RX ファミリ

Amazon Web Services を利用した FreeRTOS OTA の実現方法 (202406-LTS 版)

はじめに

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。

【注】本アプリケーションノートは RX マイコン用 FreeRTOS のリファレンスプロジェクトである、iot-reference-rx の v202406.01-LTS-rx-1.1.0 以降に対応した手順となります。
FreeRTOS-v202210.01 以前の FreeRTOS を使用する場合は「RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現方法(v202210.01-LTS-rx-1.1.3 以降対応版) (R01AN7037)」を参照してください。

【注】本アプリケーションノートでは CK-RX65N v2 ボードおよび Ethernet での動作環境に基づいた実装例を示していますが、他のボードや RYZ014A PMOD モジュール等の通信制御の組み合わせでもご利用いただけます。
各ボードおよび通信制御の組合せについては下記を参照下さい。
[GitHub] https://github.com/renesas/iot-reference-rx/blob/main/Getting_Started_Guide.md#getting-started-guide

【注】 当社は、RYZ014A 型名の既存 LTE モジュールの製造を中止し、この製品の出荷を終了することを発表しました。
RYZ014A の出荷終了に伴い、CK-RX65N v1 ボードの出荷も終了となります。
現在の設計または生産中に RYZ014A を使用している場合、Sequans の製品型名 GM01Q が RYZ014A とピン及び機能に互換性のある代替品となります。

なお、RYZ014A の EOL 通知は下記を参照下さい。
[本リンク] https://www.renesas.com/document/eln/plc-240004-end-life-eol-process-select-part-numbers?r=1503996
[製品ページ] https://www.renesas.com/products/wireless-connectivity/cellular-iot-modules/ryz014a-lte-cat-m1-cellular-iot-module

動作確認デバイス

RX65N、RX651 グループ

ハードウェア

CK-RX65N v2

目次

1. 概要

1.1 システム概要

本項では、CK-RX65N v2 搭載のデュアルバンク機能対応マイコン RX65N を使用し、OTA を実現する場合の動作概要を示します。

デュアルバンク機能対応マイコンでは、ROM を Execution area（実行領域）と Temporary area(バッファ領域)に分割することが可能です。Execution area と Temporary area は動的に切り替えることができ、Execution area で既存バージョンのファームウェアを動作させながら更新ファームウェアを Temporary area に書き込むことが可能です。

以下に OTA 実行時のメモリ配置および、デュアルバンク機能を使用したバンクスワップによるメモリ切り替えの動作を示します。

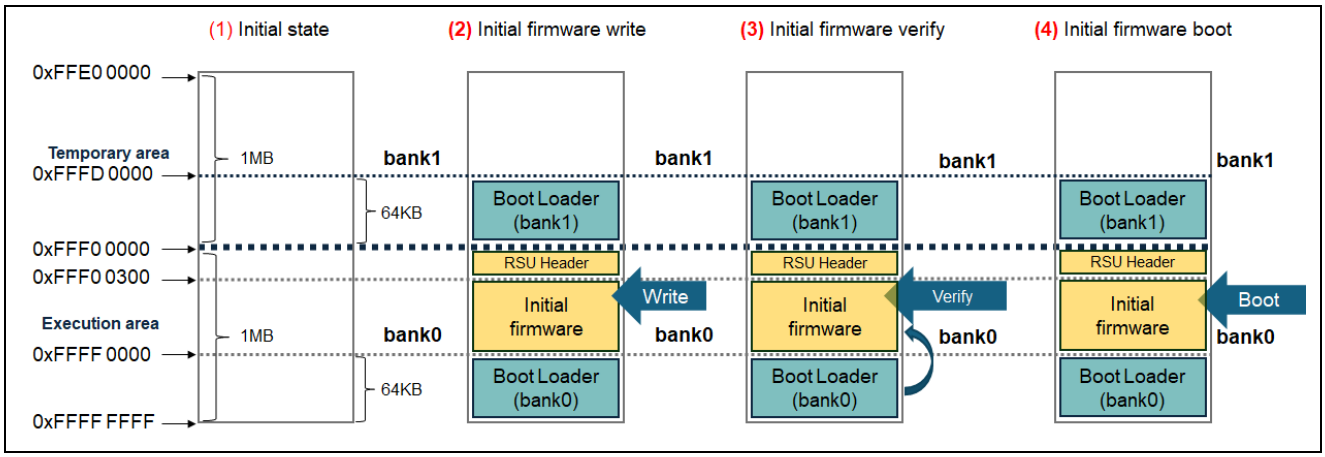


図 1-1 OTA の動作概要(1)

(1) Renesas Flash Programmer にてイレーズを実行した状態（ブランク状態）です。

(2) Renesas Flash Programmer にてブートローダと初期ファームウェアを結合したデータ（注）を書き込んだ状態です。

【注】Boot Loader(bank0) + Initial firmware + RSU Header + Boot Loader(bank1)が結合したデータを指します。RSU Header の詳細については「RX ファミリ ファームウェアアップデートモジュール Firmware Integration Technology アプリケーションノート (R01AN6850)」の「5.2 イメージファイル」をご確認ください。

(3) リセット解除後、ブートローダ(bank0)がファームウェアの検証を行います。

(4) 初期ファームウェアを起動します。

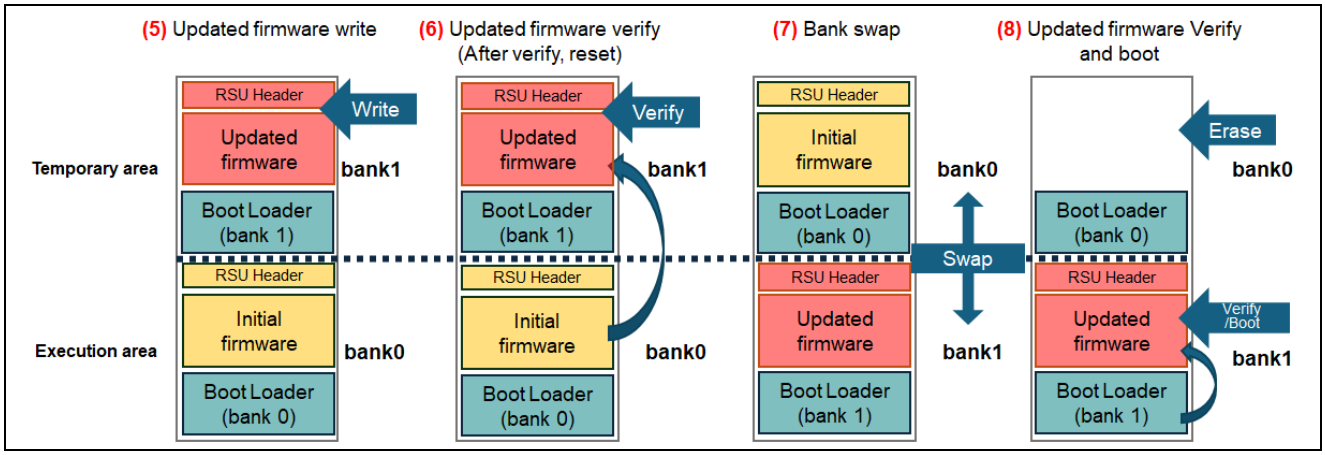


図 1-2 OTA の動作概要(2)

(5) Amazon Web Services（AWS）から更新されたファームウェアを受信すると bank1 に書き込みを行います。
bank1 に書き込み中は BGO 機能により初期ファームウェアの動作が実行されます。

(6) 初期ファームウェアによって更新ファームウェアの検証を行います。

(7) bank0 と bank1 を入れ替え（バンクスワップ）、Execution area に bank1 を配置します。

(8) 更新ファームウェアをブートローダ(bank1)が検証します。

検証に失敗した場合は旧ファームウェアに戻し、旧ファームウェアを起動します。
また、検証に成功した場合は bank1 に書き込んだ更新ファームウェアを実行します。

更新ファームウェア実行後はセルフテストを実施し、合格した場合は bank0 の初期ファームウェアをイレーズします。
セルフテストに失敗した場合は旧ファームウェアに戻しリセットを実行します。

1.2 動作確認環境(ハードウェア)

本デモプロジェクトの動作確認環境(ハードウェア)を以下に示します。

表 1-1 動作確認環境(ハードウェア)

項目	詳細
使用ボード	CK-RX65N v2（Ethernet）

1.3 動作確認環境(ソフトウェア)

本デモプロジェクトの動作確認環境(ソフトウェア)を以下に示します。

表 1-2 動作確認環境(ソフトウェア)

項目	詳細
統合開発環境	e^2 studio 2025-04
コンパイラ	Renesas CC-RX v3.07.00 GCC for Renesas RX v8.3.0.202411
FreeRTOS	v202406.01-LTS-rx-1.1.0
ログモニタツール	Tera Term v4.108
Python	Python 3.11.0
鍵生成ツール	Win64 OpenSSL v3.4.1
フラッシュ書き込みツール	Renesas Flash Programmer V3.19.00
Renesas Image Generator	Version3.03（Firmware Update module Rev.2.04 同梱）
AWS コマンドラインインターフェイス（CLI）	AWS Command Line Interface Version 2

2. 事前準備

2.1 Tera Term のインストール

(1) Tera Term のダウンロードサイトへアクセス

以下のリンクより Tera Term のダウンロードサイトにアクセスします。

Tera Term ダウンロードサイト(GitHub)

(2) Tera Term インストーラーのダウンロード

Tera Term 4.xxx のバージョンを指定してインストーラーをダウンロードしてください。

【注】Tera Term v5.xxx は本アプリケーションのデモプロジェクトでは一部対応できない機能があるため、使用できません。

(3) インストーラーの実行

インストーラーを実行し、案内に沿って Tera Term をインストールします。
インストーラーの実行は管理者権限で行ってください。

(4) Tera Term の起動の確認

スタートメニューから Tera Term のアイコンをクリックして、Tera Term が起動することを確認します。

2.2 Python のインストール

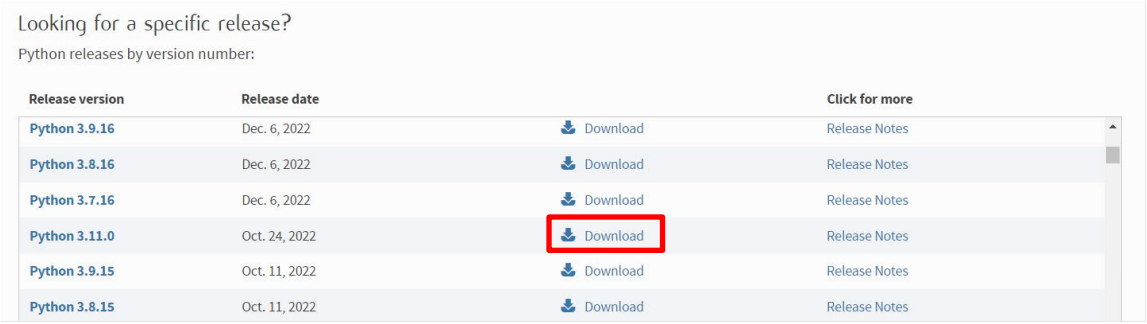
(1) Python のダウンロードサイトへアクセス

以下のリンクより Python のダウンロードサイトにアクセスします。

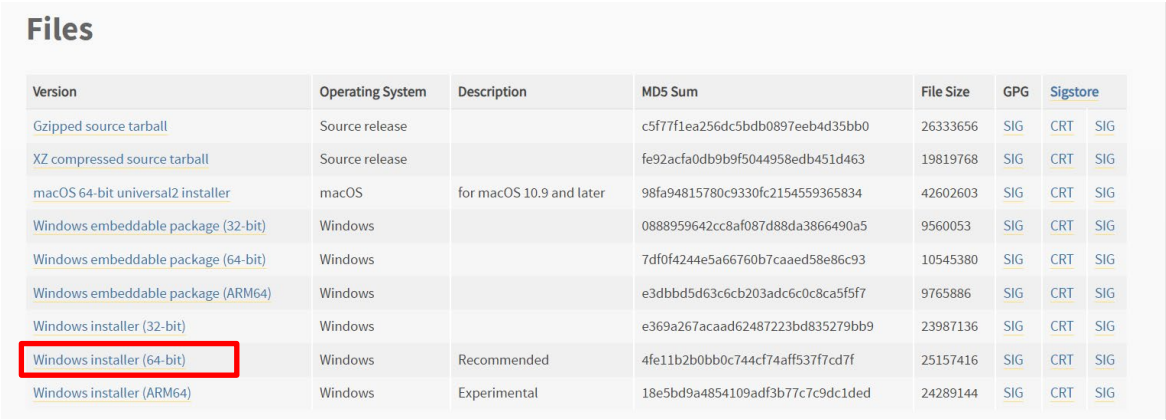
Python ダウンロードサイト

(2) Python インストーラーのダウンロード

バージョンリストより Python 3.11.0 の Download を選択します。

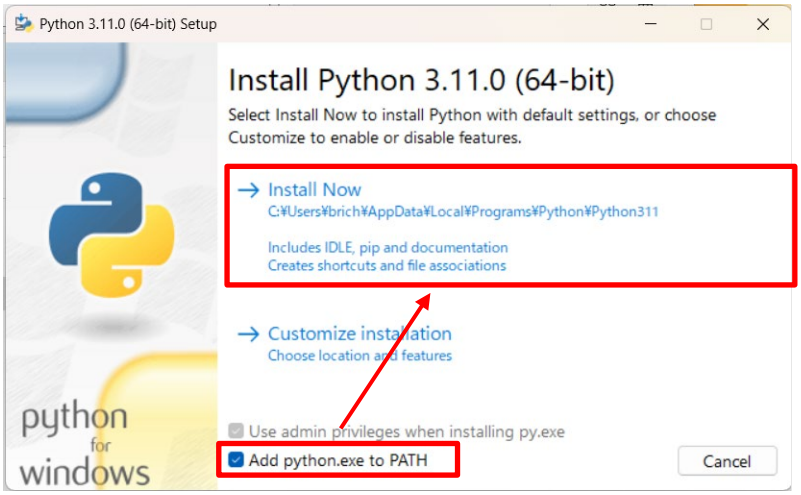


使用する OS に合わせたインストーラーをダウンロードしてください。



(3) Python のインストール

インストーラーを実行し、案内に沿って Python をインストールします。
インストール画面で[Add python.exe to PATH]にチェックを入れてください。



(4) Python のインストールの確認

コマンドプロンプトを起動し、Python 3.11.0 がインストールされていることを確認します。
以下のコマンドを実行して、バージョン情報が表示されることを確認してください。

```
> python -V
```

【注】V は大文字で入力してください

コマンド実行後は以下の様に表示されます。

```
C:¥Users>python -V
Python 3.11.0
```

(5) Python へ暗号化ライブラリのインストール

Python に暗号化ライブラリ「pycryptodome」をインストールします。
以下のコマンドを実行して、暗号化ライブラリをインストールしてください。

```
> pip install pycryptodome
```

コマンド実行後は以下の様に表示されます。

```
C:¥Users>pip install pycryptodome
Requirement already satisfied: pycryptodome in c:¥users¥(      )¥appdata¥local¥programs¥python¥p
ython311¥lib¥site-packages (3.18.0)

[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

2.3 OpenSSL のインストール

(1) OpenSSL のダウンロードサイトを開く

以下のリンクより Win32/Win64 OpenSSL のダウンロードサイトにアクセスします。

（Win32/Win64 OpenSSL Installer for Windows - Shining Light Productions (slproweb.com)）

(2) OpenSSL インストーラーのダウンロード

OpenSSL のインストーラーをダウンロードします。
使用する OS に合わせたインストーラーをダウンロードしてください。

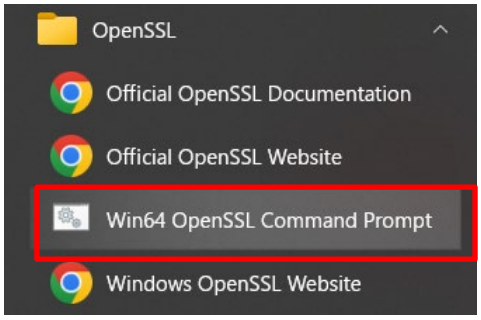
File	Type	Description
Win64 OpenSSL v3.4.1 Light EXE | MSI	5MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v3.4.1 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v3.4.1 EXE | MSI	221MB Installer	Installs Win64 OpenSSL v3.4.1 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v3.4.1 Light EXE | MSI	4MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v3.4.1 (Only install this if you need 32-bit OpenSSL for Windows). Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

(3) OpenSSL インストーラーの実行

インストーラーを実行し、案内に沿って OpenSSL をインストールします。
OpenSSL の DLL の保存先には[The OpenSSL binaries directory]を選択してください。

(4) OpenSLL コマンドプロンプトの実行

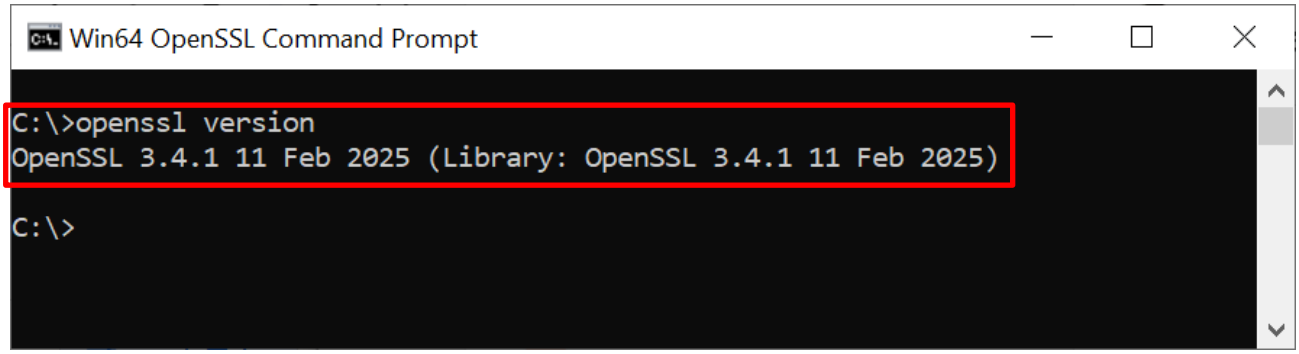
スタートメニューから Win64 OpenSSL Command Prompt を実行します。



(5) OpenSSL の実行確認

コマンドプロンプトで openssl コマンドが実行できることを確認します。
以下のコマンドを実行して、バージョン情報が表示されることを確認してください。

```
> openssl version
```



2.4 Renesas Image Generator のインストール

Renesas Image Generator は、ファームウェアアップデートモジュールで使用するファームウェアイメージを生成するユーティリティツールです。Renesas Image Generator はファームウェアアップデートモジュールが使用する以下のイメージを生成することができます。

- 初期イメージ：ブートローダとアプリケーションプログラムで構成されるシステムの初期設定時にフラッシュライタで書き込むイメージファイル(拡張子 mot)
- 更新イメージ：ファームウェアアップデート対象のイメージファイル(拡張子 rsu)

【注】 Firmware Update module Rev.2.00 以降のバージョンでは、Python スクリプトを使用したファームウェアイメージの生成にのみ対応しております。

Renesas Image Generator は FIT モジュールの Firmware Update module Rev.2.04 に同梱されています。
次の手順で Renesas Image Generator を入手できます。

(1) Renesas Image Generator 導入フォルダをオープン

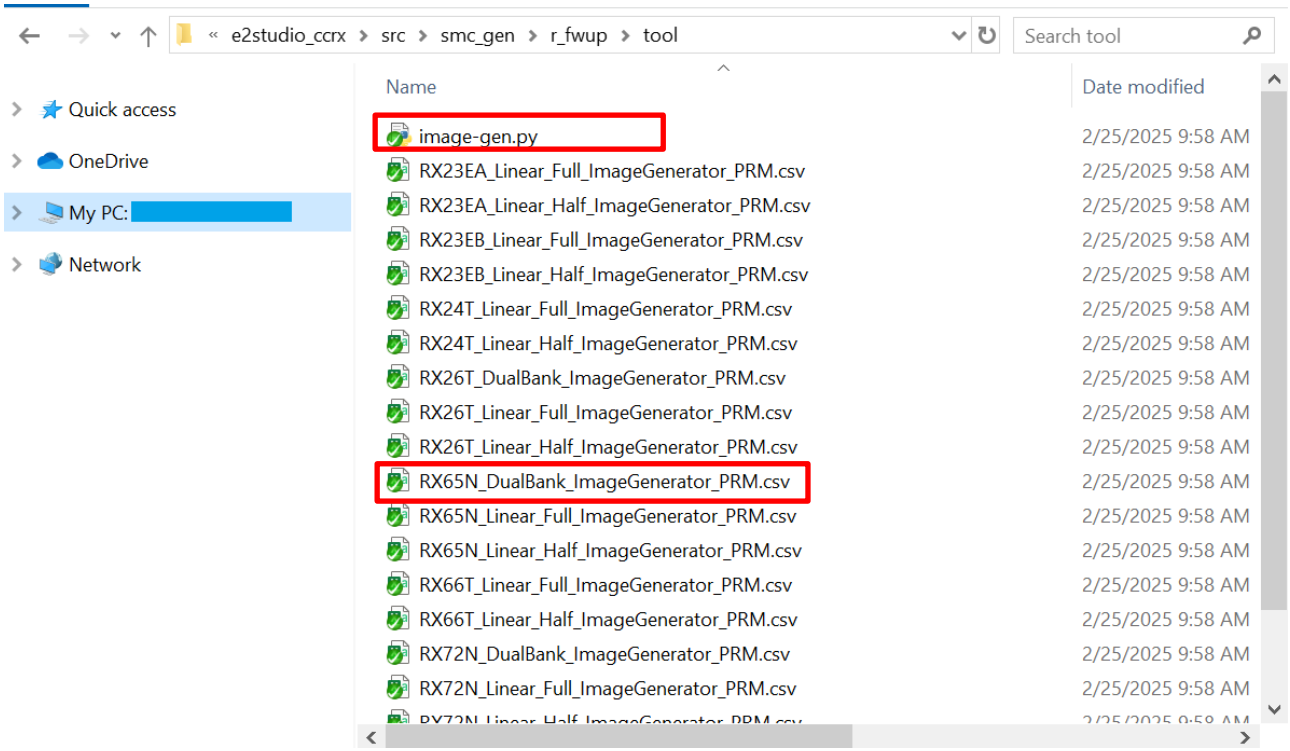
サンプルプロジェクトをインポート後に以下のフォルダを開いてください。
サンプルプロジェクトのインポート手順は「4.2.1」を参照してください・

```
\iot-reference-rx\Projects\aws_ether_ck_rx65n_v2\e2studio_ccrx\src\smc_gen\r_fwup\tool
```

上記のパスは「4.2.1」でプロジェクトをインポートして導入した場合のフォルダとプロジェクト名の例です。

(2) Renesas Image Generator フォルダの確認

「tool」フォルダ下には以下のように Renesas Image Generator スクリプトファイル(image-gen.py)と各デバイス用のパラメータファイル（*_ImageGenerator_PRM.csv）が含まれています。



(3) 使用するファイルの確認

Renesas Image Generator で使用するファイルを確認します。
本アプリケーションノートでは以下のファイルを使用します。

- image-gen.py： Renesas Image Generator スクリプトファイル
- RX65N_DualBank_ImageGenerator_PRM.csv： RX65N デュアルバンク用パラメータファイル

任意のフォルダに「Renesas Image Generator」というフォルダを作成し、上記ファイルをコピーしてください。

2.5　AWS コマンドラインインターフェイス（CLI）のインストール

AWS コマンドラインインターフェイス（CLI）は、Amazon Web Services（AWS）の各種サービスをコマンドラインシェルからコマンドを使用して管理・操作するためのツールです。
本アプリケーションノートでは AWS の各種設定について AWS CLI を使用した操作でガイドします。

【注】 AWS CLI は Version 1 と Version 2 に互換性がありません。必ず Version 2 をインストールしてください。
Version 1 がインストール済みの場合は以下のページを参照し、Version 2 へ更新してください。
AWS CLI バージョン 1 から AWS CLI バージョン 2 への移行

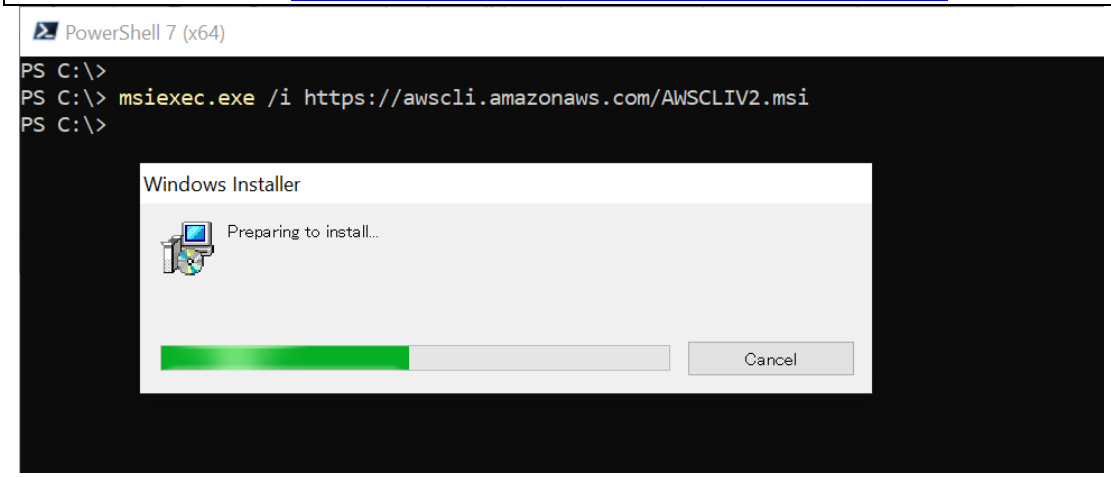
(1)　コマンドラインシェルの実行

Windows PowerShell か Windows コマンドプロンプトを実行してください。本書では PowerShell を使用した画面例で説明します。
本アプリケーションノートでは、コマンドラインシェルはコマンドプロンプトと呼びます。

(2)　AWS CLI のインストール

コマンドプロンプトで以下のコマンドを入力してください。自動的に AWS CLI v2 のインストーラーがダウンロードされ、実行されます。

