RX ファミリ

AWS IoT Fleet Provisioning の実現方法 (202406-LTS 版)

要旨

クラウドサービス Amazon Web Service™（以下、AWS）において、「AWS IoT」サービスに接続するためには IoT デバイスのプロビジョニングが必要となります。ここで、プロビジョニングとは、「モノ、秘密鍵、デバイス証明書」などの認証情報の生成および、運用 / 管理する仕組みを指します。プロビジョニングには、製品製造時の認証情報の書き込み (初期インストール)、鍵データの管理 (保護) および 更新方法等の検討が必要で、これらのデータは RX ファミリ内蔵フラッシュメモリに保存されます。
IoT デバイスのプロビジョニング方法は後から変更することが非常に困難なため、製品開発の段階から検討を開始し、量産フェーズに至るまでに検証を完了させる必要があります。

本書は、AWS で準備されている様々なプロビジョニング方式のうち、製造工程 および IoT デバイスの使用開始時におけるプロビジョニング処理を自動化する「フリートプロビジョニング方式」について解説します。

フリートプロビジョニング方式を導入することで、煩わしいプロビジョニング作業に手間と時間をかけることなく、より安全で快適なプロビジョニング動作を実現することが可能です。

本アプリケーションノートで学べること

- ✓ AWS が提供するプロビジョニング手法の概要
- ✓ デモを用いたフリートプロビジョニングの実現、動作確認方法(デモの動作手順は、”4 フリートプロビジョニング デモの実行方法” から解説します)

本書の内容でプロビジョニングの実現が可能ですが、プロビジョニングで保存される秘密鍵やデバイス証明書などの大切なデータは RX ファミリ内蔵フラッシュメモリに『平文』の状態で格納されている状態です。RX ファミリに書き込まれるユーザープログラムにセキュリティホールが内在しており任意メモリの読み出しが出来た場合、フラッシュメモリ内のプロビジョニングされたデータが漏えいし、アタッカによってユーザーの AWS のアカウントに不正ログインされる可能性が残ってしまいます。

RX ファミリに搭載される Trusted Secure IP（TSIP）を利用すると秘密鍵、デバイス証明書を暗号化して保持できるため、プロビジョニングデータの漏えいの危険性をさらに軽減することが可能です。TSIP の詳細については下記ページをご参照ください。

https://www.renesas.com/software-tool/trusted-secure-ip-driver

更なるセキュリティの向上を目指す場合は、TSIP の活用を是非ご検討ください。

また、プロビジョニングデータの漏えいリスクはソフトウェアの品質を上げることで軽減することができますがゼロにすることはできません。特に IoT デバイスのようなアタッカからの脅威に晒されやすいデバイスにおいてはソフトウェアの不具合があった場合、ファームウェアアップデート機能を使って速やかに修正することが推奨されます。

【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、DA16600 モジュール（Wi-Fi）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] iot-reference-rx/Getting_Started_Guide.md at main · renesas/iot-reference-rx (github.com)

【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
[製品ページ] https://www.renesas.com/ryz014a

動作環境

本アプリケーションノートで説明する動作は以下の環境で確認しました。

統合開発環境	e2 studio 2025-12
ボード	CK-RX65N v2
ツールチェーン	CC-RX Compiler v3.07.00 GCC for Renesas RX v14.2.0.202505
エミュレータ	CK-RX65N 搭載の E2OB (E2 Lite On Board)

本アプリケーションノートを他のマイコンへ適用する場合、
そのマイコンの仕様にあわせて製品固有の設定を見直し、十分評価を行った上で適用してください。

関連アプリケーションノート

本アプリケーションノートに関連するドキュメントの情報を以下に示します。

必要に応じて参照してください。

- Amazon Web Services を利用した FreeRTOS OTA の実現方法(202406-LTS 版) (R01AN7662)

RX クラウドソリューション開発に必要なボード、関連プログラム、開発環境に関する情報は、

以下のリンクにまとめられています。

https://www.renesas.com/rx-cloud

また、AWS が公開している下記情報も参考になります。

- AWS IoT におけるデバイスへの認証情報のプロビジョニング
  動画: https://youtu.be/gcJwNEQ2eLY
  資料: https://pages.awscloud.com/rs/112-TZM-766/images/EV_iot-deepdive-aws2_Sep-2020.pdf
- フリートプロビジョニングテンプレートのドキュメント
  https://docs.aws.amazon.com/iot/latest/developerguide/provision-template.html
- AWS IoT ポリシーのドキュメント
  https://docs.aws.amazon.com/iot/latest/developerguide/iot-policies.html
- AWS IoT API リファレンスドキュメント：CSR から証明書を作成する方法について
  https://docs.aws.amazon.com/iot/latest/apireference/API_CreateCertificateFromCsr.html
- フリートプロビジョニングを使用したデバイス証明書がないデバイスのプロビジョニング
  https://docs.aws.amazon.com/ja_jp/iot/latest/developerguide/provision-wo-cert.html
- フリートプロビジョニングを用いて、IoT デバイスと AWS IoT Core の初期セットアップを自動化する方法 https://aws.amazon.com/jp/blogs/news/how-to-automate-onboarding-of-iot-devices-to-aws-iot-core-at-scale-with-fleet-provisioning/

目次

- 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/)

1. 用語

本資料中の用語を説明します。

表 1-1 用語集

用語	意味
AWS	Amazon Web Services, Inc.が提供する クラウドコンピューティングサービス
FreeRTOS	組み込みシステム用のオープンソースの リアルタイムオペレーティングシステム
プロビジョニング	デバイスプロビジョニング。AWS IoT Core と 通信するためにデバイス認証を行うこと。
フリートプロビジョニング	IoT デバイスの初回起動時に、 自動でプロビジョニングを行う機能。

2. デバイスプロビジョニング

IoT デバイスのプロビジョニングとは、AWS IoT および その他のクラウドベースのアプリケーションをセキュアに接続するために、デバイスのユニークな ID (X.509 証明書や秘密鍵等) を生成し、これらの ID を AWS IoT エンドポイントに登録して、必要なアクセス許可 (IoT ポリシー等) を関連付けるプロセスを指します。（図 2-2 を参照）

AWS IoT におけるデバイスプロビジョニングは、Just-In-Time-Registration (JITR) や Just-In-Time-Provisioning (JITP) 等の AWS IoT Core 機能を使用して、デバイスアイデンティティを AWS クラウドに登録し、必要な権限を関連付けるプロセスの自動化や、複数デバイスのプロビジョニングを行うことができます。しかし、一意の ID を安全に生成してデバイスに書き込むプロセスは、ユーザーの責任で行う必要があり、多数のデバイスを製造する OEM ベンダにとって、このプロセスは手作業 かつ 時間のかかる作業となります。

この課題への対処方法の１つとして、本書で扱うフリートプロビジョニングがあります。

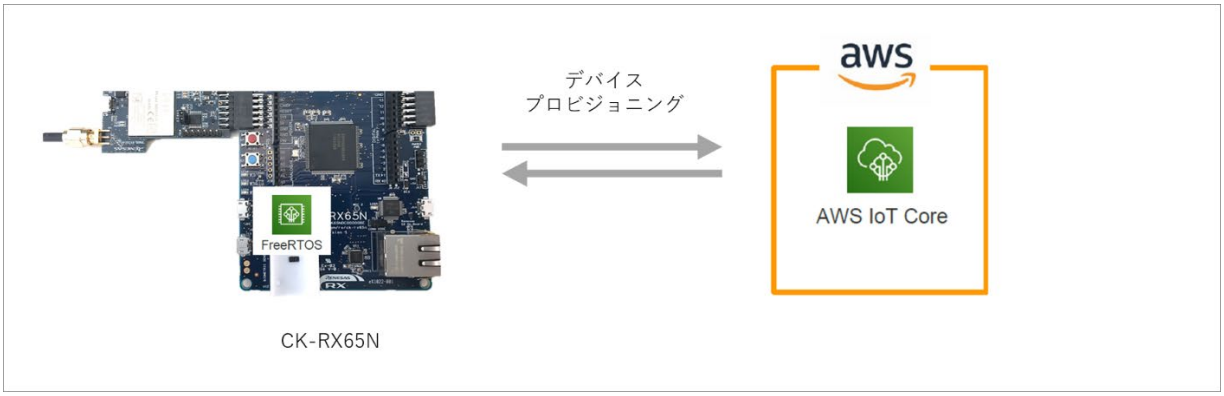

図 2-1 デバイスプロビジョニング

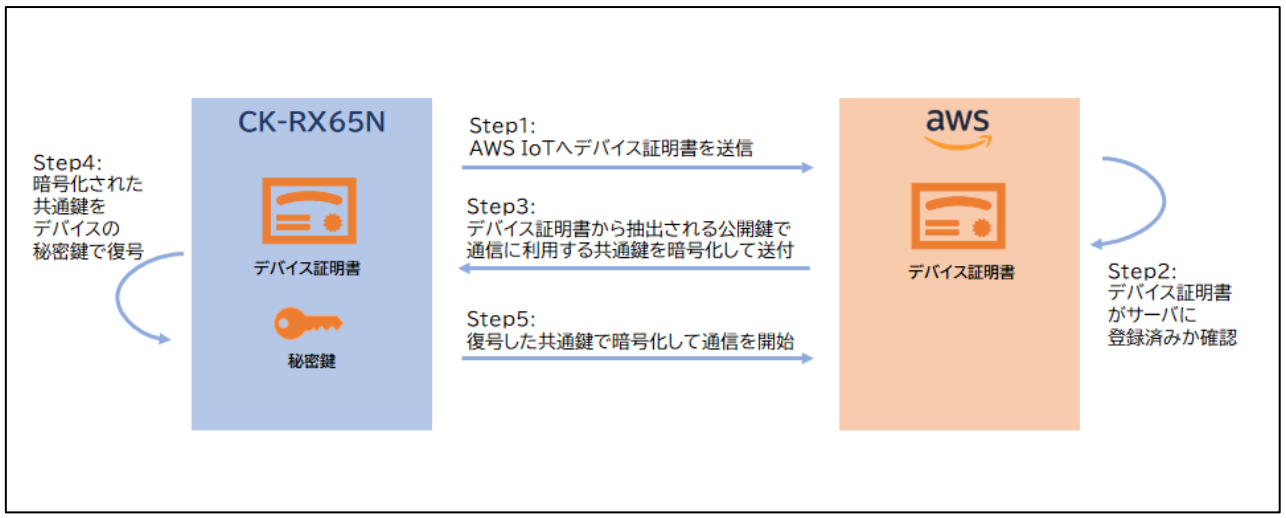

図 2-2 IoT デバイスのプロビジョニングの概要

2.1 AWS IoT におけるプロビジョニング方式

AWS IoT では、下記のプロビジョニング方式を選択することが可能です。

AWS では、ユーザーが用途に合わせて最適なデバイスプロビジョニング方式を選択できるようにするため、市場の要求および様々なユースケースを想定し、複数のプロビジョニング方式を用意しています。ユーザーのプロビジョニング方式選択に役立つ、プロビジョニング方式毎の仕組みおよびメリット / デメリットに関する情報は、下記資料にまとめられています。プロビジョニング方式検討の際には、下記資料を参照することをお勧めします。

https://pages.awscloud.com/rs/112-TZM-766/images/EV_iot-deepdive-aws2_Sep-2020.pdf#page=115

[AWS IoT におけるプロビジョニング方式]

1. AWS IoT による秘密鍵・証明書発行＆事前登録 (デバイスキッティング時登録)
2. AWS IoT による証明書発行＆事前登録 (デバイスキッティング時登録)
3. フリートプロビジョニング (Fleet Provisioning) 登録（本書で説明）
4. 独自 CA による証明書発行＆ AWS IoT への事前登録
5. 独自 CA による証明書発行＆ JITR による登録
6. 独自 CA による証明書発行＆ JITP による登録
7. CA 登録無しの証明書登録（マルチアカウント登録）

量産検討を始める前に FreeRTOS の動作を確認したい場合は、AWS 上で秘密鍵と証明書を発行してソースコードに埋め込むことができる形式に変換し、変換したコードを FreeRTOS のソースコード群に埋め込む形の「AWS IoT による秘密鍵・証明書発行＆事前登録」が最も簡単です。しかしながらこの方式は製造時にデバイス個別の証明書を埋め込むことが困難です。このため、本書では CA（Certification Authority（認証局））の運用が不要で最も量産時の工数負荷を減らせる Fleet Provisioning にフォーカスを当てています。

【注】一部の RX ファミリには Trusted Secure IP (TSIP)が搭載されています。
TSIP を用いると、RSA や楕円曲線暗号の鍵ペアをチップ内の乱数生成器で生成し、公開鍵を抽出しそれを任意の CA に送り証明書を付けてもらい、それを送り返してもらう形で、JITR または JITP を行うことが可能です。
この方式がセキュリティ強度、実装コスト低減度ともより高くなります。

2.2 フリートプロビジョニング方式

フリートプロビジョニングは、IoT デバイス毎に、初回起動時にプロビジョニングを行う手法で
大きく分けて二つの方式が存在します。

1. クレームによるプロビジョニング (プロビジョニングクレーム証明書を用いたアプローチ)
2. 信頼できるユーザー (モバイル / Web アプリユーザー等) によるプロビジョニング

またフリートプロビジョニングは個別の証明書と秘密鍵の取得手順について二つの方式が存在します。

A) AWS 認証局により新しい個別の証明書と秘密鍵を作成してもらいデバイスに送信する。
(CreateKeysAndCertificate)

B) 鍵ペアをデバイス内部で作成し、証明書署名リクエスト(CSR)を AWS に送付することで個別の証明書のみを作成してもらいデバイスに送信する。
(CreateCertificateFromCsr)

本書では、1.および B)の方式の組み合わせ(図 2-4 参照)でフリートプロビジョニングデモを実装しています。

本書で解説している上記プロビジョニング方式には下記のメリットがあります。

メリット :

- デバイス秘密鍵がデバイスの外に出ない
- 生産工場が AWS IoT に接続する必要がない
- デバイスごとに個別の証明書を発行して登録する手順が不要になる

一方で、下記のデメリットもあり、双方を理解して使用することが必要です。

デメリット :

- プロビジョニングクレーム証明書が流出した場合を考慮する必要がある
- デバイス側でプロビジョニングのリクエストや受け取りを行う実装が必要

2.3 クレームによるプロビジョニング (クレーム証明書を用いたアプローチ)

デバイスはプロビジョニングクレーム証明書と秘密鍵が埋め込まれた状態で製造可能です。これらの証明書が AWS IoT に登録されている場合、AWS IoT はそれらをデバイスが通常のオペレーションで使用できる一意のデバイス証明書と交換できます。このプロセスには、以下のステップが含まれます。

クレームによるプロビジョニングは、全デバイスが共通のプロビジョニングクレーム証明書を使用して製造されるシナリオを想定して設計されています。これらのプロビジョニングクレーム証明書には、デバイスに次のことのみを許可します。

1. AWS IoT Core との最初の接続を確立する
2. アイデンティティの証明をする
3. デバイスが以降の通信で使用する必要な権限が付与された ID をリクエストする

全デバイス共通のプロビジョニングクレーム証明書は、工場などで初期ソフトウェアと共にデバイスに書き込まれます。デバイスに既に固有の秘密鍵が搭載されている場合、AWS IoT Core によって署名されるプロビジョニングクレーム証明書とともに証明書署名リクエスト (CSR) を送信することが可能(図 2-4 参照)です。

フリートプロビジョニングは、デバイスによって提示されたプロビジョニングクレーム証明書の検証に加えて、関連するデバイスの属性が適切であるかを検証するための、Lambda ベースのプロビジョニングフックも利用可能です。デバイス属性の例には、シリアル番号、MAC ID、デバイスの場所などが含まれます。このプロセス中に送信されたカスタム属性に基づいて特定のデバイスのプロビジョニングステータスの承認または拒否を自動化するには、プロビジョニングトランザクションで Lambda 関数の利用を検討してください。

（本アプリケーションノートのデモプロジェクトでは Lambda は使用しません）

AWS Lambda を使用したプロビジョニングの方法は下記を参照ください。

https://docs.aws.amazon.com/ja_jp/iot/latest/developerguide/provision-wo-cert.html
「AWS CLI での事前プロビジョニングフックの使用」

2.3.1 クレームによるプロビジョニングの概要 (プロビジョニングクレーム証明書利用時)

デバイスに電源が供給されている かつ ネットワーク接続が可能な場合、以下のワークフローが実行されます。

CreateKeysAndCertificate 方式および CreateCertificateFromCsr 方式におけるワークフローは図 2-3 および図 2-4 も参照してください。

また下記に本書で解説するフリートプロビジョニングデモのベースとなった AWS IoT Fleet Provisioning Demo の詳細なワークフローを確認することができます(CreateCertificateFromCsr 方式)。

https://aws.github.io/aws-iot-device-sdk-embedded-C/latest/docs/doxygen/output/html/fleet_provisioning_demo.html

1. デバイスは、事前にデバイスに書き込まれたクレーム証明書を使用して、安全な TLS 1.2 接続を介して AWS IoT Core に接続します。デバイスに CSR がある場合、それはプロビジョニングクレーム証明書とともに提示されます。
2. 証明書には非常に制限の厳しいポリシーが関連付けられており、フリートプロビジョニングプロセスに関連付けられた IoT トピックへのアクセスのみを提供します。
3. フリートプロビジョニングサービスは、トランザクションを安全に分離するための「 所有権の証明 」をするトークンと、正規の証明書 / 秘密鍵ペイロードを返します。このトークンは、「4.」の MQTT リクエストで提示され、証明書をアクティブ化するためのテンプレート処理を開始するために使用されます。 CSR が提示された場合、証明書はその CSR から生成されます。
4. デバイスは AWS IoT Core に MQTT リクエストを送信し、所有権トークン、アカウント所有者によって作成されたフリートプロビジョニングテンプレートの名前、および（ オプションで ）プロビジョニング検証用のデバイス属性を提示します。 Lambda ベースのプロビジョニングフックを利用して、事前に承認されたリストに対するデバイスのシリアル番号や MAC ID の検証など、追加の検証を有効にすることが推奨されます。
5. フリートプロビジョニングテンプレートが実行され、プロビジョニングトランザクションが実行され、結果が返されます。一般的には、 Lambda でのデバイス属性を検証、証明書のアクティブ化、プロダクションポリシーのアタッチ、モノ/ グループの作成などをします（オプション）。
6. プロビジョニングトランザクションの結果に基づいて、新しい証明書のステータスが返されます。成功した場合、プロビジョニングクレーム証明書は不要となり、デバイスは有効化されたデバイス証明書を使用して AWS IoT Core に再接続します。トランザクションが拒否された場合、「 アクセス拒否 」エラーがデバイスに返されます。

