RL78/G23

LTE 通信での Amazon Web Services 接続 Getting Started Guide: RL78/G23-128p Fast Prototyping Board + FreeRTOS

はじめに

本ドキュメントは、ルネサス製マイコンボードとセルラーIoT モジュールを組み合わせて Amazon Web Services （以降 AWS）に接続する方法を示します。

関連文書

RL78/G23 ユーザーズマニュアル ハードウェア編 (R01UH0896)

RL78/G22, RL78/G23, RL78/G24 ファームウェア アップデート モジュール (R01AN6374)

RL78/G23-128p Fast Prototyping Board ユーザーズマニュアル (R20UT4870)

Renesas Flash Driver RL78 Type01 ユーザーズマニュアル (R20UT4830)

注意事項：当社での RYZ024A モジュールの取り扱いについて

当社は、RYZ024A 型名の既存 LTE モジュールの製造を中止し、製品の出荷を終了することを発表しました。

現在の設計または生産中に本製品を使用している場合、Sequans 社の製品型名 GM02S が、RYZ024A とピン及び機能コンパティビリティ代替品となります。本ドライバは、以下の組み合わせにて継続してご利用可能です。

- RYZ024A Cellular モジュール制御モジュール： Sequans 社 GM02S が互換のある代替モジュールとなります。

なお、RYZ024A の EOL 通知は製品ページ内の本リンクをご参照ください。

目次

注：
- AWS™は Amazon.com, Inc. or its affiliates の商標です。(https://aws.amazon.com/trademark-guidelines/)
- FreeRTOS™は Amazon Web Services, Inc.の商標です。(https://freertos.org/copyright.html)
- GitHub® は GitHub,Inc. のトレードマークです。(https://github.com/logos)

1.　概要

サンプルプログラム「iot-reference-rl78」は RL78 ファミリ、AWS、FreeRTOS を使用した IoT ソリューションのリファレンスを提供します。このサンプルプログラムはルネサスエレクトロニクスが提供する他の様々な製品と連携しており、AWS IoT のデモを簡単に試すことができます。

1.1　デモプロジェクトの概要

サンプルプログラムは下記のデモプロジェクトを含みます。これらのデモプロジェクトは、ルネサス製マイコンボード RL78/G23-128p Fast Prototyping Board とセルラーIoT モジュールを用いて AWS クラウドに接続する動作を実現します。

表 1-1 デモプロジェクトの一覧

デモプロジェクトの名称	説明
デモプロジェクト(PubSub)	MQTT 通信によるシンプルなデータアップロードを行う。
デモプロジェクト(OTA)	OTA によるファームウェアアップデートを行う。

各デモプロジェクトの概要は下記の章を参照してください。

- 2　ハードウェア説明
- 3　ソフトウェア説明

各デモプロジェクトの実行方法は下記の章を参照してください。

- デモプロジェクト(PubSub)
  - 4　デモプロジェクト(PubSub/OTA)共通のセットアップ
  - 5　デモプロジェクト(PubSub)固有のセットアップ
- デモプロジェクト(OTA)
  - 4　デモプロジェクト(PubSub/OTA)共通のセットアップ
  - 6　デモプロジェクト(OTA)固有のセットアップ

1.2　動作確認条件

デモプロジェクトは下記の条件で動作を確認しています。

表 1-2　動作確認条件 (RL78/G23)

項目	内容
使用マイコン	RL78/G23 (R7F100GSN CF 768KB)
使用ボード	RL78/G23-128p Fast Prototyping Board (RTK7RLG230CSN000BJ)
動作周波数	高速オンチップ・オシレータ・クロック: 32MHz
動作電圧	3.3V
統合開発環境	ルネサスエレクトロニクス製 e^2 studio 2024-01.1
C コンパイラ	ルネサスエレクトロニクス製 CC-RL V1.12.01
ファームウェア書き込みツール	Renesas Flash Programmer V3.14.00
スマート・コンフィグレータ(SC)	Renesas Smart Configurator for RL78 24.1.0.v20231218-0132
ボードサポートパッケージ(BSP)	v1.60 (r_bsp)
フラッシュライブラリ	RL78/G2x 用 Renesas Flash Driver (RFD) RL78 Type01 V1.20 注：コードフラッシュライブラリ -> RL78/G2x 用 Renesas Flash Driver RL78 Type01 パッケージ V1.20
ファームウェアアップデートモジュール(FWUP)	RL78/G22,RL78/G23,RL78/G24 ファームウェア アップデート モジュール v2.01
ファームウェアイメージ生成ユーティリティツール	Renesas Image Generator V3.03 注：ファームウェアアップデートモジュール(FWUP)に同梱
Python	Python 3.10.1
OpenSSL	OpenSSL 3.1.4

表 1-3　動作確認条件 (その他、OSS ライブラリ等)

項目	内容
iot-reference-rl78	v202210.01-LTS-rl78-1.0.0 (Based FreeRTOS 202210.01-LTS) https://github.com/renesas/iot-reference-rl78/tree/v202210.01-LTS-rl78-1.0.0
FreeRTOS Cellular Interface	1.3.0 https://github.com/FreeRTOS/FreeRTOS-Cellular-Interface
FreeRTOS Kernel	10.5.1 https://github.com/FreeRTOS/FreeRTOS-Kernel
backoffAlgorithm	1.3.0 https://github.com/FreeRTOS/backoffAlgorithm
coreJSON	3.2.0 https://github.com/FreeRTOS/coreJSON
coreMQTT Client	2.1.1 https://github.com/FreeRTOS/coreMQTT
coreMQTT Agent	1.2.0 https://github.com/FreeRTOS/coreMQTT-Agent
AWS IoT Over-the-air Update	3.4.0 https://github.com/aws/ota-for-aws-iot-embedded-sdk
tinycbor	0.5.2 https://github.com/intel/tinycbor
FreeRTOS-Plus network_transport	バージョンなし https://www.freertos.org/network-interface.html
Logging Interface	1.1.3 https://github.com/aws/amazon-freertos/tree/main/libraries/logging
TinyCrypt Cryptographic Library	0.2.8 https://github.com/intel/tinycrypt

1.3　機材一覧

デモプロジェクトで必要な機材の一覧を示します。

表 1-4　機材一覧

項目	内容
マイコンボード	RL78/G23-128p Fast Prototyping Board RTK7RLG230CSN000BJ - RL78/G23-128p Fast Prototyping Board
セルラーIoT モジュール	PMOD Expansion Board for RYZ024A (以降、RYZ024A と表します) RTKYZ024A0B00000BE - PMOD Expansion Board for RYZ024A
SIM カード	LTE 通信が可能であること 例：RTKYZ024A0B00000BE 付属 SIM カード(注) - Truphone 製 DHA-SIM-132 - Nippon SIM 製
USB-UART 変換ボード	Pmod USBUART https://reference.digilentinc.com/reference/pmod/pmodusbuart/start
Micro USB Type-B ケーブル x3	・USB-UART 変換ボードと PC を接続 ・マイコンボードと PC を接続 ・RYZ024A の電源供給用
ジャンパワイヤ x3	USB-UART 変換ボード と マイコンボードを接続
ジャンパピン x3	マイコンボードの電源選択用(J15,J16,J19)

注：

PMOD Expansion Board for RYZ024A (RTKYZ024A0B00000BE) に付属の Truphone SIM カードをご使用の場合は、SIM カードの Activation が必要です。
「RA6M5 Group RYZ024A PMOD LTE Connectivity with RA6M5 MCU Quick Start Guide」(R21QS0007) を参照し SIM カードの Activation を行ってください。

機材接続の全体図

各デモの機材接続の全体図は下記を参照してください。

- デモプロジェクト(PubSub)：図 4-1　本デモプロジェクトのハードウェア全体構成
- デモプロジェクト(OTA)：同上

デバッグ用機材に関する注意事項

本デモプロジェクトは COM port デバッグを使用しますが、エミュレータ用いてデバッグをすることも可能です。
エミュレータを使用する場合、エミュレータ接続用コネクタの実装と回路変更が必要です。詳細は「7.2.1 章　ジャンパピン設定、コネクタ実装、パターンカット」もしくは以下を参照してください。

表 1-5　デバッグ機材の一覧

項目	内容
エミュレータ	E2 エミュレータ Lite https://www.renesas.com/jp/ja/software-tool/e2-emulator-lite-rte0t0002lkce00000r

2.　ハードウェア説明

2.1　デモプロジェクト(PubSub)

2.1.1　システム構成

デモプロジェクト(PubSub)のシステム構成を示します。

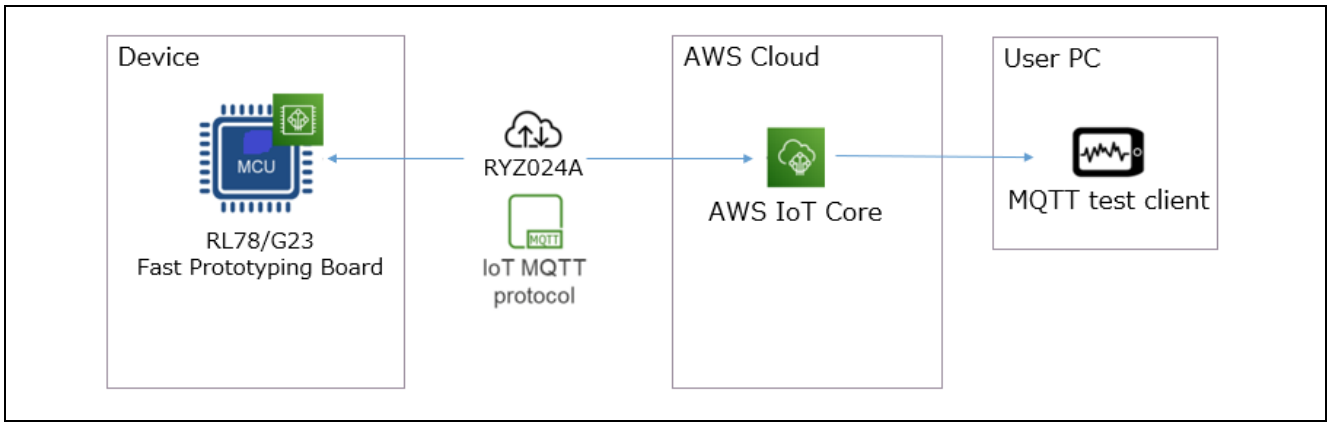


図 2-1　デモプロジェクト(PubSub)のシステム構成

2.1.2　使用端子一覧

デモプロジェクト(PubSub)で使用する端子と機能を示します。

表 2-1　デモプロジェクト(PubSub)の使用端子と機能

端子名	入出力	内容
P143/RxD3	入力	RYZ024A との UART 通信(受信)
P144/TxD3	出力	RYZ024A との UART 通信(送信)
P00	出力	RYZ024A へのリセット
P142	出力	RYZ024A との UART 通信(RTS)
P14/RxD2	入力	ターミナル入力
P13/TxD2	出力	ターミナル出力
P50	出力	LED1

2.2　デモプロジェクト(OTA)

2.2.1　システム構成

デモプロジェクト(OTA)のシステム構成を示します。

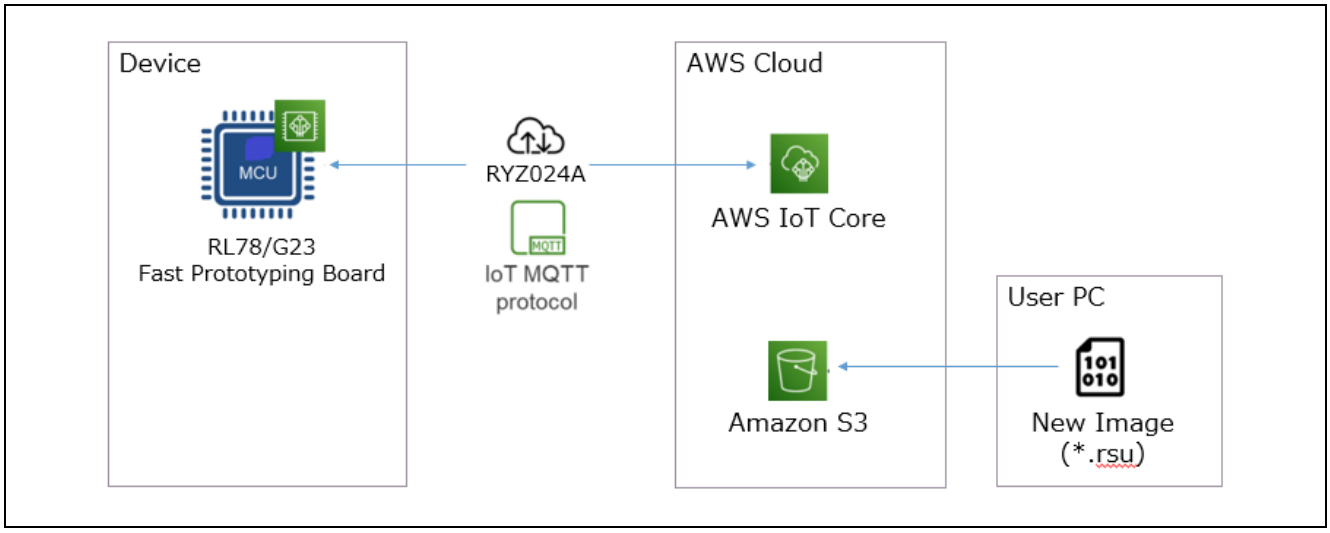


図 2-2　デモプロジェクト(OTA)のシステム構成

2.2.2　使用端子一覧

デモプロジェクト(OTA)で使用する端子と機能を示します。

表 2-2　デモプロジェクト(OTA)の使用端子と機能

端子名	入出力	内容
P143/RxD3	入力	RYZ024A との UART 通信(受信)
P144/TxD3	出力	RYZ024A との UART 通信(送信)
P00	出力	RYZ024A へのリセット
P142	出力	RYZ024A との UART 通信(RTS)
P14/RxD2	入力	ターミナル入力
P13/TxD2	出力	ターミナル出力
P50	出力	LED1

3.　ソフトウェア説明

3.1　デモプロジェクト(PubSub)

3.1.1　デモプロジェクト構成

本デモプロジェクトはマイコンボードから AWS に接続し MQTT ライブラリを使用して定期的にメッセージを発行します。

3.1.2　オプション・バイトの設定一覧

オプション・バイトの設定を示します。

表 3-1　オプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	11101111B	ウォッチドッグ・タイマ 動作停止 (リセット解除後、カウント停止)
000C1H/040C1H	00111010B	LVD0 オフ（ RESET 端子により外部リセットを使用）
000C2H/040C2H	11101000B	HS（高速メイン）モード＆ 高速オンチップ・オシレータ・クロック (fIH): 32MHz
000C3H/040C3H	10000100B	オンチップ・デバッグ許可

3.2　デモプロジェクト(OTA)

3.2.1　デモプロジェクト構成

本デモプロジェクトのファームウェア更新の仕組みは、ファームウェアアップデートモジュールが提供している方式のうち、「半面更新方式（バッファ面は内蔵フラッシュ）」を使用しています。詳細は、「RL78/G22,RL78/G23,RL78/G24 ファームウェア アップデート モジュール」を参照してください。

以下にファームウェア更新の仕組みの図とメモリマップを示します。

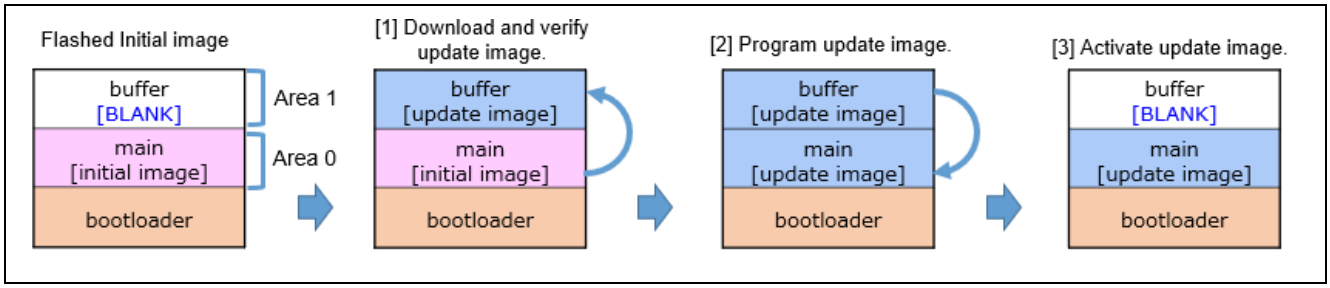


図 3-1　ファームウェア更新の仕組み

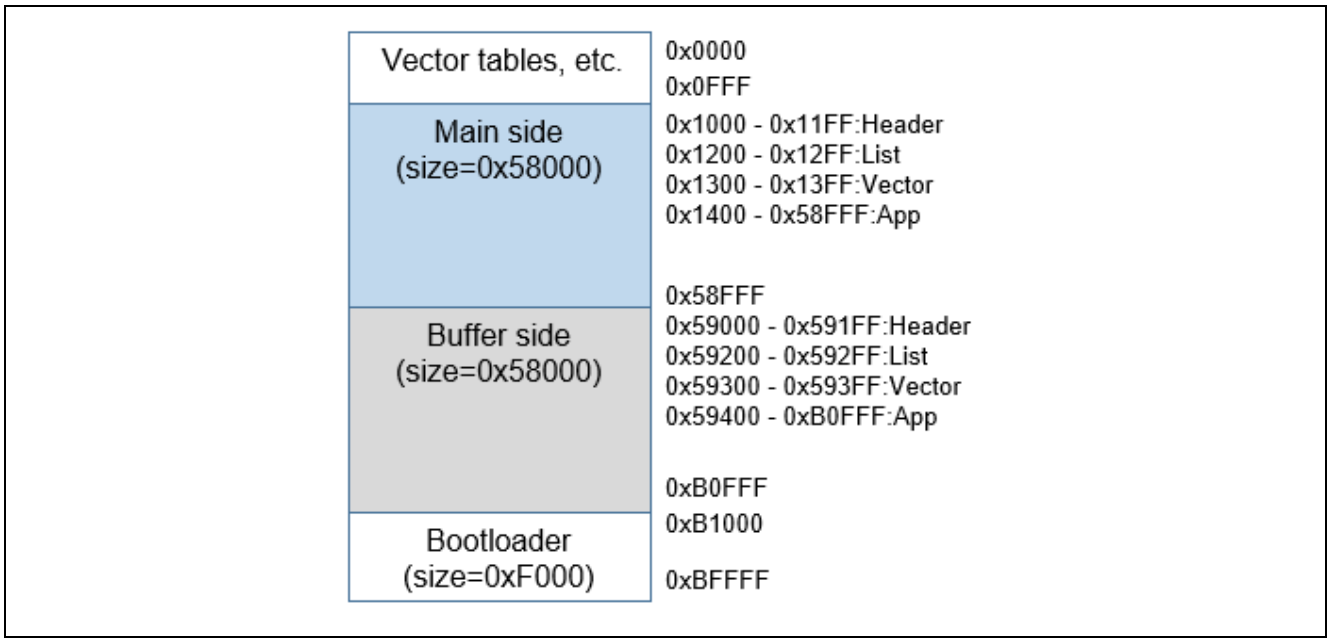


図 3-2　デモプロジェクト(OTA デモ)のメモリマップ

3.2.2　オプション・バイトの設定一覧

オプション・バイトの設定を示します。

表 3-2　オプション・バイト設定

アドレス	設定値	内容
000C0H/040C0H	11101111B	ウォッチドッグ・タイマ 動作停止 (リセット解除後、カウント停止)
000C1H/040C1H	00111010B	LVD0 オフ（ RESET 端子により外部リセットを使用）
000C2H/040C2H	11101000B	HS（高速メイン）モード＆ 高速オンチップ・オシレータ・クロック (fIH): 32MHz
000C3H/040C3H	10000100B	オンチップ・デバッグ許可

3.3　フォルダ構成

サンプルプログラムのフォルダ構成を以下に示します。

表 3-3　サンプルプログラムのフォルダ構成

Folder Name	Description
iot-reference-rl78	The sample program described in this Getting Started Guide.
├──Common	
│　├──FreeRTOS_common	
│　└──ports	
│　　└──ota_pal	
├──Configuration	
│　└──rl78g23-fpb	
│　　├──ota	OTA demo configurations.
│　　├──pubsub	PubSub demo configurations.
│　　└──test	
├──Demos	
│　├──common	
│　├──include	
│　├──mqtt_agent	
│　├──OtaOverMqtt	OTA demo source codes.
│　└──SimplePubSub	PubSub demo source codes.
├──IDT_config	
├──Middleware	
│　├──3rdparty	
│　├──Application-Protocols	
│　│　└──network_transport	
│　├──AWS	
│　│　└──ota-for-aws-iot-embedded-sdk	
│　├──FreeRTOS	FreeRTOS Kernel and libraries.
│　│　├──backoffAlgorithm	
│　│　├──coreJSON	
│　│　├──coreMQTT	
│　│　├──coreMQTT-Agent	
│　│　├──FreeRTOS-Cellular-Interface	
│　│　└──FreeRTOS-Kernel	
│　└──logging	
├──Projects	
│　└──rl78g23-fpb	
│　　├──application_code	
│　　├──flash_proj	
│　　├──helper	
│　　├──modules	
│　　├──projects	Import below folders to IDE.
│　　│　├──aws_ryz024a_rl78g23-fpb	PubSub demo and OTA demo. Select by Build Configurations.
│　　│　├──boot_loader	Boot loader for OTA demo.
│　　│　└──test_aws_cellular_ryz024a	
│　　└──rtos_skelton	
├──Test	
└──Tools	

3.4　コードサイズ

以下の条件におけるデモプロジェクトの ROM、RAM サイズを下表に示します。

- CC-RL
  - コンパイル・オプション
    - -Odefault：オブジェクト・サイズと実行速度の両方に効果のある最適化
  - リンク・オプション
    - -optimize=symbol_delete：一度も参照のない変数／関数を削除

表 3-4　デモプロジェクトの ROM、RAM サイズ

デモプロジェクト名	ROM (byte)	RAM (byte)
aws_ryz024a_rl78g23-fpb (デモプロジェクト(PubSub))	142311	29913
aws_ryz024a_rl78g23-fpb (デモプロジェクト(OTA))	234729	36790
boot_loader	22147	1348

4.　デモプロジェクト(PubSub/OTA)共通のセットアップ

デモプロジェクト(PubSub)とデモプロジェクト(OTA)で共通のセットアップについて説明します。

4.1　ハードウェアのセットアップ

4.1.1　全体の構成

最初に、本デモプロジェクトを構成するハードウェア全体の構成を示します。

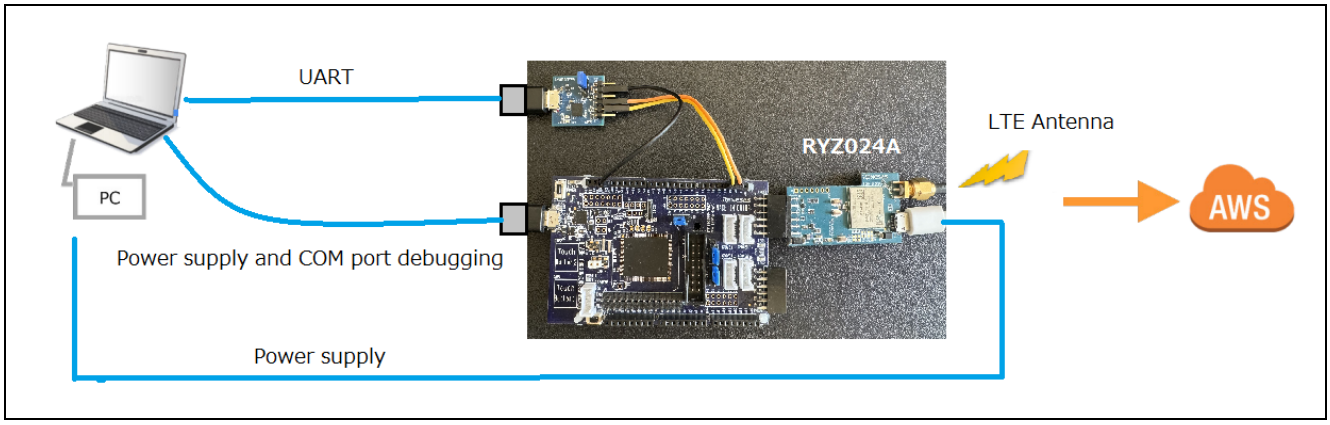


図 4-1　本デモプロジェクトのハードウェア全体構成

4.1.2　ハードウェアの接続方法

ハードウェアの接続方法を以下に示します。

①　アクティベート済の SIM カードを RYZ024A に挿入します。

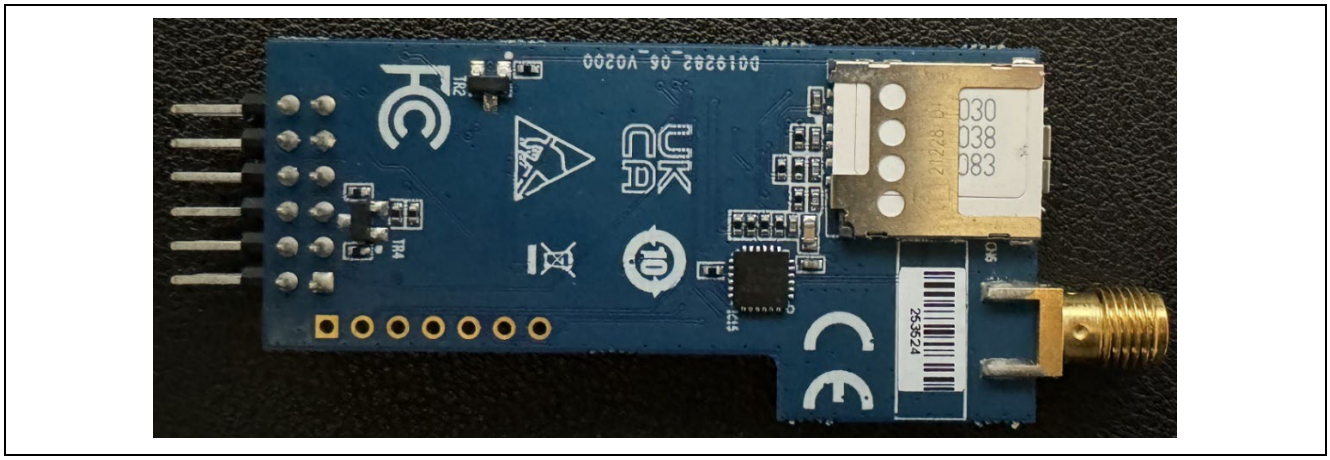


図 4-2　アクティベート済の SIM カードを RYZ024A に挿入

②　アンテナと電源供給用の USB ケーブルを RYZ024A に接続します。

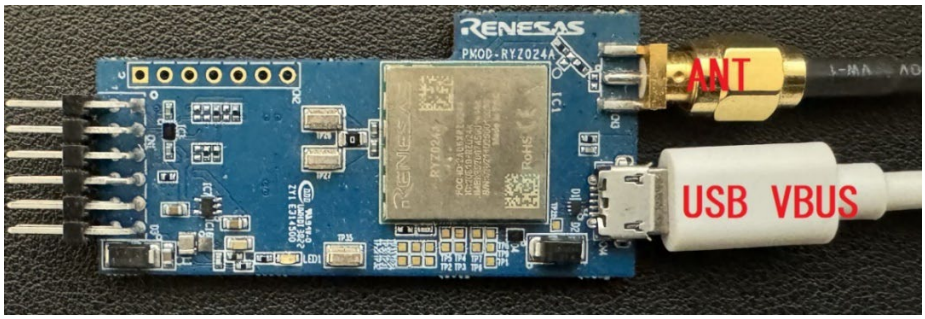


図 4-3　アンテナと電源供給用の USB ケーブルを RYZ024A に接続

③　RYZ024A をマイコンボードの PMOD1 に接続します。

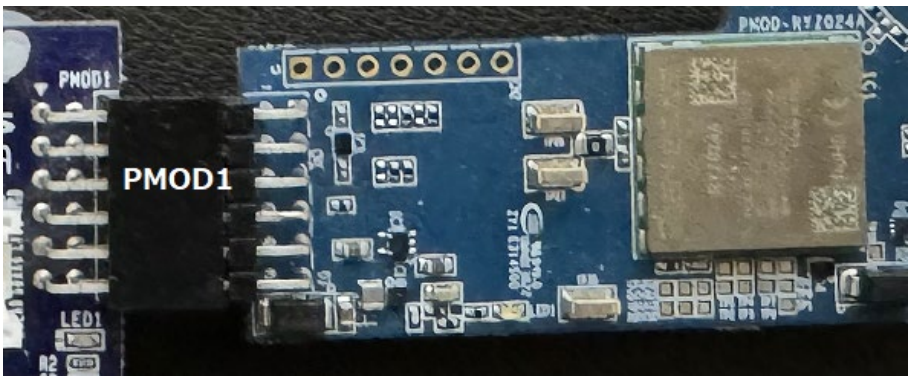


図 4-4　RYZ024A をマイコンボードの PMOD1 に接続

④　USB-UART 変換ボードをマイコンボードに接続します。

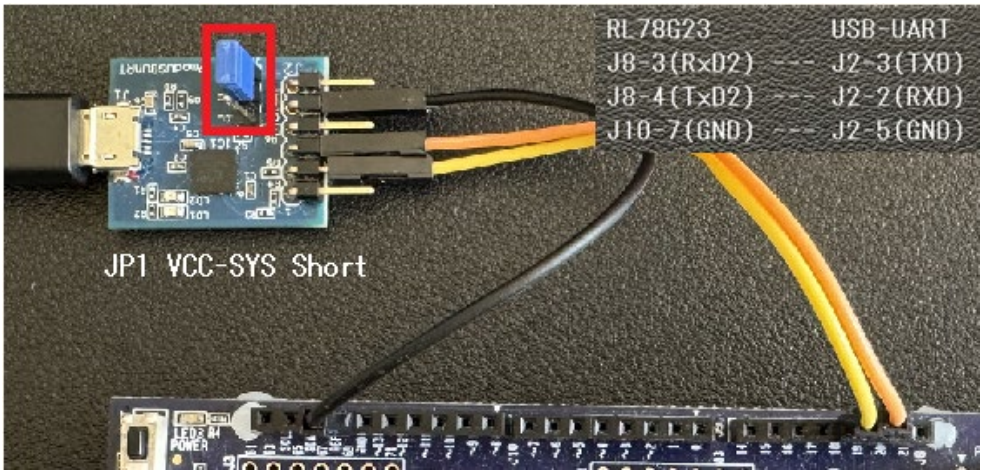


図 4-5　マイコンボードに USB-UART 変換ボードを接続

⑤　マイコンボードの電源選択ヘッダを J20 2-3 ショートに設定し 3.3V 電源を選択します。

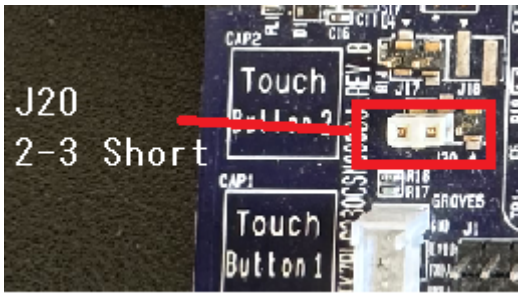


図 4-6　マイコンボード電源を 3.3V に設定

⑥　マイコンボードにエミュレータ用コネクタを実装するための回路変更を実施している場合、USB シリアル変換器を使用した COM port デバッグの設定にします。
回路変更を実施していない場合、この操作は不要です。

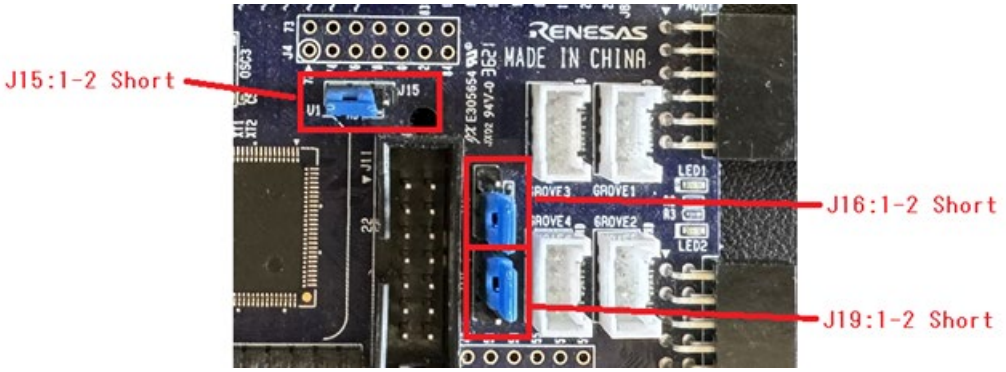


図 4-7　**COM port** デバッグ使用時設定(部品面)

⑦　USB ケーブルを接続してマイコンボードへ電源を供給します。

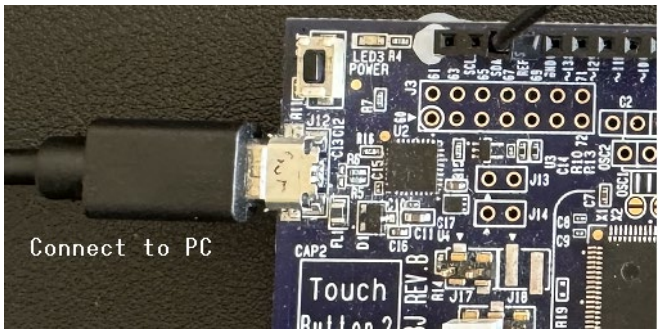


図 4-8　マイコンボードに電源を供給

⑧　COM port 番号を確認します。

ファームウェア書き込みおよびデバッグで COM port 番号を使用します。

⑨　USB ケーブルを抜いてマイコンボードへの電源供給を停止します。

4.2　ソフトウェアのセットアップ

4.2.1　ターミナルソフトの設定

デモプロジェクトのログ出力時にターミナルソフト(例：Tera Term 等)が必要です。以下にシリアルポートの設定を示します。

表 4-1 シリアルポート設定

項目	内容
ボーレート	115200 bps
データ	8 bit
パリティ	なし
ストップビット	1 bit
フロー制御	なし

4.2.2　フラッシュライタのインストール

初期イメージの書き込みに使用します。

Renesas Flash Programmer (Programming GUI)

