RX ファミリ

TSIP ドライバを用いた TLS 実装による FreeRTOS OTA 実現方法

要旨

IoT デバイスでは、セキュリティの確保とリアルタイム処理が重要な要素となります。

このアプリケーションノートでは、RX ファミリに搭載されるセキュリティ H/W である Trusted Secure IP (TSIP)を使用し、TLS 通信の安全な鍵管理と、暗号化・復号処理の高速化について説明します。具体的な実装例とサンプルコードも提供しております。これにより、読者はより効率的で安全な IoT システムの構築が可能となります。

本アプリケーションノートでは FreeRTOS with IoT Libraries（AWS の IoT サービスを利用するための FreeRTOS およびライブラリ群）へ TSIP ドライバを組み込み、動作を確認しています。

これにより、TSIP ドライバを組み込んだ FreeRTOS with IoT Libraries 上で OTA デモアプリケーションを動作させることができます。

【注】 本アプリケーションノートでは FreeRTOS with IoT Libraries の RX 対応版であるデモプロジェクトの iot-reference-rx を使用します。
また、v202210.01-LTS-rx-1.3.0 以降の iot-reference-rx に対応します。

【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

動作確認デバイス

RX65N：R5F565NEHDF

参考ドキュメント

本アプリケーションノートでは OTA デモアプリケーションを実行します。
OTA に関する詳細の手順は以下のアプリケーションノートを参照してください。

- RX65N における Amazon Web Services を利用した FreeRTOS OTA の実現方法(v202210.01-LTS-rx-1.1.3 以降対応版)（R01AN7037）

TSIP を用いた TLS 通信や TSIP ドライバの API の詳細については以下のアプリケーションノートを参照してください。

- RX ファミリ TSIP(Trusted Secure IP)モジュール Firmware Integration Technology (R20AN0548)

ファームウェアアップデートの詳細は以下のアプリケーションノートを参照してください。

- 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)

目次

1. 概要 6
1.1 TSIP を用いた TLS 通信のメリット 6
1.2 TSIP を用いた TLS フロー 6
1.3 TSIP ドライバでサポートしている Cipher Suite 6
1.4 用語の定義 7
1.5 動作確認環境(ハードウェア) 8
1.6 動作確認環境(ソフトウェア) 8

2. 事前準備 9
2.1 Gpg4win（Kleopatra）のインストール 10
2.2 Renesas Key Wrap サービス・Kleopatra の初期設定 13
2.3 Cygwin のインストール 19
2.4 Security key Management Tool のインストール 20

3. AWS の設定 21
3.1 AWS コンソールにて必要な設定 21

4. デモプロジェクトの準備 22
4.1 ワークスペースの作成 24
4.2 デモプロジェクトのダウンロード 24
4.3 プロジェクトのインポート 27

5. 鍵と証明書の作成 31
5.1 TSIP 用鍵と証明書の準備 31
5.1.1 証明書・鍵の作成フロー 32
5.1.2 ルート CA 証明書の入手 33
5.1.3 RSA の鍵ペアとクライアント証明書の入手 34
5.1.4 ルート CA 証明書の署名生成 37
5.1.5 鍵のラップとプロジェクトへの登録 39
5.1.5.1 鍵のラップの概要 39
5.1.5.2 UFPK と W-UFPK の作成 41
5.1.5.3 鍵データのラップ 50
5.2 OTA 用鍵ペアと証明書の生成 63

6. プロジェクトの構築 64
6.1 初期バージョンのファームウェア構築と実行 64
6.1.1 プロジェクトのインポート 64
6.1.2 プロジェクト設定とビルド 65
6.1.3 初期ファームウェアの作成 75
6.1.4 初期ファームウェアの実行 81
6.1.5 AWS IoT 情報の登録 83
6.1.6 MQTT 通信状況の確認 89
6.2 更新用ファームウェア構築と実行 93

6.2.1 更新ファームウェアの作成 93
6.2.2 ファームウェアの更新 94

7. 付録 97
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 97

8. トラブルシューティング 98

改訂記録 100

【注】

- AWS™は Amazon.com, Inc. or its affiliates の商標です。(https://aws.amazon.com/trademark-guidelines)
- FreeRTOS™は Amazon Web Services, Inc.の商標です。(https://freertos.org/copyright.html)
- Git®は Software Freedom Conservancy, Inc.のトレードマークです。
  (https://www.git-scm.com/about/trademark)
- GitHub®は GitHub, Inc.のトレードマークです。(https://github.com/logos)
- Arm®は Arm Limited or its subsidiaries のトレードマークです。
  (https://www.arm.com/company/policies/trademarks/guidelines-trademarks)
- Mbed™は Arm Limited or its subsidiaries のトレードマークです。
  (https://www.arm.com/company/policies/trademarks/guidelines-trademarks)
- OpenSSL™は OpenSSL Software Foundation のトレードマークです。
  (https://www.openssl.org/policies/general/TrademarkPolicy.html)

1. 概要

1.1 TSIP を用いた TLS 通信のメリット

TSIP ドライバでは TLS 向けの API をサポートしています。本 API を利用することにより以下 2 点のメリットがあります。

- **メリット 1：TLS プロトコル処理中で平文の鍵情報を扱わないため、デバイス内に格納されたお客様の鍵情報の漏洩リスクを減らすことができます。**
- **メリット 2：暗号処理をハードウェアでアクセラレートすることにより、フル S/W 処理に比べ暗号処理を高速化できます。**

1.2 TSIP を用いた TLS フロー

以下に本デモプロジェクトにおける TLS のフローを示します。
このフローは鍵交換方式が RSA とした場合の例となります。

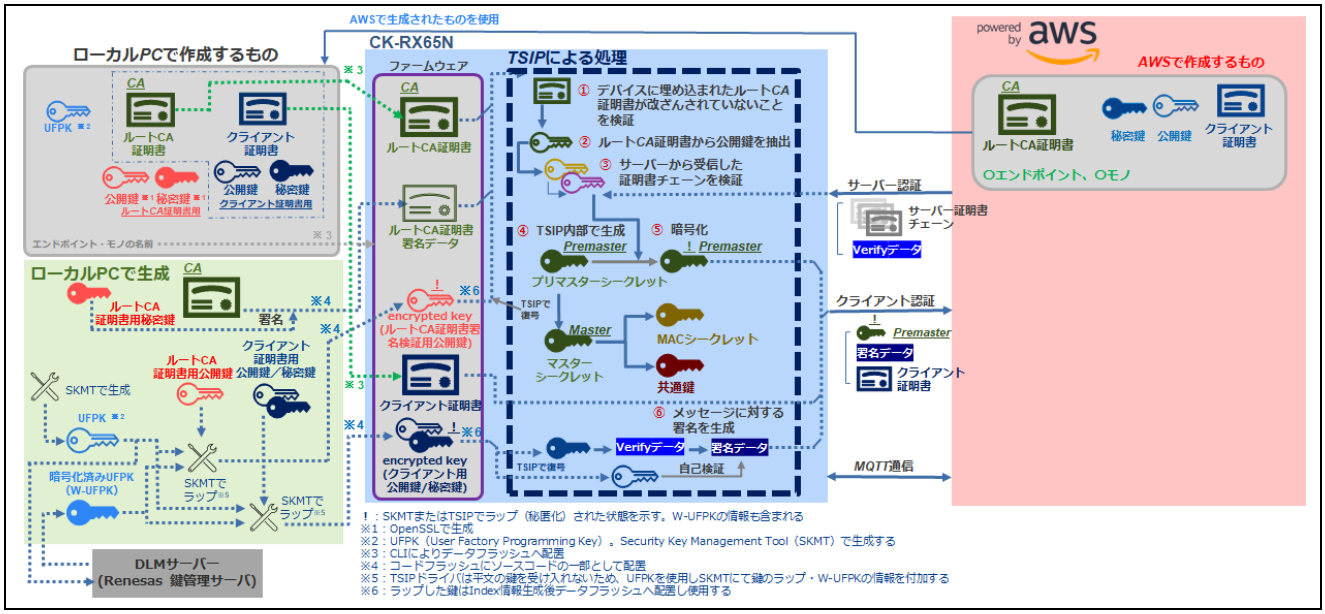


図 1-1 TSIP を用いた TLS フロー

1.3 TSIP ドライバでサポートしている Cipher Suite

TSIP ドライバは TLS1.2 に準拠した以下の Cipher Suite をサポートしています。

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

1.4 用語の定義

本アプリケーションノートで使用される用語の定義を以下に示します。
表に記載の鍵などの使用箇所は「図 1-1 TSIP を用いた TLS フロー」を参照してください。

表 1-1 用語

用語	内容
鍵注入	工場でデバイスに Wrapped Key を注入すること。
ユーザー鍵 (User key)	ユーザーが使用する平文状態の暗号鍵。デバイス上では使用しない。 鍵交換方式が RSA または ECC の場合は公開鍵と秘密鍵がそれぞれユーザー鍵となる。
Encrypted key	ユーザー鍵に UFPK による暗号化および MAC 値の付加をして生成される鍵情報。同一のユーザー鍵に対する Encrypted Key は 各デバイスで共通の値となる。
Wrapped Key	Encrypted Key を鍵の注入により TSIP で使用できる形式に変換したデータ。Wrapped Key は HUK でラップされているため、同一の Encrypted Key から変換した Wrapped Key はデバイスごとに固有の値となる。
UFPK (User Factory Programming Key)	鍵注入においてユーザー鍵から Encrypted Key を生成するために使用する、ユーザーが設定する鍵束。 デバイス上では使用しない。
W-UFPK (Wrapped UFPK)	UFPK を Renesas DLM サーバ上の HRK によりラップすることで生成される鍵情報。TSIP 内部にて HRK で UFPK に復号されて使用される。
Hardware Root Key (HRK)	TSIP 内部とルネサス内セキュアルームのみに存在する共通の暗号鍵
Hardware Unique Key (HUK)	TSIP 内部で導出する、鍵の保護のために使用するデバイス固有の暗号鍵
ラップ（ラッピング）	本アプリケーションノートでは Encrypted key を生成する過程で UFPK を使用して暗号化・MAC 付与を行うことをラップと呼ぶ。 TSIP ドライバは平文のユーザー鍵を入力として受け入れないため、暗号鍵はラップした状態で入力する必要がある。
Renesas DLM(Device Lifecycle Management)サーバ (https://dlm.renesas.com/)	Renesas Key Wrap サービスで使用される Renesas 鍵管理サーバ。UFPK のラップに使用する。

1.5 動作確認環境(ハードウェア)

本デモプロジェクトの動作確認環境(ハードウェア)を以下に示します。

表 1-2 動作確認環境(ハードウェア)

項目	詳細
使用ボード	CK-RX65N v1（Cellular / Ethernet）（注 1）
Cellular モジュール	RYZ014A PMOD モジュール（CK-RX65N v1 同梱）
SIM	LTE Cat-M1 対応 SIM（microSIM）（注 2）
デバッガ	E2 Lite エミュレータオンボード(CK-RX65N v1 上に搭載)

【注】 1. 本アプリケーションノートでは Cellular 通信を使用しています。

【注】 2. CK-RX65N v1 付属の SIM カードを使用する場合は、以下アプリケーションノートの 「4.1.5 Activating SIM card」を参照し、SIM カードのアクティベーションを行ってください。
SIM activation, Creating the trial account and using Dashboard with RYZ014A or Ethernet Application for AWS - Getting Started Guide (R01QS0064)

1.6 動作確認環境(ソフトウェア)

本デモプロジェクトの動作確認環境(ソフトウェア)を以下に示します。

表 1-3 動作確認環境(ソフトウェア)

項目	詳細
統合開発環境	e² studio 2024-04
コンパイラ	e² studio 用 RX コンパイラ CC-RX V3.06.00
FreeRTOS	v202210.01-LTS-rx-1.3.0
ドライバパッケージ（RDP）	RX Driver Package V1.42
TSIP ドライバ	RX ファミリ TSIP(Trusted Secure IP)モジュール Firmware Integration Technology Rev.1.20
ファームウェアアップデートモジュール	RX ファミリ ファームウェアアップデート モジュール Firmware Integration Technology Rev.2.03
ログモニタツール	Tera Term v4.106
Python 実行環境	Python 3.11.0
鍵生成ツール	Win64 OpenSSL v3.2.1
PGP 暗号化・復号ツール	Gpg4win（Kleopatra）v4.3.1
シェルスクリプト（bash）実行環境	Cygwin version 3.4.6
フラッシュ書き込みツール	Renesas Flash Programmer v3.14.00
Renesas Image Generator	Version3.03（ファームウェアアップデートモジュール Rev.2.03 同梱）
鍵作成ツール	Security Key Management Tool V.1.06

2. 事前準備

本アプリケーションノートでは、デモプロジェクトの実行のため、以下のソフトウェアツールを使用します。

表 2-1 使用するソフトウェアツール一覧

ソフトウェアツール名	用途
Tera Term	プログラムのシリアル動作ログを確認するために使用
Python	Renesas Image Generator プログラムを動作させるためのインタプリタとして使用
OpenSSL	TSIP 用の鍵変換、OTA 用の鍵生成に使用
Gpg4win（Kleopatra）	TSIP 用鍵のラップに使用する UFPK の PGP 暗号化に使用
Cygwin	Bash スクリプトを実行するために使用
Renesas Image Generator	OTA に使用するファームウェアイメージの作成に使用
Security Key Management Tool	TSIP 用鍵をラップするために使用

以下のソフトウェアツールは AWS IoT OTA と共通のものを使用します。インストール手順をアプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.事前準備」の章に記載しておりますので各ソフトウェアツールの項を参照してください。

- Tera Term：「2.1」節
- Python：「2.2」節
- OpenSSL：「2.3」節
- Renesas Image Generator：「2.4」節

また、「2.5」節にターゲットボード CK-RX65N v1 の接続についての説明を記載しています。この項の手順で CK-RX65N v1 の接続を行ってください。