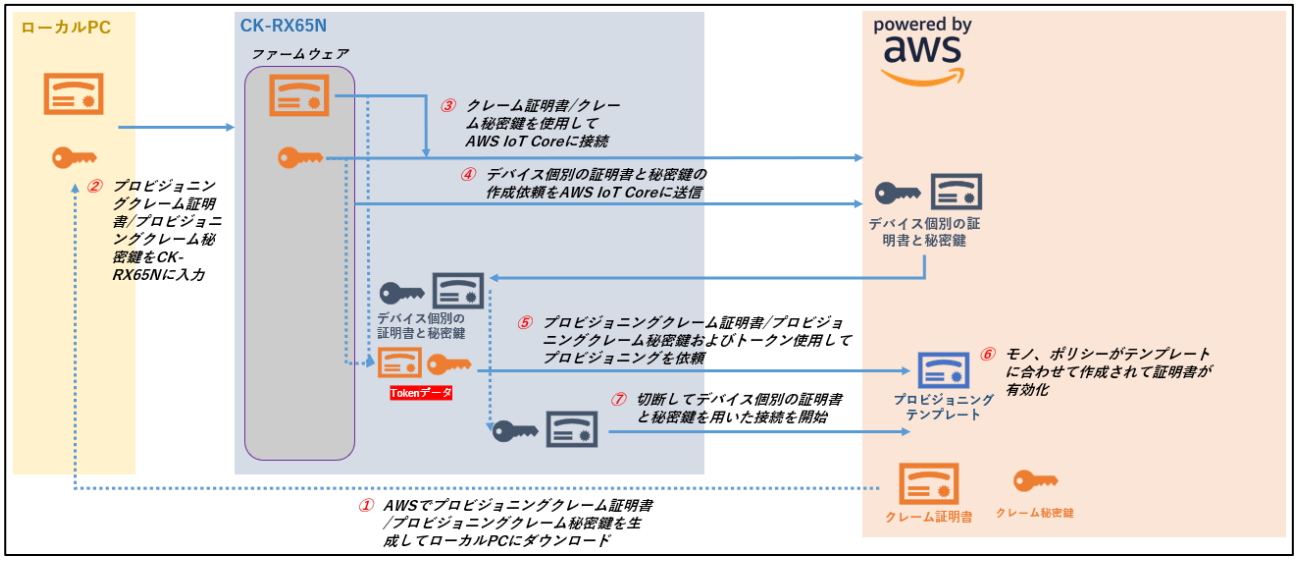

図 2-3 クレームによる CreateKeysAndCertificate 方式でのプロビジョニングのワークフロー

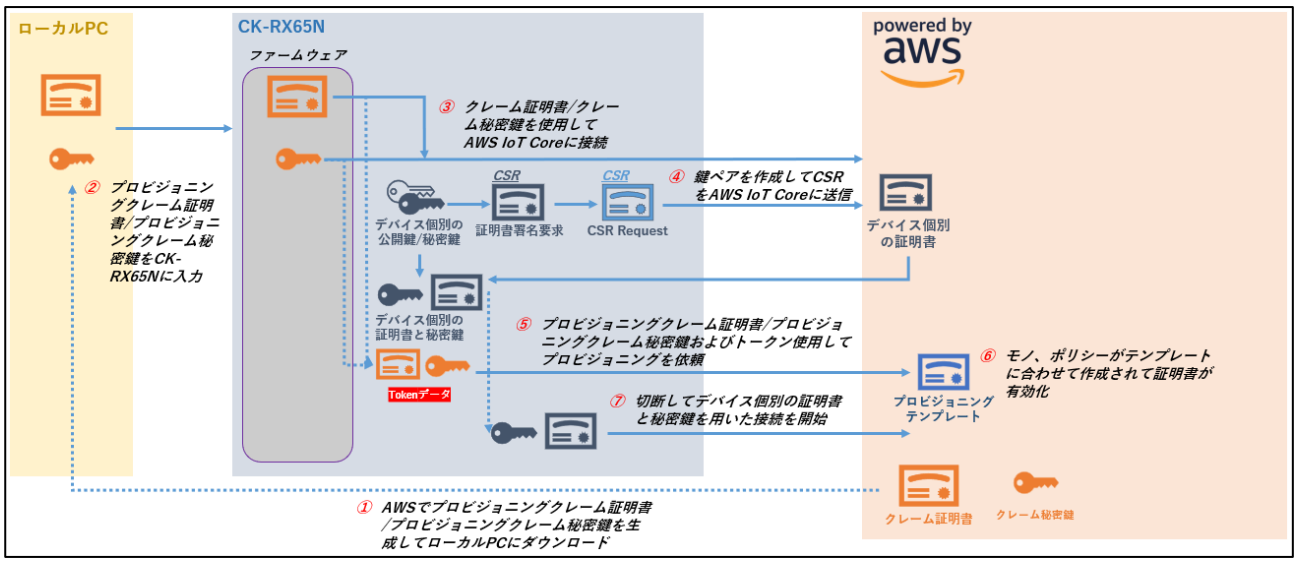

図 2-4 クレームによる CreateCertificateFromCsr 方式でのプロビジョニングのワークフロー

2.3.2 ユニークなモノの名前の決定方法

フリートプロビジョニング実行時に、各デバイスでモノの名前が重複しないように MQTT リクエストを送信する際にデバイスのシリアル番号をペイロードに含ませることができます。

シリアル番号の決定方法は大きく分けてふたつの方式があります。

1. 乱数生成器が生成したランダム値またはデバイスが持つユニークな ID 値をシリアル番号として使用する。
2. 暫定的に決定したシリアル番号を Lambda ベースのプロビジョニングフックおよび Amazon S3 や任意のデータベースを使用することで、ユニークなシリアル番号に変更する。

本書では、RX ファミリに搭載されているユニーク ID を活用することで 1.の方法でモノの名前が重複することを防止しています。

図 2-5 ランダム値またはユニークな ID を使用したモノの名前の決定方法

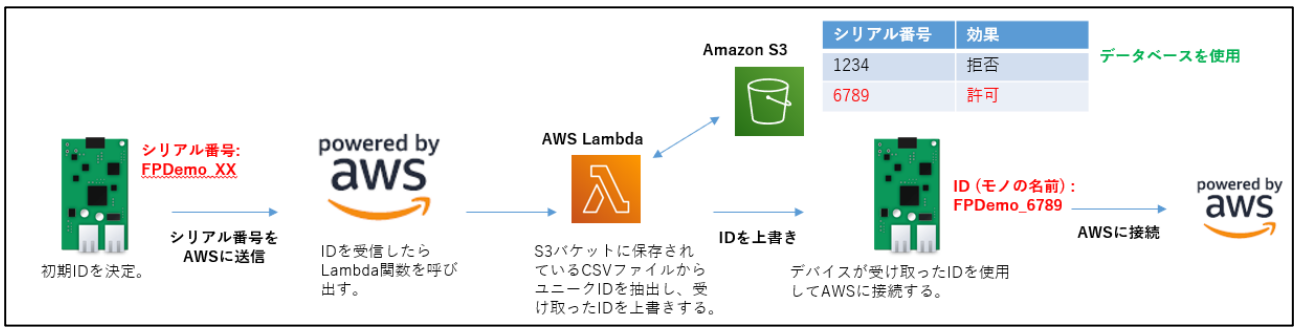

図 2-6 Amazon S3 またはデータベースを使用したモノの名前の決定方法

3. 準備

本章以降、本アプリケーションノートに同梱するプロジェクトのインポートから、CK-RX65N v2 ボードでフリートプロビジョニングのデモを実行するまでの流れを説明します。

3.1 ハードウェア環境

本デモプロジェクトのハードウェア環境を下表に示します。

表 3-1 ハードウェア構成

Item	Content	Provider	Description
使用ボード	CK-RX65N v2	Renesas Electronics Corporation	RX65N MCU 搭載のクラウドキット
PC	Windows11	—	デモのホスト PC

3.2 ソフトウェア環境

本デモプロジェクトのソフトウェア環境を下表に示します。

表 3-2 ソフトウェア構成

Item	Content	Version	Description
統合開発環境	e^2 studio	v2025-12	—
ツールチェーン	CC-RX	v3.07.00	—
	GCC for Renesas RX	v14.2.0.202505	—
FreeRTOS	v202406.04-LTS-rx	V1.2.0	—
通信ソフト	Tera Term	Ver 5.5.0	ログ表示用
AWS コマンドラインインターフェイス（CLI）	AWS Command Line Interface	Version 2	AWS 設定用

3.3 Tera Term インストールと設定

デモでは、ログ出力に Tera Term を使用します。

(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)

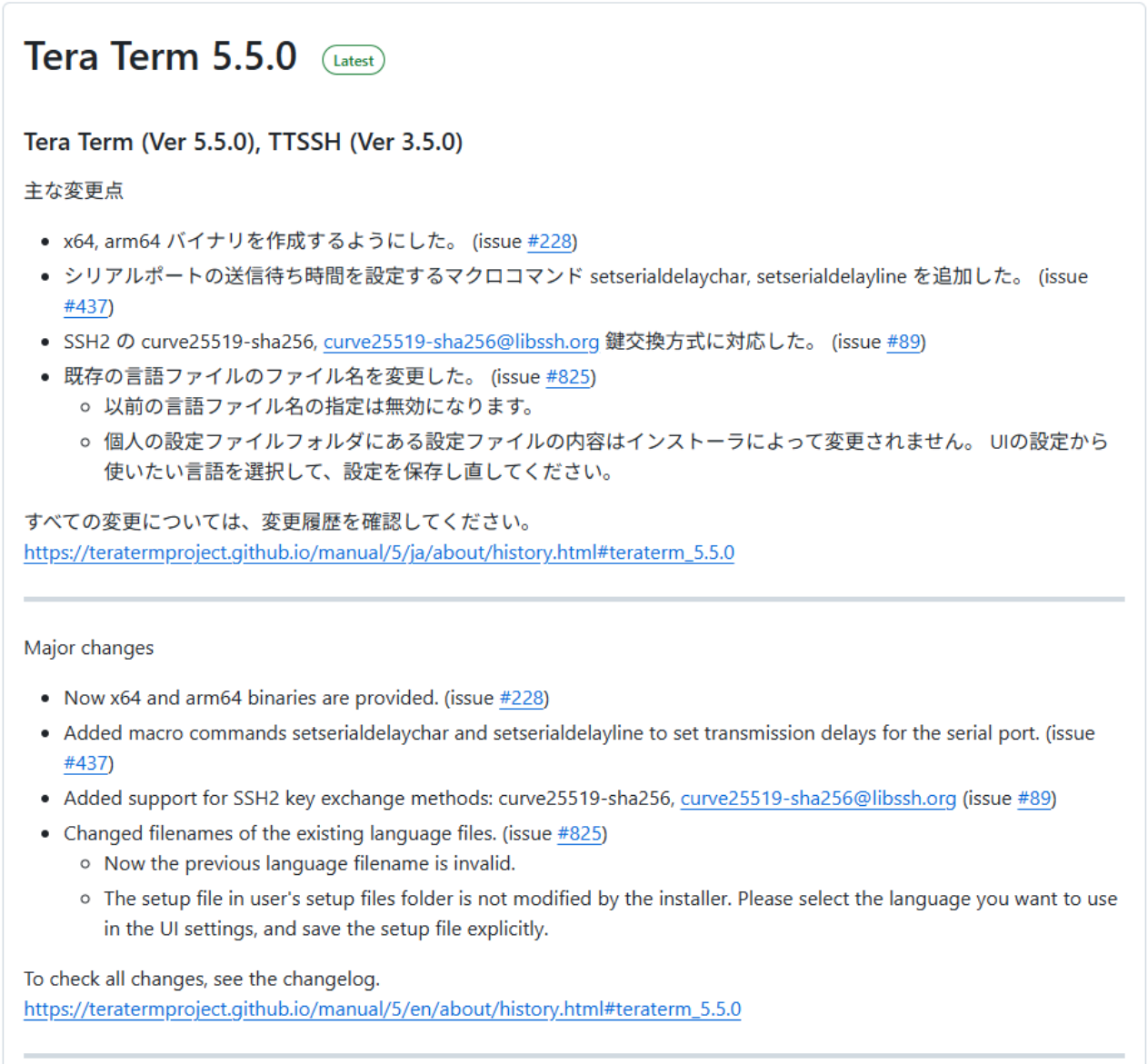

図 3-1 ダウンロードする Tera Tarm

(2) Tera Term インストーラーのダウンロード

使用している OS に適合する最新バージョンの Tera Term の exe ファイルを指定してダウンロードしてください。

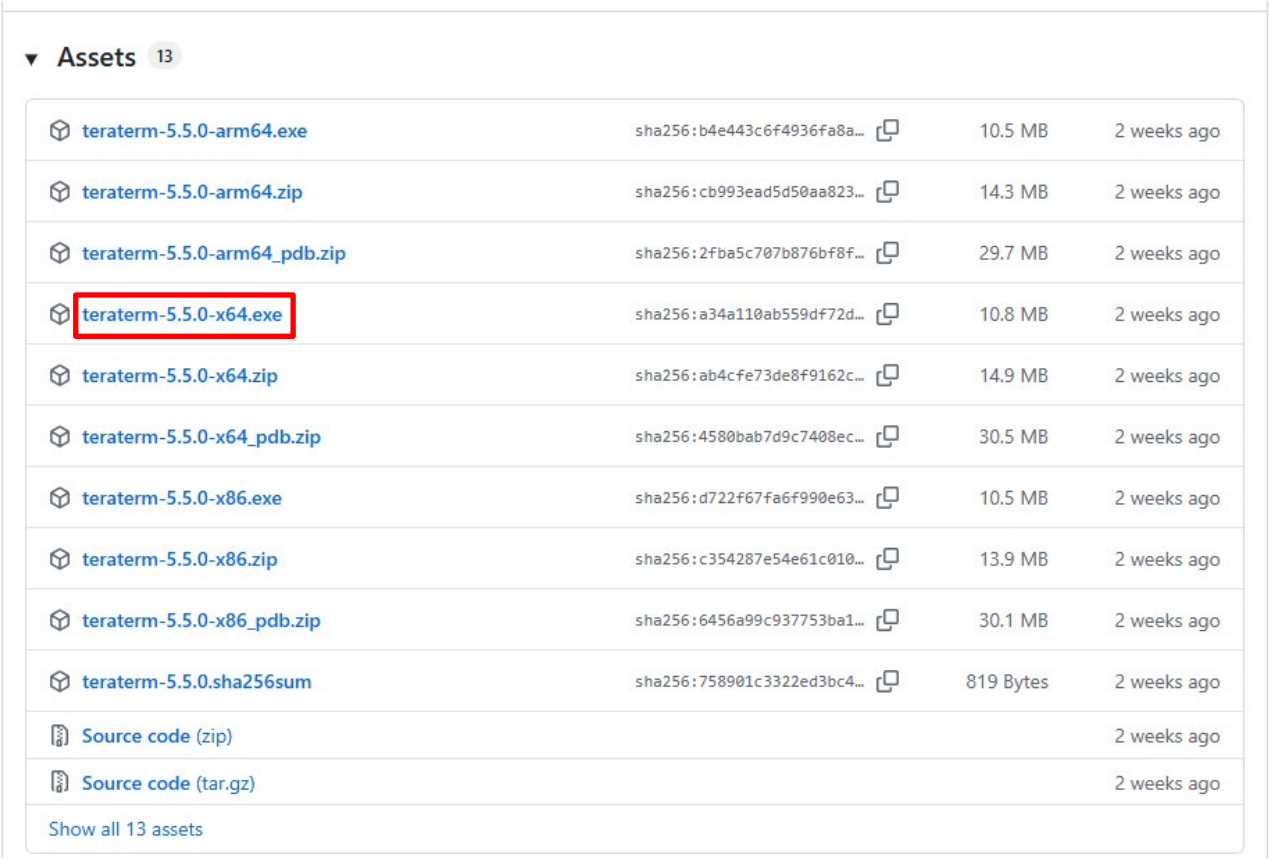

図 3-2 ダウンロードファイルの選択

(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。

3.4 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。

【注】PowerShell をご利用の際は Version7 以降の使用を推奨します。
【注】各コマンドを本アプリケーションノートの文字列をコピーしてコマンドラインシェルに貼り付けて実行する際は、コマンド間のスペースが半角スペースになっていることを確認してください。
まれにご利用の環境によっては貼り付け後に半角スペースが全角スペースになる場合があります。

(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
```

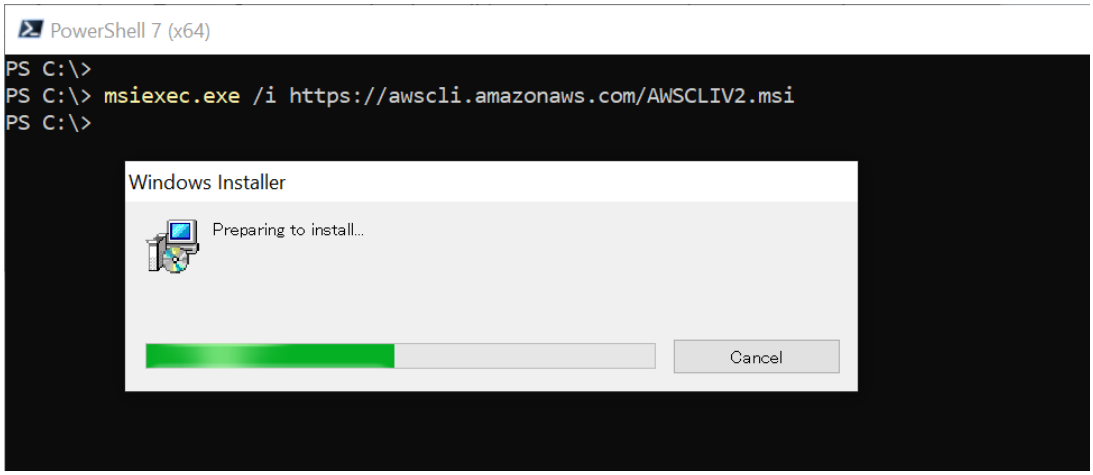

図 3-3 AWS CLI のダウンロードとインストーラーの実行

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。

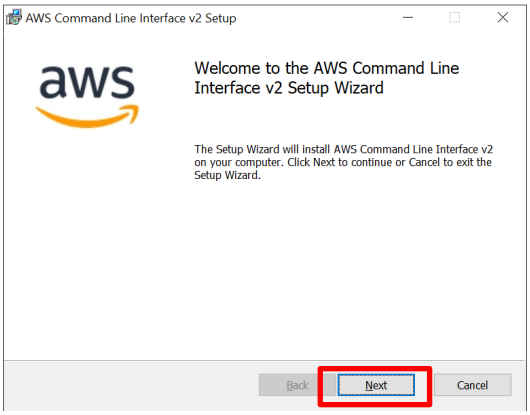

図 3-4 AWS CLI のインストール完了

(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。
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。

```
> aws --version
```

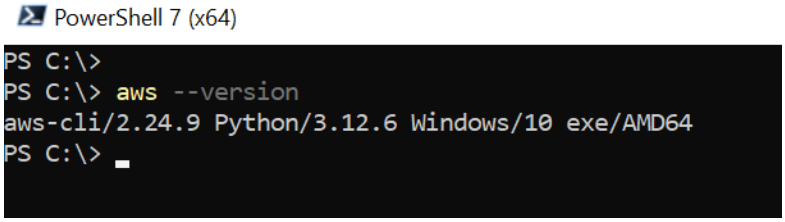

図 3-5 AWS CLI のバージョン確認

【注】 ここで表示される Python のバージョンは AWS CLI の実行ファイルに組み込まれている Python のライブラリのバージョンを示しています。システムにインストールされている Python のバージョンとは異なる場合がありますが、特に問題はありません。

3.5 FreeRTOS プロジェクト

本デモプロジェクトのソフトウェア構成を**図 3-1** に示します。

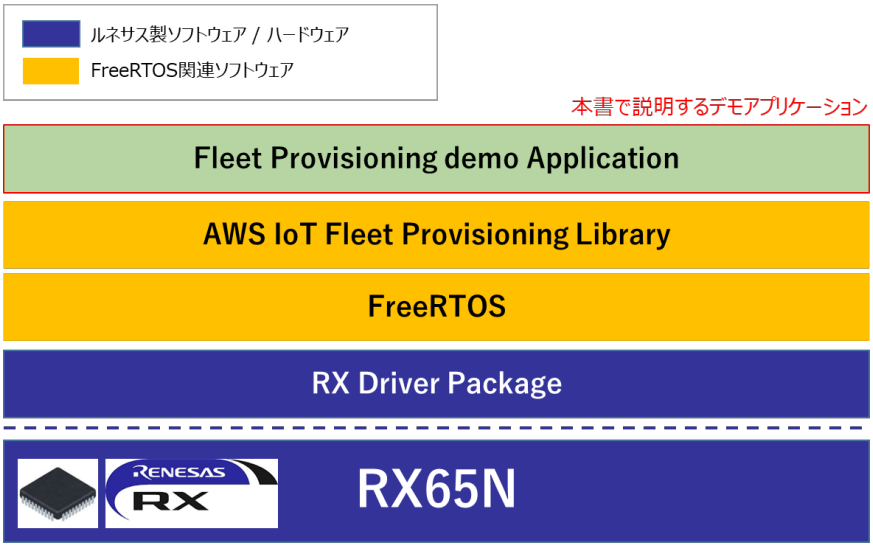

図 3-6 本アプリケーションノートに含まれるデモプロジェクトの構成イメージ

フリートプロビジョニング機能の実現のために、FreeRTOS の AWS IoT Fleet Provisioning Library を使用しています。
RX Driver Package、FreeRTOS、AWS IoT Fleet Provisioning Library およびデモアプリケーションは下記の GitHub のリポジトリに配置されています。デモアプリケーションセットの取得方法については、「4.4 サンプルプロジェクトの作成」を参照してください。
（デモアプリケーションセットには RX Driver Package、FreeRTOS、AWS IoT Fleet Provisioning Library が含まれています）

- デモアプリケーション ： https://github.com/renesas/iot-reference-rx

4. フリートプロビジョニング デモの実行方法

フリートプロビジョニングデモアプリケーションについて、実行方法を説明します。

4.1 実行環境の準備

デモを実行するための環境を準備します。

CK-RX65N v2 ボードを用いた例を図 4-1 に示します。

AWS に接続するための通信インターフェースとして Ethernet または Cellular 及び Wi-Fi を使用することができます。
本アプリケーションノートでは Ethernet を使用した例として説明します。

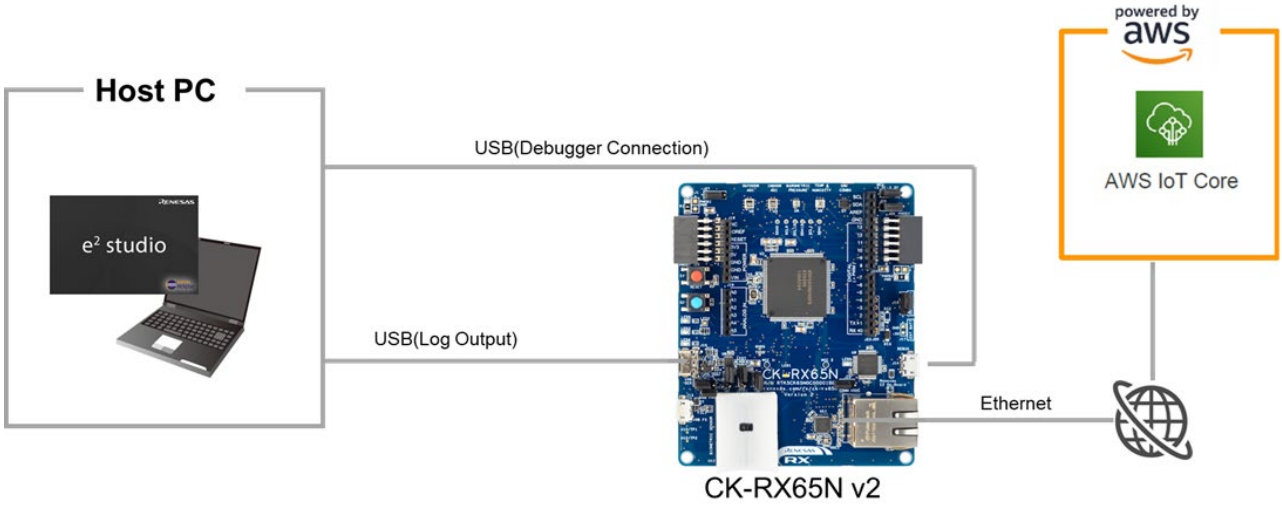

図 4-1 デモ実行環境

【注】 図 4-1 は Ethernet を使用した場合の接続例です。
コネクティビティごとの接続方法については、アプリケーションノート「RX ファミリ Amazon Web Services を利用した FreeRTOS OTA の実現方法(202406-LTS 版)」（R01AN7662）の「2.6 CK-RX65N v2 の接続」を参照してください。

4.2 AWS の準備

本章ではフリートプロビジョニングデモアプリケーションを実行するための AWS の設定手順を説明します。

また、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。

AWS のアカウントの作成から AWS CLI のセットアップと初期設定については、アプリケーションノート「RX ファミリ Amazon Web Services を利用した FreeRTOS OTA の実現方法(202406-LTS 版)」（R01AN7662）の「3.1 AWS サインイン環境の作成」を参照してください。

上記 APN にも記載がありますが、本アプリケーションノートにて AWS CLI を利用する際には以下の点に留意してください。

(a)作業フォルダの作成

アプリケーションノート R01AN7662 の「3.1.5 作業フォルダの作成」を参照し、事前に AWS CLI で作業を行うための任意のフォルダを作成してください。
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。

(b)AWS CLI コマンド入力時のプロファイル名

アプリケーションノート R01AN7662 の「3.1.4(2) SSO プロファイルの作成」を参照し、CLI のプロファイルを作成してください。
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>*はここ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。

4.3 フリートプロビジョニング AWS の設定

フリートプロビジョニングのデモを実行するための AWS 設定を行います。
主な手順は以下のとおりです。

1. ポリシーの設定
2. クレーム証明書、クレーム鍵の生成
3. フリートプロビジョニングテンプレートの作成

AWS の設定には AWS CLI を使用します。
アプリケーションノート「RX ファミリ Amazon Web Services を利用した FreeRTOS OTA の実現方法(202406-LTS 版)」（R01AN7662）の「3.2 CLI を使用した AWS への SSO ログイン」を参照し、作業前に AWS CLI にて SSO ログインを行ってください。

4.3.1 フリートプロビジョニングデモのポリシー作成

フリートプロビジョニングデモ実行時に使用するポリシーを作成します。

(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「4.2 AWS の準備」で作成した作業フォルダに「policy_fp.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_fp.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive",
        "iot:RetainPublish"
      ],
      "Resource": [
        "arn:aws:iot:<region name>:<account id>:topic/$aws/certificates/create-from-
csr/*",
        "arn:aws:iot:<region name>:<account id>:topic/$aws/provisioning-templates/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:<region name>:<account id>:topicfilter/$aws/certificates/create-
from-csr/*",
        "arn:aws:iot:<region name>:<account id>:topicfilter/$aws/provisioning-
templates/*"
      ]
    }
  ]
}
```