```
> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

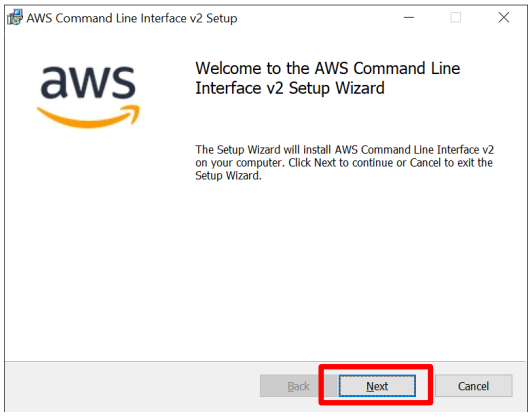


また、以下ページからダウンロードとインストールの手順を参照することも可能です。

AWS CLI のインストールと更新の手順

(3)　インストーラーの実行

インストーラーの実行後、しばらく待つと[Next]ボタンが押せるようになります。
[Next]ボタンを押下後、案内に従って AWS CLI インストールを行ってください。
インストール後はコマンドプロンプトの再起動を行ってください。



(4) インストールの確認

インストールが完了したら、コマンドプロンプトで以下を入力してください。インストールされた AWS CLI のバージョンが表示されます。

```
> aws --version
```

PowerShell 7 (x64)

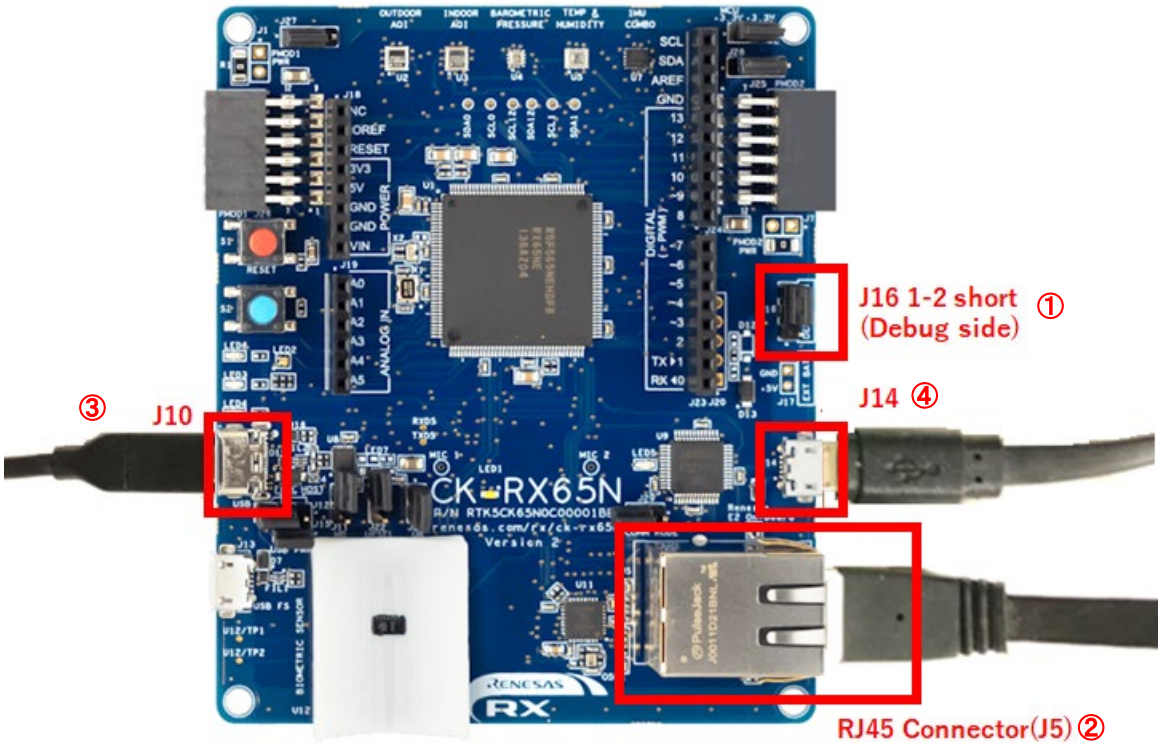
```
PS C:\>
PS C:\> aws --version
aws-cli/2.24.9 Python/3.12.6 Windows/10 exe/AMD64
PS C:\>
```

インストールされたバージョンが 2.xx.x となっていることが確認出来たら完了です。

2.6 CK-RX65N v2 の接続

(1) イーサネット接続の場合

ベースボードの接続とジャンパーを設定します。



ベースボード CK-RX65N v2

① ベースボードの J16 の 1-2 をショートします（デバッグ許可）。

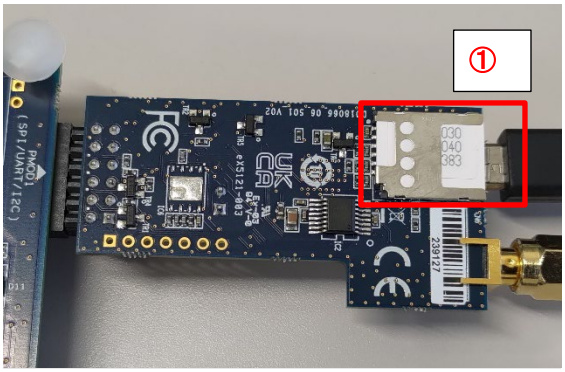
② ベースボードの RJ45 コネクタ（J5）へ LAN ケーブルを接続します（インターネット接続）。

③ ベースボードの J10 と PC を USB ケーブルで接続します（USB シリアル接続）。

④ ベースボードの J14 と PC を USB ケーブルで接続します（デバッガ接続）。

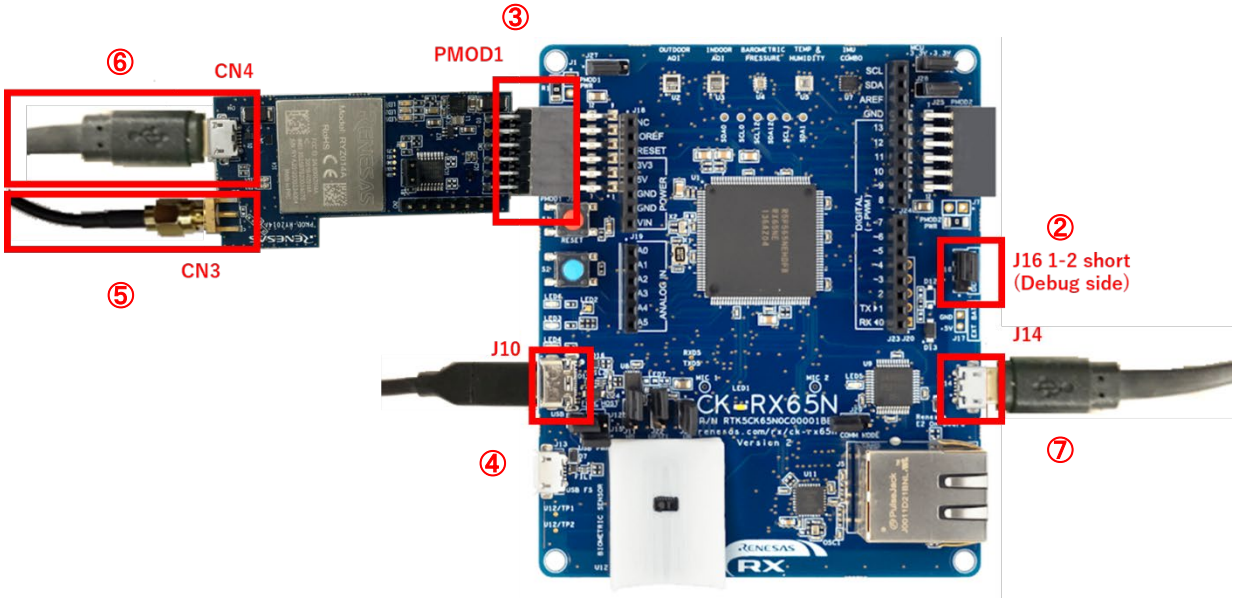
(2) セルラー接続（RYZ014A）の場合

RYZ014A 使用してセルラーで通信する場合は、以下の接続を参照してください。



RYZ014A PMOD 裏面

① RYZ014A PMOD の CN6 に SIM カードを挿入してください。



ベースボードおよび RYZ014A PMOD 表面

② ベースボードの J16 の 1-2 をショートします（デバッグ許可）。

③ ベースボードの PMOD1 に RYZ014A PMOD を接続します。

④ ベースボードの J10 と PC を USB ケーブルで接続します（USB シリアル接続）。

⑤ RYZ014A PMOD の CN3 にアンテナを接続します。

⑥ RYZ014A PMOD の CN4 に USB ケーブルを接続し、電源供給します。

⑦ ベースボードの J14 と PC を USB ケーブルで接続します（デバッガ接続）。

【注】 予備の USB ケーブルをお持ちの場合は、手順⑥を実施してください。
RYZ014A PMOD へ直接電源供給を行わない場合は、通信が不安定になる場合があります。

3. AWS の設定

本章では FreeRTOS デモを実行するための AWS の設定手順を説明します。

本アプリケーションノートでは IAM Identity Center ユーザー（ワークフォースユーザー）を使用した Single-Sign-On（SSO）を利用したコマンドラインインターフェイス（CLI）による手順を解説します。
IAM Identity Center のワークフォースユーザーは IAM ユーザーとアカウントの管理が異なるので注意してください。

また、AWS CLI のコマンドの詳細は以下の AWS のドキュメントを参考にしてください。

AWS CLI コマンドの例 - AWS Command Line Interface

3.1 AWS サインイン環境の作成

以下の手順に従って AWS にサインインし、IAM Identity Center のワークフォースユーザーを作成し AWS へサインインできる環境を作成してください。

3.1.1 AWS マネジメントコンソールへのサインイン

AWS のルートユーザー（管理アカウント）を作成し、AWS マネジメントコンソールへサインインします

(1) サインインアカウントの取得

AWSへサインインするためには、AWS アカウントが必要です。
以下のAWSドキュメントを参照し、AWS のアカウントを作成してください。

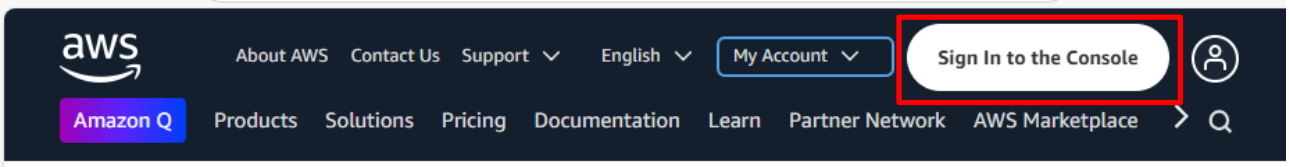
- Set up AWS account - AWS IoT Core

(2) AWS コンソールへのサインイン

作成した AWS アカウント（管理アカウント）で AWS マネジメントコンソールへサインインします。

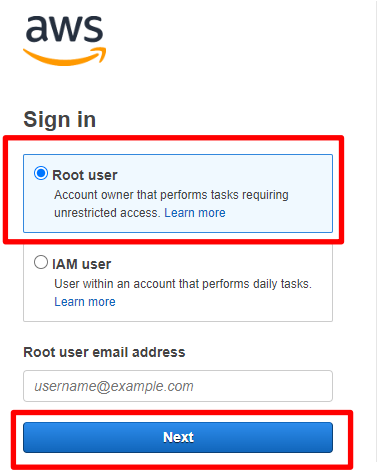
(a) AWS マネジメントコンソールにサインイン

AWS（https://aws.amazon.com/）にアクセスし、[Sign In to the Console（コンソールにサインイン）]をクリックします。



(b)　E メールアドレスの入力

ルートユーザーを選択してから、ルートユーザーの E メールアドレスを入力し[Next（次へ）]をクリックします。
（過去サインインしていた場合、本手順はスキップされる場合があります）。
IAM Identity Center サービスは管理アカウントでしか作成ができませんので、ルートユーザーでサインインしてください。

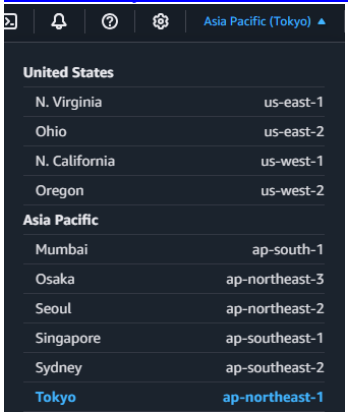


(c)　パスワードの入力

パスワードを入力して、[Sign in（サインイン）]をクリックしてください。

3.1.2　AWS のリージョン設定

AWS マネジメントコンソールにログイン後、画面右上にあるリージョンを設定してください。
IAM Identity Center は使用できるリージョンが決まっているため、適切なリージョンを選択してください。使用できるリージョンは以下を参照してください、
IAM Identity Center　リージョンのデータストレージとオペレーション



3.1.3　IAM Identity Center ワークフォースユーザーの作成

AWS のアカウント取得後は管理アカウントで AWS マネジメントコンソールにサインイン後、IAM Identity Center サービスにてワークフォースユーザーを作成し、AWS へ SSO できる環境を作成します。

以下の手順で IAM Identity Center のインスタンス作成後、ユーザーを作成しユーザーやグループへ必要な許可セット（権限）を割り当ててください。

(1)　IAM Identity Center の有効化

以下ページを参照し、IAM Identity Center を有効化します。有効化後は管理アカウント用のインスタンスが生成されます。
IAM Identity Center を有効にする - AWS IAM Identity Center
管理アカウントで作成された IAM Identity Center インスタンスでのみユーザー管理が可能です。

(2)　ワークフォースユーザーの作成

以下ページを参照し、IAM Identity Center でユーザーを作成します。ここで作成するユーザーがワークフォースユーザーのアカウントとなります。
アイデンティティセンターディレクトリにユーザーを追加する - AWS IAM Identity Center
ユーザーが作成されると、作成したユーザーのメールアドレスへ AWS から認証メールが届きますので、メールの指示に従ってパスワードの登録、多要素認証（MFA）の設定を行います。

(3)　グループの作成

以下ページを参照し、必要に応じてユーザーの属するグループを作成します。
アイデンティティセンターディレクトリにグループを追加する - AWS IAM Identity Center
グループを作成した場合は、作成したグループにユーザーを追加します

(4)　許可セットの作成

以下ページを参照し、ユーザーやグループに割り当てる許可セットを作成します。
アクセス許可セットの作成、管理と削除 - AWS IAM Identity Center
通常は AdministratorAccess を作成で問題ありません。組織ポリシーで AdministratorAccess 以外の許可セットを設定する場合は、OTA を実行するために以下の許可セットをユーザーに割り当てる必要があります。

- AWSIoTFullAccess
- AmazonFreeRTOSOTAUpdate
- AWSIoTDeviceTesterForFreeRTOSFullAccess

【注】これらの許可セットは AdministratorAccess に含まれます。

(5)　AWS アカウントへアクセスするユーザー・グループの割り当て

以下ページを参照し、作成したユーザーおよびグループを AWS の管理アカウントに割り当てて、許可を付与します。
AWS アカウントへユーザーアクセスを割り当てる - AWS IAM Identity Center
これにより、ワークフォースユーザーは AWS へサインインできるようになります。

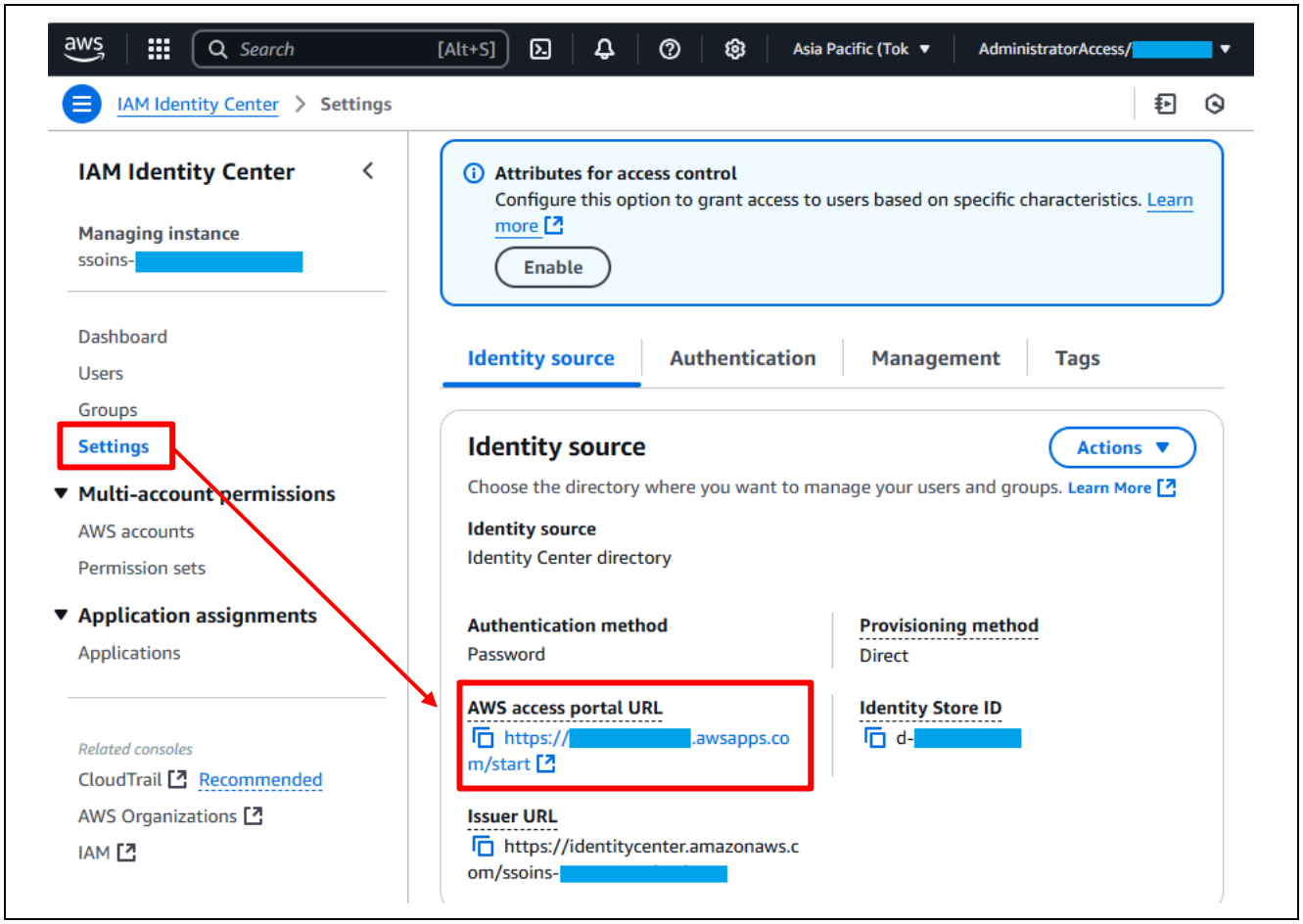
【注】IAM ユーザーによるアクセスキーを使用したサインインでも AWS CLI を利用することができます。この場合は以下を参照しアクセスキーを入手できます。
IAM ユーザーのアクセスキーを管理します。 - AWS Identity and Access Management

3.1.4　AWS CLI サインインのための準備

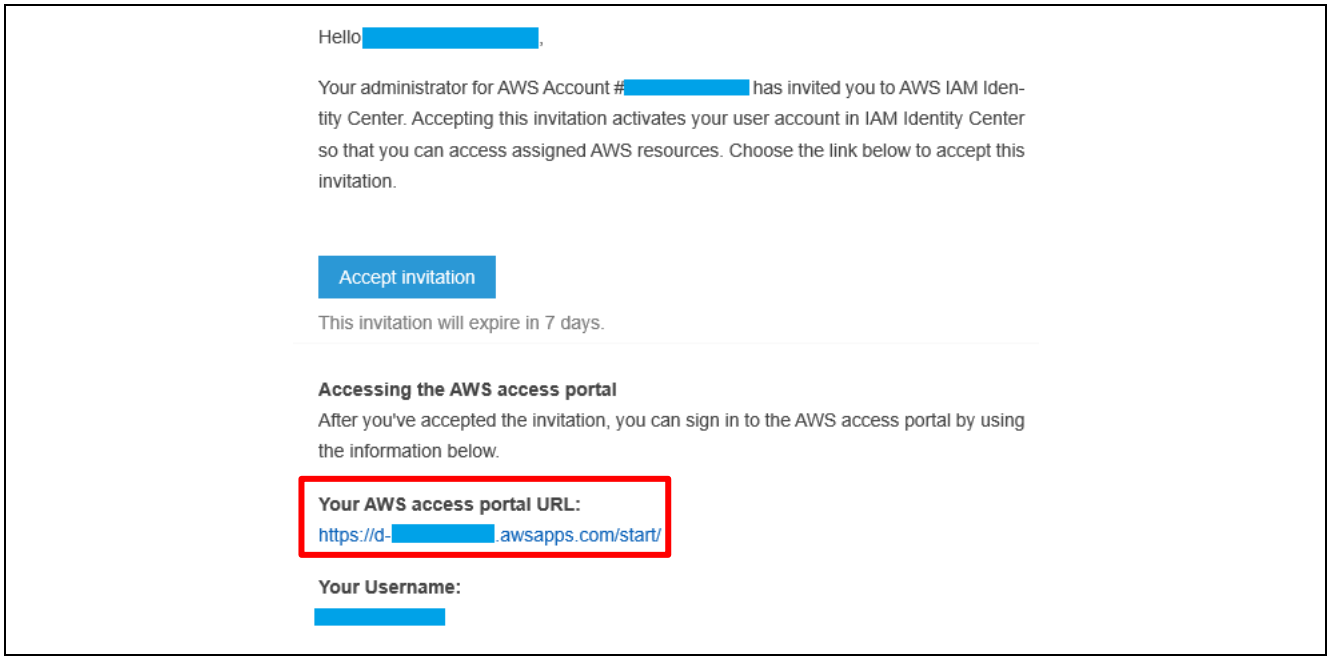
IAM Identity Center ワークフォースユーザーを使用し CLI へサインインするためのプロファイルを作成します。プロファイルは一度のみ作成すれば流用が可能です。

(1)　AWS アクセスポータルの URL を取得

ルートユーザーで AWS マネジメントポータルへサインインし、IAM Identity Center のメニューの[Settings（**設定**）]をクリックして、アイデンティティソースより「AWS access portal URL」を記録してください。
ルートユーザーでサインインできない場合はアカウント管理者へ AWS access portal URL **を**お問い合わせください。



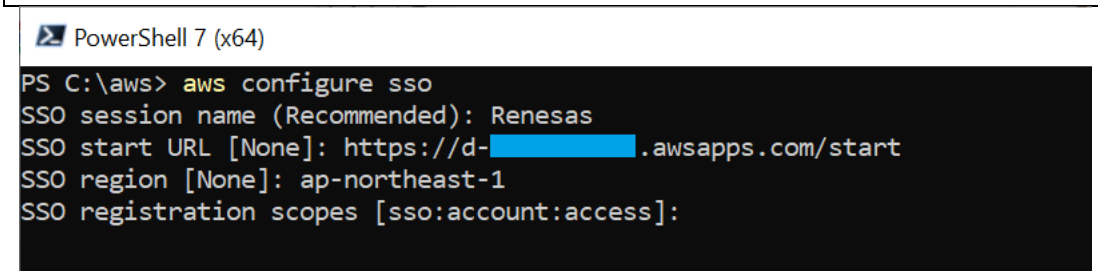
また、AWS access portal URL は、初回のワークフォースユーザーの認証時のメールに記載されています。