4.2.3　デモプロジェクトに SIM カード情報を追加

デモプロジェクトの下記のマクロに SIM カード情報を設定します。SIM カード情報については使用する SIM カードのマニュアルを参照してください。

- iot-reference-rl78\Projects\rl78g23-fpb\modules\r_config\r_aws_cellular_config.h
  - AWS_CELLULAR_CFG_AP_NAME：アクセスポイント名
  - AWS_CELLULAR_CFG_AP_USERID：アクセスポイントのユーザ ID (注 1)
  - AWS_CELLULAR_CFG_AP_PASSWORD：アクセスポイントのパスワード (注 1)
  - AWS_CELLULAR_CFG_PIN_CODE：PIN コード (注 2)
  - AWS_CELLULAR_CFG_AUTH_TYPE：認証方式

注 1: SIM カード情報が何もない場合、マクロには空の文字列を指定してください。
注 2: SIM カード情報が何もない場合、マクロには空の値を指定してください。

この文書で紹介している SIM カードの設定方法を以下で示します。

① RTKYZ024A0B00000BE 付属 SIM カード(注) - Truphone 製の場合

iot-reference-rl78\Projects\rl78g23-fpb\modules\r_config\r_aws_cellular_config.h

```
#define AWS_CELLULAR_CFG_AP_NAME        "iot.truphone.com" /* Access point name */
#define AWS_CELLULAR_CFG_AP_USERID      ""     /* Login ID */
#define AWS_CELLULAR_CFG_AP_PASSWORD    ""     /* Access point password */
#define AWS_CELLULAR_CFG_PIN_CODE              /* SIM card PIN code */
#define AWS_CELLULAR_CFG_AUTH_TYPE      (0)    /* Authentication protocol type
(0=None,1=PAP,2=CHAP)*/
```

② DHA-SIM-132 - Nippon SIM 製の場合

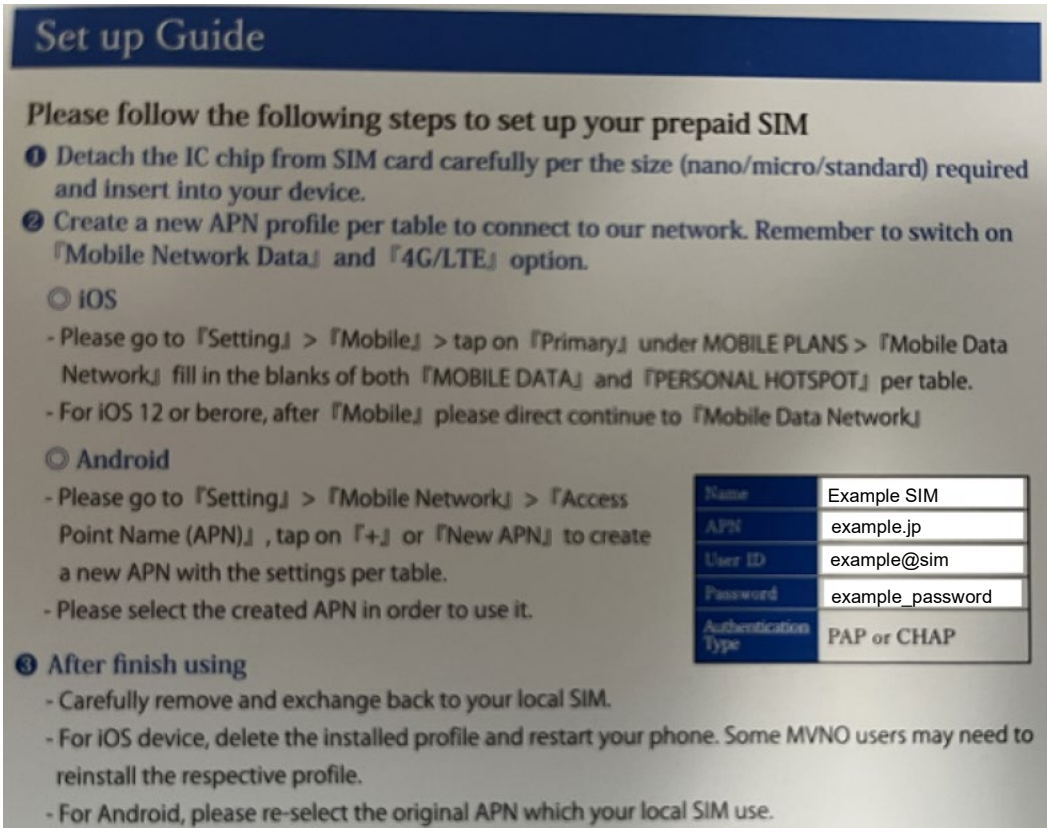


図 4-9　DHA-SIM-132 のマニュアル

iot-reference-rl78\Projects\rl78g23-fpb\modules\r_config\r_aws_cellular_config.h

```
#define AWS_CELLULAR_CFG_AP_NAME        "example.jp"        /* Access point name */
#define AWS_CELLULAR_CFG_AP_USERID      "example@sim"       /* Login ID */
#define AWS_CELLULAR_CFG_AP_PASSWORD    "example_password" /* Access point password */
#define AWS_CELLULAR_CFG_PIN_CODE                           /* SIM card PIN code */
#define AWS_CELLULAR_CFG_AUTH_TYPE      (2)     /* Authentication protocol type
(0=None,1=PAP,2=CHAP)*/
```

4.2.4　デモプロジェクトに AWS IoT 接続設定を追加

デモプロジェクトに AWS IoT 接続に必要な設定を追加します。手順を以下に示します。
ユーザ環境に合わせて変更する箇所を黄色ハイライトで示します。

① デバイスを IoT Core サービスに登録し、接続に必要な情報（エンドポイント、モノの名前、クレデンシャル）を取得します。詳細は以下を参照してください。

[デバイスを AWS IoT に登録する · renesas/iot-reference-rx Wiki · GitHub]

② デモプロジェクトにエンドポイント、モノの名前を設定します。
iot-reference-rl78\Demos\include\aws_clientcredential.h

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

③　デモプロジェクトにクレデンシャル（クライアント証明書、秘密鍵）を設定します。
iot-reference-rl78\Demos\include\aws_clientcredential_keys.h

注意事項：各行末に\n”を付与してください。

```
/*
 * @brief PEM-encoded client certificate.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
 * to the certificate that will be used for TLS client authentication.
 *
 * @note Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----\n"\
 * "...base64 data...\n"\
 * "-----END CERTIFICATE-----\n"
 */
#define keyCLIENT_CERTIFICATE_PEM \
"-----BEGIN CERTIFICATE-----\n"\
"MIIDWTCCAkGgAwIBAgIUFeYR3JSsJbTOS7huEq++YBGgwtowDQYJKoZIhvcNAQEL\n"\
・・・
"7qHumsC6fsEapoptgcfEpdERl4c9hJR45jHamDVhxZjitQD4klLA0gqTlBNL\n"\
"-----END CERTIFICATE-----\n"
…
/*
 * @brief PEM-encoded client private key.
 *
 * @todo If you are running one of the FreeRTOS demo projects, set this
…
* @note Must include the PEM header and footer:
 * "-----BEGIN RSA PRIVATE KEY-----\n"\
 * "...base64 data...\n"\
 * "-----END RSA PRIVATE KEY-----\n"
 */
#define keyCLIENT_PRIVATE_KEY_PEM \
"-----BEGIN RSA PRIVATE KEY-----\n"\
"MIIEowIBAAKCAQEA3Fb7O7jQW4lgHmPE3AInUTWUCaR7kWeWHubEk9YbNf3xwxdg\n"\
・・・
"s/OlVUiygf0RgeoMVx/3GzZPfmTrB0cQ8XZ7mxCd2dgY9UXQ/oja\n"\
"-----END RSA PRIVATE KEY-----\n"
```

5.　デモプロジェクト(PubSub)固有のセットアップ

デモプロジェクト(PubSub)固有のセットアップ手順を示します。

5.1　事前準備

特にありません。

5.2　プロジェクトのインポート

プロジェクト「aws_ryz024a_rl78g23-fpb」を e² studio にインポートします。次の手順でインポートウィザードを開いてください。

File > Import… > Existing Projects into Workspace > Next

次に、プロジェクト「aws_ryz024a_rl78g23-fpb」を選択してください。この時、ワークスペースにプロジェクトをコピーしないオプション設定にしてください。その後、Finish ボタンをクリックします。

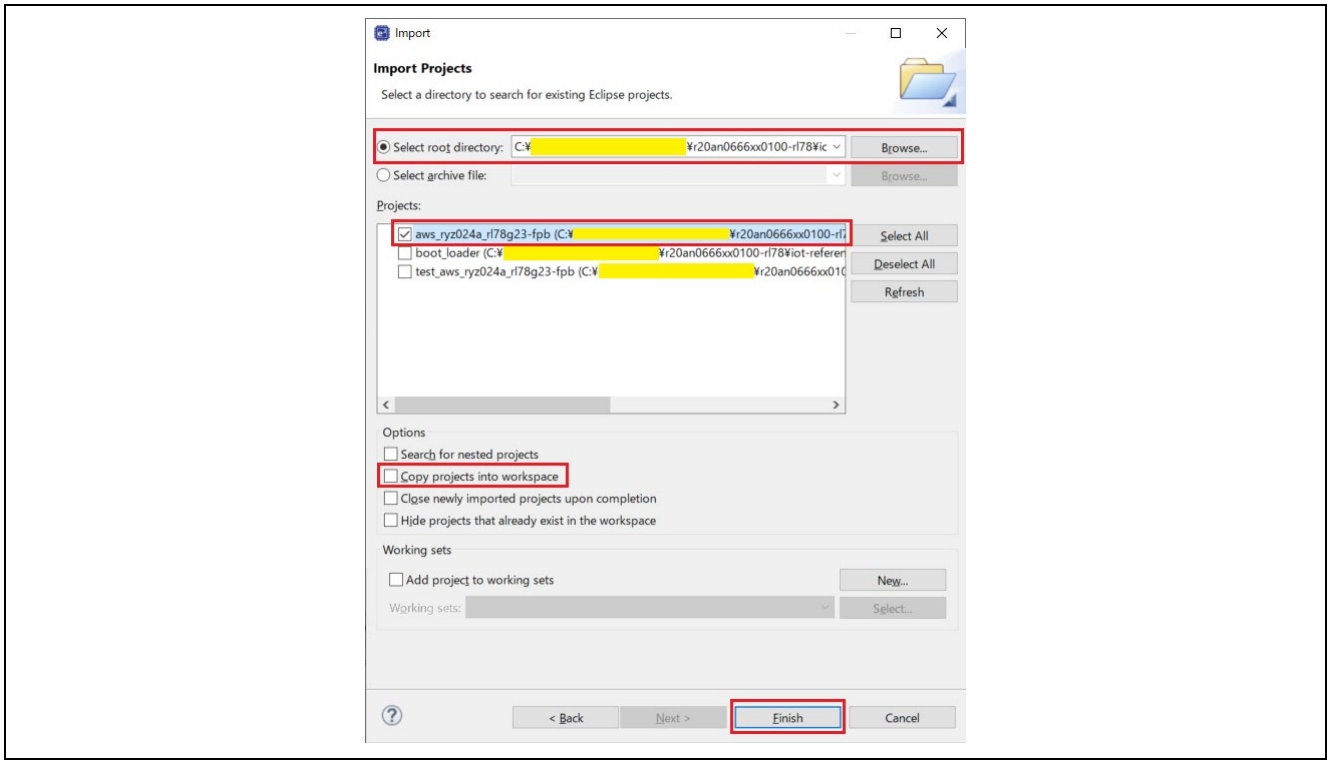


図 5-1　プロジェクト「aws_ryz024a_rl78g23-fpb」を選択

プロジェクト・エクスプローラー画面にインポートされたプロジェクトが表示されます。

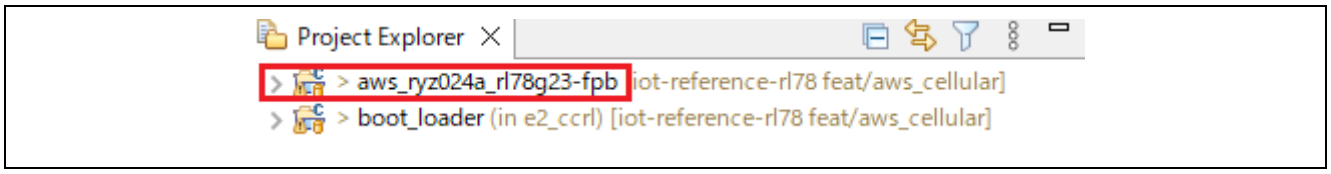


図 5-2　プロジェクト「aws_ryz024a_rl78g23-fpb」のインポート完了

5.3　ビルド構成を設定

aws_ryz024a_rl78g23-fpb プロジェクトのビルド構成“HardwareDebug”をアクティブにします。

Build Configurations > Set Active > Select “HardwareDebug”

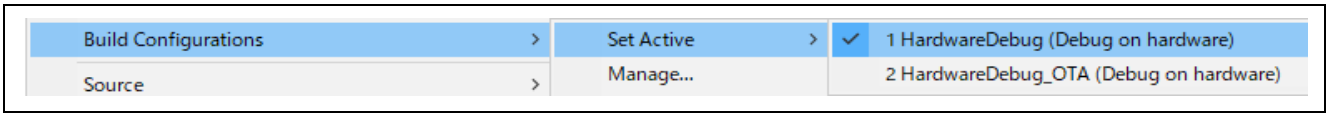


図 5-3　ビルド構成“HardwareDebug”をアクティブにする

5.4　デモプロジェクトのビルド

aws_ryz024a_rl78g23-fpb プロジェクトをビルドし、MOT ファイルを作成します。
その後、プロジェクトフォルダ直下の HardwareDebug フォルダに aws_ryz024a_rl78g23-fpb.mot が生成されていることを確認します。

5.5　MQTT テストクライアントの準備

AWS マネジメントコンソールにアクセスし、IoT Core サービス内の「MQTT テストクライアント」から「pubsub_demo」をサブスクライブすることでマイコンボードから送られるメッセージをテキスト形式で確認できます。

①　タブ“Subscribe to a topic”を選択します。

AWS IoT > MQTT test client >Select “Subscribe to a topic”

②　トピックフィルタに“pubsub_demo/#”を入力し”Subscribe”を押します。

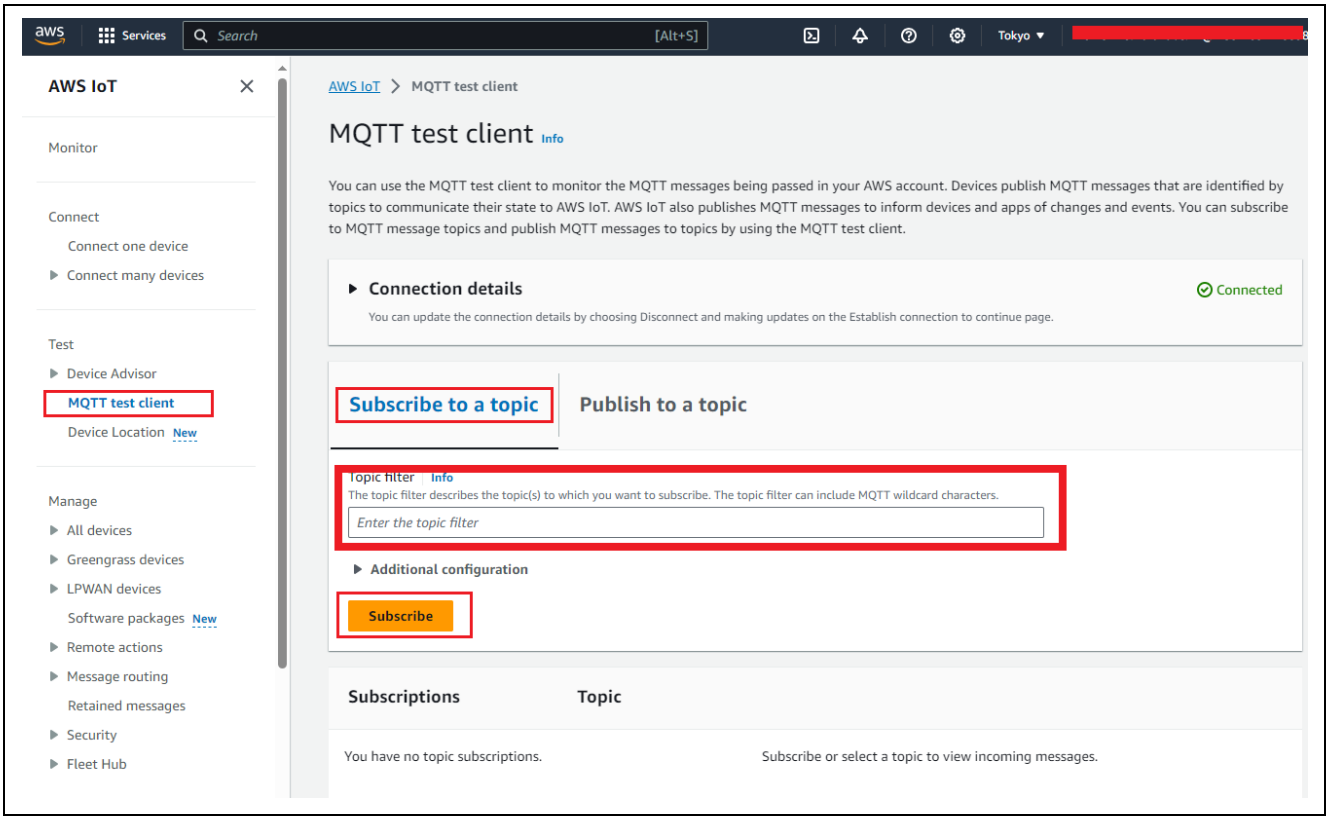


図 5-4　MQTT テストクライアント設定

5.6　デモプロジェクトの実行

デモプロジェクト(PubSub)の実行手順を示します。

① Renesas Flash Programmer でマイコンボードに aws_ryz024a_rl78g23-fpb.mot を書き込みます。

書き込み方法は「7 章 Renesas Flash Programmer の使用方法」を参照してください。

② 書き込みが終了するとデモプロジェクト(PubSub)が起動します。

ターミナルで PubSub Demo Task0 と PubSub Demo Task1 のメッセージ送信結果が SUCCESS であることを確認します。

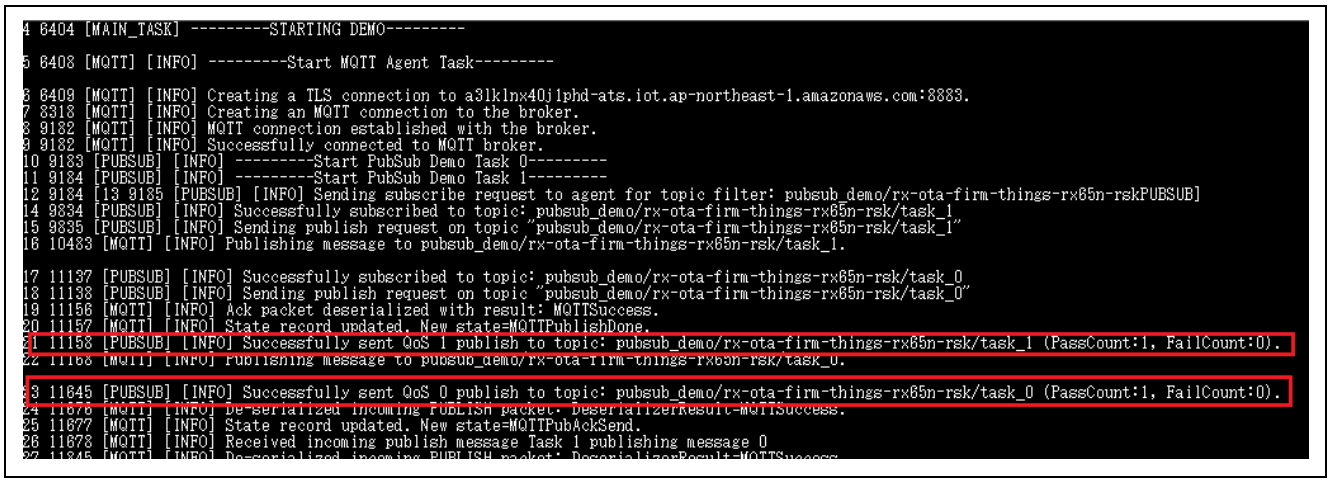


図 5-5　ターミナルでデモプロジェクトの実行結果を確認

③ MQTT テストクライアントで PubSub Demo Task0 と PubSub Demo Task1 から送られたメッセージが表示されていることを確認します。

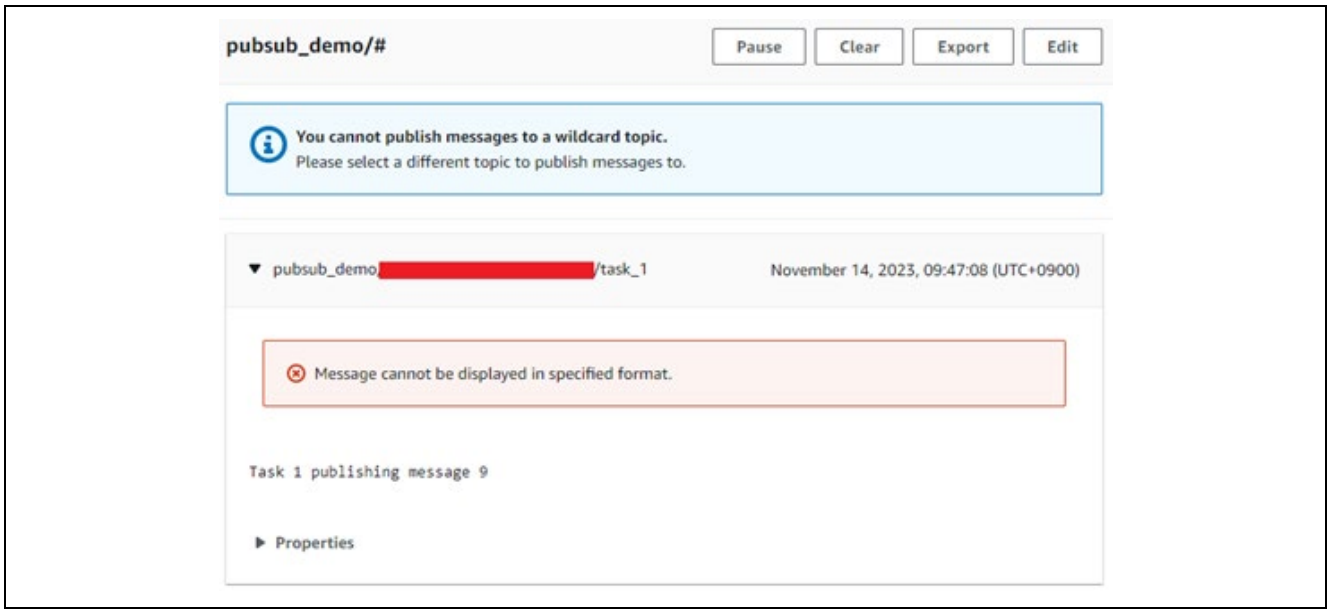


図 5-6　MQTT テストクライアントでデモプロジェクトの実行結果を確認

5.7　デモプロジェクトのデバッグ方法

デモプロジェクト(PubSub)を e² studio から起動してデバッグする手順を示します。

① デモプロジェクトをビルドします。

「5.2 章　プロジェクトのインポート」、「5.3 章　ビルド構成を設定」、「5.4 章　デモプロジェクトのビルド」を参照してください。

② デバッグを開始します。

「8 章 デバッグ手順」を参照してください。

6. デモプロジェクト(OTA)固有のセットアップ

本デモプロジェクトはマイコンボードから AWS に接続し AWS IoT OTA を使用したファームウェア更新を行います。本章はセットアップ手順を示します。

6.1 事前準備

6.1.1 ツールのインストール

デモプロジェクト実行に必要なツールをインストールします。

① Python をインストールする

1. Python は Renesas Image Generator の動作に必要です。バージョン 3.9.0 以上をインストールしてください Python は https://www.python.org/から入手できます。
2. Python をインストール後、以下のコマンドで pycryptodome パッケージをインストールしてください。

```
> pip install pycryptodome
```

② OpenSSL をインストールする

初期イメージと更新イメージ作成時にコード署名検証を行うために必要な鍵を作成します。作成には OpenSSL を使用します。

1. OpenSSL をインストールしていない場合は次の URL を Web ブラウザで開いてください。
   Win32/Win64 OpenSSL Installer for Windows - Shining Light Productions (slproweb.com)
2. Win64OpenSSL v3.x.x Light をダウンロードしてインストールしてください。

③ Renesas Image Generator をダウンロードする

「RL78/G22,RL78/G23,RL78/G24 ファームウェア アップデート モジュール」に含まれる Renesas Image Generator(V3.03)をダウンロードしてください。

6.1.2　署名生成/検証用鍵の作成

OpenSSL を使用してファームウェア検証用の鍵を生成します。黄色ハイライト箇所は入力するコマンドを示します。

① CA 証明書

```
$ openssl ecparam -genkey -name secp256r1 -out ca.key
using curve name prime256v1 instead of secp256r1
$ openssl req -x509 -sha256 -new -nodes -key ca.key -days 3650 -out ca.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:JP
State or Province Name (full name) [Some-State]:Tokyo
Locality Name (eg, city) []:Kodaira
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Renesas Electronics
Organizational Unit Name (eg, section) []:Software Development Division
Common Name (e.g. server FQDN or YOUR name) []:Renesas Tarou
Email Address []:Tarou.Renesas@sample.com
```

② 楕円曲線暗号(secp256r1)の鍵ペア

```
$ openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
using curve name prime256v1 instead of secp256r1
```

③ 鍵ペアの証明書

```
$ openssl req -new -sha256 -key secp256r1.keypair > secp256r1.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:JP
State or Province Name (full name) [Some-State]:Tokyo
Locality Name (eg, city) []:Kodaira
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Renesas Electronics
Organizational Unit Name (eg, section) []:Software Development Division
Common Name (e.g. server FQDN or YOUR name) []: Renesas Tarou
Email Address []: Tarou.Renesas@sample.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

④　CA 証明書を使用して鍵ペアの証明書を作成

```
$ openssl x509 -req -sha256 -days 3650 -in secp256r1.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out secp256r1.crt
Signature ok
subject=C = JP, ST = Tokyo, L = Kodaira, O = Renesas Electronics, OU = Software
Development Division, CN = Renesas Tarou, emailAddress = Tarou.Renesas@sample.com
Getting CA Private Key
```

⑤　楕円曲線暗号(secp256r1)の秘密鍵を抽出

```
$ openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey
read EC key
writing EC key
```

⑥　楕円曲線暗号(secp256r1)の公開鍵を抽出