2.1 Gpg4win（Kleopatra）のインストール

本アプリケーションノートでは、UFPK を PGP 暗号化／復号して W-UFPK を生成するための手順に Gpg4win（Kleopatra）を使用します。

以下の手順で Gpg4win 4.3.1 のインストールを行ってください。

(1) Gpg4win のダウンロード

Windows 用の GnuPG を扱っている下記のサイトにアクセスします。

https://www.gpg4win.org/

下図の Download ボタンをクリックします。

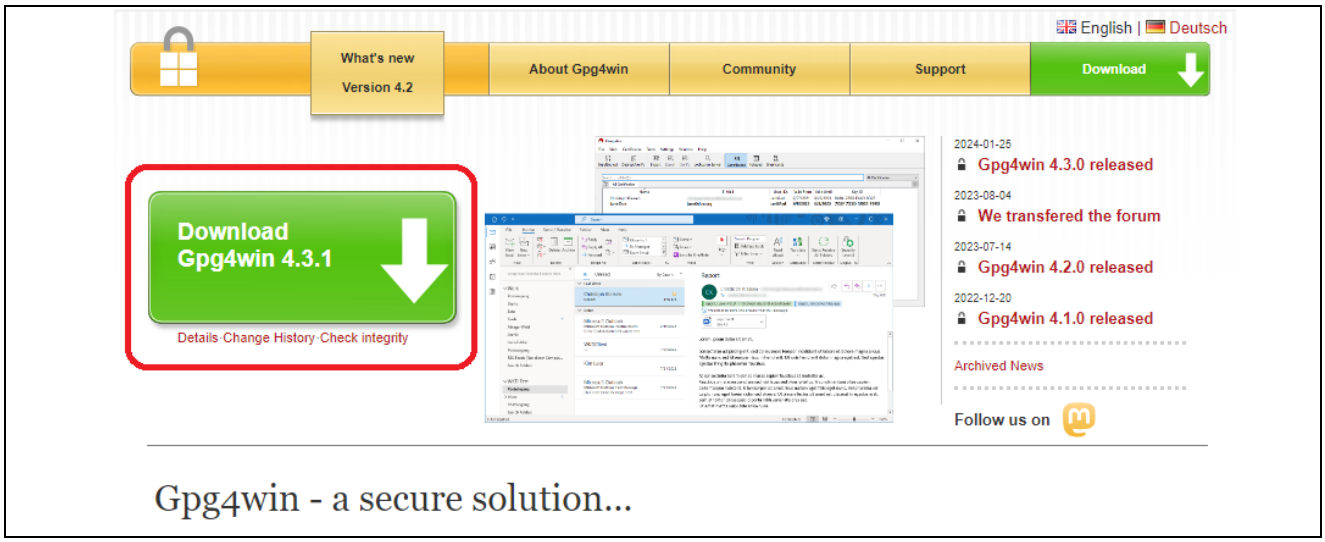


図 2-1 Gpg4win のダウンロード(1)

次の画面で「$0」を選択し、Download ボタンをクリックするとインストールファイルがダウンロードされます。

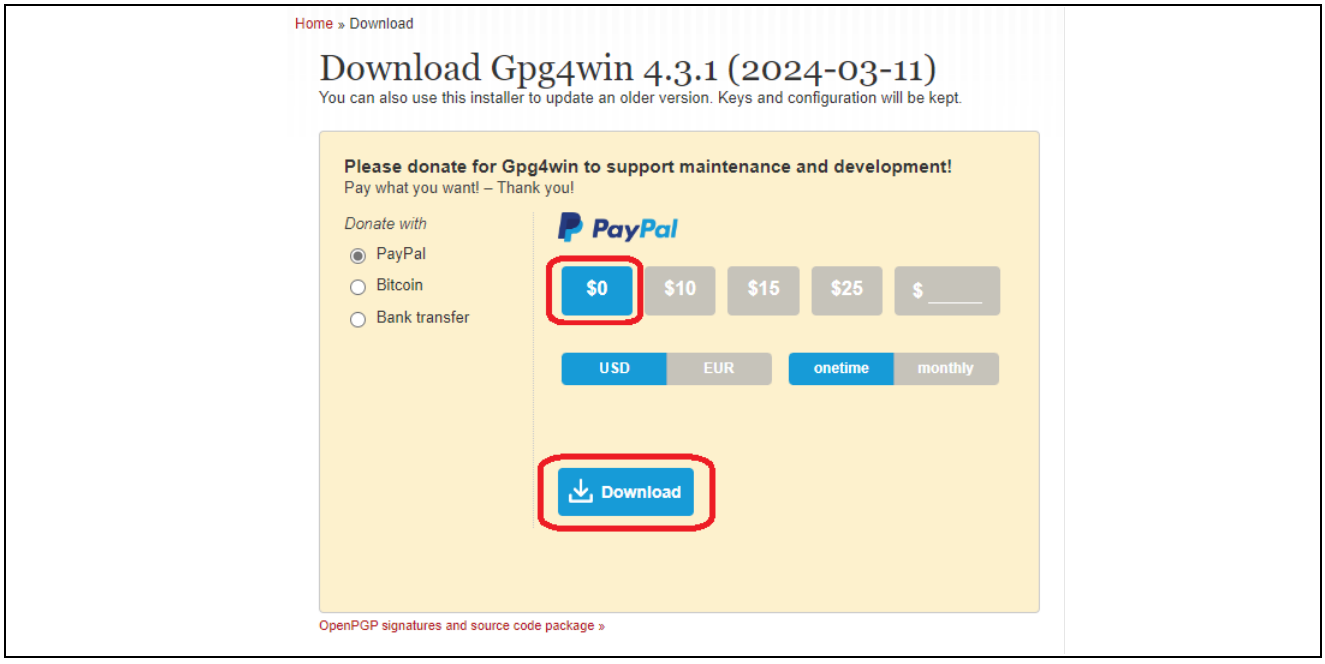


図 2-2 Gpg4win のダウンロード(2)

(2) Gpg4win のインストールを実行

「(1)」で入手したインストールファイルを Windows の「管理者として実行」で実行してください。インストーラが開始されます。[Next]ボタンをクリックすると以下のように言語の選択が表示されるので、使用する言語を選択して[OK]ボタンをクリックしてください。

図 2-3 Gpg4win インストール(1)

(3) コンポーネントの選択

以下のようにインストールするコンポーネントの選択画面が表示されるので、初期設定のまま[Next]をクリックします。

本アプリケーションノートでは鍵ペアを管理するツールである Kleopatra を使用します。以下画面で Kleopatra のチェックは入れたままでインストールを実行してください。

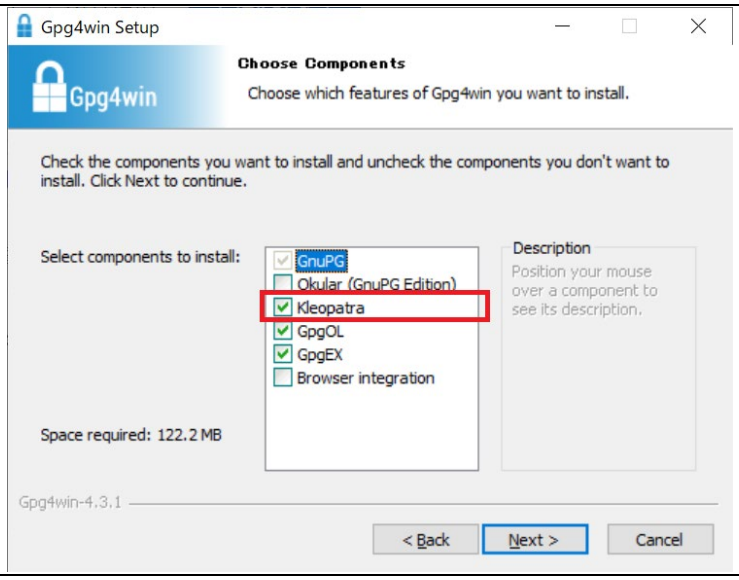


図 2-4 Gpg4win インストール(2)

(4) インストールの完了

[Next]ボタンを最後までクリックしてインストールを行ってください。

インストールが完了すると Kleopatra が実行されます。

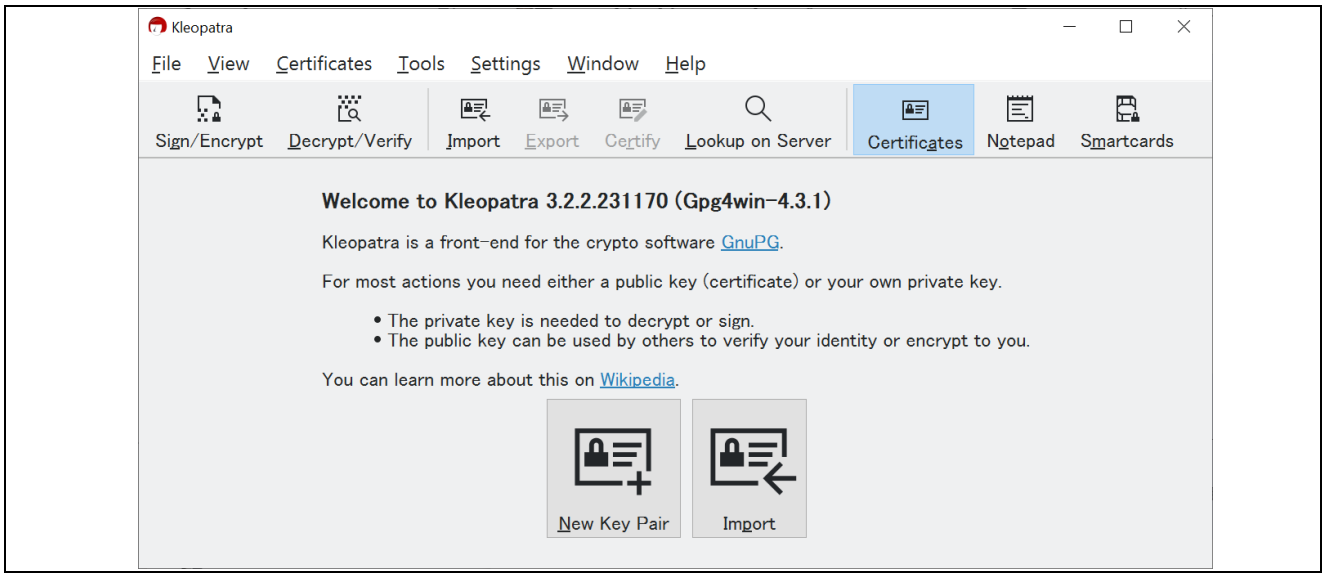


図 2-5 Kleopatra の実行画面

2.2 Renesas Key Wrap サービス・Kleopatra の初期設定

TSIP の処理ではユーザー鍵漏洩防止のため、平文のユーザー鍵が TSIP 外部に出ないような仕組みとしています。本アプリケーションノートでは Renesas Key Wrap サービスを使用して UFPK の暗号化（W-UFPK の生成）を行う処理を紹介しています。
本章では Renesas Key Wrap サービスの初期設定と、UFPK 暗号化の際に Renesas Key Wrap サービスとのファイル送受信に使用する PGP 鍵を、Kleopatra を使用して生成し登録する手順を説明します。

以下の手順で鍵の設定と登録を行ってください。本項で作成した情報は「5.1.5 鍵のラップとプロジェクトへの登録」の項で使用します。鍵の詳細についても「5.1.5」項を参照してください。

(1) Renesas Key Wrap サービスを登録する

Renesas Key Wrap サービスを初めて使用する際には、初回のみユーザー登録と PGP 鍵交換が必要です。以下の URL にログインして初回登録を実施してください。
Key Wrap サービス Login (renesas.com)

なお、Renesas Key Wrap サービスの詳細は以下の操作マニュアルに記載されています。
KeyWrap サービス システム操作マニュアル.pdf (renesas.com)

ユーザー登録と初回の PGP 鍵交換が完了したら Renesas Key Wrap サービスへログインしてください。

(2) OpenPGP 鍵ペアの作成

Renesas DLM サーバと公開鍵の交換するため、Kleopatra にて鍵ペアを作成します。

Kleopatra を起動したら、画面より[New Key pair（新しい鍵ペア）]ボタンをクリックします。「Create OpenPGP Certificate（OpenPGP 証明書を作成画面）」が表示されるので、「Name（名前）」と「Email address（メールアドレス）」を入力してください。また、「Protect the generated key with a passphrase（作成された鍵をパスフレーズで保護する）」にチェックを入れパスフレーズでセキュリティを強化することもできます。パスフレーズを設定した場合は入力したパスフレーズを忘れないようにしてください。

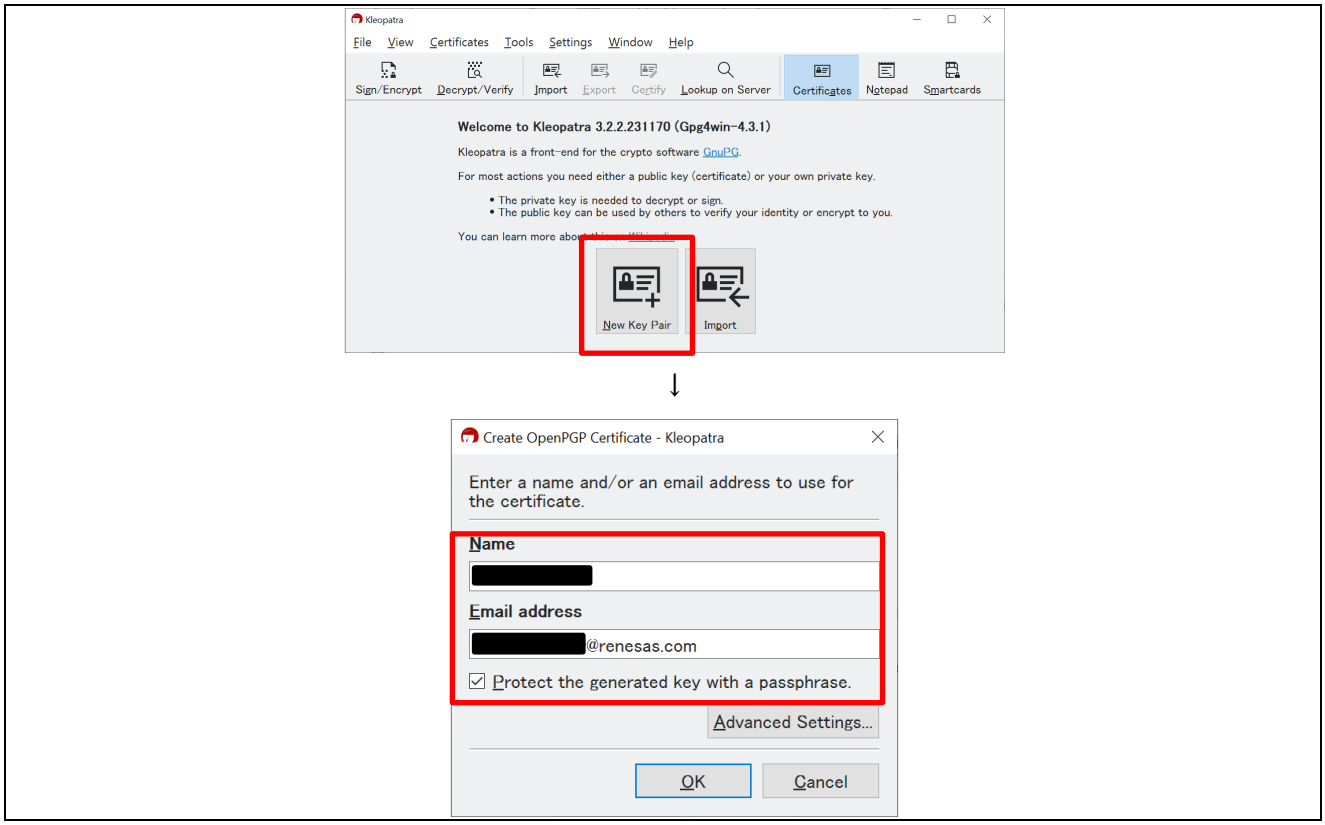


図 2-6 OpenPGP 鍵の作成(1)