(2) SSO プロファイルの作成

コマンドプロンプトを実行し、以下のコマンドを実行します。

```
> aws configure sso
```

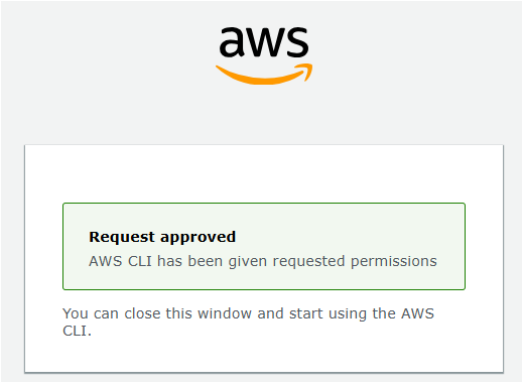


コマンドを実行すると以下の 4 項目の入力を要求されますので、それぞれ設定内容の通り入力してください。

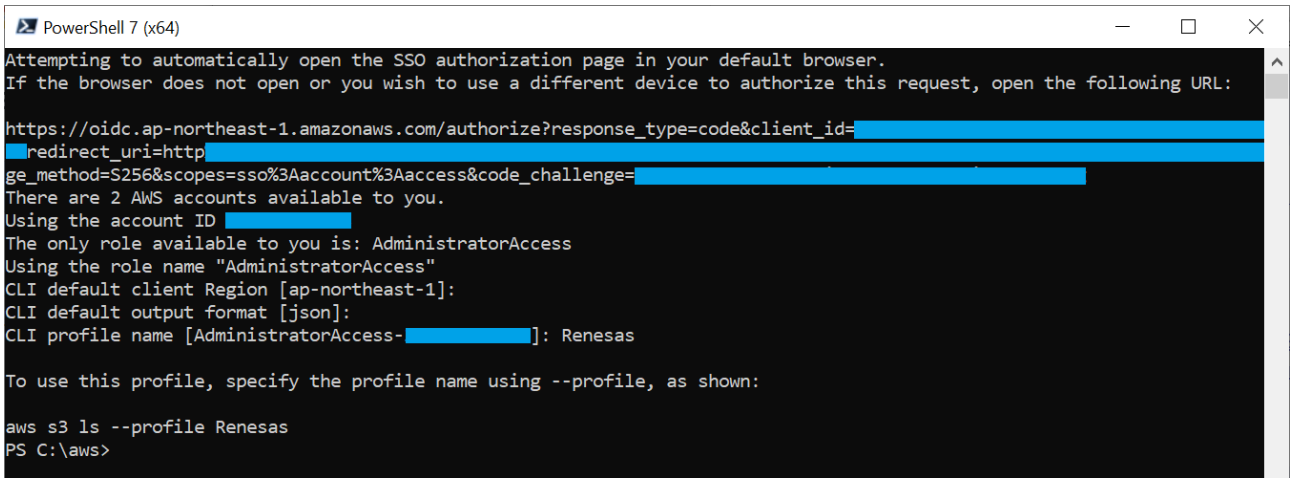
項目	内容	設定内容
SSO session name	セッション名	登録する SSO プロファイルのセッション名です。任意の文字列を入力します。
SSO start URL	SSO の開始 URL	SSO を開始する URL を登録します。 「(1)」で取得した AWS アクセスポータルの URL を入力します。
SSO region	リージョン名	SSO を使用するリージョンを入力します。 IAM Identity Center のインスタンスを作成したリージョンの文字列を入力してください。
SSO registration scopes	アクセス許可のスコープ	何も入力しないで[Enter]を押してください。 初期値である「sso:account:access」が設定されます。

SSO registration scopes の項目で[Enter]を押すと WEB ブラウザが開き、AWS のサインイン画面が表示されるので、ワークフォースユーザーのアカウントでログインしてください。
正常にサインインできると以下の様に表示されます。

【注】サインイン済みの情報が残っている場合はサインイン画面に遷移せず、「Request approved」の画面が表示される場合があります。



アカウントと許可セットが認証されるとコマンドプロンプトに以下の様に表示されます。



以下の 3 項目の入力を要求されますので、それぞれ設定内容の通り入力してください。

項目	内容	設定内容
CLI default client Region	CLI のリージョン	CLI で使用するリージョンを入力します。 基本は最初に設定したリージョンのままとしてください。
CLI default output format	CLI で出力される書式	「json」と入力してください
CLI profile name	CLI のプロファイル名	任意のプロファイル名の文字列を入力します。CLI のコマンドはこのプロファイル名を指定して入力します。

以上でプロファイルの作成は完了です。

CLI のコマンドを入力する際は、コマンドの終端に「--profile *PROFILE_NAME*」を入力し、プロファイルを指定します。
「PROFILE_NAME」は本項で設定したプロファイル名を指定します。

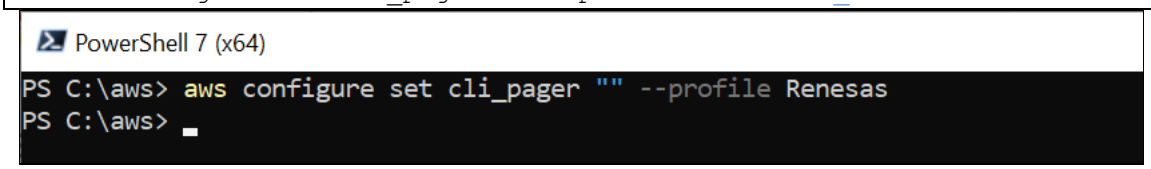
本アプリケーションノートではプロファイル名を「Renesas」と設定した例で説明します。

(3) プロファイルへ追加の設定

プロファイルへ CLI 操作のための追加設定を行います。
この設定は CLI のコマンドの結果がコマンドプロンプトの 1 画面に収まらなかった場合に表示を一時停止する機能を解除します。
本アプリケーションノートでは JSON データを表示する場合に表示される情報量が多い場合があるため設定を行います。

以下のコマンドを入力してください。

```
> aws configure set cli_pager "" --profile <PROFILE_NAME>
```



(4) CLI を使用した AWS へのログインの確認

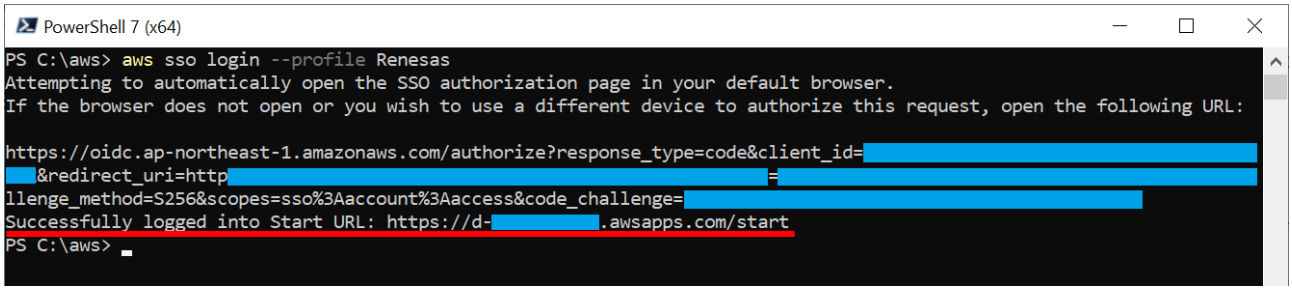
作成したプロファイルで AWS へログインします。以下のコマンドを入力してください。

```
> aws sso login --profile <PROFILE_NAME>
```

コマンド入力後は、WEB ブラウザが開かれ、AWS のサインイン画面が表示されますので、ワークフォースユーザーのアカウントでサインインしてください。

【注】サインイン済みの情報が残っている場合はサインイン画面に遷移せず、「Request approved」の画面が表示される場合があります。

CLI による AWS ログイン後はコマンドプロンプトに以下の様に表示されます。
下線のように「Successfully logged into Start URL:～」と表示されればログイン成功です。

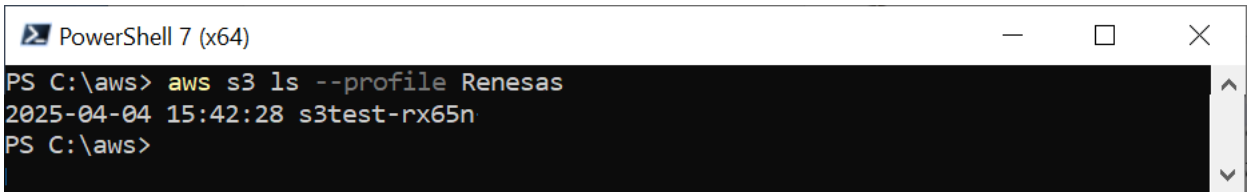


(5) CLI の動作確認

SSO でログイン出来たら、以下の S3 バケットのリストを取得するコマンドを入力し、正常に CLI が操作できるかを確認してください。
エラーが出なければ CLI のコマンドが正常に受け付けされています。
アカウントに S3 バケットが存在する場合はバケットのリストが表示されます。

```
> aws s3 ls --profile <PROFILE_NAME>
```

コマンドを実行すると以下の様に表示されます。
以下の画面は既存のバケット（s3test-rx65n）が存在する例です。



(6) AWS からのログアウト

コマンドプロンプトで以下のコマンドを入力してください。

```
> aws sso logout
```

3.1.5 作業フォルダの作成

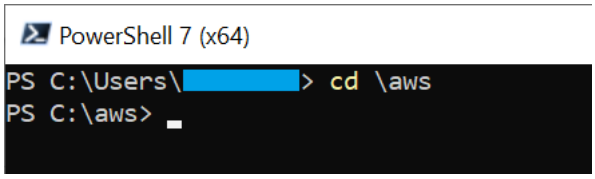
AWS CLI で作業を行うための任意のフォルダを作成します。このフォルダには CLI のコマンドに使用する設定ファイル等を配置します。
本アプリケーションノートでは「C:\aws」にフォルダを作成した例で説明を行います。

3.2　CLI を使用した AWS への SSO ログイン

AWS CLI を使用して AWS へログインします。

(1)　作業フォルダへ移動

コマンドプロンプトを起動し、「3.1.5」で作成した作業フォルダにカレントディレクトリを移動します。



(2)　AWS へログイン

コマンドプロンプトで以下のコマンドを入力してください。

```
> aws sso login --profile <PROFILE_NAME>
```

<PROFILE_NAME>には「3.1.4(2)」で作成したプロファイル名を入力します。
また、プロファイル名はこの後説明するすべての CLI コマンドの最後に入力してください。

SSO のログインについての詳細は、「3.1.4(4)」を参照してください。

3.3　デバイスを AWS に登録する

デバイスを作成し、必要な権限等を割り当てます。

3.3.1　ポリシーの設定

AWS IoT Core サービスを使用して、接続するデバイスに対して AWS のリソースへアクセス許可（ポリシー）を設定します。
本アプリケーションノートで接続するデバイスには、以下のポリシーを設定します。

- iot:Connect　　: AWS IoT に接続する
- iot:Publish　　: トピックをパブリッシュ（送信）する
- iot:Subscribe　　: トピックをサブスクライブ（受信）する
- iot:Receive　　: AWS IoT からメッセージを受信する

(1) ポリシーを設定した json ファイルの作成

ポリシーの設定ファイルを作成します。テキストエディタで「3.1.5」で作成した作業フォルダに「policy.json」というファイルを作成し、以下のコードを入力します。

- policy.json

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

青字の部分が割り当てるアクセス許可名となります。

(2) ポリシーの作成

以下のコマンドを入力して、ポリシーを作成します。

```
> aws iot create-policy --policy-name <POLICY_NAME> --policy-document
file://policy.json --profile <PROFILE_NAME>
```

- <POLICY_NAME>には任意のポリシー名を入力します（例：rx65n_ota_demo_policy）

コマンドを実行すると、以下の様に表示されます。

作成したポリシー名は後の処理で使用するため控えておいてください。

3.3.2 Amazon S3 バケットの作成

Amazon S3 はオンラインストレージの Web サービスで、更新用ファームウェアを格納するために使用します。

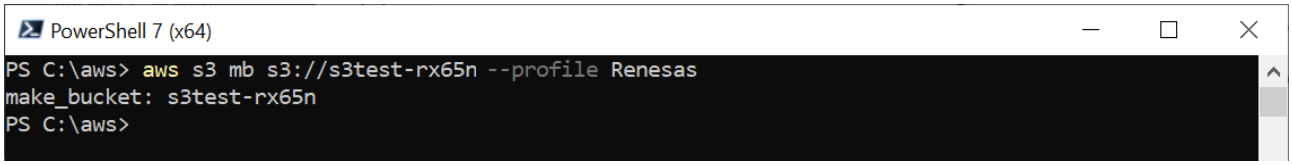
(1) S3 バケットの作成

以下のコマンドを入力してください。

```
> aws s3 mb s3://<BUCKET_NAME> --profile <PROFILE_NAME>
```

- <BUCKET_NAME>には任意のバケット名を入力します（例：s3test-rx65n）。

コマンドを実行すると、以下の様に表示されます。



設定したバケット名は、後の処理で使用するため控えておいてください。

【注１】 バケット名はグローバルで一意である必要があります。以下のようなエラーメッセージが表示された場合、すでに使用されている名前のため別の名前を使用してください。

```
make_bucket failed: s3://s3test-rx65n An error occurred
(BucketAlreadyExists) when calling the CreateBucket operation: The
requested bucket name is not available. The bucket namespace is shared
by all users of the system. Please select a different name and try
again.
```

【注２】バケット名は小文字、数字、ピリオド (.)、およびハイフン (-) のみが使用できます。

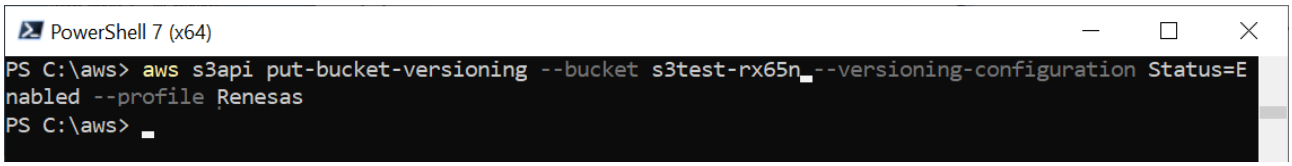
(2) S3 バケットへバージョニングの有効化

作成したバケットに格納するファイルをバージョン管理できるように設定します。
以下のコマンドを入力してください。

```
> aws s3api put-bucket-versioning --bucket <BUCKET_NAME> --versioning-
configuration Status=Enabled --profile <PROFILE_NAME>
```

- <BUCKET_NAME>には「(1)」で作成したバケット名を入力します。

コマンドを実行すると、以下の様に表示されます（実行結果は何も表示されません）。



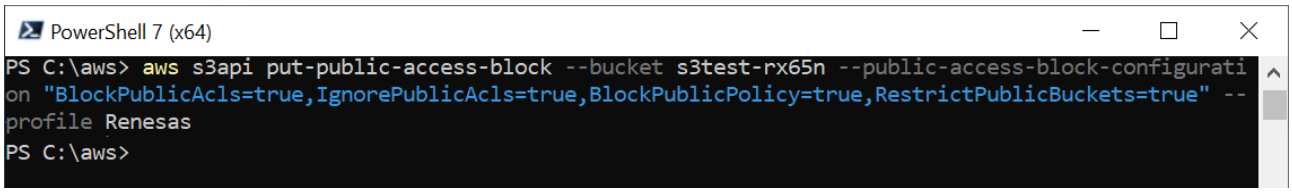
(3) S3 バケットへパブリックアクセスブロックの設定

作成したバケットへのパブリックからのアクセスを制限します。
以下のコマンドを入力してください。

```
> aws s3api put-public-access-block --bucket <BUCKET_NAME> --public-access-
block-configuration
"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPub
licBuckets=true" --profile <PROFILE_NAME>
```

- *<BUCKET_NAME>*には「(1)」で作成したバケット名を入力します。

コマンドを実行すると、以下の様に表示されます（実行結果は何も表示されません）。



3.3.3 IAM ユーザーに OTA の実行権限を割り当てる

AWS IAM サービスを使用して OTA 更新ジョブを作成するためにアクセス権限が付与されたロールを作成します。

(1) OTA の実行権限を設定した json ファイルの作成

OTA の実行権限設定ファイルを作成します。
テキストエディタで「3.1.5」で作成した作業フォルダに「Test-Role-Trust-Policy.json」というファイルを作成し、以下のコードを入力します。

- Test-Role-Trust-Policy.json

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

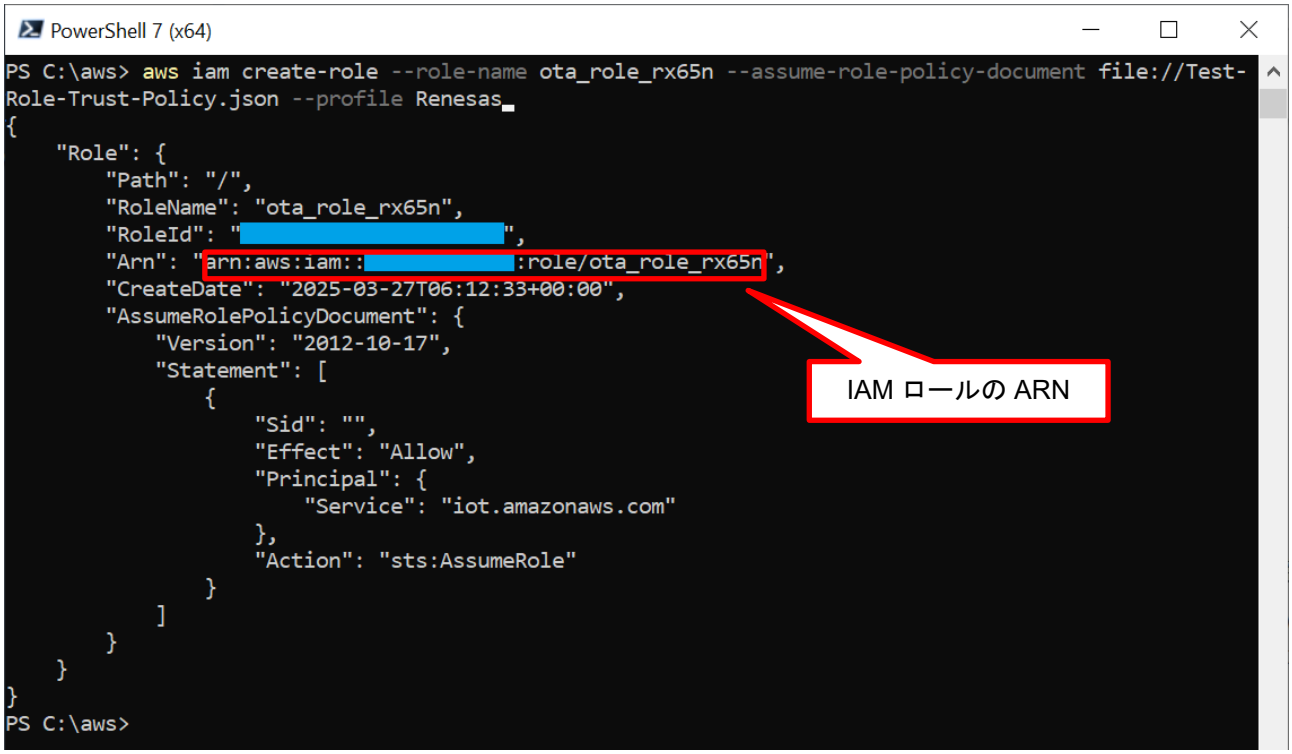
(2) IAM ロールの作成

OTA に使用する IAM ロールを作成します。
以下のコマンドを入力してください。

```
> aws iam create-role --role-name <ROLE_NAME> --assume-role-policy-document
file://Test-Role-Trust-Policy.json --profile <PROFILE_NAME>
```

- <ROLE_NAME>には任意のロール名を入力します（例：ota_role_rx65n）。

コマンドを実行すると、以下の様に表示されます。



ここで入力したロール名と、表示された IAM ロールの Amazon Resource Name（ARN）（Arn：上図の赤枠内）は後の処理で使用するため、控えておいて下さい。

(3) IAM ロールへの管理ポリシーのアタッチ

AWS IoT と FreeRTOS OTA の管理ポリシーを IAM ロールにアタッチします。
以下の 4 つの管理ポリシーを IAM ロールにアタッチします。

- AWSIoTThingsRegistration
- AWSIoTRuleActions
- AWSIoTLogging
- AmazonFreeRTOSOTAUpdate

以下の 4 つのコマンドを順に入力してください。

```
> aws iam attach-role-policy --role-name <ROLE_NAME> --policy-arn
arn:aws:iam::aws:policy/service-role/AWSIoTThingsRegistration --profile
<PROFILE_NAME>
```

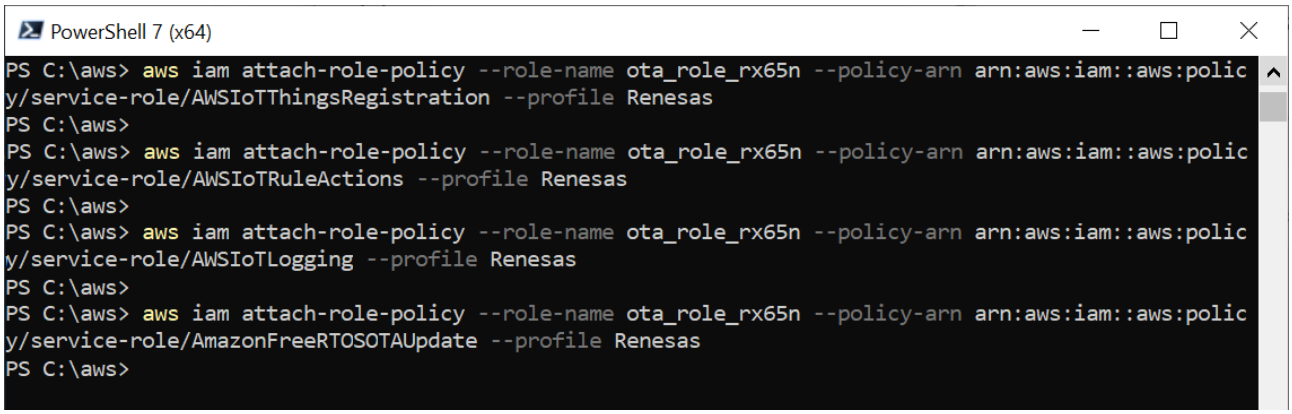
```
> aws iam attach-role-policy --role-name <ROLE_NAME> --policy-arn
arn:aws:iam::aws:policy/service-role/AWSIoTRuleActions --profile
<PROFILE_NAME>
```

```
> aws iam attach-role-policy --role-name <ROLE_NAME> --policy-arn
arn:aws:iam::aws:policy/service-role/AWSIoTLogging --profile <PROFILE_NAME>
```

```
> aws iam attach-role-policy --role-name <ROLE_NAME> --policy-arn
arn:aws:iam::aws:policy/service-role/AmazonFreeRTOSOTAUpdate --profile
<PROFILE_NAME>
```

- <ROLE_NAME>には「(2)」で作成した IAM ロール名を入力します。

コマンドを実行すると、以下の様に表示されます（それぞれの実行結果は何も表示されません）。



(4) AWS 各種サービスに IAM ロールを渡す許可を設定

インラインポリシーを使用して、AWS 各種サービスに IAM ロールを渡す許可を設定します。

(a) インラインポリシーを設定した json ファイルの作成

アクセス許可を設定したインラインポリシーのファイルを作成します。
テキストエディタで「3.1.5」で作成した作業フォルダに「inline-policy1.json」というファイルを作成し、以下のコードを入力します。

- inline-policy1.json

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "*"
        }
    ]
}
```

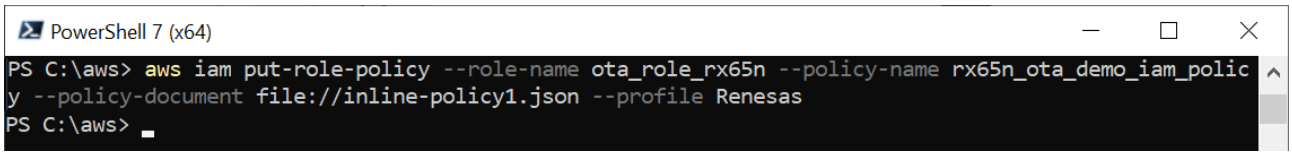
(b) IAM ロールへインラインポリシーをアタッチ

IAM ロールへインラインポリシーをアタッチします。
以下のコマンドを入力してください。