```
$ openssl ec -in secp256r1.keypair -outform PEM -pubout -out secp256r1.publickey
read EC key
writing EC key
```

6.1.3　OTA アップデートのための設定

6.1.3.1　Amazon S3 バケットの作成

①　Amazon S3 > Buckets > “Create bucket”

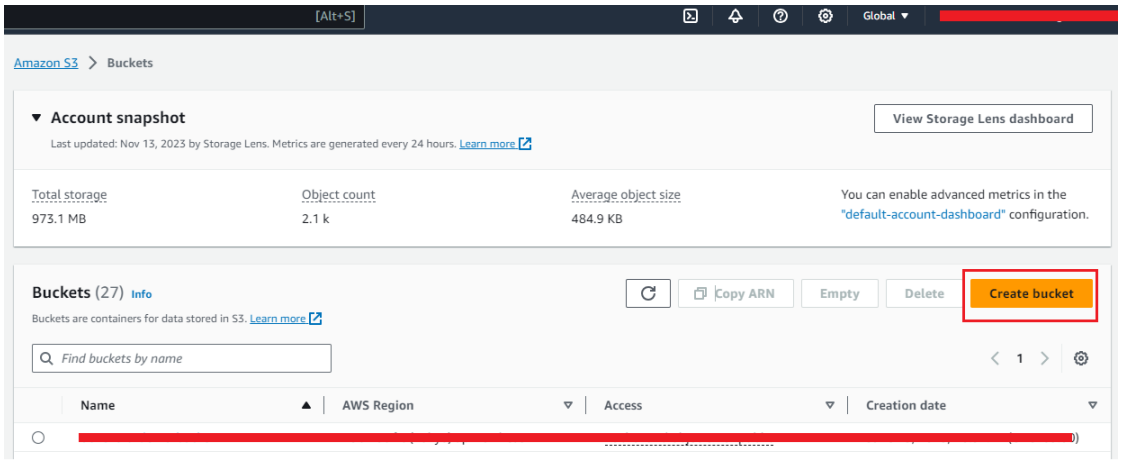


図 6-1　Create bucket

②　General Configuration

- Bucket name: Your bucket name
- AWS Region: Asia Pacific (Tokyo) ap-northeast-1

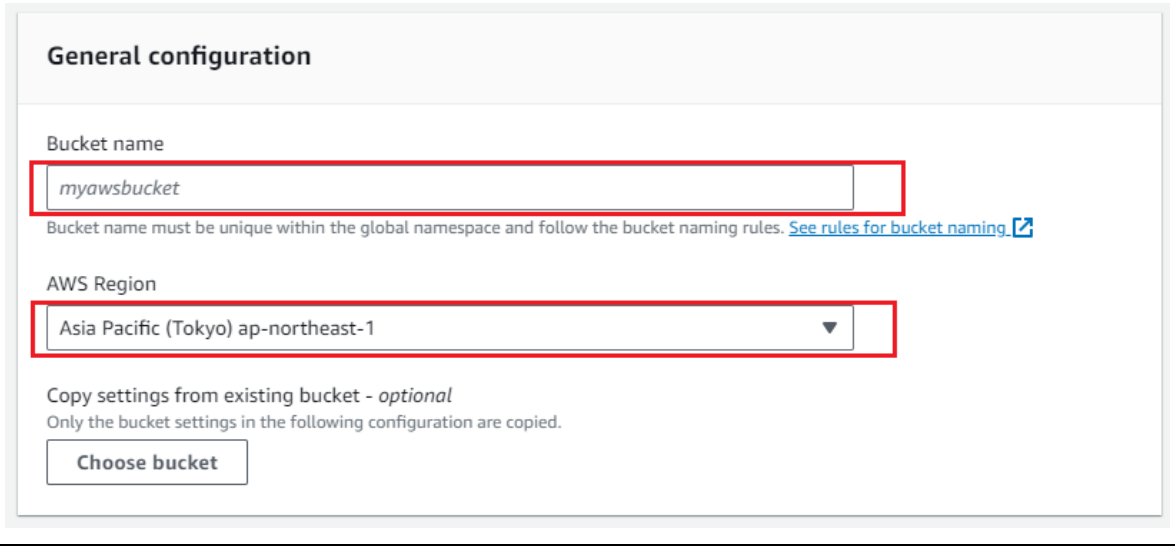


図 6-2　General configuration

③　Object Ownership

- ACLs disabled を選択

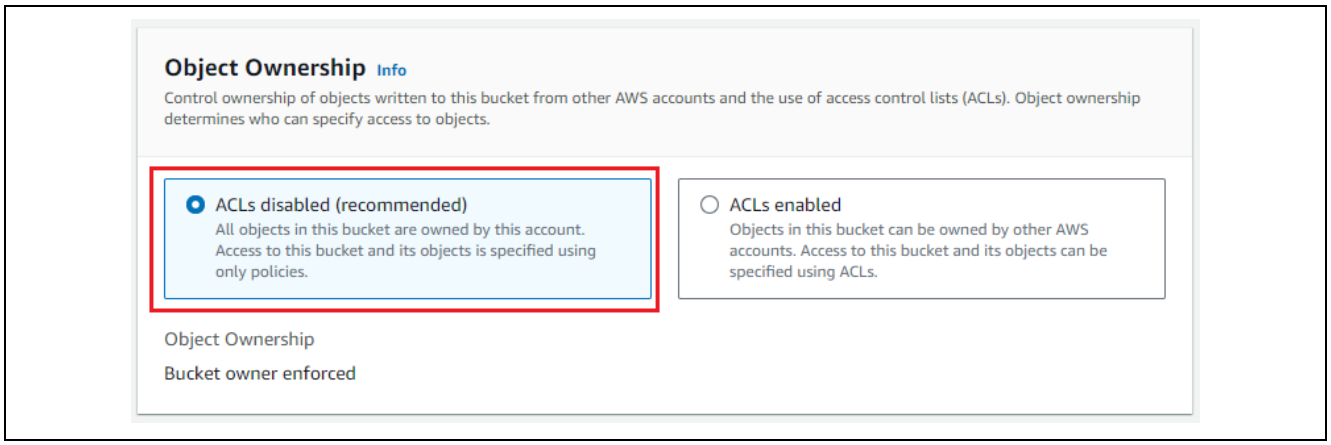


図 6-3　Object Ownership

④　Block Public Access settings for this bucket

- Block all public access を選択

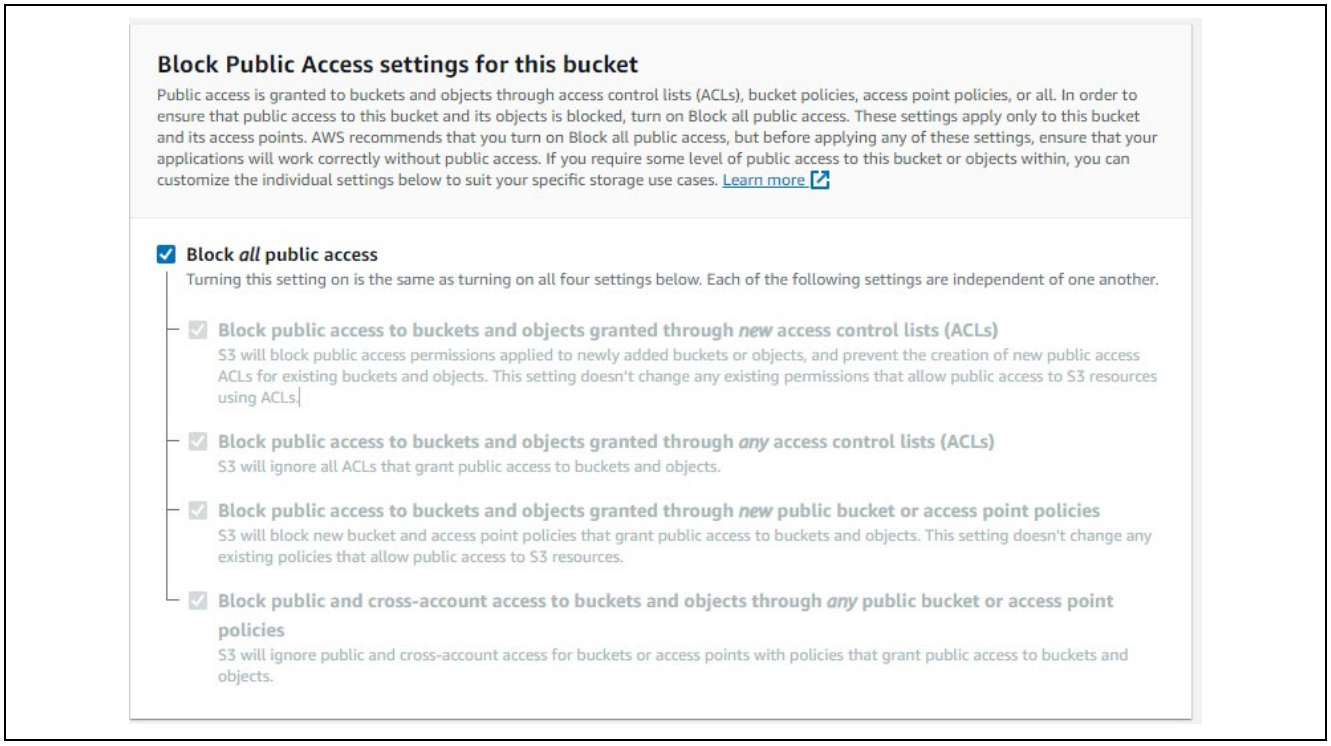


図 6-4　Block Public Access settings for this bucket

⑤　Bucket Versioning

- Bucket Versioning: Disable

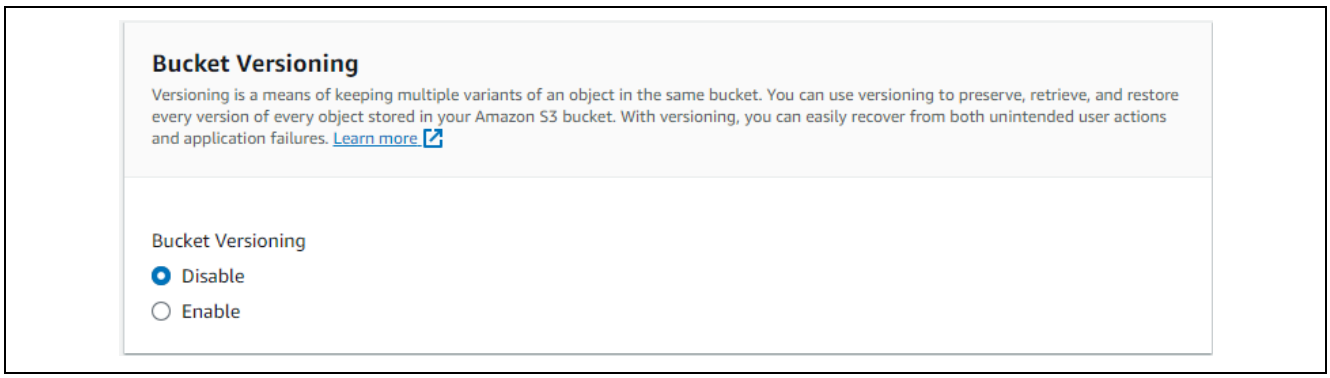


図 6-5　Bucket Versioning

⑥　Default encryption

- Encryption type: Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Bucket Key: Enable

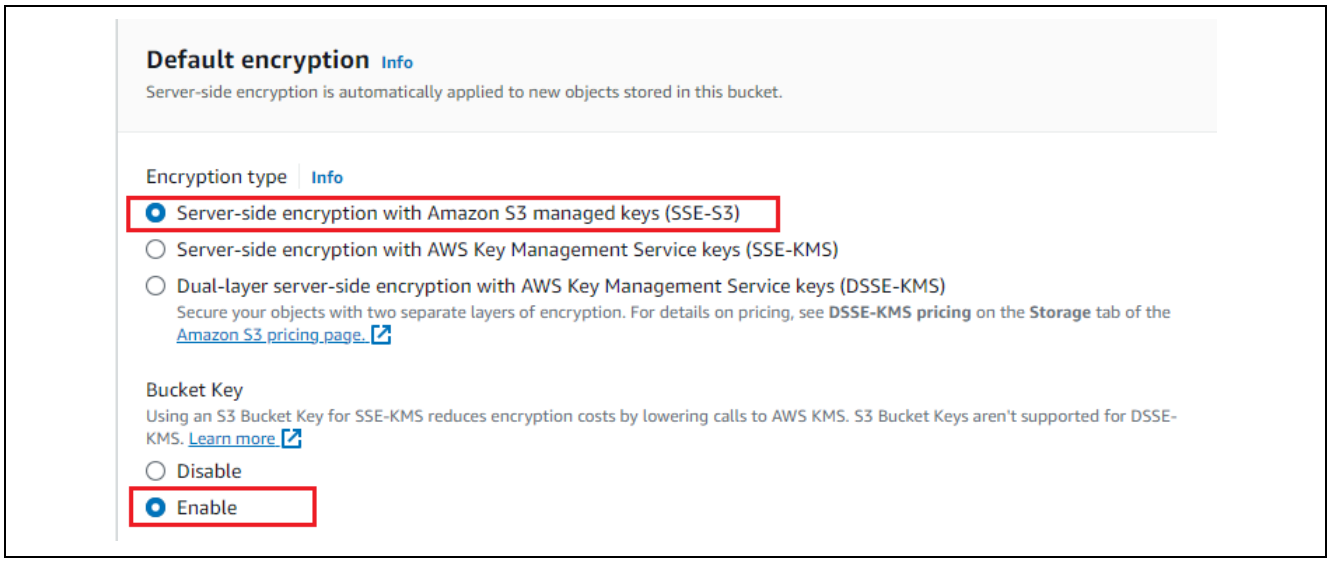


図 6-6　Default encryption

⑦　“Create bucket” をクリック

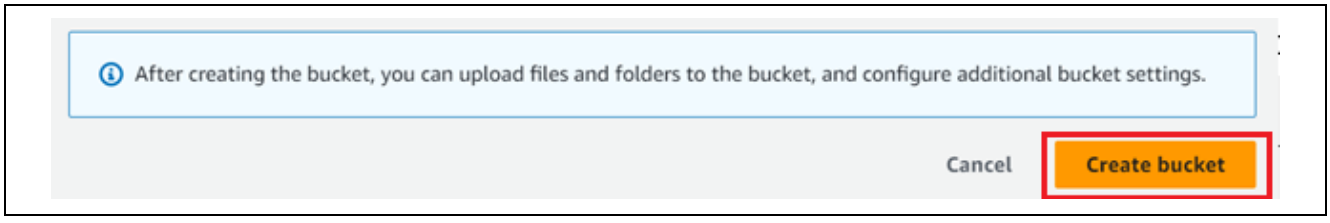


図 6-7　Click “Create bucket”

6.1.3.2　OTA 更新用サービスロールの作成

①　IAM > Roles > ”Create role”

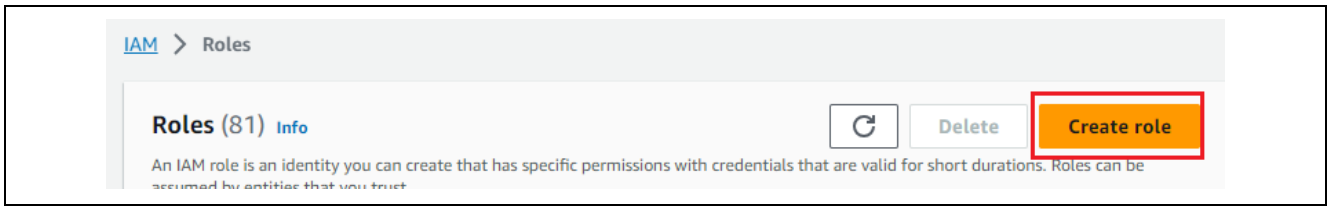


図 6-8　IAM > Roles > Create role

②　Step1: Select trusted entity

- Trusted entity type: AWS service
- Use case: Service or use case > IoT

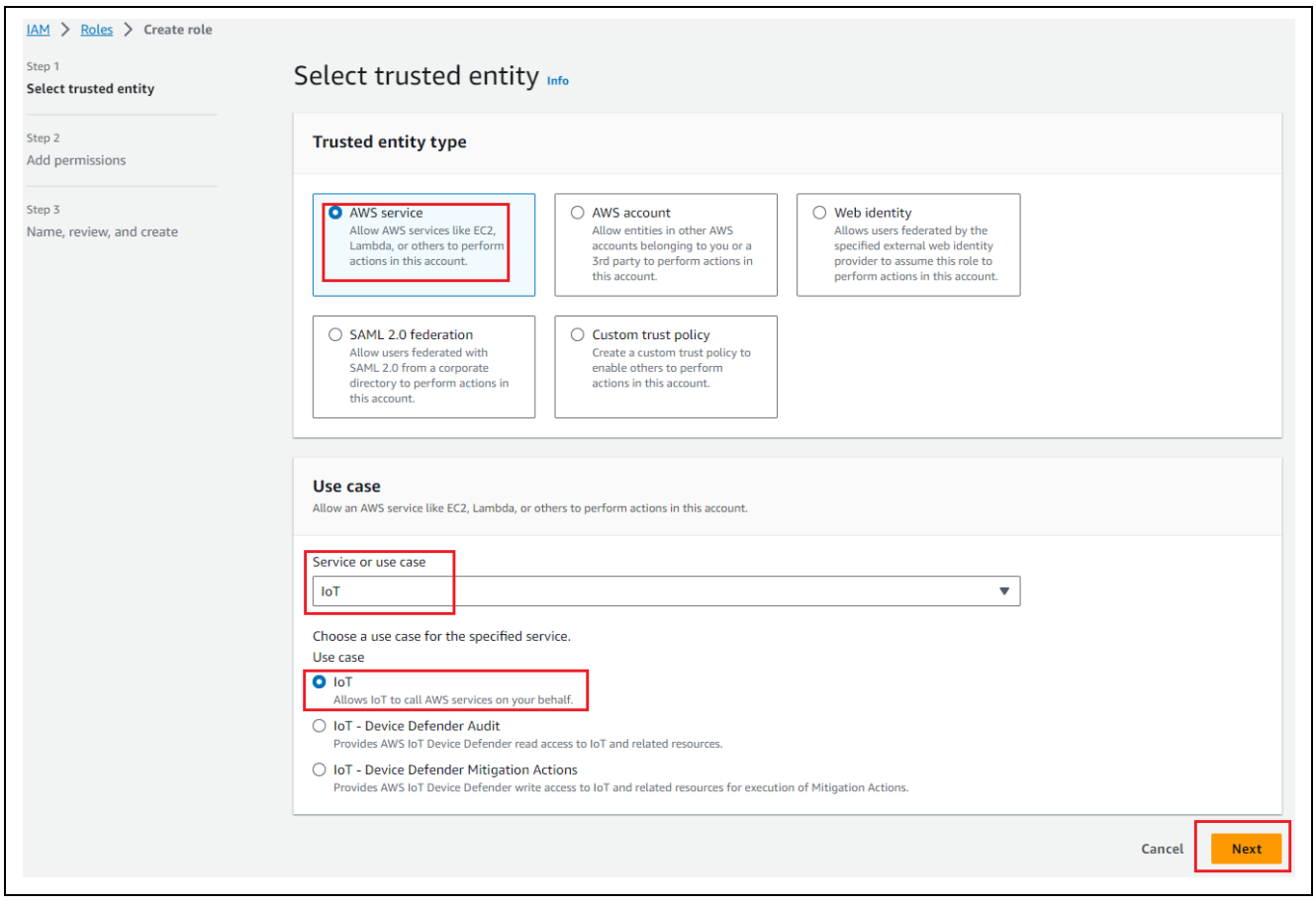


図 6-9　Step1: Select trusted entity

③　Step2: Add permissions

- AWSIoTLogging
- AWSIoTRuleActions
- AWSIoTThingsRegistration

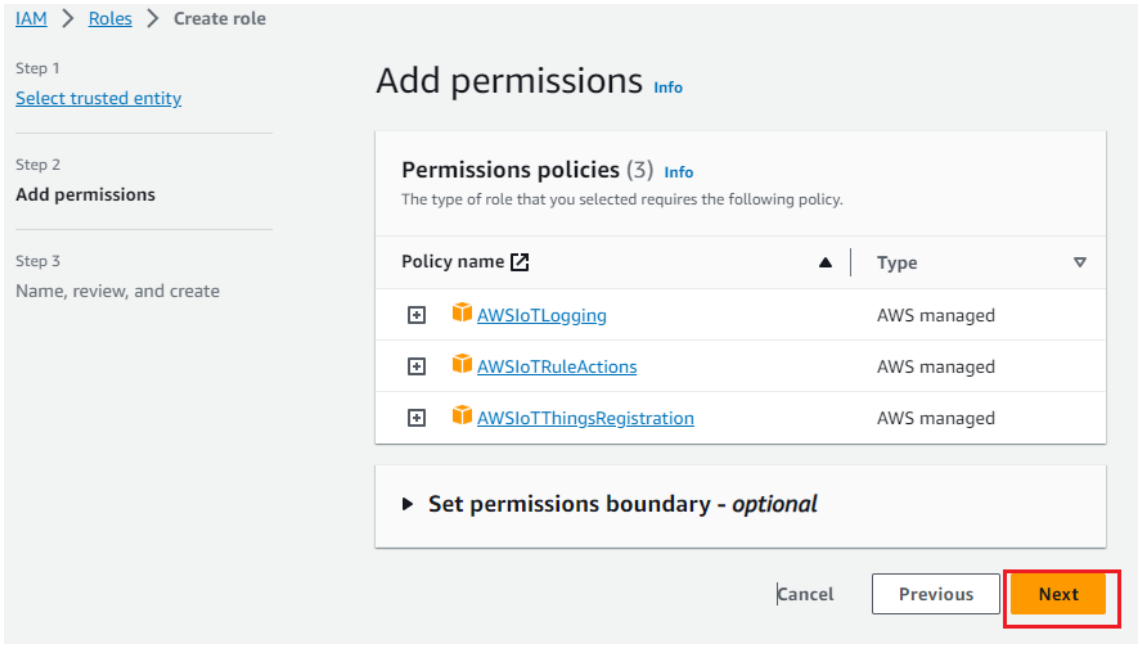


図 6-10　Step2: Add permissions

④　Step3: Name, review, and create > Role details

- Role name: 任意
- Description: 任意

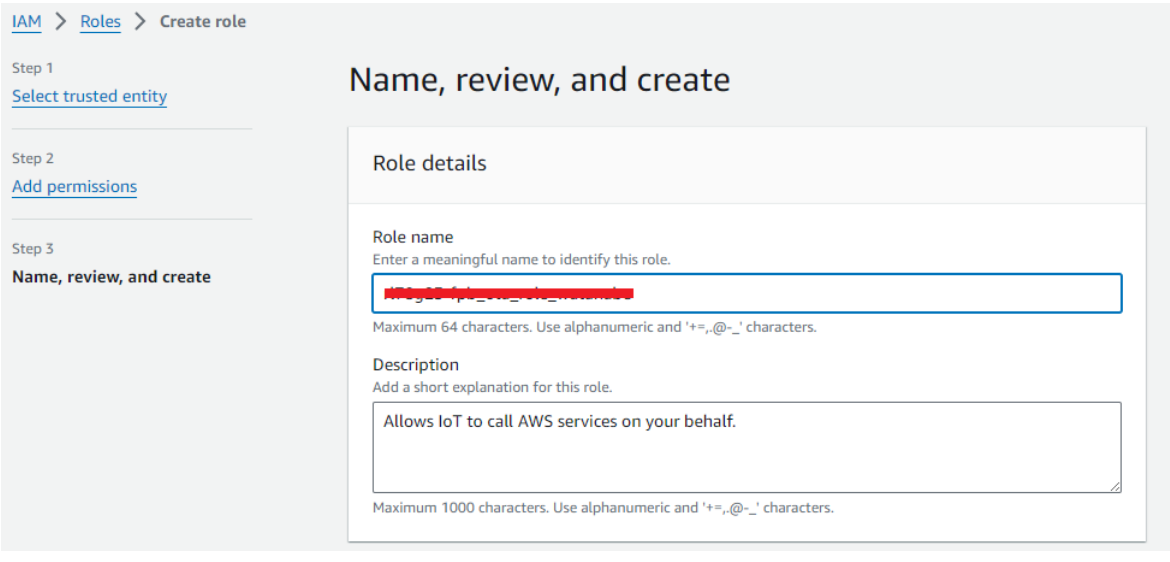


図 6-11　Step3: Name, review, and create > Role details

⑤　Step3: Name, review, and create > Step 1: Selected trusted entities

- Default

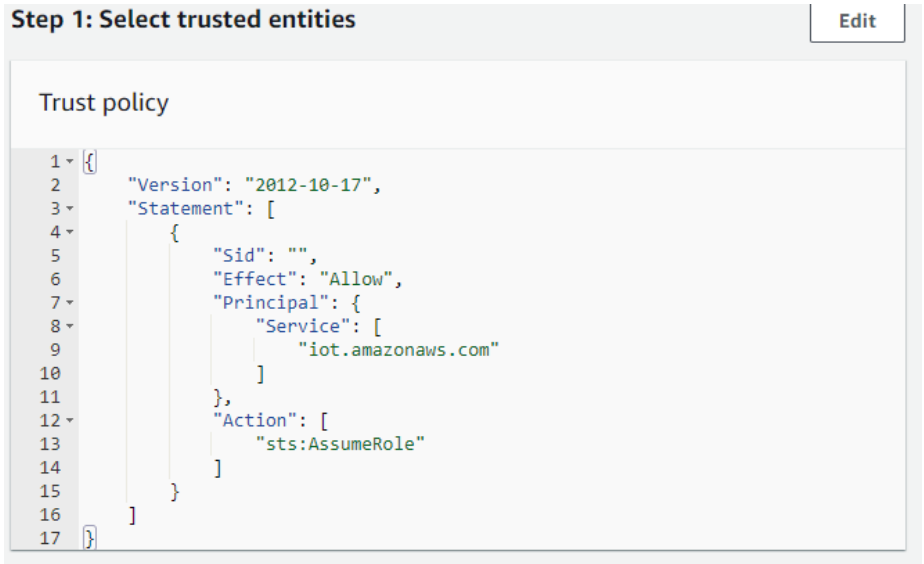


図 6-12　Step3: Name, review, and create > Step 1:Selected trusted entities

⑥　Step3: Name, review, and create > Step 2: Add permissions

- Default

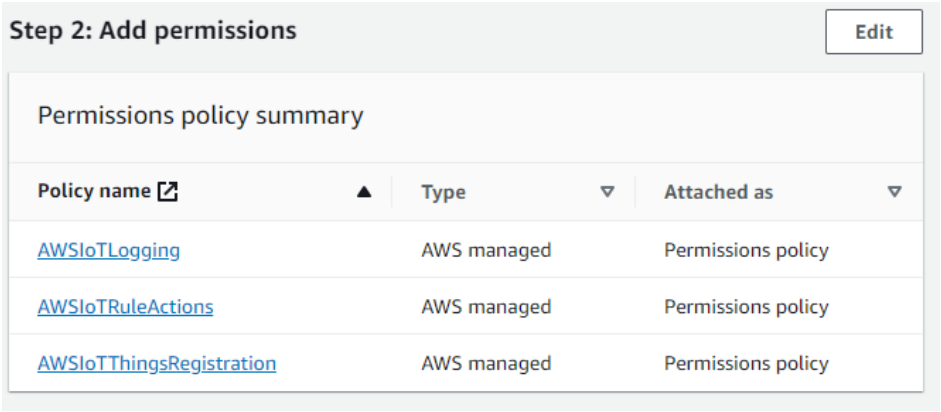


図 6-13　Step3: Name, review, and create > Step 2: Add permissions

⑦　Step3: Name, review, and create > Step 3: Add tags

- Default
- “Create role” をクリック

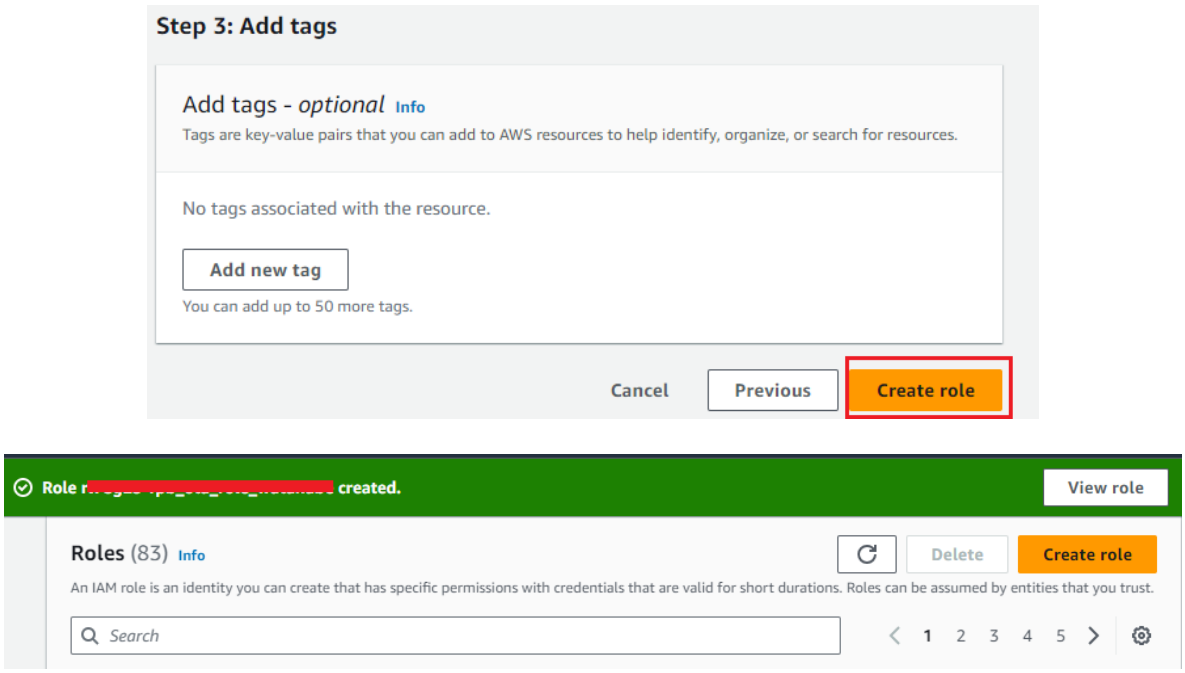


図 6-14　Step3: Name, review, and create > Step 3: Add tags

6.1.3.3　OTA 更新用ユーザポリシーの作成

①　「6.1.3.2 章　OTA 更新用サービスロールの作成」で作成したロールを開きます。

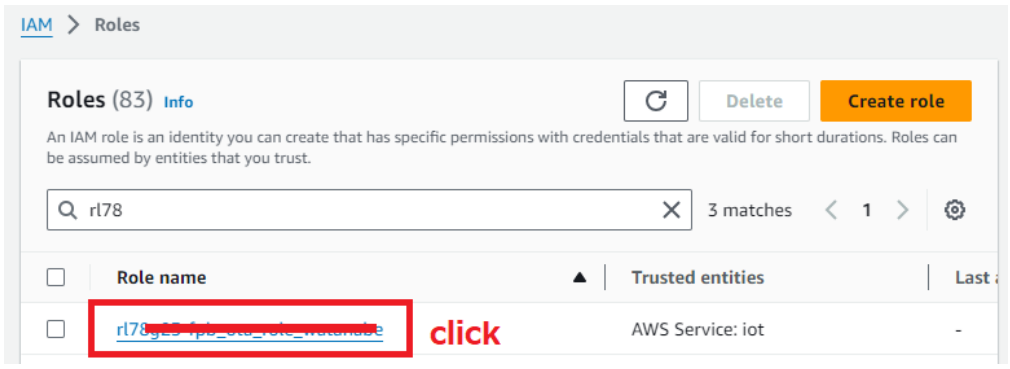


図 6-15　作成した OTA 更新用サービスロールを開く

②　My role > Summary: Default

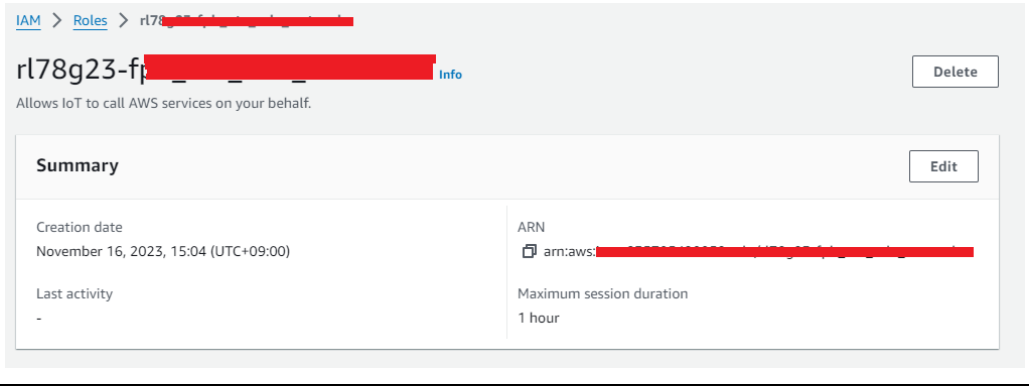


図 6-16　作成した OTA 更新用サービスロールのサマリーを表示

③　ポリシー”AmazonFreeRTOSOTAUpdate”をアタッチします。

- My role > Permissions policies > “Add permissions“ > Attach policies

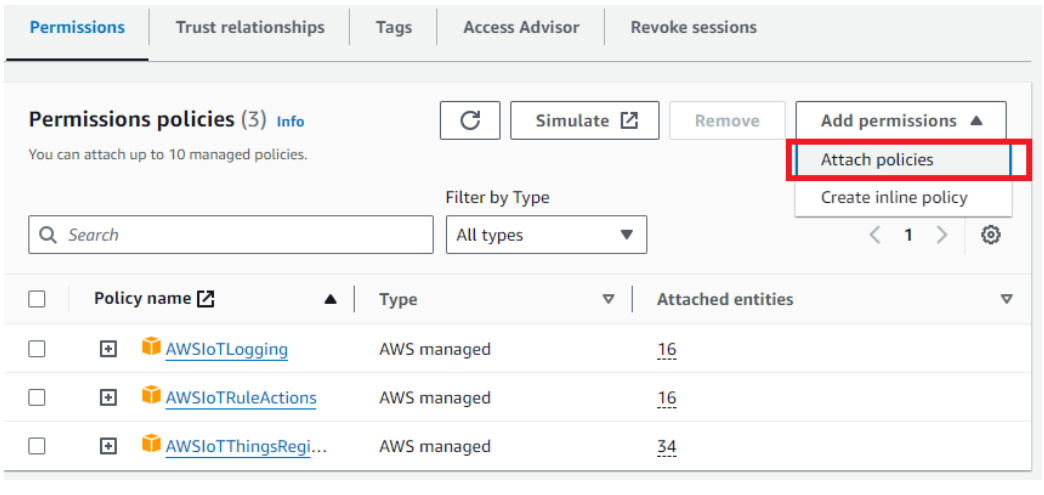


図 6-17　作成した OTA 更新用サービスロールにポリシーをアタッチ

- Choose “AmazonFreeRTOSOTAUpdate” > “Add permissions”

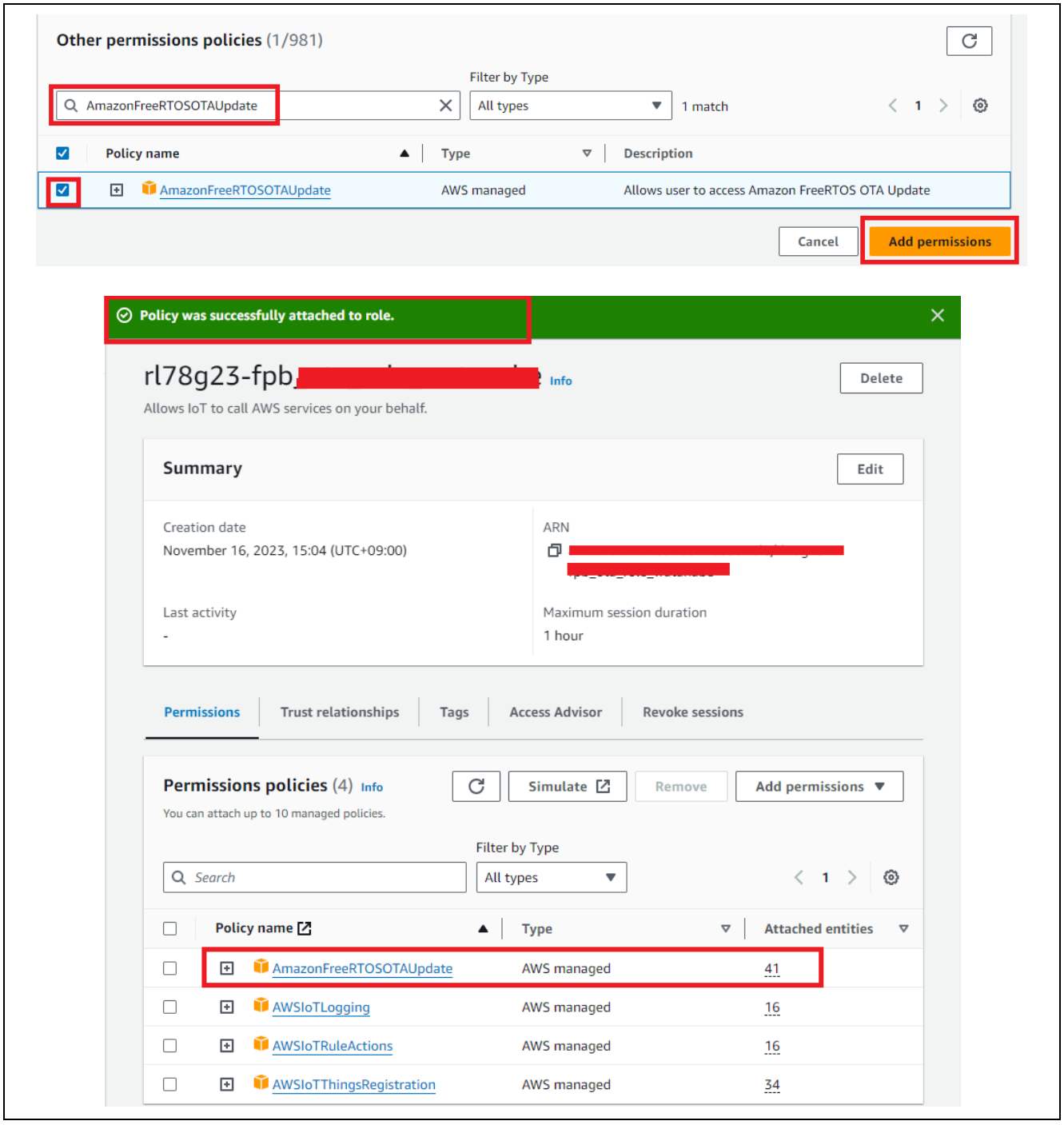


図 6-18　ポリシー”AmazonFreeRTOSOTAUpdate”を作成した OTA 更新用サービスロールにアタッチ

④　インラインポリシー(S3)を追加します。

- “Add permissions“ > Create inline policy > “JSON”

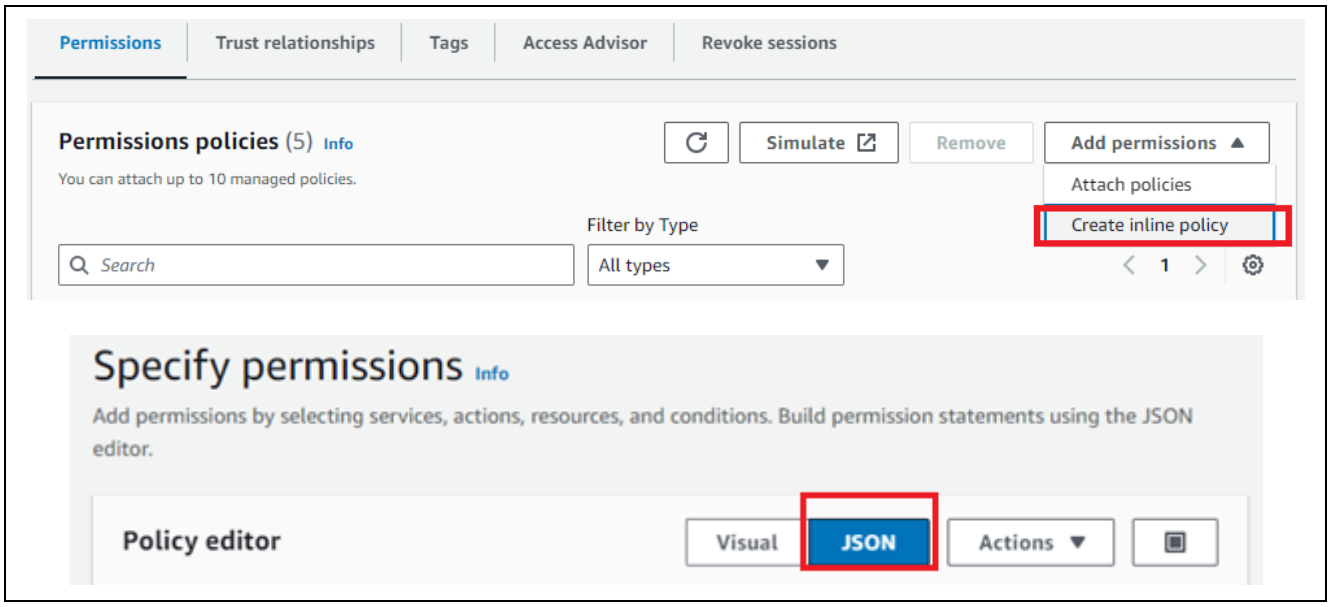


図 6-19　S3 のインラインポリシーを作成

- Policy エディタに以下をペースト> Next
  - s3-bucket-test は「6.1.3.1 章　Amazon S3 バケットの作成」で作成した Bucket name に変更

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion",
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-test/*"
        }
    ]
}
```