図 4-2 フリートプロビジョニングデモのポリシードキュメント

上記のコードの以下の部分は、ご利用の環境に応じて以下を入力してください。

- <region name>: 使用するリージョンに合わせて入力
- <account id>: ご自身のアカウント ID を入力

【注１】リージョン名は AWS マネジメントコンソール（WEB）で選択したリージョンの文字列です。アジアパシフィック（東京）の場合は、「ap-northeast-1」となります。
【注２】アカウント ID（Account ID）は AWS マネジメントコンソール（WEB）に右上のアカウント名をクリックすることで表示される 12 桁の数字からハイフンを除いたものとなります。

(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 <DEMO_POLICY_NAME> --policy-document
file://policy_fp.json --profile <PROFILE_NAME>
```

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

コマンドを実行すると、以下の様に表示されます。

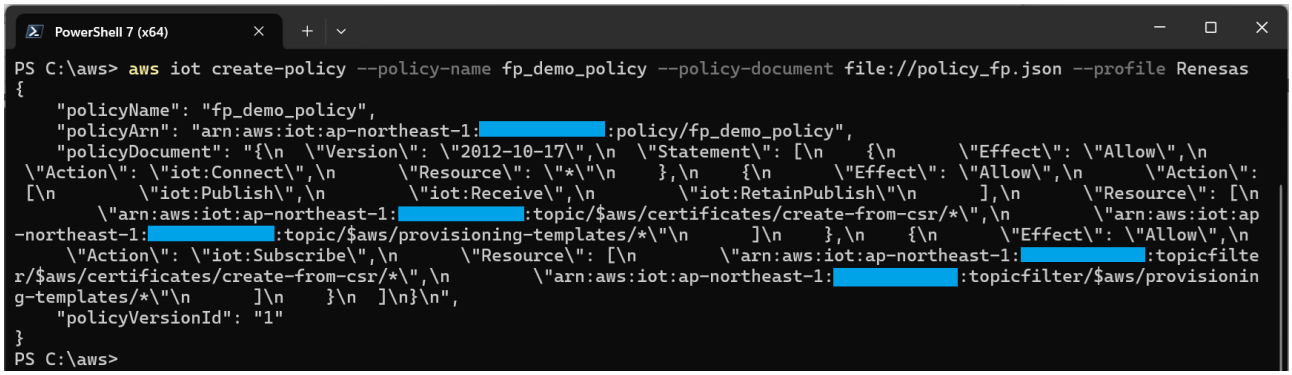

図 4-3 フリートプロビジョニングデモのポリシー作成

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。

4.3.2　フリートプロビジョニングで作成されたモノのポリシー作成

フリートプロビジョニング実行後に作成されたモノにアタッチするポリシーを作成します。
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

【注】ここで作成するポリシーはアプリケーションノート「RX ファミリ Amazon Web Services を利用した FreeRTOS OTA の実現方法(202406-LTS 版)」（R01AN7662）の「3.3.1 ポリシーの設定」で作成するポリシーと同じものです。
すでに OTA の実行でこのポリシーを作成済みの場合はこの節はスキップできます。
この場合は作成したポリシー名を控えておいてください。

(1)　モノの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「4.2 AWS の準備」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": "*"
        }
    ]
}
```

図 4-4 フリートプロビジョニングモノのポリシードキュメント

青字の部分が割り当てるアクセス許可名となります。

(2) モノのポリシーの作成

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 <THING_POLICY_NAME> --policy-document
file://policy.json --profile <PROFILE_NAME>
```

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

コマンドを実行すると、以下の様に表示されます。

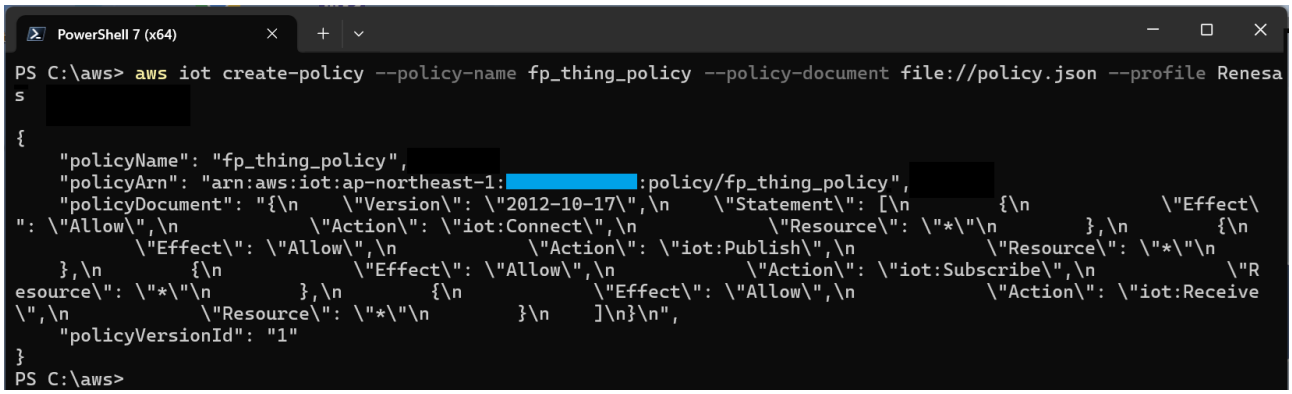

図 4-5 フリートプロビジョニングモノのポリシー作成

作成したモノの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。

4.3.3　プロビジョニングクレーム証明書、鍵ペアの生成

フリートプロビジョニングにおけるプロビジョニングクレーム証明書とプロビジョニングクレーム鍵ペアを生成します。

(1)　プロビジョニングクレーム証明書と鍵ペアの生成とダウンロード

プロビジョニングクレーム証明書を作成してアクティブ化し、プロビジョニングクレーム証明書とプロビジョニングクレーム鍵ペア（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 "fp-certificate.pem.crt" --public-key-outfile "fp-public.pem.key" --
private-key-outfile "fp-private.pem.key" --profile <PROFILE_NAME>
```

コマンドを実行するとプロビジョニングクレーム証明書・鍵ペアが作成され、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）。

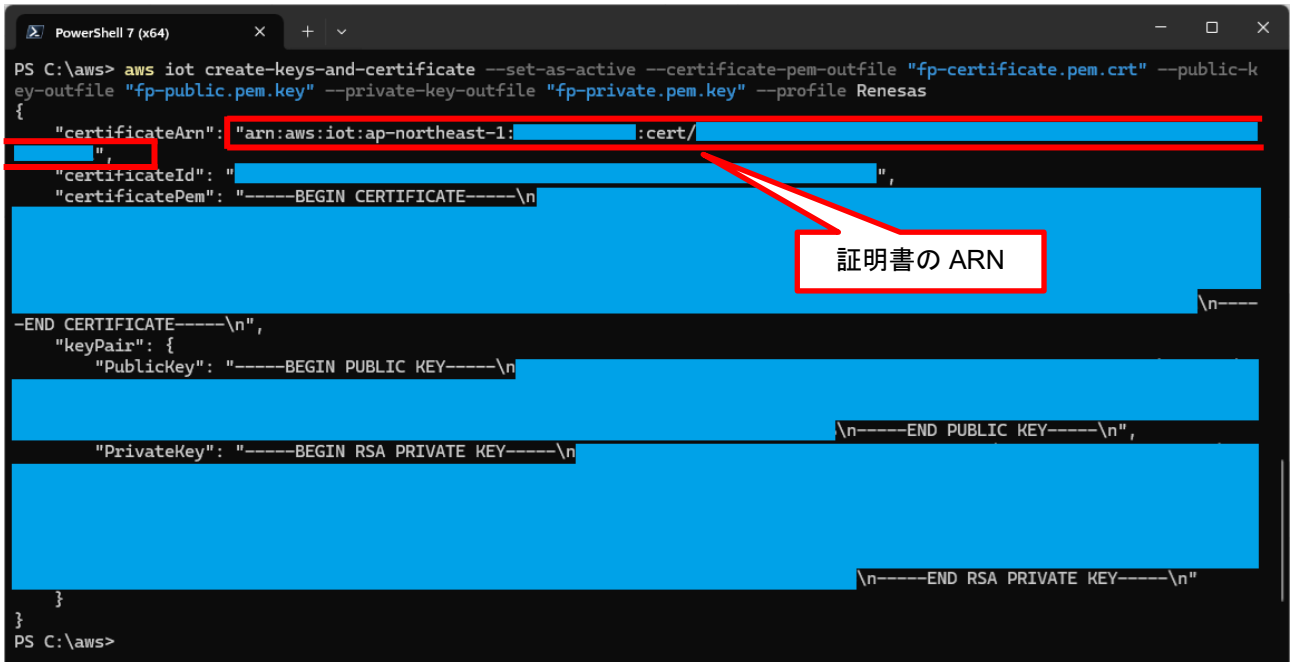

図 4-6 プロビジョニングクレーム証明書・鍵の生成とダウンロード

表示されたプロビジョニングクレーム証明書の 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「4.2 AWS の準備」で作成した作業フォルダにダウンロードされます。

表 4-1 ダウンロードしたプロビジョニングクレーム証明書・鍵ファイル

ファイル名	内容
fp-certificate.pem.crt	プロビジョニングクレーム証明書
fp-public.pem.key	プロビジョニングクレーム公開鍵
fp-private.pem.key	プロビジョニングクレーム秘密鍵

証明書、秘密鍵はフリートプロビジョニングにおいて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】プロビジョニングクレーム証明書・公開鍵・秘密鍵は証明書生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。

(2) クレーム証明書にポリシーをアタッチ

作成したプロビジョニングクレーム証明書にフリートプロビジョニングデモの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 <DEMO_POLICY_NAME> --target
<CERTIFICATE_ARN> --profile <PROFILE_NAME>
```

- <DEMO_POLICY_NAME>には「4.3.1(2) ポリシーの作成」で作成したフリートプロビジョニングデモのポリシー名を入力します。
- <CERTIFICATE_ARN>には「(1)プロビジョニングクレーム証明書と鍵の生成とダウンロード」で作成したプロビジョニングクレーム証明書の 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（成功すると実行結果は何も表示されません）。

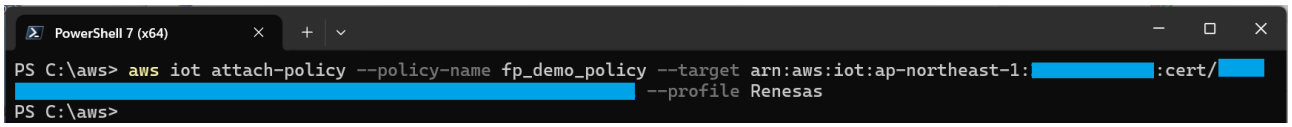

図 4-7 プロビジョニングクレーム証明書にポリシーをアタッチ

以上でプロビジョニングクレーム証明書・鍵の生成に関する設定は完了です。

4.3.4 ロールの作成

フリートプロビジョニングに使用するロールを作成します。

(1) ロールを設定した json ファイルの作成

ロールの設定ファイルを作成します。テキストエディタで「4.2 AWS の準備」で作成した作業フォルダに「trust-policy_fp.json」というファイルを作成し、以下のコードを入力します。

- trust-policy_fp.json

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

図 4-8 ロールドキュメント

(2) ロールの作成

以下のコマンドを入力して、ロールを作成します。

```
> aws iam create-role --role-name <ROLE_NAME> --assume-role-policy-document
file://trust-policy_fp.json --path /service-role/ --profile <PROFILE_NAME>
```

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

コマンドを実行すると、以下の様に表示されます。

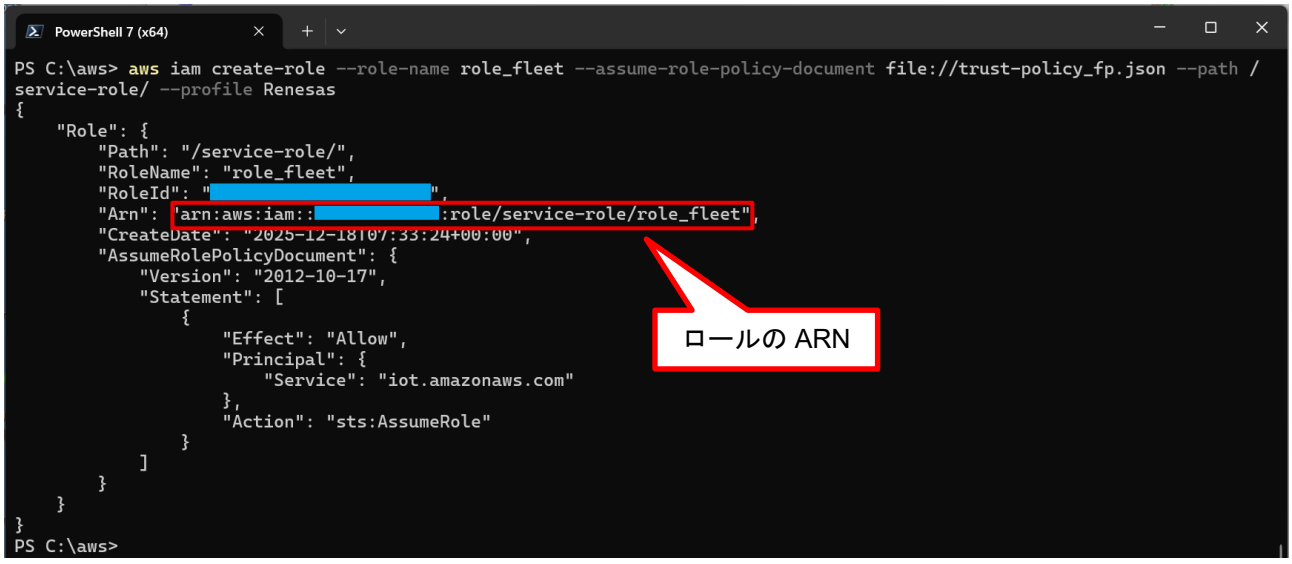

図 4-9 ロールの作成

表示されたロールの 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。
また、作成したロール名も後の処理で使用するため控えておいてください。

(3) ロールに許可ポリシーをアタッチ

以下のコマンドを入力して、ロールにポリシーをアタッチします。

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

- <ROLE_NAME>には「4.3.4(2) ロールの作成」で作成した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。（成功すると実行結果は何も表示されません）。

図 4-10 ロールに許可ポリシーをアタッチ

4.3.5 プロビジョニングテンプレートの作成

IoT デバイスが初回接続時に必要な AWS リソースを自動的に作成・紐付けするための設定となるプロビジョニングのテンプレートを作成します。

(1) テンプレートの json ファイルの作成

プロビジョニングのテンプレートの設定ファイルを作成します。テキストエディタで「4.2 AWS の準備」で作成した作業フォルダに「template.json」というファイルを作成し、次のコードを入力します。

- template.json

```
{
  "Parameters": {
    "SerialNumber": {
      "Type": "String"
    },
    "AWS::IoT::Certificate::Id": {
      "Type": "String"
    }
  },
  "Resources": {
    "policy_<THING_POLICY_NAME>": {
      "Type": "AWS::IoT::Policy",
      "Properties": {
        "PolicyName": "<THING_POLICY_NAME>"
      }
    },
    "certificate": {
      "Type": "AWS::IoT::Certificate",
      "Properties": {
        "CertificateId": {
          "Ref": "AWS::IoT::Certificate::Id"
        },
        "Status": "Active"
      }
    },
    "thing": {
      "Type": "AWS::IoT::Thing",
      "OverrideSettings": {
        "AttributePayload": "MERGE",
        "ThingGroups": "DO_NOTHING",
        "ThingTypeName": "REPLACE"
      },
      "Properties": {
        "AttributePayload": {},
        "ThingGroups": [],
        "ThingName": {
          "Fn::Join": [
            "",
            [
              "<THING_NAME_PREFIX>",
              {
                "Ref": "SerialNumber"
              }
            ]
          ]
        }
      }
    }
  }
}
```

図 4-11 プロビジョニングテンプレートドキュメント

<THING_POLICY_NAME>（2 か所）には「4.3.2(2) モノのポリシーの作成」で作成した、「**フリートプロビジョニングモノ**」のポリシー名を入力します。

- *<THING_NAME_PREFIX>*には任意の文字列を入力します。この文字列と MCU 固有の ID から、AWS に登録されるモノの名前が生成されます。（例：fp_thing_）

(2) プロビジョニングテンプレートの作成

以下のコマンドを入力して、プロビジョニングテンプレートを作成します。

