RX ファミリ

IoT デバイスのためのプロビジョニング手法

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。

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。ファームウェアアップデートについてはアプリケーションノート（R01AN5548）を合わせてご参照ください。

【注】本アプリケーションノートでは CK-RX65N v1 ボードおよび RYZ014A PMOD モジュール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。

ドライバは以下の組み合わせにてご利用可能です。
- RYZ014A Cellular モジュール制御モジュール：Sequans GM01Q が互換のある代替モジュールとなります。

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	e^2 studio 2024-04
ボード	CK-RX65N
ツールチェーン	CC-RX Compiler v3.05.00
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。

- Renesas ルネサス MCU におけるファームウェアアップデートの設計方針(R01AN5548)
- RX ファミリ RX65N における Amazon Web Services を利用した
  FreeRTOS OTA の実現方法(R01AN5549)
- Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
- RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology(R01AN1723)

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。（日本語のみ）

- 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

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。

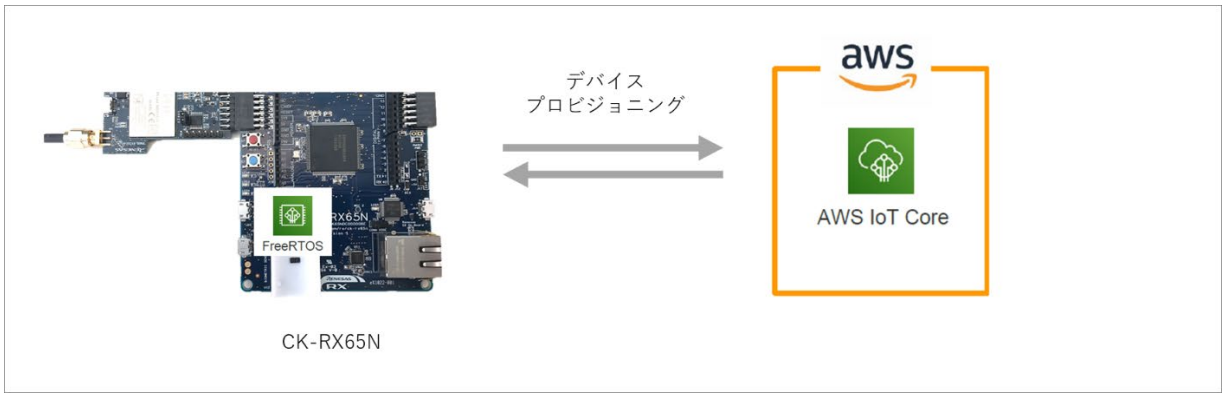

図 2-1 デバイスプロビジョニング

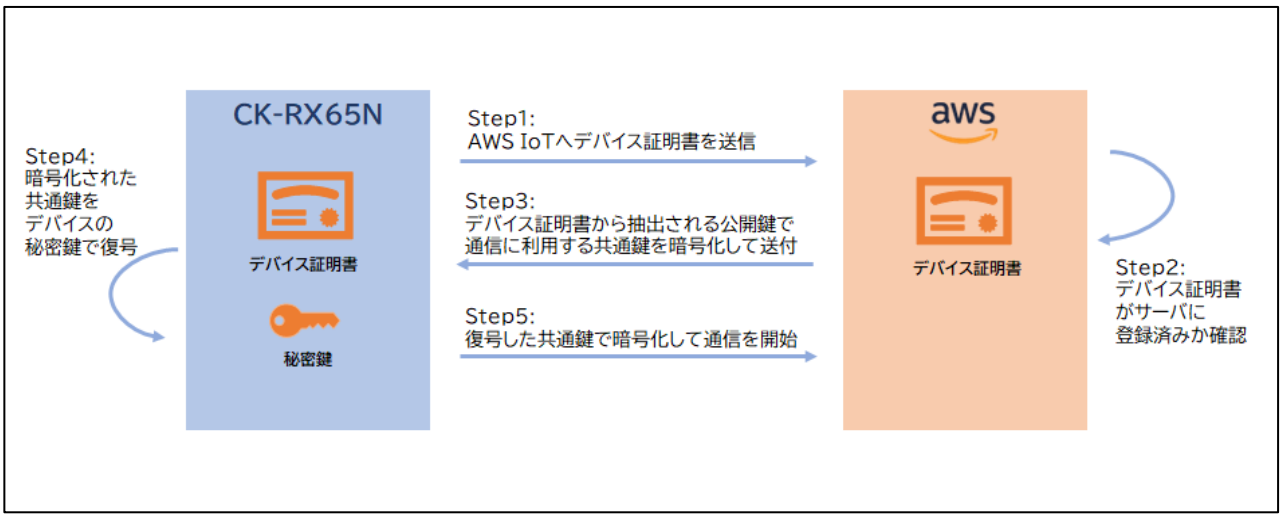

図 2-2 IoT デバイスのプロビジョニングの概要

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）

量産検討の前段階で、FreeRTOS の動作を確認したい場合は、AWS 上で秘密鍵証明書を発行してソースコード変換し、そのソースコードを 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。

※　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、将来的な実現を検討しています。

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 (モバイル / Web アプリユーザ等) 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
- 個別の証明書発行やデバイス登録する構成を組まなくて良い

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、後続の呼び出しで証明書をアクティブに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「 本番 」証明書に対して非推奨 / ローテーションされます。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。

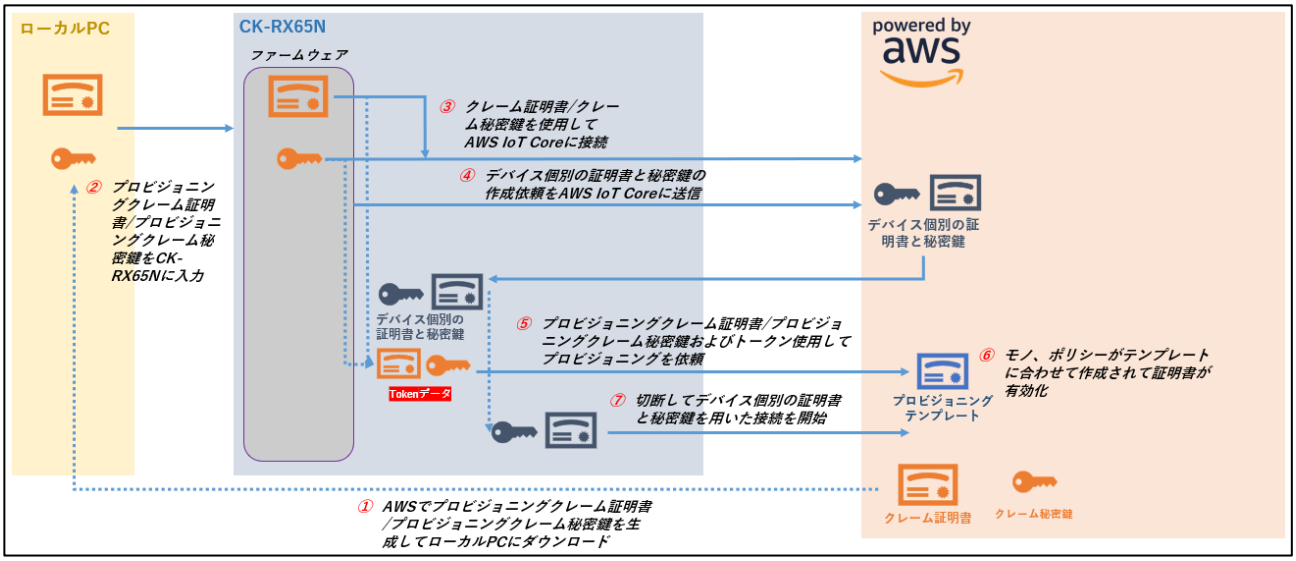

図 2-3 クレームによる CreateKeysAndCertificate 方式でのプロビジョニングのワークフロー

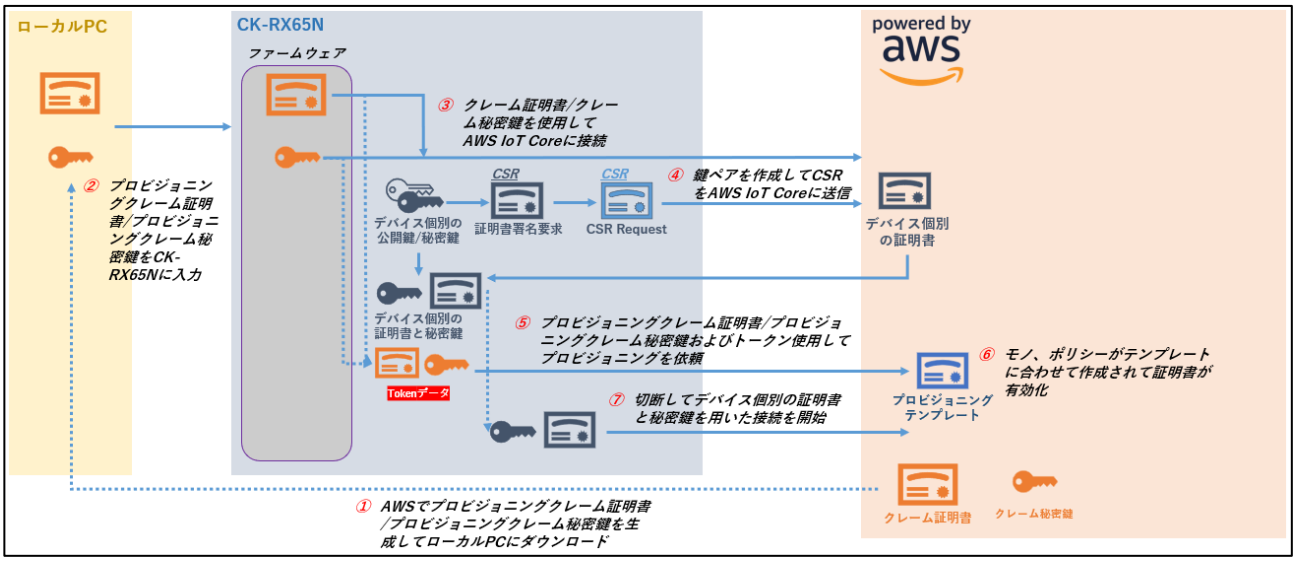

図 2-4 クレームによる CreateCertificateFromCsr 方式でのプロビジョニングのワークフロー

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

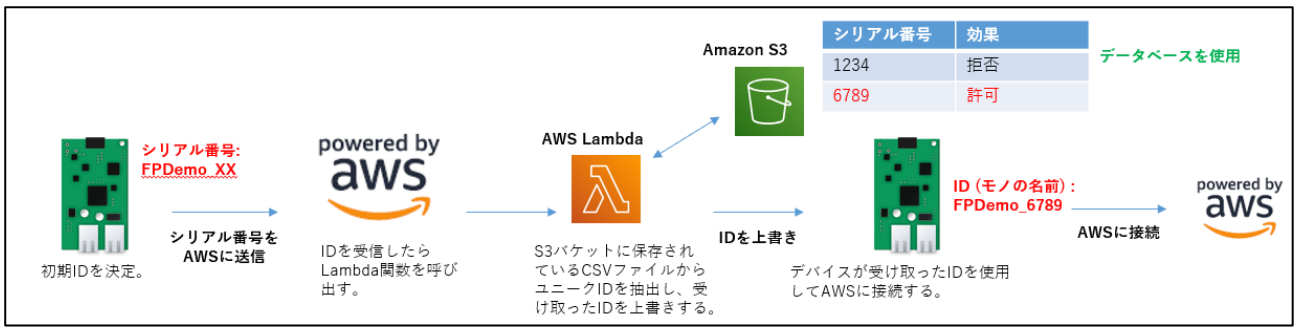

図 2-6 Amazon S3 またはデータベースを使用したモノの名前の決定方法

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 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	Renesas Electronics Corporation	RX65N MCU 搭載のクラウドキット
PC	Windows10（推奨）	—	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	2024-04	—
ツールチェーン	CC-RX	v3.05.00	—
通信ソフト	Tera Term	Ver 4.106	ログ表示用
FreeRTOS	v202210.01-LTS-rx	V1.2.1	
エミュレータ	CK-RX65N 搭載の E2OB (E2 Lite On Board)	—	—

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にアクセスする

Tera Term ダウンロードサイト(GitHub)

(2)　Tera Term のインストーラーをダウンロードする

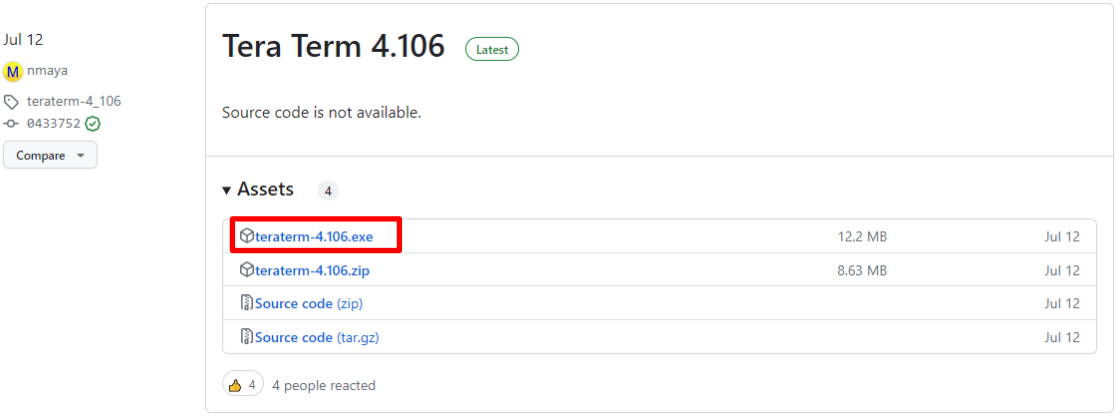

図 3-1 ダウンロードする Tera Tarm

(3)　インストーラーを実行し、案内に沿って Tera Term をインストールする

(4)　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する

(5)　下表に示す Tera Term 設定をおこなう

表 3-3 Tera Term 設定

用語	意味
ボー・レート	115200
データ長	8bit
パリティ	none
ストップビット	1bit
フロー制御	none

3.4 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。

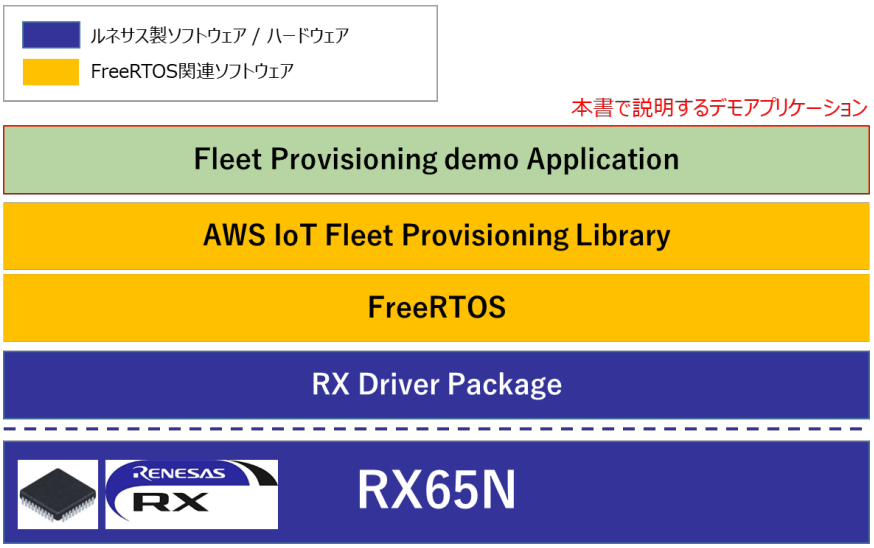

図 3-12 本アプリケーションノートに含まれるデモプロジェクトの構成イメージ

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リポジトリより取得することができます。

デモアプリケーション ： iot-reference-rx : FreeRTOS reference repository

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 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 を使用することができます。

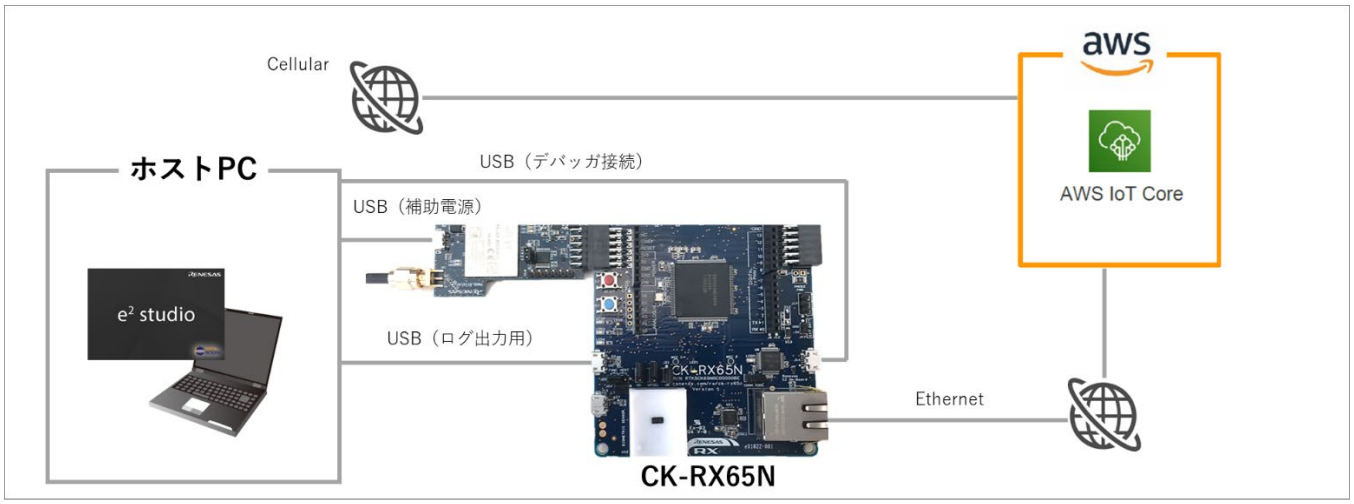

図 4-1 デモ実行環境

4.2　AWS の準備

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には AWS のアカウントが必要です。
アカウントをお持ちでない場合は、まずアカウントを作成してからコンソールにログインしてください。
なお、本アプリケーションノートで参照する AWS コンソールは 2023 年 9 月時点のものです。