```
> aws iam put-role-policy --role-name <ROLE_NAME> --policy-name <POLICY_NAME>
--policy-document file://inline-policy1.json --profile <PROFILE_NAME>
```

- <ROLE_NAME>には「(2)」で作成した IAM ロール名を入力します。
- <POLICY_NAME>には任意のポリシー名を入力します（例：rx65n_ota_demo_iam_policy）。

コマンドを実行すると、以下の様に表示されます（実行結果は何も表示されません）。



(5)　Amazon S3 へのアクセス許可を IAM ロールへ設定

インラインポリシーを使用して、更新ファームウェアを格納する Amazon S3 へのアクセス許可を IAM ロールへ追加します。

(a)　インラインポリシーを設定した json ファイルの作成

アクセス許可を設定したインラインポリシーのファイルを作成します。
テキストエディタで「3.1.5」で作成した作業フォルダに「inline-policy2.json」というファイルを作成し、以下のコードを入力します。

- inline-policy2.json

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

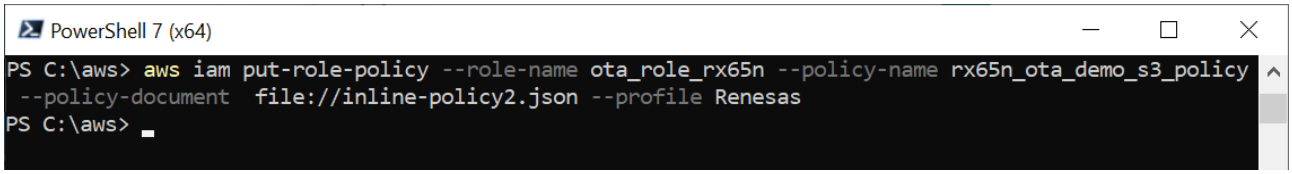
(b)　IAM ロールへインラインポリシー（Amazon S3）をアタッチ

IAM ロールへインラインポリシーをアタッチします。
以下のコマンドを入力してください。

```
> aws iam put-role-policy --role-name <ROLE_NAME> --policy-name <POLICY_NAME>
--policy-document  file://inline-policy2.json --profile <PROFILE_NAME>
```

- <ROLE_NAME>には「(2)」で作成した IAM ロール名を入力します。
- <POLICY_NAME>には任意のポリシー名を入力します（例：rx65n_ota_demo_s3_policy）。

コマンドを実行すると、以下の様に表示されます（実行結果は何も表示されません）。



3.3.4　デバイス（モノ）を AWS IoT に登録

AWS IoT Core サービスを使用して、接続する AWS IoT Core のデバイス（モノ）と証明書を作成し、モノに証明書を登録します。

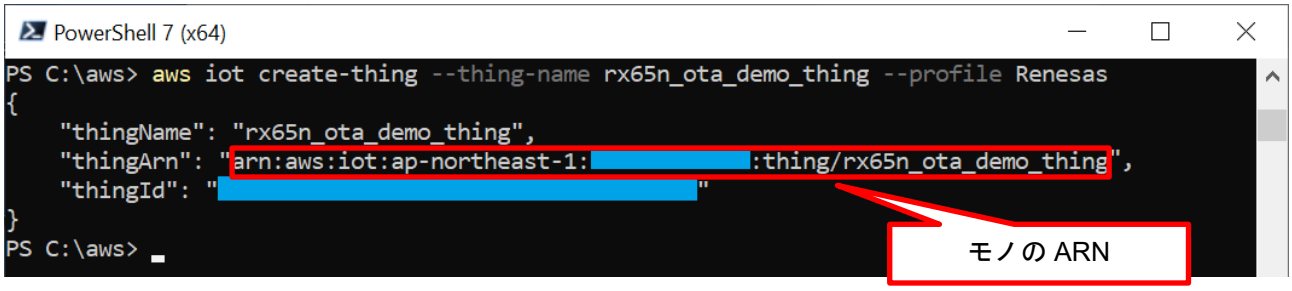
(1)　デバイス（モノ）の作成

以下のコマンドを入力してモノを作成します

```
> aws iot create-thing --thing-name <THING_NAME> --profile <PROFILE_NAME>
```

- <THING_NAME>には任意のモノの名前を入力します。（例：rx65n_ota_demo_thing）

コマンドを実行すると、以下の様に表示されます。



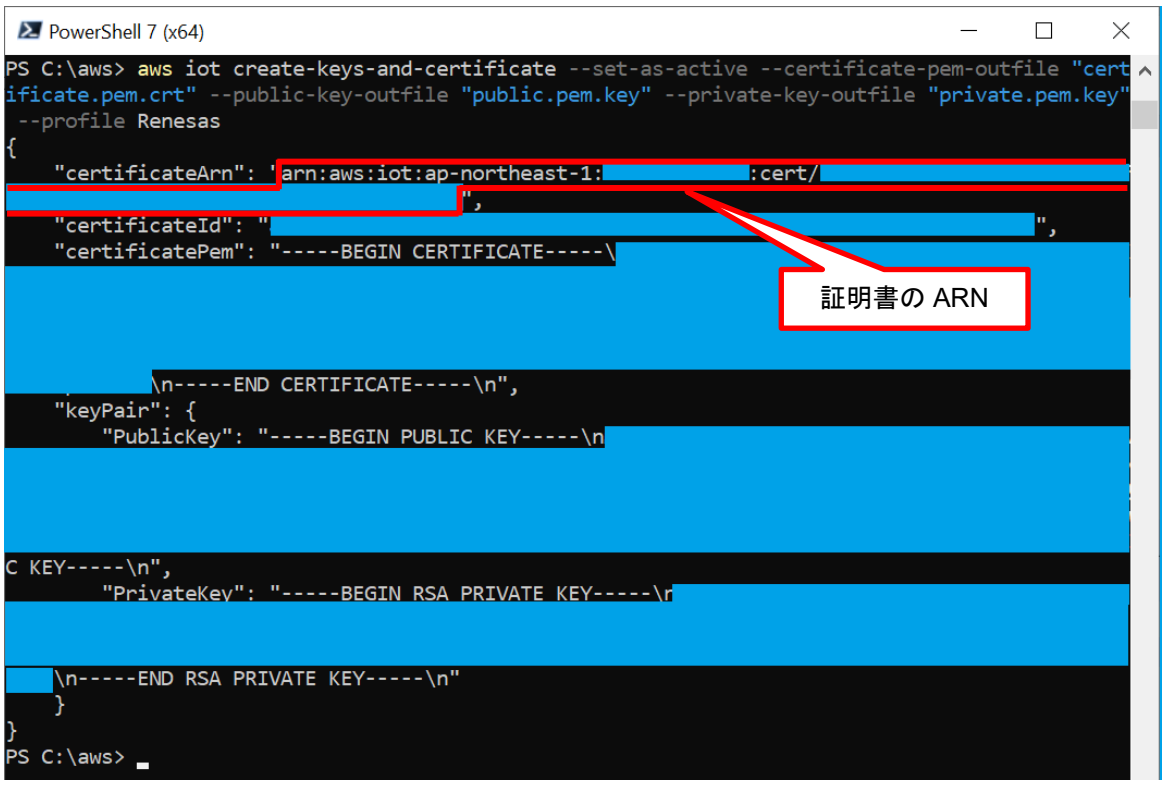
ここで入力したモノの名前と、表示されたモノの ARN（thingArn：上図の赤枠内）は後の処理で使用するため、控えておいて下さい。

(2)　デバイス証明書の作成

デバイス証明書を作成してアクティブ化し、デバイス証明書と公開鍵・秘密鍵のファイルをダウンロードします。
以下のコマンドを入力してください。

```
> aws iot create-keys-and-certificate --set-as-active --certificate-pem-
outfile "certificate.pem.crt" --public-key-outfile "public.pem.key" --private-
key-outfile "private.pem.key" --profile <PROFILE_NAME>
```

コマンドを実行するとデバイス証明書・鍵が作成され、以下の様に表示されます（表示例の文字列の長さは省略しています）。



表示されたデバイス証明書の ARN（certificateArn：上図の赤枠内）は後の処理で使用するため控えておいて下さい。

証明書作成コマンド実行後は以下の 3 つのファイルが「3.1.5」で作成した作業フォルダにダウンロードされます。

ファイル名	内容
certificate.pem.crt	モノに登録されたデバイス証明書
public.pem.key	モノに登録された公開鍵
private.pem.key	モノに登録された秘密鍵

証明書、秘密鍵はデバイス（モノ）における、パスワードに相当します。デバイスに証明書、秘密鍵を登録することで、これらを使用してデバイスが AWS に接続可能となります。

【注】デバイス証明書・公開鍵・秘密鍵は証明書作成時にしかダウンロードできません。
また、これらのファイルは安全な場所に保管し、漏洩しないように注意してください。

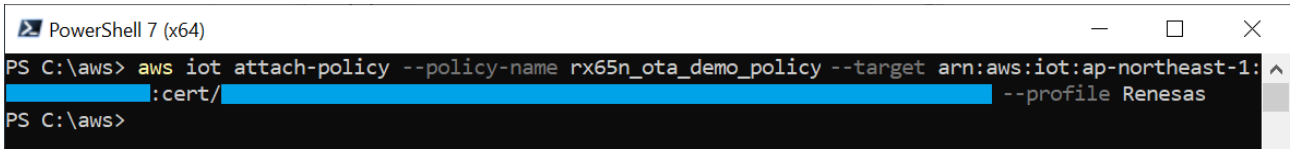
(3) デバイス証明書にポリシーをアタッチ

作成したデバイス証明書にポリシーをアタッチします。
以下のコマンドを入力して下さい。

```
> aws iot attach-policy --policy-name <POLICY_NAME> --target <CERTIFICATE_ARN>
--profile <PROFILE_NAME>
```

- <POLICY_NAME>には「3.3.1(2)」で作成したポリシー名を入力します。
- <CERTIFICATE_ARN>には「(2)」で作成したデバイス証明書の ARN を入力します。

コマンドを実行すると、以下の様に表示されます（実行結果は何も表示されません）。



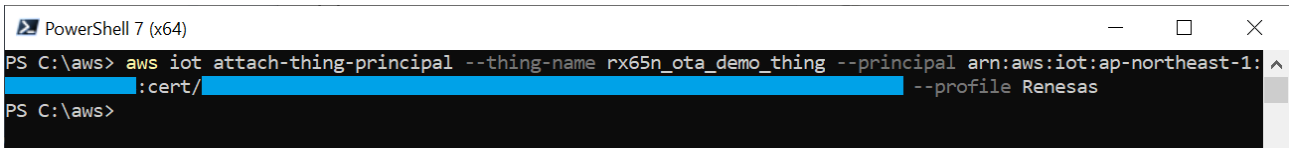
(4) モノにデバイス証明書をアタッチ

作成したモノにデバイス証明書をアタッチします。
以下のコマンドを実行してください。

```
> aws iot attach-thing-principal --thing-name <THING_NAME> --principal
<CERTIFICATE_ARN> --profile <PROFILE_NAME>
```

- <THING_NAME>には「(1)」で作成したモノの名前を入力します。
- <CERTIFICATE_ARN>には「(2)」で作成したデバイス証明書の ARN を入力します。

コマンドを実行すると、以下の様に表示されます（実行結果は何も表示されません）。

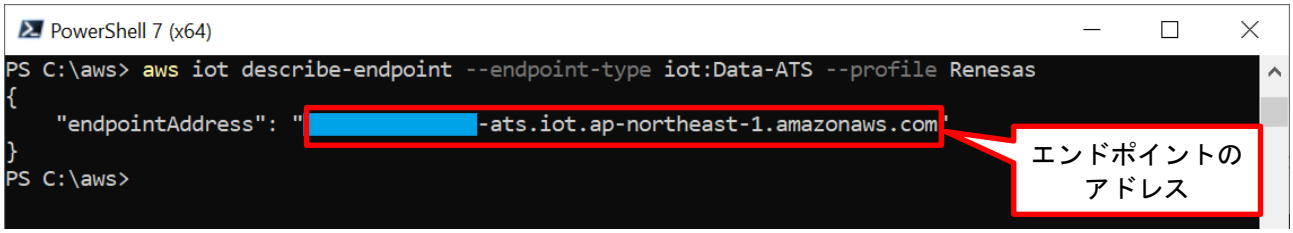


(5) エンドポイント（ドメイン）の確認

エンドポイントはデバイス（モノ）における接続先（URL）に相当します。デバイスにエンドポイントを登録することで、デバイスは指定したエンドポイントに接続します。
以下のコマンドを入力してください。

```
> aws iot describe-endpoint --endpoint-type iot:Data-ATS --profile
<PROFILE_NAME>
```

コマンドを実行すると、以下の様に表示されます。



表示されたエンドポイントのアドレス（endpointAddress：上図の赤枠内）は後の処理で使用するため控えておいて下さい。

4. デバイスの設定

OTA の実行時にファームウェアのコード署名検証に用いる証明書の作成を行います。
以下の手順で証明書を作成してください。

【注】ここで作成する証明書は「3.3.4(2)」で作成した証明書とは別のものとなります

4.1 鍵ペアと証明書の生成

(1) Win64 OpenSSL Command Prompt の起動

スタートメニューから Win64 OpenSSL Command Prompt を起動します。

(2) ECDSA の CA 秘密鍵の作成

OpenSSL を使用し、ECDSA の CA 秘密鍵を作成します。
以下のコマンドを実行します。

```
> openssl ecparam -genkey -name secp256r1 -out ca.key
```

コマンドを実行すると、以下の様に表示されます。

```
C:¥openssl>openssl ecparam -genkey -name secp256r1 -out ca.key
using curve name prime256v1 instead of secp256r1
```

(3) CA 証明書の作成

作成した CA 秘密鍵から CA 証明書を作成します。
以下のコマンドを実行します。Country Name 以降は任意の文字列を入力してください。

```
> openssl req -x509 -sha256 -new -nodes -key ca.key -days 3650 -out ca.crt
```

コマンドを実行すると、以下の様に表示されます。

```
C:¥openssl>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
```

任意の文字列を入力する

(4) ECDSA の鍵ペアの作成

ECDSA の鍵ペアを作成します。
以下のコマンドを実行します。

```
> openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
```

コマンドを実行すると、以下の様に表示されます。

```
C:¥openssl>openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
using curve name prime256v1 instead of secp256r1
```

(5) ECDSA 鍵ペアから証明書署名要求を作成

作成した ECDSA の鍵ペアから証明書署名要求を作成します。
以下のコマンドを実行します。
Country Name 以降は任意の文字列を入力してください。最後の 2 行は空白のまま Enter を押してください。

```
> openssl req -new -sha256 -key secp256r1.keypair > secp256r1.csr
```

コマンドを実行すると、以下の様に表示されます。

```
C:¥openssl>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 []:
```

任意の文字列を入力する

空白のまま Enter を押す

(6) 証明書の作成

作成した証明書署名要求/CA 証明書/CA 秘密鍵から証明書を作成します。
以下のコマンドを実行します。

```
> openssl x509 -req -sha256 -days 3650 -in secp256r1.csr -CA ca.crt -CAkey
ca.key -CAcreateserial -out secp256r1.crt
```

コマンドを実行すると、以下の様に表示されます。

```
C:¥openssl>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, emai
lAddress = Tarou.Renesas@sample.com
Getting CA Private Key
```

(7) ECDSA 鍵ペアから秘密鍵の抽出

ECDSA の鍵ペアから秘密鍵を抽出します。
以下のコマンドを実行します。

```
> openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey
```

コマンドを実行すると、以下の様に表示されます。

```
C:¥openssl>openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey
read EC key
writing EC key
```

(8) ECDSA 鍵ペアから公開鍵の抽出

ECDSA の鍵ペアから公開鍵を抽出します。
以下のコマンドを実行します。

```
> openssl ec -in secp256r1.keypair -outform PEM -pubout -out
secp256r1.publickey
```

コマンドを実行すると、以下の様に表示されます。

```
C:¥openssl>openssl ec -in secp256r1.keypair -outform PEM -pubout -out secp256r1.publickey
read EC key
writing EC key
```

(9) 生成ファイルの確認

本項の操作で以下の 4 つのファイルが作成されます。

- ECDSA 公開鍵： secp256r1.publickey
- ECDSA 秘密鍵： secp256r1.privatekey
- ECDSA 証明書（鍵ペア証明書）： secp256r1.crt
- ECDSA 証明書チェーン： ca.crt

ここで作成した上記のファイルは、プロジェクトの作成時および AWS の OTA ジョブ作成時の設定に使用します。

4.2　初期バージョンのファームウェア構築

初期バージョンのファームウェアの構築を行います。

4.2.1　プロジェクトのインポート

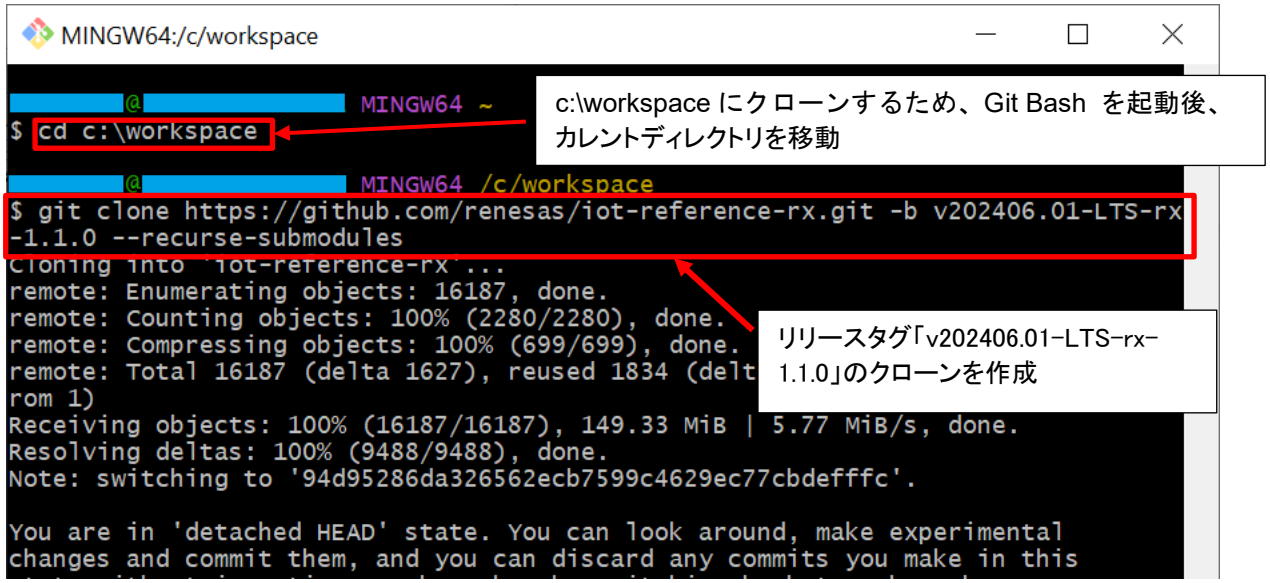
(1)　デモプロジェクトのクローン

GitHub (https://github.com/renesas/iot-reference-rx) からデモプロジェクトを任意のフォルダへクローンします。本アプリケーションノートでは、Git for Windows を使用した場合のクローン方法を説明します。
リンクを参照し、Git for Windows を事前にインストールしてください。

Git Bash を起動し、以下のコマンドを実行してください。
以下の例では、c:\workspace にクローンしています。