```
> aws iot create-provisioning-template --template-name <TEMPLATE_NAME> --
provisioning-role-arn <ROLE_ARN> --template-body file://template.json --
enabled --profile <PROFILE_NAME>
```

- <TEMPLATE_NAME>には任意のテンプレート名を入力します（例：fp_template）
- <ROLE_ARN>は「4.3.4(2) ロールの作成」で作成したロールの 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。

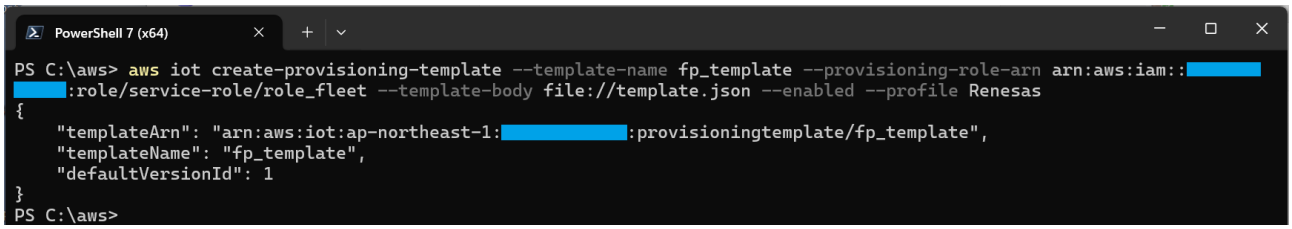

図 4-12 プロビジョニングテンプレートの作成

作成したプロビジョニングテンプレート名は後の処理で使用するため控えておいて下さい。

以上でフリートプロビジョニングに関する AWS の設定は完了です。

4.4 サンプルプロジェクトの作成

この章では、Amazon Web Service を利用した IoT デバイスのためのプロビジョニングを実行するためのサンプルプロジェクト構築方法を説明します。

(1) e² studio の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。
ワークスペースやプロジェクトファイル名はなるべく短くするようにしてください。最下層のフルパスの長さが 259byte を超えるとエラーが発生する場合があります。
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。

【例】C:¥workspace にワークスペースを作成する場合

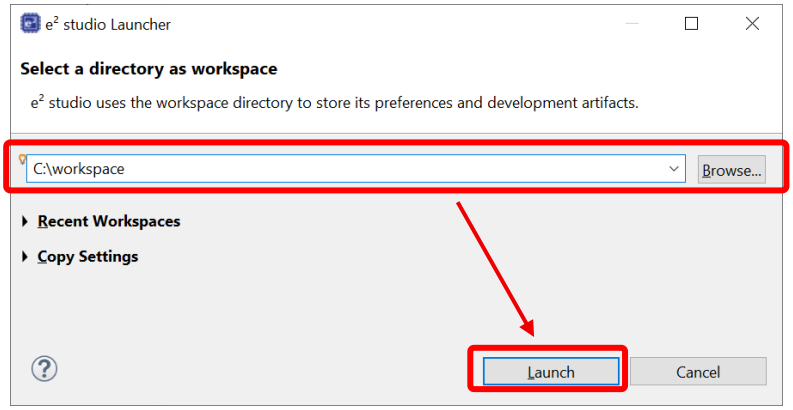

図 4-13 ワークスペース作成画面

(2) プロジェクトをインポートするフォルダの作成

プロジェクトをインポートするフォルダを作成します。
作成したワークスペースのフォルダ下に任意の名前のフォルダを作成してください。
なお、本アプリケーションノートでは「iot-reference-rx」のフォルダを作成しています。

(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、Import（インポート）ダイアログを起ち上げてください。

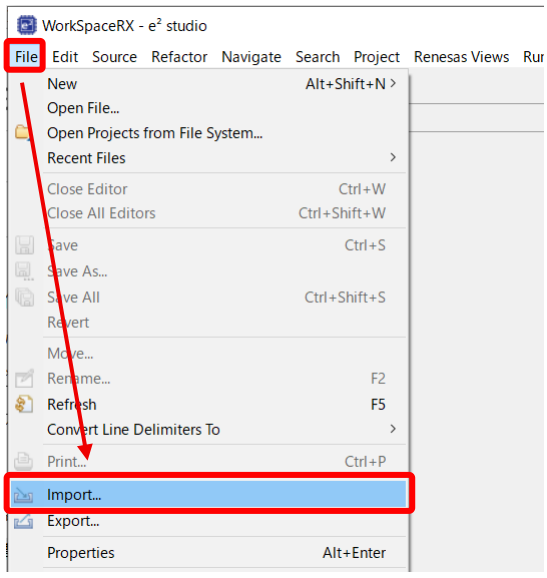

図 4-14 プロジェクトのインポート

(4) インポート方法の選択

Import（インポート）のツリーから[General（一般）]をクリックしてツリーを展開し、[Renesas GitHub FreeRTOS (with IoT libraries) Project（プロジェクト）]を選択し、[Next（次へ）]ボタンを押下して下さい。

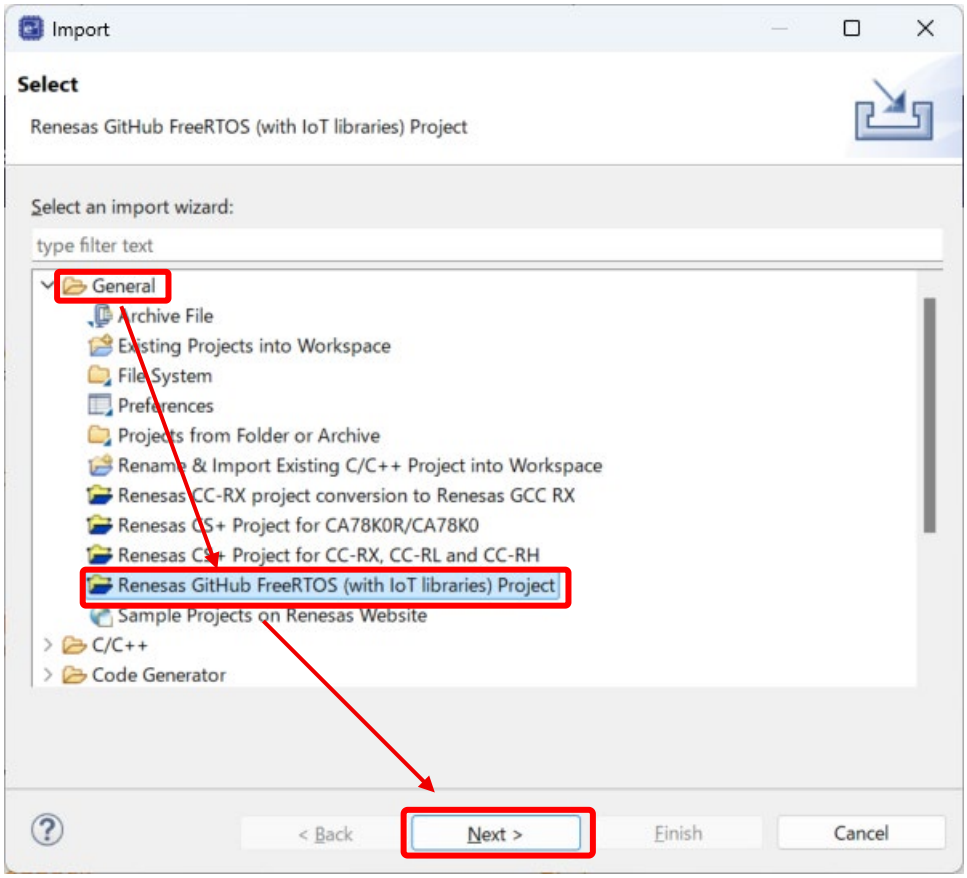

図 4-15 インポート方法の選択

(5) インポートする FreeRTOS のバージョンを選択

インポート可能な RTOS バージョンの選択画面が表示されます。
[Browse（参照）]ボタンをクリックし、「4.4(2) プロジェクトをインポートするフォルダの作成」で作成した、プロジェクトをインポートするフォルダを指定してください。
RTOS Version Setting より"202406.04-LTS-rx-1.2.0"を選択し、[Next（次へ）]ボタンをクリックしてください。GitHub から指定したフォルダへプロジェクトがダウンロードされます。

- 【注】ここでは e² studio に登録された RTOS のダウンロードフォルダにダウンロード済みのバージョンがリストに表示されます

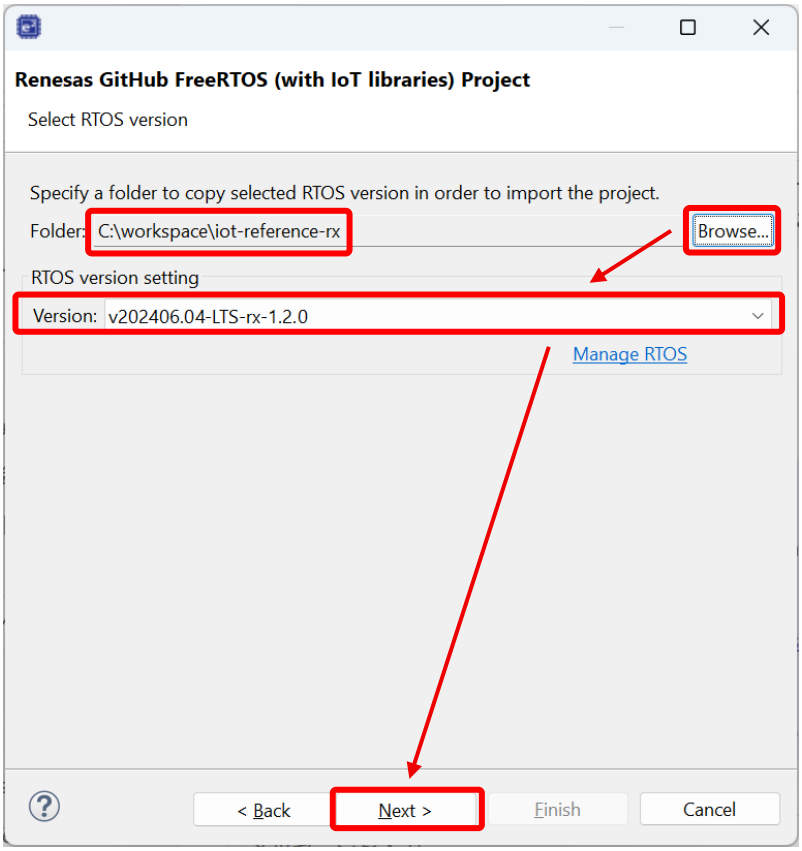

図 4-16 インポートする FreeRTOS の選択

インポート先のフォルダが空ではない場合は、ファイルを上書きするかどうかの確認のダイアログが表示されます。上書きをしていい場合は「Yes（はい）」ボタンをクリックしてください。

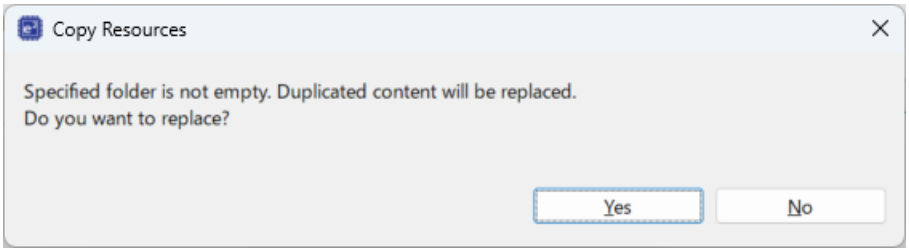

図 4-17 ファイルの上書き確認

また、初めて e² studio をご利用になる場合や、Version（バージョン）リストに必要なバージョンが表示されない場合は、GitHub より FreeRTOS のダウンロードが必要です。
この場合は[Manage RTOS Versions...]をクリックしてください。[Download Confirmation]ダイアログが表示されるので、”FreeRTOS (with IoT libraries) for RX”を選択し、[OK]ボタンをクリックしてください。

図 4-18 FreeRTOS の選択

次に FreeRTOS(with IoT libraries) Module Download ダイアログが表示されるので、使用したいバージョンをチェックして[Download（ダウンロード）]ボタン押下して必要なバージョンをダウンロードしてください。

【注】「Module Folder Path（モジュール・フォルダー・パス）」はフルパスの文字数の制限によるビルドの問題を回避するため、10 文字以内のパス名で設定してください（”c:¥afr”など）。

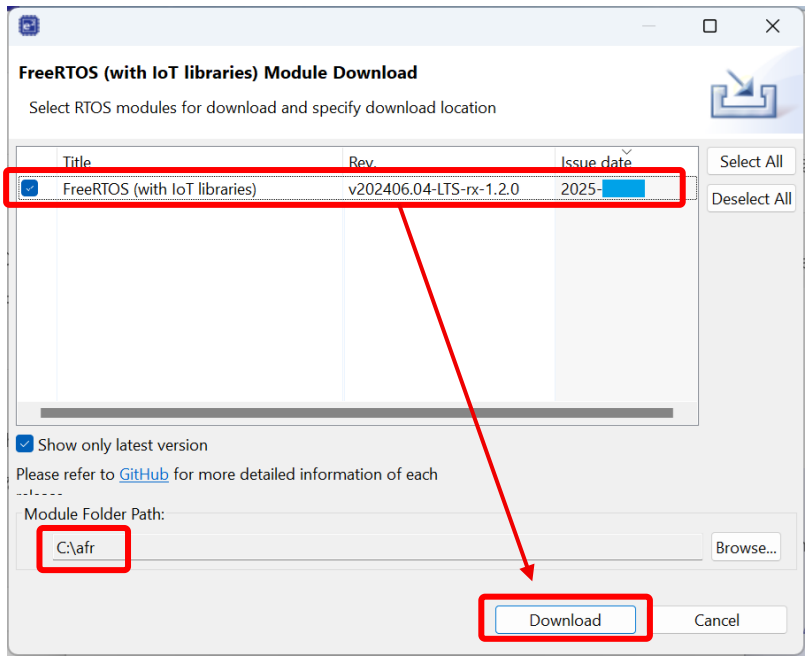

図 4-19 ダウンロードする FreeRTOS のバージョン

ダウンロードが完了すると、「4.4(6) インポートするプロジェクトの選択」のインポートするプロジェクトの選択画面に遷移します。

【注】ダウンロードされるプロジェクトは以下の GitHub のリポジトリを参照しています。
https://github.com/renesas/iot-reference-rx

(6) インポートするプロジェクトの選択

FreeRTOS のバージョンを指定すると Import Project（プロジェクトをインポート）の画面が表示されます。
リストより以下を選択し、[Finish（終了）]ボタン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）

【注】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。

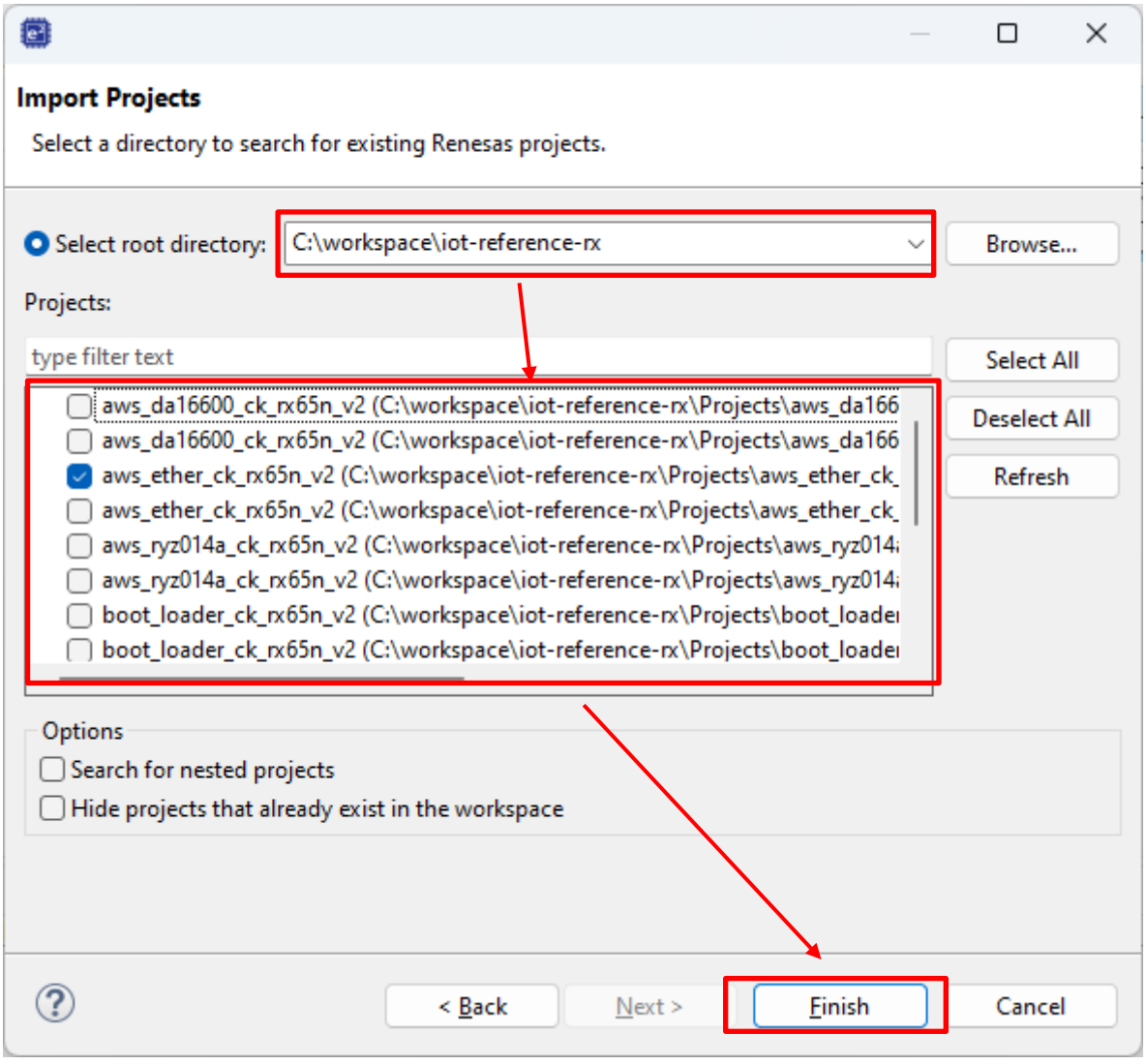

図 4-20 インポートするプロジェクトの選択

(7) インポートプロジェクトの確認

インポートが完了すると、以下のように e² studio にプロジェクトが登録されます。

プロジェクト・エクスプローラーが表示されない場合は、画面右上のパースペクティブの[C/C++]を押下してから、[Window（ウィンドウ）] ⇒[Show View（ビューの表示）] ⇒[Project Explorer（プロジェクト・エクスプローラー）]を選択してください。

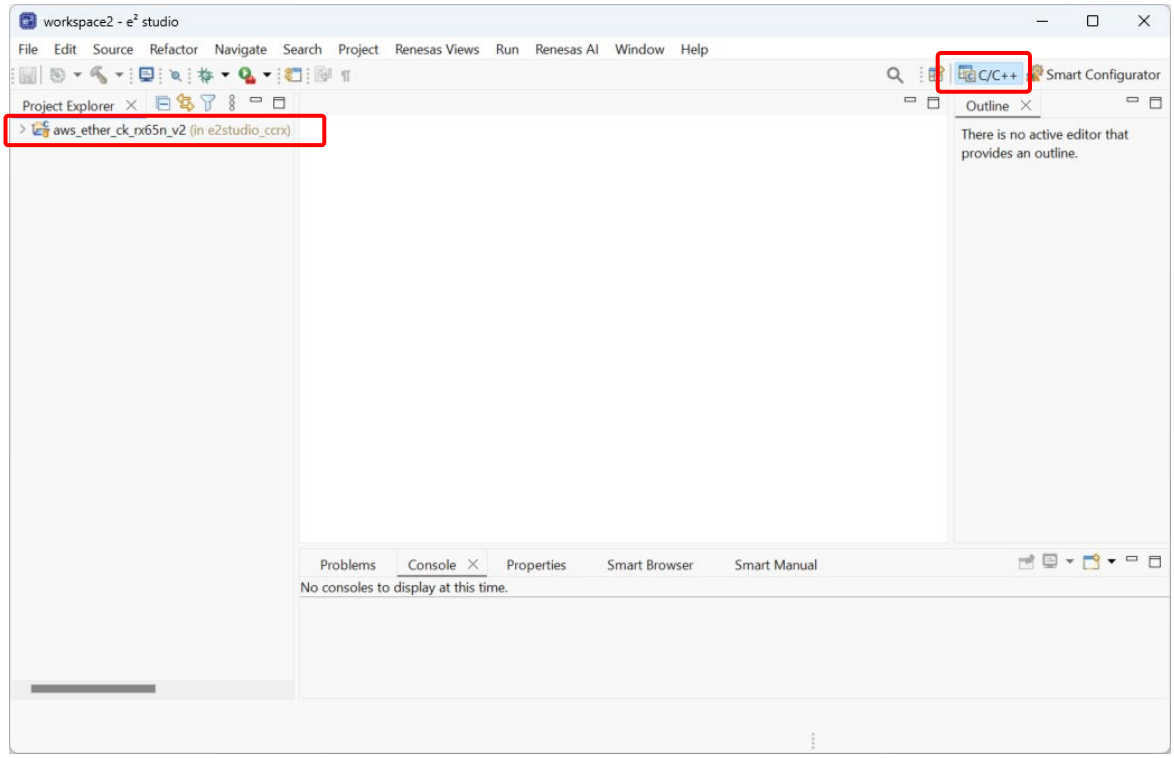

図 4-21 インポート後の画面

また、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。

表 4-2 インポート可能なプロジェクト

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】同じコネクティビティの CC-RX と GCC のプロジェクトは同時にインポートできません。

4.5　プロジェクトの設定

フリートプロビジョニングデモを実行するために必要なプログラムの修正を行います。

4.5.1　コンフィグ・ファイルの修正

e² studio のプロジェクト・エクスプローラーから aws_ether_ck_rx65n_v2/src/frtos_config/demo_config.h を開き、「ENABLE_FLEET_PROVISIONING_DEMO」を「1」に変更します。