AWS トップページ（ https://aws.amazon.com/jp/ ）

①「コンソールにサインイン」-> 「今すぐ無料サインアップ」からアカウントを作成。

②「コンソールにサインイン」からサインイン。

③ [サービス] > [IoT] > [IoT Core]を選択し、AWS IoT のコンソールを開きます。

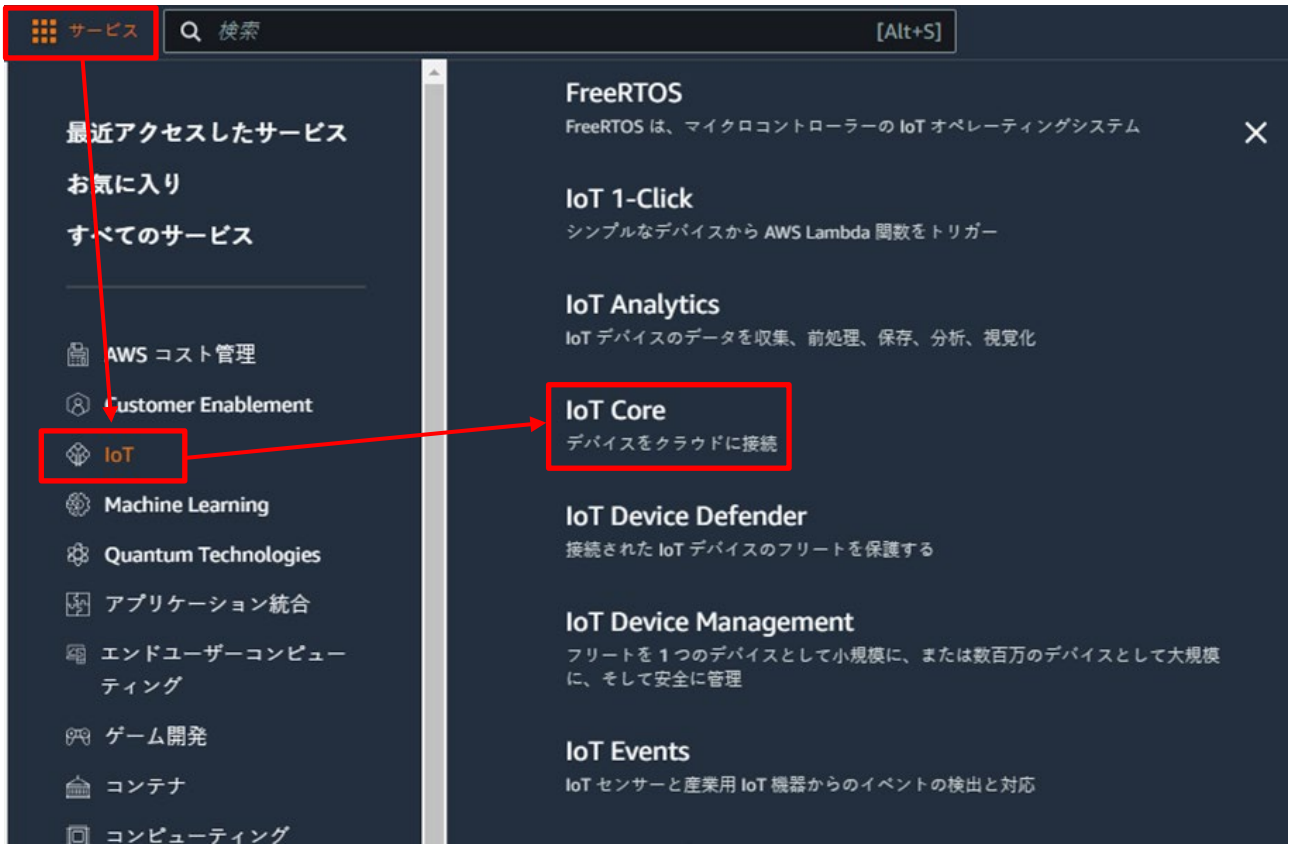

図 4-2 AWS コンソール

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。

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

4.3.1　ポリシーの設定

下記の手順で AWS IoT Core ポリシーを作成します。

最初にフリートプロビジョニング実行時に使用するポリシーを作成します。

[セキュリティ] > [ポリシー] を選択し、「ポリシーを作成」ボタンをクリックします。

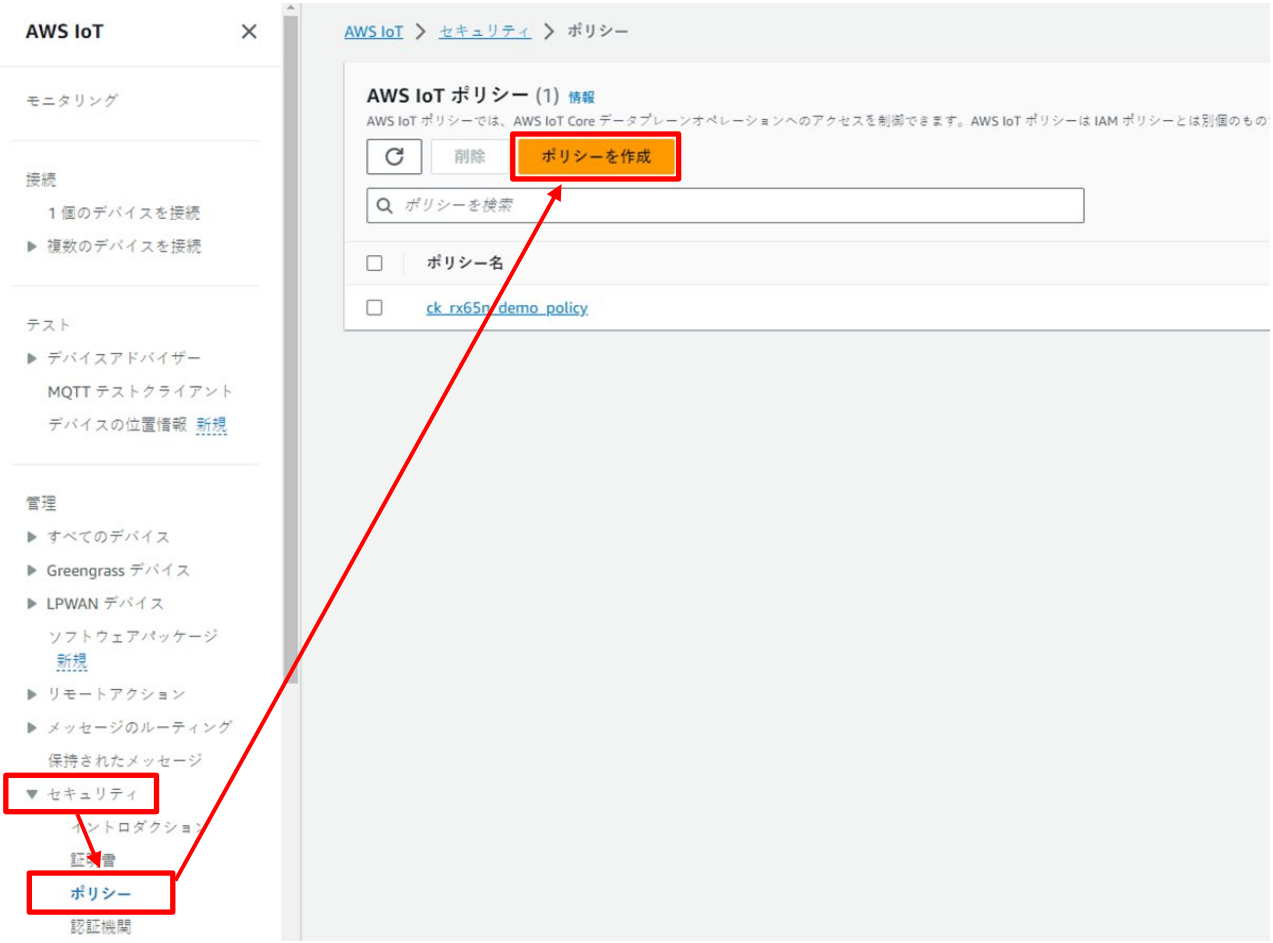

図 4-3 AWS IoT ポリシーの作成(1)

ポリシー名の欄に、任意のポリシー名を入力します。

[JSON] ボタンでポリシードキュメントの入力欄を表示し、図 4-5 に示すポリシードキュメントをコピー&ペーストします。図 4-5 のポリシードキュメントをコピー&ペーストする際、以下の変更を行ってください。

・“ap-northeast-1” を使用するリージョンに合わせて変更

・<account id> をご自身のアカウント ID に変更（アカウント ID は、アカウント ID は右上のアカウント名をクリックすることで表示される@以降の 12 桁の数字からハイフンを除いたもの）