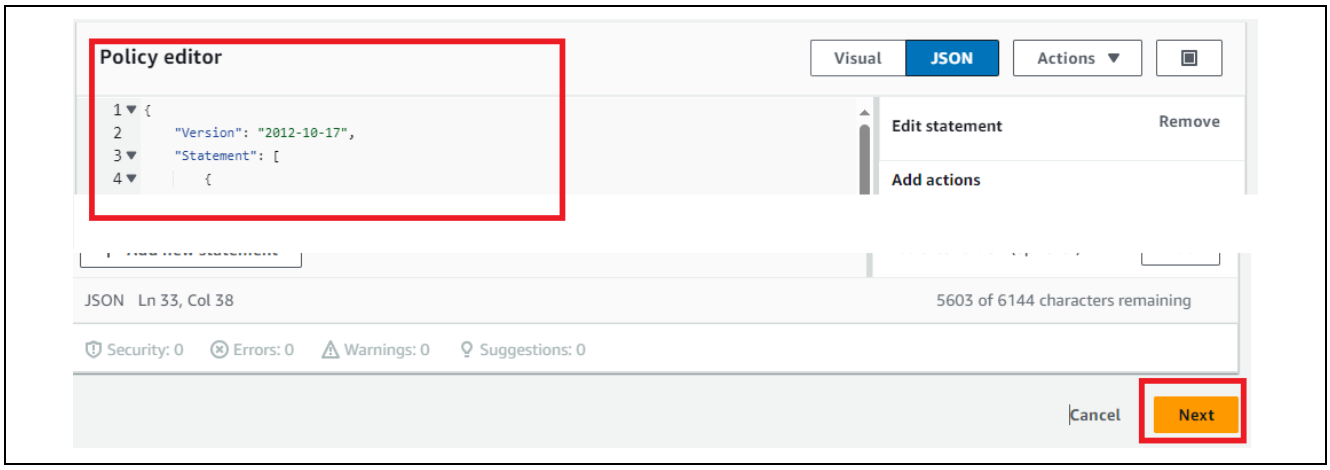


図 6-20　Policy エディタに S3 のポリシーを追加

- Policy name: 任意 (例：inline-policy-s3-test) > “Create policy”

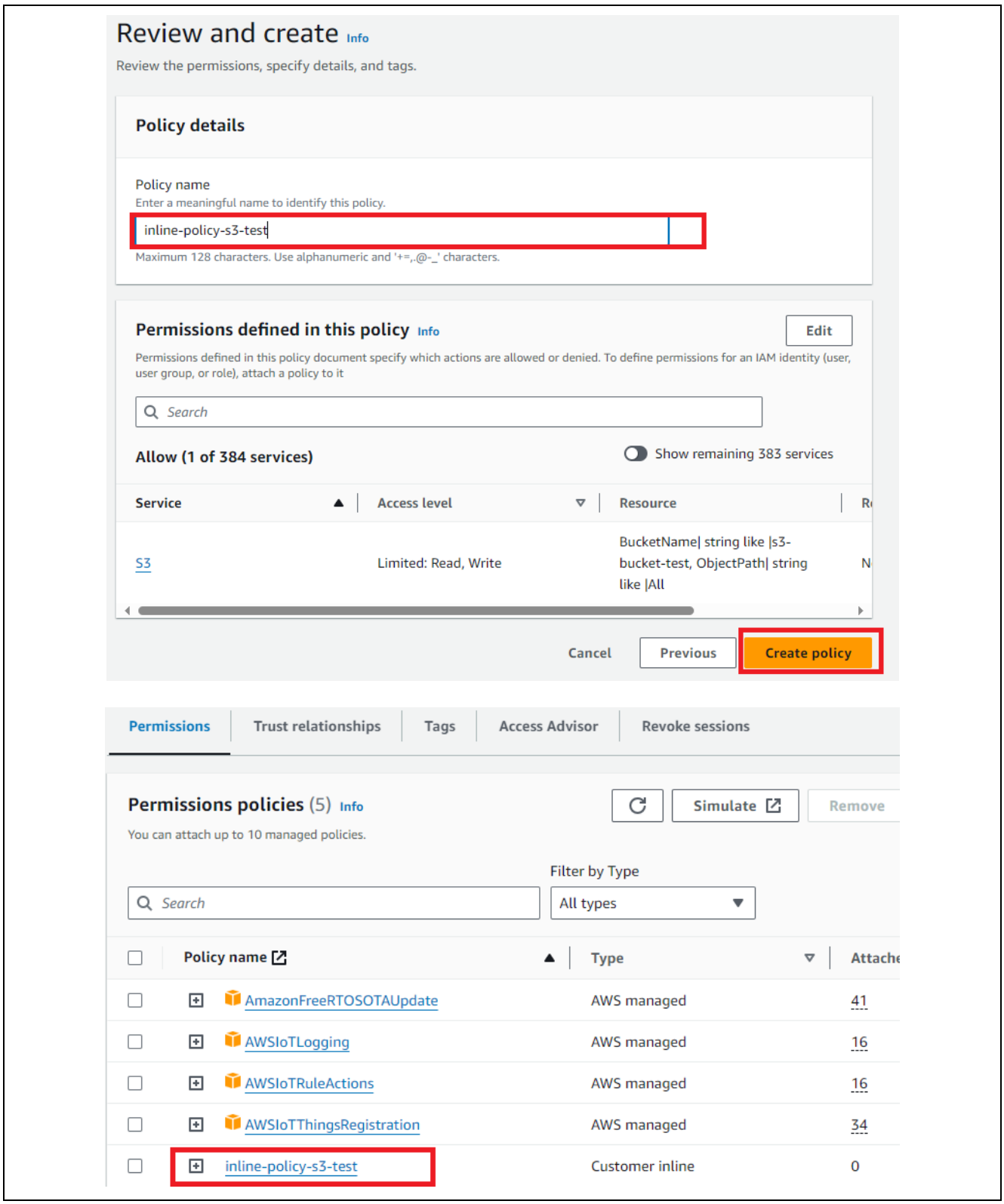


図 6-21　名前 (例：inline-policy-s3-test)を付けて S3 ポリシーを作成

⑤　IAM のインラインポリシーを追加します。

- “Add permissions“ > Create inline policy > “JSON”

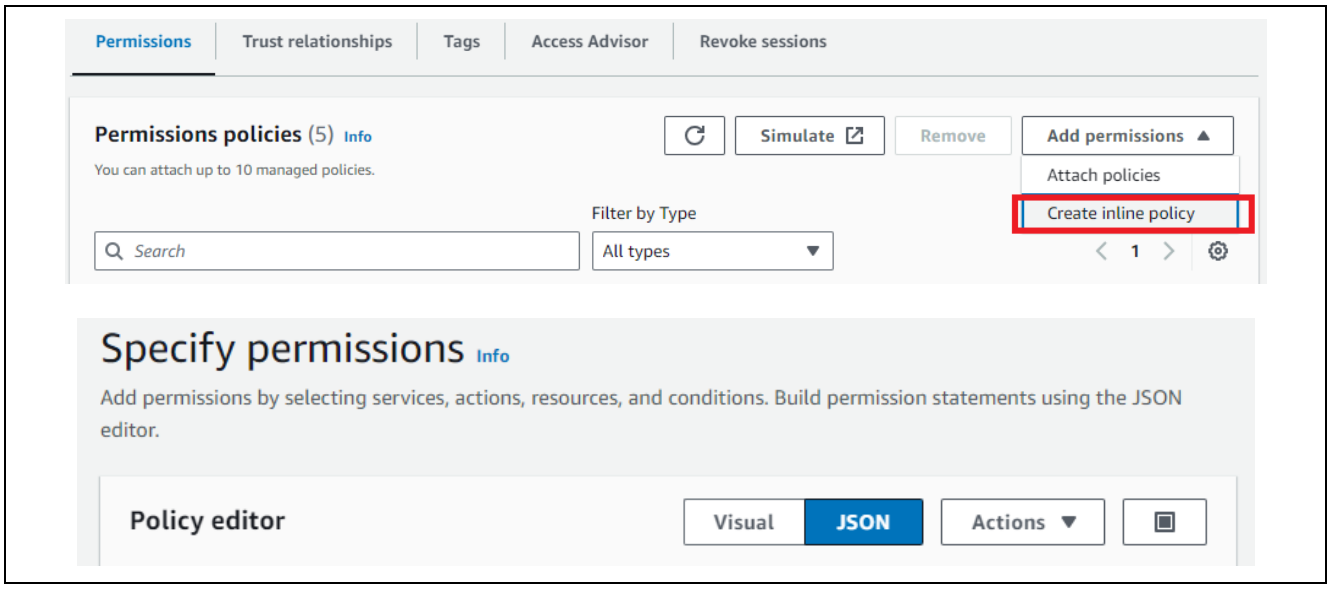


図 6-22　インラインポリシーを作成

- Policy エディタに以下をペースト> Next
  - ota-role-test は「6.1.3.2 章　OTA 更新用サービスロールの作成」で作成した Role name に変更

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::xxxxxxxxxx:role/ota-role-test"
        }
    ]
}
```

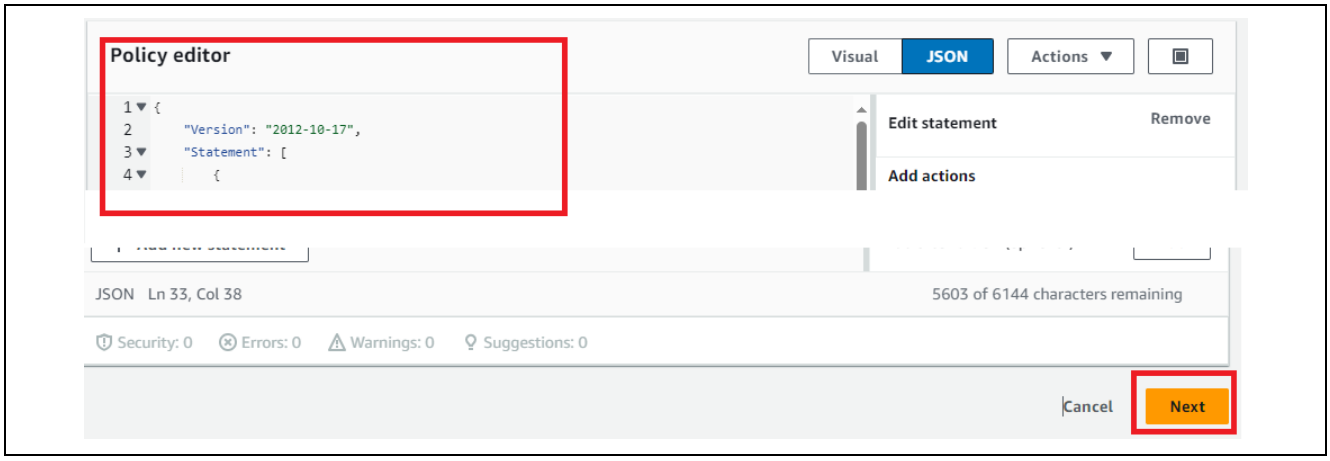


図 6-23　インラインポリシーに IAM ロールを追加

- Policy name: 任意(例：inline-policy-iam-test) > “Create policy”

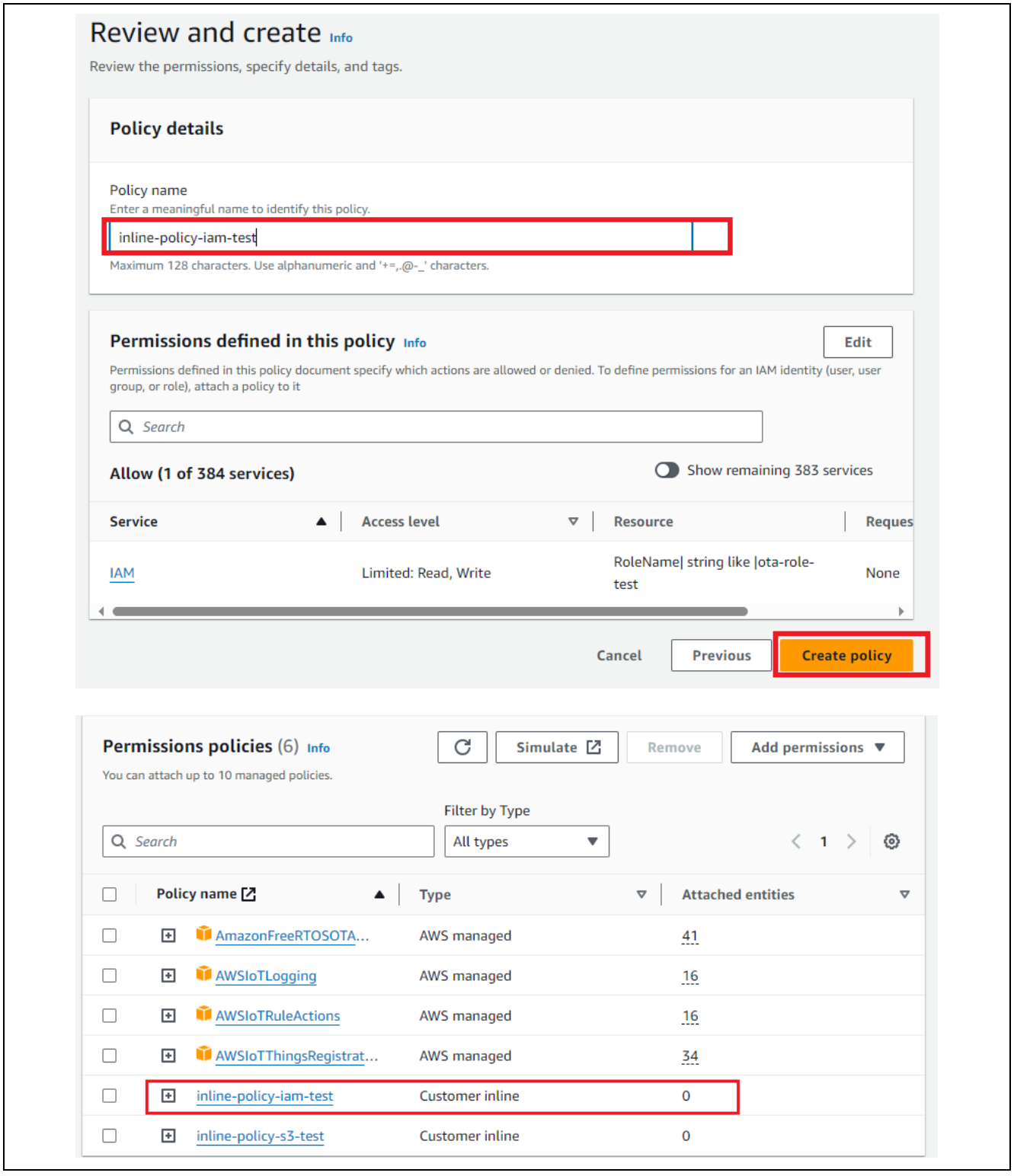


図 6-24　名前 (例：inline-policy-iam-test)を付けて IAM インラインポリシーを保存

6.1.3.4　OTA 更新用ポリシーを IAM ユーザに割り当て

① OTA 更新用ポリシーを作成します。

- IAM > policies > “Create policy” > “JSON”

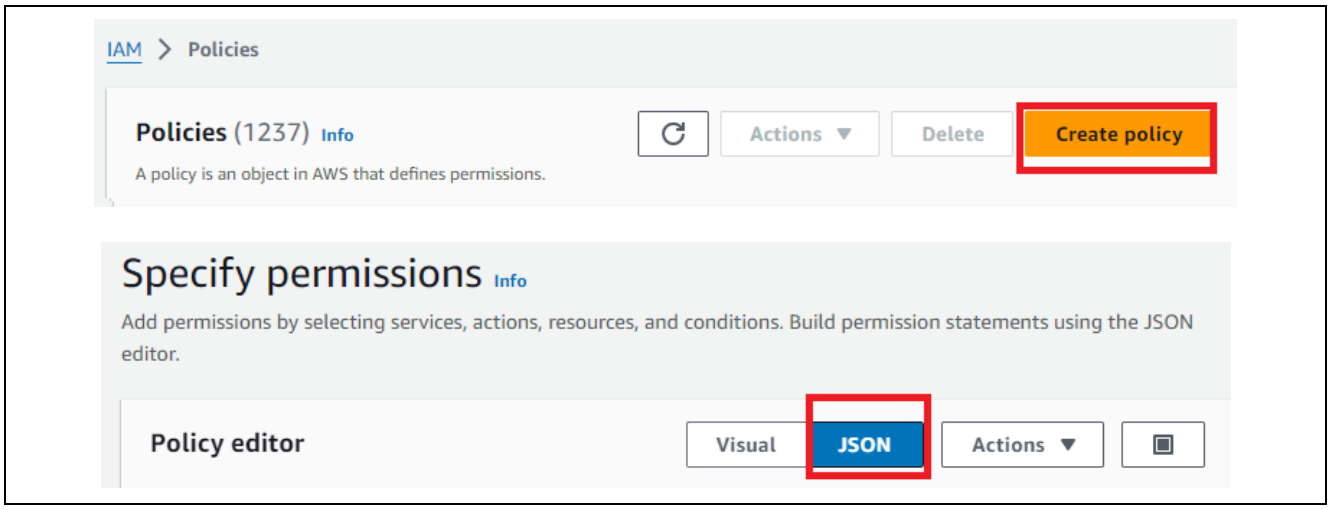


図 6-25　OTA 更新用ポリシーを作成

- Policy エディタに以下をペースト> Next
  - s3-bucket-test は「6.1.3.1 章　Amazon S3 バケットの作成」で指定した Bucket name に変更
  - ota-role-test は「6.1.3.2 章　OTA 更新用サービスロールの作成」で指定した Role name に変更

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:CreateBucket",
                "s3:PutBucketVersioning",
                "s3:GetBucketLocation",
                "s3:GetObjectVersion",
                "acm:ImportCertificate",
                "acm:ListCertificates",
                "iot:*",
                "iam:ListRoles",
                "freertos:ListHardwarePlatforms",
                "freertos:DescribeHardwarePlatform"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::s3-bucket-test/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::xxxxxxxxxx:role/ota-role-test"
        }
    ]
}
```

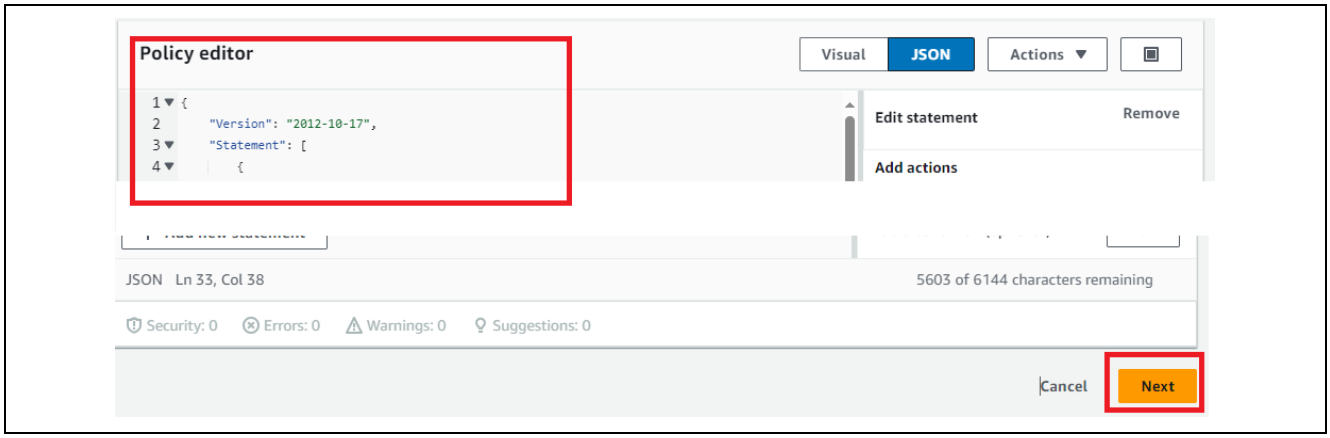


図 6-26　ポリシーエディタで OTA 更新用ポリシーを作成

- Policy name: 任意 (例：rl78g23-fpb_ota_policy) > “Create policy”

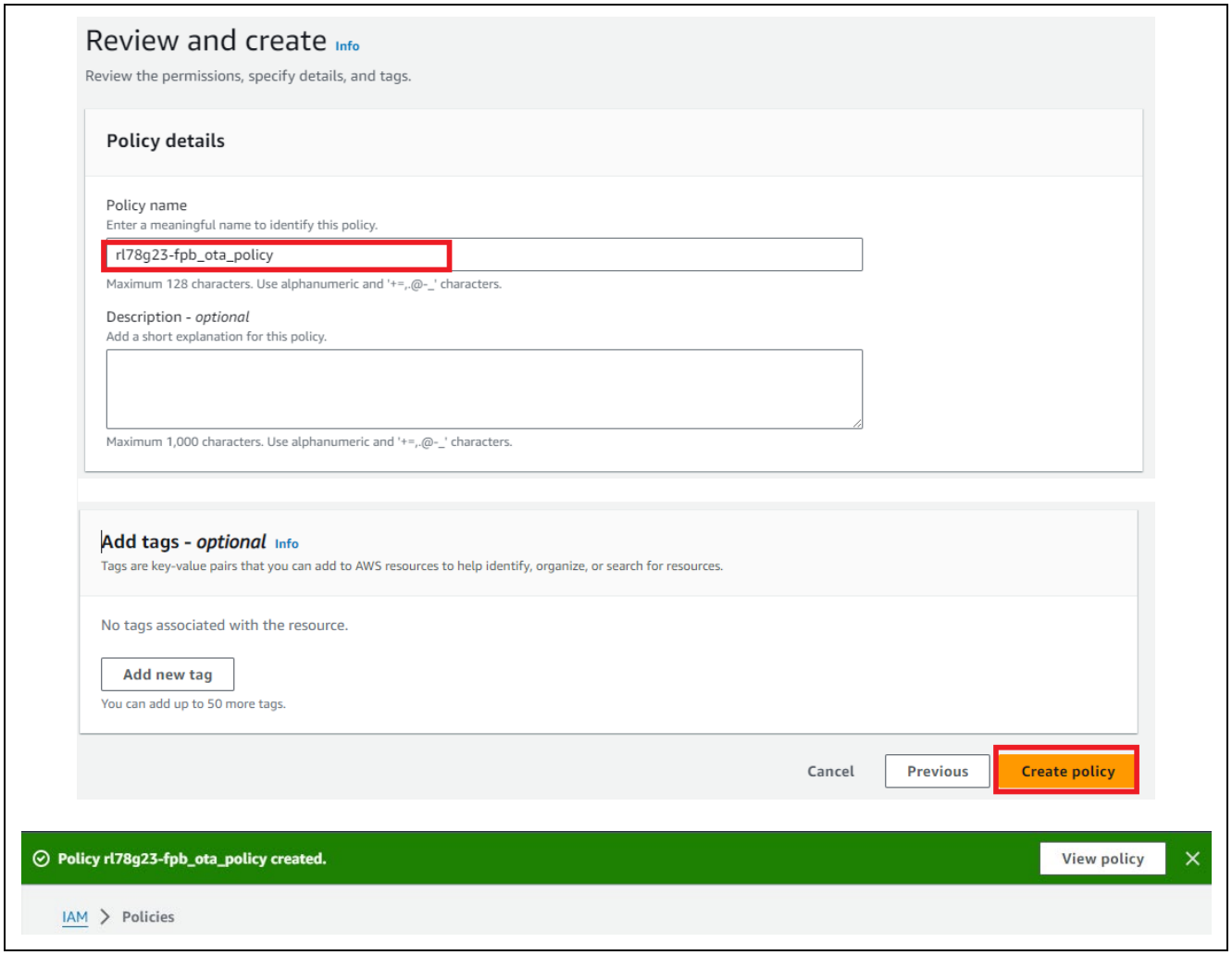


図 6-27　名前 (例：rl78g23-fpb_ota_policy)を付けて OTA 更新用ポリシーを保存

② 作成した OTA 更新用ポリシーを IAM ユーザに追加します。

- IAM > Users > Choose User > Add permissions

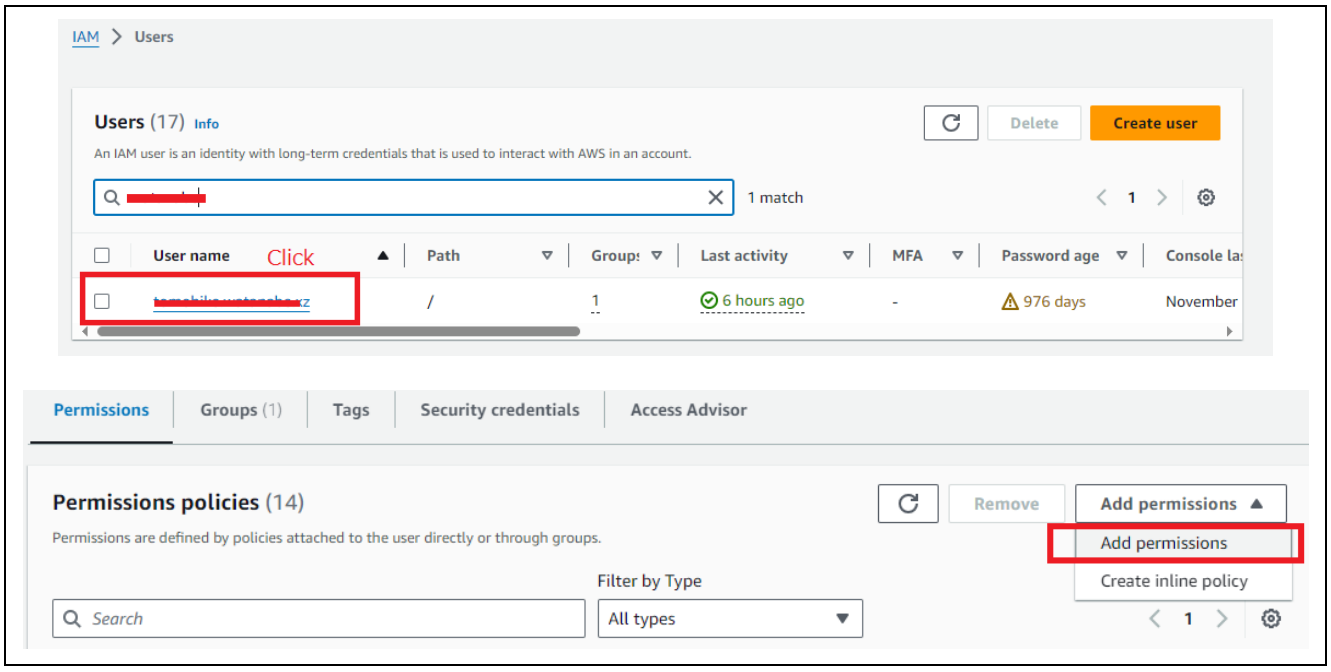


図 6-28　IAM ユーザを選択

- Permissions options: Attach policies directly
- Permissions policies > Policy name: 作成した OTA 更新用ポリシーの名前 (例：rl78g23-fpb_ota_policy)
- “Next” をクリック

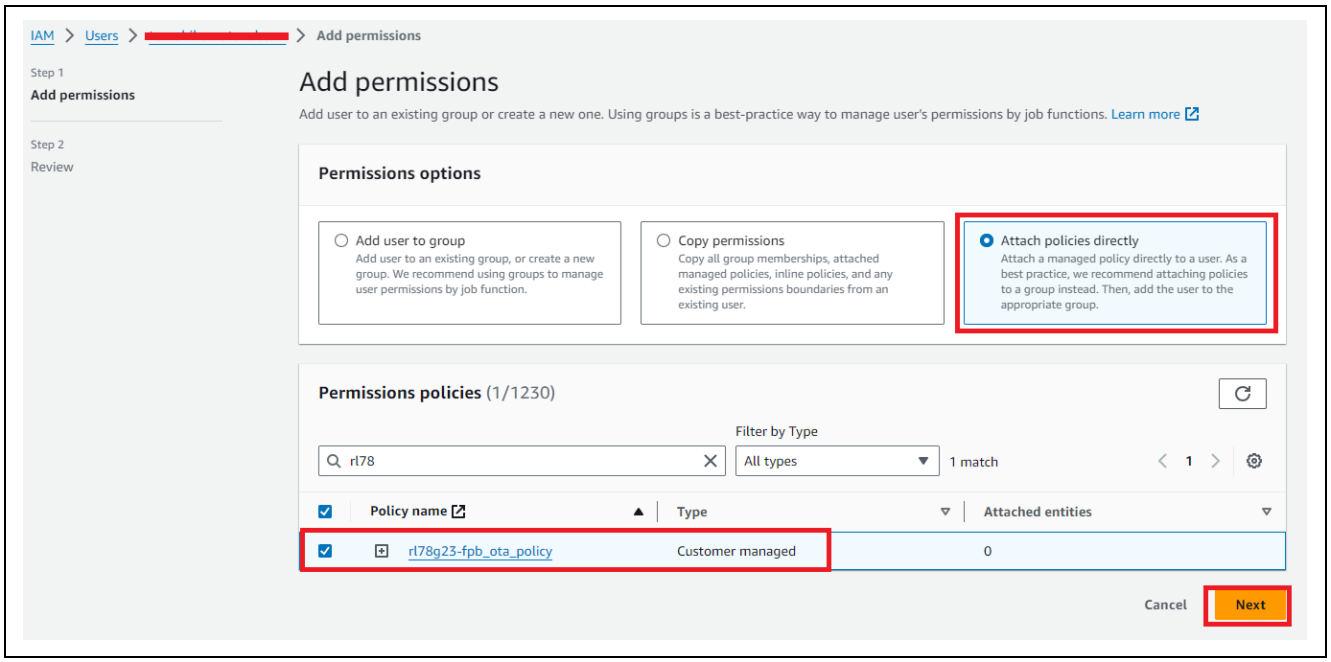


図 6-29　IAM ユーザに追加するパーミッションに OTA 更新用ポリシーを選択

- User details: Your account
- Permissions summary > name: 作成した OTA 更新用ポリシーの名前 (例：rl78g23-fpb_ota_policy)
- “Add permission” をクリック

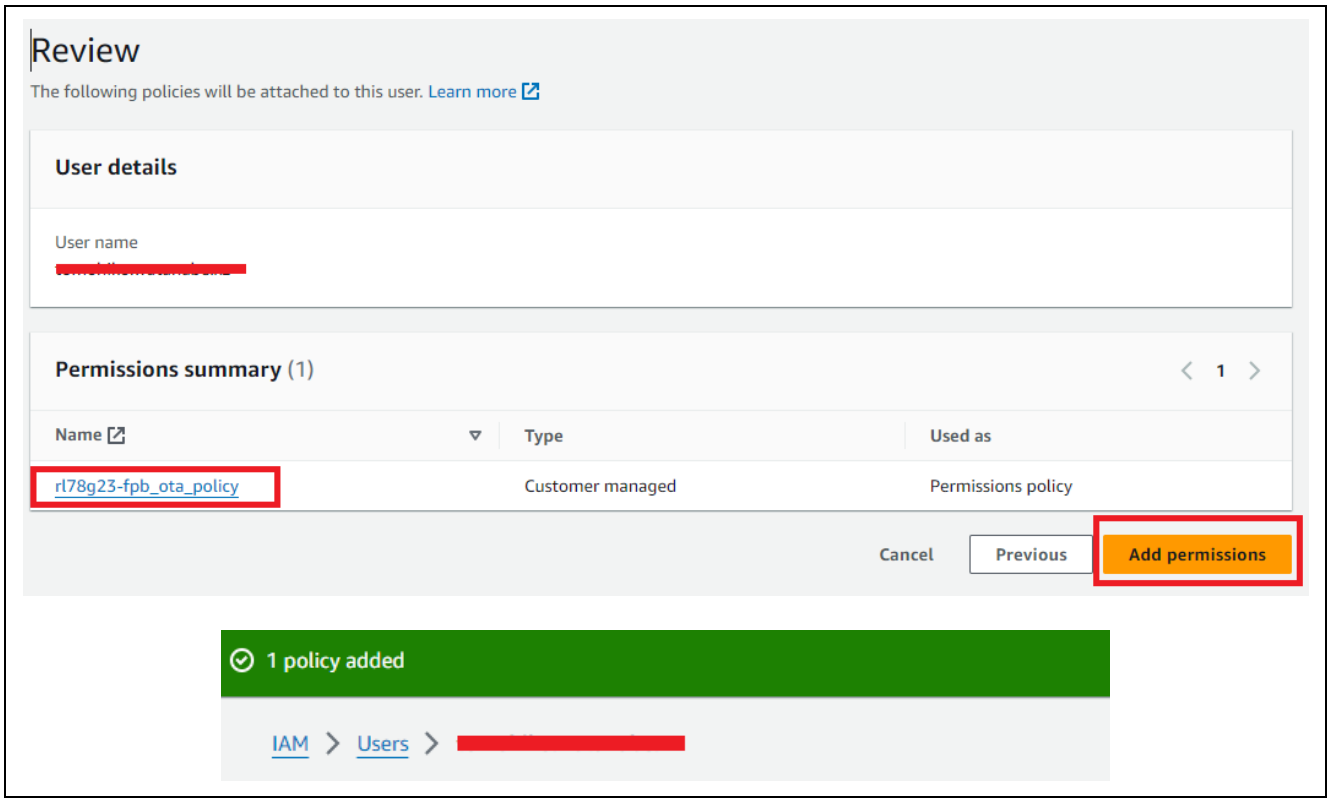


図 6-30　選択した IAM ユーザに OTA 更新ポリシーを追加

6.1.3.5　AWS IoT のコード署名へのアクセス権付与

①　IAM のポリシーを作成します。

- IAM > policies > “Create policy” > “JSON”