次に[Advanced Settings（高度な設定）]ボタンをクリックします。

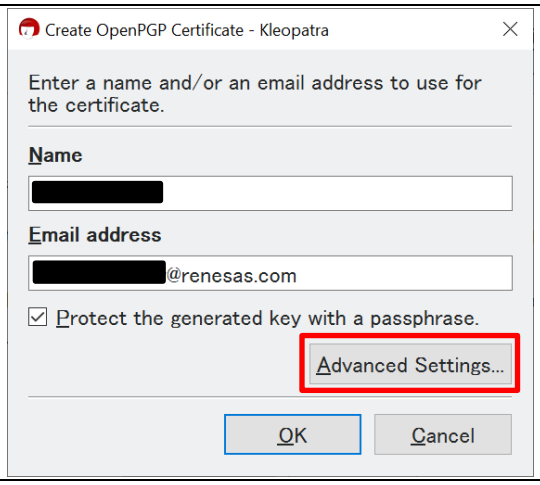


図 2-7 OpenPGP 鍵の作成(2)

「Advanced Settings（高度な設定画面）」が表示されるので、「Key Material（鍵マテリアル）」の項目に「RSA 4096 bits（RSA 4096 ビット）」を設定します。その他の設定は初期設定のままとしてください。
設定が完了したら[OK]ボタンをクリックします。

【注】 Renesas DLM サーバでは RSA 鍵のみ交換可能です。

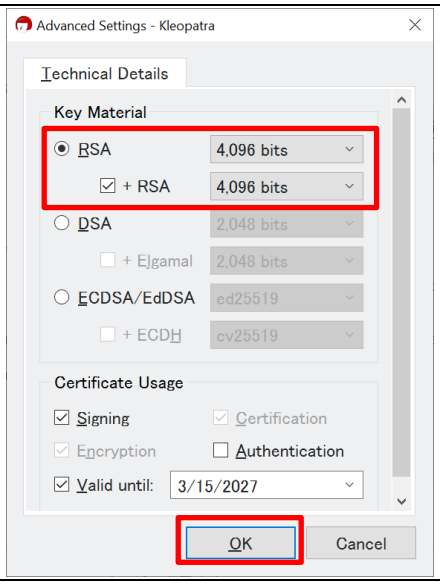


図 2-8 OpenPGP 鍵ペア-高度な設定

「Create OpenPGP Certificate（OpenPGP 証明書を作成画面）」に戻った後、[OK]ボタンをクリックすると以下のような画面が表示され、鍵ペアの生成が開始されます。鍵ペアの生成にはしばらく時間がかかります。

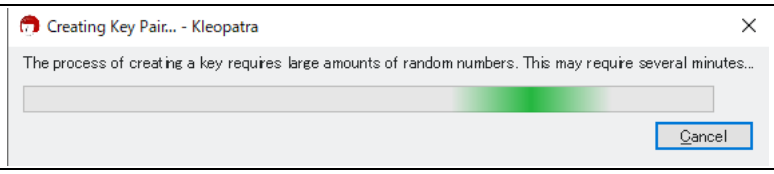


図 2-9 PGP 鍵ペアの生成

生成後、パスフレーズを設定している場合はパスフレーズの入力画面が表示されるので設定したパスフレーズを入力してください。

鍵ペアが作成出来たら、以下のような画面が表示されるので[OK]ボタンをクリックしてください。Kleopatra のメイン画面に戻ります。また、

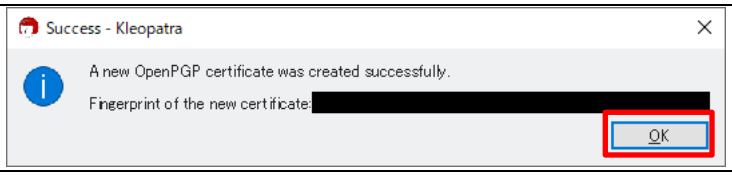


図 2-10 鍵ペアの作成

Kleopatra の画面に鍵ペアの情報が登録されます。登録した鍵ペアを選択し、[Export（エクスポート）]ボタンをクリックして、OpenPGP 公開鍵ペアを出力してください。ファイル名は、「～.asc」となります。

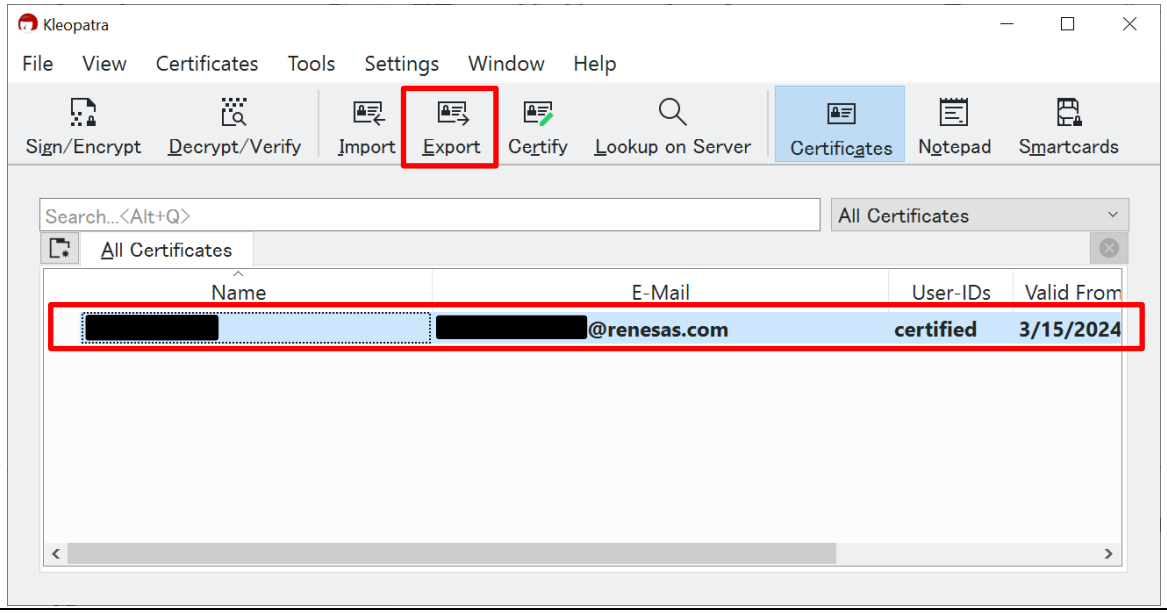


図 2-11 OpenPGP 鍵ペアの登録状態

(3) Renesas DLM サーバと PGP 公開鍵の交換

Renesas DLM サーバを使用し、「(2)」で出力した自分の OpenPGP 公開鍵とルネサスの PGP 公開鍵を交換します。「(1)」で登録したルネサス Key Wrap サービスの WEB サイトより[PGP key exchange（PGP 鍵交換）]をクリックし、作成した OpenPGP 公開鍵を登録します。
正常に登録が完了すると、ルネサスの PGP 公開鍵（keywrap-pub.key）が登録したメールアドレスに送られてくるので任意のフォルダに保存してください。

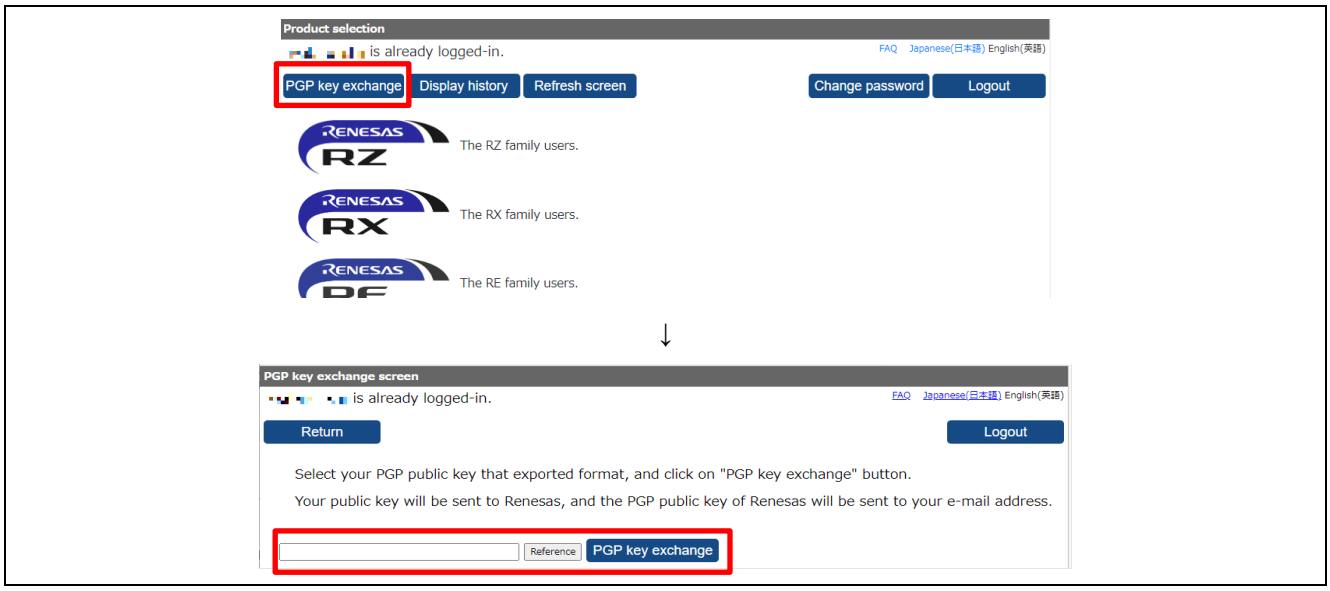


図 2-12 Renesas DLM サーバによる PGP 公開鍵交換

(4) ルネサスの OpenPGP 公開鍵の登録

ルネサス PGP 公開鍵は、Renesas DLM サーバで PGP 暗号化された鍵を復号するのに使用します。
Kleopatra にメールで受信したルネサスの PGP 公開鍵（keywrap-pub.key）を登録します。
Kleopatra のメニューより[インポート]をクリックします。

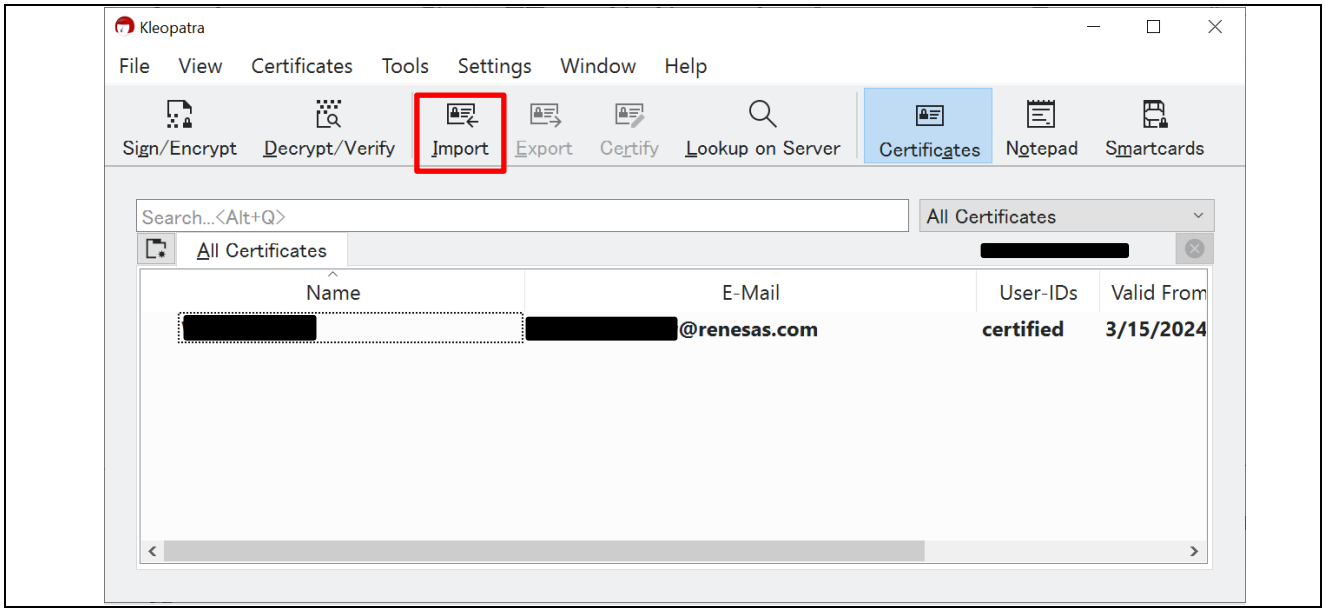


図 2-13 OpenPGP 公開鍵の登録

「証明書ファイルを選択画面」が表示されるので、ファイルの拡張子を「Any files(*)（任意のファイル(*)）」に設定し、ルネサスより送られてきた PGP 公開鍵「keywrap-pub.key」を指定して、[Open（開く）]ボタンをクリックしてください。