[作成] ボタンを押してポリシーの作成は完了です。

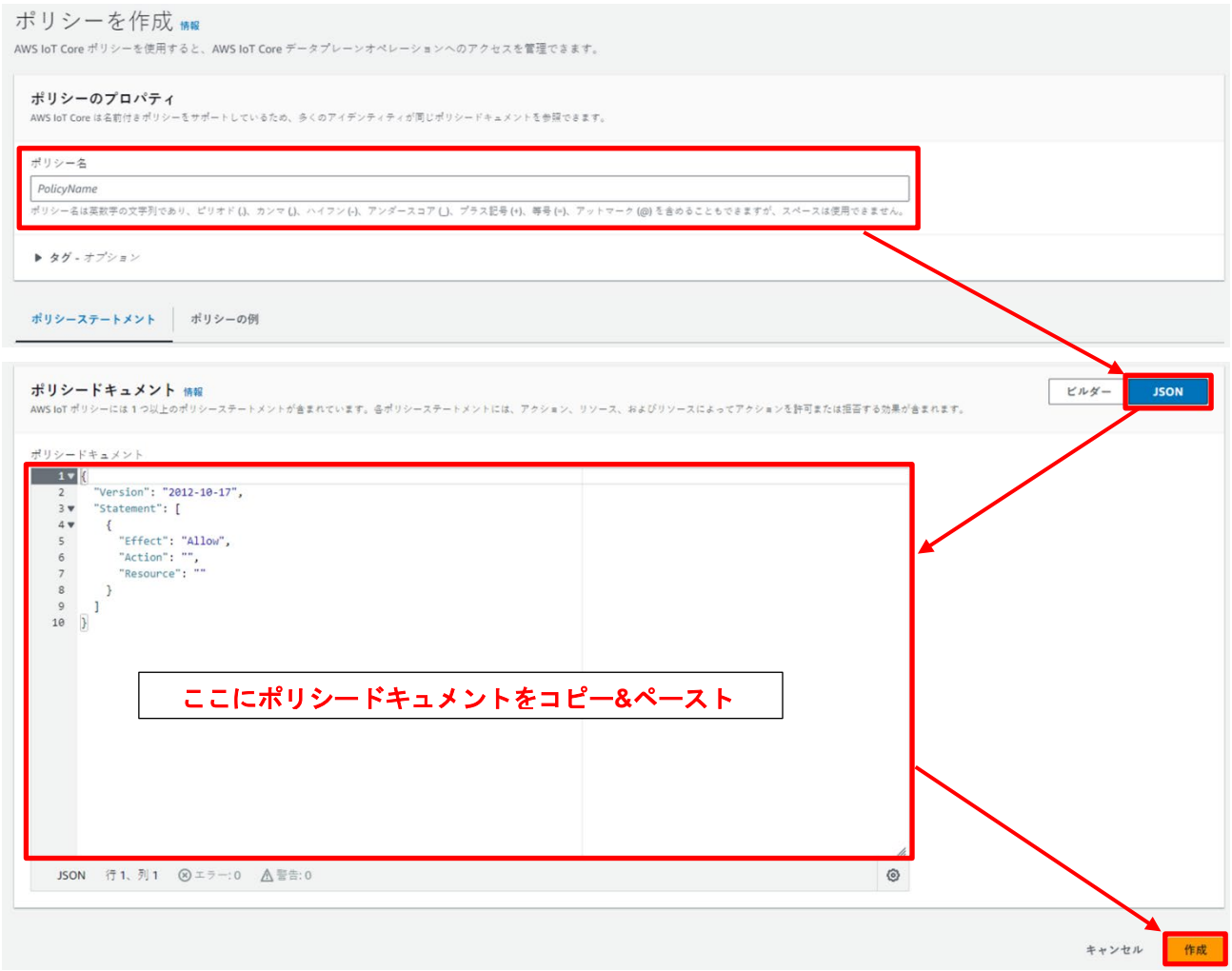

図 4-4 AWS IoT ポリシーの作成(2)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive",
        "iot:RetainPublish"
      ],
      "Resource": [
        "arn:aws:iot:ap-northeast-1:<account id>:topic/$aws/certificates/create-from-csr/*",
        "arn:aws:iot:ap-northeast-1:<account id>:topic/$aws/provisioning-templates/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:ap-northeast-1:<account id>:topicfilter/$aws/certificates/create-from-csr/*",
        "arn:aws:iot:ap-northeast-1:<account id>:topicfilter/$aws/provisioning-templates/*"
      ]
    }
  ]
}
```

図 4-5 ポリシードキュメント

次にフリー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。

[セキュリティ] > [ポリシー] を選択し、「ポリシーを作成」ボタンをクリックします。

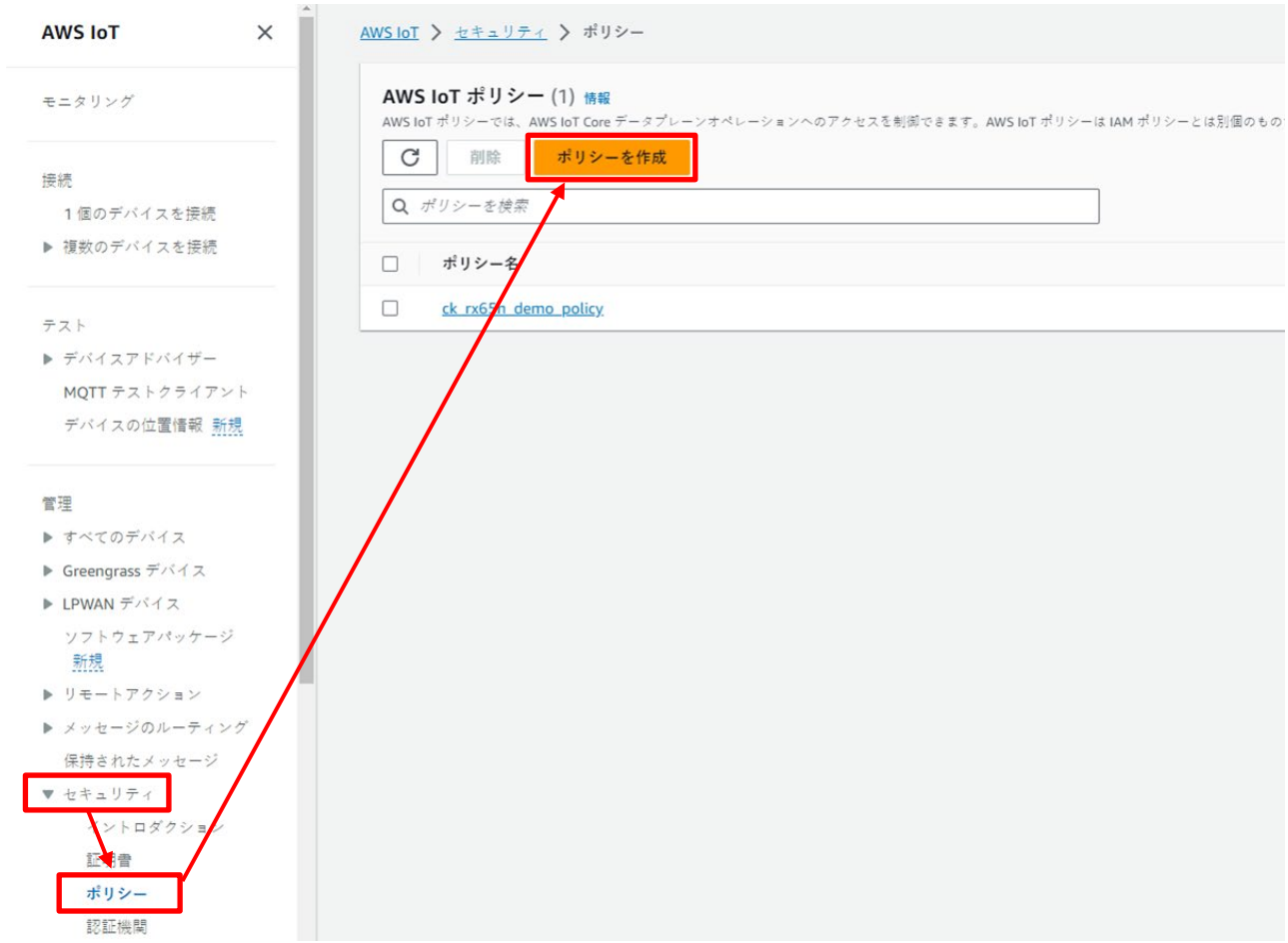

図 4-6 AWS IoT ポリシー作成(1)

ポリシー名の欄に、任意のポリシー名を入力します。

ポリシードキュメントにポリシーアクションとして、「iot:Connect」、「iot:Publish」、「iot:Subscribe」、「iot:Receive」を「許可」に設定します。またポリシーリソースはワイルドカード「*」を指定し、すべてのリソースを許可にします。デフォルトで設定できるステートメントは１つです。「新しいステートメントを追加」をクリックすることで、複数のステートメントを設定できます。

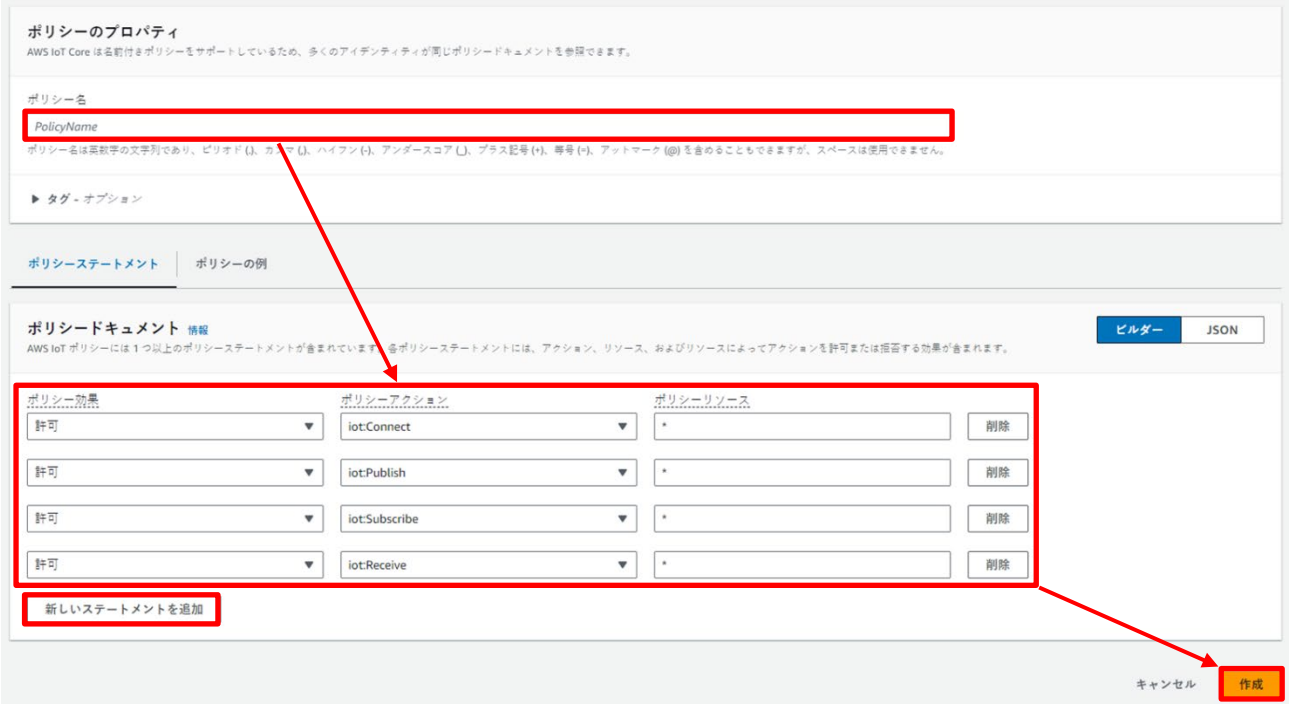

図 4-7 AWS IoT ポリシー作成(2)

4.3.2　クレーム証明書、クレーム鍵の生成

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を生成します。

[セキュリティ] > [証明書] を選択し、[証明書を追加] > [証明書を作成] をクリックします。

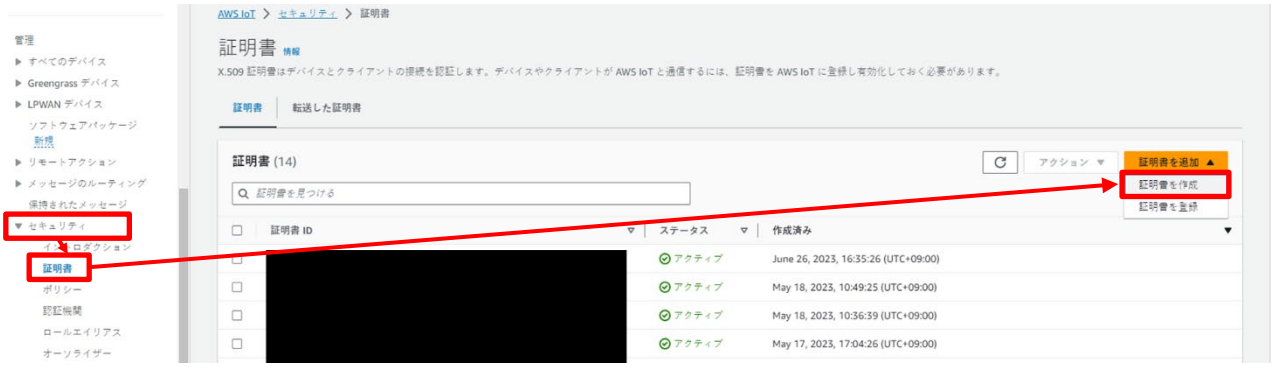

図 4-8 証明書の作成

「新しい証明書の自動生成（推奨）」を選択し、[作成]をクリックします。

図 4-9 証明書の自動生成

生成された、①証明書と ②③鍵のペアをダウンロードして、[続行]をクリックします。

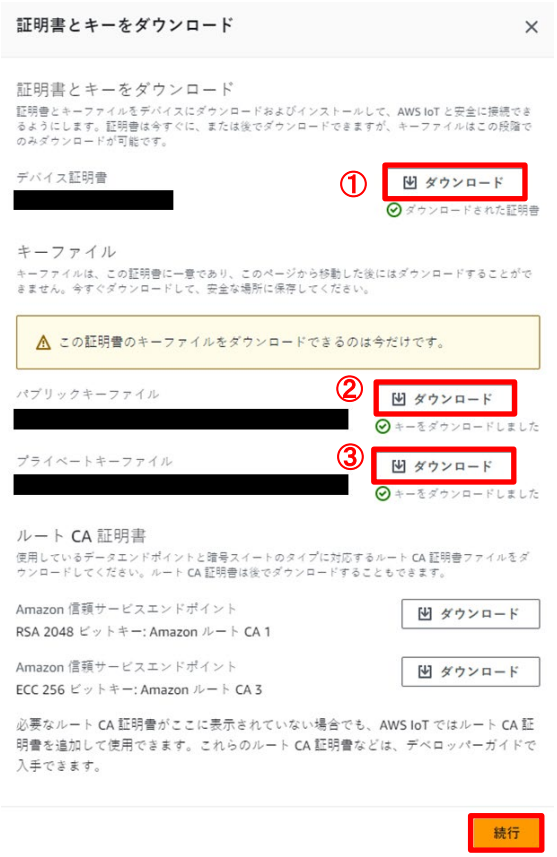

図 4-10 証明書と鍵のダウンロード

AWS コンソールの、[セキュリティ] > [証明書] を選択し、生成した証明書 ID を選択します。

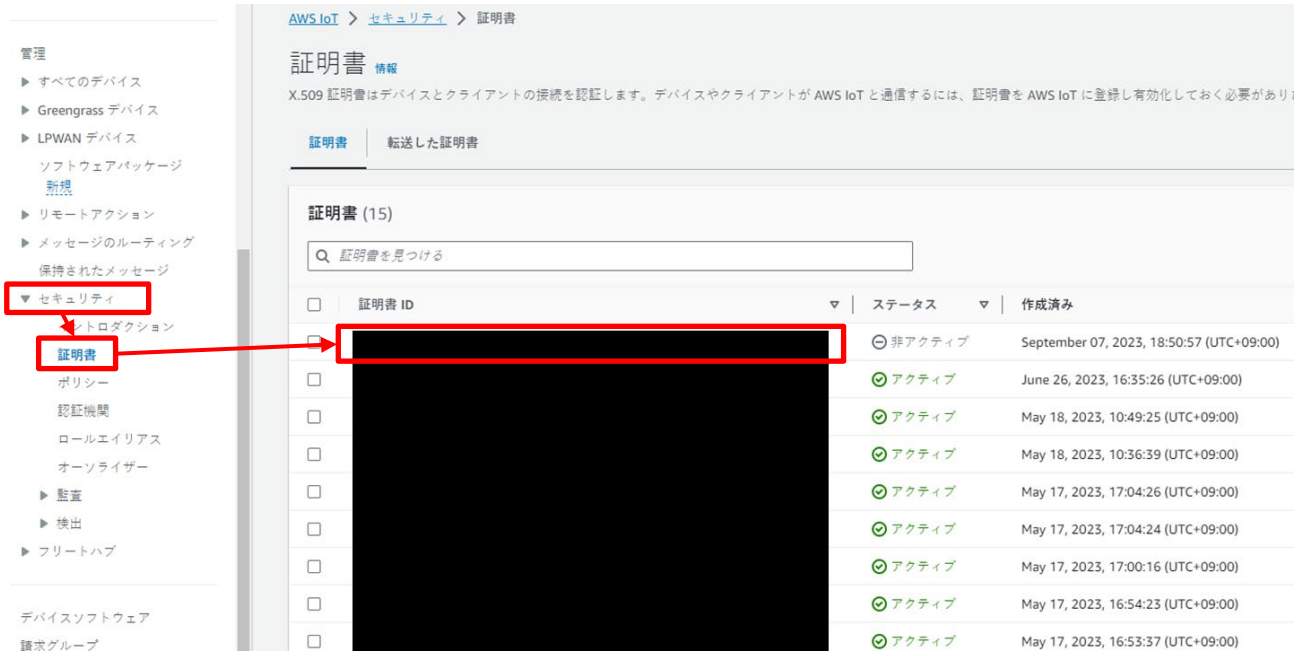

図 4-11 証明書の設定

「アクション」>「有効化」をクリックして証明書を有効化します。

また、「ポリシーをアタッチ」をクリックします。

図 4-12 証明書の設定 ポリシーのアタッチ(1)

「ポリシーをアタッチ」をクリックすると図 4-13 のダイアログが表示されます。

4.3.1 ポリシーの設定 で作成したフリートプロビジョニング実行時に使用するポリシーを選択、「ポリシーをアタッチ」で証明書にアタッチします。

以上でクレーム証明書、クレーム鍵の生成に関する設定は完了です。