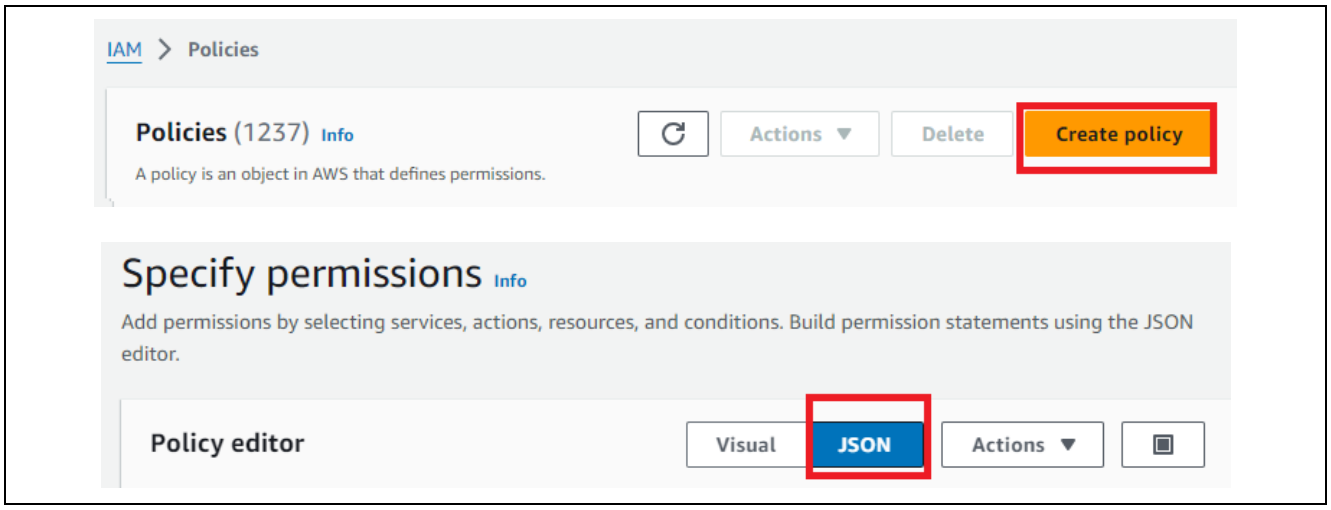


図 6-31　IAM のポリシーを作成

- Policy エディタに以下をペースト> Next

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy",
                "iam:DetachRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:DeletePolicy",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:policy/idt*",
                "arn:aws:iam::*:role/idt*"
            ]
        }
    ]
}
```

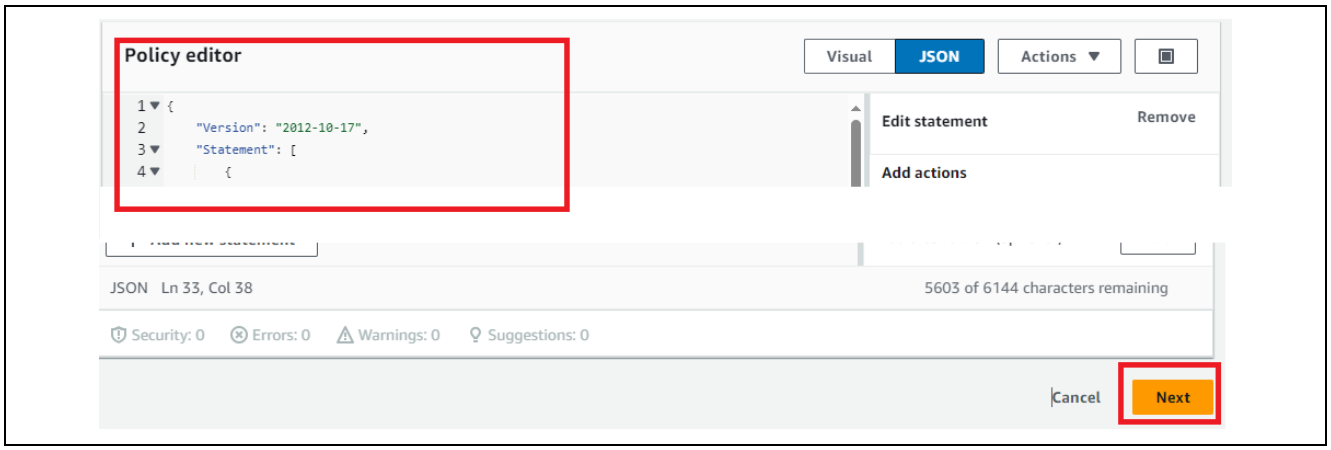


図 6-32　ポリシーエディタで IAM ポリシーを作成

- Policy name: 任意 (例：IDTFreeRTOSIAMPermissions_rl78g23-fpb) > “Create policy”

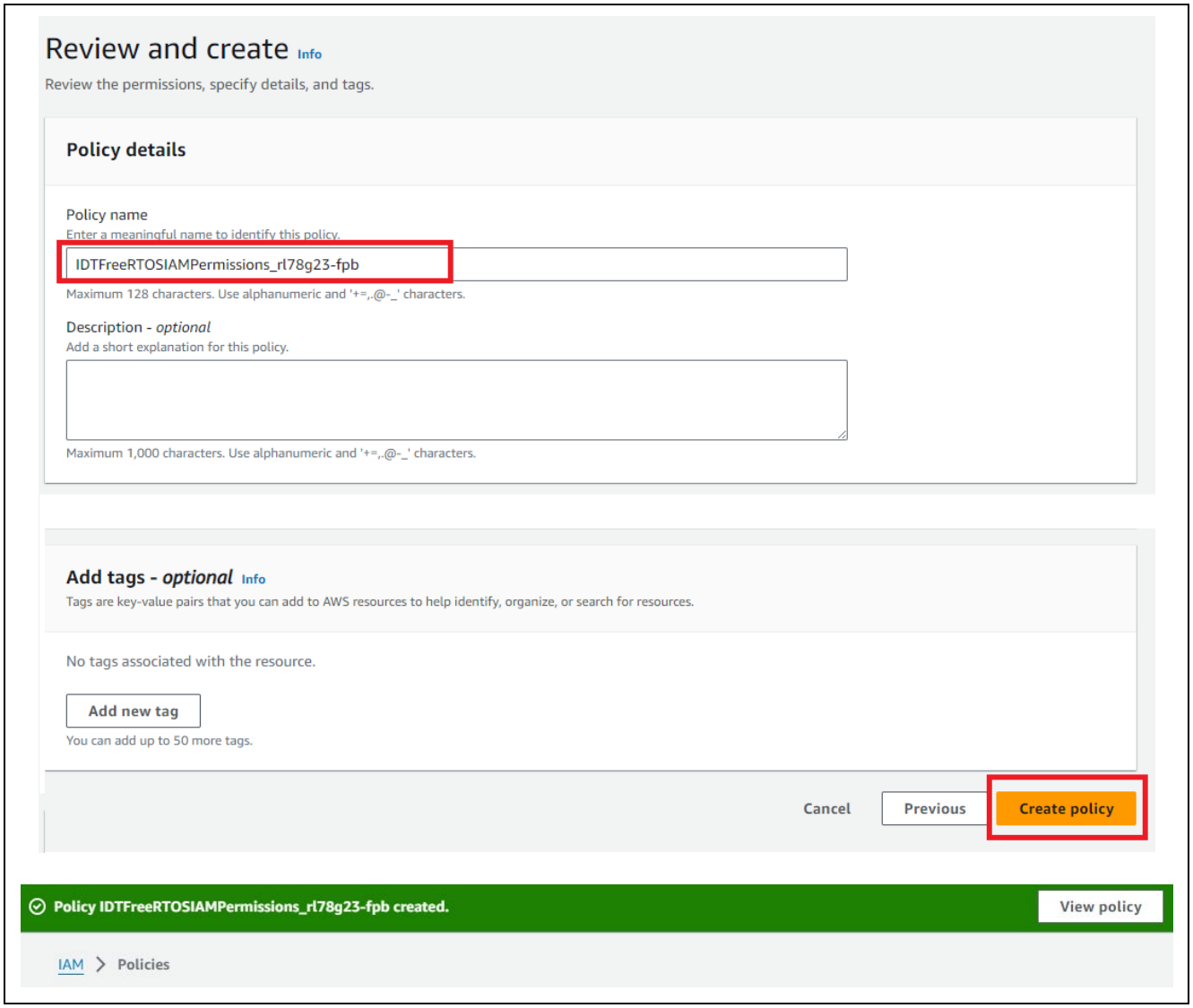


図 6-33　名前 (例：IDTFreeRTOSIAMPermissions_rl78g23-fpb)を付けて IAM ポリシーを保存

② 作成した IAM ポリシーを IAM ユーザにアタッチします。

- IAM > Users > Choose User > Add permissions

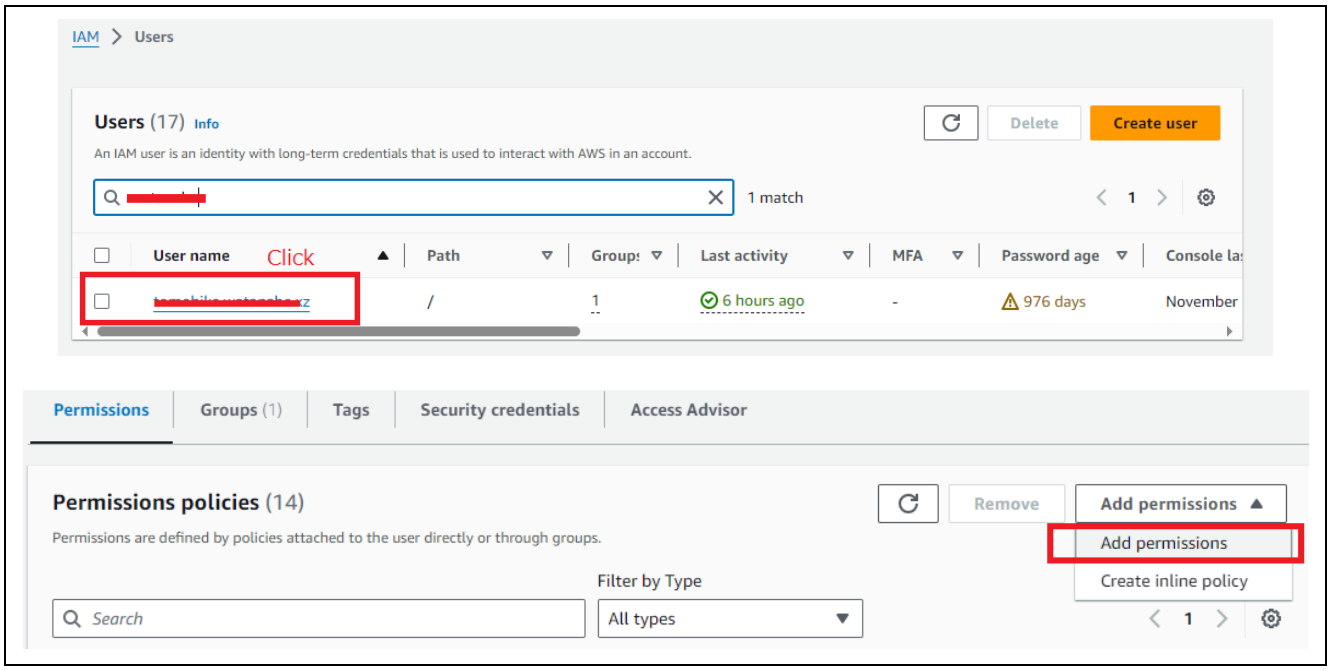


図 6-34　作成した IAM ポリシーをアサインする IAM ユーザを選択

- Permissions options: Attach policies directly
- Policy name:
  - AWSIoTDeviceTesterForFreeRTOSFullAccess
  - 作成した IAM ポリシーの名前 (例：IDTFreeRTOSIAMPermissions_rl78g23-fpb)
- “Next” をクリック > “Add permissions” をクリック

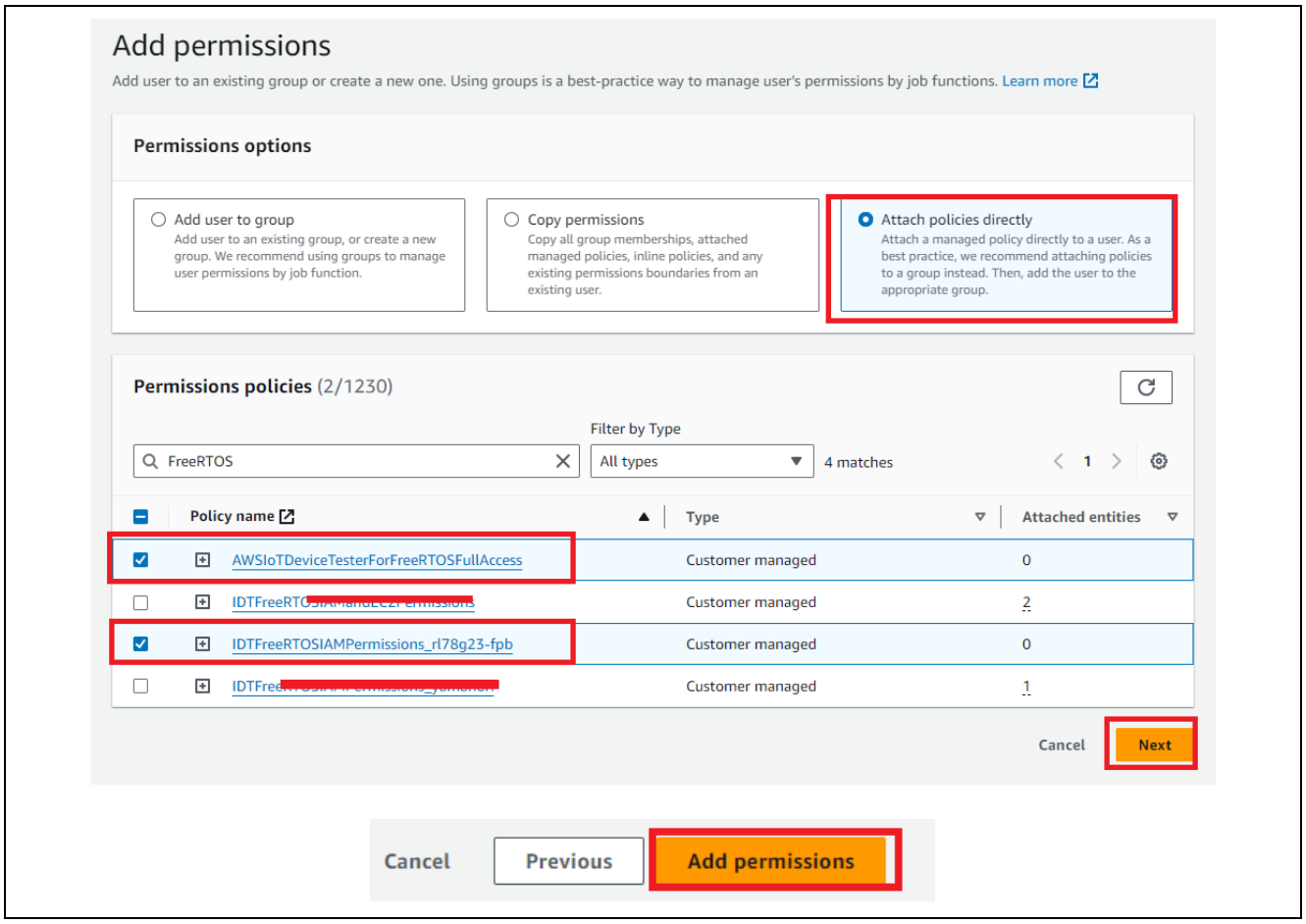


図 6-35　選択した IAM ユーザに権限を追加

6.2　初期イメージの作成

初期イメージとは、ブートローダの MOT ファイルと初期アプリケーションの MOT ファイルを「Renesas Image Generator」で結合した MOT ファイルのことです。

Renesas Image Generator は「RL78/G22,RL78/G23,RL78/G24 ファームウェア アップデート モジュール」に付属するツールです。詳細は上記リンク先のアプリケーションノートを参照してください。

この文書では初期イメージに関連するファイル名を以下とします。

- ブートローダ：boot_loader.mot
- 初期アプリケーション：aws_ryz024a_rl78g23-fpb_ota.mot
- 初期イメージ：initial_image.mot

6.2.1　ブートローダを作成

6.2.1.1　ブートローダプロジェクトのインポート

プロジェクト「boot_loader」を e² studio にインポートします。次の手順でインポートウィザードを開いてください。

File > Import… > Existing Projects into Workspace > Next

次に、プロジェクト「boot_loader」を選択してください。この時、ワークスペースにプロジェクトをコピーしないオプション設定にしてください。その後、Finish ボタンをクリックします。

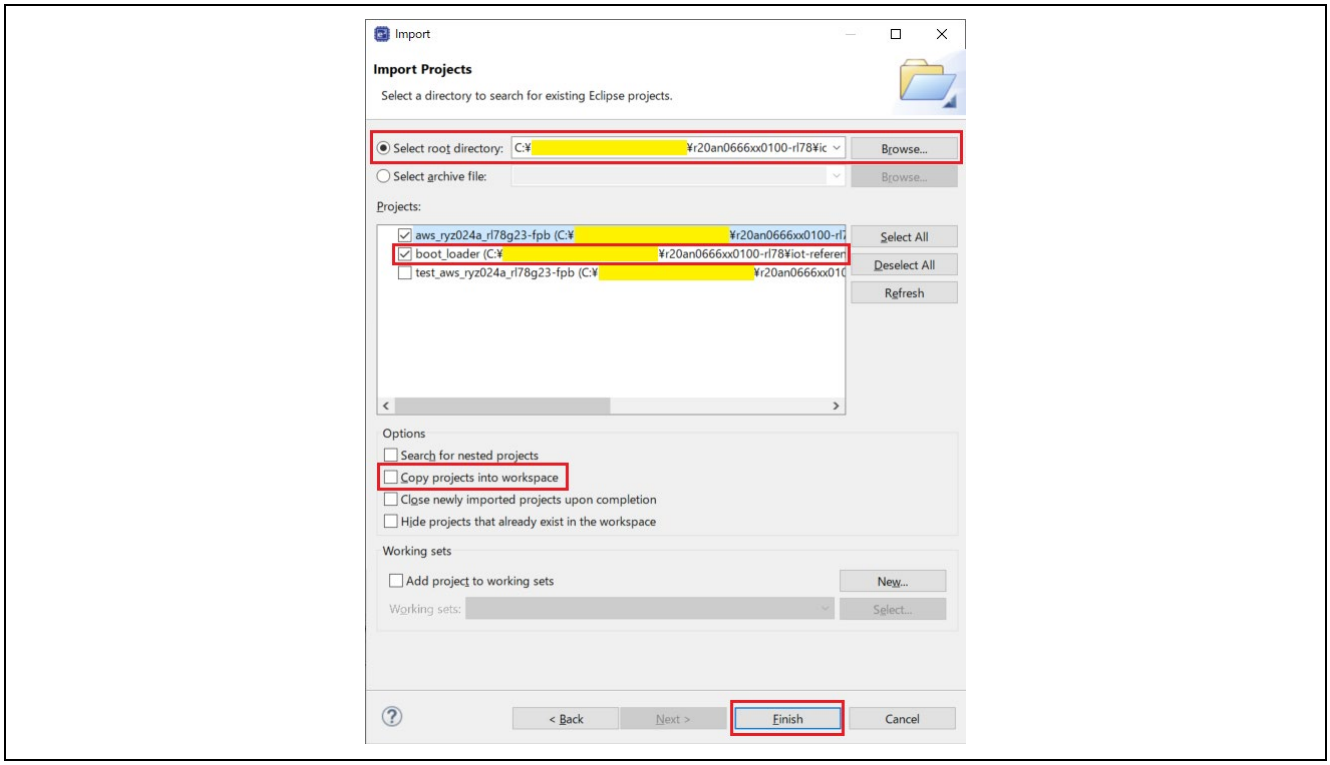


図 6-36　プロジェクト「boot_loader」を選択

プロジェクト・エクスプローラー画面にインポートされたプロジェクトが表示されます。

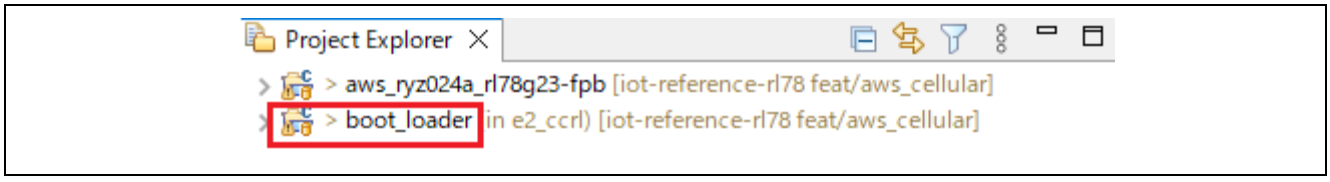


図 6-37　プロジェクト「boot_loader」のインポート完了

6.2.1.2　ファームウェア検証用の鍵をブートローダプロジェクトに追加

① プロジェクト「boot_loader」の code_signer_public_key.h にファームウェア検証用の鍵(secp256r1.publickey)を追加します。

注意事項：各行末に\を付与してください。

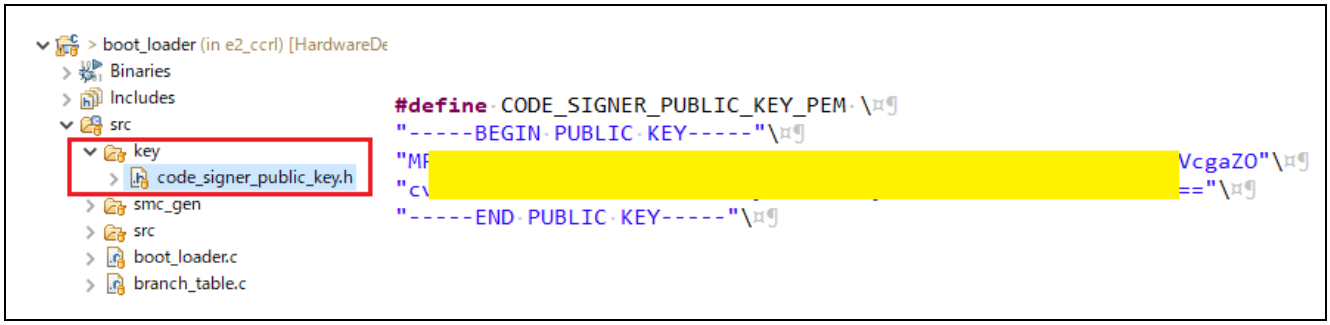


図 6-38　ブートローダにファームウェア検証用の鍵を追加

6.2.1.3　ブートローダプロジェクトのビルド

プロジェクト「boot_loader」をビルドし、MOT ファイルを作成します。
その後、プロジェクトフォルダ直下の HardwareDebug フォルダに boot_loader.mot が生成されていることを確認します。

6.2.2　初期アプリケーションを作成

6.2.2.1　初期アプリケーションのインポート

プロジェクト「aws_ryz024a_rl78g23-fpb」を e² studio にインポートします。次の手順でインポートウィザードを開いてください。

File > Import… > Existing Projects into Workspace > Next

次に、プロジェクト「aws_ryz024a_rl78g23-fpb」を選択してください。この時、ワークスペースにプロジェクトをコピーしないオプション設定にしてください。その後、Finish ボタンをクリックします。

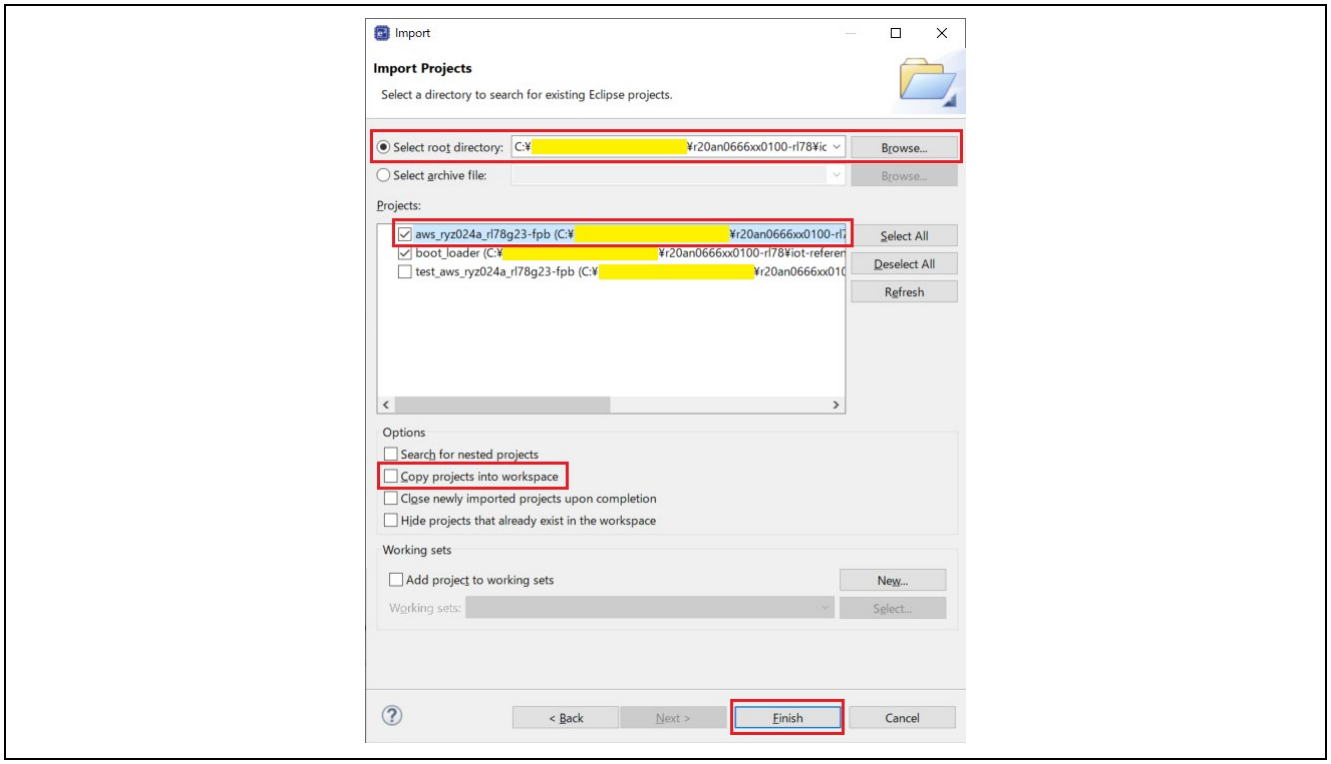


図 6-39　プロジェクト「aws_ryz024a_rl78g23-fpb」を選択

プロジェクト・エクスプローラー画面にインポートされたプロジェクトが表示されます。

図 6-40　プロジェクト「aws_ryz024a_rl78g23-fpb」のインポート完了

6.2.2.2　初期アプリケーションのビルド構成を設定

プロジェクト「aws_ryz024a_rl78g23-fpb」のビルド構成を“HardwareDebug_OTA”に設定します。

Build Configurations > Set Active > Select “HardwareDebug_OTA”

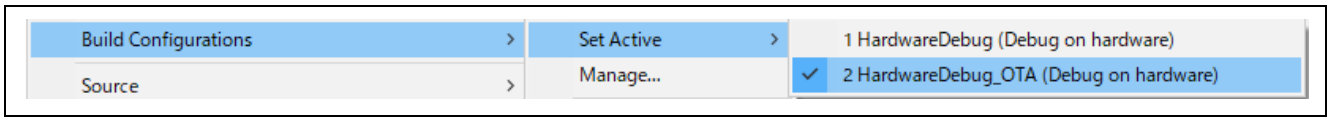


図 6-41　ビルド構成“HardwareDebug_OTA”をアクティブにする

6.2.2.3　ファームウェア検証用の鍵を初期アプリケーションに追加

プロジェクト「aws_ryz024a_rl78g23-fpb」の code_signer_public_key.h にファームウェア検証用の鍵(secp256r1.publickey)を追加します。

注意事項：各行末に\を付与してください。

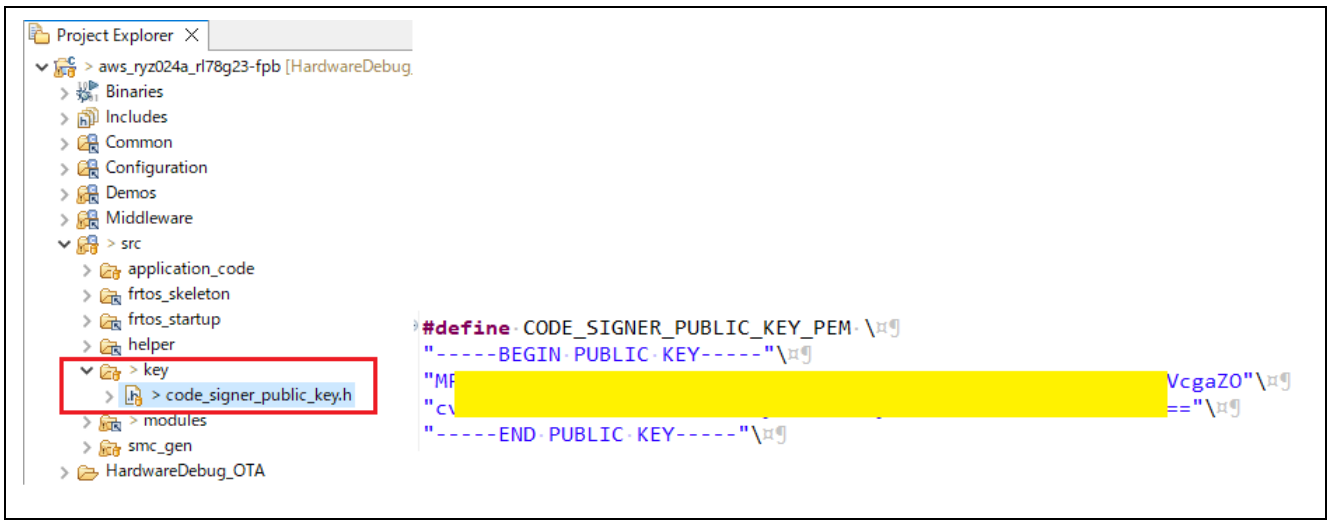


図 6-42　初期アプリケーションにファームウェア検証用の鍵を追加

6.2.2.4　初期アプリケーションのビルド

プロジェクト「aws_ryz024a_rl78g23-fpb」をビルドし、MOT ファイルを作成します。
その後、プロジェクトフォルダ直下の HardwareDebug_OTA フォルダに aws_ryz024a_rl78g23-fpb_ota.mot が生成されていることを確認します。

6.2.3　Renesas Image Generator で初期イメージを生成

ブートローダと初期アプリケーションを Renesas Image Generator で結合して初期イメージを生成します。

①　Renesas Image Generator と同じフォルダに以下のファイルを格納します。

- ブートローダ：boot_loader.mot
- 初期アプリケーション：aws_ryz024a_rl78g23-fpb_ota.mot
- 初期アプリケーション検証用の秘密鍵：secp256r1.privatekey

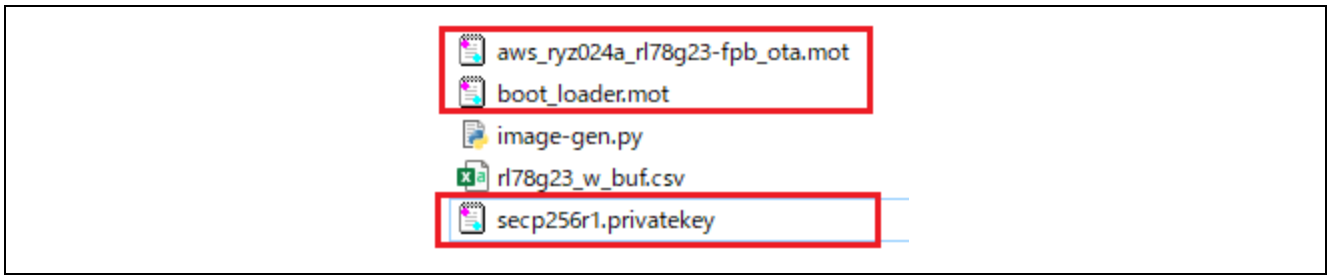


図 6-43　Renesas Image Generator と同じフォルダに必要なファイルを格納

②　以下のコマンドを実行し初期イメージを生成します。