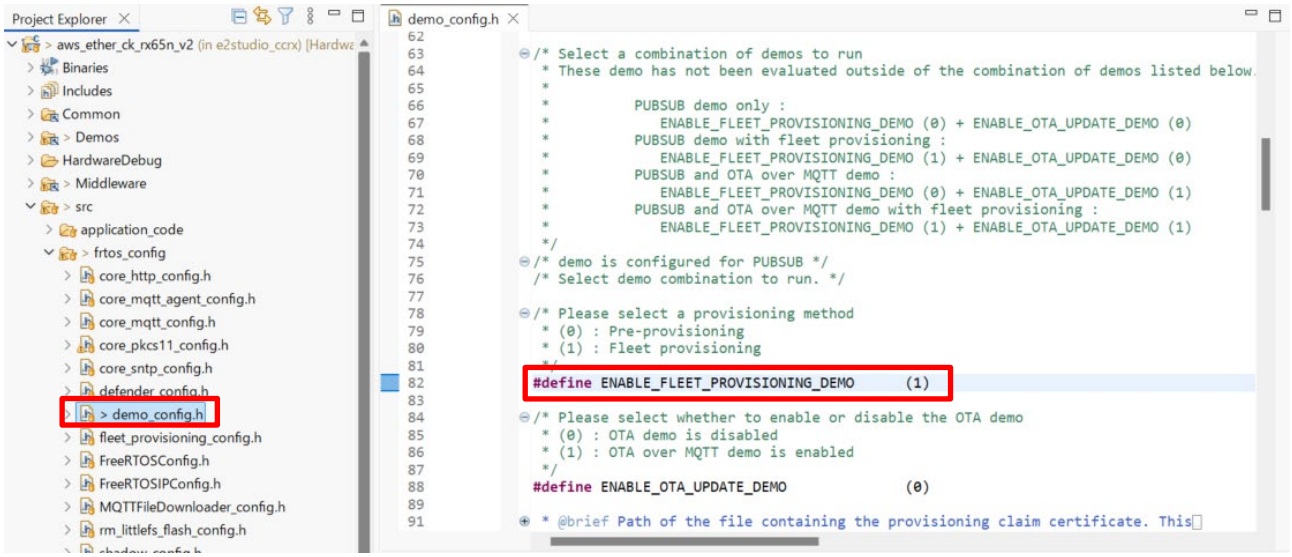

図 4-22 demo_config.h の修正箇所

4.5.2 コネクティビティごとの設定

コネクティビティにセルラーまたは Wi-Fi を使用する場合は、各コネクティビティのモジュールへ設定を行います。

(1) 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]の以下の設定値をご利用の SIM カードに合わせて設定してください。

- [Access point name]：アクセスポイント名を入力します
- [Access point login ID]：ログインする際のユーザーID を入力します
- [Access point password]：ログインする際のパスワードを入力します
- [Authentication protocol type]：認証プロトコルの種別（1: PAP, 2: CHAP）を入力します

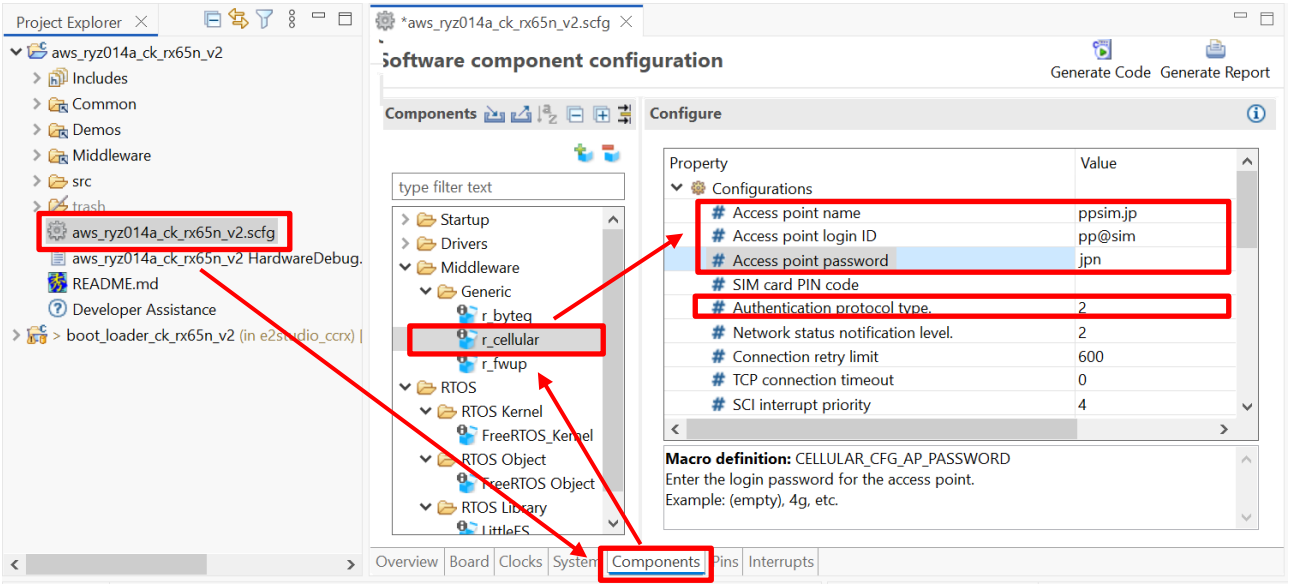

図 4-23 RYZ014A Cellular FIT モジュールの設定

(2) DA16600 Wi-Fi モジュールの設定

AWS 接続に Wi-Fi モジュールを使用する場合は、設定ファイル”aws_clientcredential.h”と Wi-Fi DA16xxx FIT モジュール(r_wifi_da16xxx)へ設定を行います。

(a) 国コードとタイムゾーンの設定

[aws_da16600_ck_rx65n_v2.scfg]を開き、[Components（コンポーネント）]タブを選択し、[r_wifi_da16xxx]の以下の設定を行ってください。

- [Timezone offset in hours]：GMT からのタイムゾーンのオフセットを-12～12 で入力します
- [Country code]：ISO 3166-1 alpha-2 規格で定義されている国コードを入力します（US、JP、CH 等）

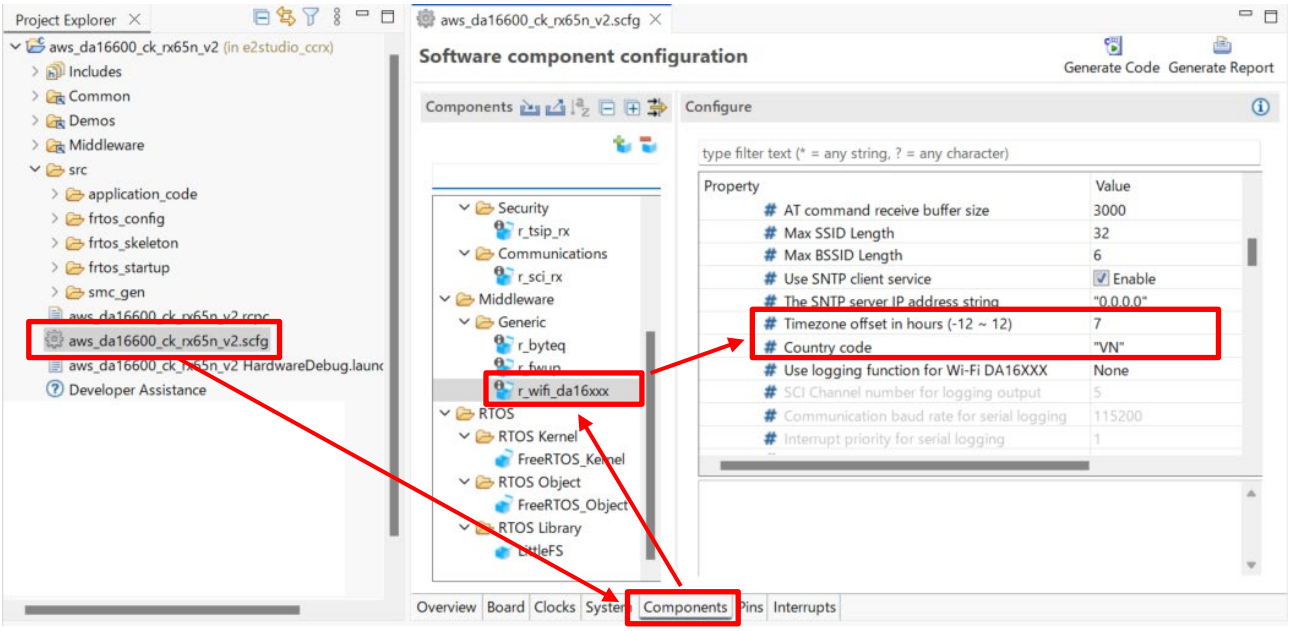

図 4-24 Wi-Fi DA16xxx FIT モジュールの設定

(b) Wi-Fi ネットワークの設定

接続する Wi-Fi ネットワークの設定を行います。
「src¥application_code¥include¥aws_clientcredential.h」で次のマクロを設定します。

- clientcredentialWIFI_SSID：Wi-Fi ネットワークのアクセスポイント名（SSID）を設定します
- clientcredentialWIFI_PASSWORD：Wi-Fi ネットワークパスワードを設定する

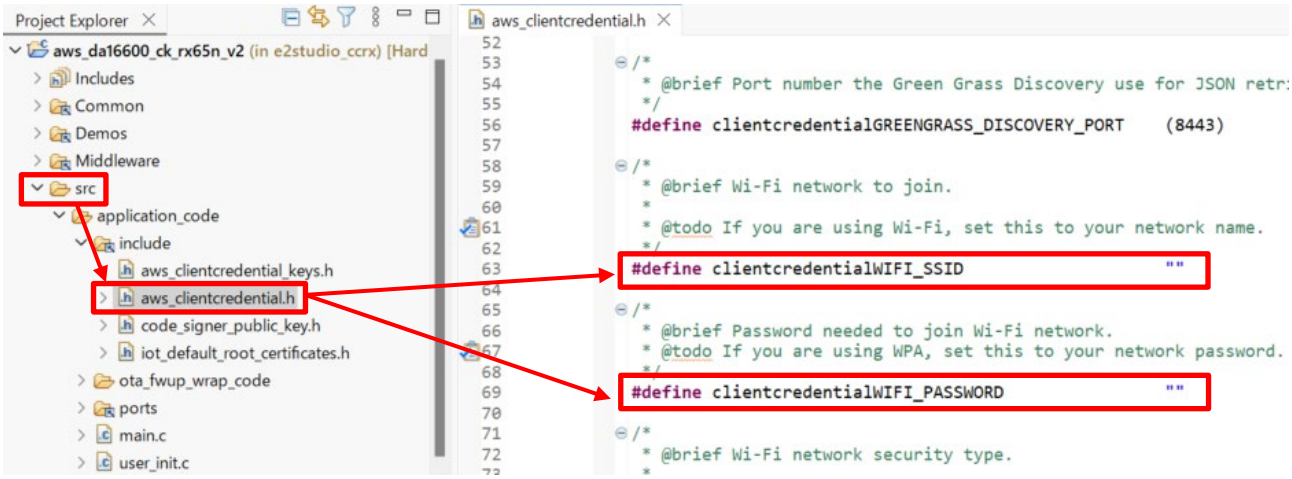

図 4-25 Wi-Fi ネットワークの設定

【注】ダブルクオテーションの間に設定の文字列を入力してください

4.6 プロジェクトの実行

4.6.1 プロジェクトのビルドとダウンロード

プロジェクトをビルド、デバイスに書き込んで、デモを実行します。

最初に、プロジェクト・エクスプローラーで”aws_ether_ck_rx65n_v2”を右クリックし、「プロジェクトのビルド」でビルドを実行します。

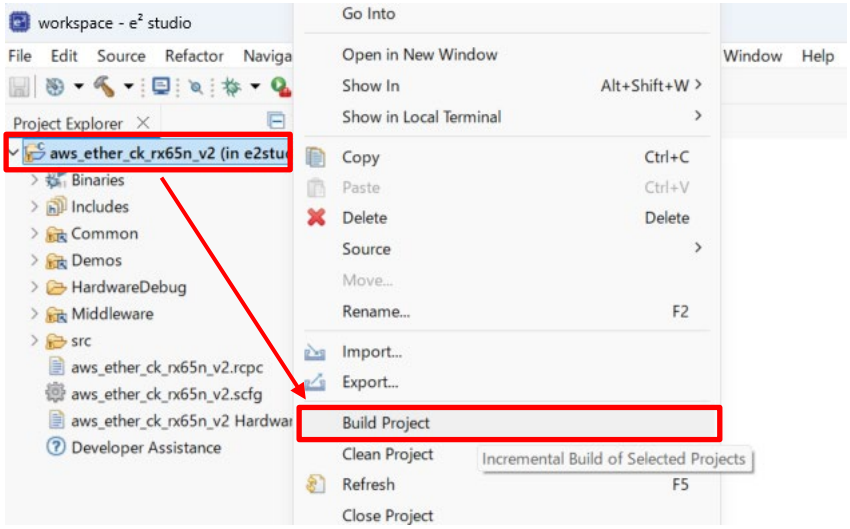

図 4-26 プロジェクトのビルド

ビルドエラーが発生していないことが確認出来たら。e² studio のメニューバーの[Run（実行）] > [Debug（デバッグ）]をクリックしてビルドした実行データをデバイスにダウンロードします。

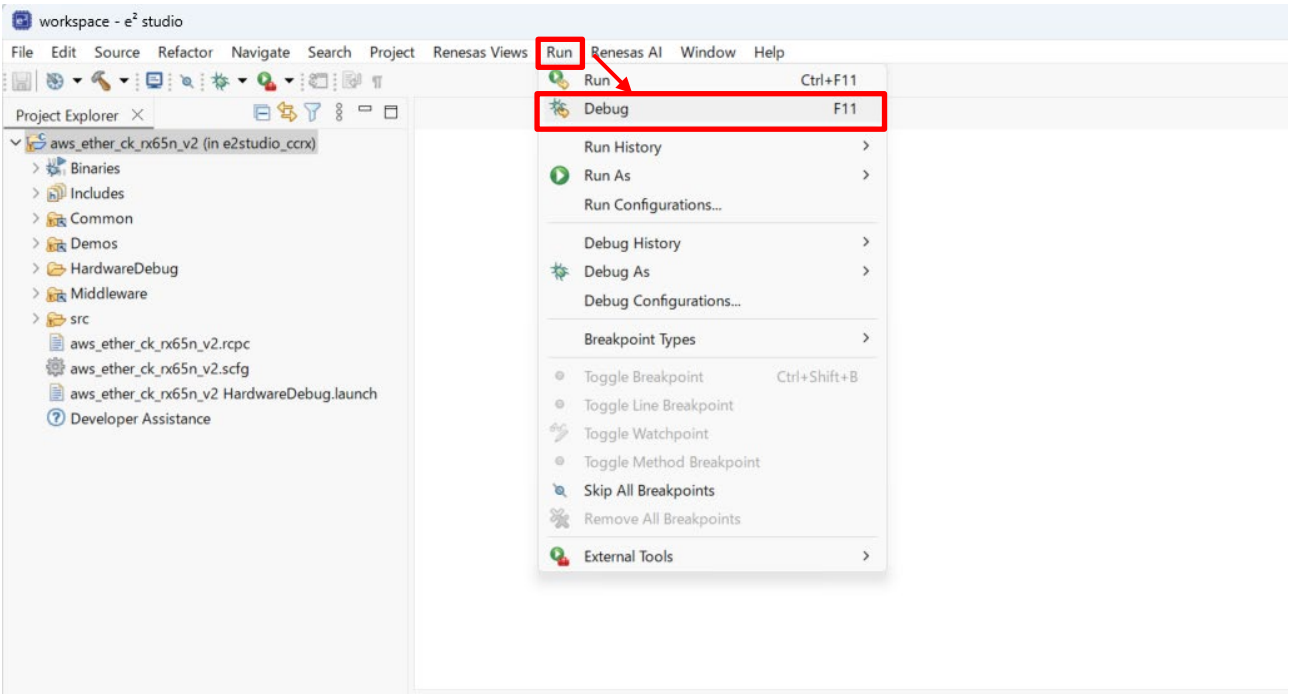

図 4-27 プロジェクトのデバッグ実行

4.6.2 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]を選択します。
CK-RX65N v2 の J10 と接続しているポートを選択し、OK をクリックしてください。

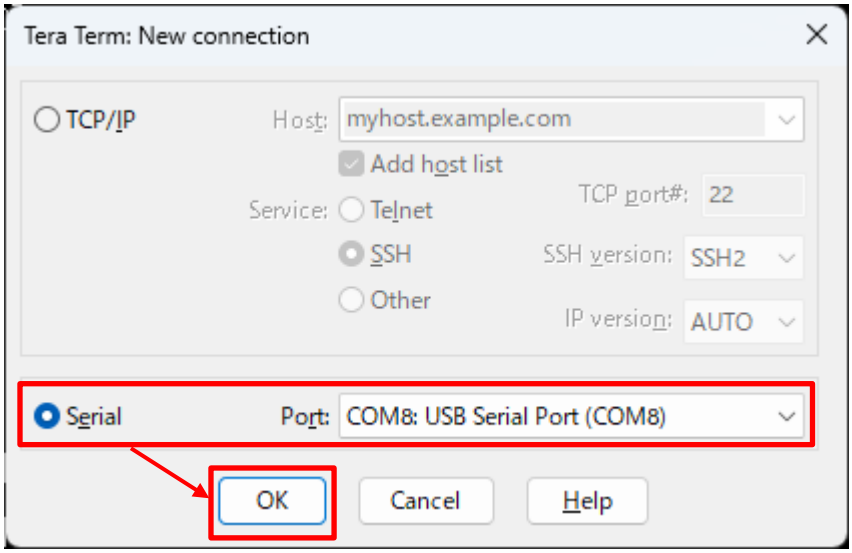

図 4-28 シリアルポートの選択

(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。

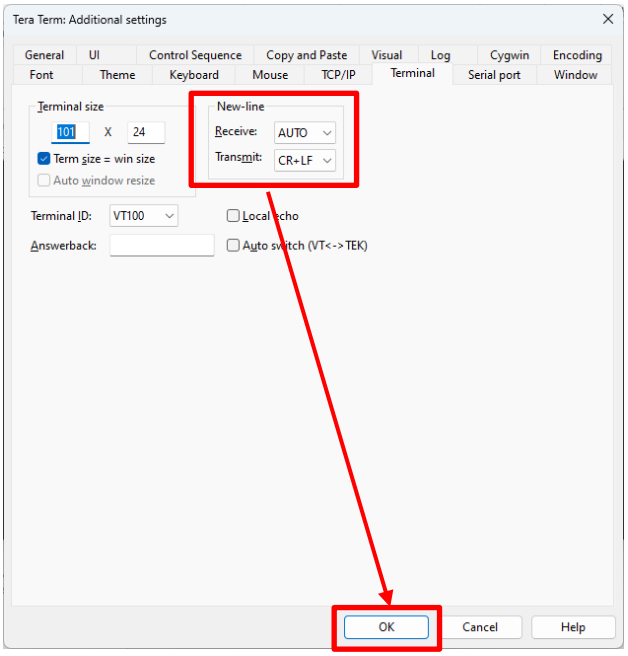

図 4-29 改行コードの設定

(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 OK クリックしてください。

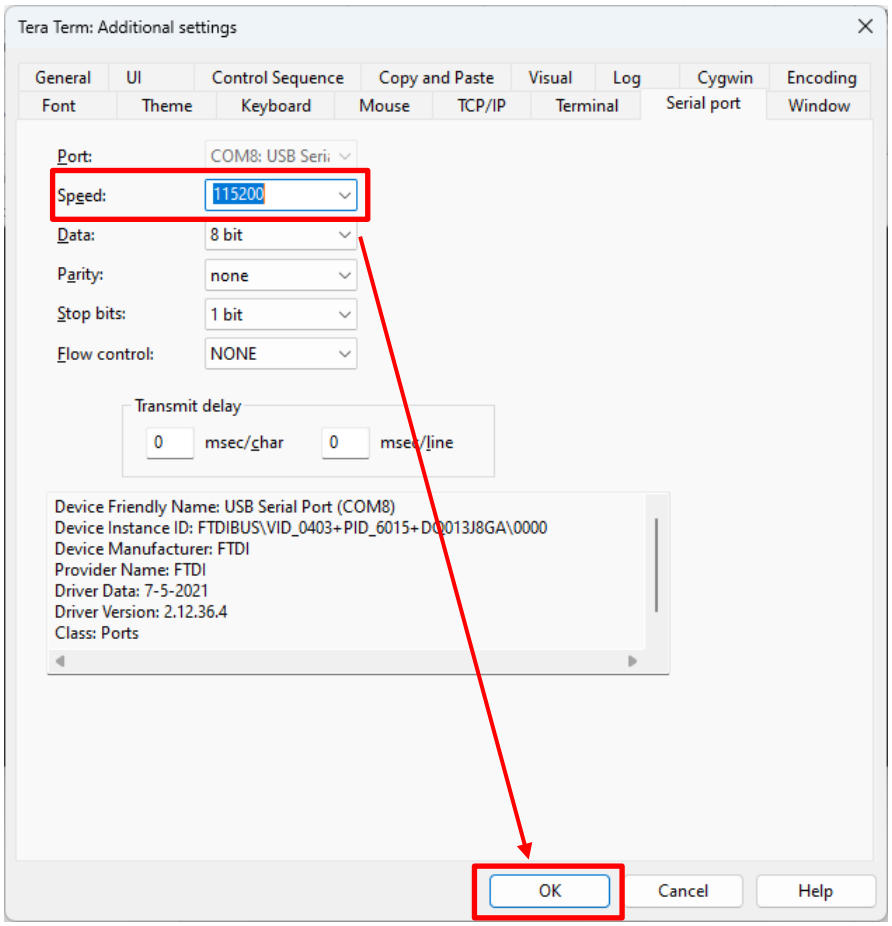

図 4-30 シリアル通信速度の設定

4.6.3　フリートプロビジョニングデモの実行と動作確認

(1)　MQTT テストクライアントの設定

AWS マネジメントコンソールで MQTT テストクライアントを設定することで、デモが送信する MQTT データをモニターすることができます。

AWS マネジメントコンソールの AWS IoT Core で[MQTT test client（MQTT テストクライアント）]を選択して、「Topic filter（トピックのフィルター）」に「**#**」を入力して[Additional configuration（追加設定）]をクリックします。
[MQTT payload display（MQTT ペイロード表示）]から[Display payloads as strings(more accurate)（ペイロードを文字列として表示）]を選択し、「Subscribe（サブスクライブ）」をクリックします。

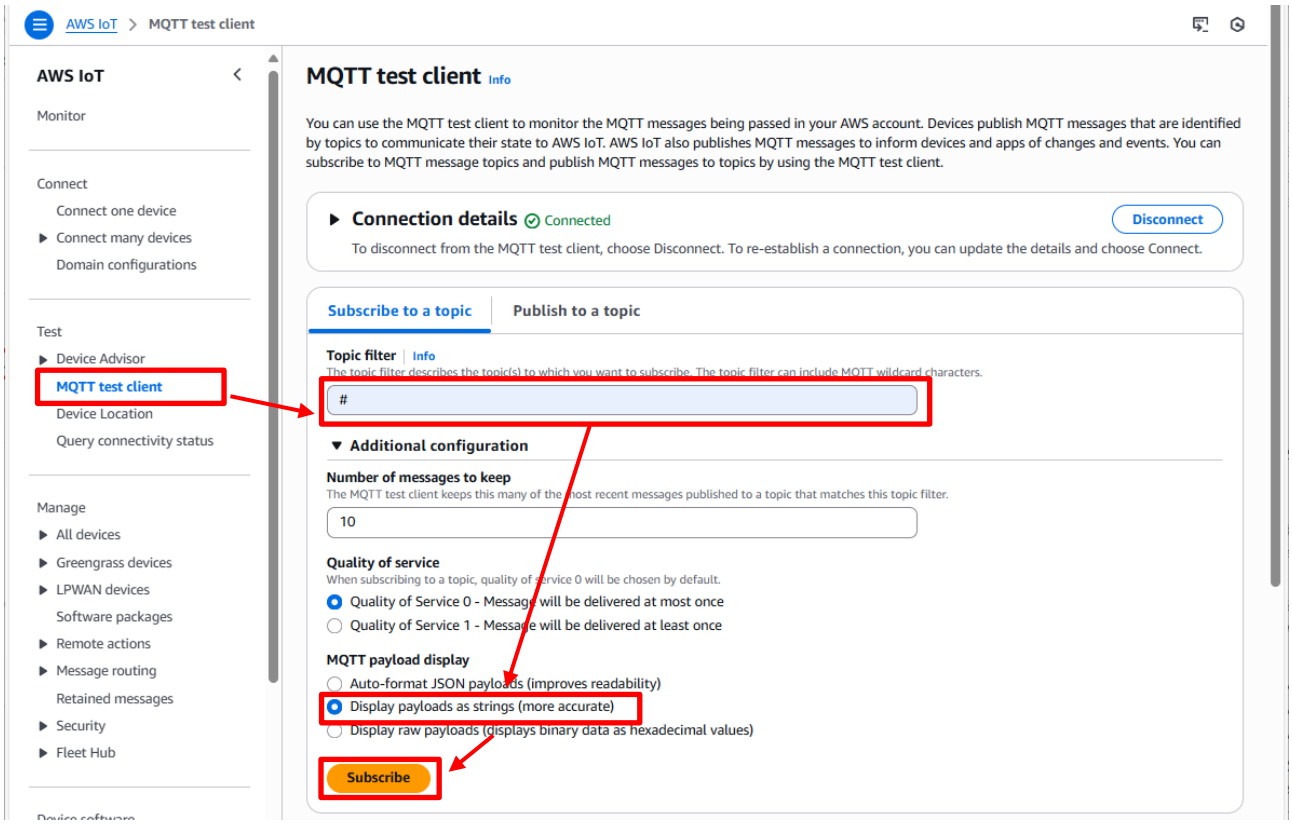

図 4-31 MQTT テストクライアントの設定

(2) フリートプロビジョニングデモの実行

e² studio で[Resume(F8)（再開(F8)）]ボタンを押してください。
デモプロジェクトが実行されます。

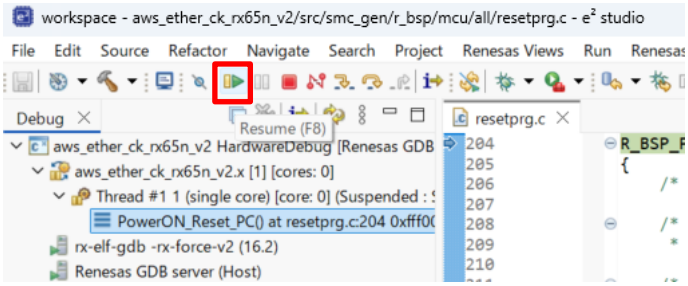

図 4-32 フリートプロビジョニングデモの実行

デモプロジェクトが実行されると、Tera Term 上に以下の様にメニュー画面が表示されるので、10 秒以内に Tera Term 上で[CLI]と入力して[Enter]を押下してください。
CLI モードに切り替わります。

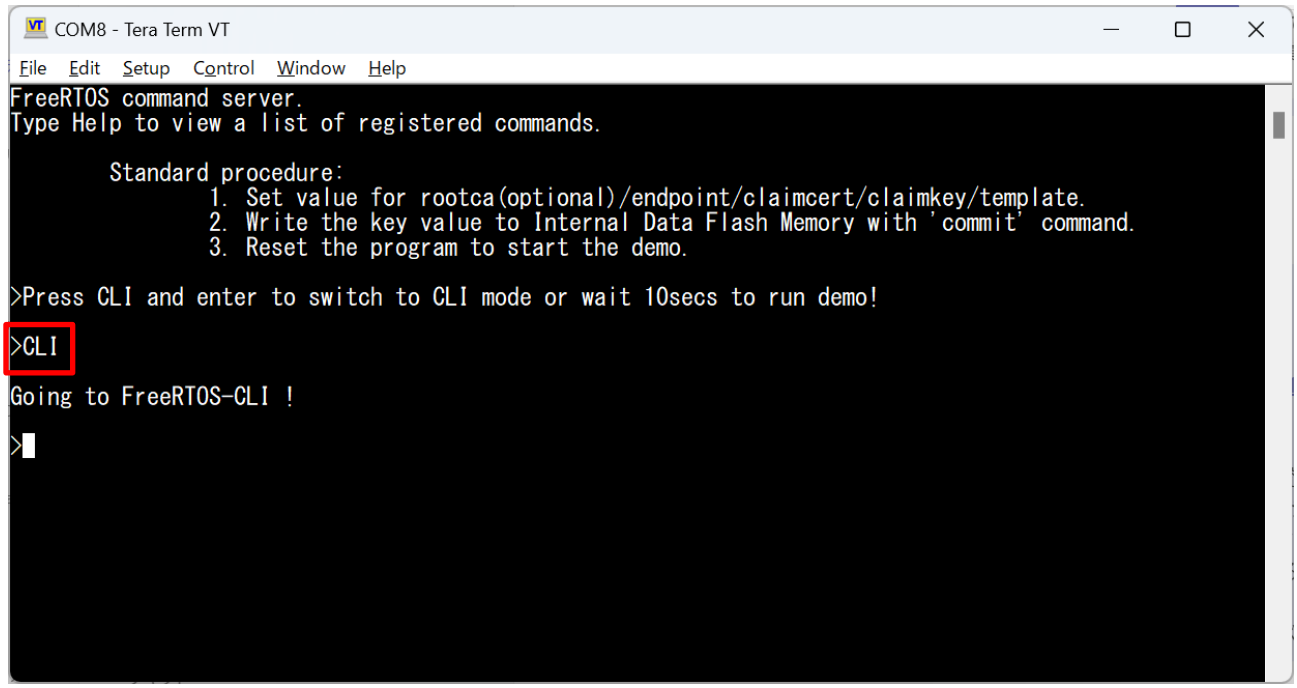

図 4-33 CLI モードへの切り替え

【注】各コマンドを本アプリケーションノートの文字列をコピーして Tera Term に貼り付けて実行する際は、コマンド間のスペースが半角スペースになっていることを確認してください。
まれに、ご利用の環境によっては貼り付け後に半角スペースが全角スペースに変換される場合があります。

(3) AWS 認証情報の登録

デモプロジェクトへ AWS 認証情報を登録します。登録は Tera Term 上でコマンド入力を行います。
ここで入力された情報は、CK-RX65N v2 ボードの MCU のデータフラッシュメモリに記録されます。

(a) 既存のクレデンシャル情報の消去

既にデモ実行のための情報が格納されている可能性があるので、Tera Term 上で「**format**」と入力して「**Enter**」を押します。
“Format OK !”と表示されればデータフラッシュの各種情報の消去は完了です。

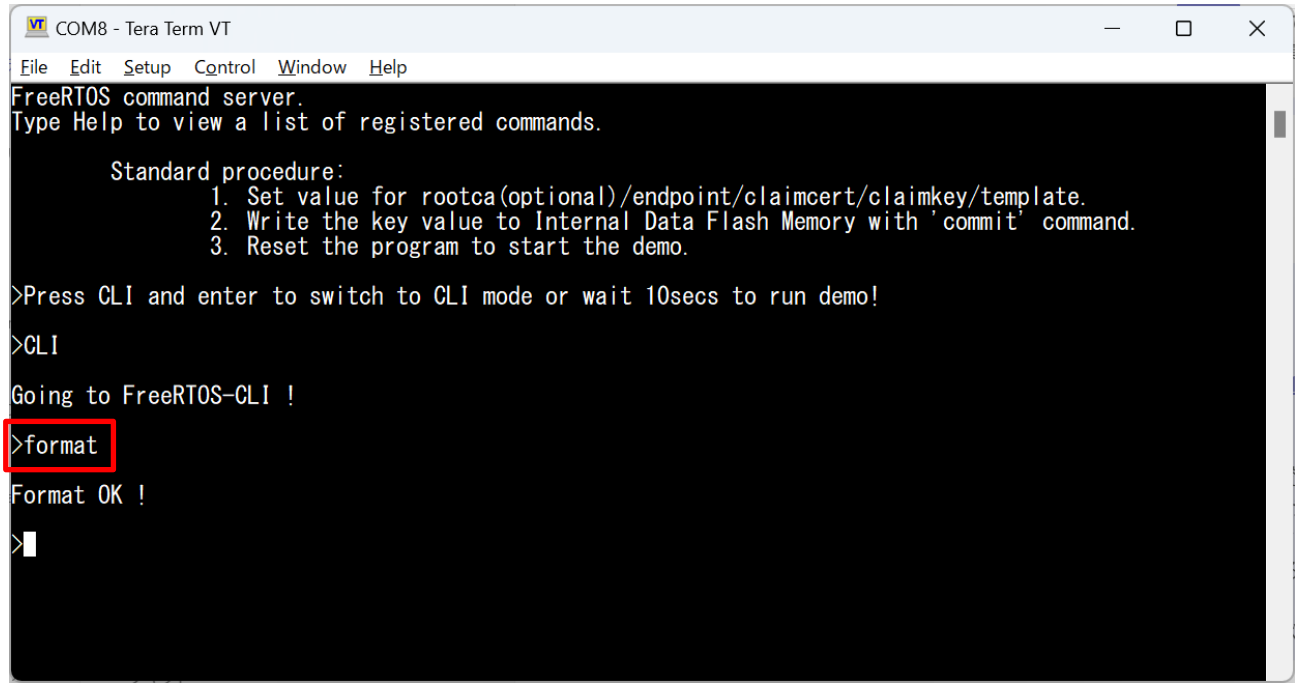

図 4-34 データフラッシュのフォーマット

(b) 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 AWS CLI でコマンドを入力してエンドポイントの確認を行います。