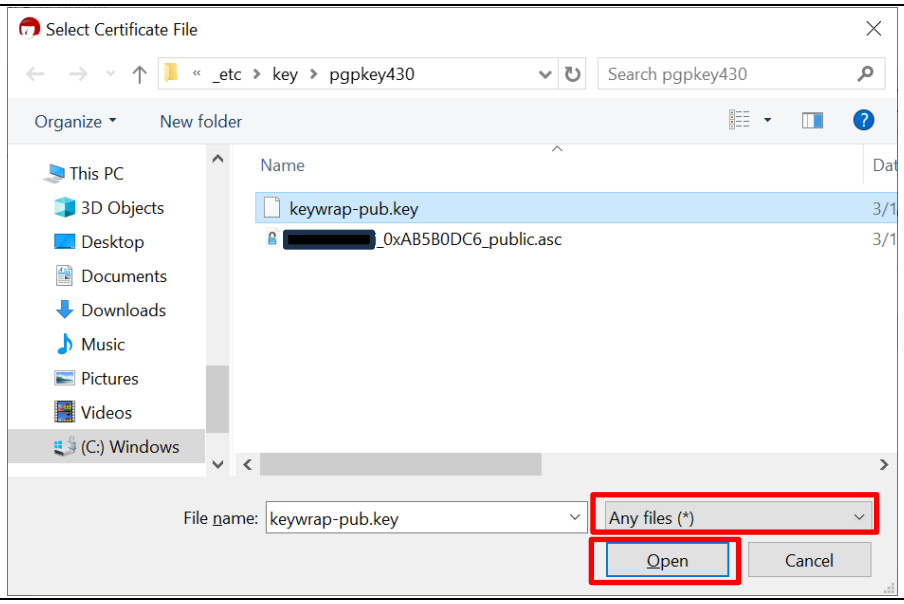


図 2-14 OpenPGP 公開鍵の指定

以下のように「You have imported a new certificate(publickey)（新しい証明書（公開鍵）をインポート）しました）」の画面が表示された場合は、[Certify（保証）]ボタンをクリックし、「Certify Certification（証明書を保証）」画面で自分が登録した証明書が選択されていることを確認してから「Certify（保証）」ボタンをクリックしてください。
「(2)」でパスフレーズを登録している場合はパスフレーズを入力します。

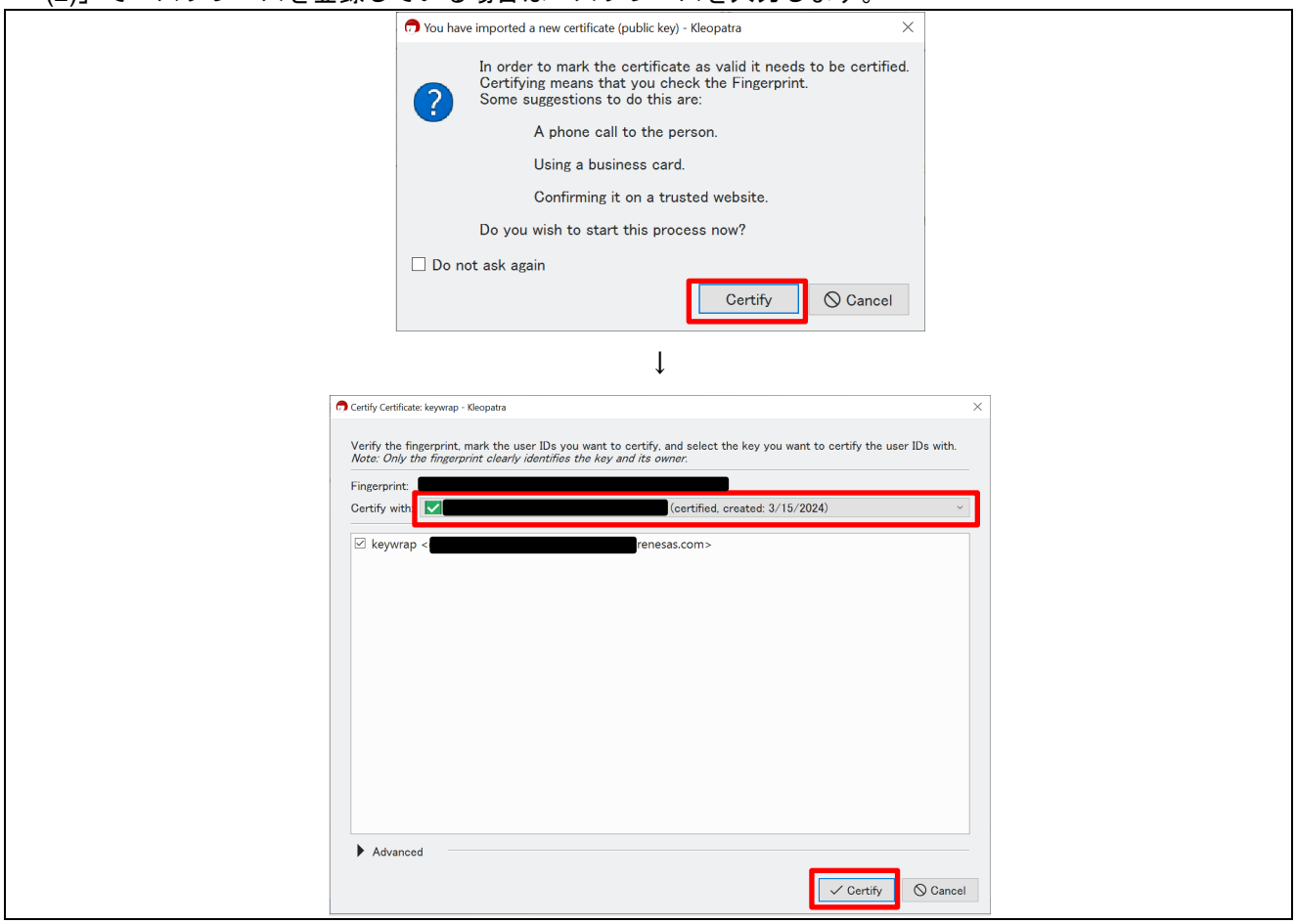


図 2-15 OpenPGP 公開鍵のインポート

「Certification successful（証明書に署名しました）画面が表示されるため、[OK]ボタンをクリックして画面を閉じてください。画面には以下のように「keywrap」が追加されます。「User-IDs（ユーザーID）」が「certified」になっていれば完了です。

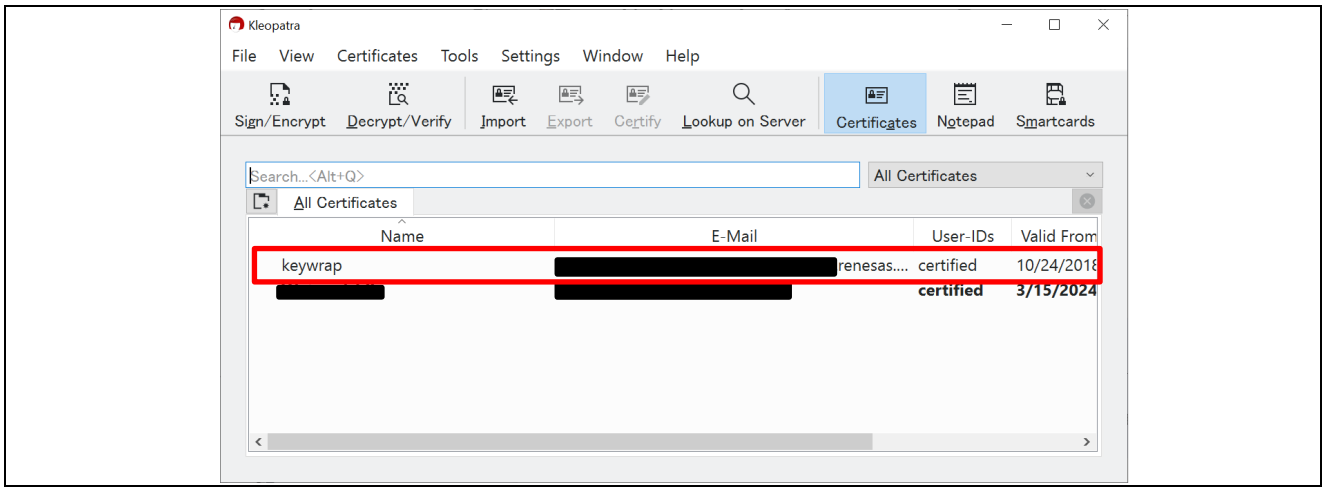


図 2-16 OpenPGP 公開鍵の登録状態

2.3 Cygwin のインストール

本アプリケーションノートでは、証明書・鍵をプロジェクト（ソースコード）へ登録を行うために Bash スクリプトを使用します。スクリプトの実行環境としては Cygwin を使用しています。

以下の手順で Cygwin のインストールを行ってください。

(1) Cygwin のダウンロードサイトにアクセスします。

Cygwin ダウンロードサイト

(2) ダウンロードサイトの以下のリンクをクリックし、インストーラをダウンロードします。

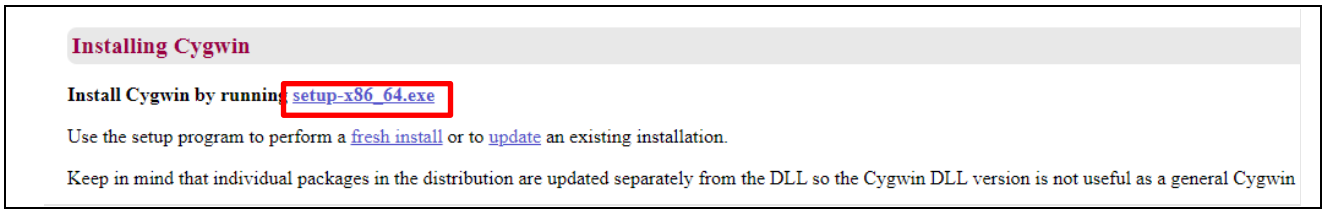


図 2-17 Cygwin のダウンロード

(3) インストーラを実行し、案内に沿って Cygwin をインストールします。

インストール時にパッケージの選択がありますが、基本は初期設定で実行してください。不足パッケージがある場合は別途導入してください。

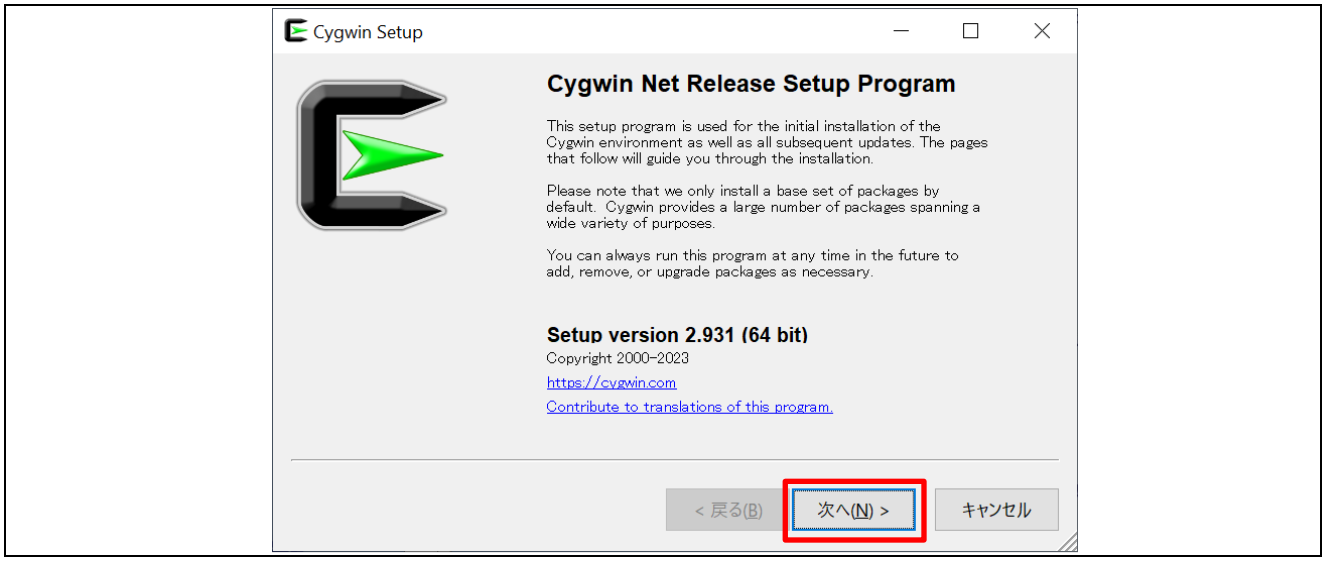


図 2-18 Cygwin のインストール

(4) スタートメニューから Cygwin65 Terminal のアイコンをクリックして、Cygwin のターミナル画面が表示されることを確認してください。

2.4 Security key Management Tool のインストール

本アプリケーションノートでは、Security key Management Tool を使用して各鍵情報を TSIP で使用できる状態へ変換します。

以下の手順で Security key Management Tool をインストールしてください。

(1) Security key Management tool のダウンロード

Security key Management tool のダウンロードサイトにアクセスし、最新版の Security Key management Tool の Windows 用をダウンロードしてください。