```
python image-gen.py -iup .\aws_ryz024a_rl78g23-fpb_ota.mot -ibp
boot_loader.mot -o initial_image -ip .\RL78_G23_ImageGenerator_PRM.csv
```

③　初期イメージ (initial_image.mot)が生成されていることを確認してください。

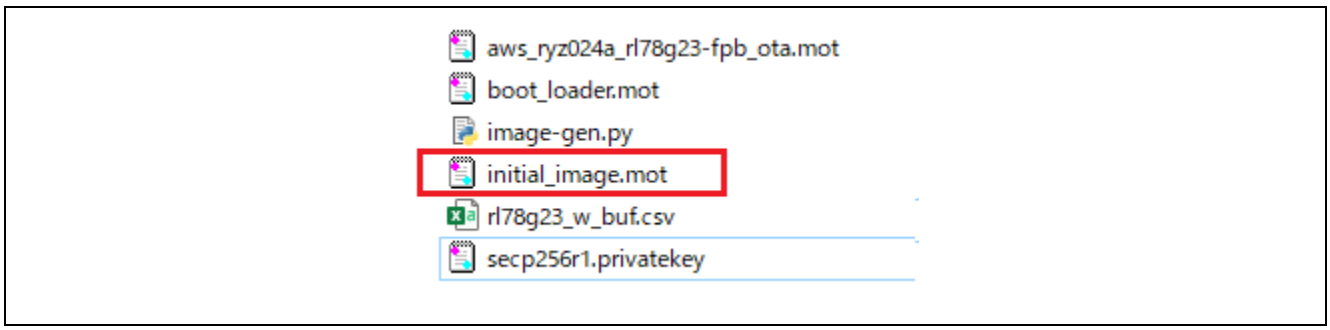


図 6-44　Renesas Image Generator と同じフォルダに初期イメージが生成される

6.3　更新イメージの作成

更新イメージとは、更新アプリケーションの MOT ファイルを「Renesas Image Generator」で変換したバイナリ形式 (拡張子 rsu)の更新用ファームウェアのことです。更新イメージのフォーマットについては「RL78/G22,RL78/G23,RL78/G24 ファームウェア アップデート モジュール」を参照してください。

この文書では更新イメージに関連するファイル名を以下とします。

- 更新アプリケーション：aws_ryz024a_rl78g23-fpb_ota_093.mot
- 更新イメージ：aws_ryz024a_rl78g23-fpb_ota_093.rsu

6.3.1　更新アプリケーションを作成

6.3.1.1　アプリケーションのソースコードを変更

更新アプリケーションを作成するために、
iot-reference-rl78\Configuration\rl78g23-fpb\ota\cellular\frtos_config\demo_config.h の「APP_VERSION_BUILD」マクロの定義を 2 から 3 に変更します。

```
iot-reference-rl78\Configuration\rl78g23-
fpb\ota\cellular\frtos_config\demo_config.h

/**
 * @brief Build version of the firmware.
 *
 * This is used in the OTA demo to set the appFirmwareVersion variable that
is
 * declared in the ota_appversion32.h file in the OTA library.
 */
#ifndef APP_VERSION_BUILD
    #define APP_VERSION_BUILD    3
#endif
```

6.3.1.2　更新アプリケーションのビルド

aws_ryz024a_rl78g23-fpb プロジェクトをビルドし、MOT ファイルを作成します。
その後、プロジェクトフォルダ直下の HardwareDebug_OTA フォルダに aws_ryz024a_rl78g23-fpb_ota.mot が上書き生成されていることを確認します。

6.3.1.3　更新アプリケーションの MOT ファイルをリネーム

aws_ryz024a_rl78g23-fpb_ota.mot を aws_ryz024a_rl78g23-fpb_ota_093.mot にリネームします。

6.3.2　Renesas Image Generator で更新イメージを生成

更新アプリケーションを Renesas Image Generator で更新イメージに変換します。

①　Renesas Image Generator と同じフォルダに以下のファイルを格納します。

- 更新アプリケーションの MOT ファイル：aws_ryz024a_rl78g23-fpb_ota_093.mot
- 更新アプリケーション検証用の秘密鍵：secp256r1.privatekey

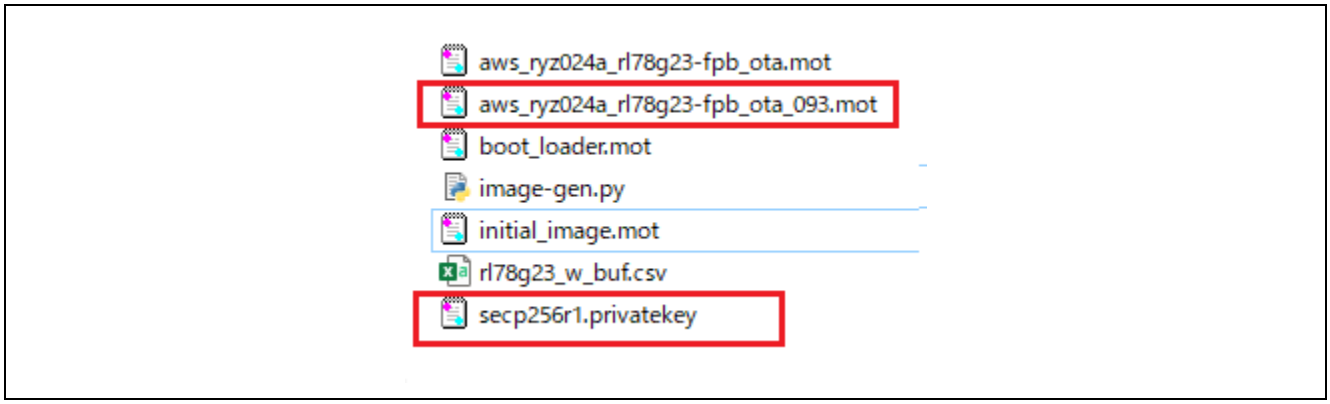


図 6-45　Renesas Image Generator と同じフォルダに必要なファイルを格納

②　以下のコマンドを実行し RSU 形式の更新イメージ aws_ryz024a_rl78g23-fpb_ota_093.rsu を生成します。

```
python image-gen.py -iup .\aws_ryz024a_rl78g23-fpb_ota_093.mot -o
aws_ryz024a_rl78g23-fpb_ota_093 -ip .\RL78_G23_ImageGenerator_PRM.csv -vt
ecdsa -ff RTOS
```

③　aws_ryz024a_rl78g23-fpb_ota_093.rsu が生成されていることを確認します。

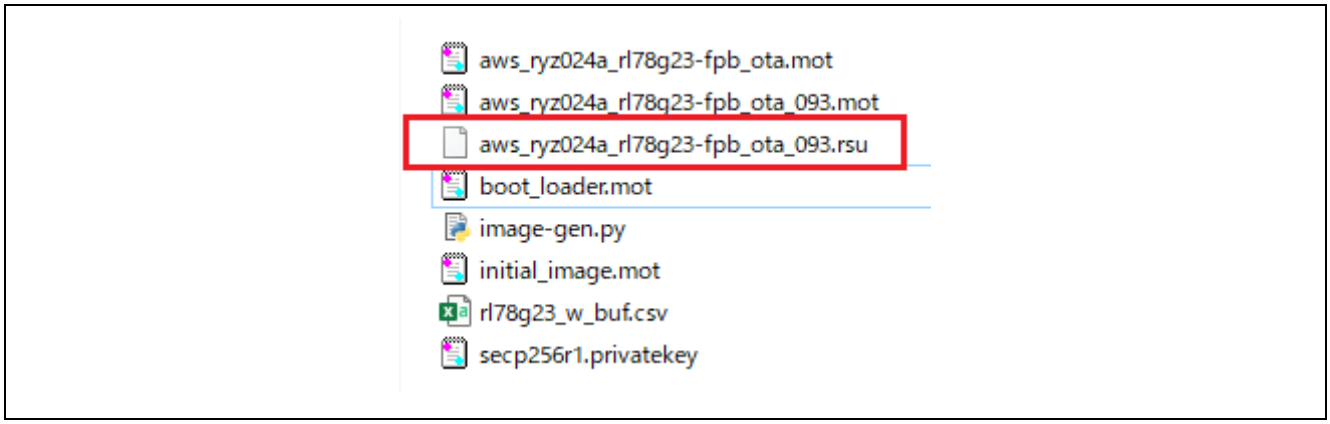


図 6-46　Renesas Image Generator と同じフォルダに更新イメージが生成される

6.4　デモプロジェクトの実行

デモプロジェクト(OTA)の実行手順を示します。

6.4.1　初期イメージ (initial_image.mot)をボードに書き込む

①　初期イメージ (initial_image.mot)を書き込みます。

書き込み方法は「7 章 Renesas Flash Programmer の使用方法」を参照してください。

②　書き込みが終了するとデモプロジェクトが起動します。

③　ターミナルで初期アプリケーション (バージョン 0.9.2)が起動していることを確認します。

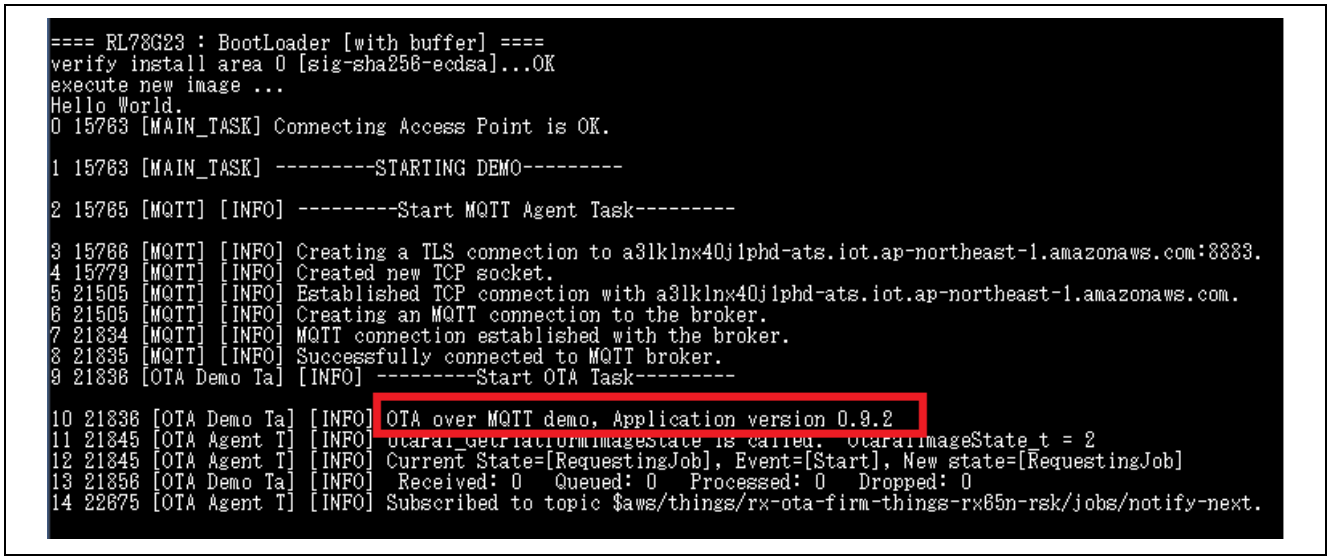


図 6-47　初期アプリケーション (バージョン 0.9.2)起動

6.4.2　更新イメージ (aws_ryz024a_rl78g23-fpb_ota_093.rsu)を OTA ジョブに登録

① AWS IoT > Manage > Remote actions > Jobs > “Create job” をクリック

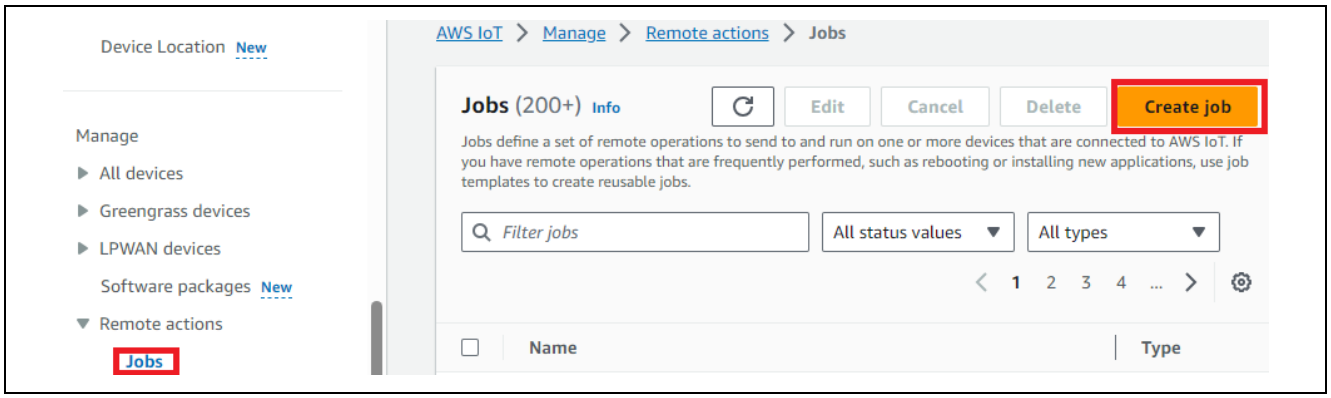


図 6-48　Jobs

② Check “Create FreeRTOS OTA update job” > “Next” をクリック

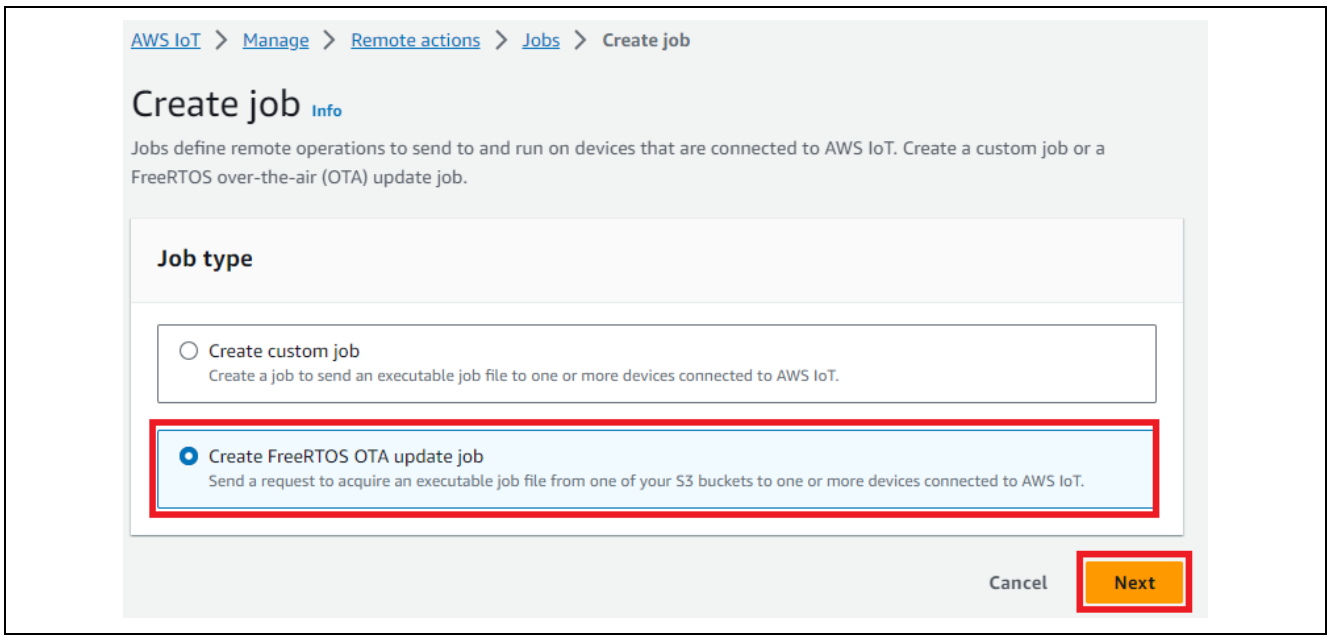


図 6-49　Create Job

③　Step1: OTA job properties

- Job name: 任意

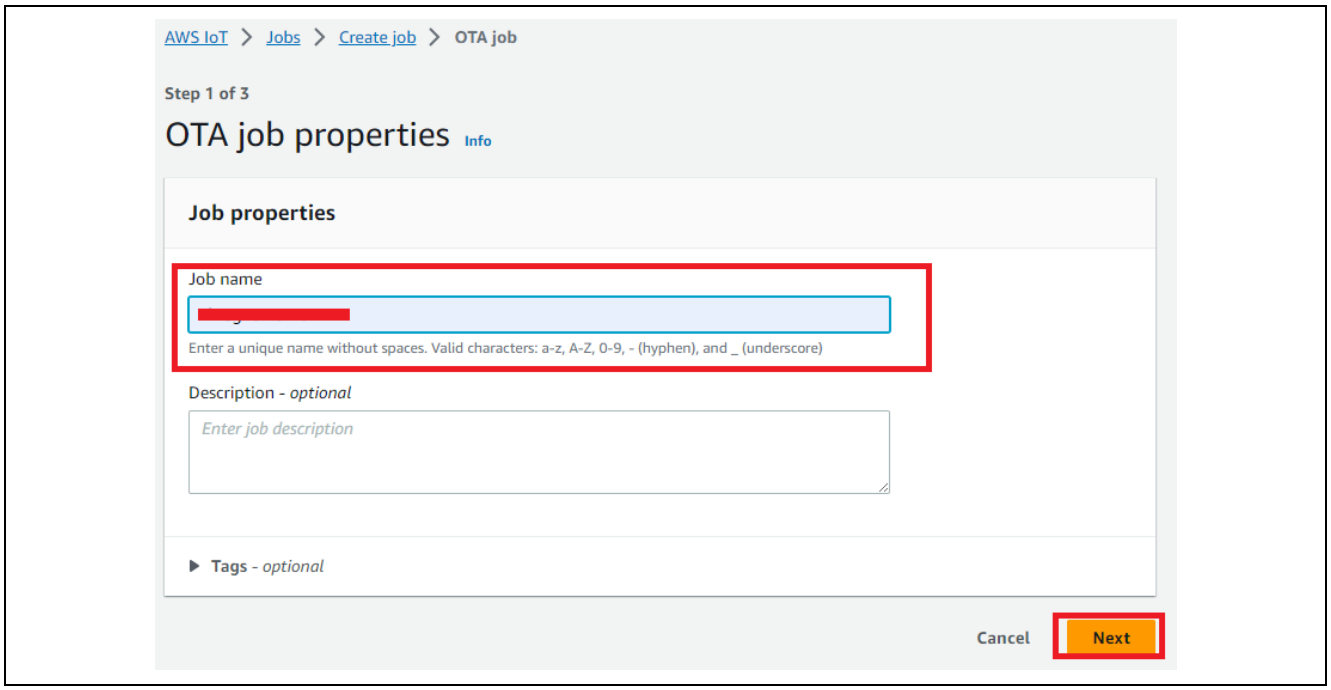


図 6-50　Step1: OTA job properties

④　Step2: OTA file configuration > Devices

- Devices to update: aws_clientcredential.h の”モノの名前”