```
> 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。

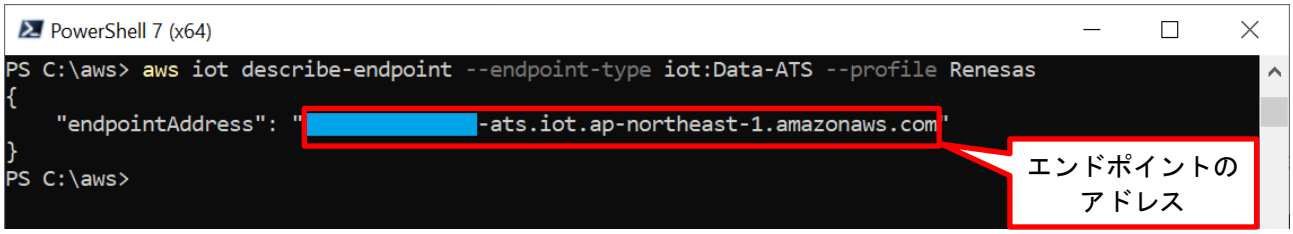

図 4-35 エンドポイントの確認

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。

(c) エンドポイントの入力

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 endpoint <ENDPOINT_NAME>
```

- *<ENDPOINT_NAME>*には「4.6.3(3)(b)」で確認したエンドポイント名を入力します。

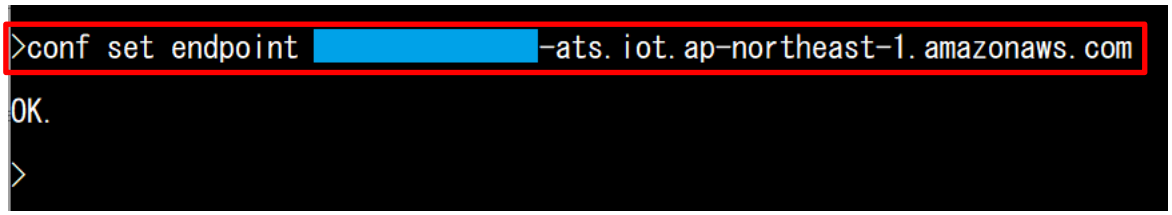

図 4-36 エンドポイントの入力

(d) プロビジョニングテンプレートの入力

プロビジョニングテンプレート名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 template <TEMPLATE_NAME>
```

- *<TEMPLATE_NAME>*には「4.3.5(2)」で作成したプロビジョニングテンプレート名を入力します。

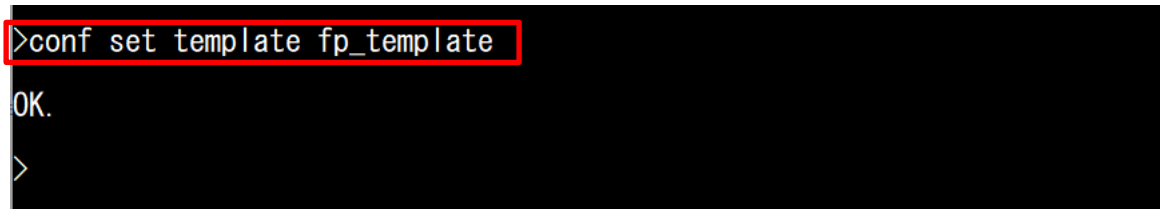

図 4-37 プロビジョニングテンプレートの入力

(e)　プロビジョニングクレーム証明書の入力

「4.3.3(1)」で生成したプロビジョニングクレーム証明書(fp-certificate.pem.crt)をターゲットボードに登録します。

- Tera Term にて「`conf set claimcert` 」と入力したのち、クレーム証明書(fp-certificate.pem.crt)を Tera Term にドラッグアンドドロップしてください
  （”`claimcert`”の後にはスペースを１文字入力します）
- 「Tera Term File Drag and Drop（Tera Term ファイルドラッグアンドドロップ）」ダイアログで「ファイル送信」の「Binary（バイナリ）」にチェックを入れてから[OK]ボタンをクリックします。
- 最後に Tera Term 上で[Enter]を押してください。

【注】クレーム証明書ファイル内の改行コードは LF に変更してから張り付けてください

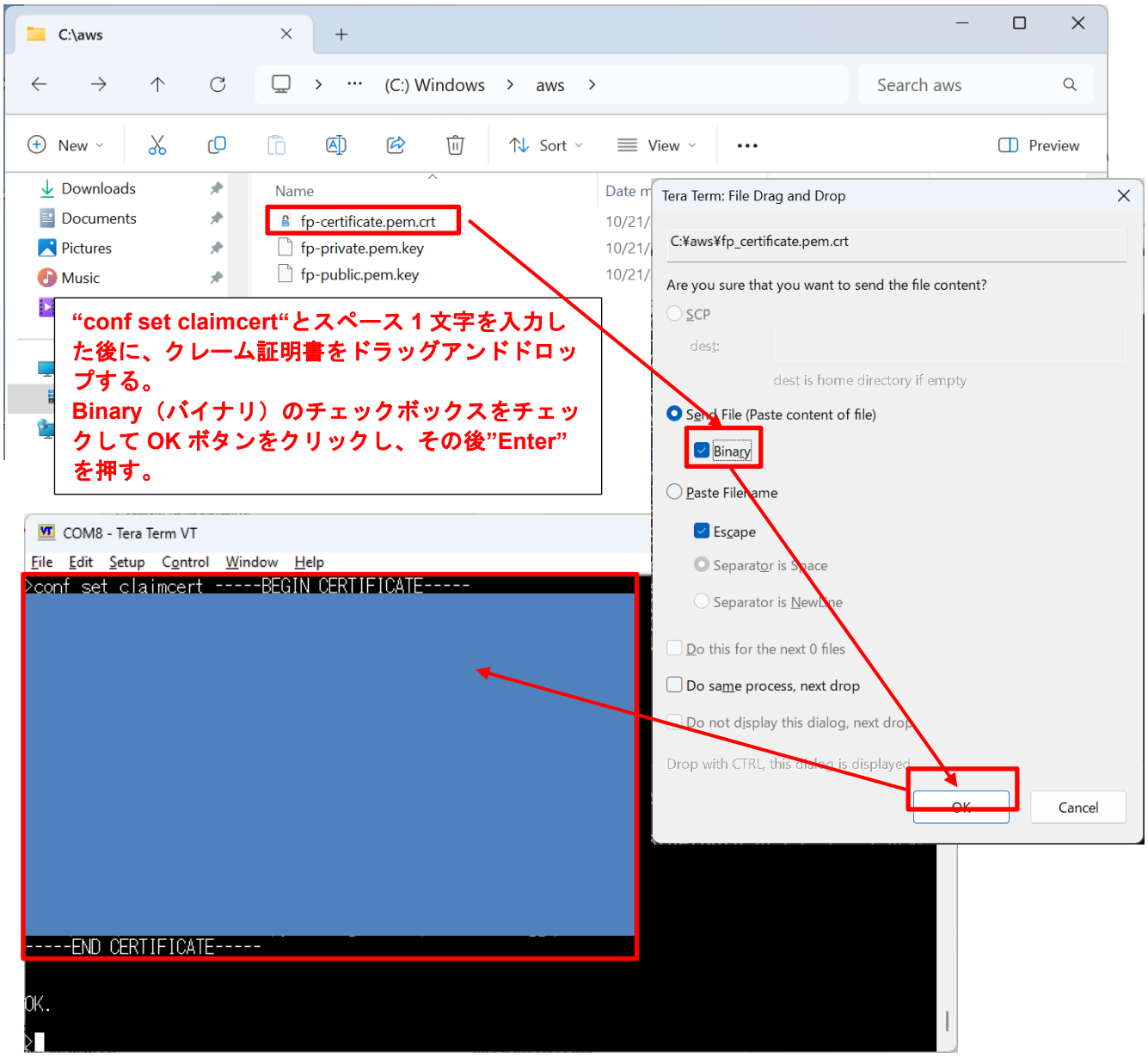

図 4-38 プロビジョニングクレーム証明書の入力

(f)　プロビジョニングクレーム秘密鍵の入力

「4.3.3(1)プロビジョニングクレーム証明書と鍵の生成とダウンロード」で生成したプロビジョニングクレーム秘密鍵(fp_private.pem.key)をターゲットボードに登録します。