図 4-13 証明書の設定 ポリシーのアタッチ(2)

4.3.3　フリートプロビジョニングテンプレートの作成

「複数のデバイスを接続」> 「プロビジョニングテンプレート」を選択し、

「プロビジョニングテンプレートを作成」ボタンをクリックします。

図 4-14 プロビジョニングテンプレートの作成(1)

「クレーム証明書を使用したデバイスのプロビジョニング」を選択し、[次へ]をクリックします

図 4-15 プロビジョニングテンプレートの作成(2)

テンプレートの作成画面で、プロビジョニングテンプレートステータス、テンプレート名、プロビジョニングロールを設定します。

プロビジョニングテンプレートステータスで「アクティブ」を選択し、プロビジョニングテンプレート名を入力します。

その後、「新しいロールを作成」をクリックして、ロール名を入力します。

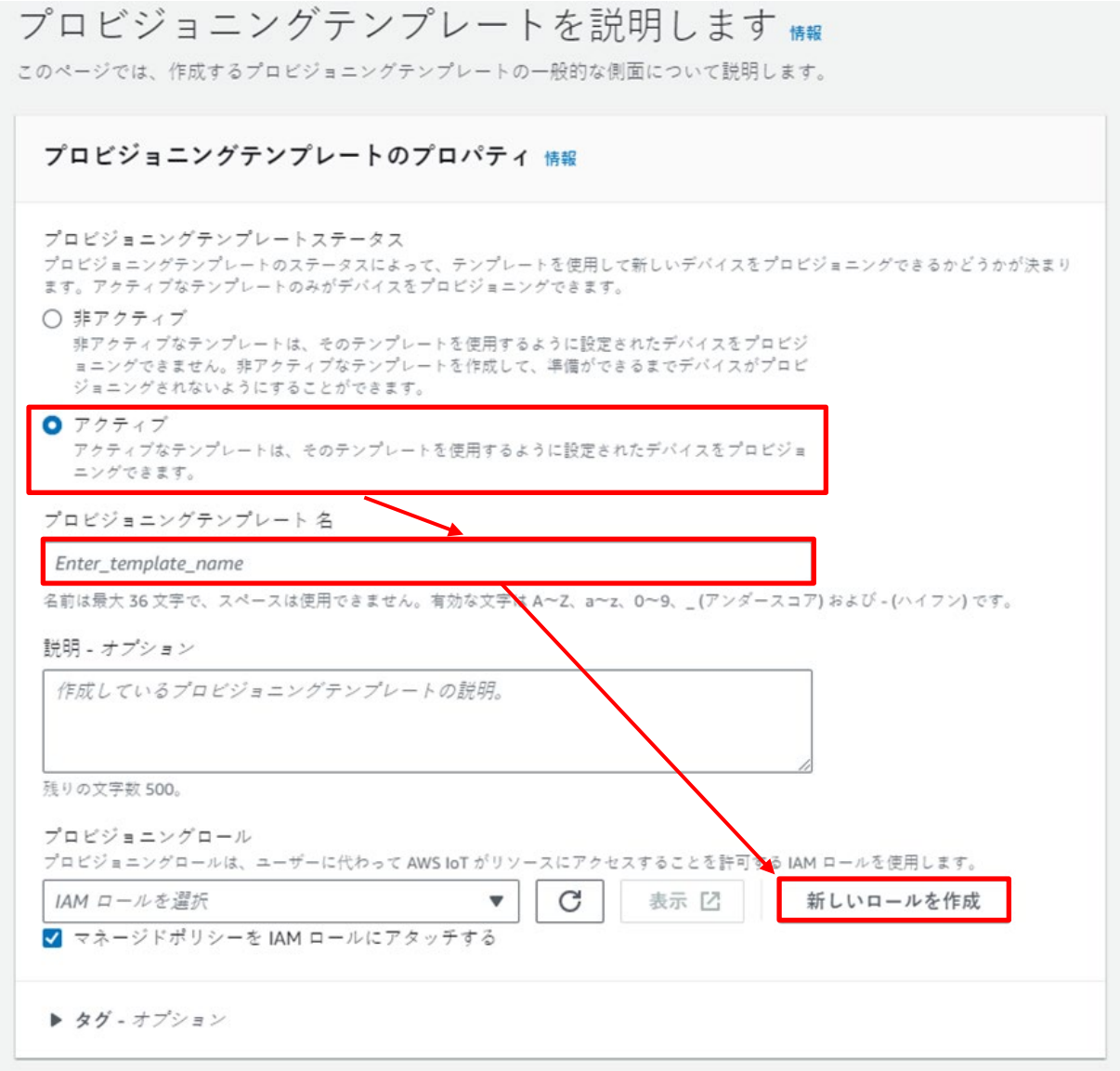

図 4-16 プロビジョニングテンプレートの作成(3)

ここではロール名を「fleet_demo」としますが、任意のロール名で作成をしてください。

次に「作成」をクリックします。

図 4-17 新しいロール作成画面

「クレーム証明書ポリシー」において 4.3.1 章で作成したフリートプロビジョニング実行時に使用するポリシーを選択します。

また「クレーム証明書」において 4.3.2 章で作成した証明書を選択して、[次へ]をクリックします。

図 4-18 プロビジョニングテンプレートの作成(4)

事前プロビジョニングアクションは「事前プロビジョニングアクションを使用しない」を選択します。

また、モノの自動作成で「デバイスのプロビジョニング時にモノのリソースを自動的に作成します」を有効に設定して、必要に応じてモノの名前プレフィックスに任意の文字列を入力します。

この文字列とプログラムで設定するシリアル番号から、AWS に登録されるモノの名前が生成されます。

プレフィックスを入力したら[次へ]をクリックします。

※ 本デモでは、事前のプロビジョニングアクションを使用しません。
事前プロビジョニングアクションの使用に関しては下記をご参照ください。
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」

図 4-19 プロビジョニングテンプレートの作成(5)

「デバイスのアクセス許可の設定」において 4.3.1 章で作成した作成されたモノにアタッチするポリシーにチェックを入れて[次へ]をクリックします。

図 4-20 プロビジョニングテンプレートの作成(6)

[テンプレートを作成]をクリックしてフリートプロビジョニングテンプレートの作成を完了します。

図 4-21 プロビジョニングテンプレートの作成(7)

4.4　サンプルプロジェクトの作成

4.5 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。

もしインポート機能にてデモプロジェクトを実行したい場合は Gatting Started Guide を参照してください。

(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 256byte を超えるとビルド時に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

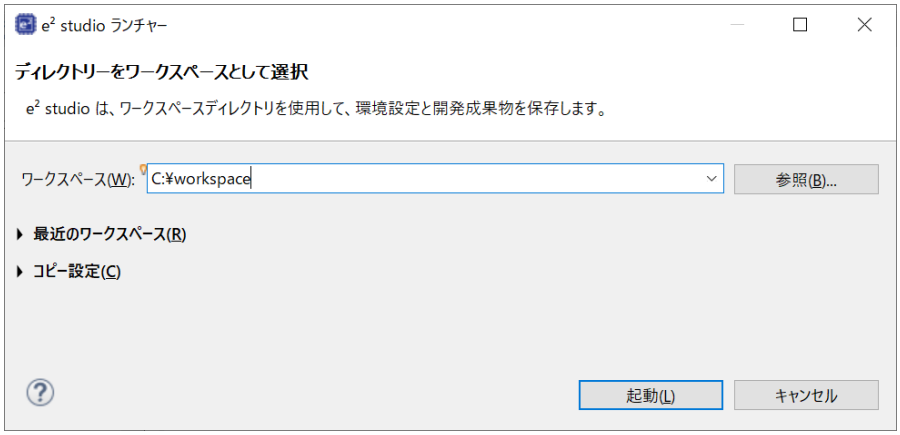

図 4-22 ワークスペース作成画面

e² studio の起動後、[ファイル]メニュー ⇒ [新規] ⇒ [Renesas C/C++ Project] ⇒ [Renesas RX] を選択し、New C/C++ Project ダイアログを起ち上げてください。

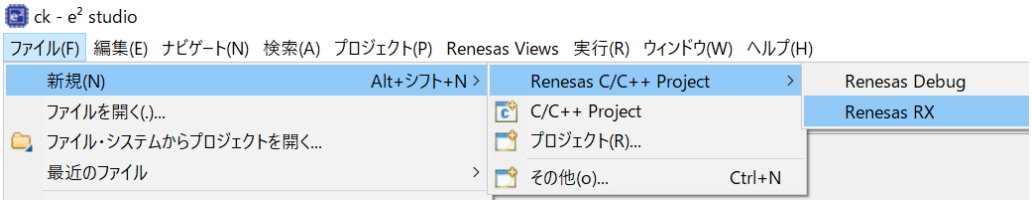

図 4-23 プロジェクトの新規作成メニュー

New C/C++ Project ダイアログで作成するプロジェクトの種類を選択します。ここでは、[All]を選択してから[Renesas CC-RX C/C++ Executable Project]を選択して、[次へ]ボタンを押下してください。