以下の画面は「Security key Management Tool V1.06」の場合の例です。

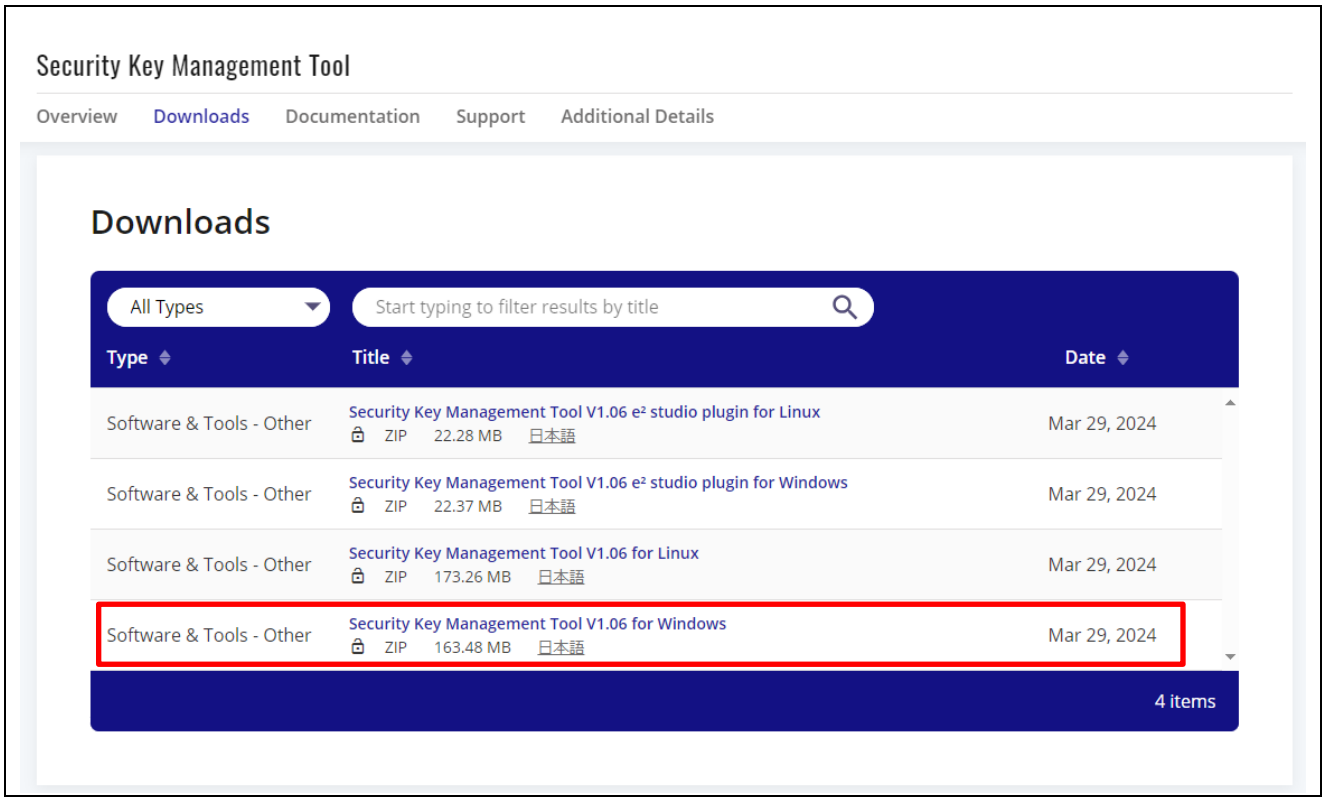


図 2-19 Security Key Management Tool のダウンロード

(2) ダウンロードが完了したらインストーラを実行し、案内に沿って Security Key Management Tool をインストールします。

(3) インストールが完了したら、スタートメニューから Security Key Management Tool を起動することを確認します。

3. AWS の設定

本アプリケーションノートの OTA デモの実行には、AWS に接続するためのアカウント（ルートユーザー、または AWS IoT と FreeRTOS クラウドサービスにアクセスできる権限を持つ IAM ユーザー）が必要です。

AWS の設定手順の詳細は、別途アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）に記載しております。また、以下の AWS ドキュメントも参考にすることができます。

- AWS のアカウントと権限の設定方法について
  https://docs.aws.amazon.com/freertos/latest/userguide/freertos-prereqs.html

また、本アプリケーションノートで実施するデモが AWS と通信できるようにするにはソースコードの修正が必要です。ソースコードの修正については「4.デモプロジェクトの準備」以降を参照してください。

3.1 AWS コンソールにて必要な設定

AWS コンソールにログインし、アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の 3 章の手順を実施して、AWS への初期設定と必要項目の設定を行ってください。

以下に本アプリケーションノートでプロジェクトのソースコード設定に必要な AWS の設定一覧を記載します。

表 3-1 AWS に必要な設定一覧

AWS での名称	内容	備考
Things (注 1) （モノ）	AWS へ接続するデバイス（モノ）の名前を登録します。	設定したモノの名前を記録 3.3.2(3)参照
Endpoint (注 1) （エンドポイント）	AWS の接続先（URL）を登録します。	設定したエンドポイント名を記録 3.3.3(1)
Device certificate （デバイス証明書）	AWS 接続に使用するクライアント証明書。 本アプリケーションノートでは「クライアント証明書」と呼称します。	AWS よりダウンロードし保存 (注 2) 3.3.2(6)
Public key file （パブリックキーファイル）	AWS 接続に使用する公開鍵です。 本アプリケーションノートでは「クライアント証明書用公開鍵」と呼称します。	AWS よりダウンロードし保存 (注 2) 3.3.2(6)
Private key file プライベートキーファイル）	AWS 接続に使用する秘密鍵です。 本アプリケーションノートでは「クライアント証明書用秘密鍵」と呼称します。	AWS よりダウンロードし保存 (注 2) 3.3.2(6)
Root CA certificate (注 3) （ルート CA 証明書）	AWS 接続に使用するルート CA 証明書です。	AWS よりダウンロードし保存

【注】 1. モノの名前・エンドポイント名はこの後実施するプロジェクトへ登録を行います。登録した名称を記録しておいてください。

【注】 2. クライアント証明書用公開鍵・秘密鍵は AWS へのデバイス登録時にのみダウンロードできるため注意してください。

【注】 3. ルート CA 証明書のダウンロードは、アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）では実施しません。本アプリケーションノートの「5.1.2」項にてダウンロード手順を説明します。

4. デモプロジェクトの準備

本章ではデモに使用するプロジェクトの作成方法をガイドします。

本アプリケーションノートで説明の CK-RX65N v1 は Cellular 版となります。付属の RYZ014A ボードを CK-RX65N v1 の PMOD1 端子に接続し、モバイルネットワークで接続します。
接続についてはアプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.5」節を参照してください。

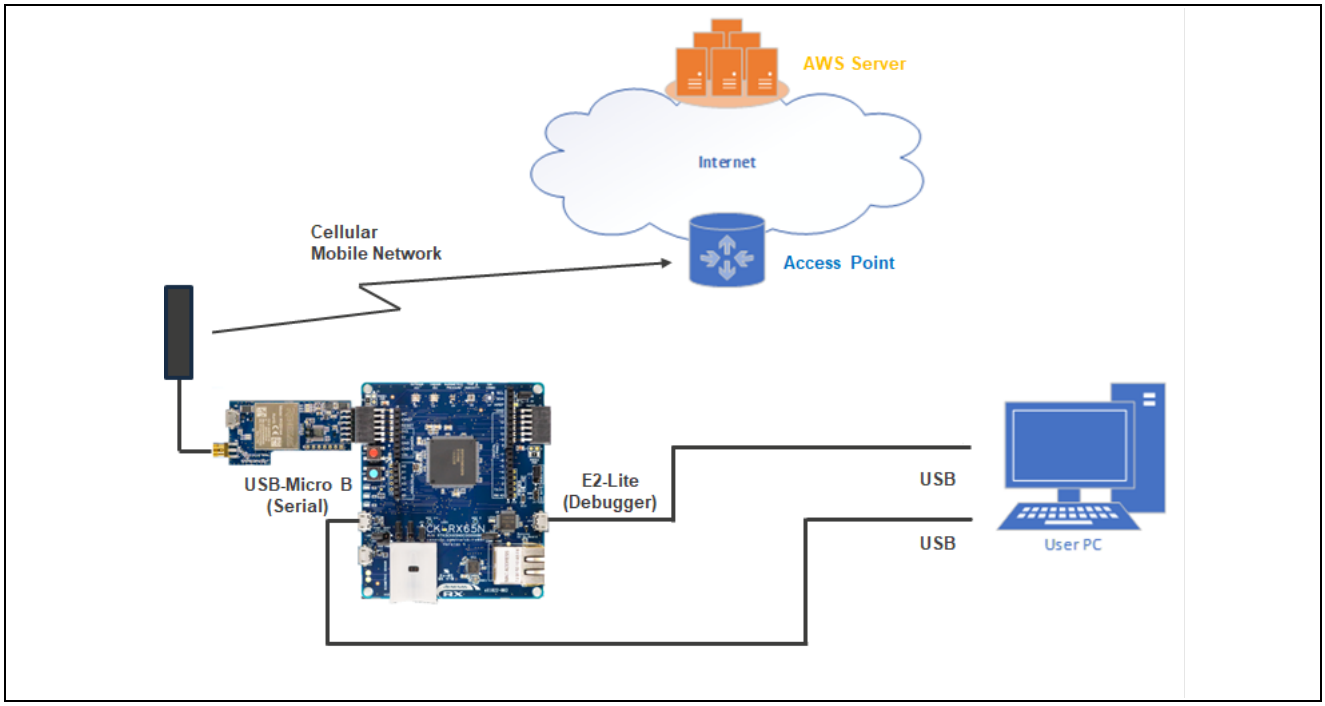


図 4-1　デモプロジェクトの接続関係

デモプロジェクトは FreeRTOS のプロジェクトをベースにしています。FreeRTOS は IoT 機器向けに必要なソースコードをまとめた IoT Libraries を提供しており、この中で暗号ライブラリとしてオープンソースの Mbed TLS を使用しています。

本デモプロジェクトでは Mbed TLS ライブラリの一部の処理を TSIP ドライバの TLS 向け API で置き換えています。以下にデモプロジェクトのソフトウェア構造を示します。

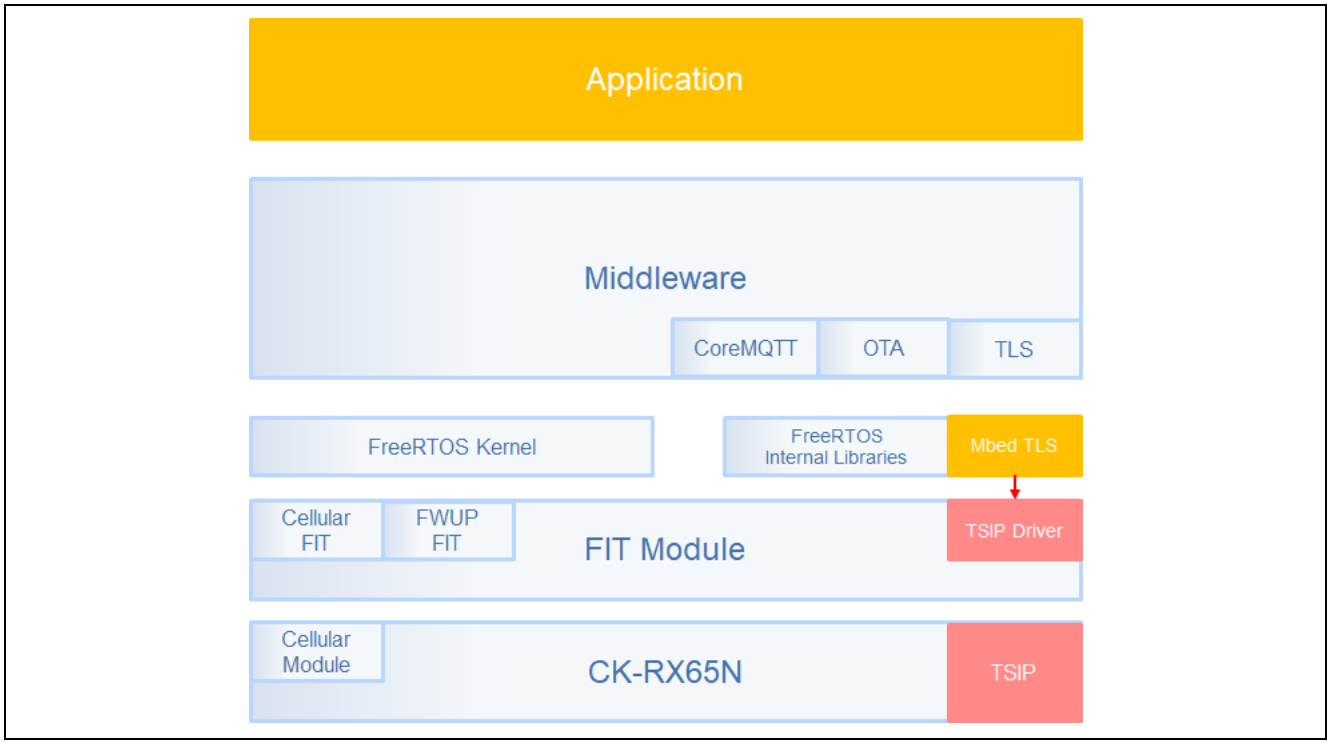


図 4-2 デモプロジェクトのソフトウェア構造

本デモプロジェクトは、以下の GitHub リポジトリにある RX 向けの FreeRTOS プロジェクトの v1.3.0 を使用します。

https://github.com/renesas/iot-reference-rx

4.1 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。
e² studio に制限があるため、ワークスペースのパス長（任意のフォルダ名を含む）は 35 文字以内としてください。36 文字以上を指定するとプロジェクトのビルド時にエラーとなります。また、パスには ASCII 文字のみ含めて入力してください。