- Tera Term にて「`conf set claimkey` 」と入力したのち、クレーム秘密鍵(fp_private.pem.key)を Tera Term にドラッグアンドドロップしてください
  （”`claimkey`”の後にはスペースを１文字入力します）
- 「Tera Term File Drag and Drop（Tera Term ファイルドラッグアンドドロップ）」ダイアログで「ファイル送信」の「Binary（バイナリ）」にチェックを入れてから[OK]ボタンをクリックします。
- 最後に Tera Term 上で[Enter]を押してください。

【注】クレーム秘密鍵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

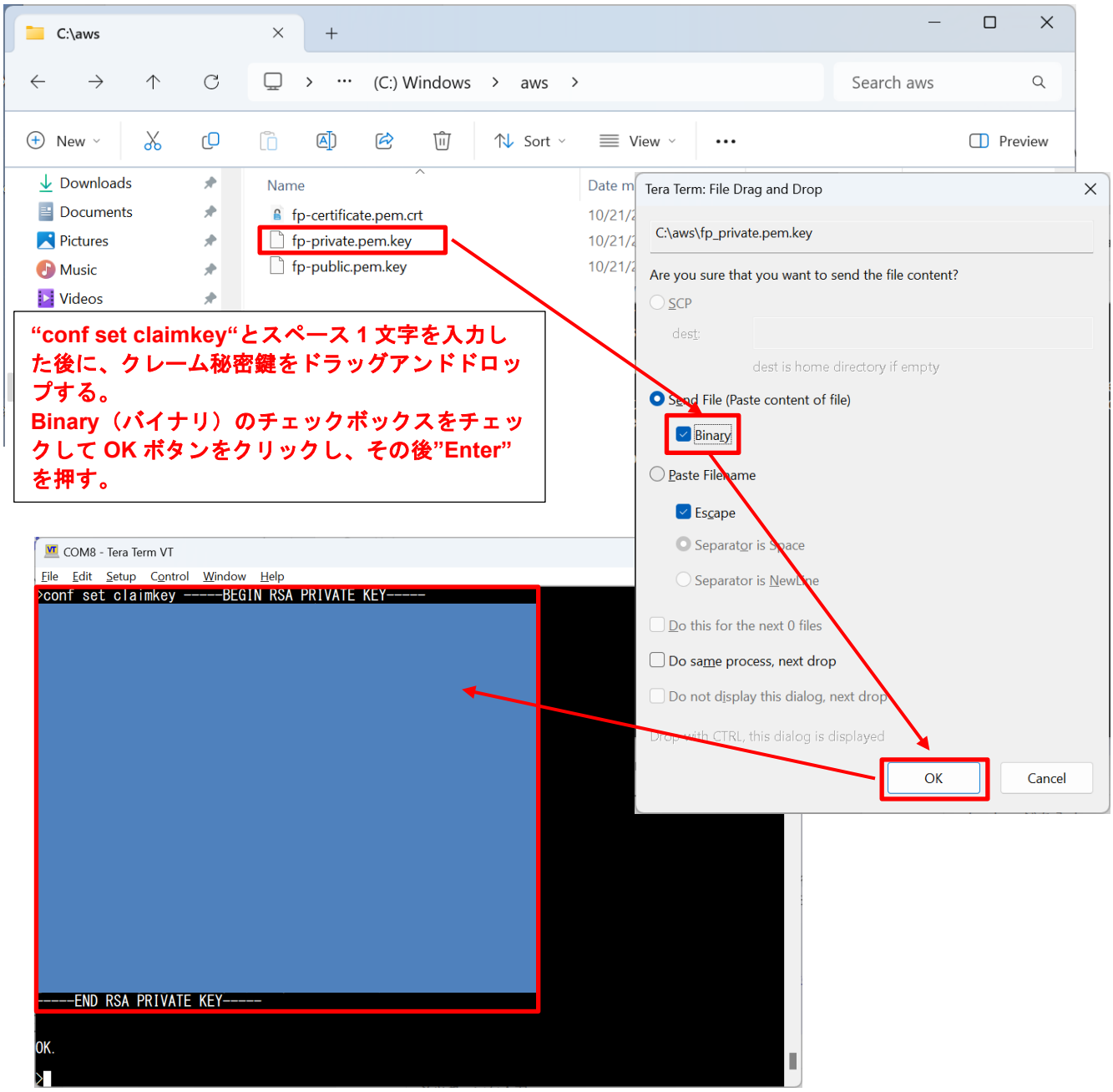

図 4-39 クレーム秘密鍵の入力

(g) 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
```

以下の様に「Configration save xxxx bytes～」と表示されたら書き込み完了です。

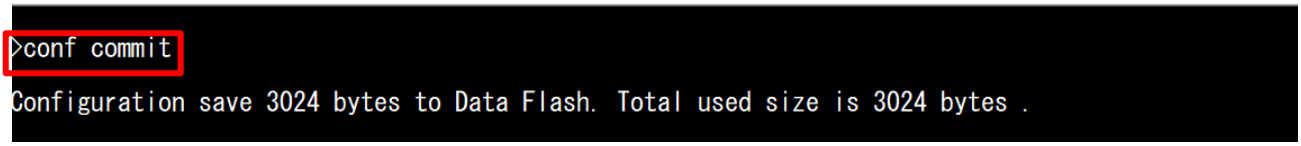

図 4-40 設定のデータフラッシュへの書き込み

(h) 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 上で何も入力しなければ 10 秒後にデモが起動します。以下の様に通信ログが表示されることを確認してください。

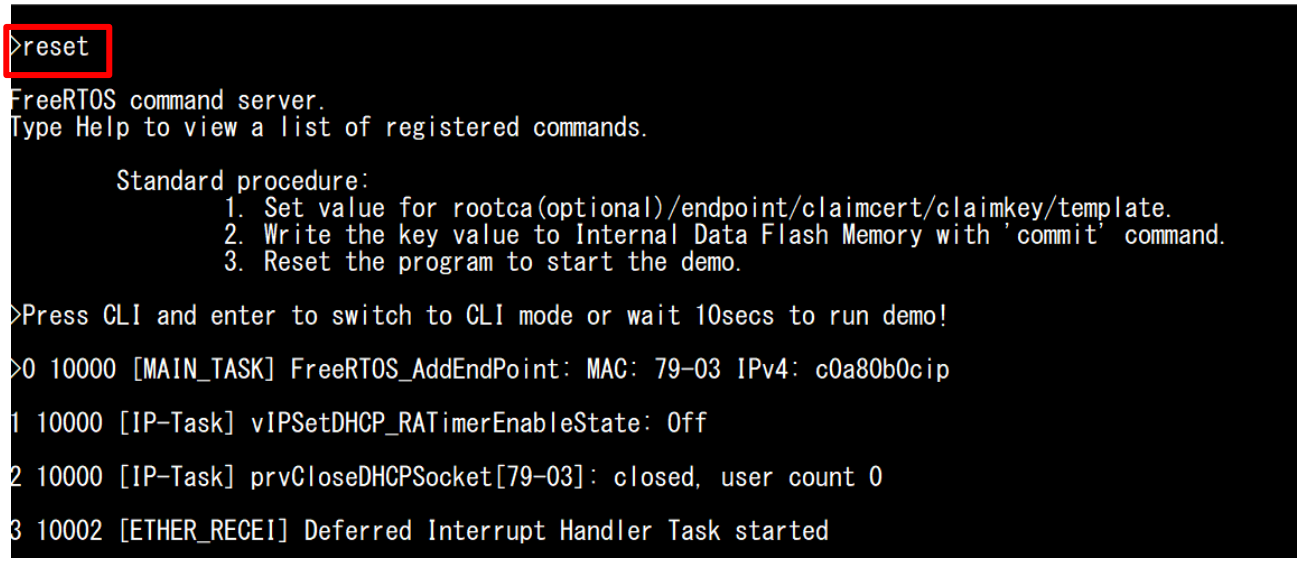

図 4-41 デモのリセット

4.6.4　実行結果の確認

「4.6.3(3)(h) Reset を実行」でフリートプロビジョニングデモをリセットすると Tera Term に実行ログが表示されます。

　デモではフリートプロビジョニングによるモノの登録とデバイス固有の証明書が付与され、その後 PubSub（MQTT 通信）デモが実行されます。

(1)　フリートプロビジョニングデモの実行確認

デモを実行すると最初にフリートプロビジョニングデモが実行されます。
ログに”Demo completed successfully.”および、”Fleet Provisioning Task Finished”が表示されていれば、フリートプロビジョニングデモは成功です。

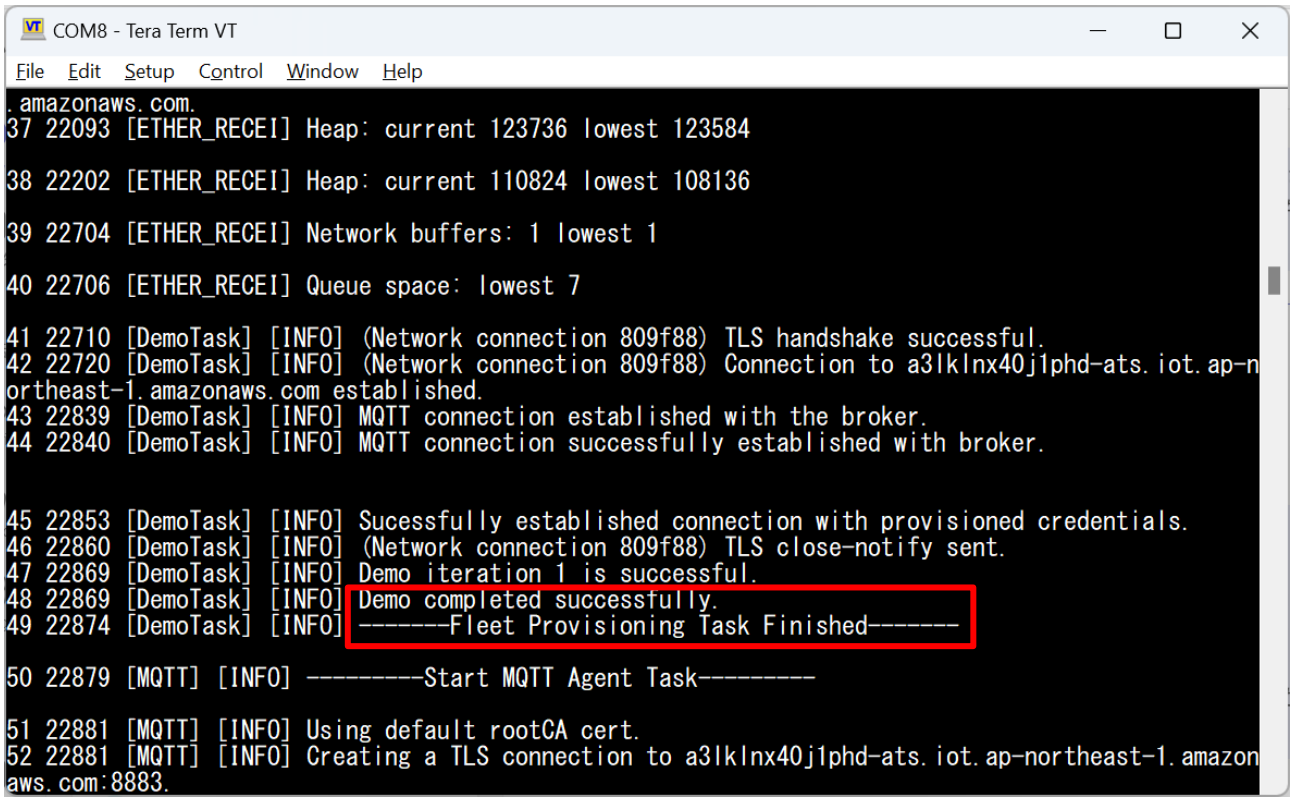

図 4-42 フリートプロビジョニングデモ成功時の実行ログ

(2) PubSub デモの実行

フリートプロビジョニングデモ実行後に AWS から取得したデバイス個別の証明書と秘密鍵を使用して PubSub デモが実行されます。

PubSub デモは MQTT 通信を使用してパブリッシュとサブスクライブを行うデモです。

以下のように Tera Term のログに”Successfully sent QoS 0 publish to topic:”および、”Received incoming publish message Task～”が表示されることが確認できます。

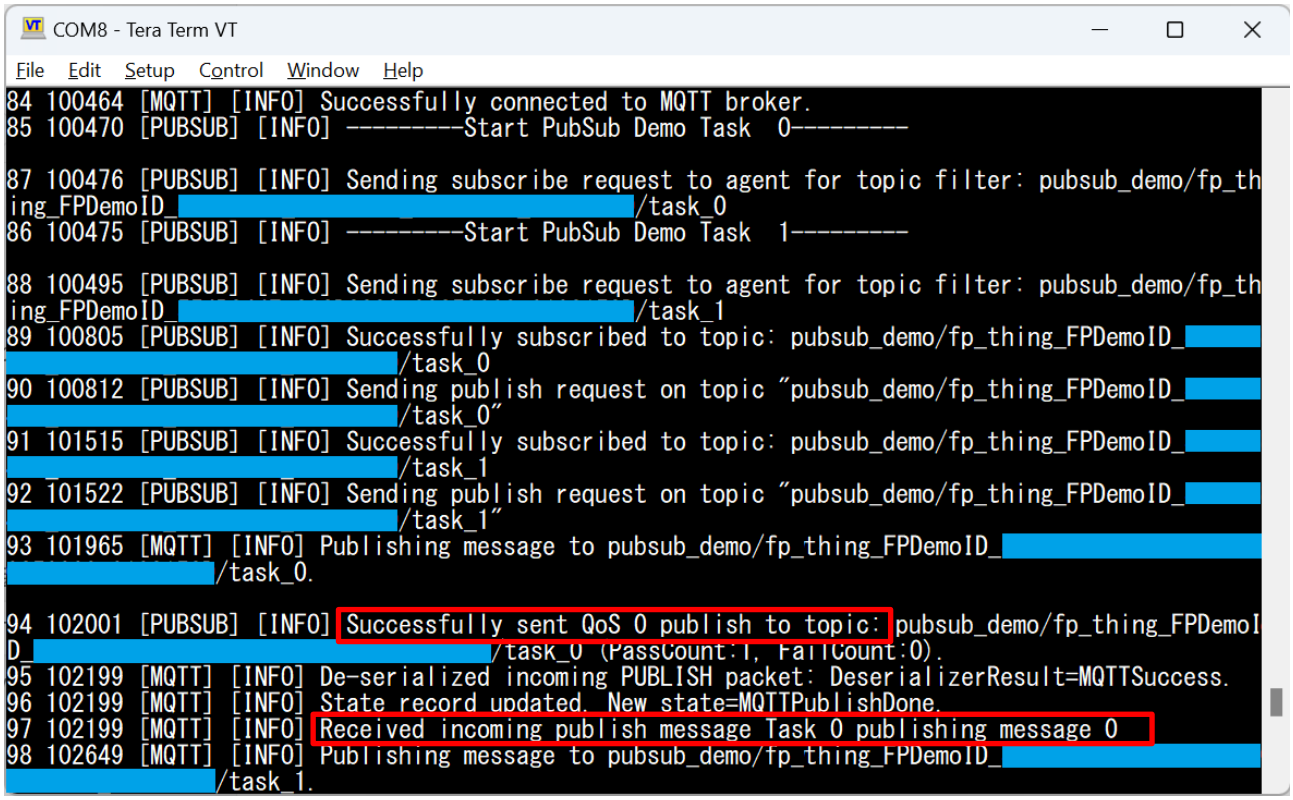

図 4-43 PubSub デモ成功時の実行ログ

PubSub デモでは PubSub Demo Task 0 と Task1 の 2 つのタスクで、message 0 から message 9 までのメッセージをそれぞれ 10 回送信します（合計で 20 回のメッセージを送信）

task 1 の message 9 を送信し、最後に”Unsubscribe successfully from task 1”と表示されると PubSub デモは終了です。

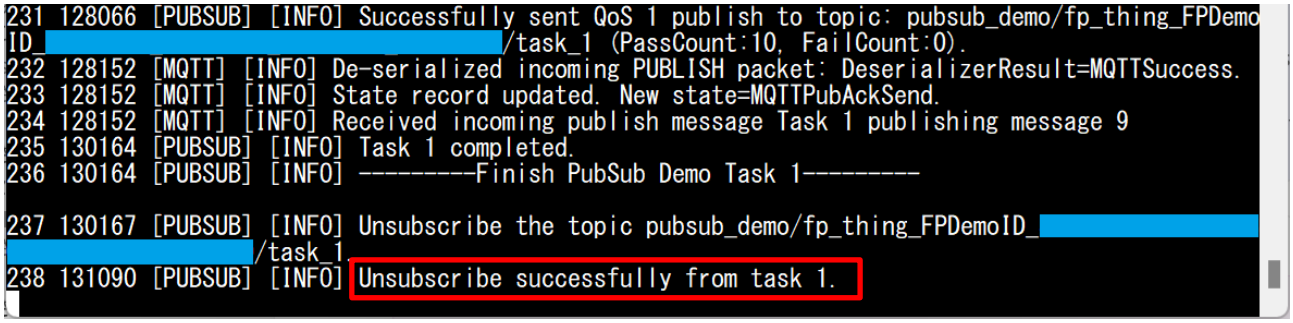

図 4-44PubSub デモの終了

また、「4.6.3(1) MQTT テストクライアントの設定」で設定した AWS マネジメントコンソールの AWS IoT Core から[MQTT test client（MQTT テストクライアント）]画面で、CK-RX65N v2 から AWS に送信された MQTT メッセージを確認できます。

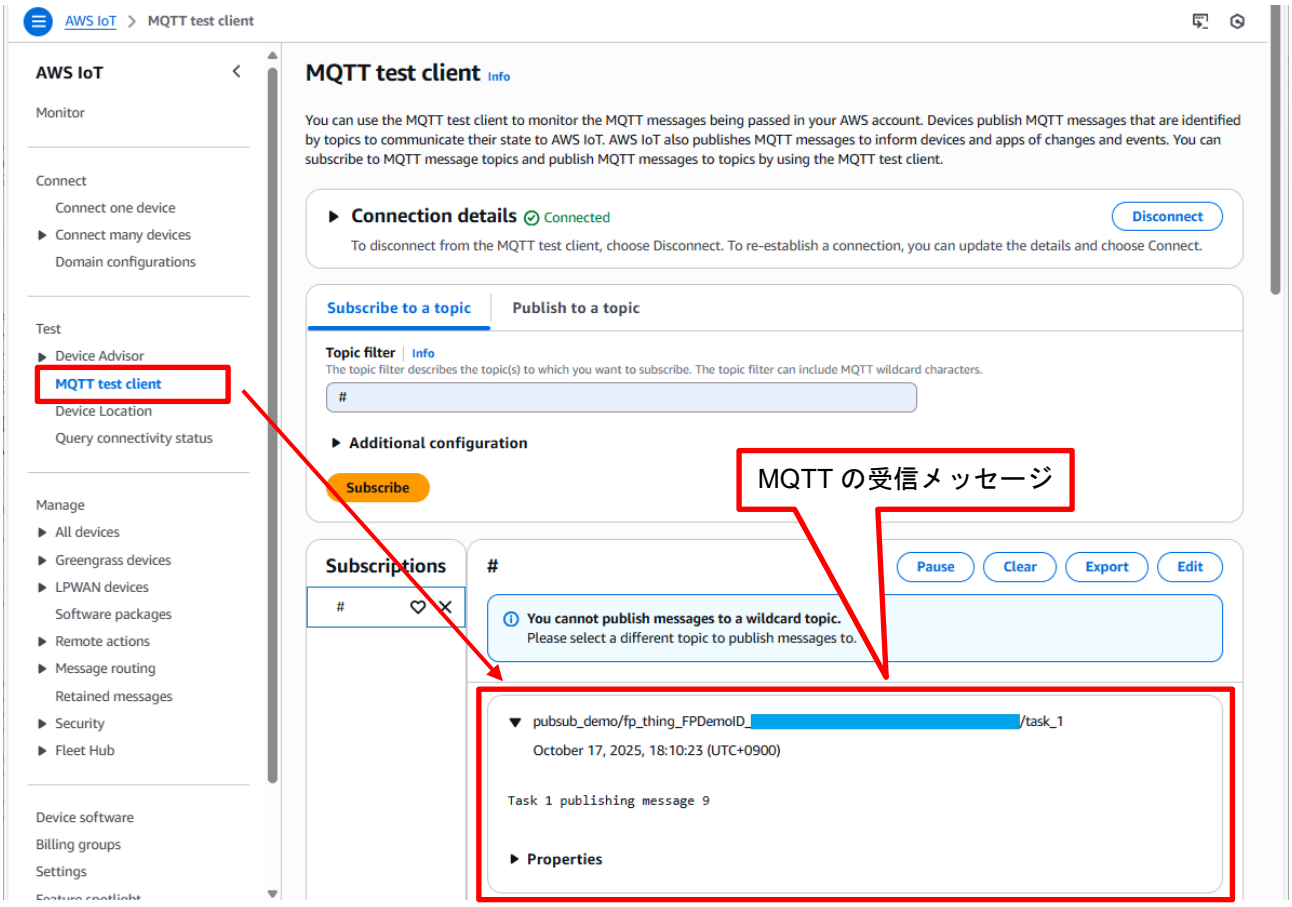

図 4-45 PubSub デモ成功時の MQTT テストクライアント

(3) モノの名前の確認

フリートプロビジョニングデモで登録されたモノの名前は、フリートプロビジョニングデモの実行中の Tera Term のログで、”Received AWS IoT Thing name”で確認できます。
このモノの名前はフリートプロビジョニングを使用して OTA を実施する際に必要となります。

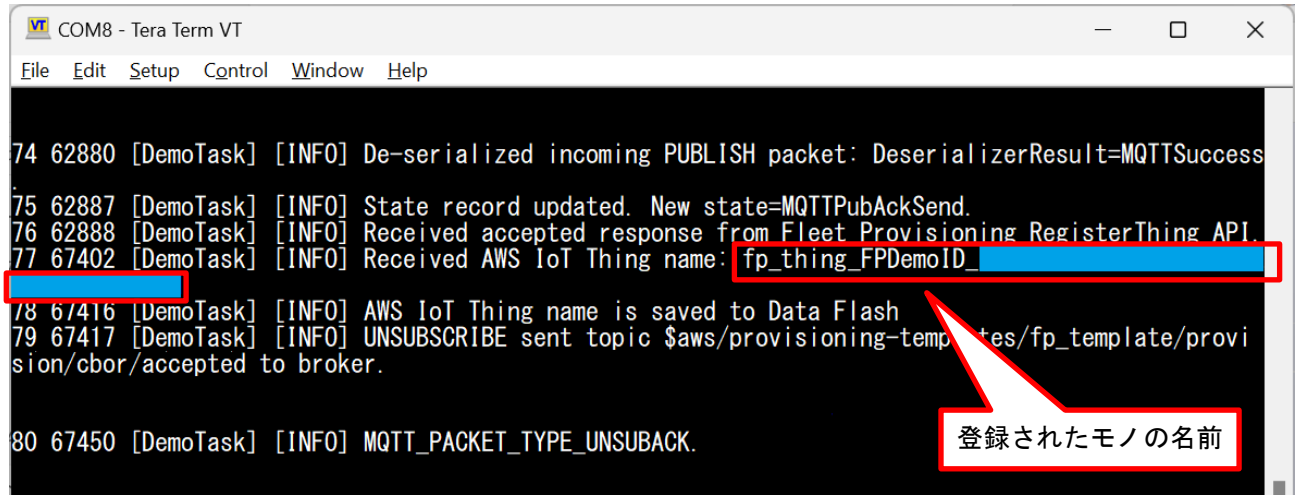

図 4-46 モノの名前の受信ログ

モノの名前は以下の法則で作成されます。

<THING_NAME_PREFIX>FPDemoID_xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxxx

① ② ③

- ①*<THING_NAME_PREFIX>*：「4.3.5(1) テンプレートの json ファイルの作成」でプロビジョニングテンプレートの json ファイルに入力したプレフィックスの文字列が使用されます
- ②固定の文字列：固定で”FPDemoID”の文字列が追加されます。
- ③MUC Unique ID：ご利用の CK-RX65N v2 ボードの MCU 固有の ID が付与されます。

また、登録されたモノの名前は AWS マネジメントコンソールの AWS IoT Core で確認できます。
[All devices（すべてのデバイス）]⇒[Things（モノ）]から Tera Term ログと一致するモノの名前を確認してクリックしてください。
OTA を実行する場合はここで表示されるモノの ARN を控えて下さい。

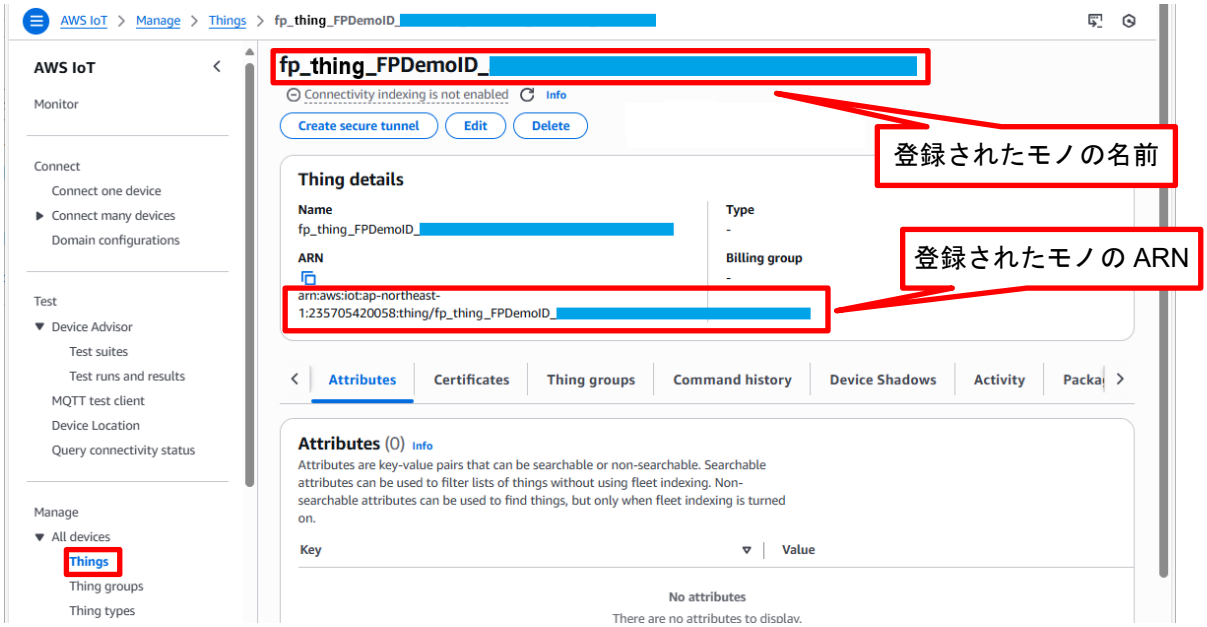

図 4-47 モノの ARN

(4) デバイス証明書の確認

登録されたモノを確認すると、フリートプロビジョニングで生成、割り当てられたデバイス個別の証明書がアタッチ、有効化されていることが確認できます。
“Received certificate with ID”で確認できます。

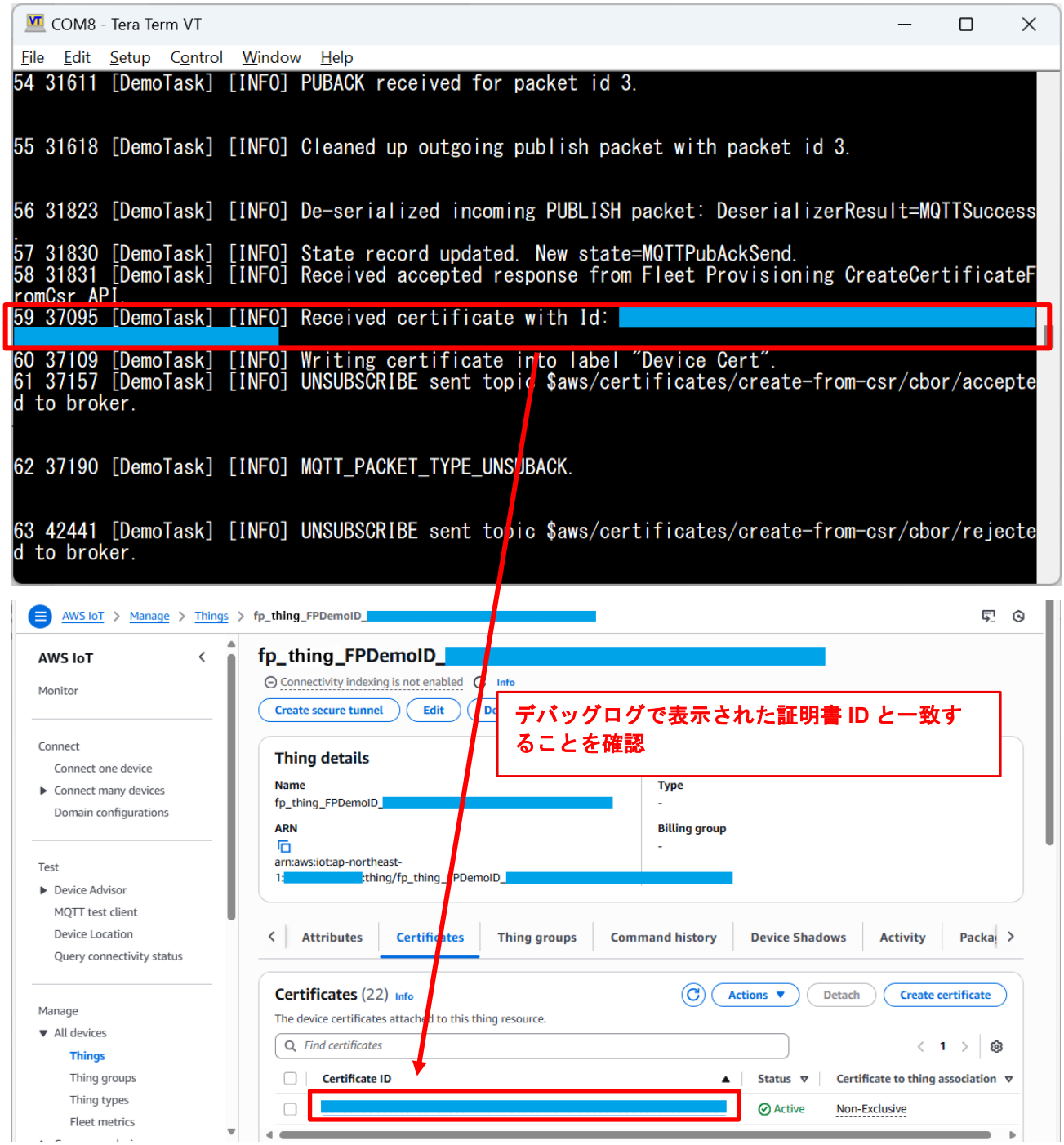

図 4-48 デバイス証明書の確認

4.6.5　フリートプロビジョニングによる OTA

フリートプロビジョニングのデモアプリケーションは OTA アップデートアプリケーションと同時に実行することができます。

基本的な OTA の手順に関しては、アプリケーションノート「RX ファミリ Amazon Web Services を利用した FreeRTOS OTA の実現方法(202406-LTS 版)」（R01AN7662）を参照してください。

フリートプロビジョニングアプリケーションと同時に OTA アップデートを実行する際は、上記のアプリケーションノートで説明されている手順とは異なる箇所があります。以下の手順を注意してください。

(a) R01AN7662 の「2.事前準備」の手順でツールの環境を整えてください。
「2.1 Tera Term のインストール」、「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」の手順は本アプリケーションノートですでに完了しているので必須ではありません。

(b) AWS CLI が実行可能なことを確認してください。
本アプリケーションノートの「4.2 AWS の準備」で説明されている手順がすでに完了している場合は R01AN7662 の「3.AWS の設定」の手順は省略できます。

(c) R01AN7662 の「3.3.1 ポリシーの設定」を実行して OTA のポリシーを作成してください。
また、本アプリケーションノートの「4.3.2 フリートプロビジョニングで作成されたモノのポリシー作成」で作成したポリシーを再利用することも可能です。

(d) R01AN7662 で説明されている「3.3.2 Amazon S3 バケットの作成」と「3.3.3 IAM ユーザーに OTA の実行権限を割り当てる」の手順を実行し OTA 用の AWS 設定を設定してください。

(e) R01AN7662 の「3.3.4 デバイス（モノ）を AWS IoT に登録」の手順は省略することができます。
ただし、本アプリケーションノートの「4.3 フリートプロビジョニング AWS の設定」の手順が未実施の場合は実行してください。

(f) R01AN7662 の「4.1 鍵ペアと証明書の生成」の手順を実行し、OTA 用の鍵ペアと証明書を作成します。

(g) 本アプリケーションノートの「4.4 サンプルプロジェクトの作成」の手順を実行してフリートプロビジョニングのデモプロジェクトをインポートしてください。
また、「4.4(6)インポートするプロジェクトの選択」でプロジェクトを選択する際にブートローダプロジェクト（boot_loader_ck_rx65n_v2）もチェックをしてインポートしてください。
R01AN7662 の「4.2.1 サンプルプロジェクトの生成」の手順は省略できます。

(h) R01AN7662 の「4.2.2 プロジェクト設定」から「4.2.4 初期ファームウェアの作成」の手順を実行し、OTA 用フリートプロビジョニングの初期ファームウェアを作成してください。

【注】「4.2.2(2)」の手順を実行する様は、フリートプロビジョニングデモ設定も以下の様に” 1 ”に設定して下さい。