プロジェクトの種類選択（New Renesas CC-RX Executable Project）ダイアログが開かれます。

GCC をご利用になる場合は[GCC for Renesas RX C/C++ Execute Project]を選択してください。

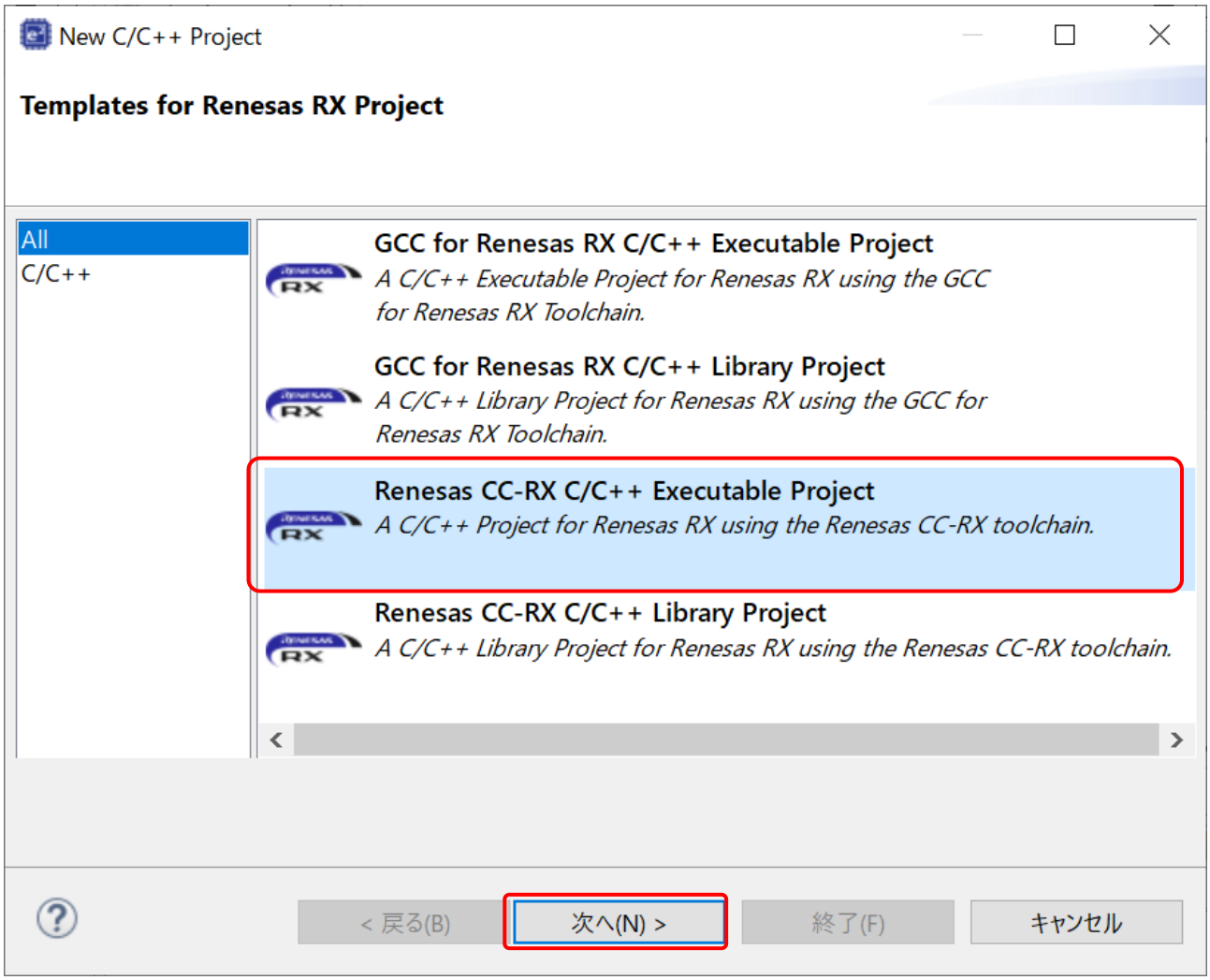

図 4-24 プロジェクトの種類選択画面

(2)　サンプルプロジェクトの生成

ここではプロジェクトの名称を設定します。プロジェクト名に”fleet_demo”と入力し、[次へ]ボタンを押下してください。ツールチェーン・デバイスとデバッグ設定ダイアログが開かれます。

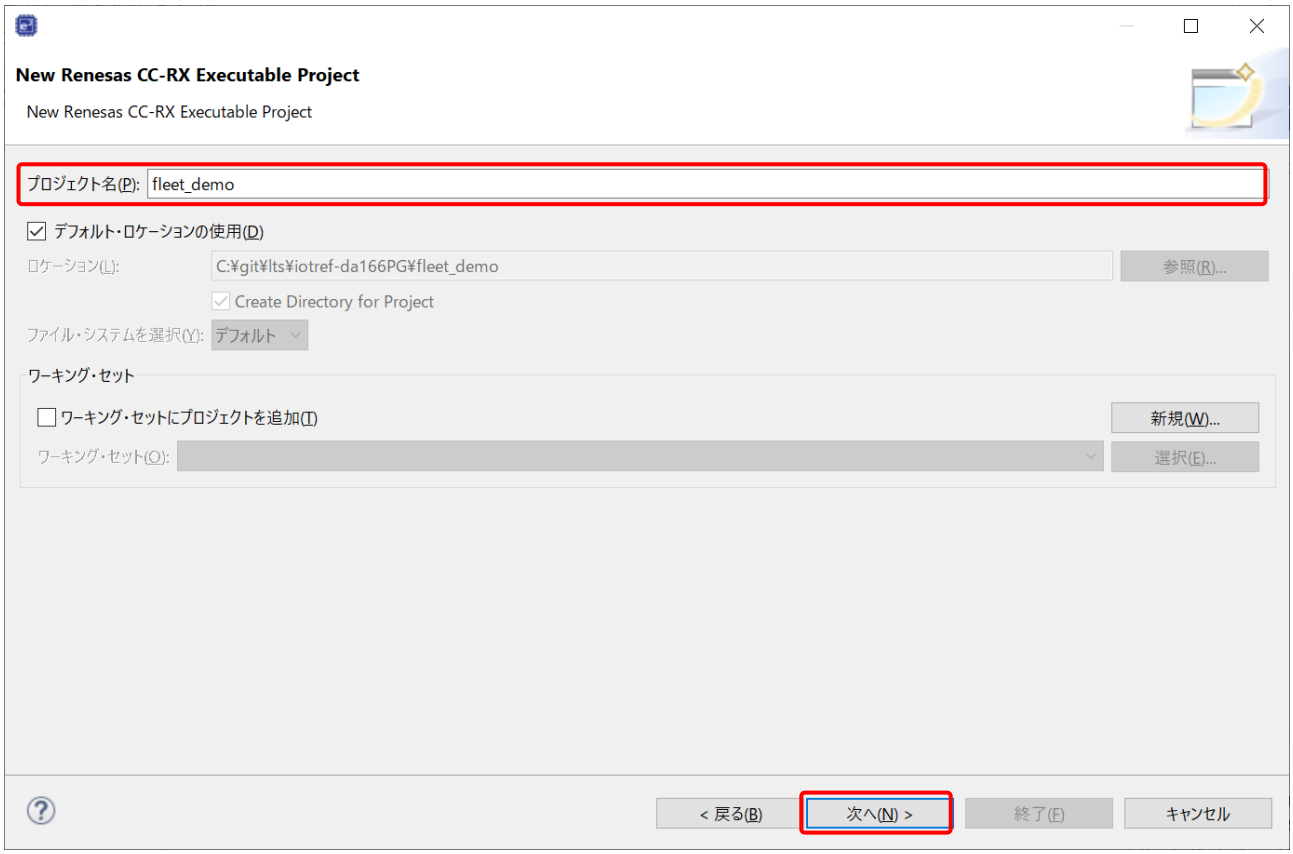

図 4-25 プロジェクトの名称設定画面

プロジェクトに使用するツールチェーンとデバイスおよび、デバッグの設定を行います。

Toolchain はプロジェクトの種類で選択されたものが設定されます。ツールチェーンのバージョンを変更したい場合は[ツールチェイン・バージョン]より必要なバージョンを選択してください。

RTOS は"Free RTOS (with IoT libraries)"を選択し、RTOS Version より"202210.01-LTS-rx-1.2.1"を選択してください。初めて e2 studio をご利用になる場合やリストに必要なバージョンが表示されない場合は、[Managed RTOS Versions...]をクリックすると RTOS 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[ダウンロード]ボタン押下してダウンロードしてください。

［Target Board］は"CK-RX65N"を選択してください（ターゲット・デバイスは自動でセットされます）。

[Bank Mode]は”Dual Bank”を選択してください。

すべての設定が完了したら[次へ]ボタンを押下してください。

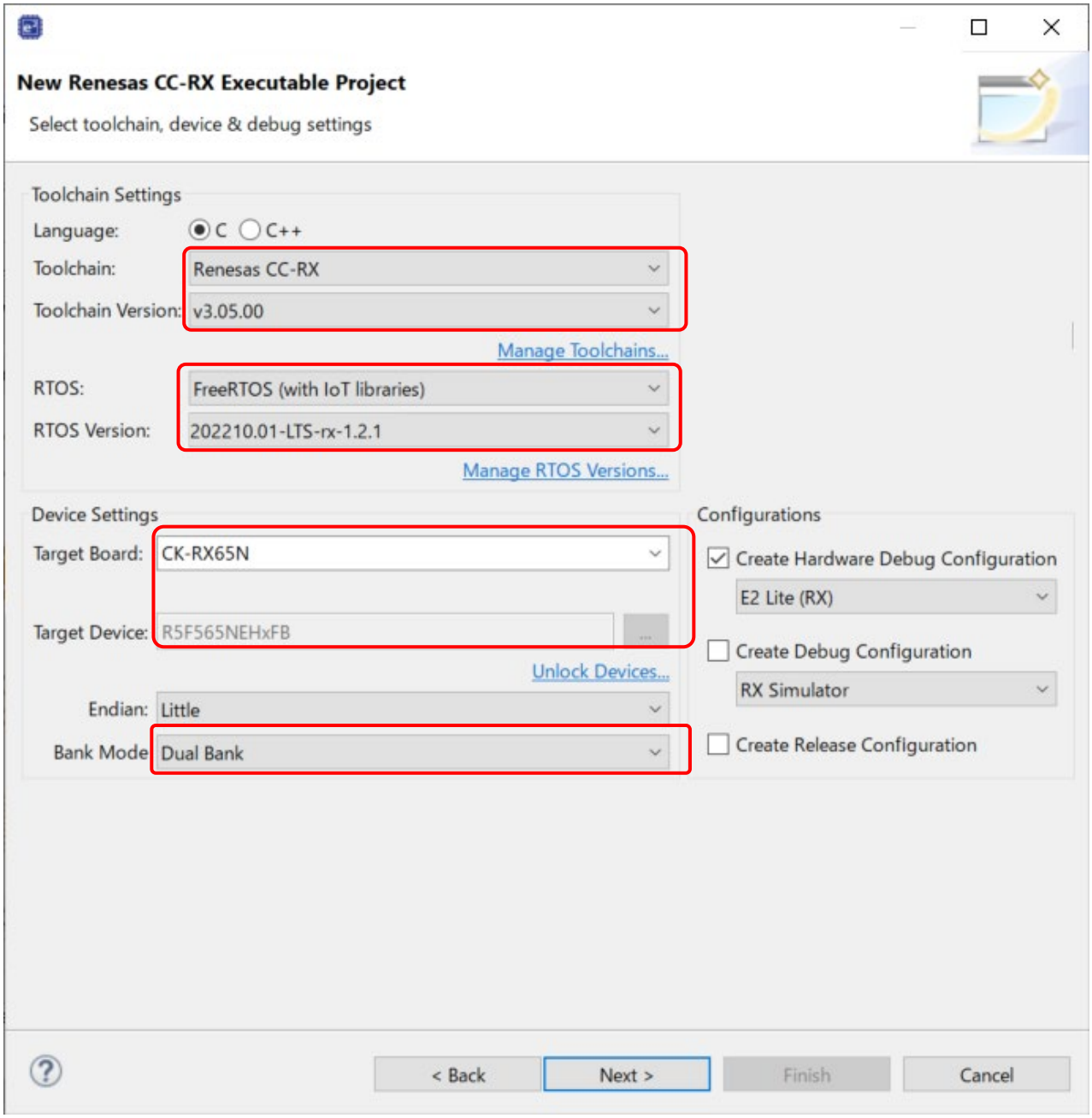

図 4-26 ツールチェーン・デバイスとデバッグ設定画面

コーディングアシストツールの選択ダイアログが表示されますのでそのまま[次へ]ボタンを押下してください。

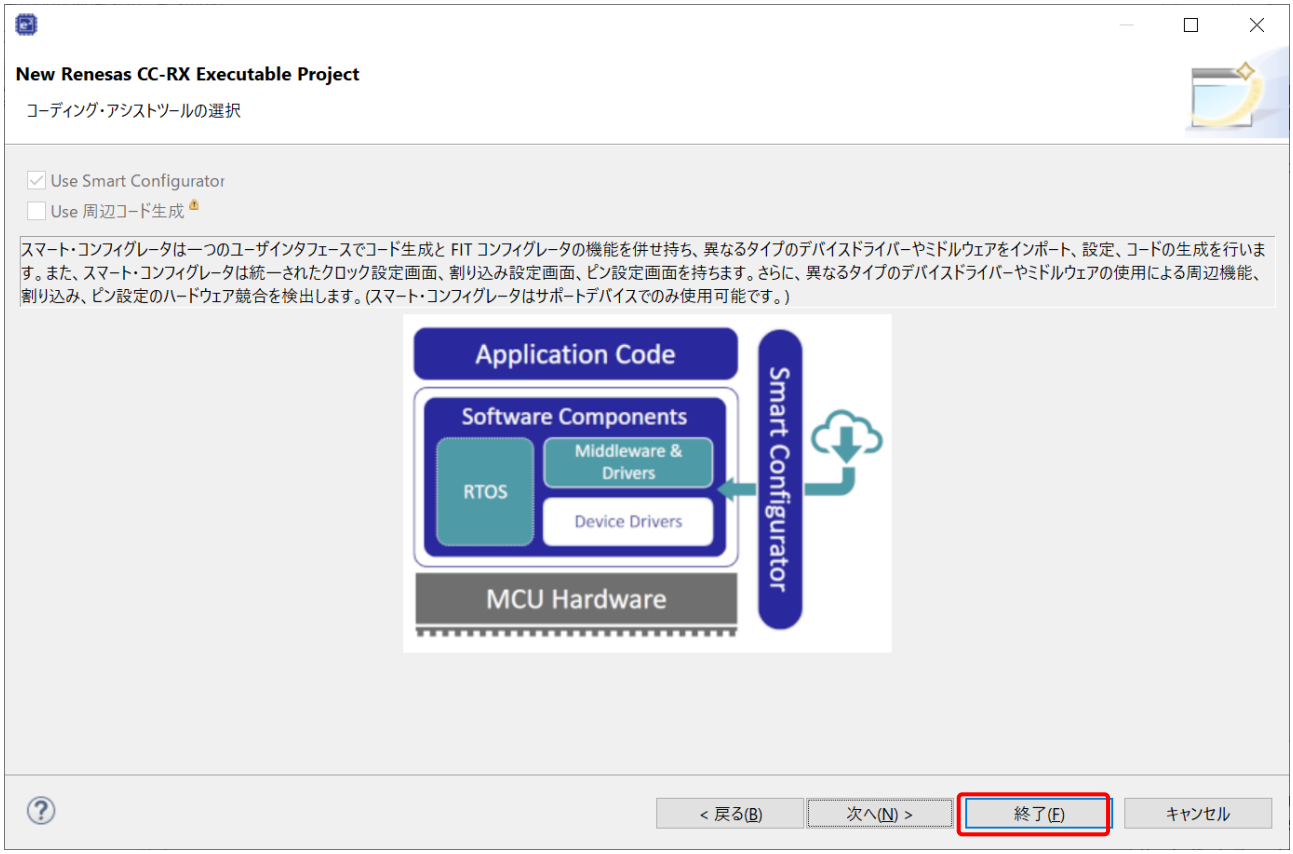

図 4-27 コーディングアシストツールの選択画面

RTOS プロジェクト設定の選択ダイアログに、サンプルプロジェクトの一覧が表示されます。

スクロールバーを操作し、リストより"(Cellular) PubSub/MQTT with Fleet Provisioning sample project"を選択し、[次へ]ボタンを押下してください。

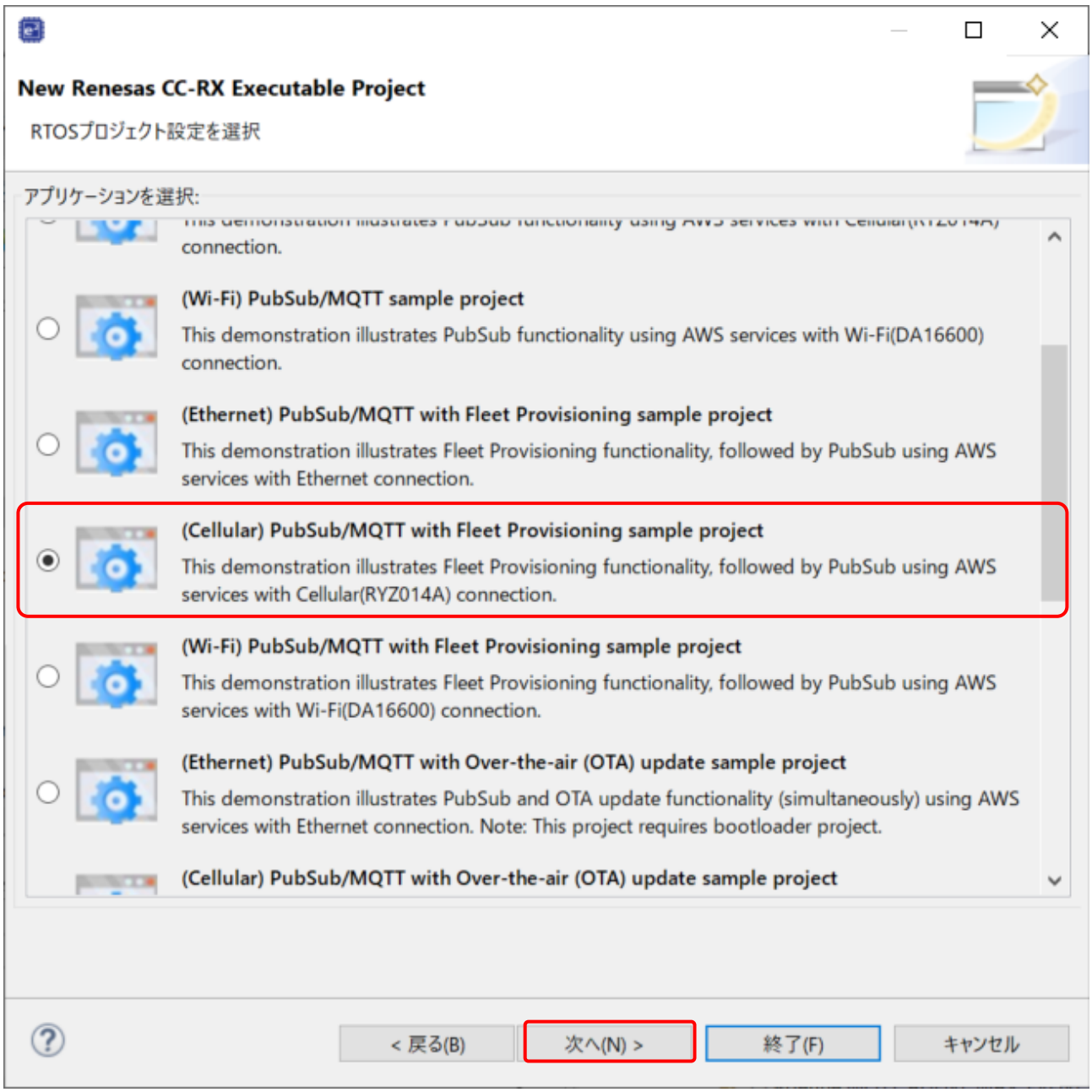

図 4-28 RTOS プロジェクト設定選択画面

生成ファイルの内容設定(Setting The Contents of Files to be Generated)ダイアログが表示されます。そのまま[次へ]ボタンを押下してください。

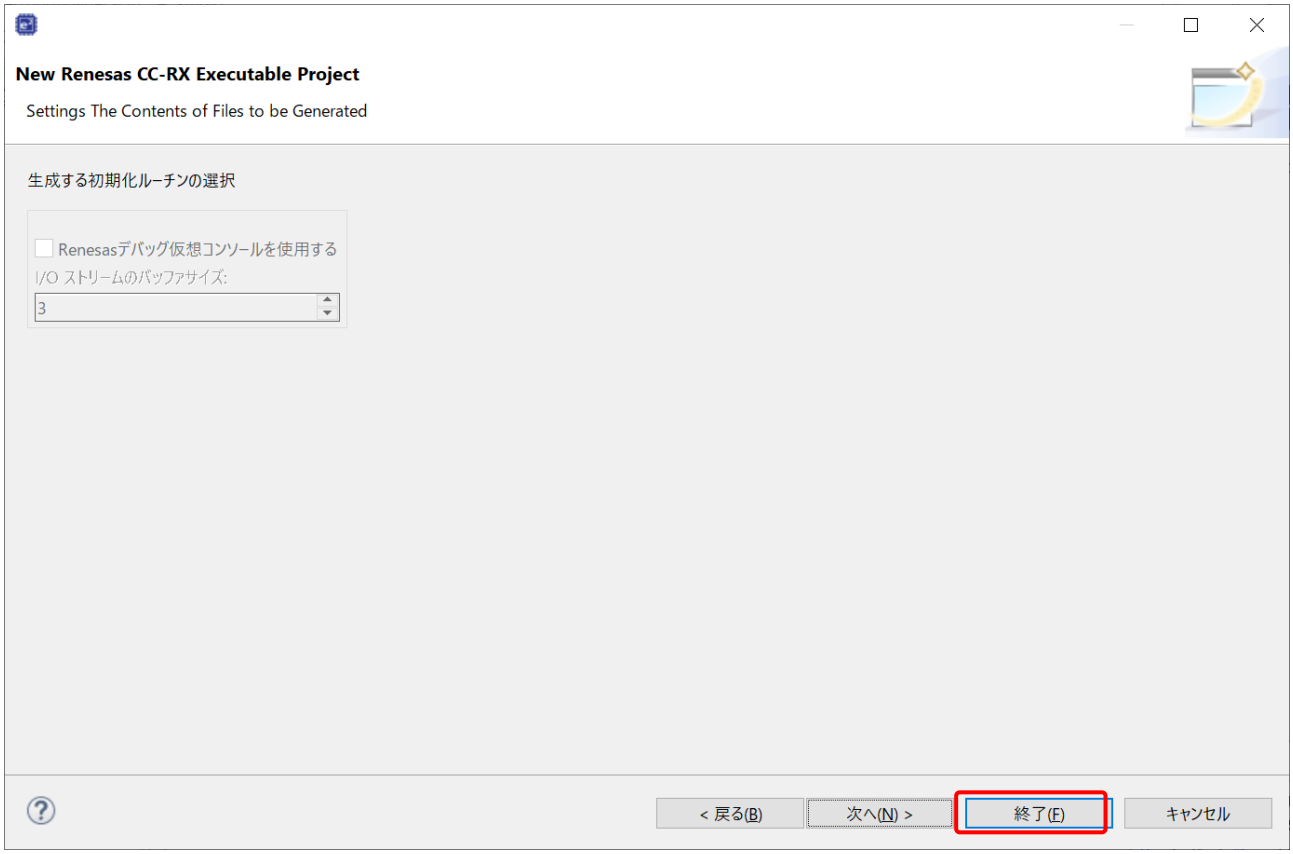

図 4-29 生成ファイルの内容設定画面

プロジェクト生成の終了画面が表示されます。問題なければ[終了]ボタンを押下してください。

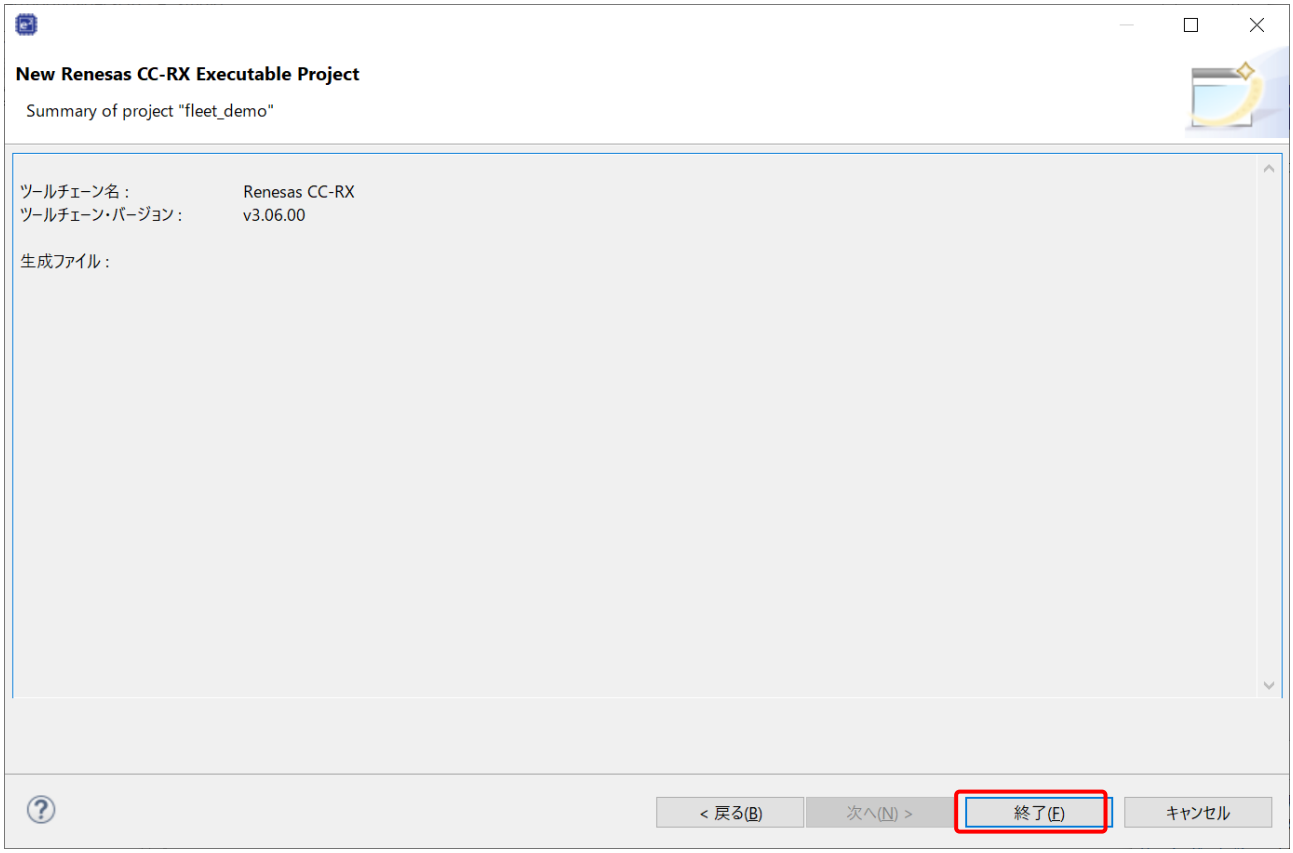

図 4-30 プロジェクト生成の終了画面

マーケットプレイスで利用可能なエディターのダイアログが表示されたら[キャンセル]ボタンを押下して閉じてください。

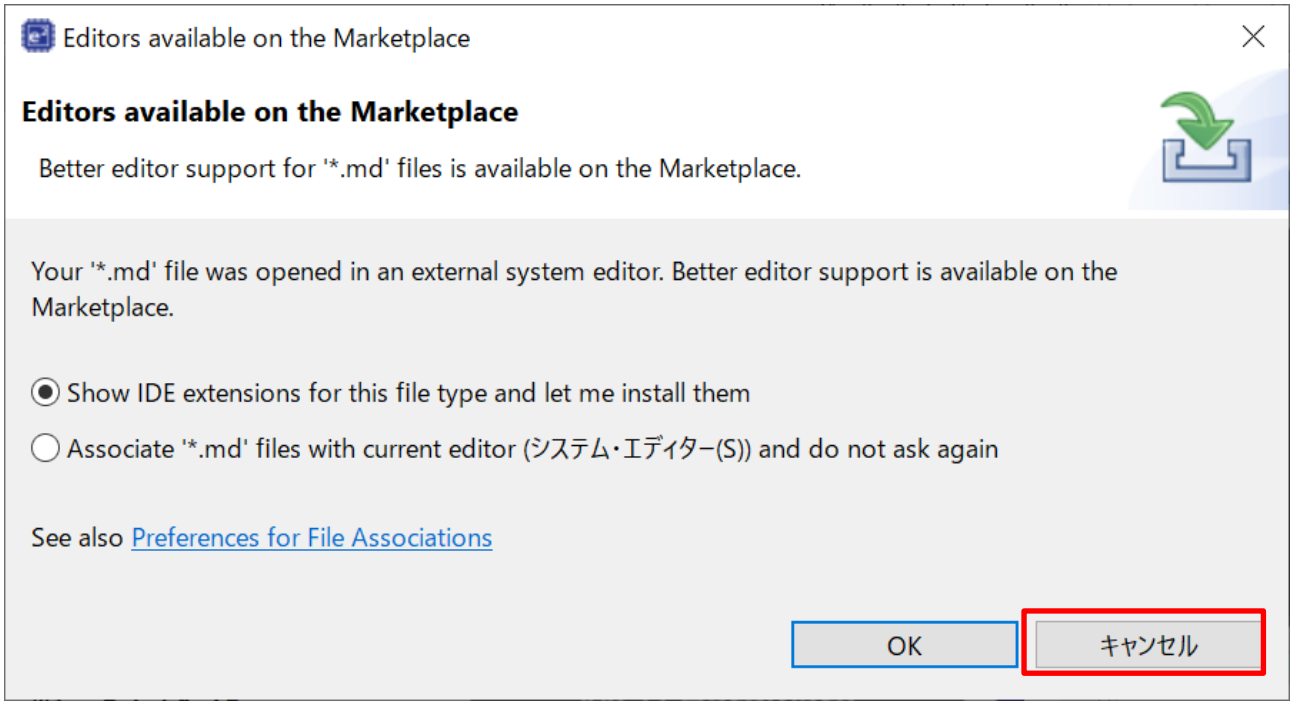

図 4-31 マーケットプレイスで利用可能なエディター画面

以下のように e² studio にプロジェクトが生成されます。

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、[ウィンドウ] ⇒[ビューの表示] ⇒[プロジェクト・エクスプローラー]を選択してください。