```
#define clientcredentialIOT_THING_NAME          "YOUR_THING_NAME
```

- Select the protocol for file transfer: MQTT

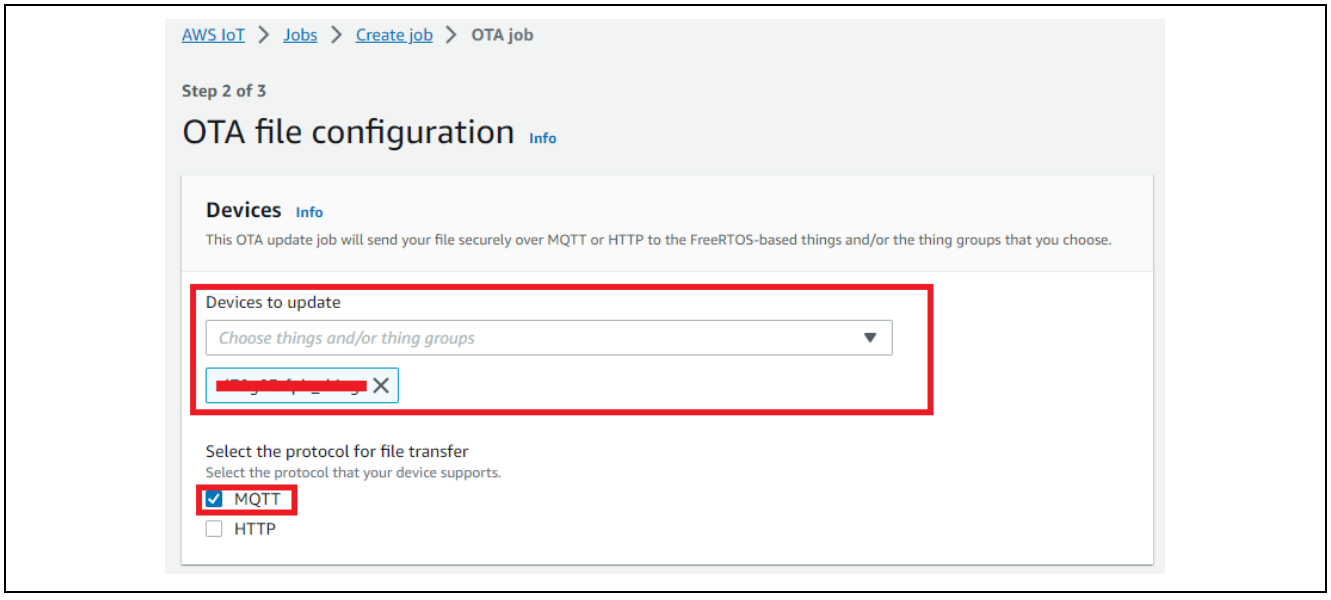


図 6-51　Step2: OTA file configuration > Devices

⑤　Step2: OTA file configurations > File

- Sign and choose your file: Sign a new file for me.

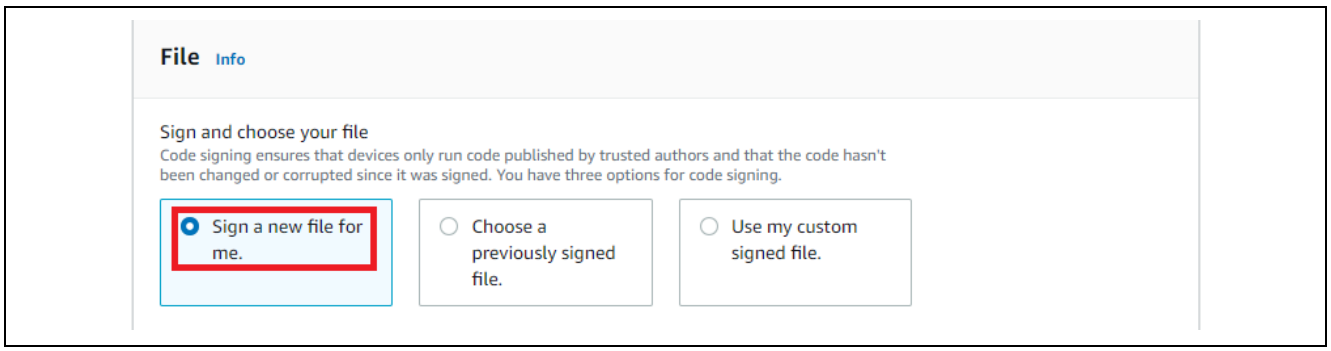


図 6-52　Step2: OTA file configurations > File (1)

- Code signing profile: “Create new profile” をクリック

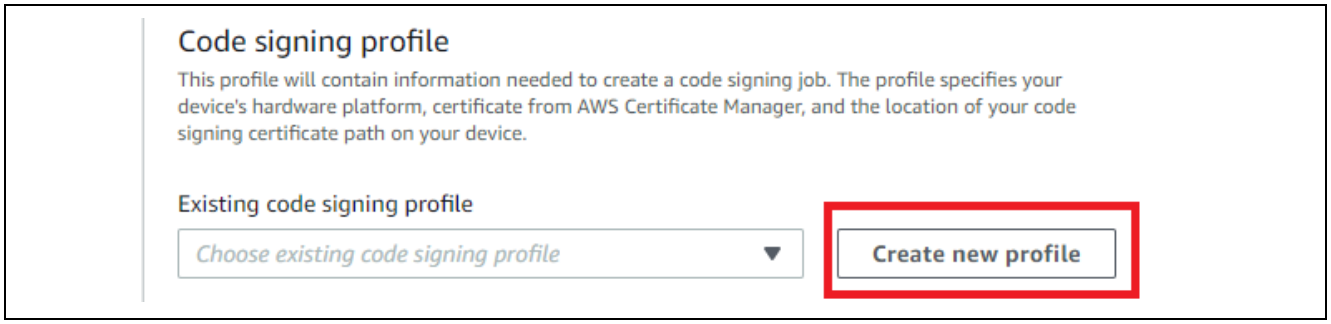


図 6-53　Step2: OTA file configurations > File (2)

- Create a code signing profile.
  - Profile name: 任意 (例：rl78g23_fpb_ota_cert)
  - Device hardware platform: Windows Simulator
  - Code signing certificate: “Import new code signing certificate”
  - Certificate body: secp256r1.crt
  - Certificate private key: secp256r1.privatekey
  - Certificate chain - optional: ca.crt
  - Path name of code signing certificate on device: 任意

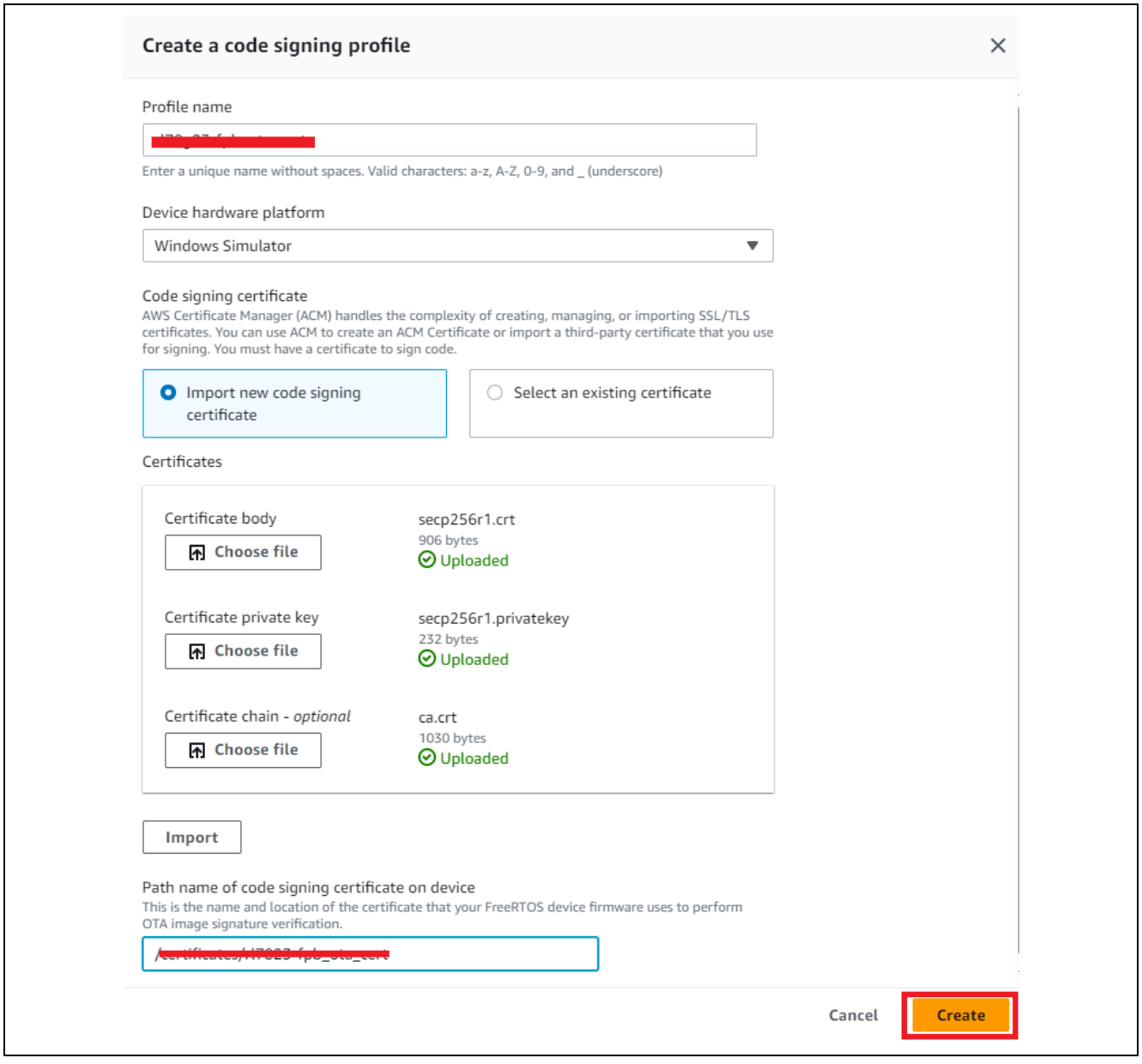


図 6-54　Create a code signing profile

- File > “Upload a new file.” > “Choose file” > aws_ryz024a_rl78g23-fpb_ota_093.rsu

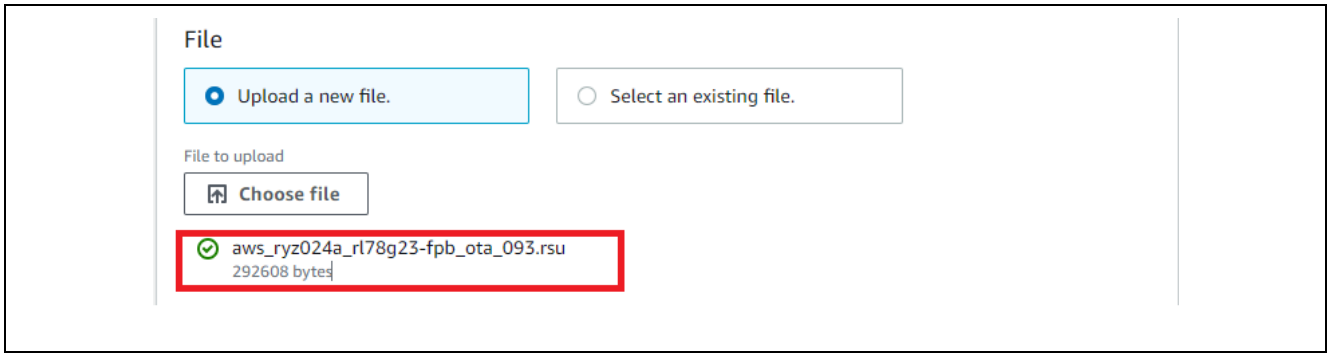


図 6-55　Upload a new file > aws_ryz024a_rl78g23-fpb_ota_093.rsu

- File upload location in S3: 作成済のバケットを指定 (「6.1.3.1 章　Amazon S3 バケットの作成」で指定した Bucket name)
- Path name of file on device: 任意

図 6-56　File upload location in S3

⑥　Step2: OTA file configurations > IAM role

- Role: 作成済のロールを指定 (「6.1.3.2 章　OTA 更新用サービスロールの作成」で指定した Role name)

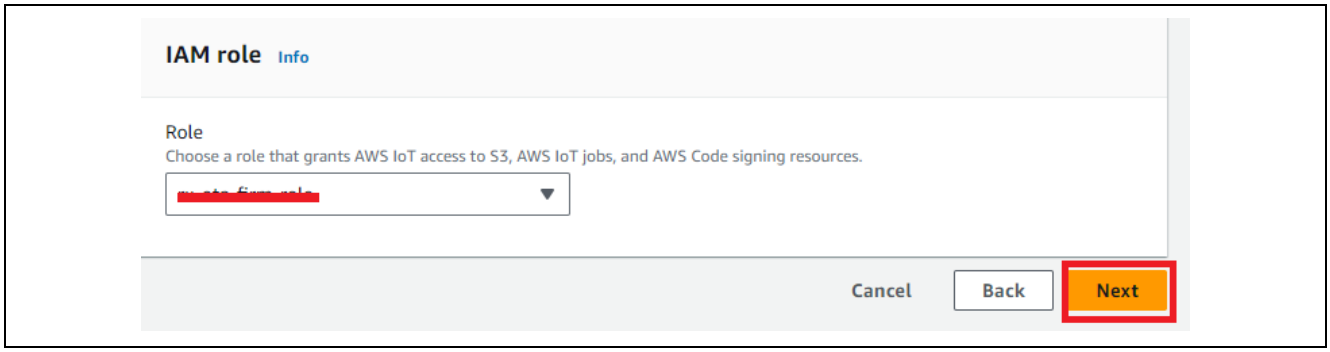


図 6-57　Step2: OTA file configurations > IAM role

⑦　Step3: OTA job configuration

- Job run type: Your job will complete after deploying to the devices and groups that you chose (snapshot)

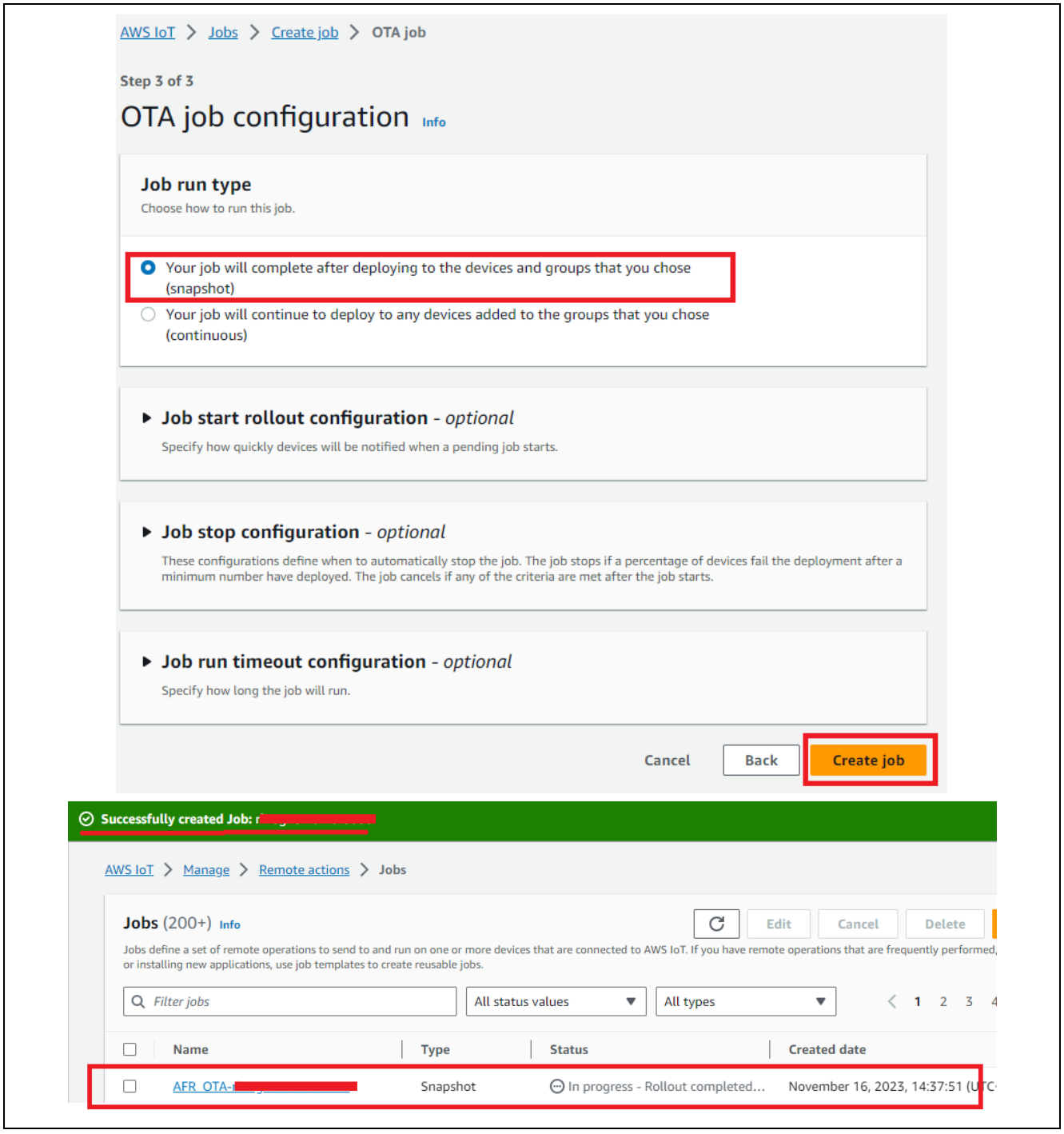


図 6-58　Step3: OTA job configuration

⑧　しばらく待つと更新イメージをマイコンボードに書き込むログがターミナルに出力されます。

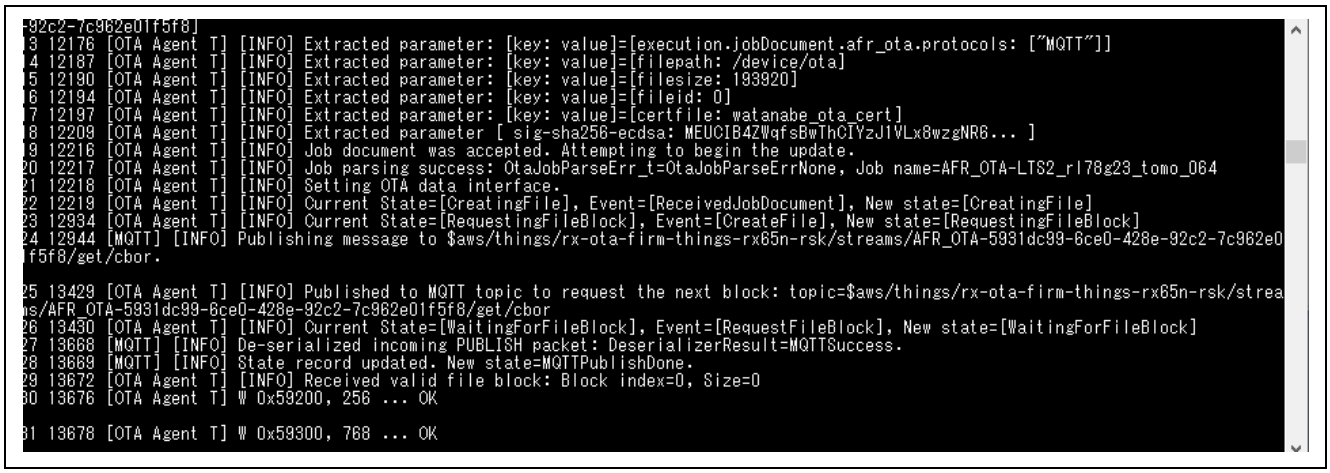


図 6-59　更新イメージをマイコンボードに書き込む

⑨　書き込みが終了すると更新イメージ (バージョン 0.9.3)が起動します。

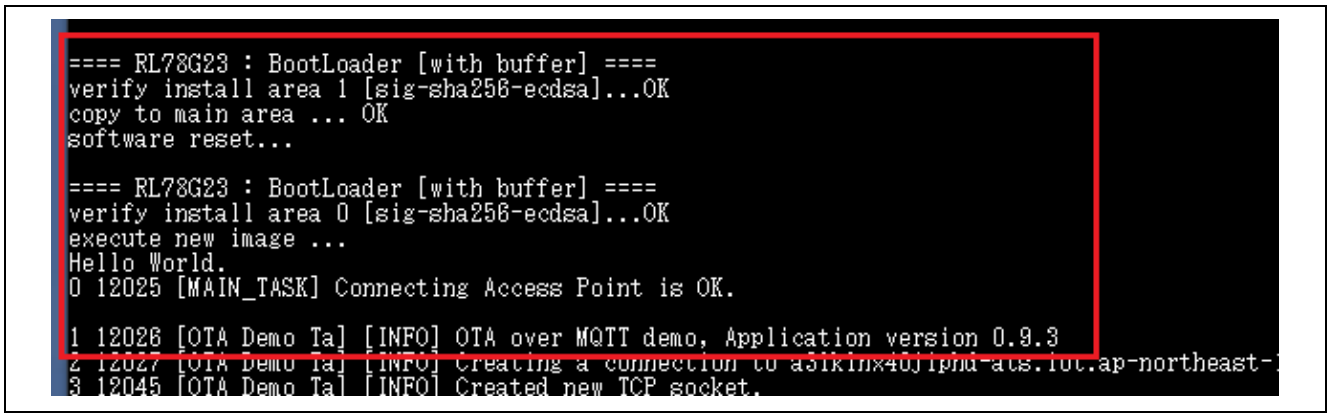


図 6-60　書き込み終了後、更新イメージ (バージョン 0.9.3)が起動

6.5　初期アプリケーションのデバッグ方法

初期アプリケーションを e² studio から起動してデバッグする手順を示します。
この手順ではブートローダを使用しないため、ダウンロードした更新イメージを起動することはできません。

① ブートローダを使用しない設定に変更します。

プロジェクト「aws_ryz024a_rl78g23-fpb」の“USE_BOOTLOADER_V2”マクロを 0 に設定して”Apply and Close”を押します。

- Configuration: HardwareDebug_OTA
- Languages: GNU C
- USE_BOOTLOADER_V2: 0

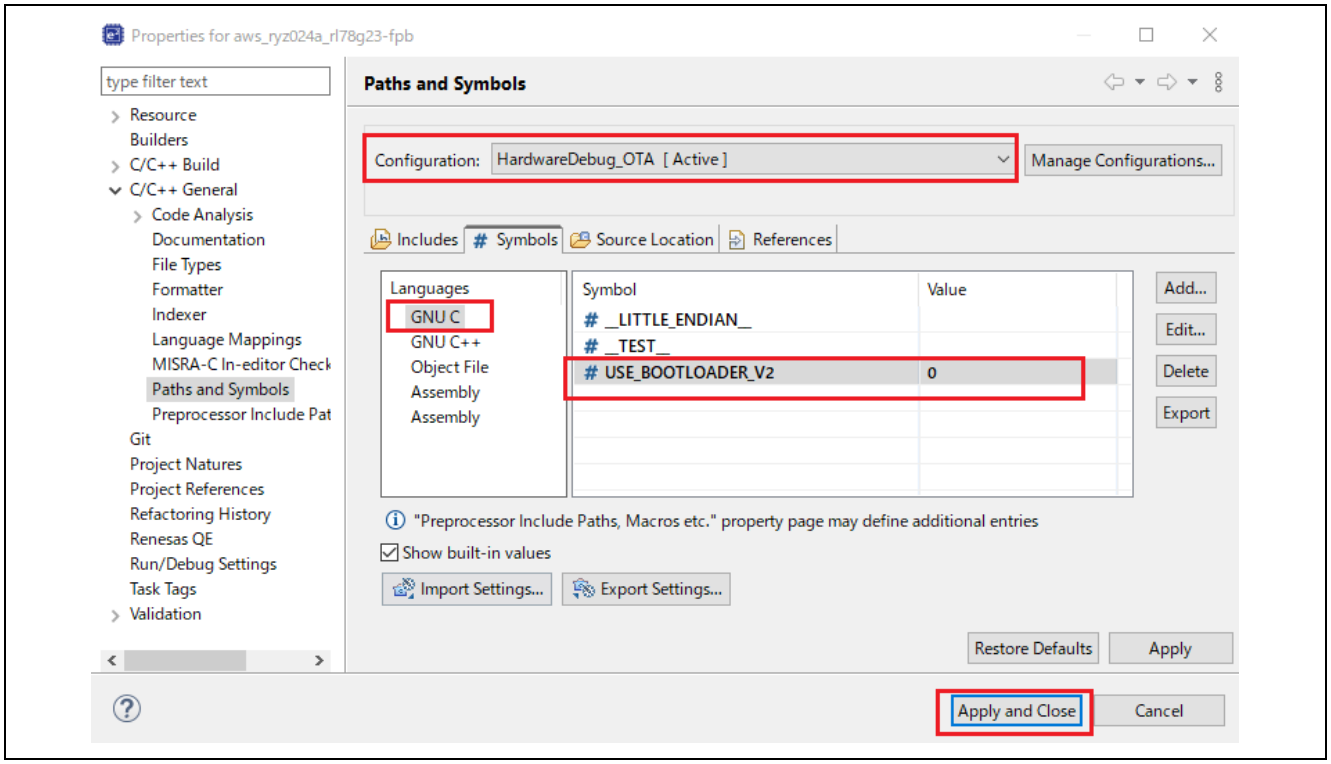


図 6-61　“USE_BOOTLOADER_V2”マクロを 0 に設定

② プロジェクト「aws_ryz024a_rl78g23-fpb」をビルドします。

③ デバッグを開始します。

「8 章　デバッグ手順」を参照してください。

7.　Renesas Flash Programmer の使用方法

Renesas Flash Programmer を用いてマイコンボードに MOT を書き込む手順を示します。

7.1　COM port を使用する場合

COM port 経由で MOT ファイルの書き込む方法を示します。

7.1.1　ジャンパピン設定

J15: 1-2 Short、J16: 1-2 Short、J19: 1-2 Short にしてください。回路変更を実施していない場合、この操作は不要です。

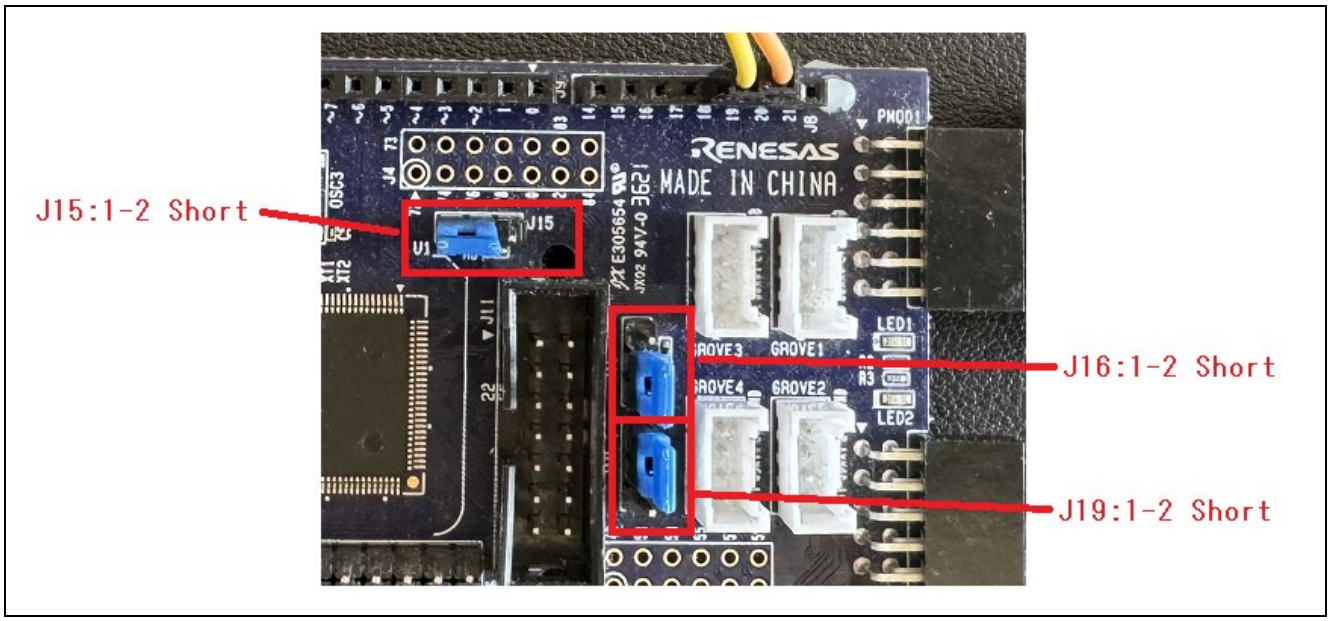


図 7-1　**COM port デバッグ使用時設定 (部品面)**

7.1.2　マイコンボードへ電源を供給

USB ケーブルを接続してマイコンボードへ電源を供給します。

7.1.3　プロジェクトを新規作成しマイコンボードに接続

①　File > New project

- Microcontroller: RL78/G2x
- Project Name: 任意 (例：rl78g23-fpb)
- Project Folder: 任意
- Tool: COM port
- Interface: 2 wire UART
- Tool Details…: COM port の番号
- “Connect” をクリック

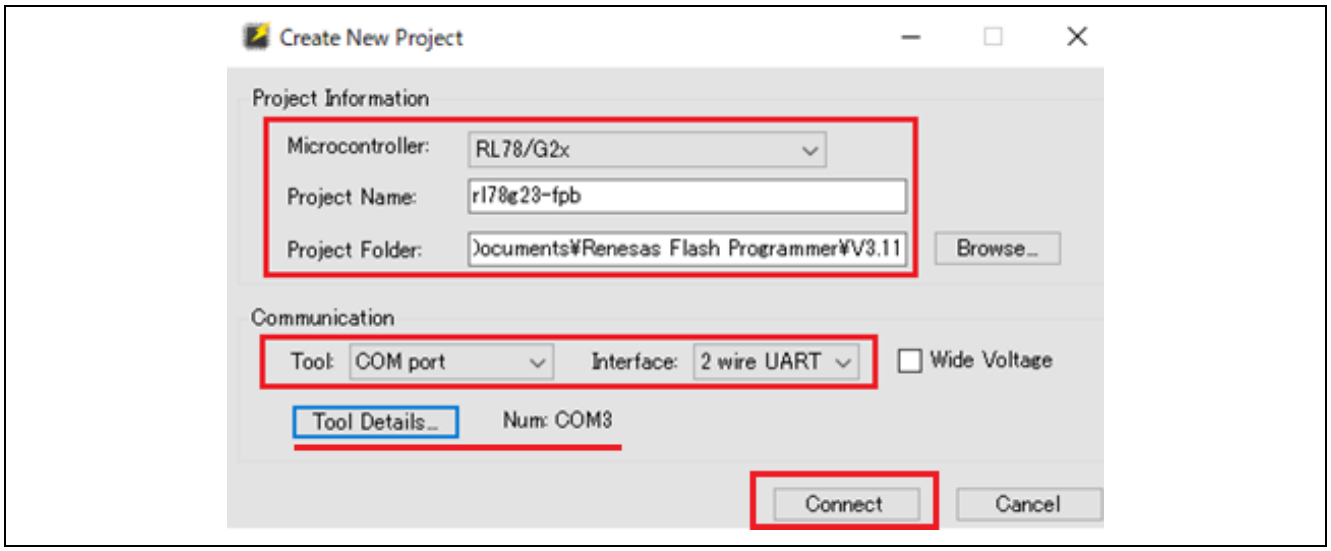


図 7-2　プロジェクトを新規作成してマイコンボードに接続

② 以下の画面が出ていれば接続成功です。

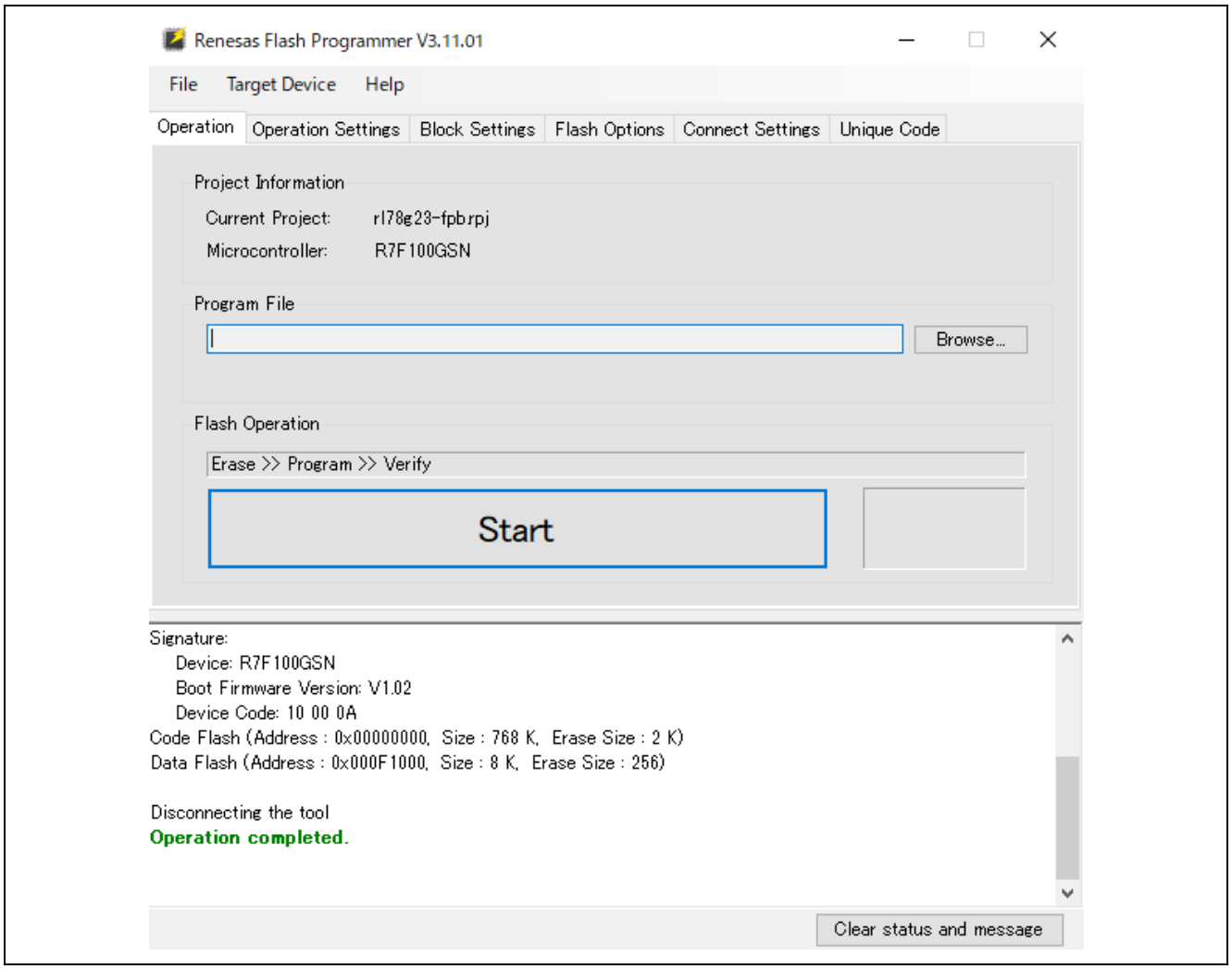


図 7-3　Operation completed (Connect)

7.1.4　マイコンボードに MOT ファイルを書き込む

①　Program File 欄に書き込む MOT ファイルのパスを入力し”Start”を押します。

- Program File: 書き込む MOT ファイル (例：initial_image.mot、aws_ryz024a_rl78g23-fpb.mot)
- “Start” をクリック

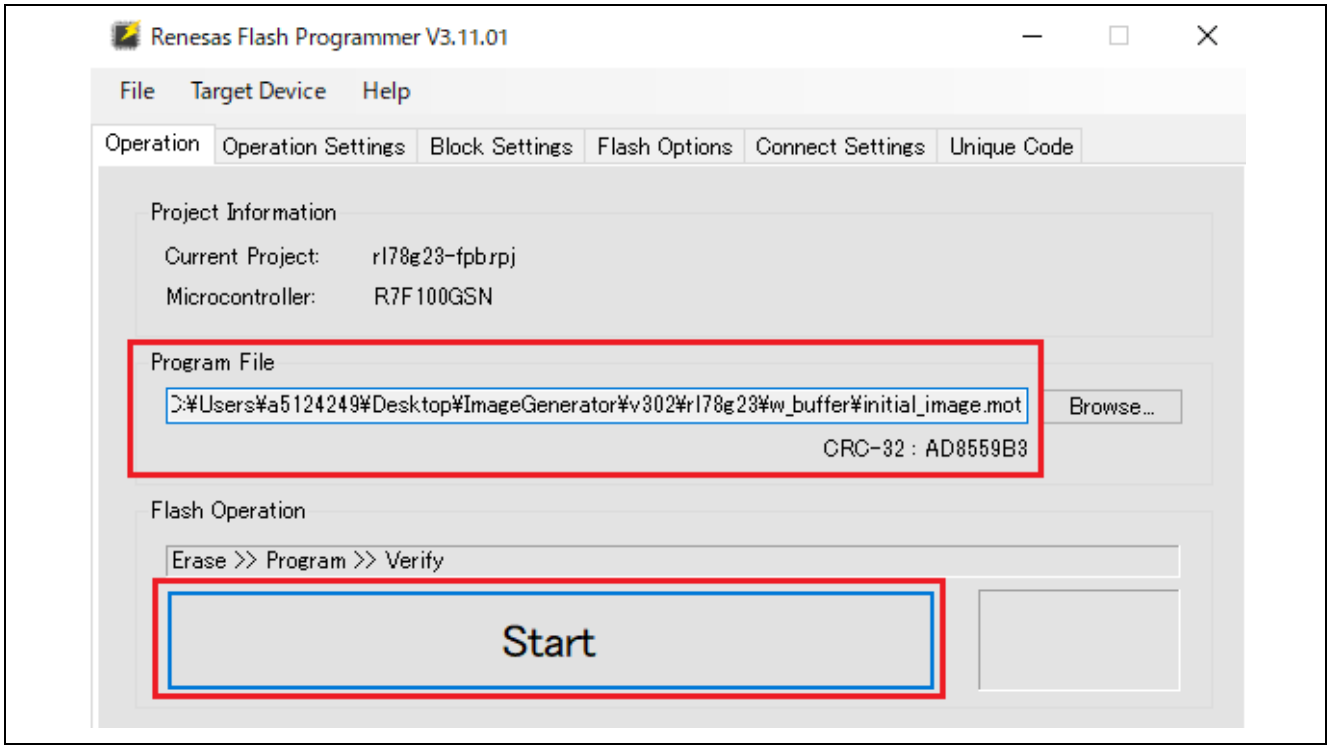


図 7-4　マイコンボードに MOT ファイルを書き込む

②　書き込みが成功していることを確認します。

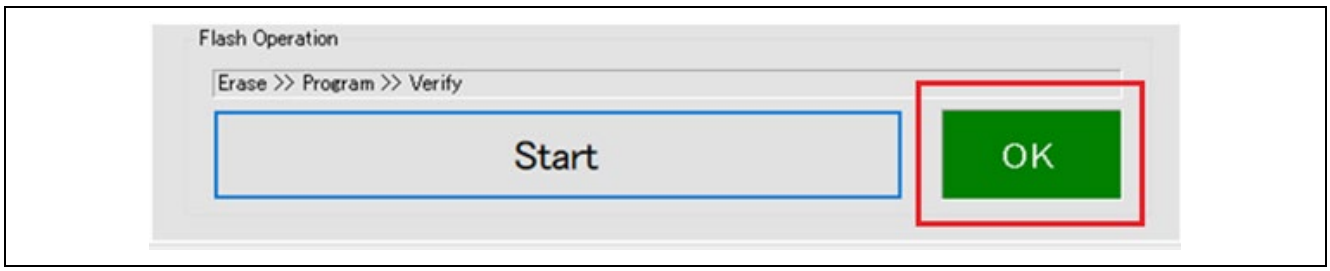


図 7-5　書き込み成功

7.2　エミュレータを使用する場合

エミュレータ経由で MOT ファイルの書き込む方法を示します。

7.2.1　ジャンパピン設定、コネクタ実装、パターンカット

14pin コネクタ（J11）は、ルネサスエレクトロニクス製のプログラミング機能付きオンチップ・デバッグ・エミュレータの E2 エミュレータまたは E2 エミュレータ Lite との接続用コネクタです（コネクタ部品は未実装です）。エミュレータを使用して、評価 MCU のプログラミングおよびデバッグを行います。
エミュレータを接続する場合には下図の回路変更が必要となります。詳細は RL78/G23-128p Fast Prototyping Board ユーザーズマニュアル Rev.1.00 5.20 章を参照してください。

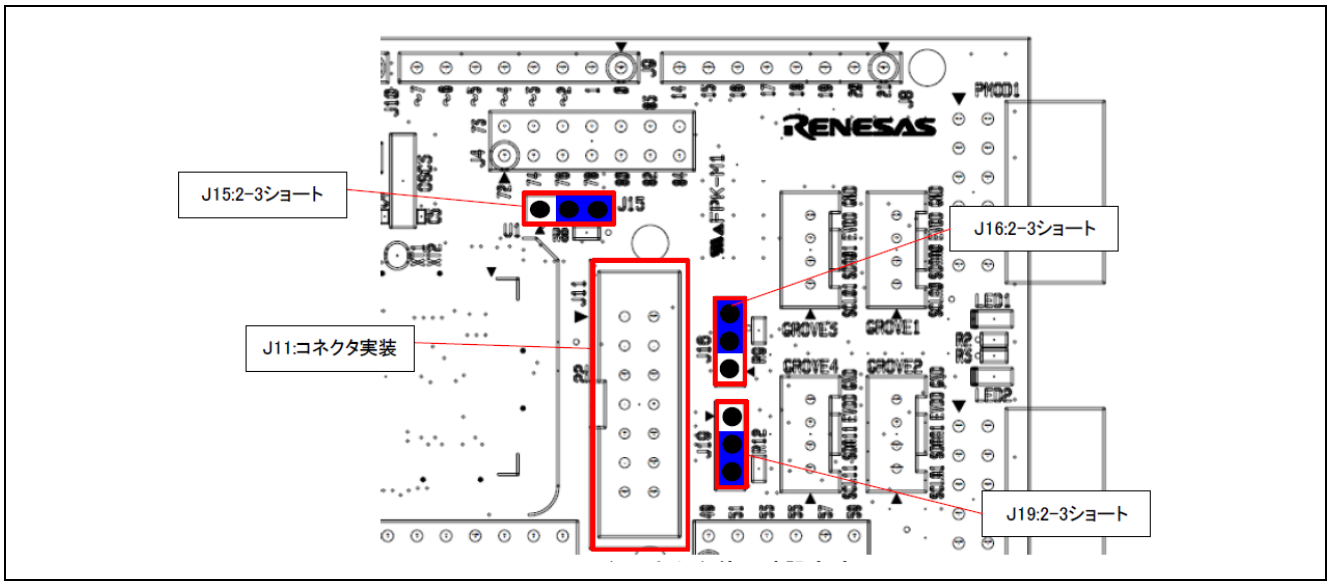


図 7-6　エミュレータコネクタ使用時設定 (部品面)

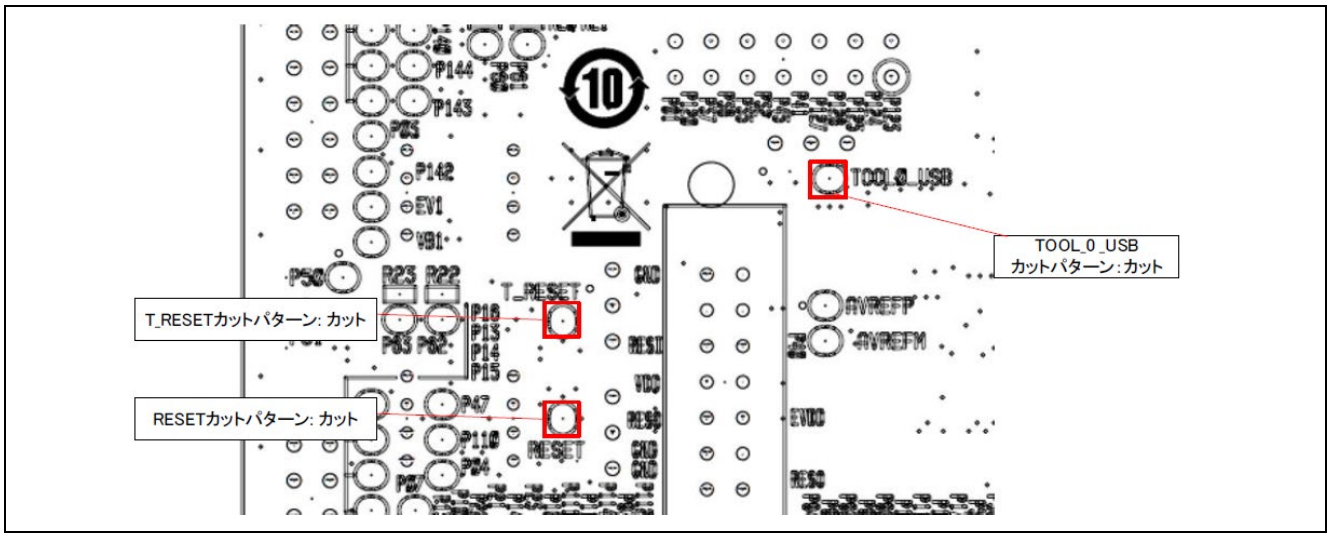


図 7-7　エミュレータコネクタ使用時設定 (ハンダ面)

また、エミュレータの使用方法については、「E1/E20/E2 エミュレータ, E2 エミュレータ Lite ユーザーズマニュアル別冊 (RL78 接続時の注意事項)」（R20UT1994）を参照してください。

7.2.2　マイコンボードへ電源を供給

USB ケーブルを接続してマイコンボードへ電源を供給します。

7.2.3　プロジェクトを新規作成しマイコンボードに接続

①　File > New project

- Microcontroller: RL78/G2x
- Project Name: 任意 (例：rl78g23-fpb)
- Project Folder: 任意
- Tool: E2 emulator
- “Connect” をクリック

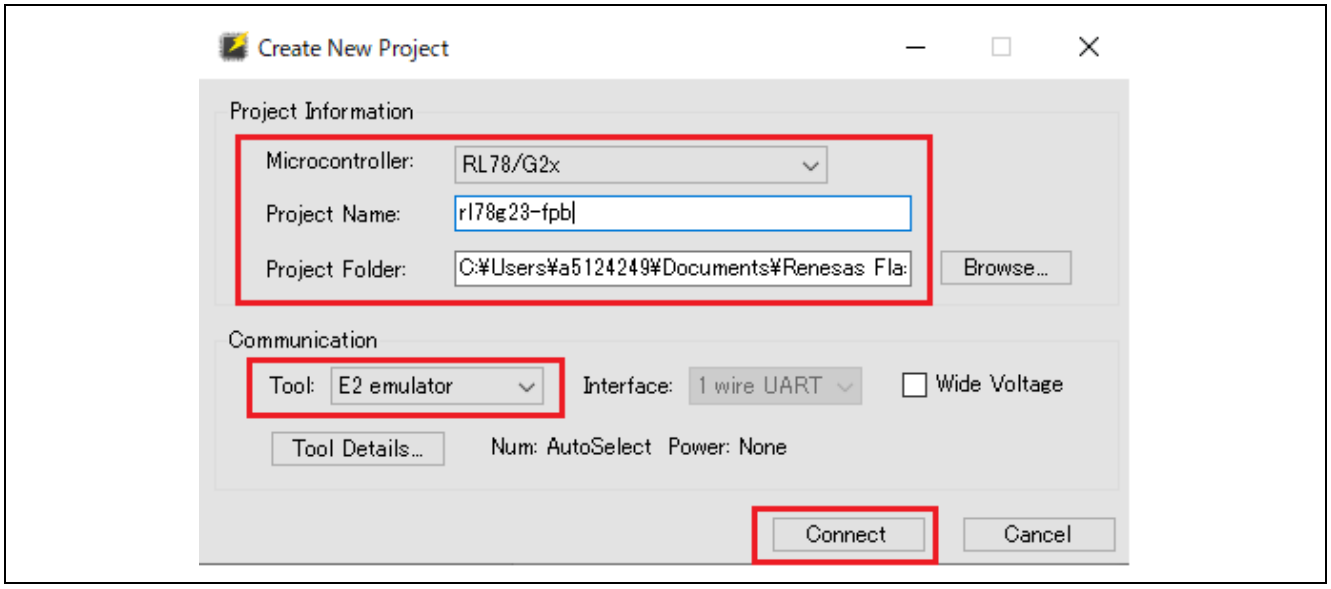


図 7-8　プロジェクトを新規作成してマイコンボードに接続

②　以下の画面が出ていれば接続成功です。

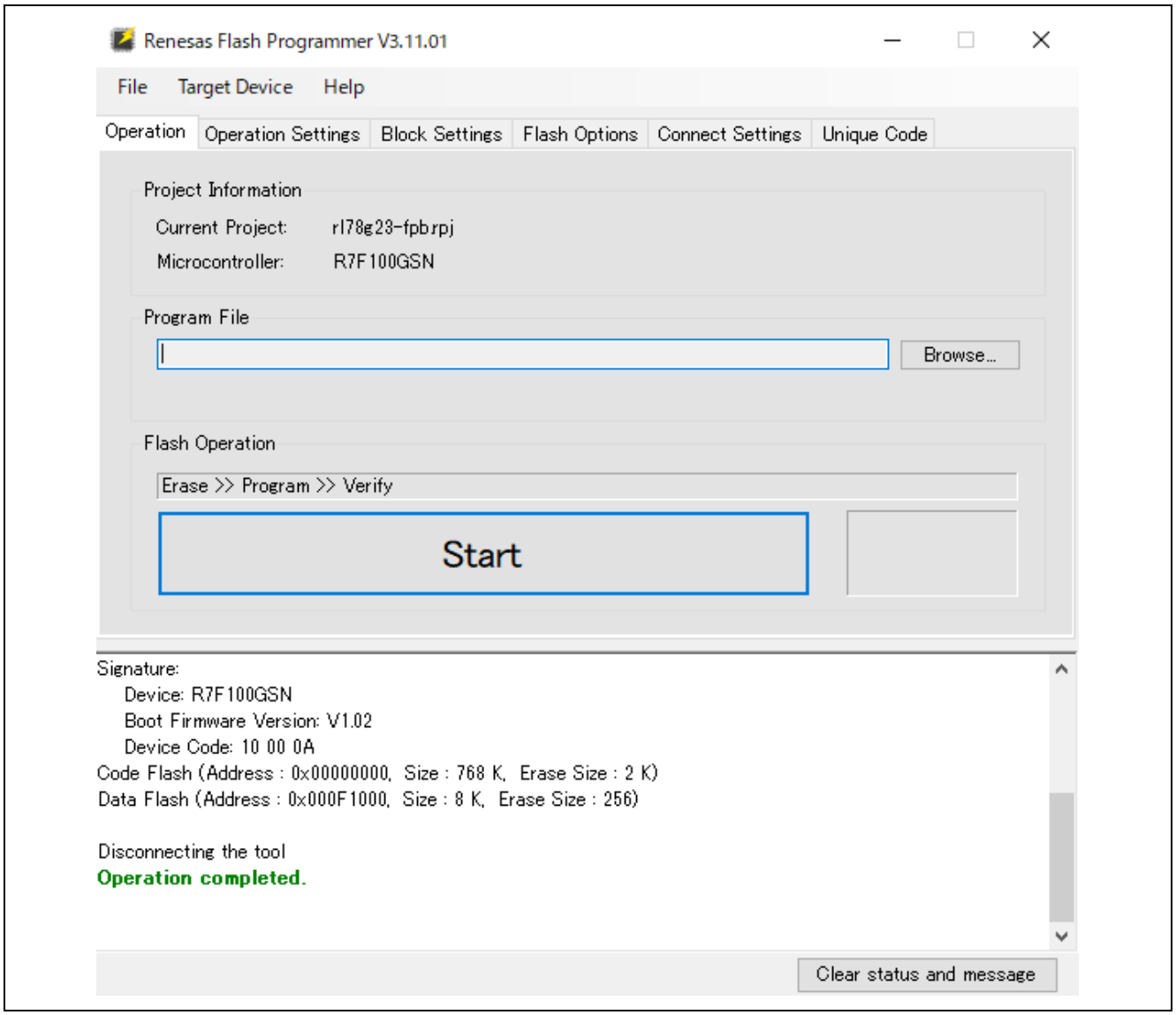


図 7-9　Operation completed(Connect)

7.2.4　マイコンボードに MOT ファイルを書き込む

「7.1.4 章　マイコンボードに MOT ファイルを書き込む」を参照してください。

8.　デバッグ手順

8.1　COM port を使用する場合

COM port を用いてデバッグする方法を示します。

8.1.1　ジャンパピン設定

「7.1.1 章　ジャンパピン設定」を参照してください。

8.1.2　マイコンボードへ電源を供給

マイコンボードと PC を USB ケーブルで接続します。

8.1.3　デバッグコンフィグレーション

デバッグしたいコンフィグレーションを選択します。

- Debug Configurations > Renesas GDB Hardware Debugging

—　デモプロジェクト(PubSub)の場合：aws_ryz024a_rl78g23-fpb HardwareDebug

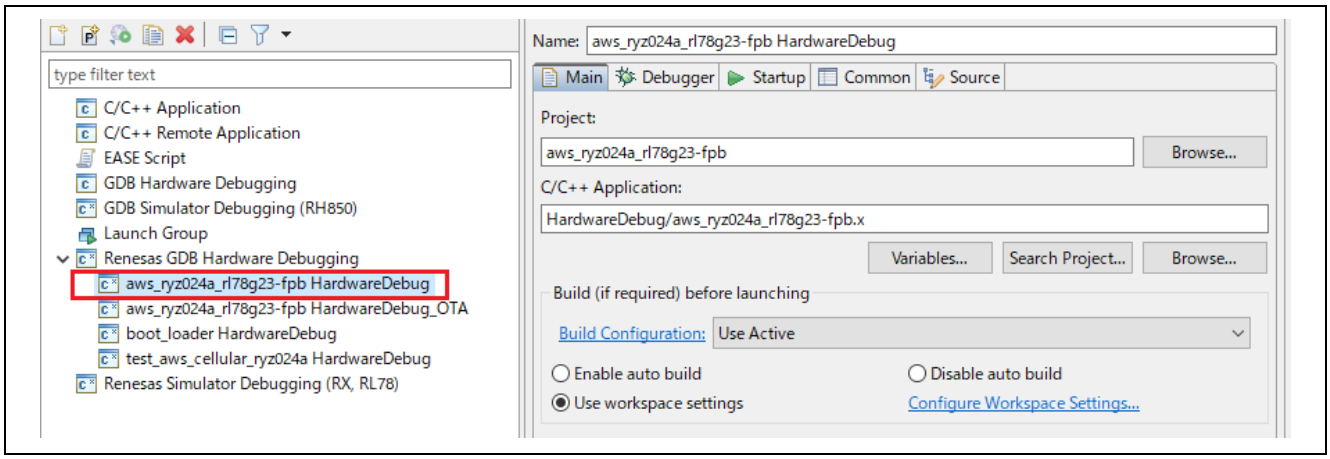


図 8-1　Debug Configurations of Project(PubSub)

—　デモプロジェクト(OTA)の場合：aws_ryz024a_rl78g23-fpb HardwareDebug_OTA

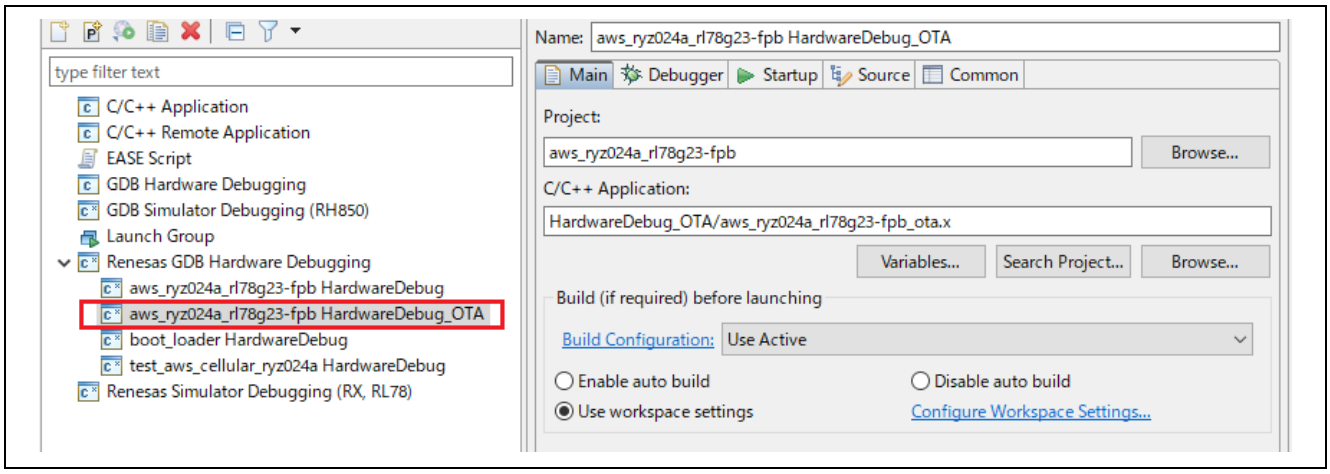


図 8-2　Debug Configurations of Project(OTA)

8.1.4　デバッガ設定

”Debugger” tab を選択する。

- Debug hardware: COM Port (RL78)

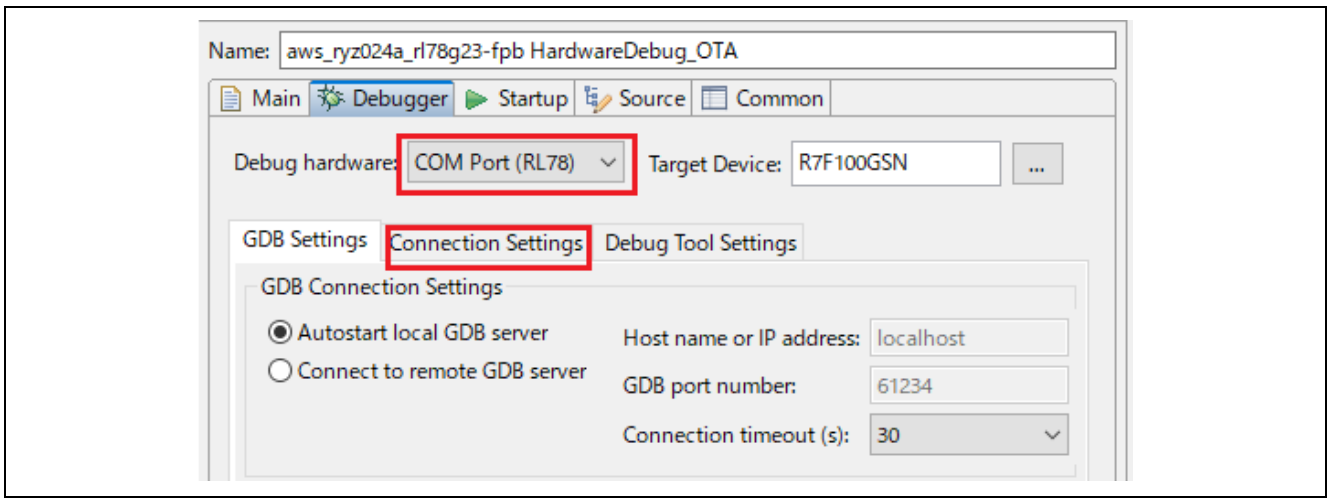


図 8-3　Debug hardware: COM Port (RL78)

“Connection Settings” tab > Connection with Target Borad を選択する。

- COM Port: COMxx
- Reset control pin: DTR

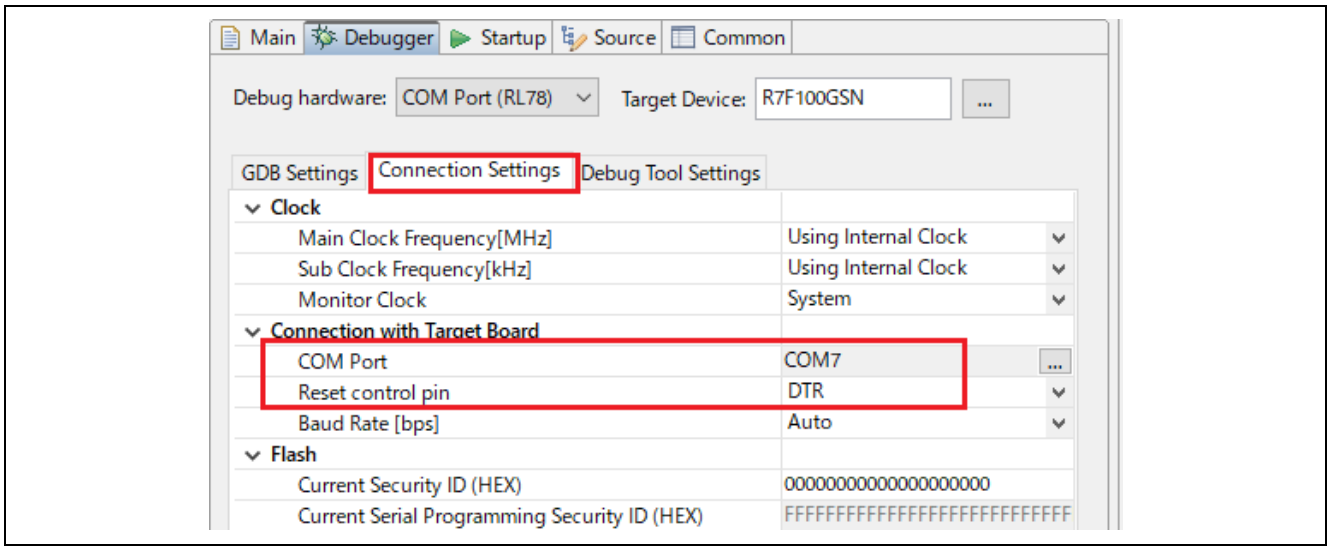
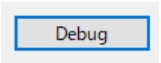


図 8-4　Connection Settings for using COM port



をクリックしてデバッグを開始する。

8.2　エミュレータを使用する場合

E2 ミュレータ Lite を用いてデバッグする方法を示します。

8.2.1　コネクタ実装、ジャンパピン設定、パターンカット

「7.2.1 章　ジャンパピン設定、コネクタ実装、パターンカット」を参照してください。

8.2.2　エミュレータをマイコンボードに接続

図の通りにエミュレータを接続します。

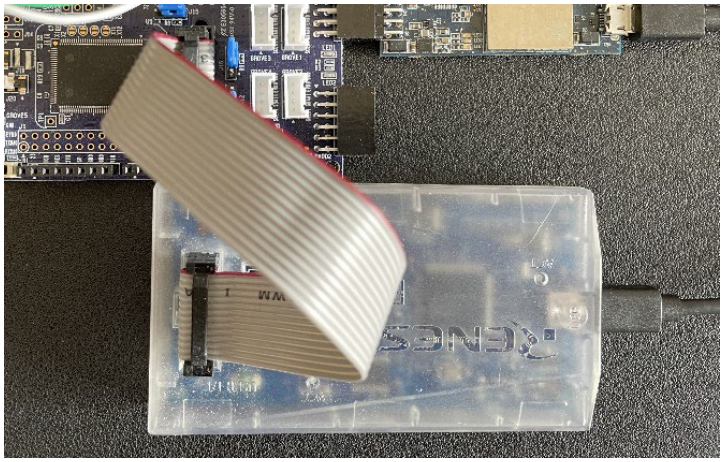


図 8-5　エミュレータをマイコンボードに接続

8.2.3　デバッグコンフィグレーション

デバッグしたいコンフィグレーションを選択します。

- Debug Configurations > Renesas GDB Hardware Debugging
  - ―　デモプロジェクト(PubSub)の場合：aws_ryz024a_rl78g23-fpb HardwareDebug

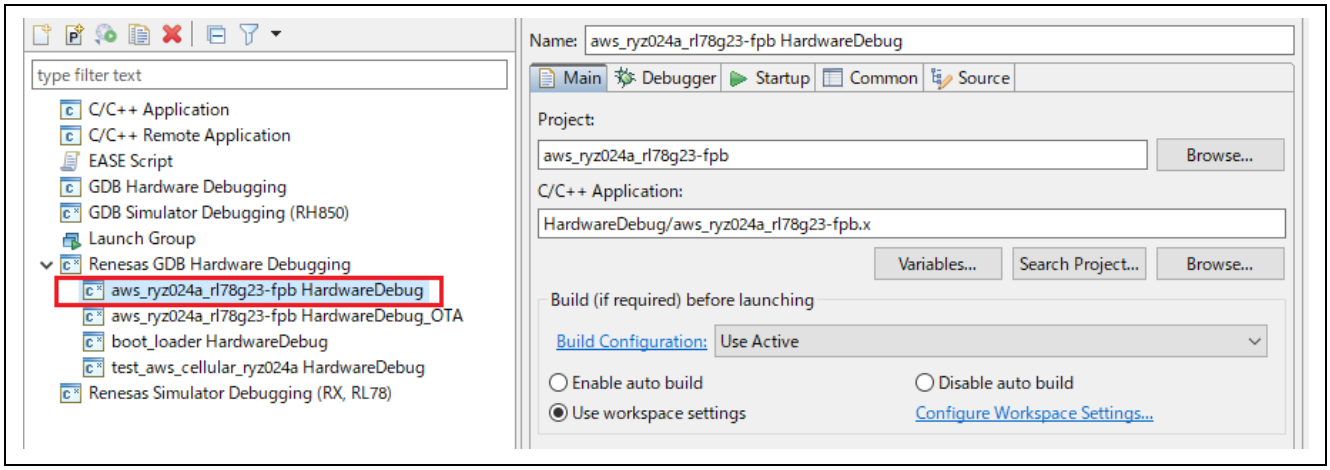


図 8-6　Debug Configurations of Project(PubSub)

  - ―　デモプロジェクト(OTA)の場合：aws_ryz024a_rl78g23-fpb HardwareDebug_OTA

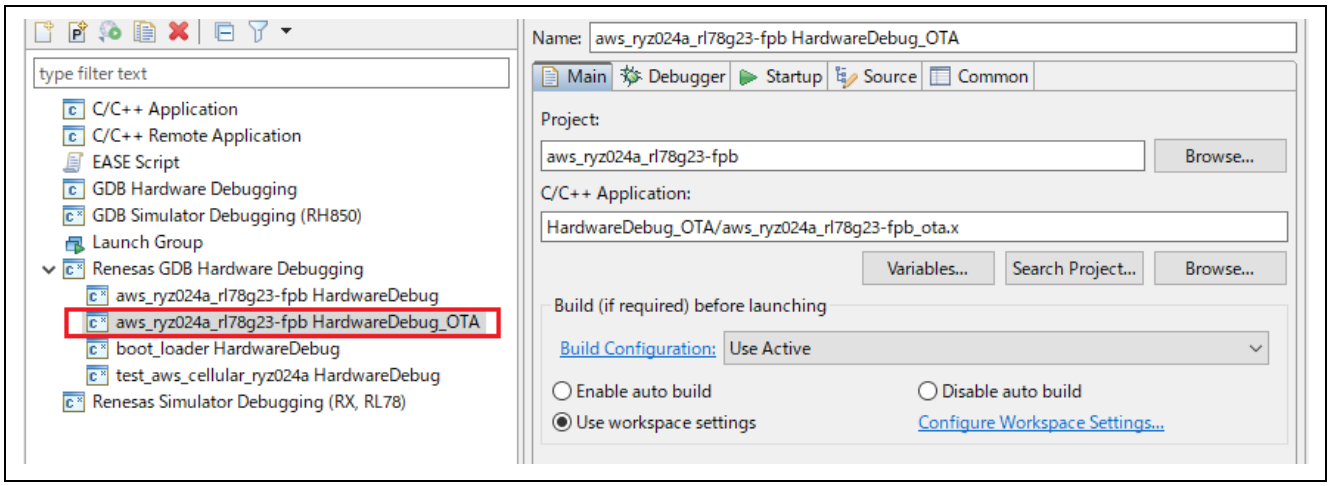


図 8-7　Debug Configurations of Project(OTA)

8.2.4　デバッガ設定

”Debugger” tab を選択する。

- Debug hardware: E2 Lite (RL78)

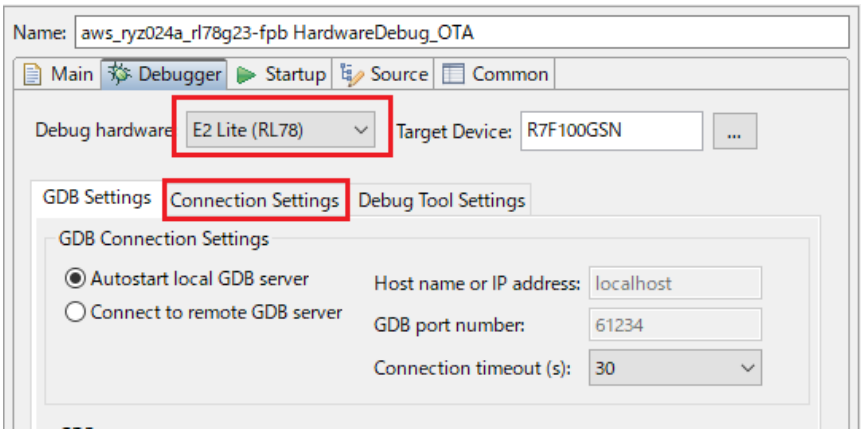


図 8-8　Debug hardware: E2 Lite (RL78)

“Connection Settings” tab > Connection with Target Borad を選択する。

- Power Target From The Emulator (MAX 200mA): No

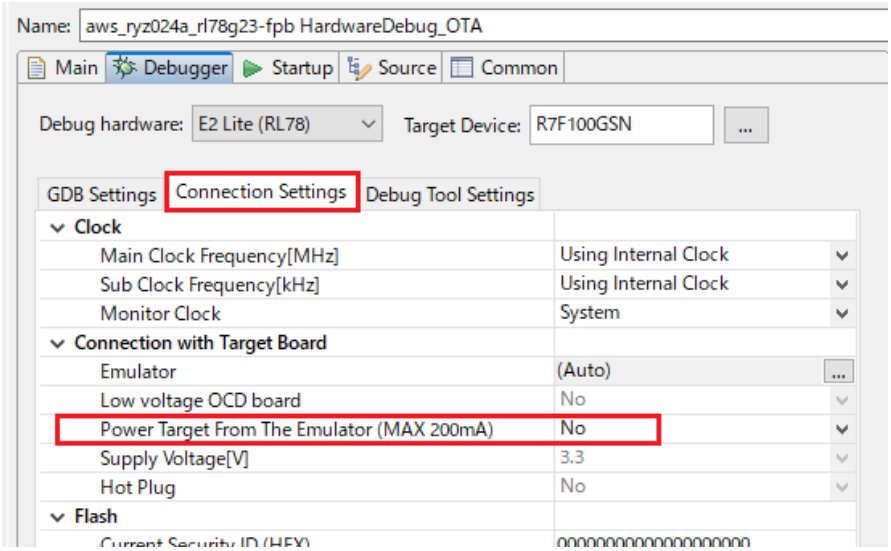
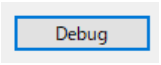


図 8-9　Connection Settings for using Emulator



をクリックしてデバッグを開始する。

9.　付録

9.1　注意事項：サードパーティ製ライブラリを RL78 に移植する際の注意点

RL78 は 16bit システムです。そのため、サードパーティ製ライブラリを RL78 で使用する際に下記の注意が必要です。

9.1.1　Int の幅が 16bit

処理系依存 の型 (int、size_t 等)を使用している箇所はコード変更が必要なことがあります。特にサイズを扱う変数は注意してください。

本デモプロジェクトは以下のサードパーティ製ライブラリを変更しています。

- tinycbor(0.5.2)　https://github.com/intel/tinycbor
- TinyCrypt Cryptographic Library (0.2.8)　https://github.com/intel/tinycrypt

9.1.2　セクションのサイズ制限

一部のセクションは 64KB - 1 境界をまたがって配置できません、言い換えると最大 64KB までしか配置できません。そのため、例えばサイズの大きいサードパーティ製ライブラリを RL78 に移植する際に、デフォルトのセクションに 64KB よりも大きいデータが配置され、リンカ・エラーが発生する場合があります。詳細は「CC-RL コンパイラ ユーザーズマニュアル」(R20UT3123) を参照してください。

この制限を回避するためにはセクションサイズを調整する必要があります。デフォルト定数セクション (.constf) を例に方法を説明します。

最初に、新しい定数セクションを定義してください。

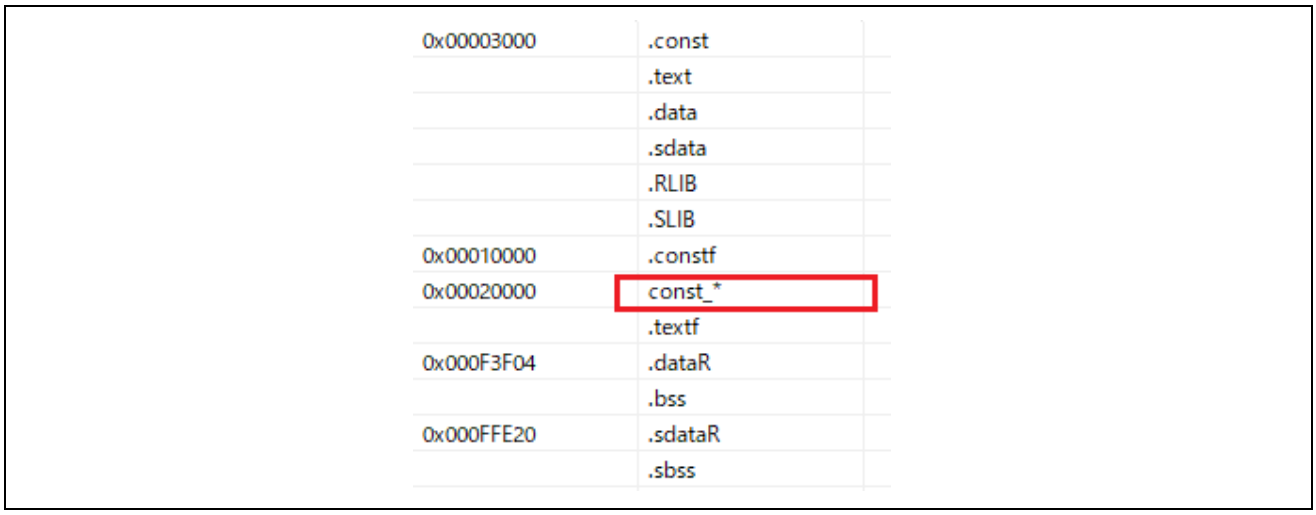


図 9-1　Newly Defined Constant Section (e² studio)

次に、下記の①または②の方法を用いて、サードパーティ製ライブラリのデータが新規に定義した定数セクションへ配置されるように変更してください。なお、本デモプロジェクトは②を採用しています。

① #pragma section 指令

#pragma section 指令をライブラリのソースコードに追記します。

example:core_mqtt.c