```
> cd c:\workspace
> git clone https://github.com/renesas/iot-reference-rx.git -b v202406.01-LTS-rx-1.1.0 --recurse-submodules
```

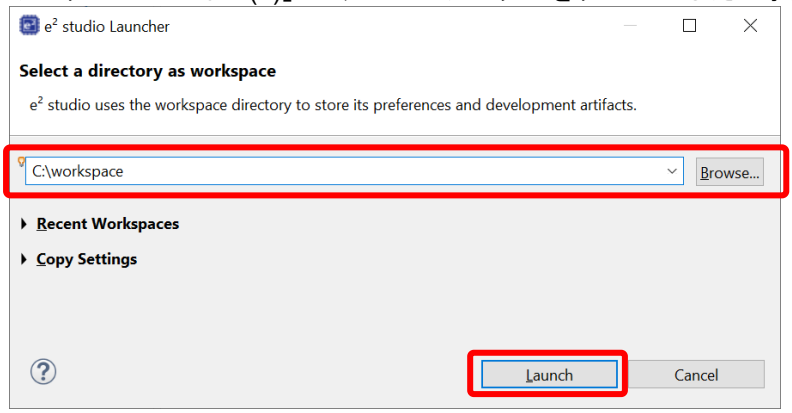
コマンドを実行すると、以下の様に表示されます。



【注１】「iot-reference-rx」の最新バージョンは GitHub でご確認ください。
【注２】e² studio に制限があるため、クローン先のパス長（任意のフォルダ名を含む）は 35 文字以内としてください。36 文字以上を指定するとプロジェクトのビルド時にエラーとなります。
またパスに日本語が存在するとエラーとなる場合がありますので、名前は英数字で入力してください。

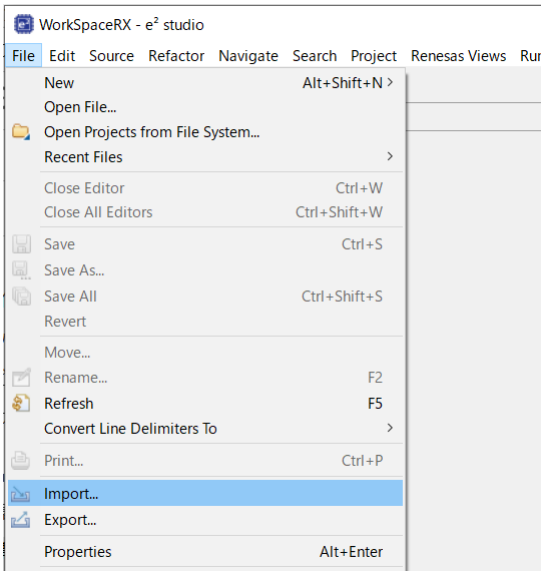
(2)　e² studio ワークスペースの作成

e² studio を起動して新しいワークスペースを作成してください。
ワークスペースは「(1)」でデモプロジェクトをクローンしたフォルダを指定します。



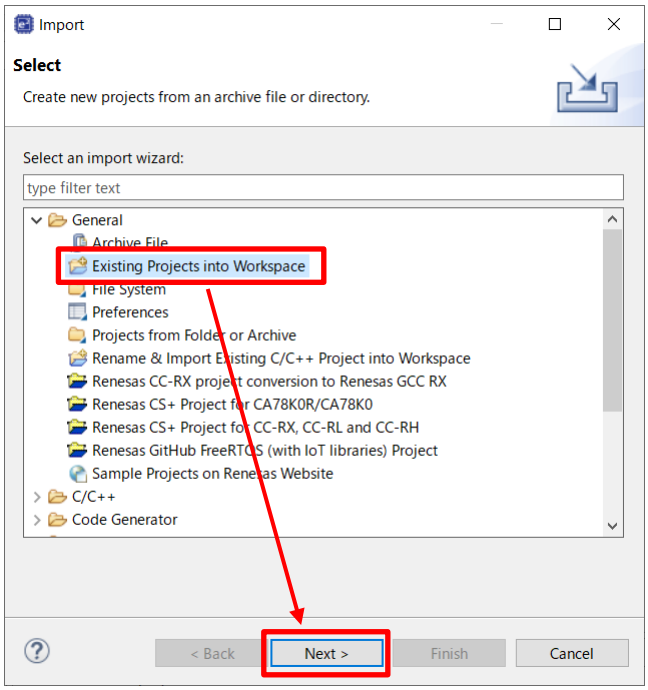
(3)　ファイルのインポートを実行

[File（ファイル）] ⇒ [Import（インポート）] を選択します。



(4)　インポート方法の選択

Existing Projects into Workspace (既存プロジェクトをワークスペースへ)を選択し、[Next]ボタンを押下して下さい。

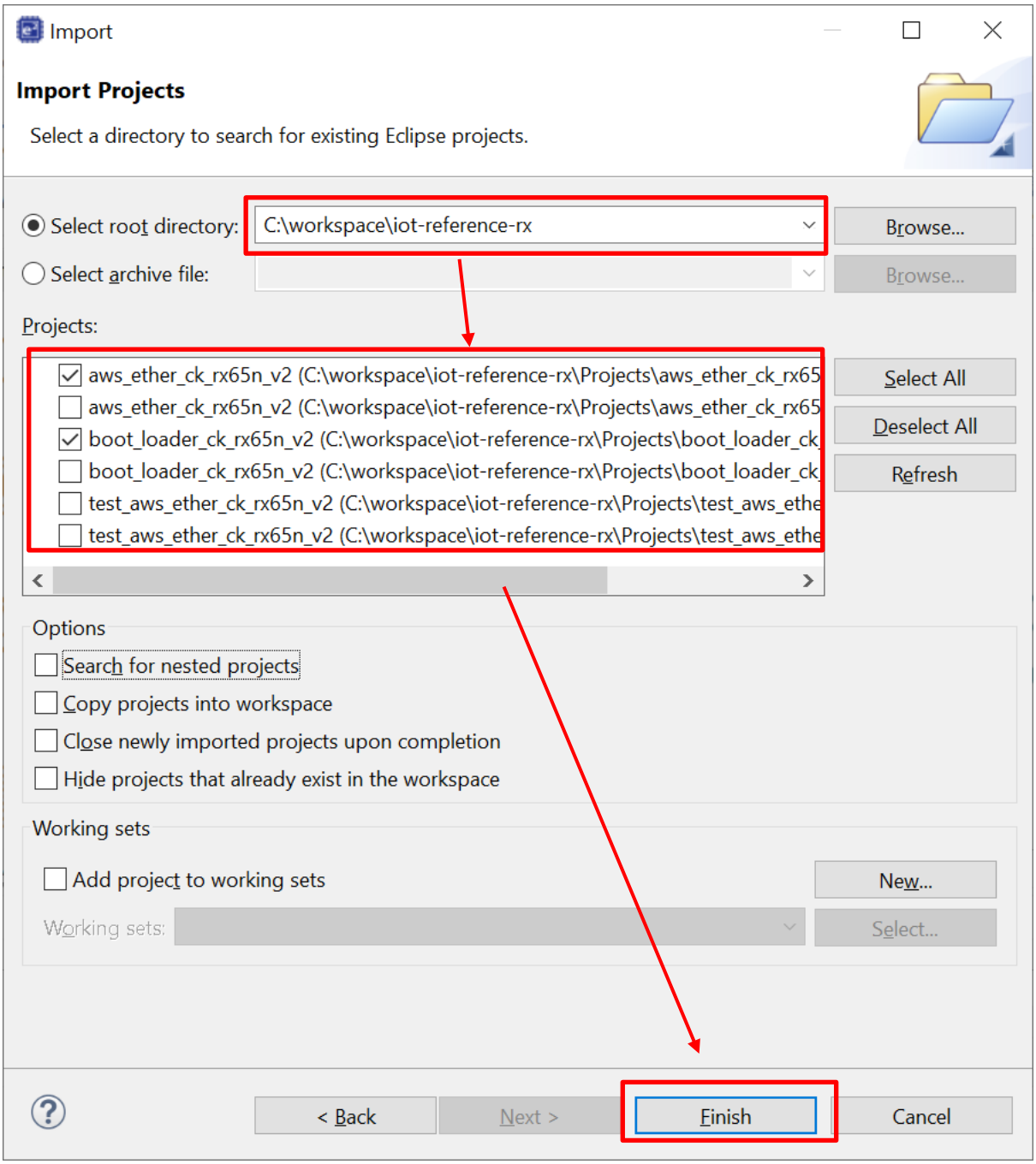


(5)　インポートするプロジェクトの選択

「Select root directory（ルート・ディレクトリーの選択）」にて(1)にてクローンしたフォルダを選択し、以下のプロジェクトにチェックして[Finish（終了）]をクリックしてください。
コンパイラはカッコ内のパス名を確認して選択してください。

- aws_ether_ck_rx65n_v2（CC-RX）
- boot_loader_ck_rx65n_v2（CC-RX）

本アプリケーションノートでは Ethernet の CC-RX コンパイラ用のプロジェクトを例に手順を説明します。



以下にインポート可能なプロジェクトの一覧を示します。
作成したいコネクティビティ／コンパイラのプロジェクトを選択することができます。

パス	内容	コンパイラ
C:\workspace\iot-reference-rx \Projects\aws_ether_ck_rx65n_v2\e2studio_ccrx	デモプロジェクト： Ethernet	CC-RX
C:\workspace\iot-reference-rx \Projects\aws_ether_ck_rx65n_v2\e2studio_gcc		GCC
C:\workspace\iot-reference-rx \Projects\aws_ryz014a _ck_rx65n_v2\e2studio_ccrx	デモプロジェクト： Cellular（RYZ014A）	CC-RX
C:\workspace\iot-reference-rx \Projects\aws_ryz014a _ck_rx65n_v2\e2studio_gcc		GCC
C:\workspace\iot-reference-rx \Projects\aws_da16600_ck_rx65n_v2\e2studio_ccrx	デモプロジェクト： Wi-Fi（DA16600）	CC-RX
C:\workspace\iot-reference-rx \Projects\aws_da16600_ck_rx65n_v2\e2studio_gcc		GCC
C:\workspace\iot-reference-rx \Projects\boot_loader_ck_rx65n_v2\e2studio_ccrx	ブートローダプロジェクト	CC-RX
C:\workspace\iot-reference-rx \Projects\boot_loader_ck_rx65n_v2\e2studio_gcc		GCC

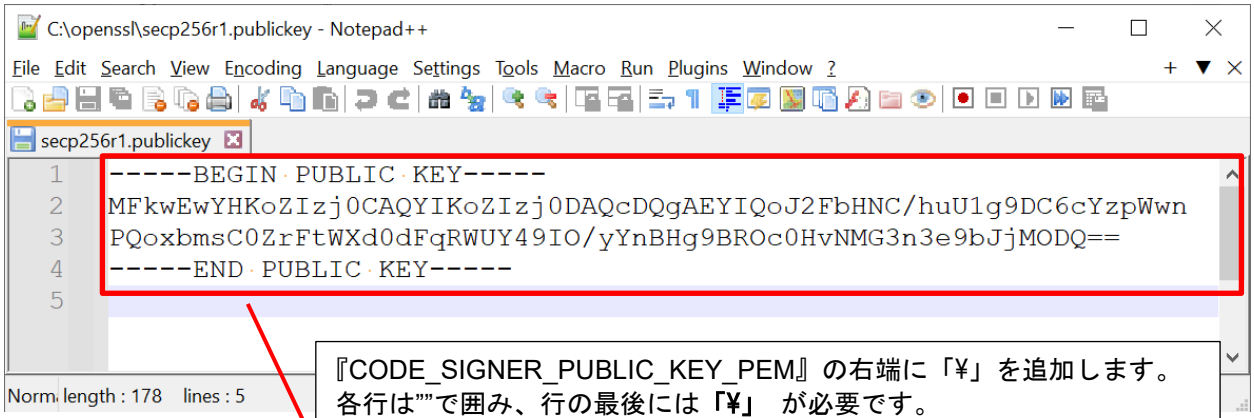
【注】先頭が「test_」となっているプロジェクトはバリデーションテスト用のプロジェクトです。
通常は使用しません。

4.2.2　プロジェクト設定

(1)　boot_loader_ck_rx65n_v2 プロジェクトに公開鍵を設定

ブートローダプロジェクトへ公開鍵を設定します。
「4.1(8)」で作成した secp256r1.publickey の内容をコピーし、以下ファイルの
「CODE_SIGNENR_PUBLIC_KEY_PEM」に公開鍵を張り付けます。

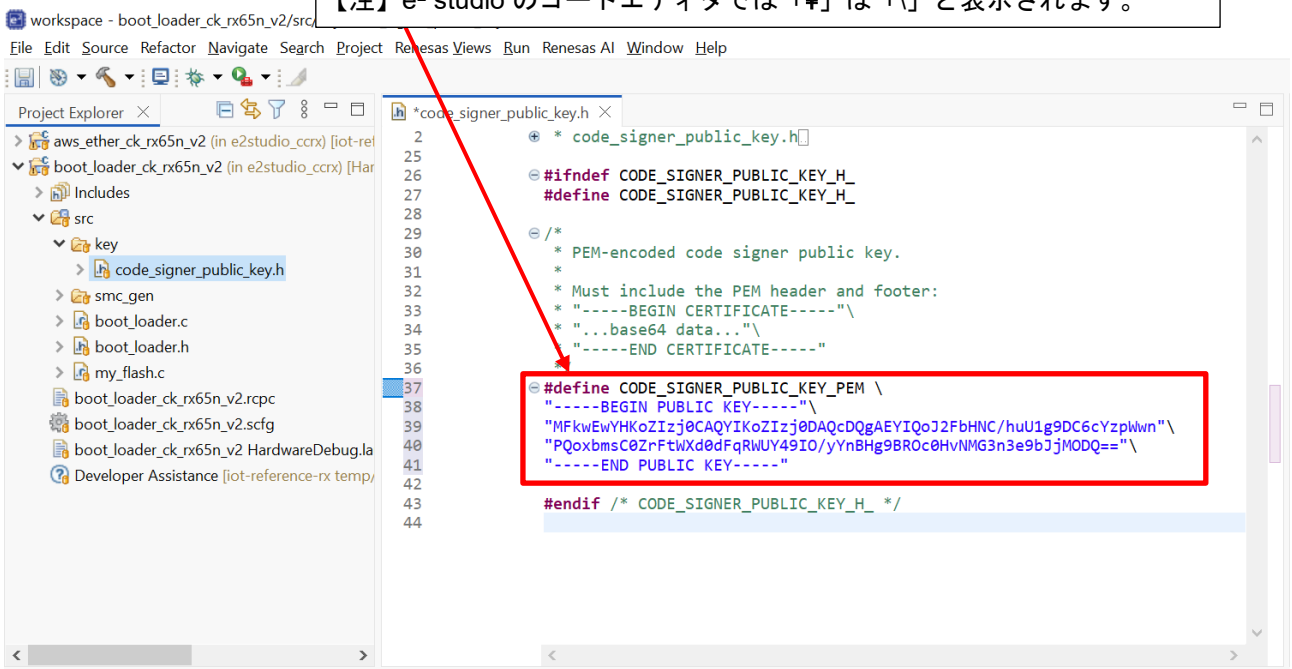
- CC-RX 版：　\boot_loader_ck_rx65n_v2\e2studio_ccrx\src\key\code_signer_public_key.h
- GCC 版：　\boot_loader_ck_rx65n_v2\e2studio_gcc\src\key\code_signer_public_key.h



『CODE_SIGNER_PUBLIC_KEY_PEM』の右端に「¥」を追加します。
各行は""で囲み、行の最後には**「¥」** が必要です。
忘れずに入力してください。

（例：”xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx”**¥**）

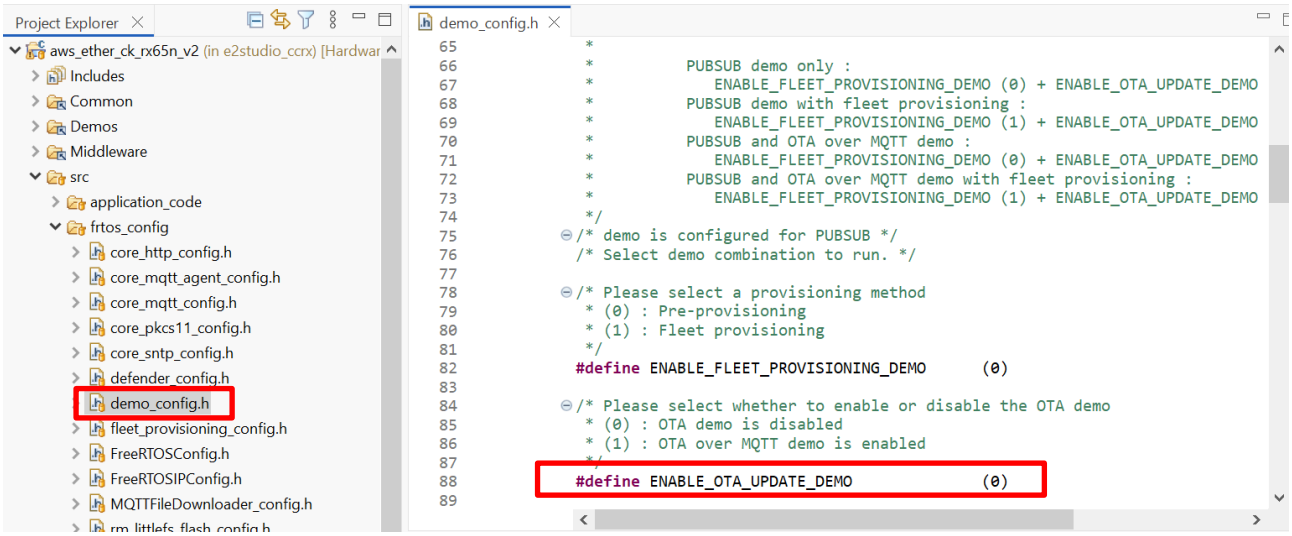
ただし、最下行の”-----END PUBLIC KEY-----"には「¥」を入力しないでください。
【注】e² studio のコードエディタでは「¥」は「\」と表示されます。



(2)　OTA アップデートデモの定義を許可

プロジェクトを OTA アップデートデモで動作するように設定を行います。
以下のファイルの ENABLE_OTA_UPDATE_DEMO 定義を 1 (許可)に設定してください。
（デフォルト 0）

- CC-RX 版：　\aws_ether_ck_rx65n_v2\e2studio_ccrx\src\frtos_config\demo_config.h
- GCC 版：　\aws_ether_ck_rx65n_v2\e2studio_gcc\src\frtos_config\demo_config.h



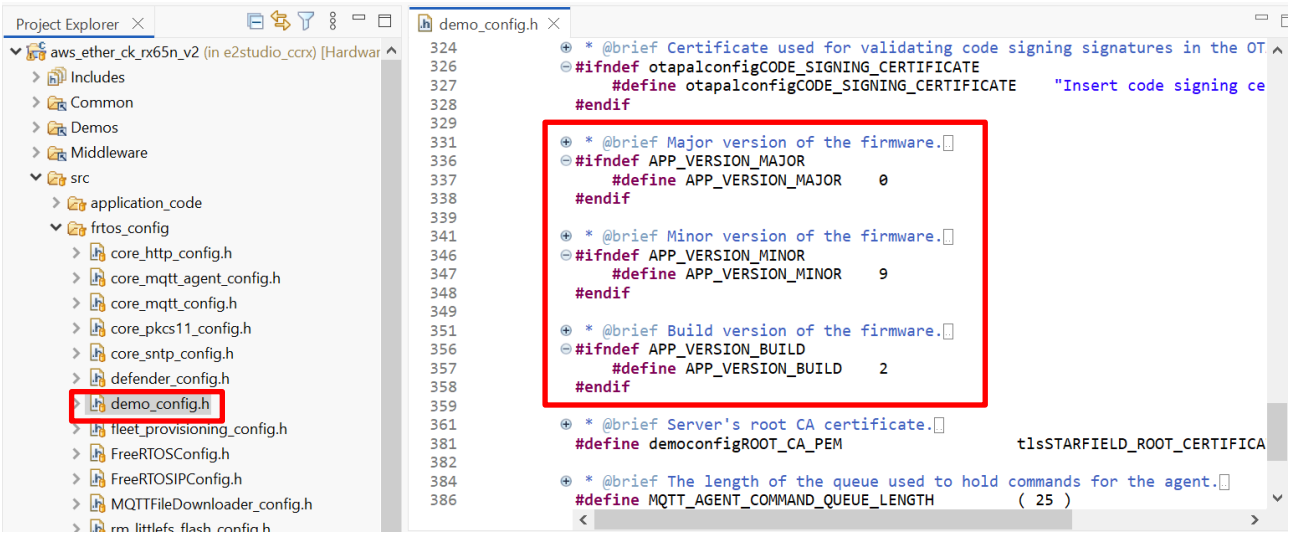
(3)　プロジェクトの初期バージョンの確認

プロジェクトの初期バージョンが 0.92 であることを確認します。
以下のファイルを開いてバージョンを確認します。

- CC-RX 版：　\aws_ether_ck_rx65n_v2\e2studio_ccrx\src\frtos_config\demo_config.h
- GCC 版：　\aws_ether_ck_rx65n_v2\e2studio_gcc\src\frtos_config\demo_config.h

バージョンは以下に設定されていることを確認して下さい。

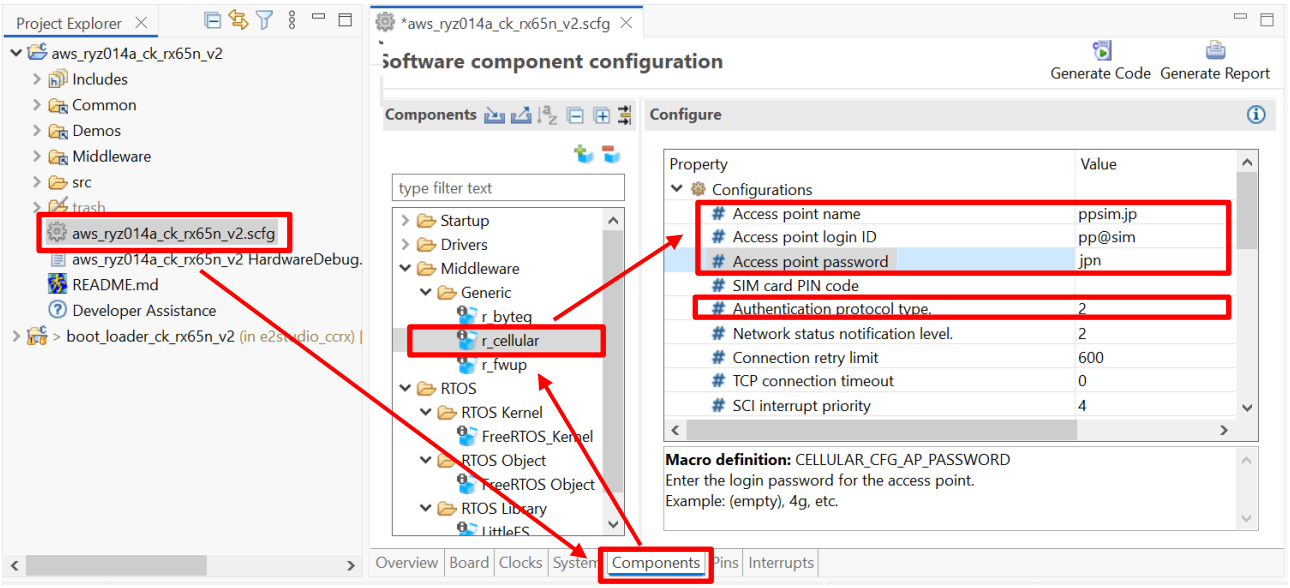
- APP_VERSION_MAJOR：　0
- APP_VERSION_MINOR：　9
- APP_VERSION_BUILD：　2



(4)　RYZ014A Cellular モジュールの設定

AWS 接続にセルラーモジュールを使用する場合は、RYZ014A Cellular FIT モジュール(r_cellular)へ設定を行います。

[aws_ryz014a_ck_rx65n_v2.scfg]を開き、[Components（コンポーネント）]タブを選択し、[r_cellular]の[Access point name]、[Access point login ID]、[Access point password]、[Authentication protocol type]を SIM カードに合わせて設定してください。



【注】DA16600 Wi-Fi モジュールによる Wi-Fi ネットワークの設定方法は GitHub「Settings of Wi-Fi network 」を参照してください。また、カントリーコードと GMT タイムゾーンの設定については「Settings of Country code and GMT timezone」を必要に応じて参照してください。

4.2.3　初期ファームウェアの作成

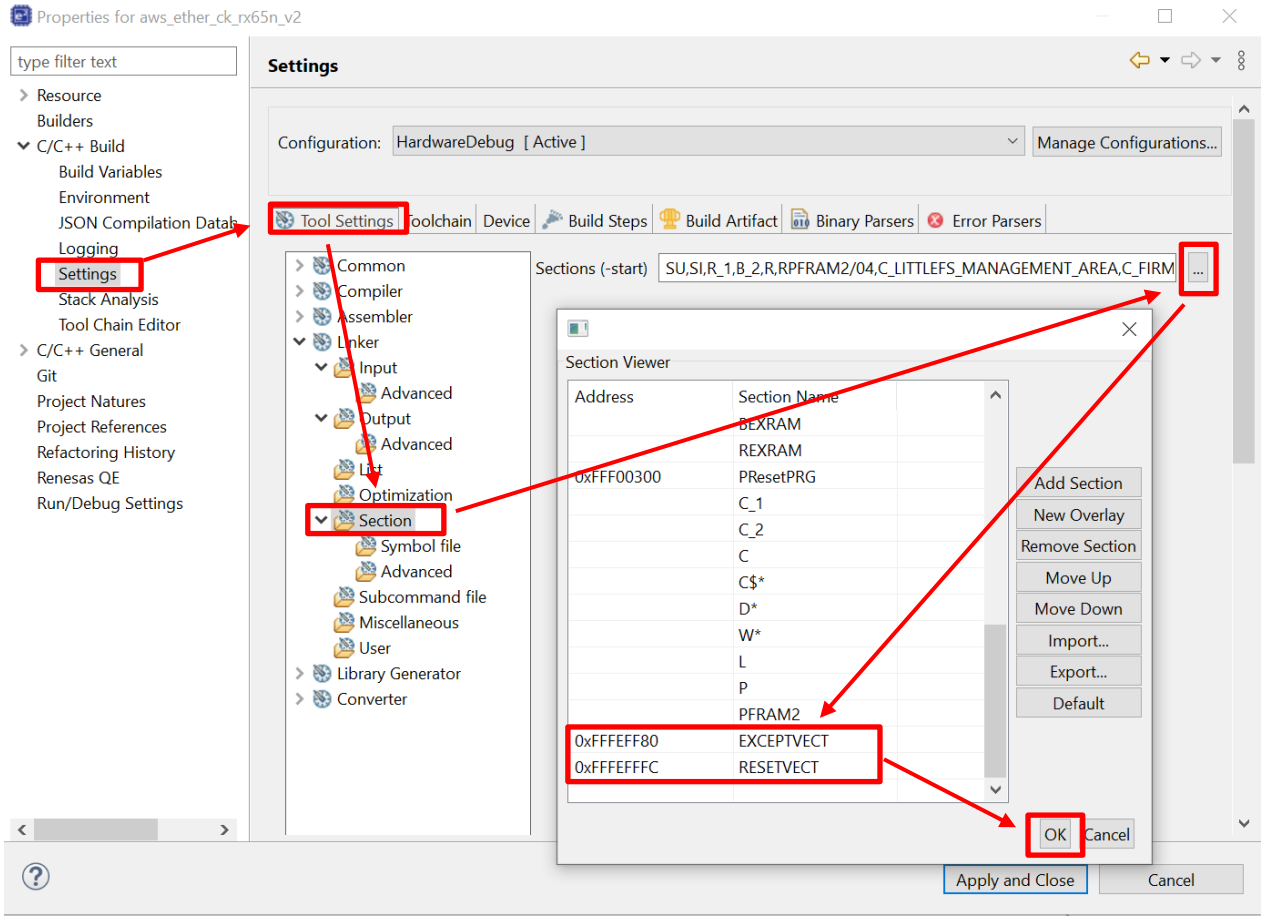
ブートローダ(boot_loader_ck_rx65n_v2)とファームウェア(aws_ether_ck_rx65n_v2)を結合して初期ファームウェアを作成します。

(1)　ファームウェア(aws_ether_ck_rx65n_v2)のベクタ変更とビルド

ファームウェアのベクタ変更の確認とビルドを実行します。
aws_ether_ck_rx65n_v2 のプロジェクトから、[Project（プロジェクト）] > [Properties（プロパティ）]を選択します。

[C/C++ Build（C++ビルド）] > [Settings（設定）]から、[Tool Settings（ツール設定）]タブの[Linker] > [Section（セクション）]から Section Viewer（セクション・ビューワー）を開き、以下のセクションのアドレスを割り当てます。

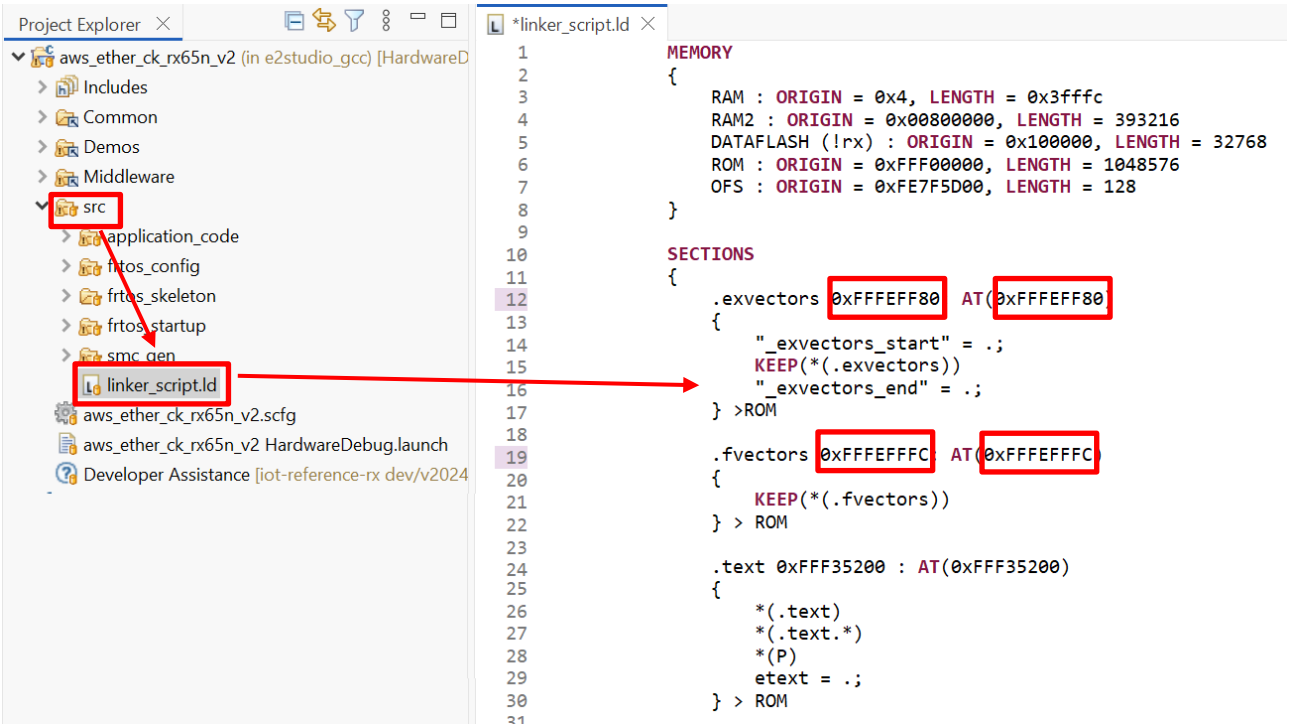
- EXCEPTVECT：0xFFFEFF80
- RESETVECT：0xFFFEFFFC



ベクタ設定が完了したら、e² studio で aws_ether_ck_rx65n_v2 プロジェクトのビルドを行いファームウェアの作成を行います。

GCC プロジェクトを構築する際はベクタ設定の変更方法が異なります。
[Project Explorer（プロジェクト・エクスプローラー）]より、src\linker_script.ld を開いて以下アドレスを変更してください。

- .exvectors：0xFFFEFF80（2 か所）
- .fvectors：0xFFFEFFFC（2 か所）



ベクタ設定が完了したら、e² studio で aws_ether_ck_rx65n_v2 プロジェクトのビルドを行いファームウェアの作成を行います。

(2) ブートローダ(bootloader)の作成

bootloader プロジェクトのビルドを行いブートローダの作成を行います。

(3) Renesas Image Generator を使用した初期ファームウェアの作成

Renesas Image Generator を使用して初期ファームウェアを生成します。
まず、Renesas Image Generator フォルダに以下のファイルを格納します。

- 4.2.3(1)でのビルド実施結果　　aws_ether_ck_rx65n_v2.mot
- 4.2.3(2)でのビルド実施結果　　boot_loader_ck_rx65n_v2.mot
- 4.1(7)で作成した秘密鍵　　secp256r1.privatekey

コマンドプロンプトを起動し、Renesas Image Generator フォルダへ移動して以下のコマンドを実行すると、userprog.mot ファイルが生成されます。
コマンドの実行からファイル作成には数分かかる場合がありますのでしばらくお待ちください。

python image-gen.py -iup *aws_ether_ck_rx65n_v2.mot* -ip *RX65N_DualBank_ImageGenerator_PRM.csv* -o *userprog* -ibp *boot_loader_ck_rx65n_v2.mot* -key *secp256r1.privatekey* -vt ecdsa -ff RTOS

(4) フラッシュ書き込みツール(Renesas Flash Programmer)のインストール

フラッシュ書き込みツールのダウンロードサイトにアクセスし、"Renesas Flash Programmer V3.19.00 Windows"をダウンロードしてインストールしてください。

(5) PC と CK-RX65N v2 の接続

「2.6 CK-RX65N v2 の接続」の CK-RX65N v2 の接続に従い PC と CK-RX65N v2 を接続します。

(6) Renesas Flash Programmer にてイレーズプロジェクトのオープン

Renesas Flash Programmer で[File（ファイル）] > [Open Project（プロジェクトを開く）]をクリックして「erase.rpj」を開きます。
イレーズプロジェクトは、本サンプルプログラムの以下フォルダにあります。