【例】C:\workspace を新規ワークスペースとして作成する場合

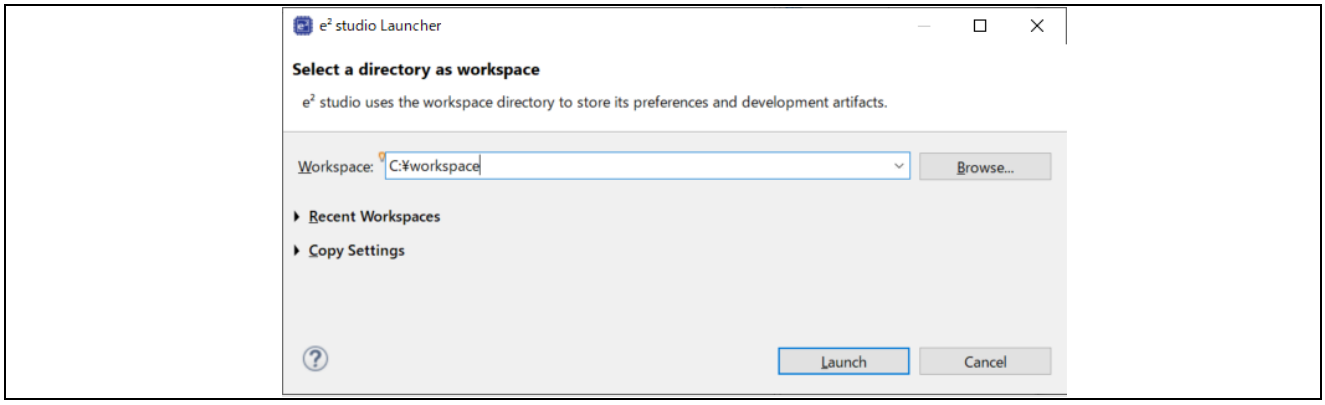


図 4-3 ワークスペース作成画面

4.2 デモプロジェクトのダウンロード

(1) デモプロジェクトのクローン

GitHub (iot-reference-rx：FreeRTOS reference repository) からデモプロジェクトをクローンします。本ドキュメントでは、Git for Windows を使用した場合のクローン方法を説明します。

GitBush を起動し、以下のコマンドを実行してください。

```
cd c:\workspace
git clone https://github.com/renesas/iot-reference-rx --recursive
```

図 4-4 デモプロジェクトのクローン

【注】 e² studio に制限があるため、クローン先のパス長（任意のフォルダ名を含む）は 35 文字以内としてください。36 文字以上を指定するとプロジェクトのビルド時にエラーとなります。
上記例では、C:\workspace にクローンしています。

(2) フォルダ構成

GitHub よりダウンロードしたデモプロジェクトは以下のようなフォルダ構成となっています。以下は主なフォルダを記載しています。

また、TSIP ドライバを使用するため、標準プロジェクトから追加・変更を行っているファイル・フォルダを赤字で示します。詳細は diff ツールなどを使用して確認してください。

```
iot-reference-rx
|--Common
|  |--common_api/r_common_api_tsip.c/h
|--Demos
|  |--key_flash_wr_with_tsip
|--IDT_config
|--Middleware
|  |--mbedtls_config/aws_mbedtls_config_with_tsip.h
|  |--mbedtls_with_TSIP
|  |--network_transport/using_mbedtls_pkcs11_with_tsip
|--Project
|  |--aws_ether_tsip_ck_rx65n
|  |--aws_ryz014a_tsip_ck_rx65n
|  |  |--e2studio_ccrx
|  |  |  |--src
|  |  |     |--application_code/main.c
|  |  |     |--frtos_startup/freertos_start.c
|  |  |     |--userdata_tsip
|  |  |--flash_project
|  |  |--key_crt_sig_generator
|--boot_loader_ck_rx65n
|  |--e2studio_ccrx
|  |  |--src
|--Test
|--Tools
|--Getting_Started_Guide.md
|--README.md
```

図 4-5 デモプロジェクトフォルダ構成

本プロジェクトでは TSIP ドライバを使用するために標準プロジェクトから以下のような変更を実施しています。

- FreeRTOS、Mbed TLS の一部の処理を TSIP ドライバの API で置き換え。それに伴う各コードの追加
- マルチタスク環境での TSIP ドライバへのアクセス衝突を回避するために、排他制御を追加
- 証明書とその署名を記述するファイル・フォルダを新たに追加
- 追加した鍵データのデータフラッシュ書き込み用処理の追加
- TSIP ドライバ接続のためのユーザー鍵・証明書の設定用ファイル格納フォルダを新たに追加

ダウンロードした各フォルダにはデモプロジェクトのソースコードのほか、デモプロジェクトを構築するためのツール類も格納されています。各フォルダの概要は以下の通りです。

表 4-1 デモプロジェクトの内容

フォルダ名	内容
Common Demos IDT_config Middleware Test Tool	各プロジェクトで使用される共通コード、ライブラリなどのモジュールが格納されています。 必要に応じてプロジェクトでリンクされます。
aws_ryz014a_tsip_ck_rx65n	本アプリケーションノートで使用する、TSIP に対応する Cellular 接続版のプロジェクトのフォルダです。 次項以降の手順で本フォルダをインポートして使用します。
aws_ether_tsip_ck_rx65n	TSIP に対応する Ethernet 接続版のプロジェクトです。 Cellular 版と同様の手順で使用することが可能です。
boot_loader_ck_rx65n	本アプリケーションノートで使用する、CK-RX65N v1 用のブートローダプロジェクトです。 次項以降の手順で本フォルダをインポートして使用します。
flash_project	OTA 実施時に CK-RX65N へ実行ファイルを書き込むために使用する Renesas Flash Programmer のプロジェクトファイルです。
key_crt_sig_generator	暗号化に使用する鍵、証明書を生成するツールと作業フォルダを格納しています。

4.3 プロジェクトのインポート

以下の手順でダウンロードしたプロジェクトをインポートします。

(1) e² studio を起動

(2) [File（ファイル）] ⇒ [Import（インポート）] を選択し、インポートダイアログを起ち上げてください。

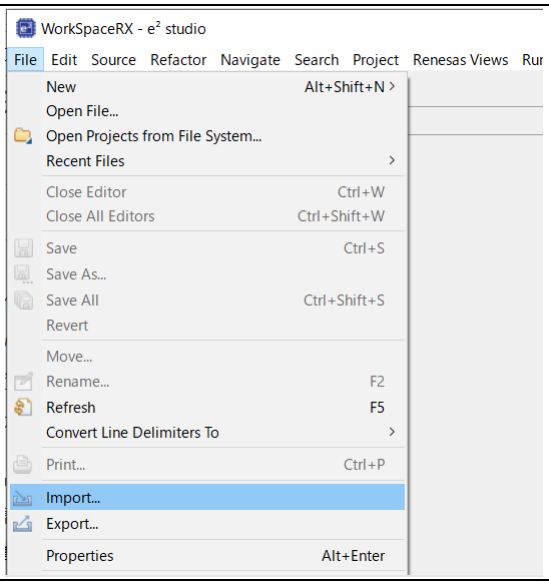


図 4-6 インポートダイアログボックスを開く

(3) 「Existing Projects into Workspace (既存プロジェクトをワークスペースへ)」を選択して、[Next（次へ）]ボタンをクリックしてください。

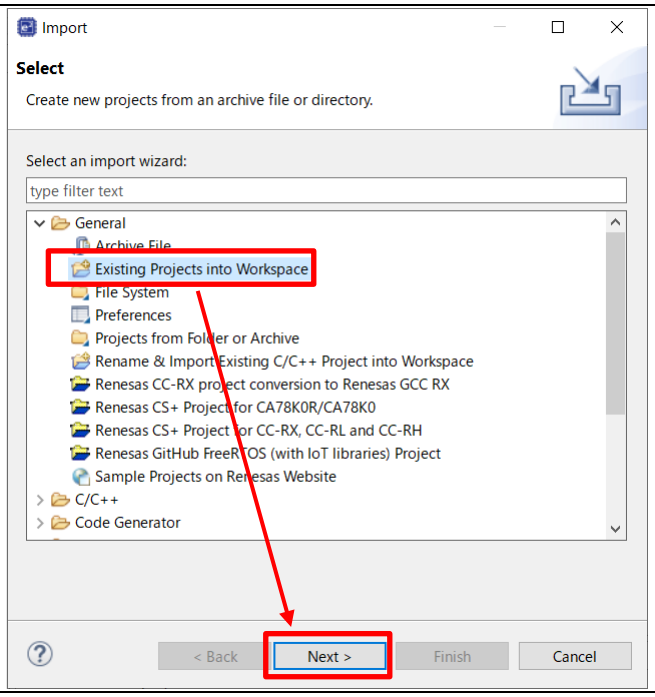


図 4-7 既存プロジェクトをワークスペースへインポート

(4) Select root directory で「4.2(1)」にてクローンしたフォルダを選択し、以下の 2 つのプロジェクトにチェックして[Finish（終了）]ボタンをクリックしてください。

- aws_ryz014a_tsip_ck_rx65
- boot_loader_ck_rx65n

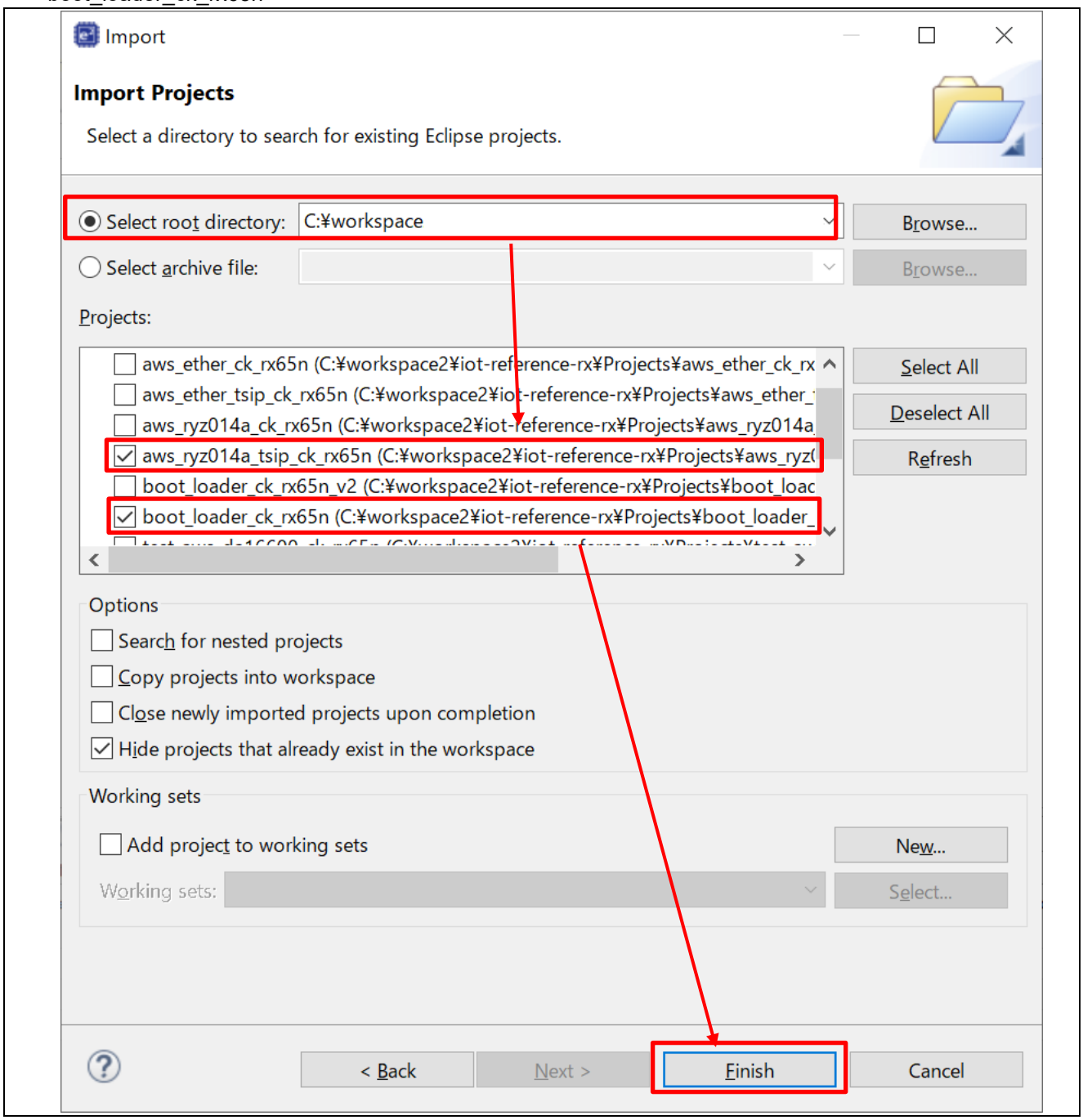


図 4-8 ブートローダとアプリケーションのプロジェクトをインポート

(5) プロジェクトが正常にインポートされると以下のようにプロジェクト・エクスプローラーにインポートした「aws_ryz014a_tsip_ck_rx65」と「boot_loader_ck_rx65n」が追加されます。
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++]をクリックしてから、[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。