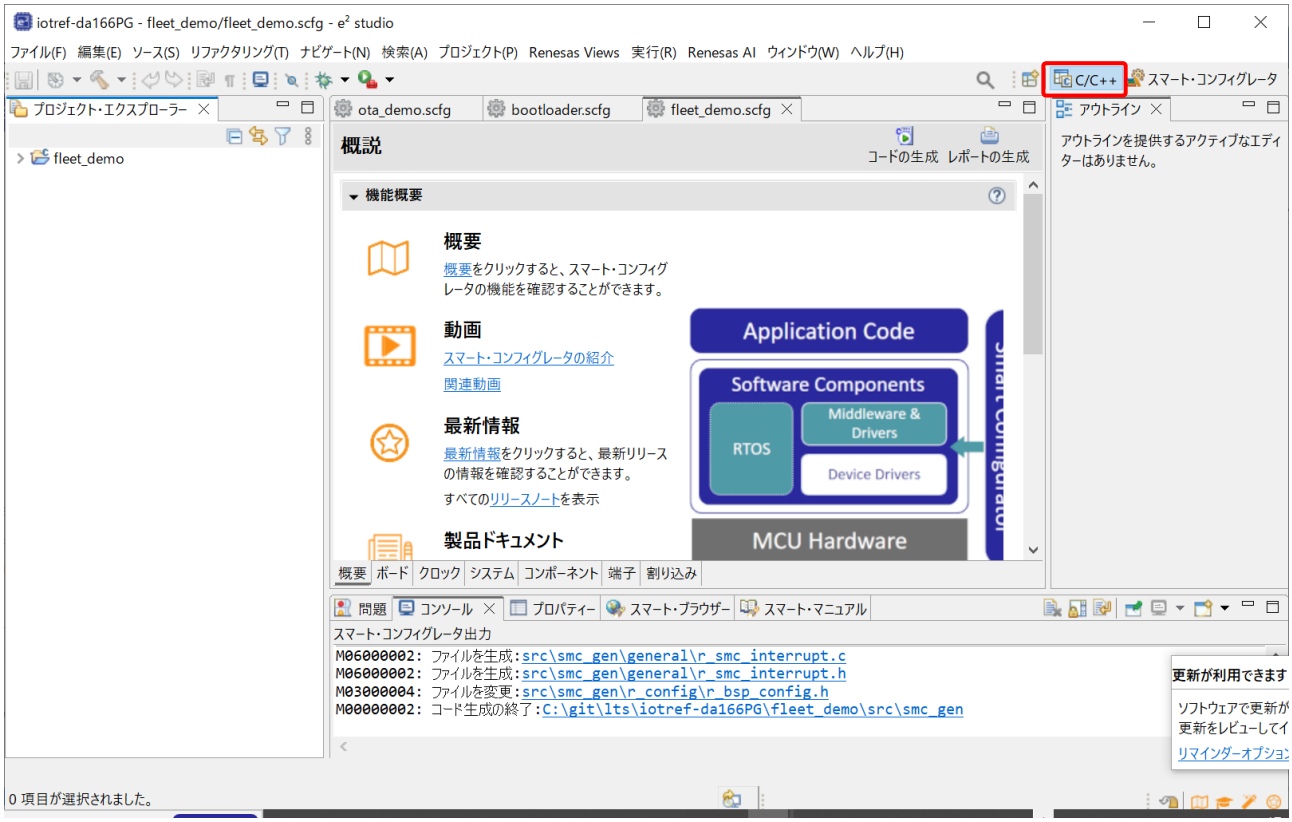

図 4-32 フリートプロビジョニングサンプルプロジェクト生成後の画面

以降、プロジェクト名「aws_ryz014a_ck_rx65n」を「fleet_demo」に適宜読み替えてください。

4.5　FreeRTOS の設定

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

e^2 studio のプロジェクト・エクスプローラーから aws_ryz014a_ck_rx65n/src/frtos_config/demo_config.h を開き、「ENABLE_FLEET_PROVISIONING_DEMO」を「1」に変更します。

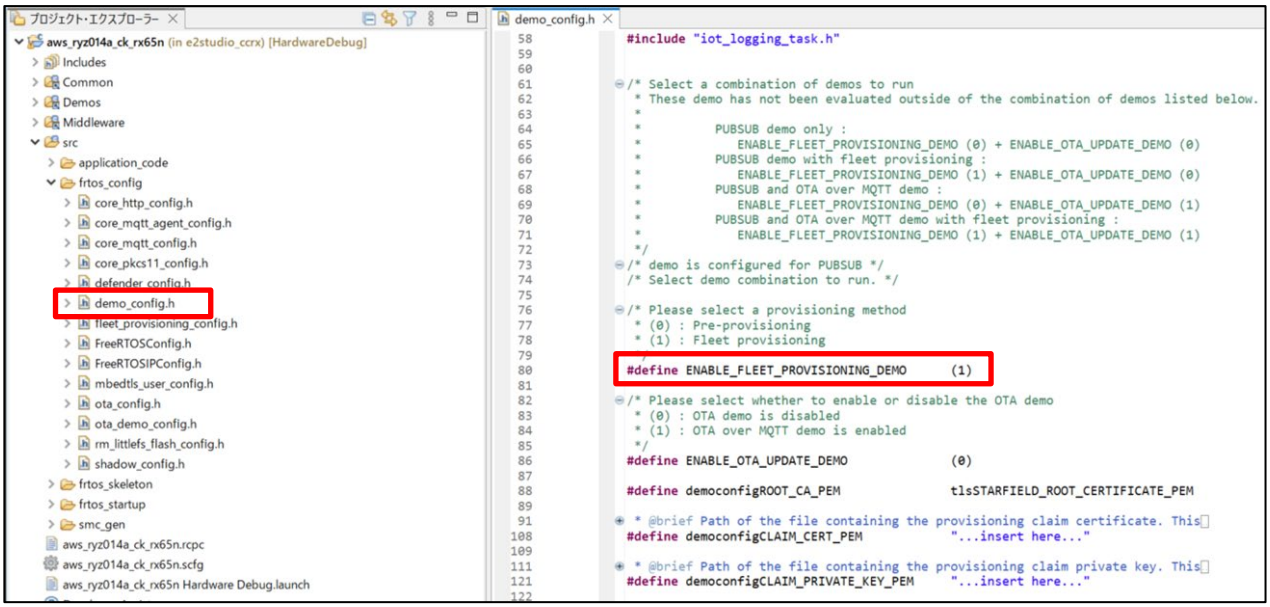

図 4-33 demo_config.h の修正箇所

4.5.2　Cellular 情報の設定

e² studio のプロジェクト・エクスプローラーから aws_ryz014a_ck_rx65n/aws_ryz014a_ck_rx65n.scfg を開き、スマート・コンフィグレータを起動します。(図 4-34)
スマート・コンフィグレータのコンポーネントタブを選択し、コンポーネントから Middleware > ジェネリック > r_cellular を選択します。ご使用の SIM カードに合わせて、「Access point name」「Access point login ID」「Access point password」「SIM card PIN code」の各項目を設定してください。入力する内容が無い場合は空欄としてください。(図 4-35)
Cellular 情報を入力したら、「コードの生成」をクリックして、設定した内容をプログラムに反映します。

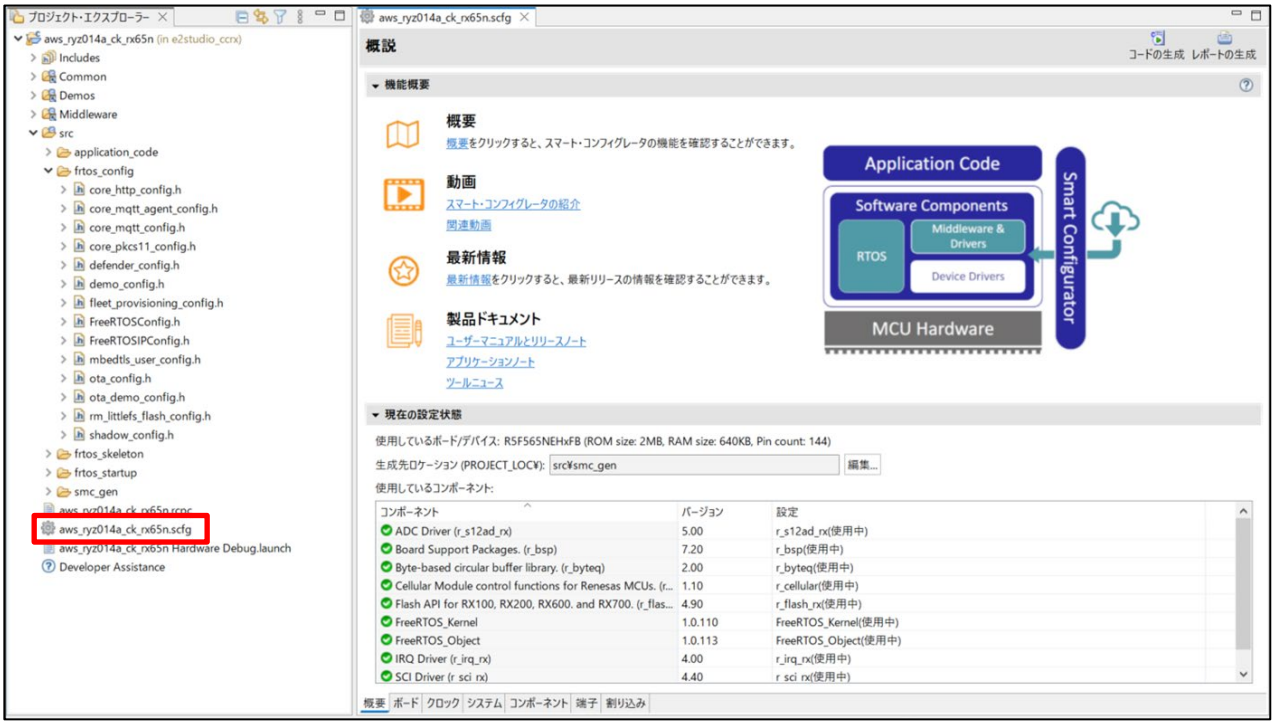

図 4-34 スマート・コンフィグレータの起動

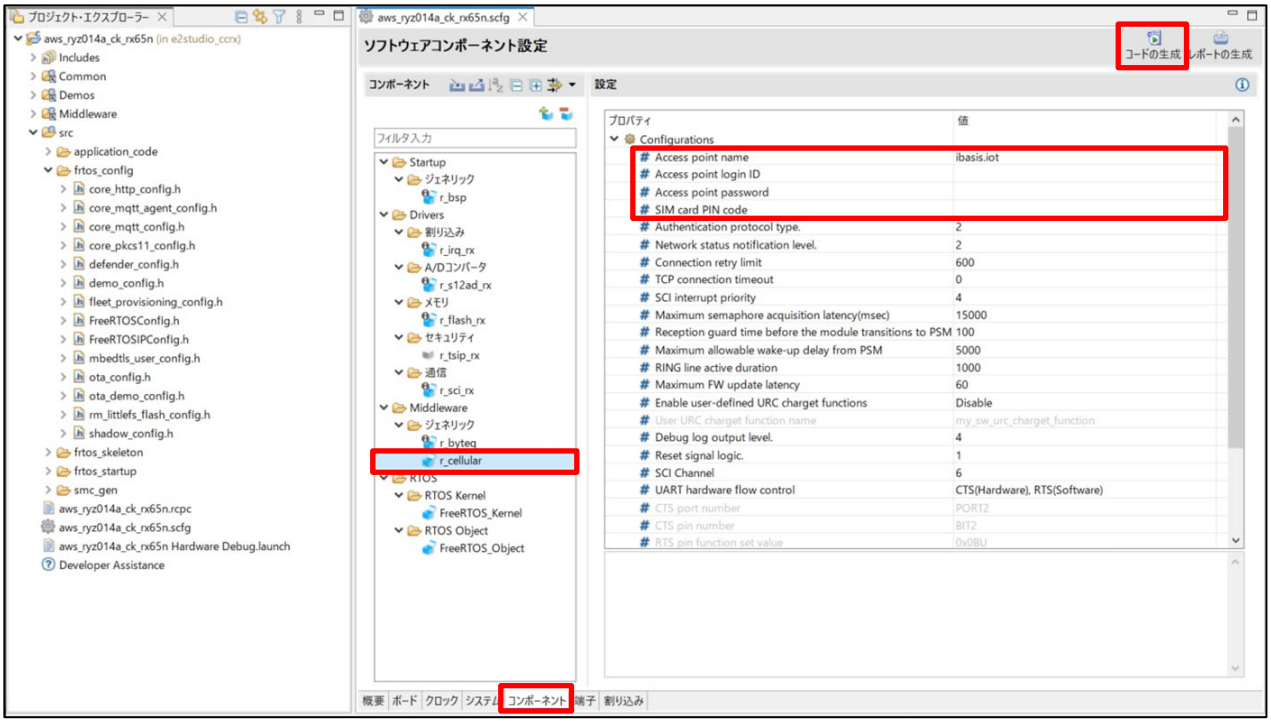

図 4-35 Cellular 情報の入力箇所

※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 (Only using Wi-Fi)」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 (Only using Wi-Fi)を必要に応じて参照してください。

4.6　ビルドと実行

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_ryz014a_ck_rx65n 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。

次に、e^2 studio メニューバーの[実行] > [デバッグの構成]をクリック、デバッグ構成画面を開きます。デバッグの構成画面左の、Renesas GDB Hardware Debugging > aws_ryz014a_ck_rx65n Hardware Debug を選択し、Debugger タブ、Connection Settings タブを選択します。(図 4-36 中 矢印)

図 4-36 の赤枠の通りに設定されていることを確認できたら、「デバッグ」ボタンで、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。

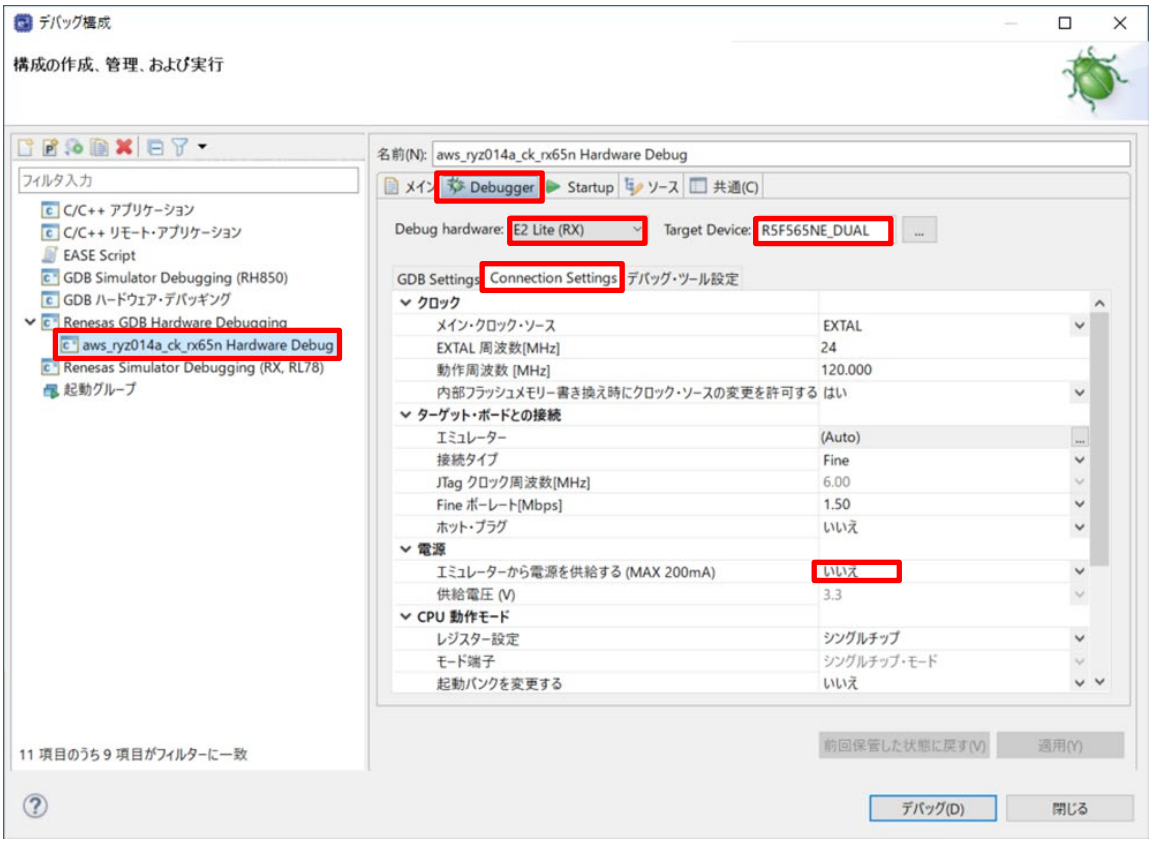

図 4-36 デバッグ構成

クレーム証明書、クレーム秘密鍵、エンドポイント、プロビジョニングテンプレート名を入力するために Tera Term を起動します。

Tera Term 起動後、「シリアル」を選択して、「USB Serial Device」を選択した後に[OK]ボタンをクリックします。

図 4-37 Tera Term の起動