```
\Projects\aws_ether_ck_rx65n_v2\flash_project\erase_from_bank1
```

(7) デバイスのイレーズの実施

[Start（スタート）]ボタンを押下し、デバイスのイレーズを実施します。
Operation completed と表示されたらイレーズ完了です。

【注】「エラー(E3000107): デバイスが接続情報と一致しません」が出力された場合は、デバイス情報がすでに初期化されているため、次の「(8)」へ進んでください。

(8) Renesas Flash Programmer でフラッシュ書き込みプロジェクトのオープン

Renesas Flash Programmer で[File（ファイル）] > [Open Project（プロジェクトを開く）]をクリックして「flash_project.rpj」を開きます。

フラッシュ書き込みプロジェクトは、本サンプルプログラムの以下フォルダにあります。

```
\Projects\aws_ether_ck_rx65n_v2\flash_project
```

(9) 初期ファームウェアの選択

次に[Add/Remove Files（ファイルの追加と削除）]をクリックし、[Add File(s)（ファイルの追加）]ボタンを押下します。

ファイル選択のダイアログで「4.2.3(3)」で作成した初期ファームウェア(userprog.mot)を選択します。

(10) 初期ファームウェアの書き込み

[Start（スタート）]ボタンを押下して初期ファームウェアの書き込みを行います。
Operation completed と表示されたら初期ファームウェアの書き込みは完了です。

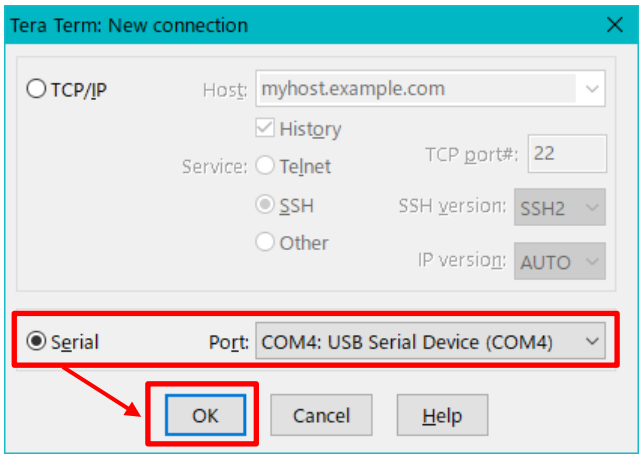
【注】「エラー(E3000107): デバイスが接続情報と一致しません」が出力された場合は、デバイス情報が初期化されていないため、「(7)」のイレーズ処理を実施して下さい。

4.2.4　AWS IoT 情報の登録

aws_ether_ck_rx65n_v2 を動作させて、AWS IoT の情報を Tera Term にて設定します。設定した情報はデータフラッシュに書き込まれます。

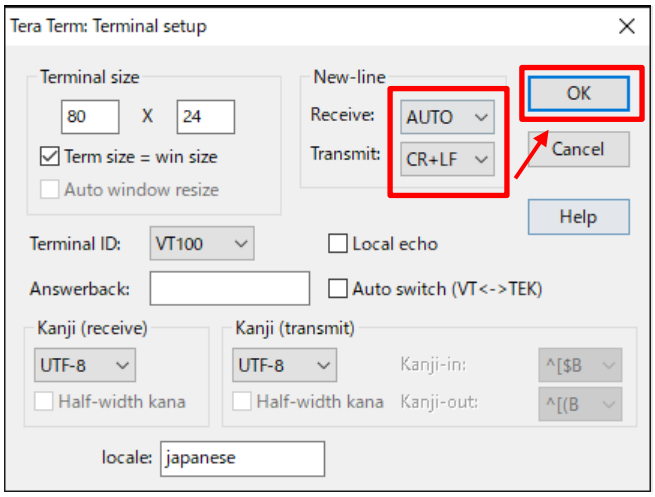
(1)　Tera Term の起動とシリアルポートの設定

Tera Term を起動してターゲットボードと接続するシリアルポートを設定します。
メニューの[File] > [New Connection…]から、[Serial]を選択して OK をクリックしてください。



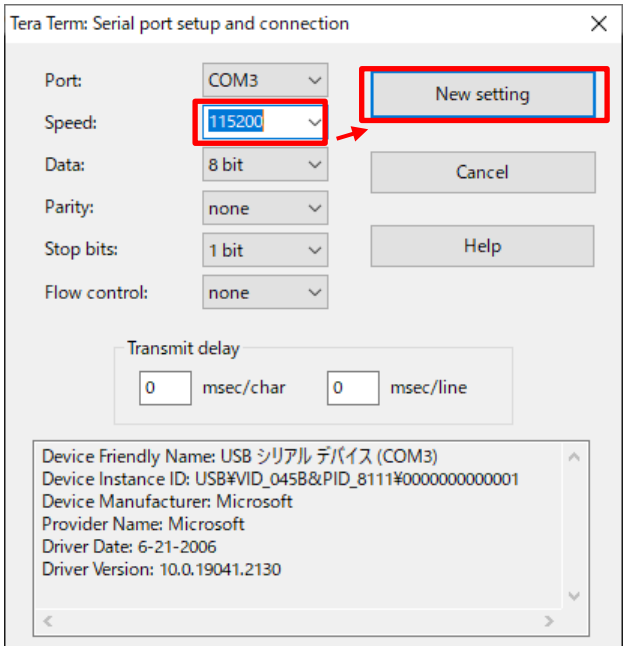
(2)　改行コードの設定

シリアルポートの送受信における改行コードの設定を行います。
メニューの Setup > Terminal…を開き、New-line の Receive を Auto、Transmit を CR+LF を選択して OK をクリックしてください。



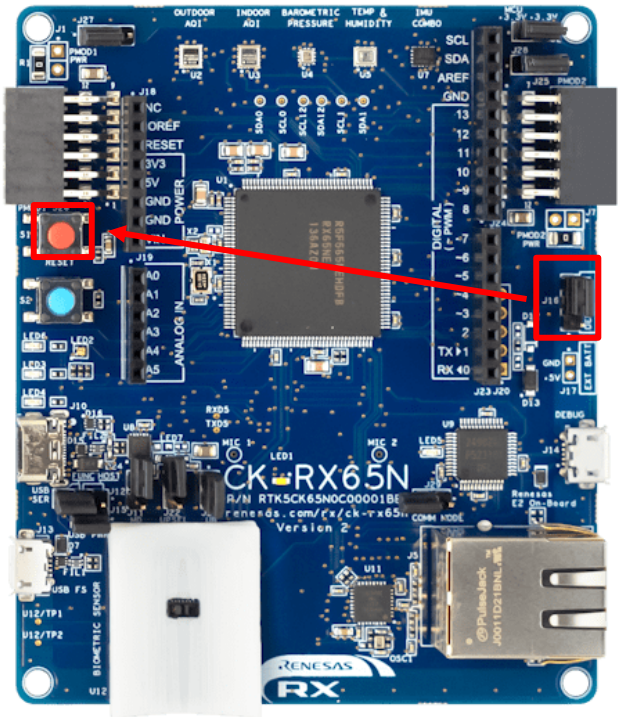
(3)　シリアル通信速度の設定

メニューの Setup > Serial port…を開いて Speed を 115200 に設定して New setting をクリックしてください。



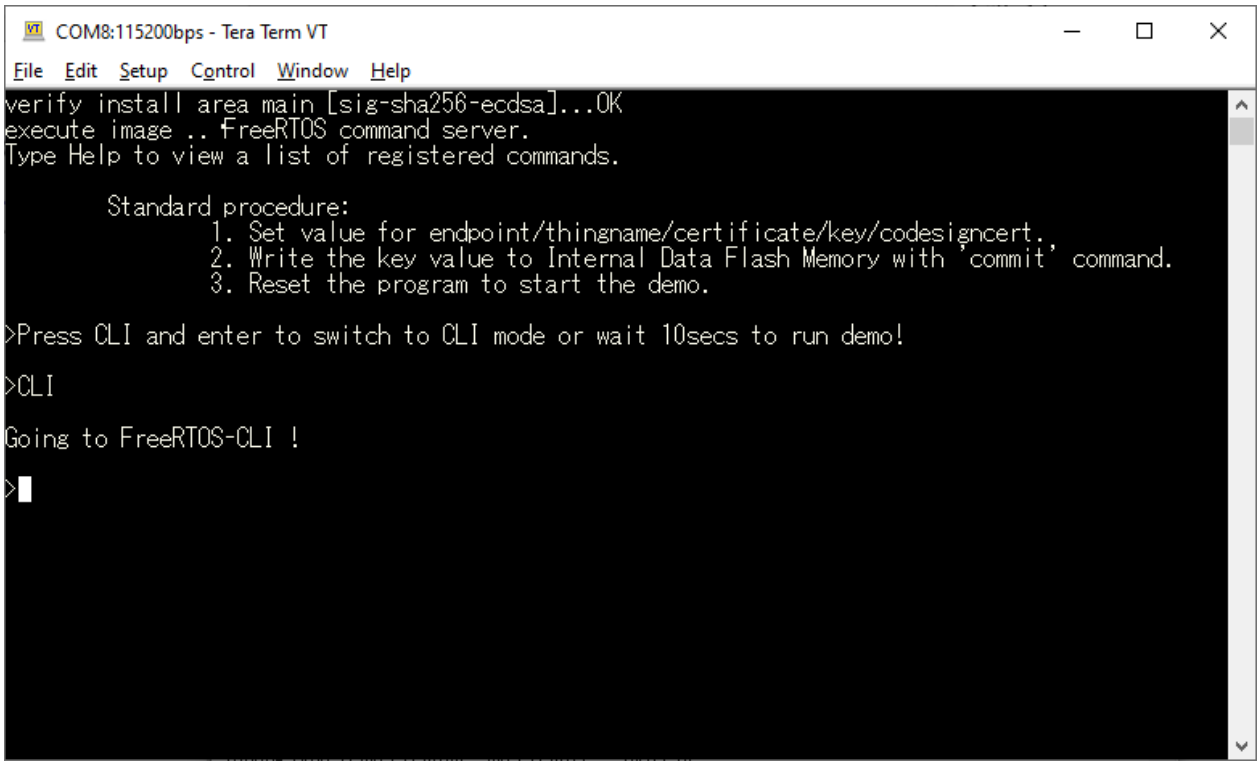
(4)　ターゲットボードのリセットとファームウェアの実行

CK-RX65N v2 の J16 を RUN 側に接続し、RESET SW を押下してください。初期ファームウェアが実行されます。



(5) CLI メニューの表示

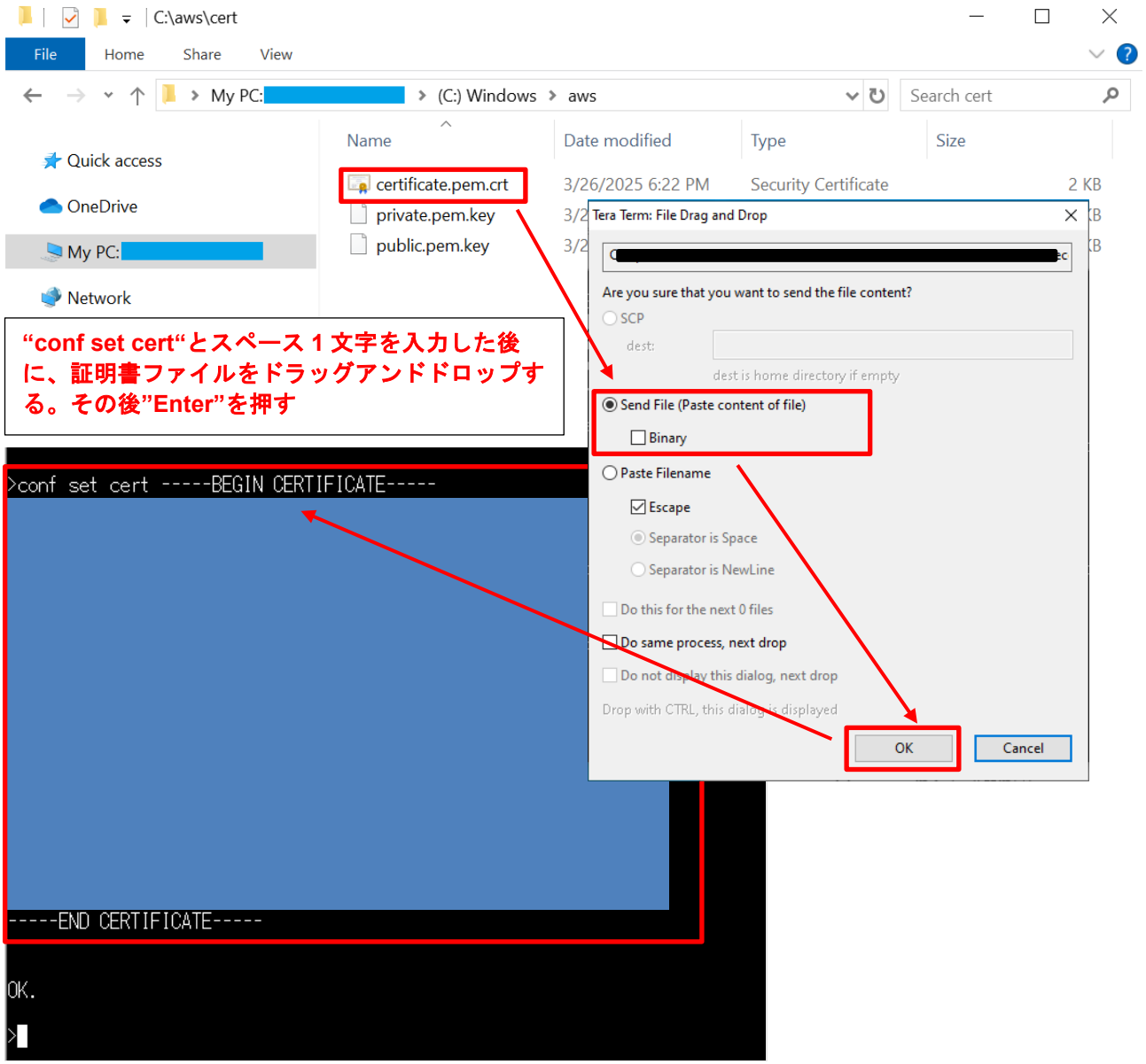
初期ファームウェアが実行されると Tera Term の画面にメニューが表示されます。10 秒以内に「CLI」と入力して[Enter]を押下してください。



(6)　デバイス証明書の登録

「3.3.4(2)」でダウンロードしたデバイス証明書をターゲットボードに登録します。
Tera Term にて「`conf set cert` 」と入力したのち、証明書ファイル(certificate.pem.crt)を Tera Term にドラッグアンドドロップ（ファイル送信）してください。最後に Tera Term 上で[Enter]を押してください。

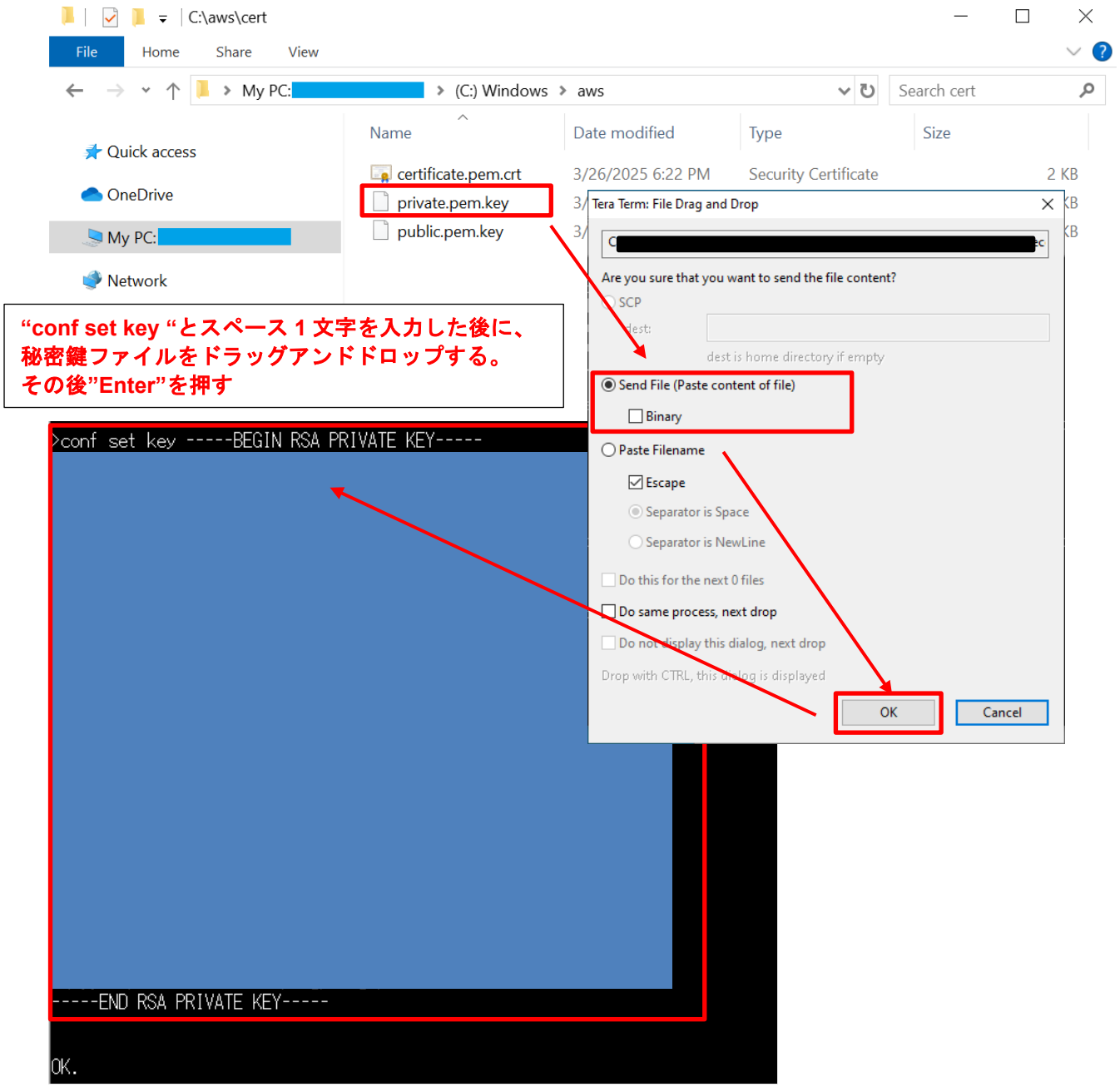
【注】証明書ファイルの改行コードはテキストエディタ等で LF に変更してからドラッグアンドドロップしてください。



(7)　秘密鍵の登録

「3.3.4(2)」でダウンロードした秘密鍵をターゲットボードに登録します。
Tera Term にて「conf set key 」と入力したのち、秘密鍵ファイル(private.pem.key)を Tera Term にドラッグアンドドロップ（ファイル送信）してください。最後に Tera Term 上で[Enter]を押してください。

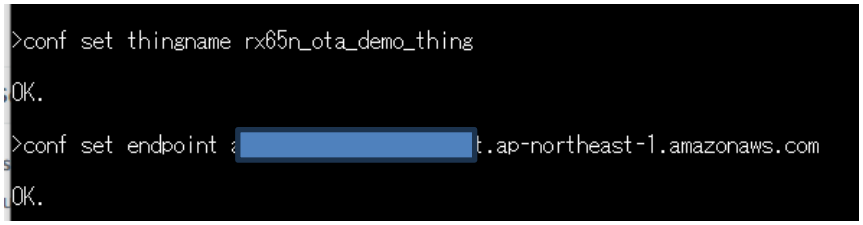
【注】鍵ファイルの改行コードは LF に変更してからドラッグアンドドロップしてください。



(8)　エンドポイントの登録

「3.3.4(1)」で設定したモノの名前および、「3.3.4(5)」で控えたエンドポイントをターゲットボードに登録します。
Tera Term で以下のコマンドを実行します。

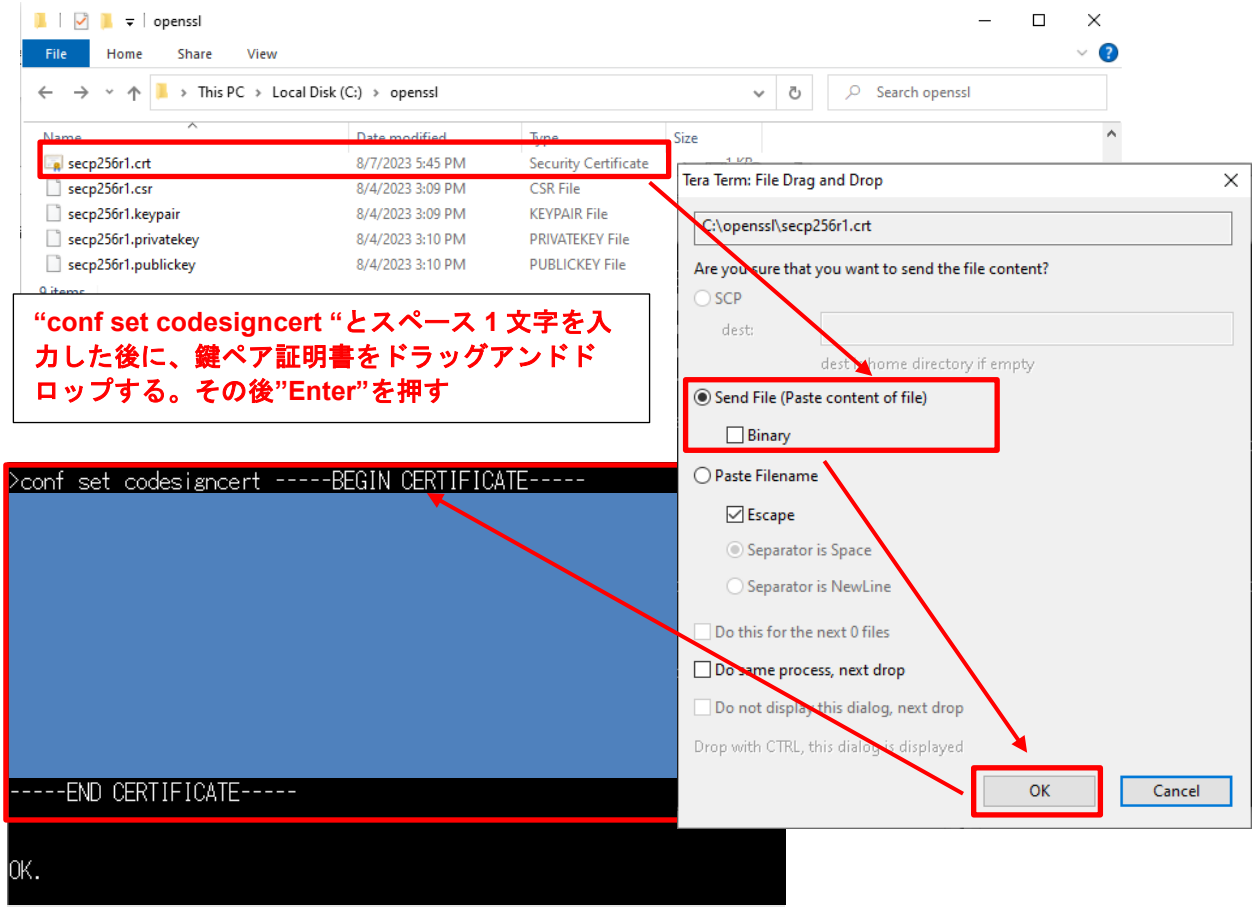
- `conf set thingname` <モノの名前>
- `conf set endpoint` <エンドポイント名>



(9)　コード署名検証用の証明書の登録

「4.1(6)」で生成した鍵ペア証明書(secp256r1.crt)をターゲットボードに登録します。
Tera Term にて「`conf set codesigncert` 」と入力したのち、鍵ペア証明書(secp256r1.crt)を Tera Term にドラッグアンドドロップ(ファイル送信)してください。最後に Tera Term 上で[Enter]を押してください。

【注】証明書ファイルの改行コードは LF に変更してから張り付けてください



(10) 設定値をデータフラッシュへの書き込む

AWS IoT の設定を Commit（データフラッシュに書き込み）します。
Tera Term で以下のコマンドを実行します。