```
/* Please select a provisioning method
 * (0) : Pre-provisioning
 * (1) : Fleet provisioning
 */
#define ENABLE_FLEET_PROVISIONING_DEMO      (1)

/* Please select whether to enable or disable the OTA demo
 * (0) : OTA demo is disabled
 * (1) : OTA over MQTT demo is enabled
 */
#define ENABLE_OTA_UPDATE_DEMO              (1)
```

(i) デモアプリケーションで AWS クレデンシャル情報を登録するには、R01AN7662 の「4.2.5 AWS IoT 情報の登録」を実行してください。
R01AN7662 の「4.2.5(6) デバイス証明書の登録」以降の CLI へのクレデンシャル情報の登録手順は一部異なるため、以下の操作に置き換えて実行してください。

① エンドポイントの入力（本アプリケーションノート「4.6.3(3)(c)」参照）
② プロビジョニングテンプレートの入力（本アプリケーションノート「4.6.3(3)(d)」参照）
③ プロビジョニングクレーム証明書の入力（本アプリケーションノート「4.6.3(3)(e)」参照）
④ プロビジョニングクレーム秘密鍵の入力（本アプリケーションノート「4.6.3(3)(f)」参照）
⑤ コード署名検証用の証明書の入力（R01AN7662「4.2.5(9)」参照
⑥ 設定値のデータフラッシュへの書き込み（R01AN7662「4.2.5(10)」参照
⑦ リセットの実行（R01AN7662「4.2.5(11)」参照

【注】R01AN7662 の「4.2.5(6)デバイス証明書の登録」と「4.2.5(7)秘密鍵の登録」の操作は必須ではありません。

(j) R01AN7662 の「5. ファームウェアの更新」の手順を実行し、更新ファームウェアの作成し OTA を実行します。

【注】「5.2.3 OTA ジョブの実行」ではフリートプロビジョニングで割り当てられたモノの名前を json ファイルの”ターゲット ARN（OTA 実行対象のモノ）”に入力してください。
これは本アプリケーションノートの「4.6.4(3) モノの名前の確認」で確認できます。

5. まとめ

先に述べた通りプロビジョニング方式には複数の方式が存在し、かつセキュリティ強度も様々です。

実際に適用するターゲットの市場での用途やシステムの規模 (デバイスの個数)、求められるセキュリティレベルに応じて、適切なプロビジョニング方式を選択し、導入することが今や不可欠です。

しかし、プロビジョニング機能を実現するためにセキュアな工場を自社で持ち、管理 / 運用することは簡単ではありません。そのため、デバイスプロビジョニングの工程において、フリートプロビジョニング方式が着目されることになり、市場の要求として急増していると考えられます。

今回、本書で解説したプロビジョニング方式はほんの一例であり、全てのユーザーの要求に必ずしも合致するものではありませんが、導入することによるメリット / デメリットに関して理解を深めていただけたのではないかと思います。

本書がみなさまの快適な生産ライン構築の実現に少しでもお役に立てば幸いです。

6. ウェブサイトおよびサポート

AWS re:Post : https://repost.aws

Renesas FreeRTOS GitHub : https://github.com/renesas/iot-reference-rx

7. 付録

7.1 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。

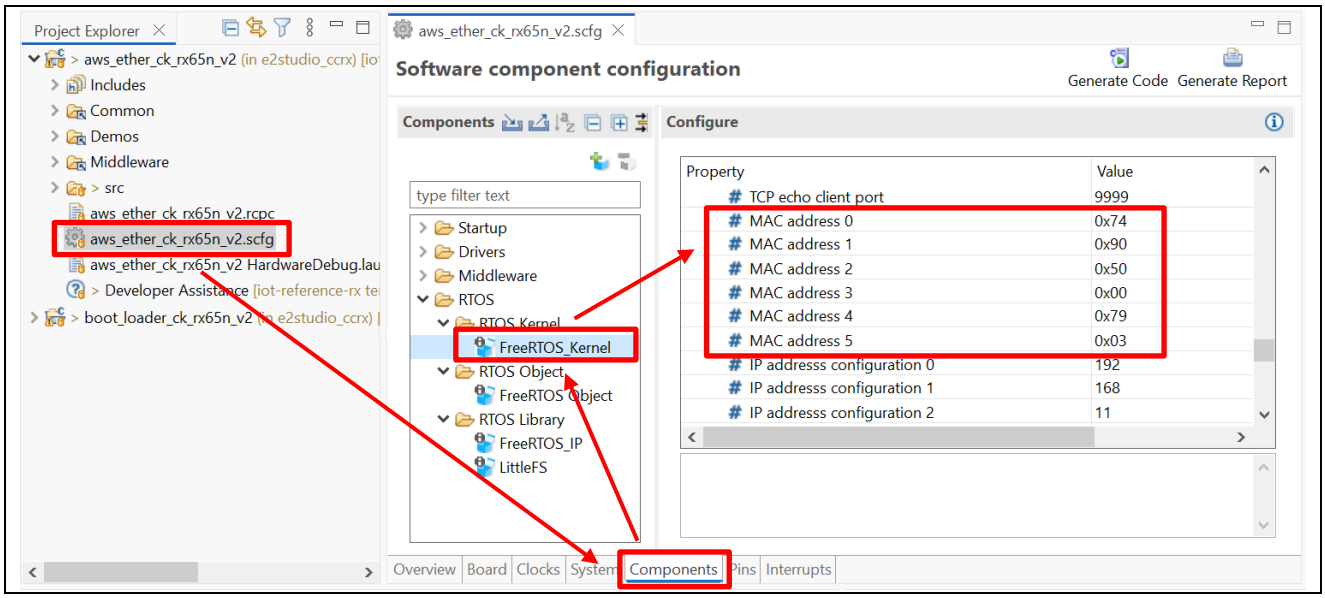

図 7-1 MAC アドレスの設定

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。

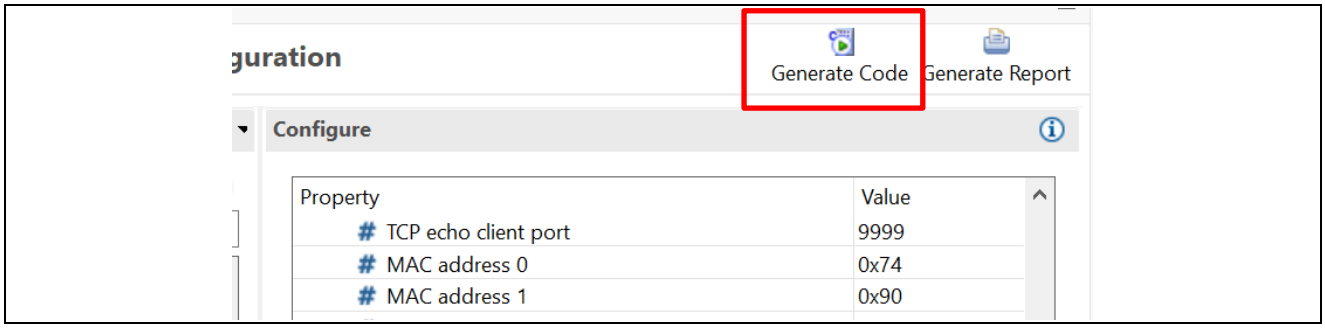

図 7-2 コードの生成

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2026.3.10	ー	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。