「設定」>「シリアルポート」をクリックして、赤枠のようにシリアルポートの設定を変更した後に[現在の接続を再設定]ボタンをクリックします。

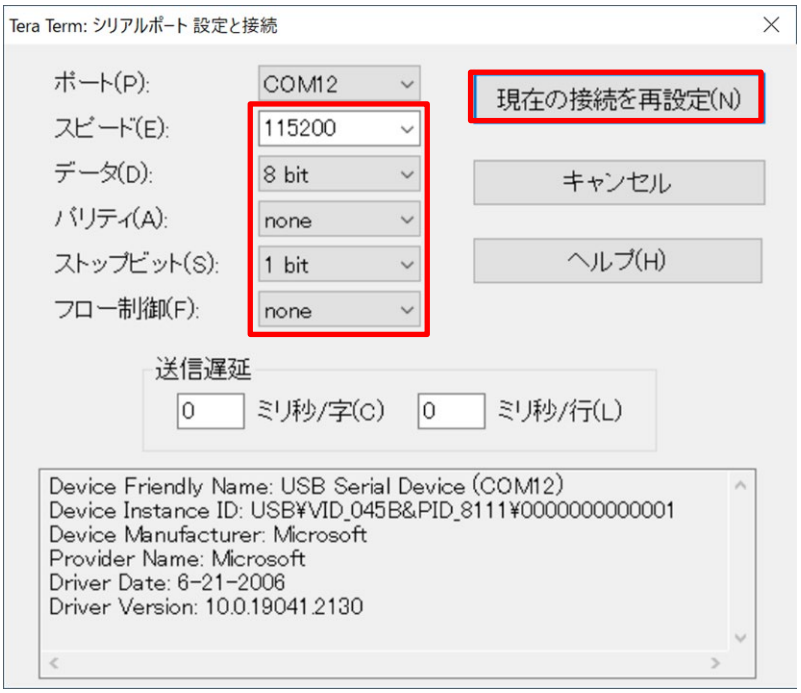

図 4-38 シリアルポート設定

「設定」>「端末」をクリックして、赤枠のように「受信」を「AUTO」、「送信」を「CR+LF」のように設定した後、[OK]ボタンをクリックします。

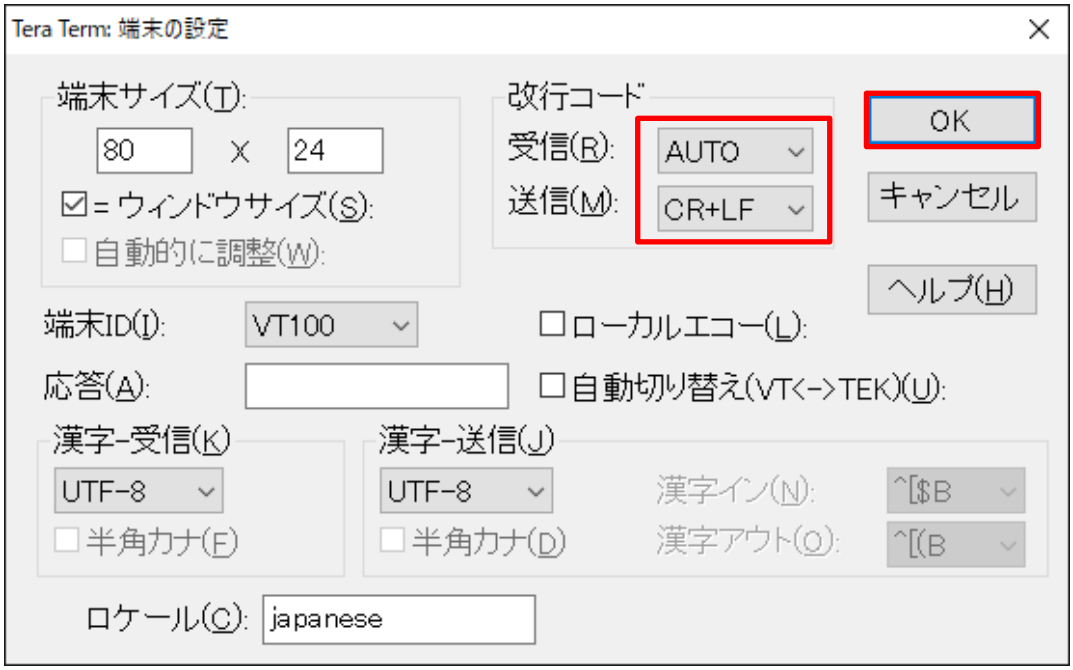

図 4-39 端末設定

AWS IoT Core で[MQTT テストクライアント]を選択して、「トピックのフィルター」に「#」を入力して「サブスクライブ」をクリックします。

図 4-40 MQTT テストクライアントの設定

e² studio で「再開（F8）」ボタンを押すと Tera Term 上に下記のような文字列が表示されるので、10 秒以内に Tera Term 上で「**CLI**」と入力して「**Enter**」を押します。

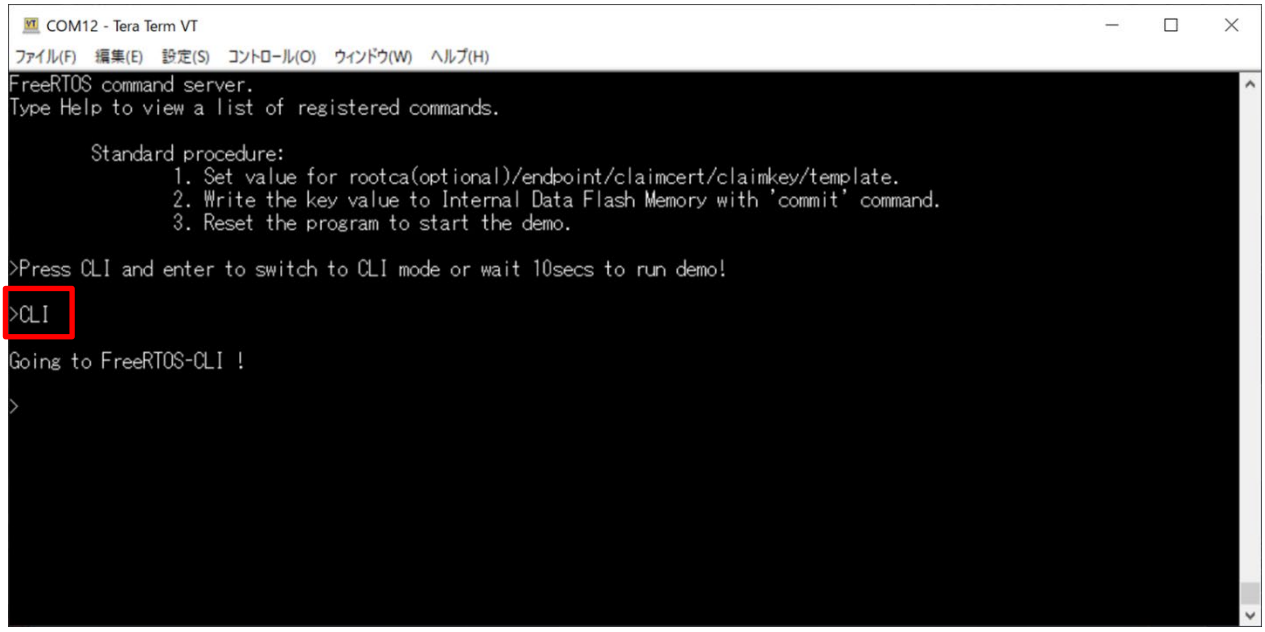

図 4-41 CLI による情報入力(1)

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。

これで格納された各種情報はすべて消去されます。

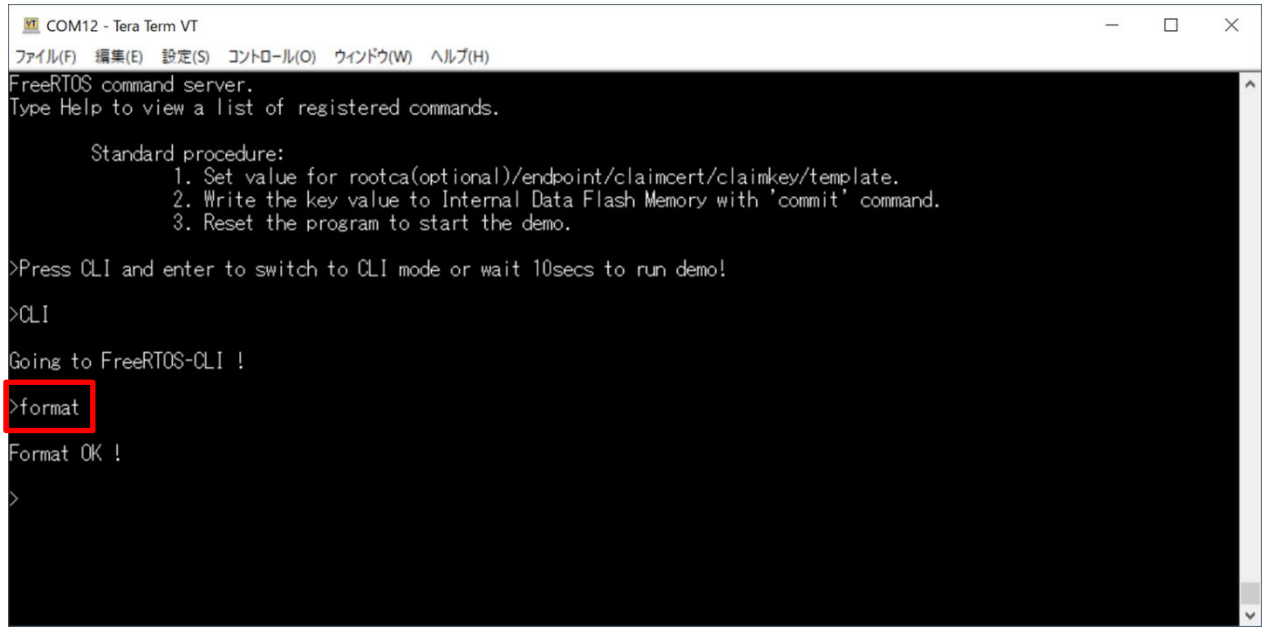

図 4-42 CLI による情報入力(2)

エンドポイントを入力するために Tera Term 上で「**conf set endpoint <endpoint>**」と入力して「**Enter**」を押します。

「**<endpoint>**」は AWS IoT Core の「設定」>「デバイスデータエンドポイント」>「エンドポイント」に表示されている「**xxxxxxxxxx.amazonaws.com**」を入力します。

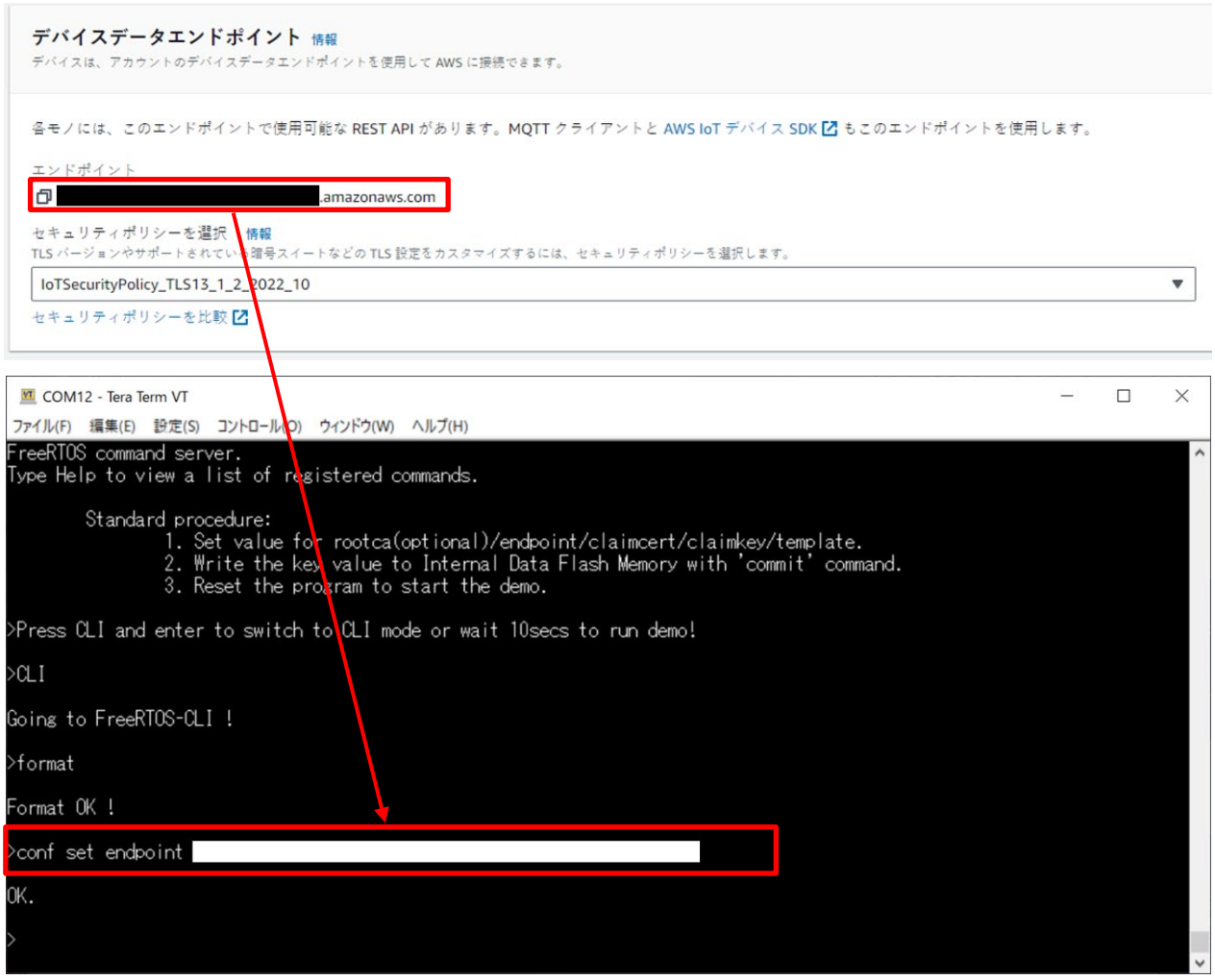

図 4-43 CLI による情報入力(3)

プロビジョニングテンプレート名を入力するために Tera Term 上で「**conf set template <template_name>**」と入力して「**Enter**」を押します。

「**<template_name>**」は 4.3.3 章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。

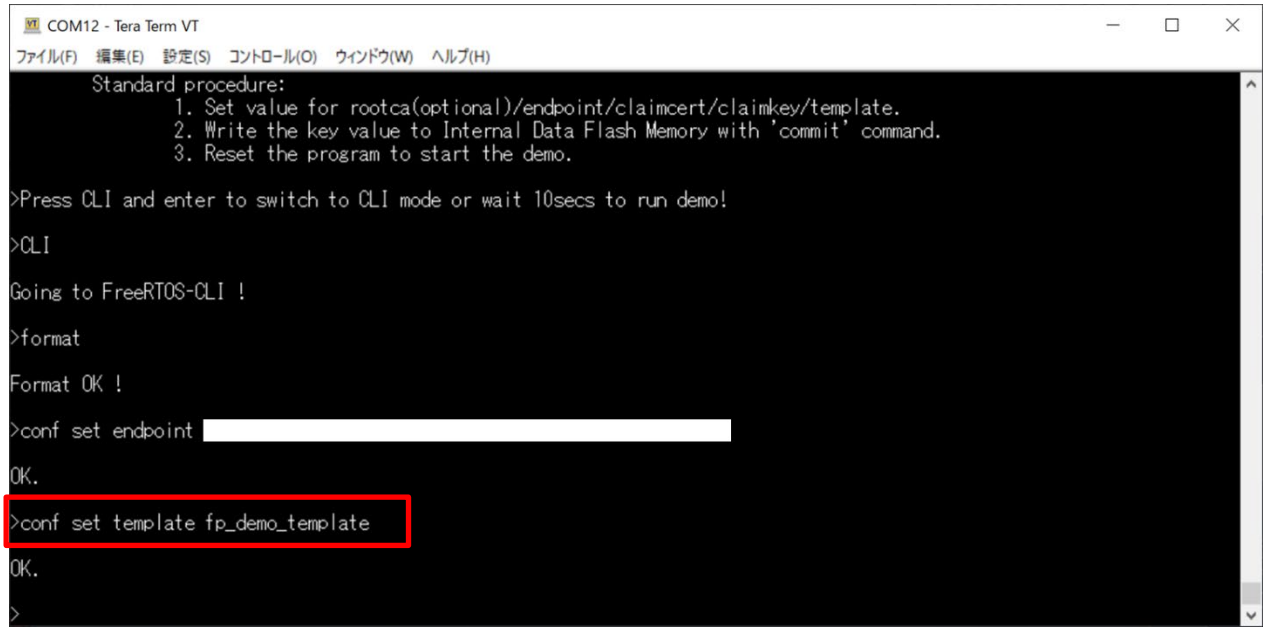

図 4-44 CLI による情報入力(4)

プロビジョニングクレーム証明書を入力するために Tera Term 上で「**conf set claimcert** 」と入力します。次に 4.3.2 章で作成したプロビジョニングクレーム証明書ファイル(**xxxx-certificate.pem.crt**)を Tera Term にドラッグアンドドロップ(ファイル送信)します。最後に Tera Term 上で「**Enter**」を押します。

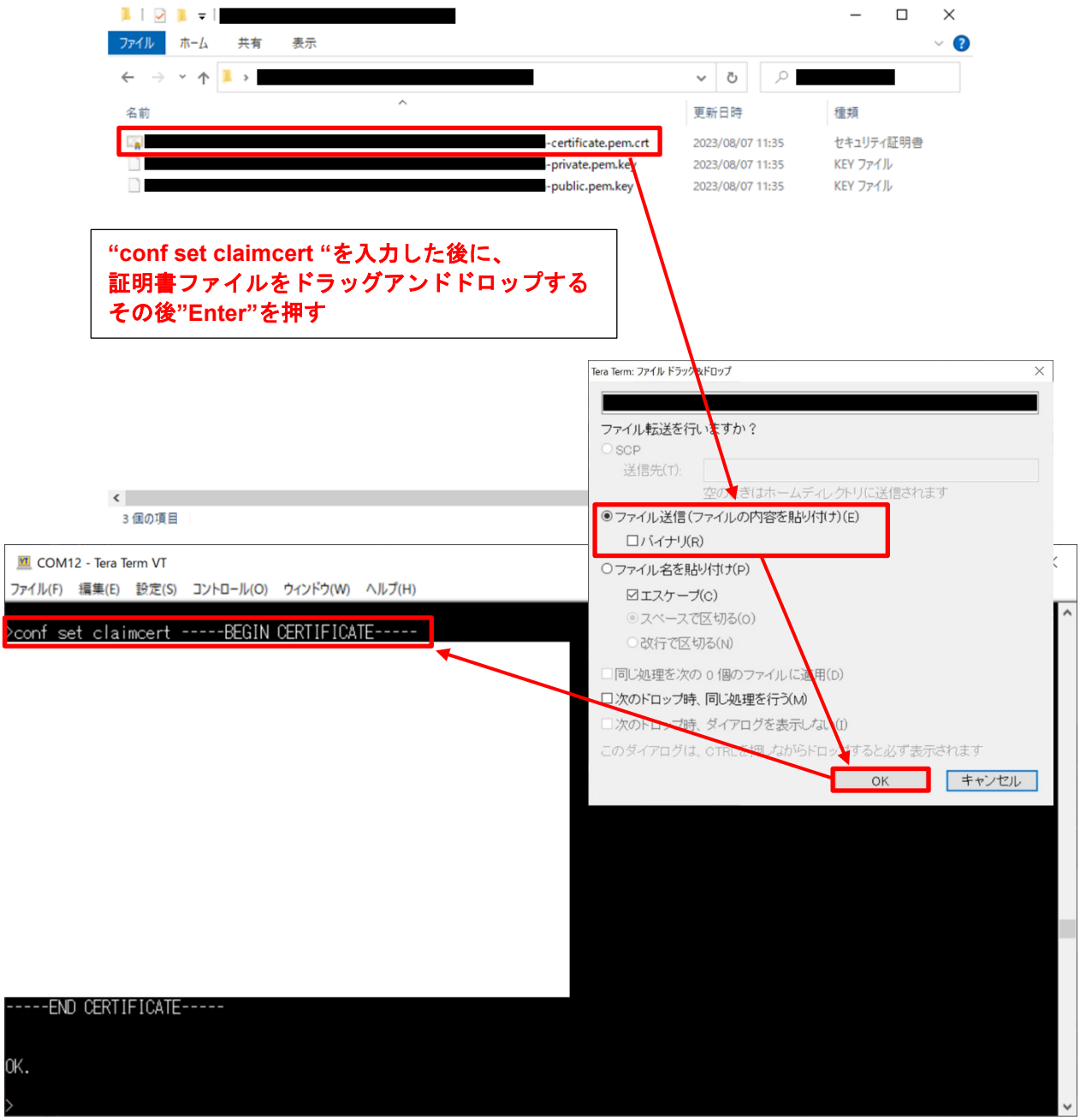

図 4-45 CLI による情報入力(5)

プロビジョニングクレーム秘密鍵を入力するために Tera Term 上で「**conf set claimkey** 」と入力します。次に 4.3.2 章で作成したプロビジョニングクレーム秘密鍵ファイル(**xxxx-private.pem.key**)を Tera Term にドラッグアンドドロップ(ファイル送信)します。最後に Tera Term 上で「**Enter**」を押します。

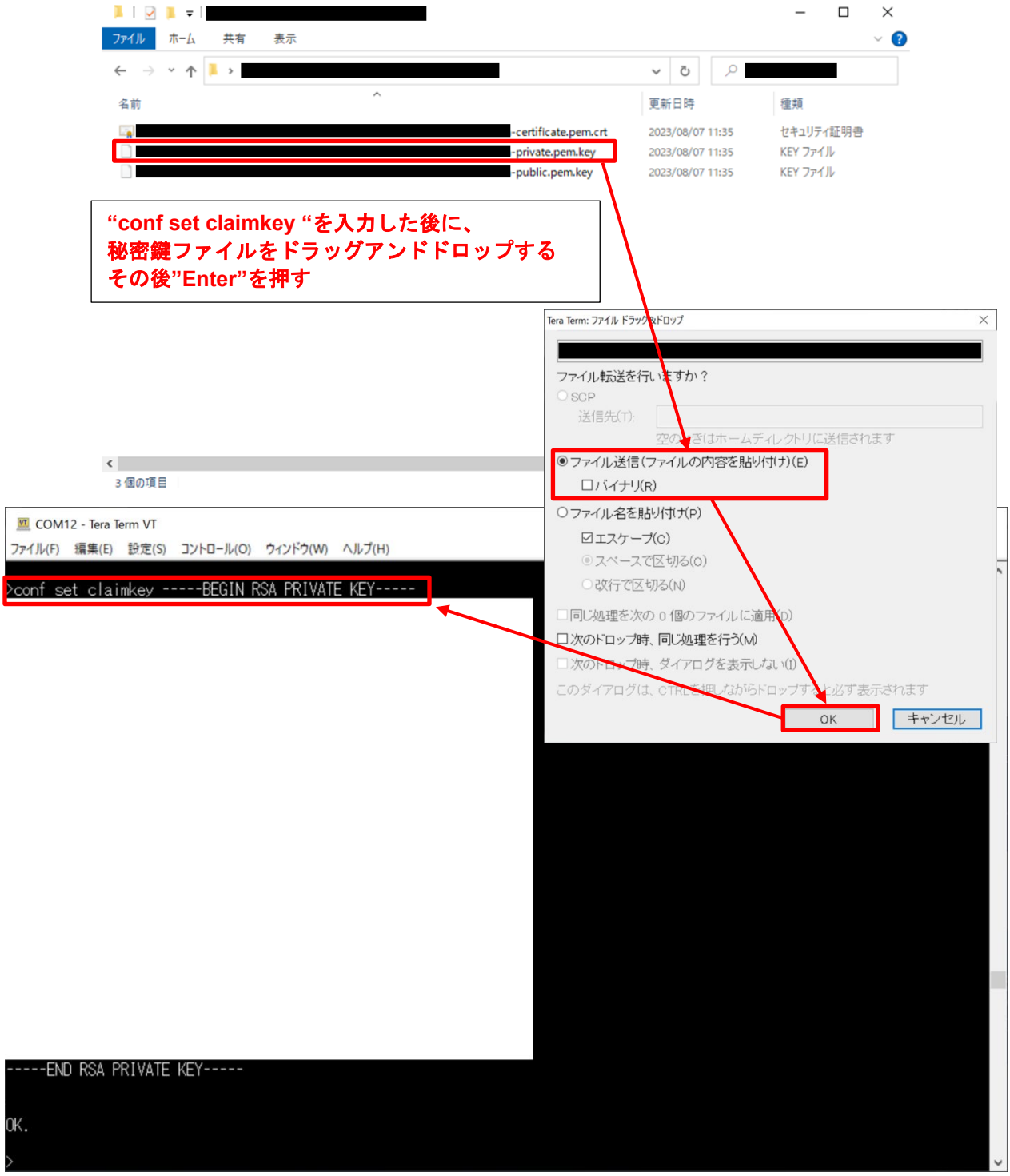

図 4-46 CLI による情報入力(6)

今まで入力した情報をデータフラッシュに格納するために、Tera Term 上で「**conf commit**」と入力して「**Enter**」を押します。

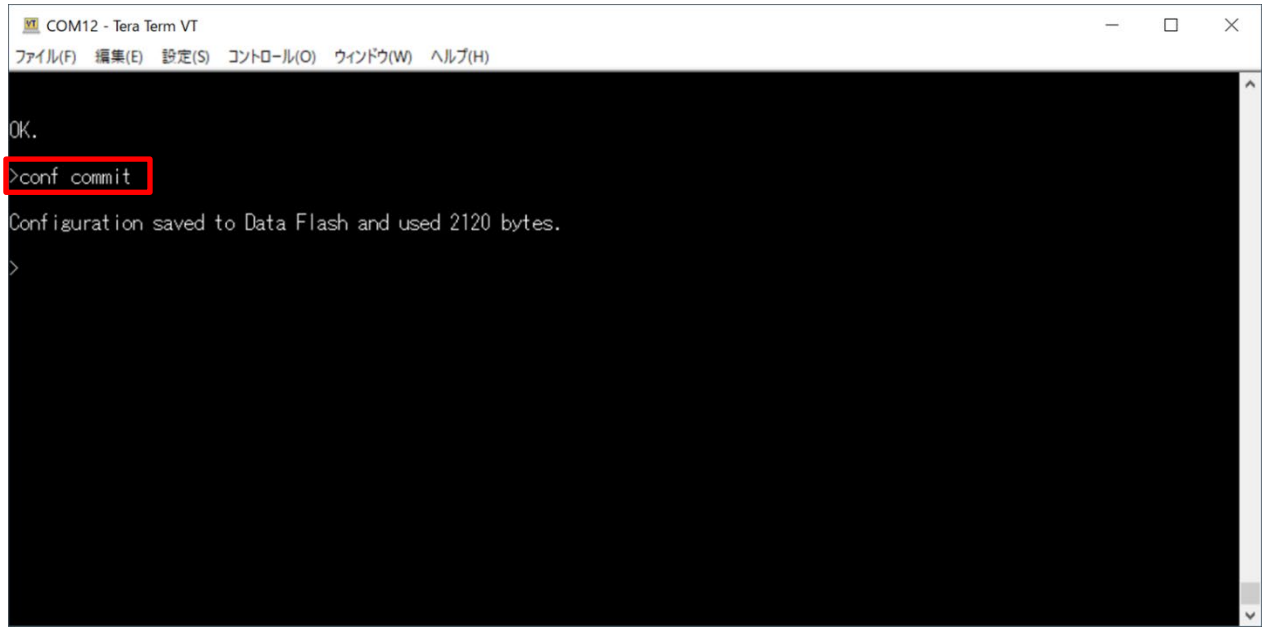