```
> conf commit
```

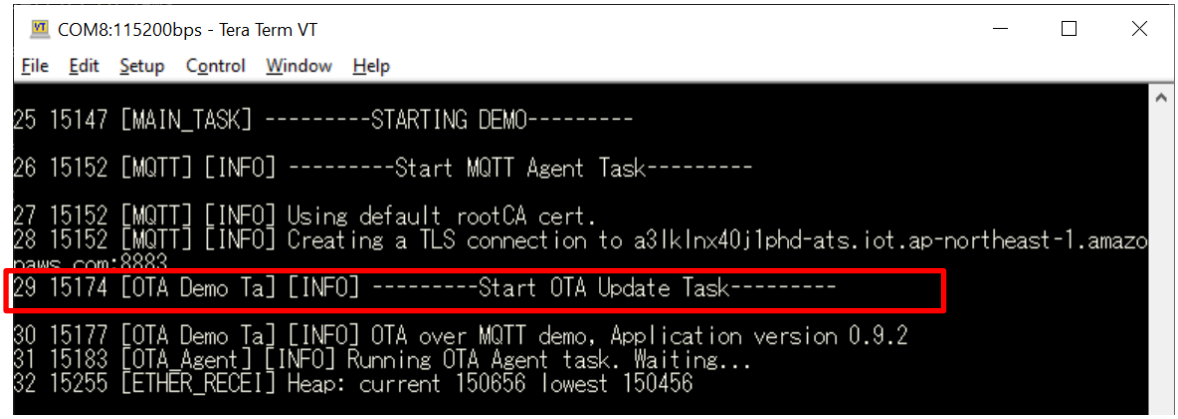
```
>conf commit

0 4472481 [CLI] Destroyed Certificate.

1 4472485 [CLI] Write certificate...

2 4472545 [CLI] Destroyed Private key.

3 4472685 [CLI] Write Private key...

Configuration saved to Data Flash and used 2879 bytes.
```

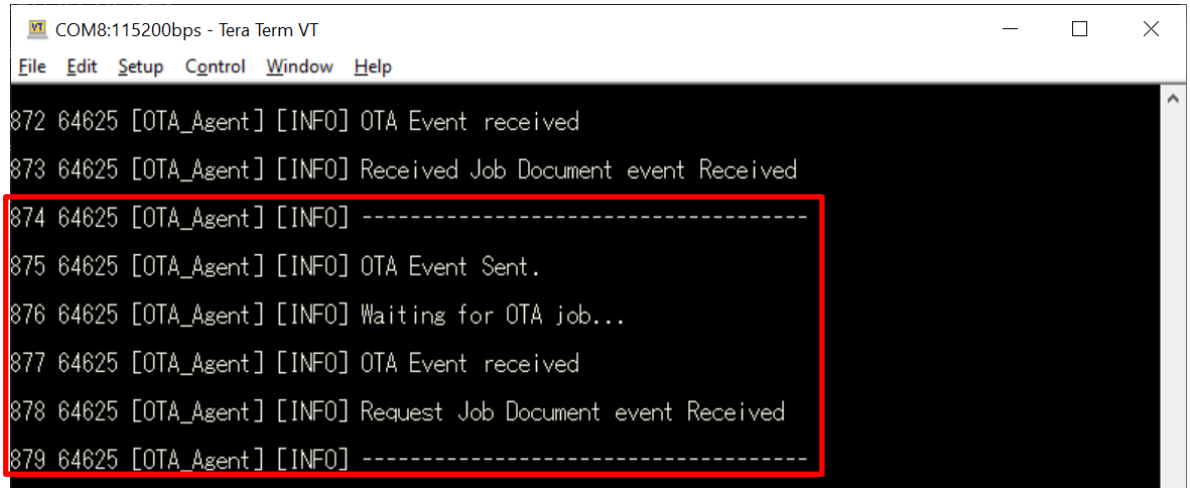
(11) Reset を実行

ソフトウェアリセットを実行し、ファームウェアを再起動します。Tera Term で以下のコマンドを実行します。

```
> reset
```

リセット実行後、Tera Term に通信ログが表示され、OTA タスクが実行され、OTA ジョブ待ちになっていることを確認してください。





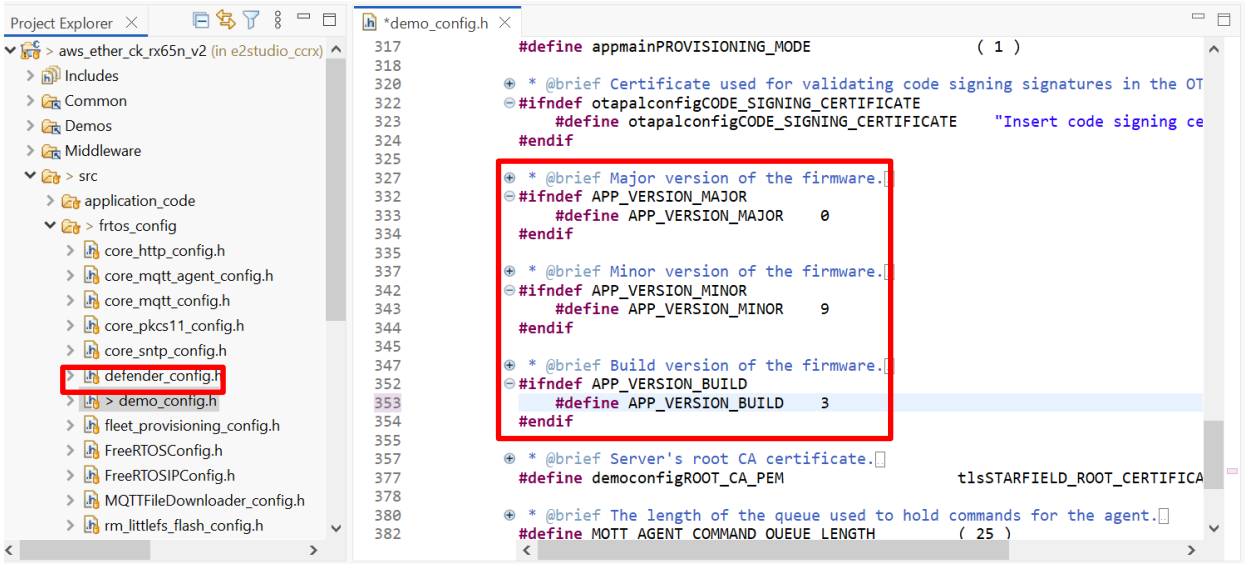
5. ファームウェアの更新

5.1 更新用ファームウェア構築

5.1.1 バージョンの変更

(1) ファームウェアのバージョンを変更

更新ファームウェアのバージョンを「v0.9.3」に変更します。
\aws_ether_ck_rx65n_v2\src\frtos_config\ demo_config.h の APP_VERSION_BUILD 定義を 3 にしてビルドを再実行してください。



(2) Renesas Image Generator を使用した更新ファームウェアの作成

5.1.1(1)で再ビルドしたファームウェア(aws_ether_ck_rx65n_v2.mot)を Renesas Image Generator フォルダに上書きし、コマンドプロンプトで以下コマンドを実行します。

コマンドの実行からファイル作成には数分かかる場合がありますのでしばらくお待ちください。

python image-gen.py -iup *aws_ether_ck_rx65n_v2.mot* -ip *RX65N_DualBank_ImageGenerator_PRM.csv* -o *user_093* -key *secp256r1.privatekey* -vt ecdsa -ff RTOS

上記コマンドで、user_093.rsu ファイルが生成されます。

5.2　ファームウェアの更新

AWS にて、ファームウェアの更新を行うために更新ファームウェアのアップロードと OTA 更新ジョブの作成を行います。

以下作業は AWS CLI を使用します。あらかじめ、「3.2」を参照し、コマンドプロンプトにて AWS CLI を使用し、AWS へのログインを行い、カレントディレクトリを「3.1.5」で作成した作業フォルダに移動しておいてください。

5.2.1　更新ファームウェアを AWS へアップロード

更新ファームウェアを AWS の S3 バケットへアップロードします。

(1)　ファームウェアを作業フォルダへコピー

「5.1.1(2)」で作成した更新ファームウェア（user_093.rsu）を「3.1.5」で作成した作業フォルダへコピーします。

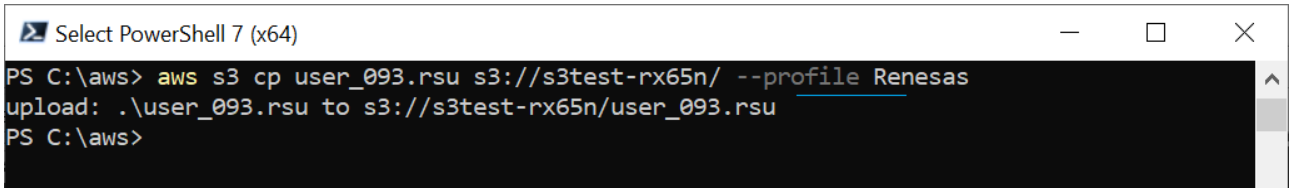
(2)　ファームウェアの S3 バケットへアップロード

作成したファームウェアを S3 バケットへアップロードします。
以下のコマンドを入力してください。

```
> aws s3 cp <FIRMWARE_NAME> s3://<BUCKET_NAME>/ --profile <PROFILE_NAME>
```

- *<FIRMWARE_NAME>*には更新ファームウェアのファイル名を入力します。
- *<BUCKET_NAME>*には「3.3.2(1)」で作成した S3 バケット名を入力します。

コマンドを実行すると、以下の様に表示されます。



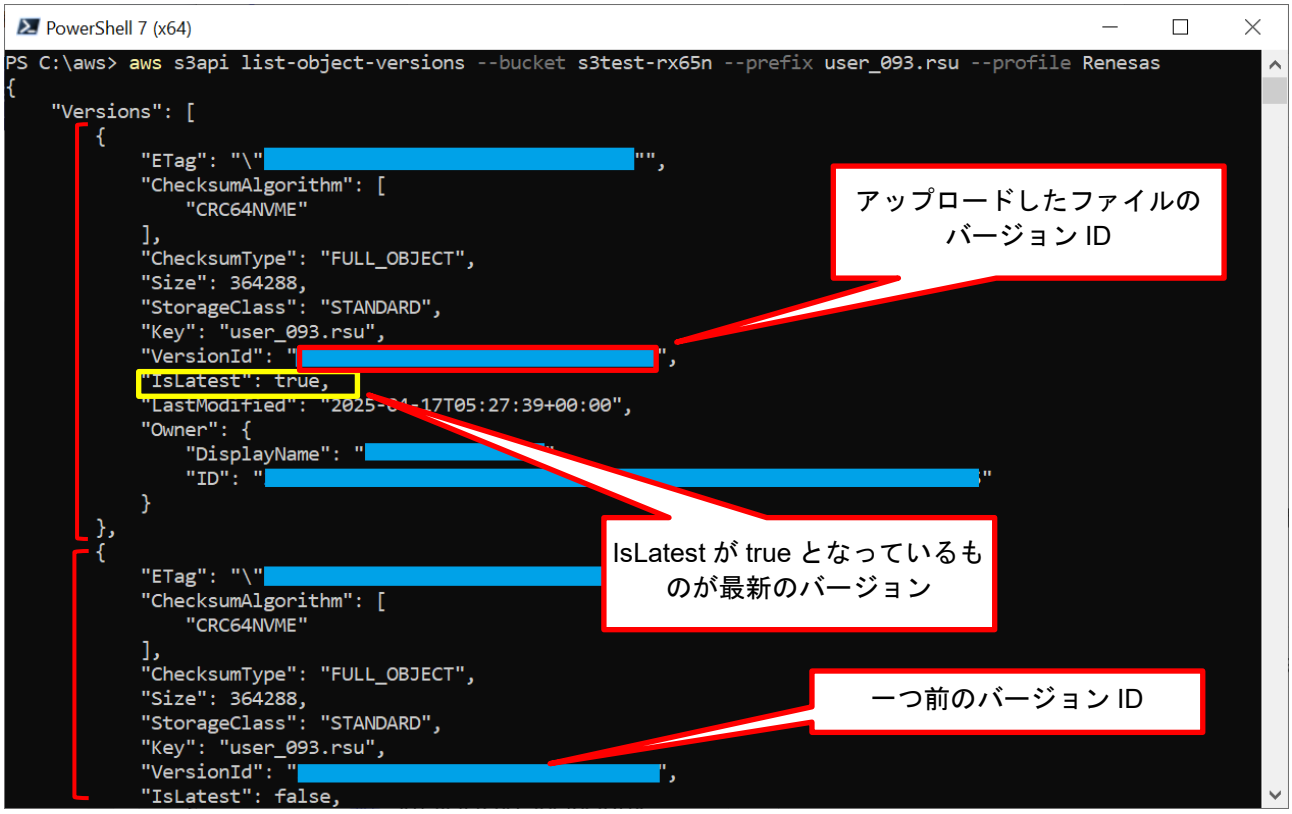
(3) バージョン ID の確認

アップロードした更新ファームウェアのバージョン ID を確認します。
以下のコマンドを入力してください。

```
> aws s3api list-object-versions --bucket <BUCKET_NAME> --prefix
<FIRMWARE_NAME> --profile <PROFILE_NAME>
```

- <BUCKET_NAME>には「(2)」でアップロードした S3 バケット名を入力します
- <FIRMWARE_NAME>には更新ファームウェアのファイル名を入力します。

コマンドを実行すると、以下の様に表示されます。



表示された更新ファームウェアファイルのバージョン ID（VersionId：上図の赤枠内）は、後の処理で使用するため控えておいて下さい。

また、バージョン ID は下の「IsLatest」フィールド（黄枠内）が「true」と表示されている最新バージョンの VersionId を読み取ってください（同じファイル名で複数回ファイルをアップロードした場合、上図の Versions のフィールドが複数回分表示されます）。

5.2.2　コード署名証明書とプロファイルの作成

OTA におけるアップロードした更新ファームウェアの信頼性を保証するためのコード署名証明書を作成します。

過去にコード署名証明書のアップロードとプロファイルの作成を実施していた場合は、本項の手順は省略できます。OTA ジョブ作成時の json ファイルに、本項で作成したプロファイル名を記述してください。

(1)　証明書と鍵ファイルの準備

「4.1(9)」で作成した以下の 3 つのファイルを「3.1.5」で作成した AWS CLI の作業フォルダにコピーします。

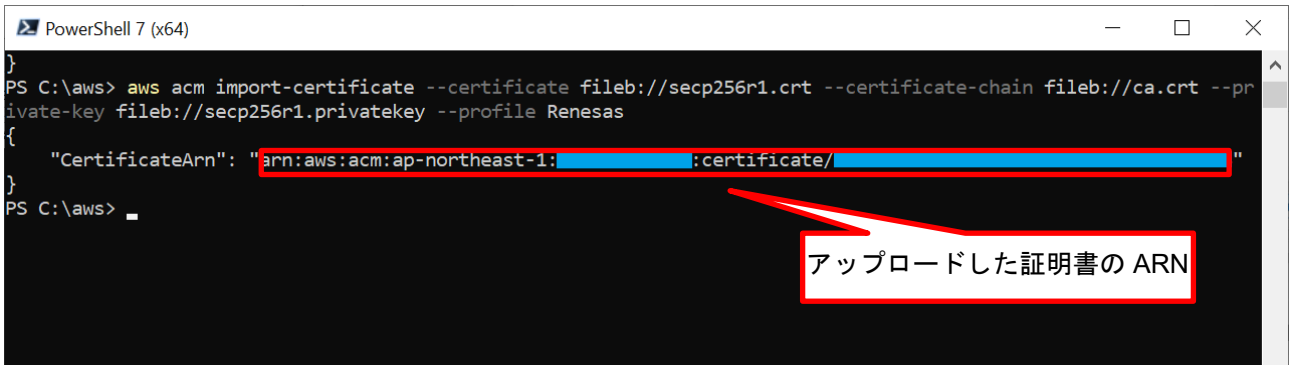
- ECDSA 証明書：secp256r1.crt
- ECDSA 秘密鍵：secp256r1.privatekey
- ECDSA 証明書チェーン：ca.crt

(2)　新しいコード署名証明書のインポート

「(1)」で準備したコード署名証明書用の各ファイルを AWS へインポートします。
以下のコマンドを入力してください。

```
> aws acm import-certificate --certificate fileb://secp256r1.crt --
certificate-chain fileb://ca.crt --private-key fileb://secp256r1.privatekey --
profile <PROFILE_NAME>
```

コマンドを実行すると、以下の様に表示されます。



表示されたコード署名証明書の ARN（CertificateArn：上図の赤枠内）は、後の処理で使用するため控えておいて下さい。

(3) コード署名証明書のプロファイルの作成

(a) プロファイルを設定した json ファイルの作成

コード署名証明書プロファイルの設定ファイルを作成します。
テキストエディタで「3.1.5」で作成した作業フォルダに「profile.json」というファイルを作成し、以下のコードを入力します。

- profile.json

```
{
    "profileName": "rx65n_ota_demo_profile",
    "signingMaterial": {
        "certificateArn": "arn:aws:acm:ap-northeast-1:x
xxxxxxxxxxx:certificate/yyyyyyyy-yyy-yyyy-yyyy-yyyyyyyyyyyy"
    },
    "platformId": "AmazonFreeRTOS-Default",
    "signingParameters": {
        "certname": "dummy"
    }
}
```

プロファイル名

証明書 ARN

上記 json ファイルの赤線部分のフィールドは以下の情報を入力してください。

項目	内容	設定内容
profileName	プロファイル名	任意のプロファイル名を入力します。 (例: rx65n_ota_demo_profile)
certificateArn	証明書 ARN	「(2)」でインポートしたコード署名証明書の ARN を入力します

ここで設定したプロファイル名（profileName）は、後の処理で使用するため控えておいて下さい

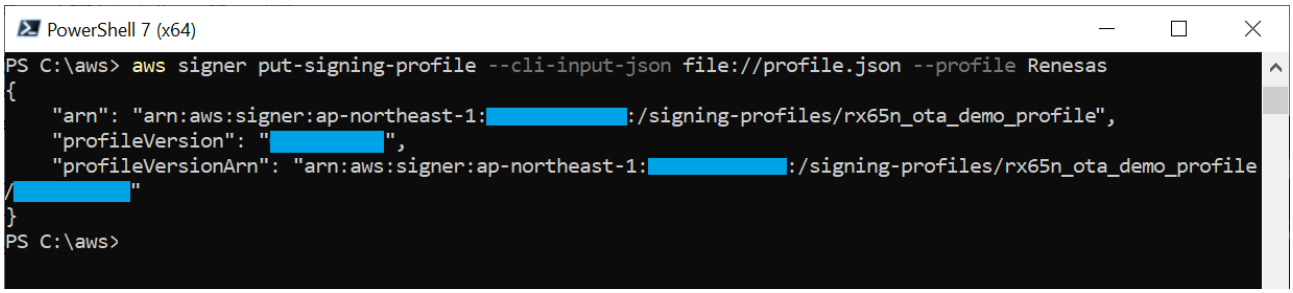
(b) コード署名証明書プロファイルの作成

「(a)」で作成した json ファイルを使用してプロファイルを作成します。

以下のコマンドを入力してください。

```
> aws signer put-signing-profile --cli-input-json file://profile.json --
profile <PROFILE_NAME>
```

コマンドを実行すると、以下の様に表示されます。



5.2.3 OTA ジョブの実行

OTA ジョブを作成し、OTA を実行します。

(1) OTA ジョブの詳細を設定した json ファイルの作成

OTA ジョブの設定ファイルを作成します。
テキストエディタで「3.1.5」で作成した作業フォルダに「create-ota-update.json」というファイルを作成し、以下のコードを入力します。

- create-ota-update.json

