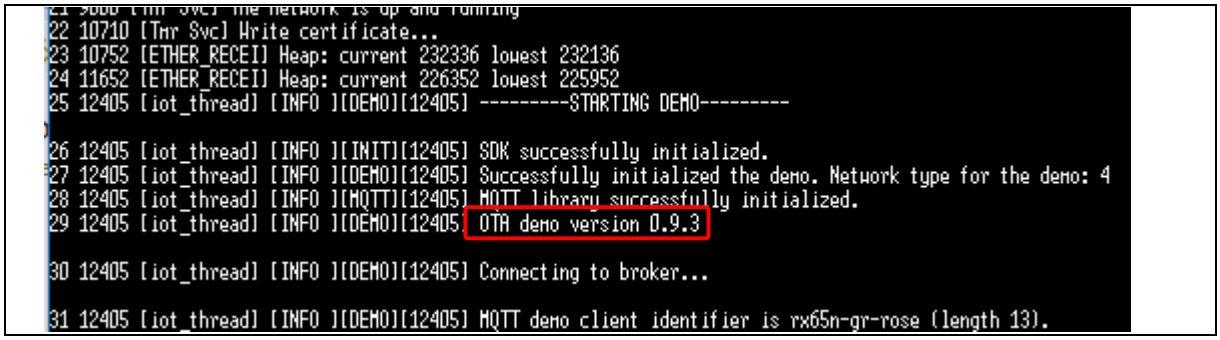


⑥ ID を指定して、[作成]をクリックします。

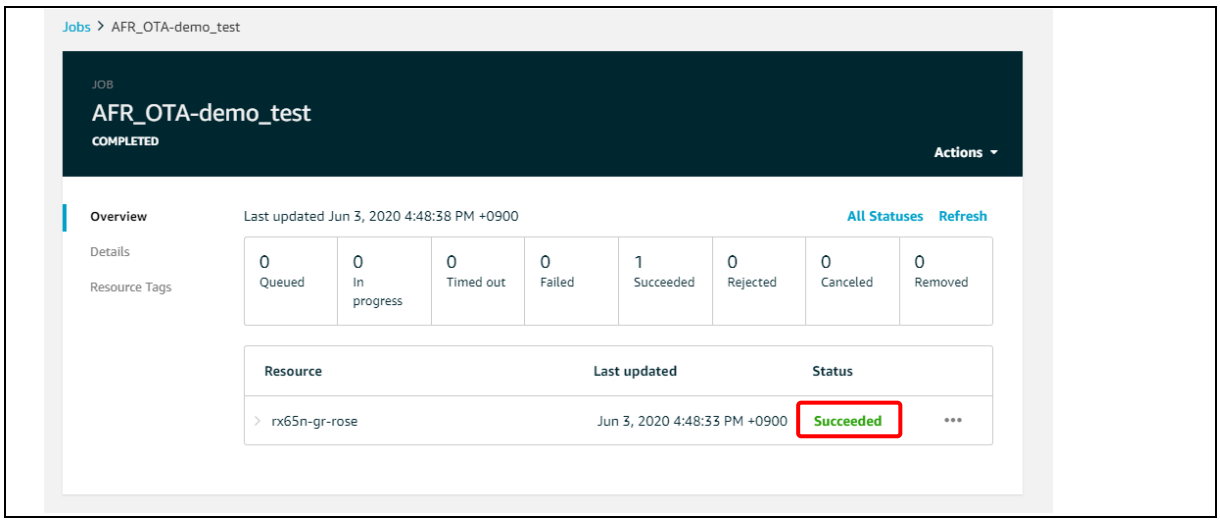


⑦ Tera Term を再び開き、ファームウェアが更新されたことを確認します。

OTA デモバージョンは 0.9.3 であり、更新が正常に行われています。



⑧ ジョブのステータスが「Succeeded」であることを確認します。



3. 制限事項

本アプリケーションノートの制限事項を以下に記載します。

- Big エンディアンの FreeRTOS OTA プログラムは正常に動作しません。
  Little エンディアンでプログラムをビルドし動作させてください。

4. 付録

4.1 動作確認環境

本アプリケーションノートの動作確認環境を以下に示します。

図 4.1 動作確認環境 (R01AN5549xx0102)

統合開発環境	e2 studio 7.8.0 e2 studio 2020-10
C コンパイラ	CC-RX Compiler v3.02.00 GCC 8.3.0.202004
使用ボード	RSKRX65N-2MB (型名：RTK50565Nxxxxxxxxx) RX65N Cloud Kit (型名：RTK5RX65Nxxxxxxxxx)
デバッガ	E2 エミュレータ E2 エミュレータ Lite
ソフトウェア	Amazon FreeRTOS パッケージ 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
エンディアン	Little エンディアン

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Aug. 31, 2020	ー	初版発行
1.01	Oct. 30. 2020	ー	節を追加
1.02	May. 28.2021	ー	GCC に新規対応。
		3	1.1 章にサインインの詳細な手順を追加。
		8-9	1.2 章の「Amazon S3 バケットの作成」手順を確認する画像を変更。
		12	1.5 章の secp256r1.privatekey に関する誤字を修正。
		26	2.2 章に画像を追加。
		37	制限事項の章を追加。 Big エンディアンに関する制限事項を追加。
		38	動作確認環境の章を追加。 R01AN5549xx0102 に対する下記の動作確認環境を追加。 - CC-RX を v3.02.00 に更新。 - GCC を 8.3.0.202004 に更新。 - RX65N Cloud Kit を追加。 - Amazon FreeRTOS パッケージを v202002.00-rx-1.0.5 に更新。 - Renesas Secure Flash Programmer.exe を v1.0.1 に更新。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。