図 4-47 CLI による情報入力(7)

デモを起動するために、Tera Term 上で「**reset**」と入力して「**Enter**」を押します。

リセット後、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。

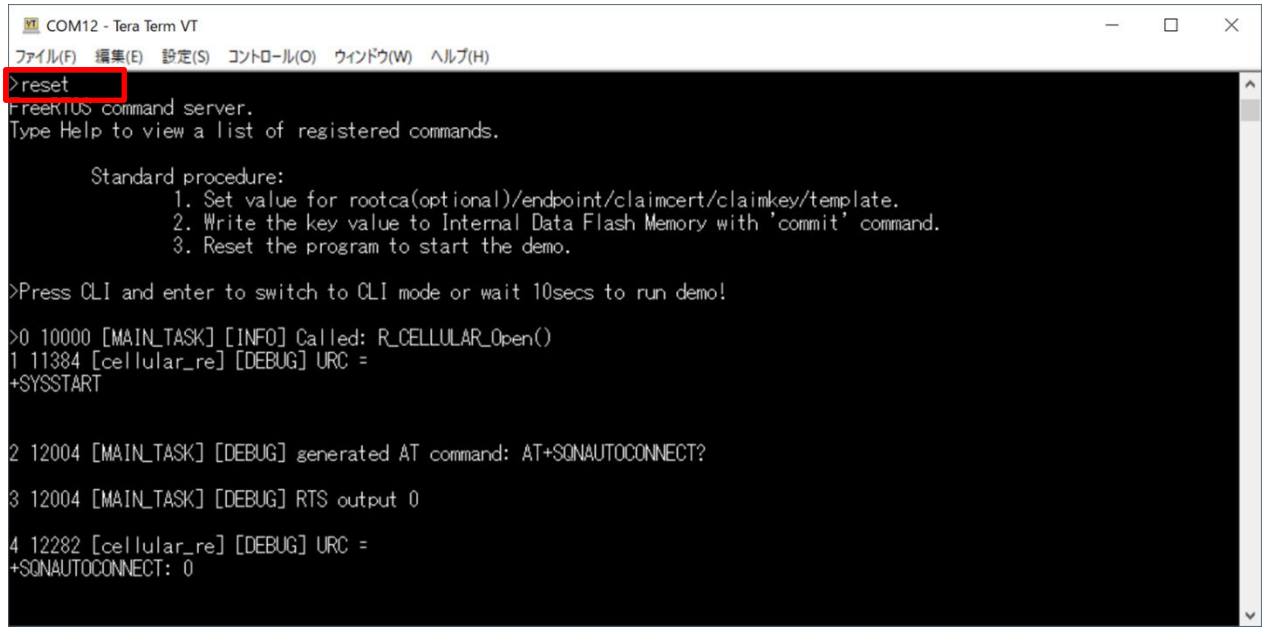

図 4-48 CLI による情報入力(8)

4.7　実行結果の確認

図 4-49 にフリートプロビジョニングデモの実行ログを示します。

（Tera Term に表示されるログ）

ログの最後に”Demo completed successfully.”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。デモが正常終了すると AWS IoT Core に新たにモノが登録され、デバイス個別の証明書が付与されています。

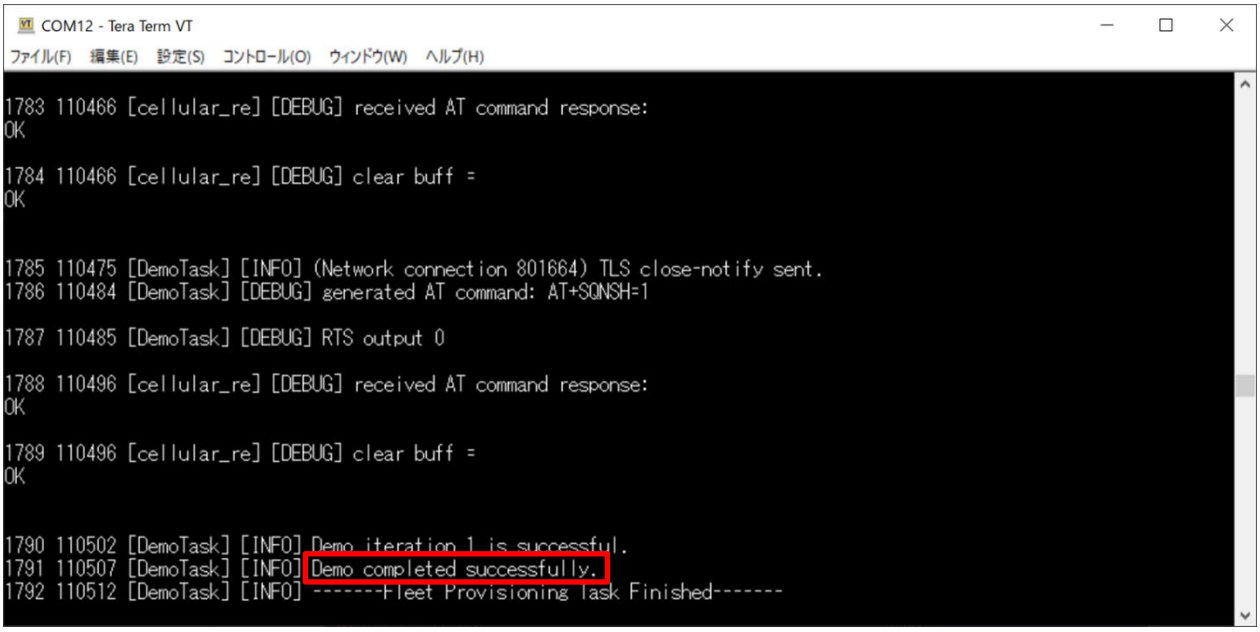

図 4-49 フリートプロビジョニングデモ成功時の実行ログ

また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。**図 4-50** で示すように Tera Term のログに”Successfully sent QoS 0 publish to topic:”が表示されることが確認できます。

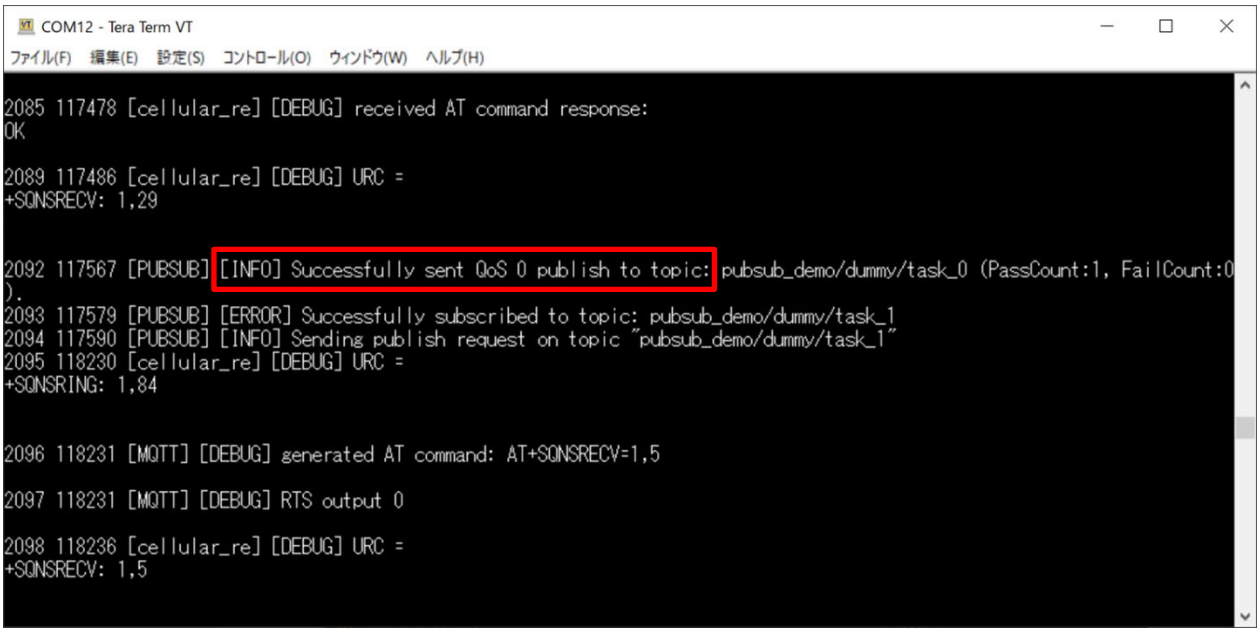

図 4-50 PubSub デモ成功時の実行ログ

AWS IoT Core の[MQTT テストクライアント]で CK-RX65N から AWS に送信された MQTT メッセージを確認できます。

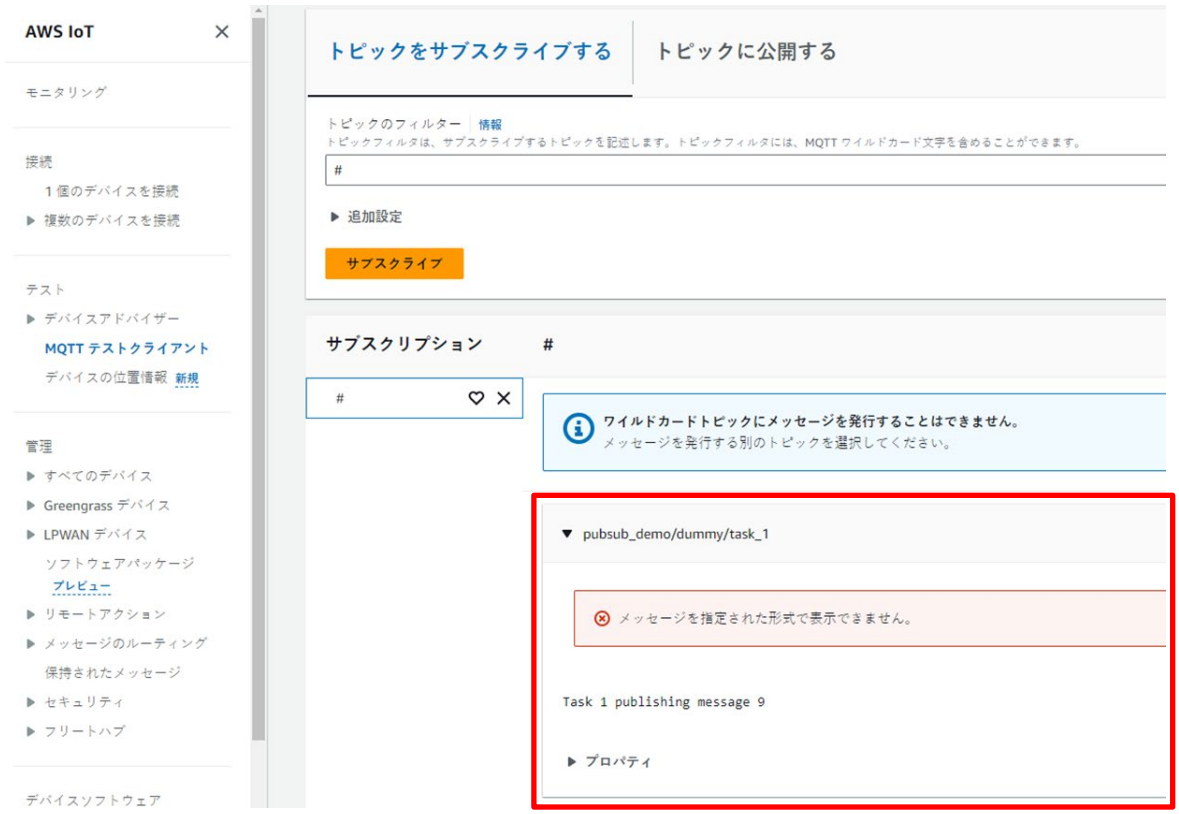

図 4-51 PubSub デモ成功時の MQTT テストクライアント

フリートプロビジョニングデモで登録されたモノは AWS コンソールから確認することができます。

「すべてのデバイス」からモノを選択します。
デモが正常終了している場合は、モノが登録されています。（図 4-52 内の
“FPDemoID_xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxxx”）

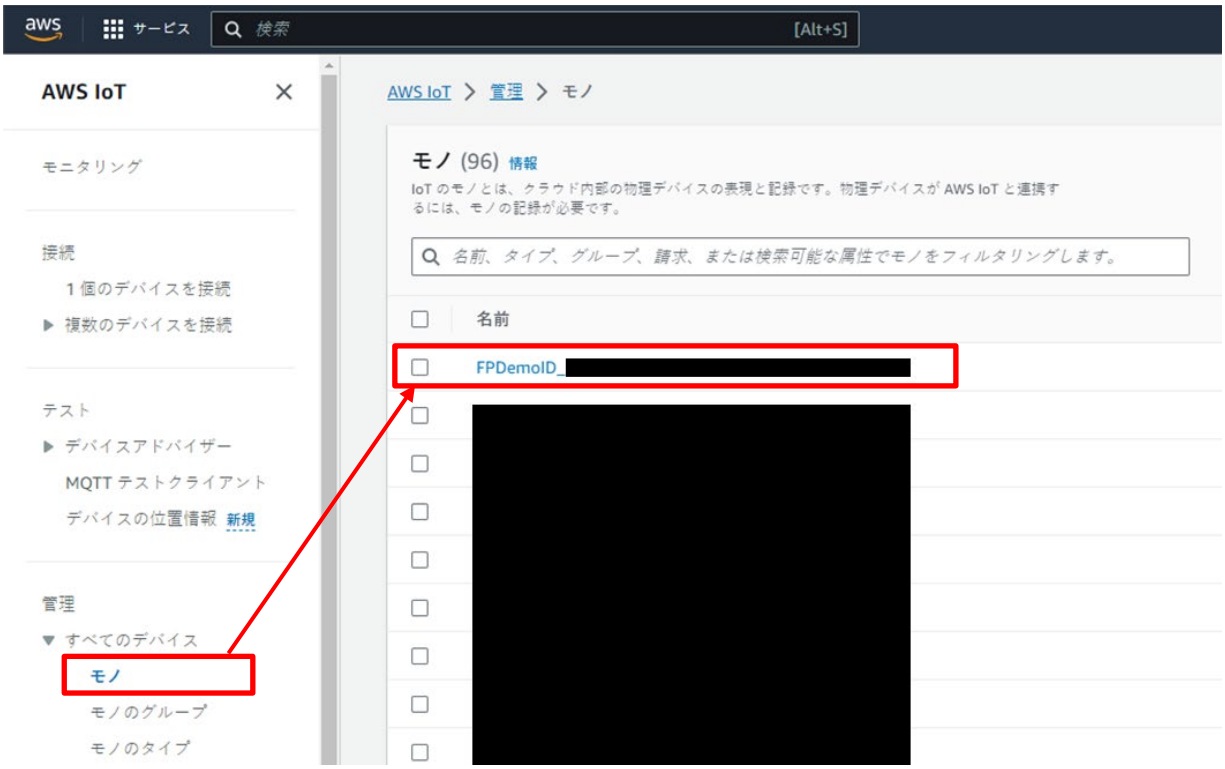

図 4-52 デモの実行結果確認(1)

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（図 4-53 内の “証明書 ID”））がアタッチ、有効化されていることが
確認できます。

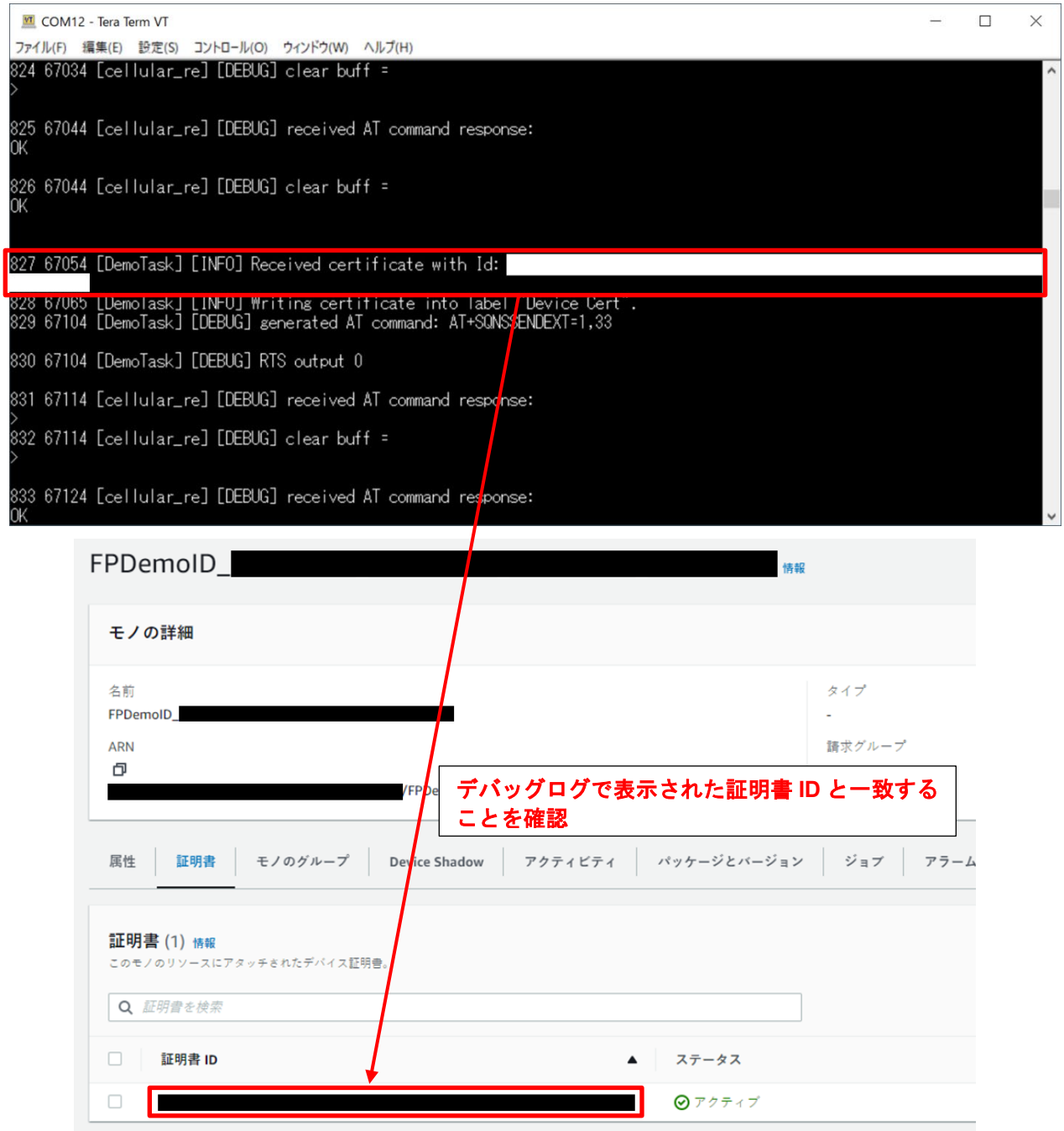

図 4-53 デモの実行結果確認(2)

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.scfg を開き、Components（コンポーネント）タグを選択します。
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

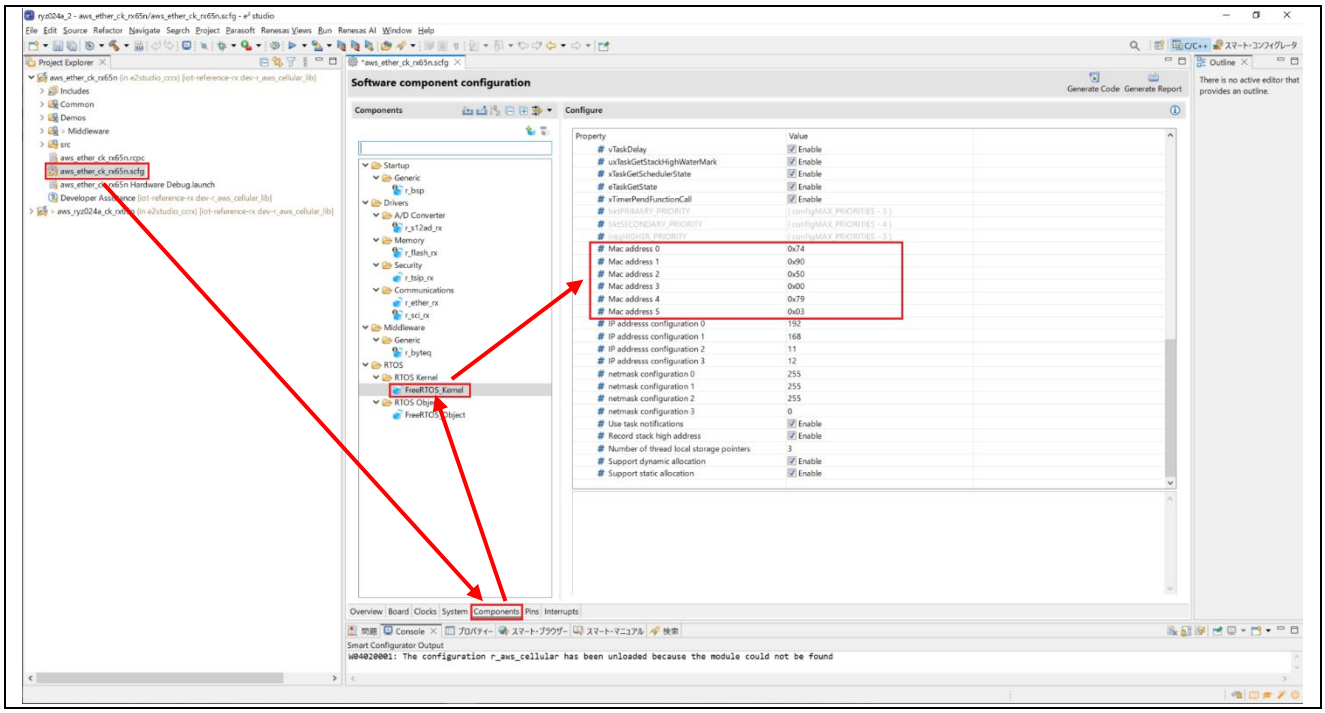

図 7-1 MAC アドレスの設定

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。

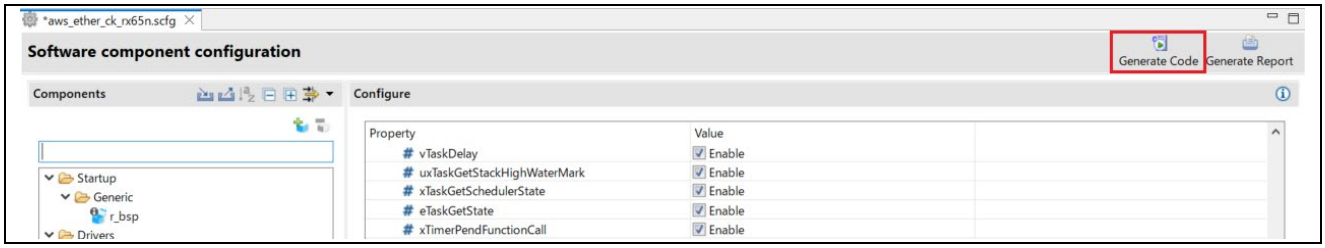

図 7-2 コードの生成

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2023/09/30	-	初版発行
1.10	2024/06/14	-	3.2 ソフトウェア環境の更新 3.3 Tera Term インストールと設定を更新 4.3.1 ポリシーの設定を一部更新 4.5 サンプルプロジェクトの生成を追加
			以下、余白

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。