```
{
    "otaUpdateId": "rx65n_ota_demo_job",
    "targets": [
        "arn:aws:iot:ap-northeast-1:xxxxxxxxxxxx:thing/rx65n_ota_demo_thing"
    ],
    "protocols": [
        "MQTT"
    ],
    "targetSelection": "SNAPSHOT",
    "awsJobExecutionsRolloutConfig": {
        "maximumPerMinute": 1000
    },
    "files": [
        {
            "fileName": "/device/updates",
            "fileLocation": {
                "s3Location": {
                    "bucket": "s3test-rx65n",
                    "key": "user_093.rsu",
                    "version": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
                }
            },
            "codeSigning": {
                "startSigningJobParameter": {
                    "destination": {
                        "s3Destination": {
                            "bucket": "s3test-rx65n",
                            "prefix": "SignedImages/"
                        }
                    },
                    "signingProfileName": "rx65n_ota_demo_profile"
                }
            }
        }
    ],
    "roleArn": "arn:aws:iam::xxxxxxxxxxxx:role/ota_role_rx65n"
}
```

上記 json ファイルの赤線部分のフィールドは次の情報を入力してください。

項目	内容	設定内容
otaUpdateId	OTA アップデート ID	実行する OTA ジョブの名前です。 任意のジョブ名を入力します。 （例：rx65n_ota_demo_job） 【注】OTA ジョブ名は重複できないため、過去に使用したジョブ名とは別の名前を設定してください。
targets	ターゲット ARN	OTA 実行対象のモノ（デバイス）の ARN を入力します。 「3.3.4(1)」で設定されたモノの ARN を入力してください。
bucket （files→fileLocation→s3Location）	S3 バケット	更新ファームウェアをアップロードしたバケット名を入力します。 「3.3.2(1)」で設定したバケット名を入力してください。
key （files→fileLocation→s3Location）	S3 キー	更新ファームウェアのファイル名を指定します。 「5.2.1(2)」でアップロードしたファイル名を入力してください。
version （files→fileLocation→s3Location）	S3 バケットバージョン	更新ファームウェアのバージョン ID を入力します。 「5.2.1(3)」で確認したアップロードファイルのバージョン ID を入力してください。
bucket （files→codeSigning→ startSigningJobParameter→ destination→s3Destination）	コード署名に使用するバケット	コード署名を行う S3 バケット名を入力します。 「3.3.2(1)」で設定したバケット名を入力してください。
signingProfileName （files→codeSigning→ startSigningJobParameter）	コード署名のプロファイル名	コード署名に使用するプロファイル名を入力します。 「5.2.2(3)(a)」で設定したプロファイル名を入力して下さい。
roleArn	ロール ARN	OTA に使用する IAM ロールの ARN を入力します。 「3.3.3(2)」で作成した IAM ロールの ARN を入力してください。

ここで設定した OTA アップデート ID（otaUpdateId）は、後の処理で使用するため控えておいて下さい

(2) OTA ジョブの実行

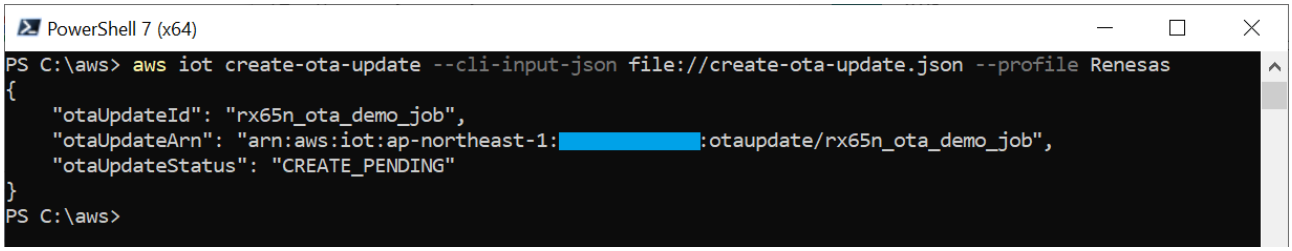
あらかじめ CK-RX65N v2 ボードで初期ファームウェアを実行し、OTA 実行待ちになっていることを確認してください（「4.2.4(11)」を参照）。

「(1)」で作成した json ファイルを使用して OTA ジョブを実行します。

以下のコマンドを入力してください。

```
> aws iot create-ota-update --cli-input-json file://create-ota-update.json --profile <PROFILE_NAME>
```

コマンドを実行すると、以下の様に表示されます。
正常にジョブが実行されるとターゲットボードに更新ファームウェアのダウンロードが開始されます。



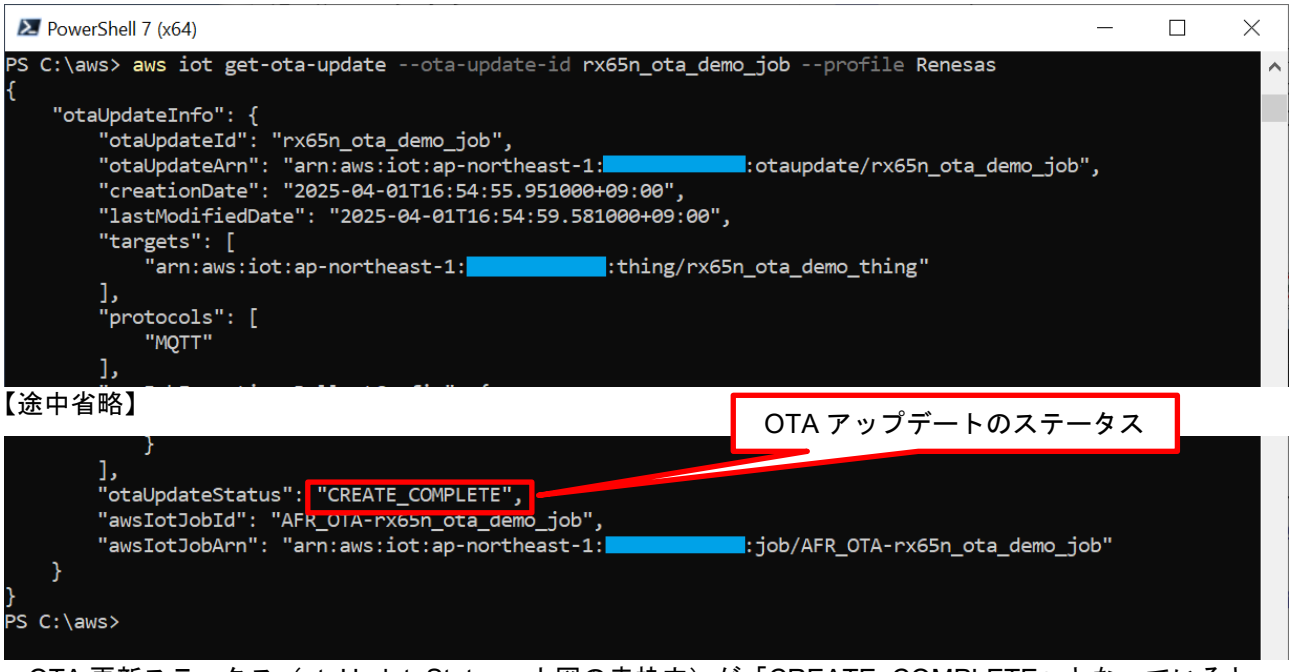
(3) OTA ジョブ実行状況の確認

OTA ジョブを実行した後の状況を確認します。
以下のコマンドを入力してください。

```
> aws iot get-ota-update --ota-update-id <JOB_NAME> --profile <PROFILE_NAME>
```

- <JOB_NAME>には「(1)」で設定した OTA ジョブ名（OTA アップデート ID）を入力します。

コマンドを実行すると、以下の様に表示されます。



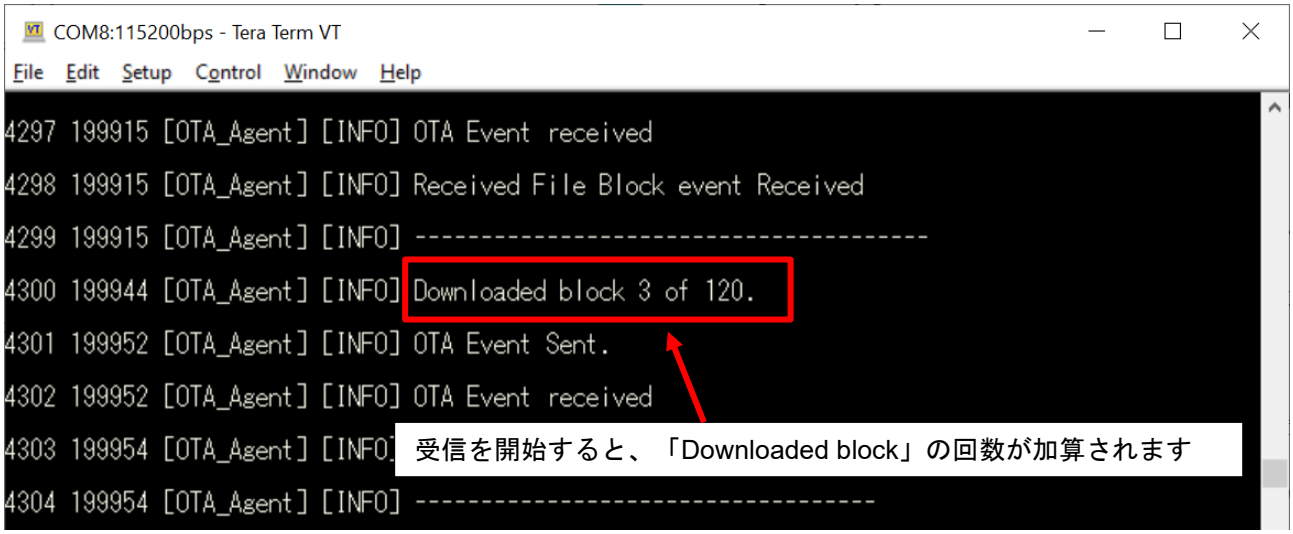
【途中省略】

OTA 更新ステータス（otaUpdateStatus：上図の赤枠内）が「CREATE_COMPLETE」となっていると OTA ジョブの実行は成功です。

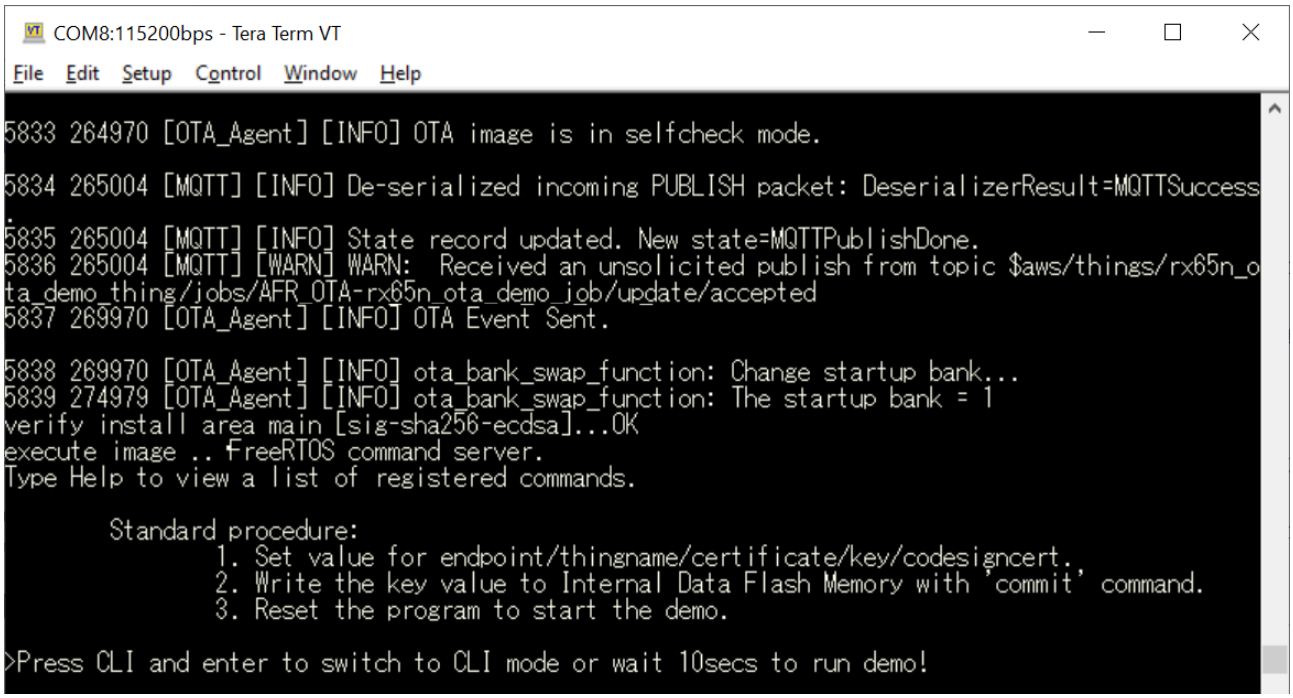
設定等に問題がある場合など、OTA ジョブがエラーになった場合は「CREATE_FAILED」等が表示されます。

(4)　ファームウェアの受信が完了するまで待つ

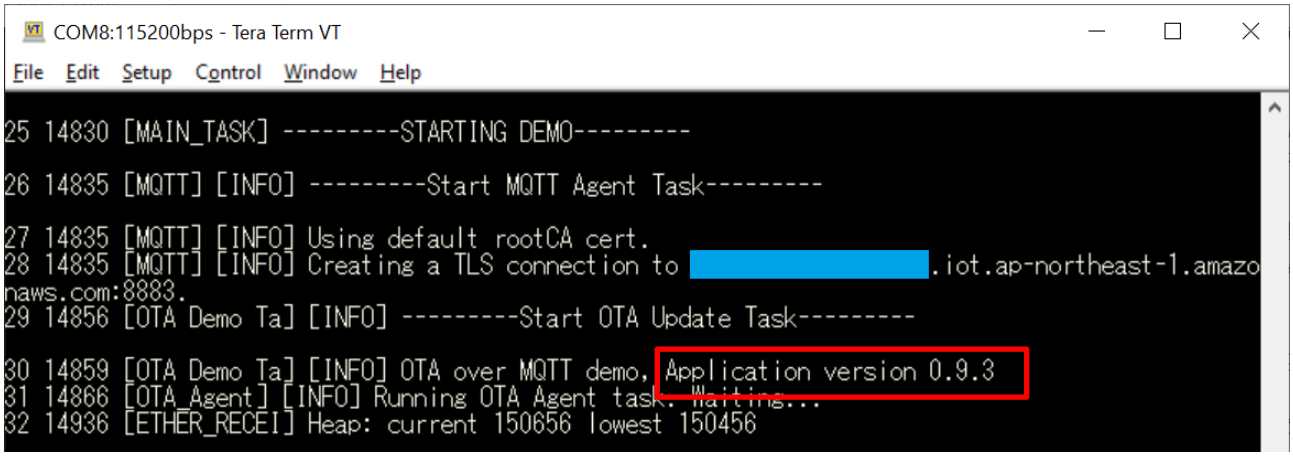
OTA ジョブが開始されると、受信およびファームウェアの書き込みを行います。



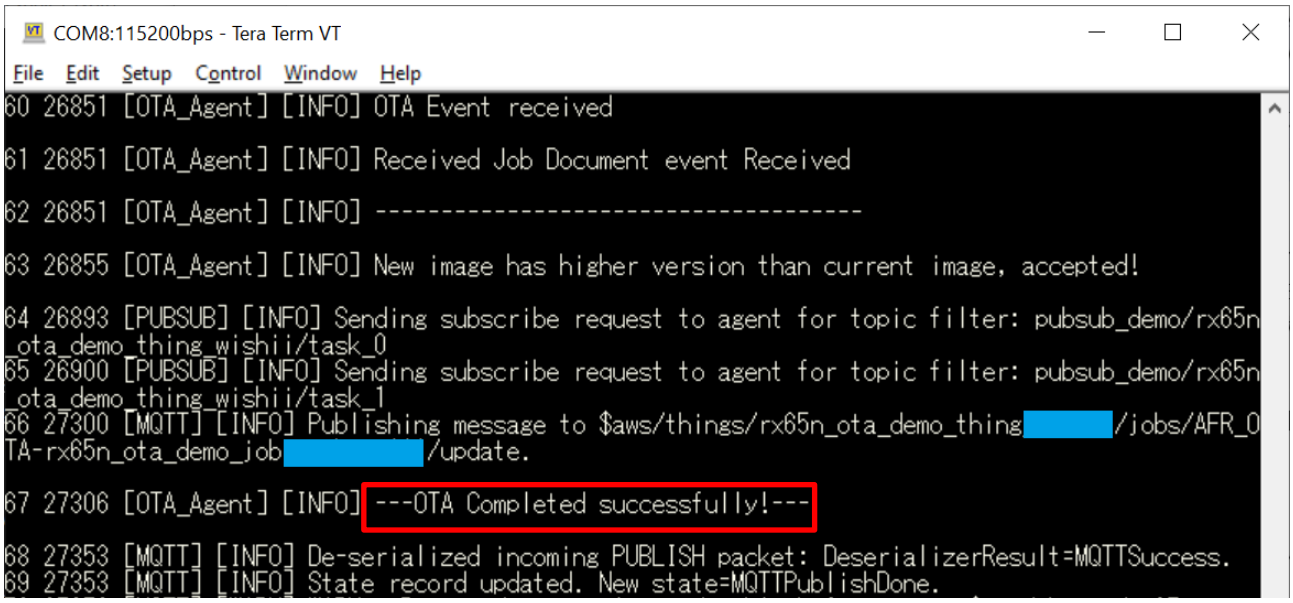
更新が完了し署名検証に成功すると、CPU リセット後に更新ファームウェアが実行され、最初のメニューが表示されます。



(5)　リセット後にファームウェアのバージョンが「version 0.9.3」になっていることを確認します。



また、OTA ジョブが正常に完了すると以下の様に「OTA Completed successfully!」と表示されます。



6. 付録

6.1 同一 LAN 環境内において複数の機器を同時に動作させる場合の注意事項

サンプルコードに含まれる MAC アドレスはルネサスエレクトロニクス株式会社のベンダ ID から割り当てられたアドレスを使用しています。

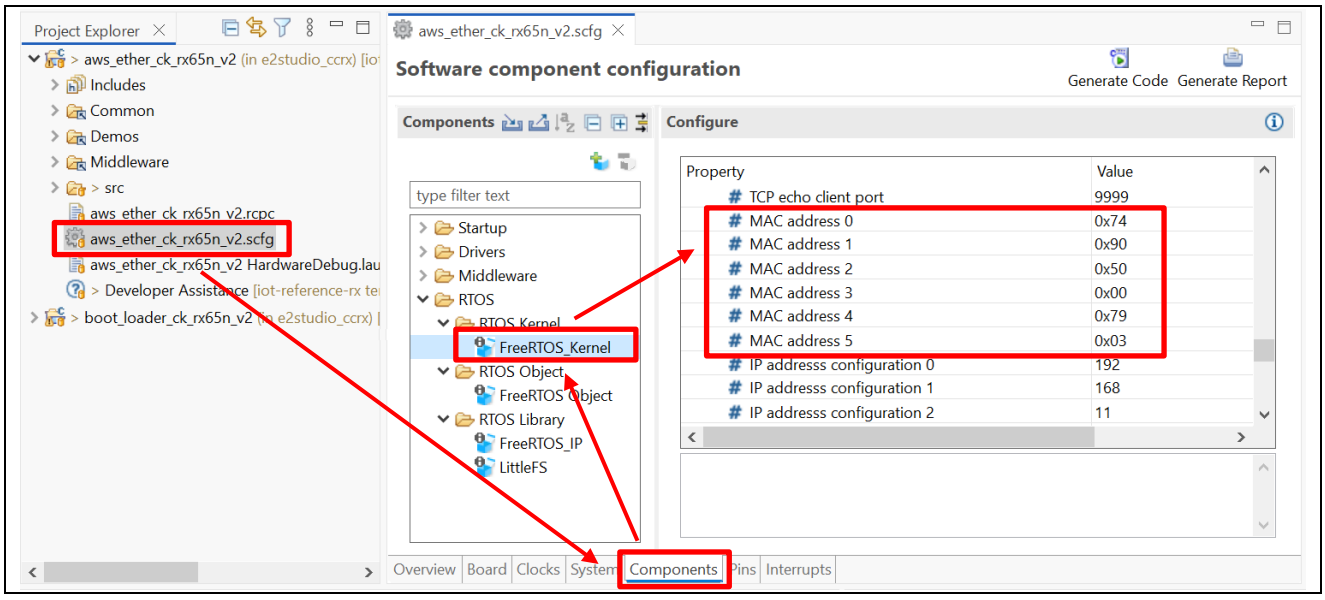
同一 LAN 環境内においてサンプルプログラムを複数の機器で同時に動作させる場合は、MAC アドレスが重複しないように変更してください。

複数の機器で MAC アドレスが重複するとサンプルプログラムが正しく動作しない可能性があります。

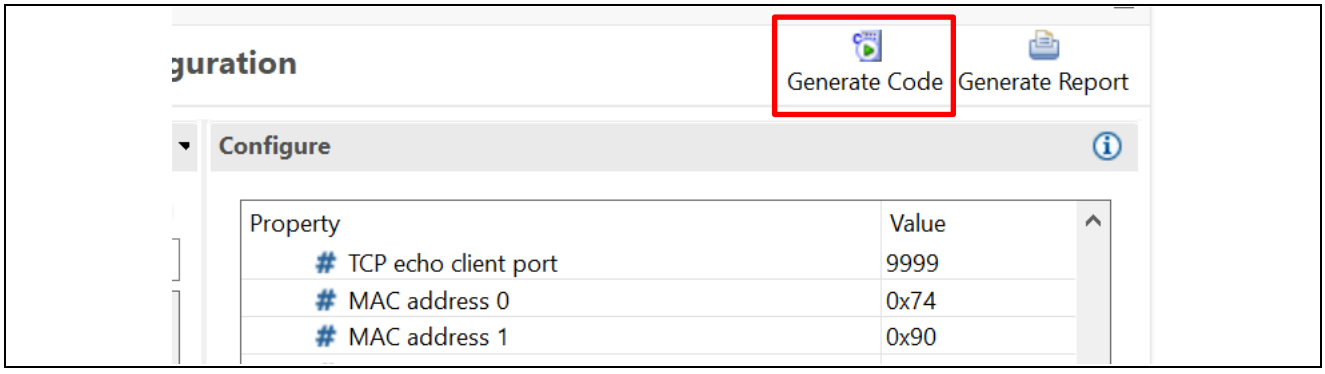
以下に MAC アドレスの変更手順を示します。

スマート・コンフィグレータ aws_ether_ck_rx65n_v2.scfg を開き、Components（コンポーネント）タブを選択します。
ツリーより[RTOS]→[RTOS Kernel]→[FreeRTOS_Kernel]を選択し、Property から「MAC address 0～5」の Value を任意の 16 進数の値に変更してください。
値は 0xXX（XX は任意の 16 進数の値）で入力します。

なお、お客様が製品化する際には必ず IEEE に申請した MAC アドレスを使用してください。



上記の設定変更を行った場合は、設定後画面右上の[Generate Code（コードの生成）]ボタンをクリックして、スマート・コンフィグレータの変更内容をコードに反映してください。



7. トラブルシューティング

本サンプルを実行する際に想定されうるトラブルおよびその解決策を下表に示します。

表 7-1 トラブルシューティング

No	トラブル内容	原因	解決策	参照
1	初期ファームウェアの作成コマンドが失敗する	Python のインストールフォルダが環境変数のパスに正しく設定されていません。	Python を再インストールしてください。また、2.2(3)の手順で、「Add python.exe to PATH」にチェックが入っていることを確認してください。	2.2
2		暗号化ライブラリがインストールされていない	暗号化ライブラリをインストールしてください。	2.2(5)
3	初期ファームウェアが書き込めない	CK-RX65N v2 がデバッグ設定になっていない	CK-RX65N v2 の J16 の設定が 1-2 ショート（DEBUG）になっていることを確認してください。	2.6
4	初期ファームウェアが起動しない	CK-RX65N v2 が RUN 設定になっていない	CK-RX65N v2 の J16 の設定が 2-3 ショート（RUN）になっていることを確認してください。	4.2.4(4)
5	セルラー通信が開始できない	RYZ014A PMOD が正しく接続されていない	RYZ014A PMOD の接続を見直してください。	2.6
6		SIM カードが挿入されていない	SIM カードを挿入してください。	2.6
7		SIM カードの設定が正しくされていない	r_cellular のコンフィグ設定を見直してください。	4.2.2(4)
8		CK-RX65N v1 付属の SIM カードを使用しており、かつ SIM カードのアクティベーションができていない	SIM カードのアクティベーションを行ってください。	4.2.2(4)
9	セルラー通信中にエラーが発生する	通信環境が悪い	RYZ014A PMOD にアンテナおよび電源の接続を行ってください。また、アンテナを窓際など通信環境の良い場所においてください。	2.6
10	AWS への接続でエラーが発生する	AWS IoT 情報が設定されていない、または間違えている	再度、AWS IoT 情報の設定を行ってください。	4.2.4
11	ブートローダ起動後にファームウェアが起動しない	・ブートローダに公開鍵が正しく設定がされていない ・ファームウェアのバージョン ID の登録が間違っている	・ブートローダの公開鍵設定を見直してください。 ・アップロードしたファームウェアのバージョンを確認してください。	4.2.2(1) 5.2.1(3)
12	OTA アップデート後にファームウェアが起動しない	プロジェクトが正しく設定されていない	ファームウェアおよびブートローダの設定を見直してください。	4.2.2 4.2.3(1)
13	OTA ジョブを再実行した場合にエラーになる	OTA ジョブ名が重複している	OTA ジョブ名は再利用できません。別の名前を設定してください。	5.2.3(1)

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2025.6.20	-	初版発行

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。