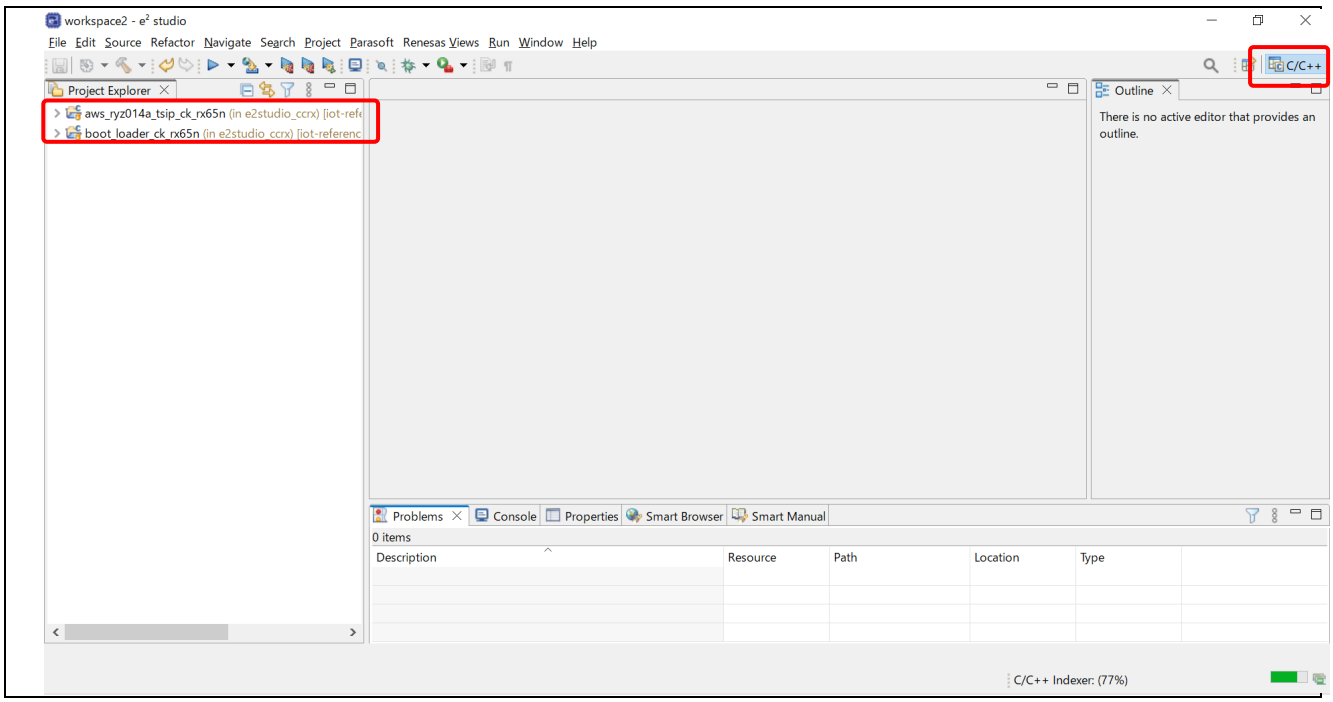


図 4-9 プロジェクトインポート後の画面

(6) プロジェクト環境設定の確認

インポートした 2 つのプロジェクトについて、メニューより [Project（プロジェクト）] > [Properties（プロパティ）] > [C/C++Build（C/C++ビルド）] > [Settings（設定）]から「Toolchain（ツールチェーン）タブをクリックし、ツールチェーンの設定が「Renesas CC-RX」になっていることを確認します。

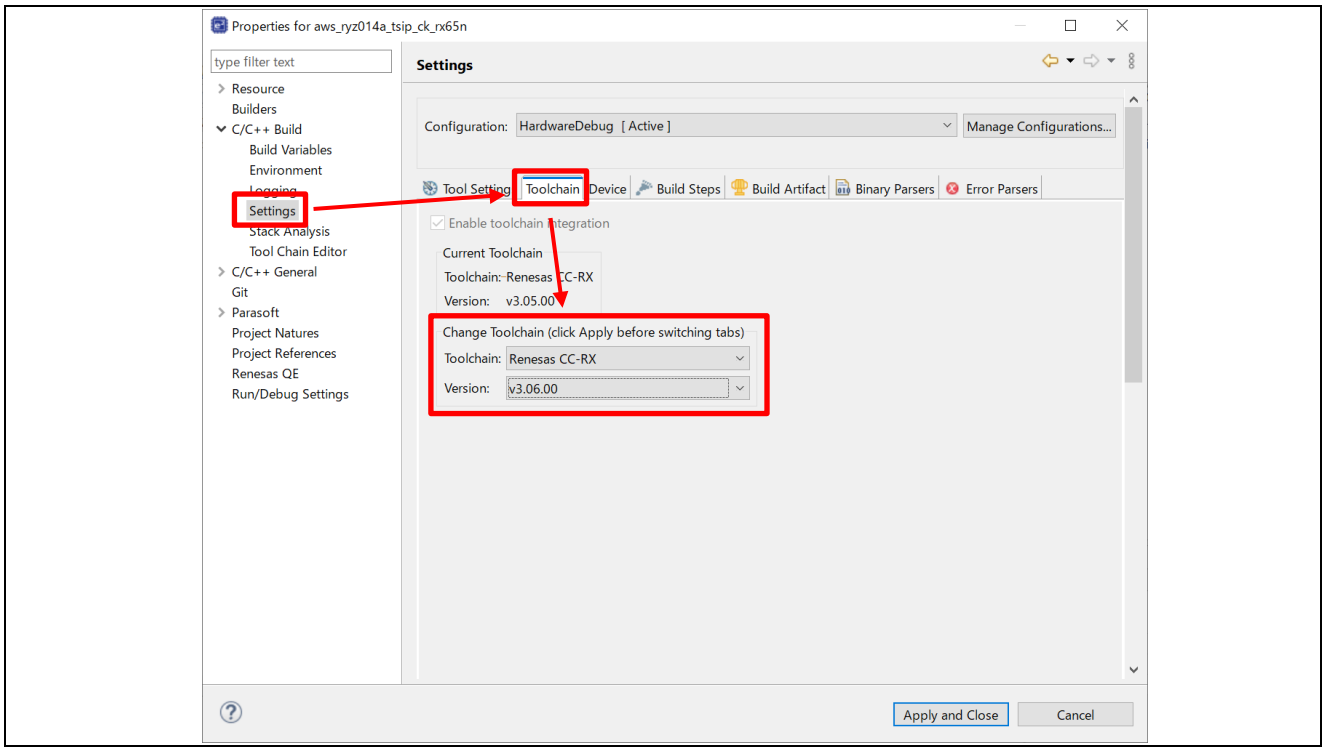


図 4-10 ツールチェーンの確認

続いて、Tool Settings（ツール設定）タブを選択し、[Converter] > [Output（出力）]から [Motorola S format file（モトローラ S 形式ファイルを出力する）]が選択されていることを確認してください。

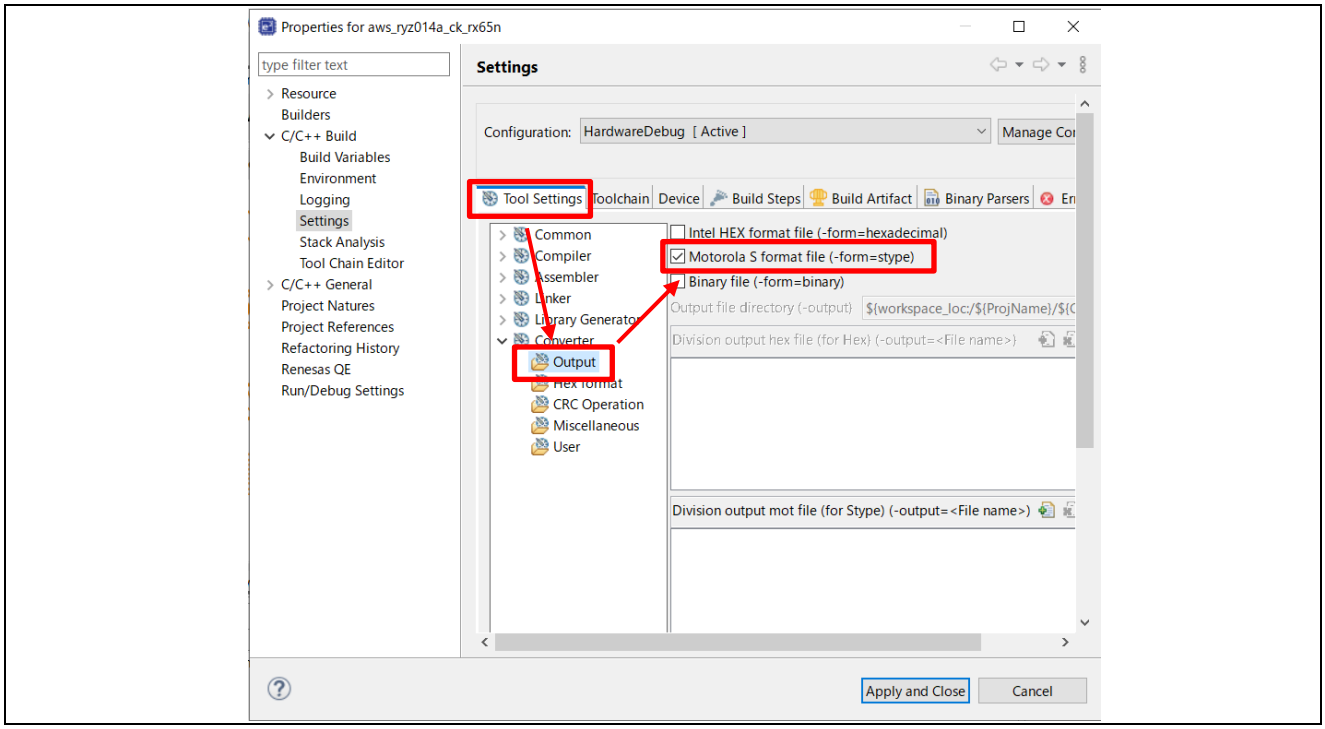


図 4-11 出力形式の確認

5. 鍵と証明書の作成

本プロジェクトでは、TSIP による TLS 接続および、OTA の実行のために複数の鍵・証明書を作成する必要があります。

以下の手順で鍵・証明書データを作成し、「4.デモプロジェクトの準備」で e² studio にインポートしたサンプルプルジェクトへデータを登録します。

5.1 TSIP 用鍵と証明書の準備

本サンプルプログラムで TSIP を使用した TLS 通信を実施するためには「表 5-1」に示す、鍵・証明書の情報を登録する必要があります。
本節では、これらの鍵と証明書の入手方法と TSIP で使用するための変換手順および、プロジェクトへの登録手順について説明します。
手順に従って各鍵と証明書の情報を生成し、プロジェクトへ登録を行ってください。
この表は必要な情報と入手方法の概要を示しています。「表 5-1」で示した証明書・鍵の作成フローについては「図 5-1 TSIP 用鍵・証明書の作成フロー」を参照ください。

表 5-1 プロジェクトで使用する鍵と証明書の入手方法

鍵/証明書	入手方法	組み込み方法
ルート CA 証明書	AWS からダウンロードする。	CLI で組み込む (注 1)(注 2)
ルート CA 証明書署名データ	OpenSSL などのツールを使ってユーザーが作成する。(注 6)	ファイルをプロジェクトの指定フォルダに配置し組み込む
ルート CA 証明書用公開鍵	OpenSSL などのツールを使ってユーザーが作成する。(注 6) 作成データをユーザーがラップする。(注 4) ラップ後はルート CA 証明書署名検証用公開鍵として使用する。	ファイルをプロジェクトの指定フォルダに配置しに組み込む
クライアント証明書	AWS からデバイス登録時にダウンロードする。(注 5)	CLI で組み込む (注 1)(注 3)
クライアント証明書用公開鍵	AWS からデバイス登録時にダウンロードする。(注 5) ダウンロードしたデータをユーザーがラップする。(注 4)	ファイルをプロジェクトの指定フォルダに配置しに組み込む
クライアント証明書用秘密鍵	AWS からデバイス登録時にダウンロードする。(注 5) ダウンロードしたデータをユーザーがラップする。(注 4)	ファイルをプロジェクトの指定フォルダに配置しに組み込む

【注】 1. 「CLI」は本プロジェクトが持っているコマンドラインインターフェースの機能名称です。
詳細は「6.1.5」項を参照してください
【注】 2. CLI による組み込み方法については「6.1.5(3)」項を参照してください
【注】 3. CLI による組み込み方法については「6.1.5(2)」項を参照してください
【注】 4. ラップの手順については「5.1.5」項以降で説明します。
【注】 5. ダウンロード手順については「3.1」節を参照してください
【注】 6. OpenSSL を用いた作成方法は「5.1.4」項を参照してください。

5.1.1 証明書・鍵の作成フロー

「図 1-1 TSIP を用いた TLS フロー」で示したフローより、本アプリケーションノートで実施する証明書と鍵の入手フロー部分を抜き出した詳細を示します。

以下が「表 5-1」で示した証明書・鍵のうち、本アプリケーションノートの手順で使用する部分の作成フローを以下に示します。

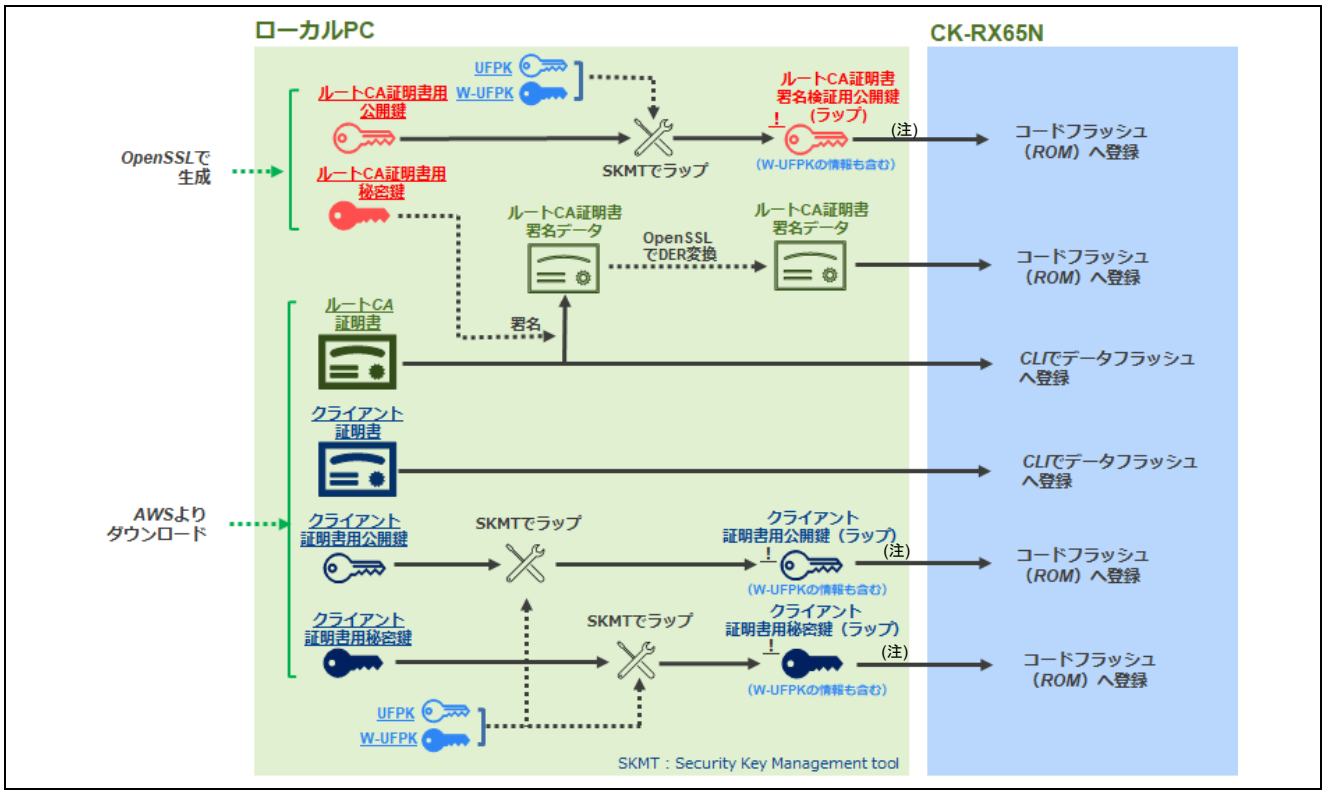


図 5-1 TSIP 用鍵・証明書の作成フロー

【注】 ルート CA 証明書署名検証用公開鍵とクライアント証明書用公開鍵・秘密鍵は「表 1-1」に記載している「Encrypted key」に相当します。それぞれの鍵情報に W-UFPK の情報を含んだファイルとして作成され、コードフラッシュへ登録します。