```
#if defined(__CCRL__) || defined(__ICCRL78__) || defined(__RL)
#pragma section const const_coreMqtt
#endif

/**
 * @file core_mqtt.c
 * @brief Implements the user-facing functions in core_mqtt.h.
 */
#include <string.h>
#include <assert.h>

...Codes...

#if defined(__CCRL__) || defined(__ICCRL78__) || defined(__RL)
#pragma section
#endif
```

図 9-2　Added #pragma section Directive (3rd party library)

② リンク・オプション -REName

リンク・オプションを下記のように指定することによって、サードパーティ製ライブラリのデータがファイル単位で新規に定義した定数セクションへ配置されるように変更します。この方法はソースファイルを編集する必要がないというメリットがあります。

```
-REName=.\Middleware\FreeRTOS\coreMQTT\source\core_mqtt.obj(.constf=const_coreMqtt_f)
```

9.1.3　ビルド Warning

サードパーティ製ライブラリを現状有姿で使用する場合、ビルド時に使用するツールチェーンは Warning などのメッセージを出力することがあります。ユーザは必要に応じてメッセージ内容をもとに Warning を解決してください。

本製品での実例を下記に紹介します。

本デモプロジェクトはオープンソースソフトウェア（OSS）であるサードパーティ製ライブラリを現状有姿で使用しています。そのため、CC-RL コンパイラは下記の 3 箇所で W0520167（" *型名 1*" 型の引数は型" *型名 2*" の引数と整合しません。）を出力し、メモリ破壊を引き起こします。

Middleware/AWS/ota-for-aws-iot-embedded-sdk/source/ota.c

```
1689:  static DocParseErr_t extractParameter( JsonDocParam_t docParam,
1690:                                        void * pContextBase,
1691:                                        const char * pValueInJson,
1692:                                        size_t valueLength )
1693:  {
           ...
1711:          char * pEnd;
               ...
1715:          *pUint32 = ( uint32_t ) strtoul( pStart, &pEnd, 0 );
           ...
1750:  }
```

図 9-3　W0520167 in ota.c

Middleware/FreeRTOS/FreeRTOS-Cellular-Interface/source/cellular_at_core.c

```
821:   CellularATError_t Cellular_ATStrtoi( const char * pStr,
822:                                       int32_t base,
823:                                       int32_t * pResult )
824:   {
           ...
827:       char * pEndStr = NULL;
           ...
838:           retStrtol = ( int32_t ) strtol( pStr, &pEndStr, base );
           ...
861:   }
```

図 9-4　W0520167 in cellular_at_core.c

Middleware/FreeRTOS/FreeRTOS-Cellular-Interface/source/cellular_pkthandler.c

```
130:   static CellularPktStatus_t urcParseToken( CellularContext_t * pContext,
131:                                             char * pInputLine )
132:   {
           ...
137:       char * pSavePtr = pInputLine, * pTokenPtr = pInputLine;
           ...
149:           pTokenPtr = strtok_r( pSavePtr, ":", &pSavePtr );
           ...
165:   }
```

図 9-5　W0520167 in cellular_pkthandler.c

図 9-3　W0520167 in ota.c を例に取り、メモリ破壊の原因を説明します。このデモプロジェクトではコンパイル・オプションで-far_rom オプションを指定しているため、strtoul 関数の第二引数は far 属性のダブルポインタ変数としてコンパイルされ、引数で指定されたメモリ領域に 3 bytes を書き込むアセンブリコードが生成されます。一方、第二引数に指定された自動変数 pEnd は near 属性のポインタ変数ですので、メモリにアロケーションされている pEnd のサイズは 2 bytes です。したがって、strtoul 関数を実行すると pEnd の後続 1 bytes のメモリ領域がメモリ破壊されます。

図 9-6　Memory corruption due to example of W0520167

対策は、ポインタ変数を far 属性に変更することです。その結果、このポインタ変数がメモリにアロケーションされるサイズは 3 bytes になります。

Middleware/AWS/ota-for-aws-iot-embedded-sdk/source/ota.c

```
1689:  static DocParseErr_t extractParameter( JsonDocParam_t docParam,
1690:                                        void * pContextBase,
1691:                                        const char * pValueInJson,
1692:                                        size_t valueLength )
1693:  {
           ...
1711:          char __far * pEnd;
               ...
1715:          *pUint32 = ( uint32_t ) strtoul( pStart, &pEnd, 0 );
           ...
1750:  }
```

図 9-7　Fixed W0520167 in ota.c

Middleware/FreeRTOS/FreeRTOS-Cellular-Interface/source/cellular_at_core.c

```
821:   CellularATError_t Cellular_ATStrtoi( const char * pStr,
822:                                        int32_t base,
823:                                        int32_t * pResult )
824:   {
           ...
827:       char __far * pEndStr = NULL;
           ...
838:           retStrtol = ( int32_t ) strtol( pStr, &pEndStr, base );
           ...
861:   }
```

図 9-8　Fixed W0520167 in cellular_at_core.c

Middleware/FreeRTOS/FreeRTOS-Cellular-Interface/source/cellular_pkthandler.c

```
130:   static CellularPktStatus_t urcParseToken( CellularContext_t * pContext,
131:                                             char * pInputLine )
132:   {
           ...
137:       char __far * pSavePtr = pInputLine, * pTokenPtr = pInputLine;
           ...
149:           pTokenPtr = strtok_r( pSavePtr, ":", &pSavePtr );
           ...
165:   }
```

図 9-9　Fixed W0520167 in cellular_pkthandler.c

ただし、「表 1-2　動作確認条件 (RL78/G23)」で示した動作環境では破壊されるメモリ領域を使用していません。そのため、このデモプロジェクトでは上記の対策を実施せず OSS を現状有姿で使用しています。

しかし、「表 1-2　動作確認条件 (RL78/G23)」以外の環境ではメモリ破壊が動作に影響を与える可能性があります。「表 1-2　動作確認条件 (RL78/G23)」以外の環境は弊社のサポート対象外です。

9.2　オープンソースソフトウェアのライセンス情報

ユーザは本製品に使用されている各 OSS が定めるライセンス条項を遵守する必要があります。ライセンス条項は各 OSS の公式 Web ページで確認してください。本製品が使用する OSS のリンク先を「表 1-3 動作確認条件 (その他、OSS ライブラリ等)」に記載しています。

10. ウェブサイトおよびサポート

本 Getting Started Guide のサンプルプログラム: https://github.com/renesas/iot-reference-rl78

AWS forum: http://forums.aws.amazon.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Feb.29.24	-	初版発行
1.01	Aug.30.24	1	RYZ024A モジュールの製造中止に関する情報を追記
		18, 71	エミュレータ用コネクタ実装のための回路変更を実施していない場合における補足情報を追記
		43	Policy エディタに記述する"Resource"の値を修正
1.02	Feb.17.25	83 - 87	「ビルド Warning」章を追加し、併せて 9 章の文章表現を修正

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。