また、以下に「図 5-1」で示した UFPK・W-UFPK の作成フローを以下に示します。

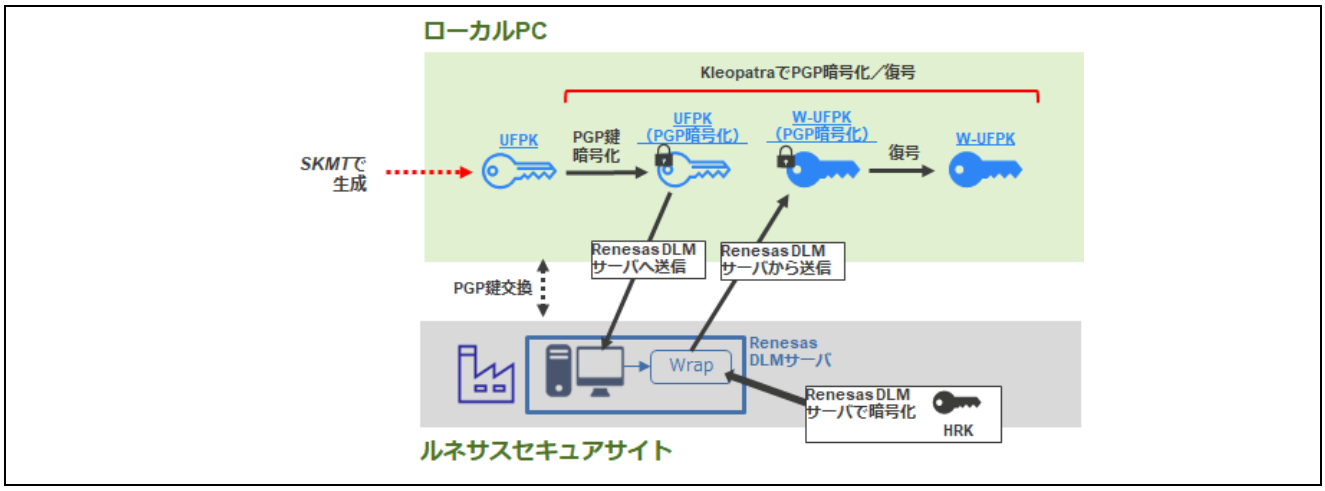


図 5-2 UFPK・W-UFPK の作成フロー

次項以降に証明書・鍵の入手方法・作成方法の手順を説明します。

5.1.2　ルート CA 証明書の入手

ルート CA 証明書を入手します。本項の作業は「図 5-1」の以下の赤で囲んだ部分に相当します。

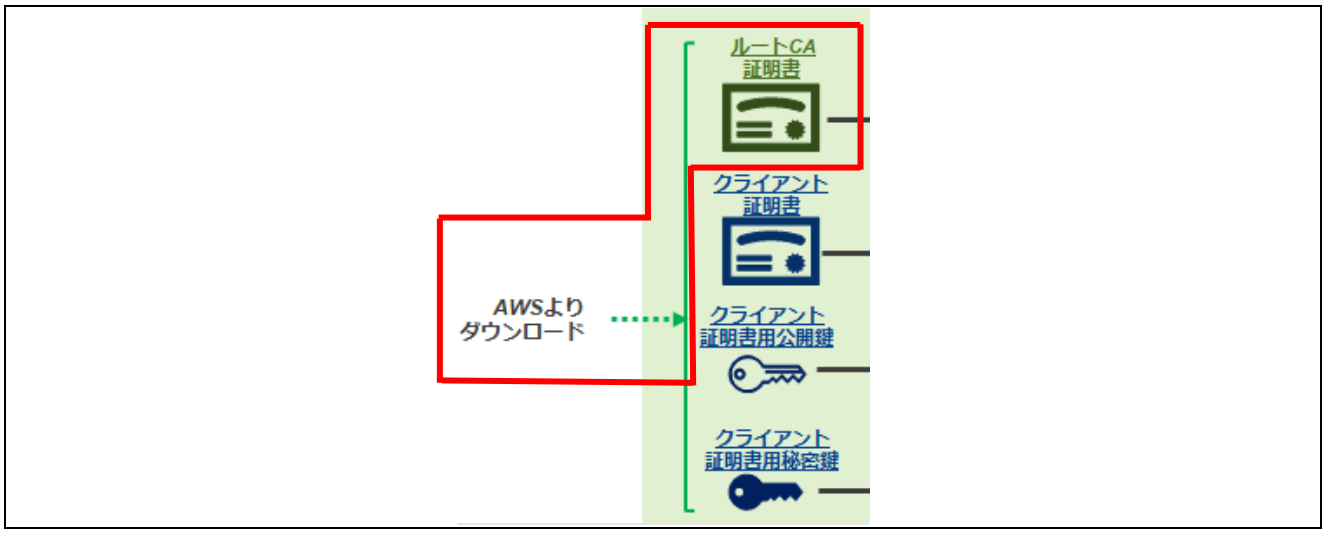
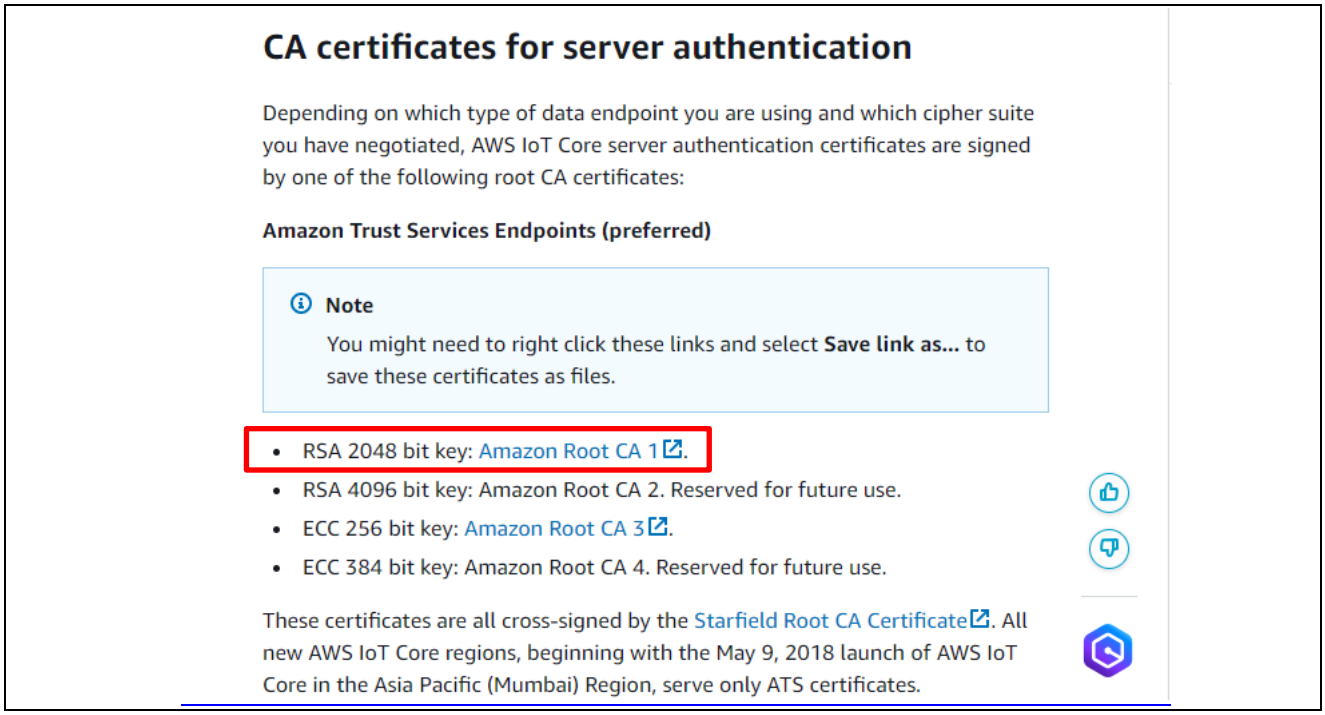


図 5-3 ルート CA 証明書の入手

AWS 接続に使用するルート CA 証明書を入手します。ルート CA 証明書は以下 URL からダウンロードしてください。

https://docs.aws.amazon.com/iot/latest/developerguide/server-authentication.html#server-authentication-certs

本プロジェクトでは RSA の証明書を使用するため「Amazon Root CA 1」を使用します。WEB ブラウザが Edge の場合はリンクを右クリックし、メニューより「名前を付けてリンクを保存」をクリックするとダウンロードできます。



CA certificates for server authentication

Depending on which type of data endpoint you are using and which cipher suite you have negotiated, AWS IoT Core server authentication certificates are signed by one of the following root CA certificates:

Amazon Trust Services Endpoints (preferred)

Note
You might need to right click these links and select **Save link as...** to save these certificates as files.

- RSA 2048 bit key: Amazon Root CA 1.
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.
- ECC 256 bit key: Amazon Root CA 3.
- ECC 384 bit key: Amazon Root CA 4. Reserved for future use.

These certificates are all cross-signed by the Starfield Root CA Certificate. All new AWS IoT Core regions, beginning with the May 9, 2018 launch of AWS IoT Core in the Asia Pacific (Mumbai) Region, serve only ATS certificates.

図 5-4 ルート CA 証明書のダウンロード

ダウンロードすると以下のファイルが入手できます。

- AmazonRootCA1.pem

ダウンロードした証明書は、プロジェクトの aws_ryz014a_tsip_ck_rx65n フォルダ下にある「**key_crt_sig_generator**」フォルダ内の以下のフォルダに配置してください。

```
key_crt_sig_generator
|-- ca
|   |-- AmazonRootCA1.pem
|-- ca-sign-keypair-rsa2048
|-- client-rsa2048
|-- output
|-- 1_rsa2048_convertCrt.sh
|-- convertCrt.sh
```

5.1.3 RSA の鍵ペアとクライアント証明書の入手

RSA の鍵ペア（クライアント証明書用公開鍵・秘密鍵）とクライアント証明書を入手します。本項の作業は「図 5-1」の以下の赤で囲んだ部分に相当します。

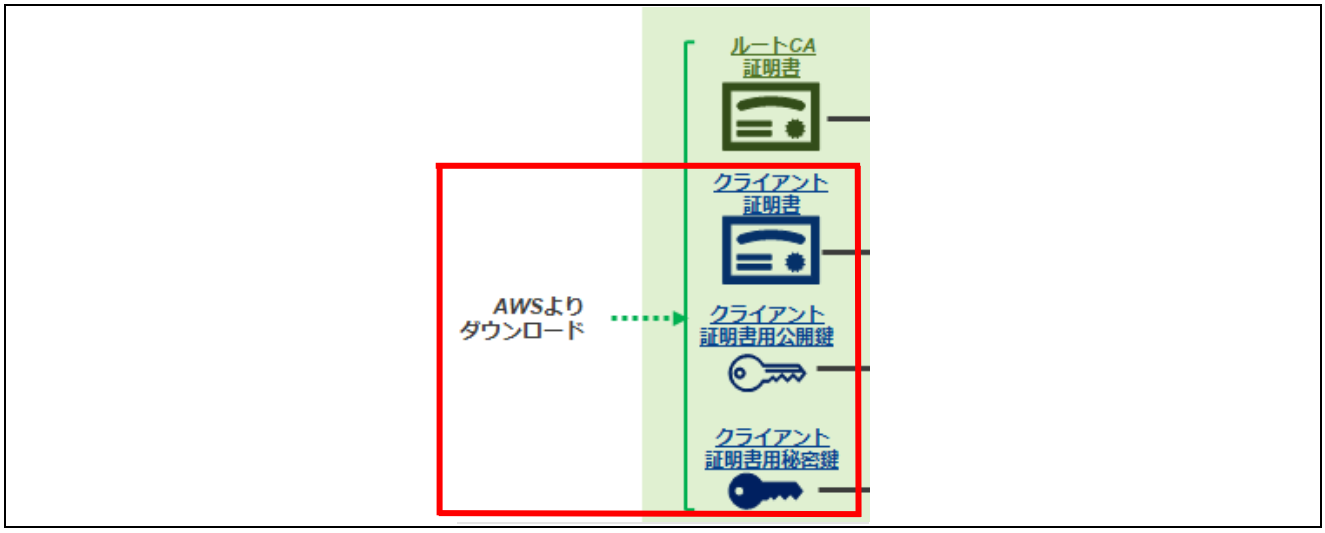


図 5-5 クライアント証明書／クライアント証明書用公開鍵・秘密鍵の入手

クライアント証明書とクライアント証明書用公開鍵・秘密鍵は AWS サーバ上で自動生成されます。AWS IoT Core サイトより、デバイス（モノ）の登録を行ってください。モノの登録時に鍵ペアとクライアント証明書をダウンロードすることができます。

ダウンロードはアプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「3.1.AWS コンソールへのサインイン」から「3.3.デバイスを AWS に登録する」の手順を実施する際に行ってください。

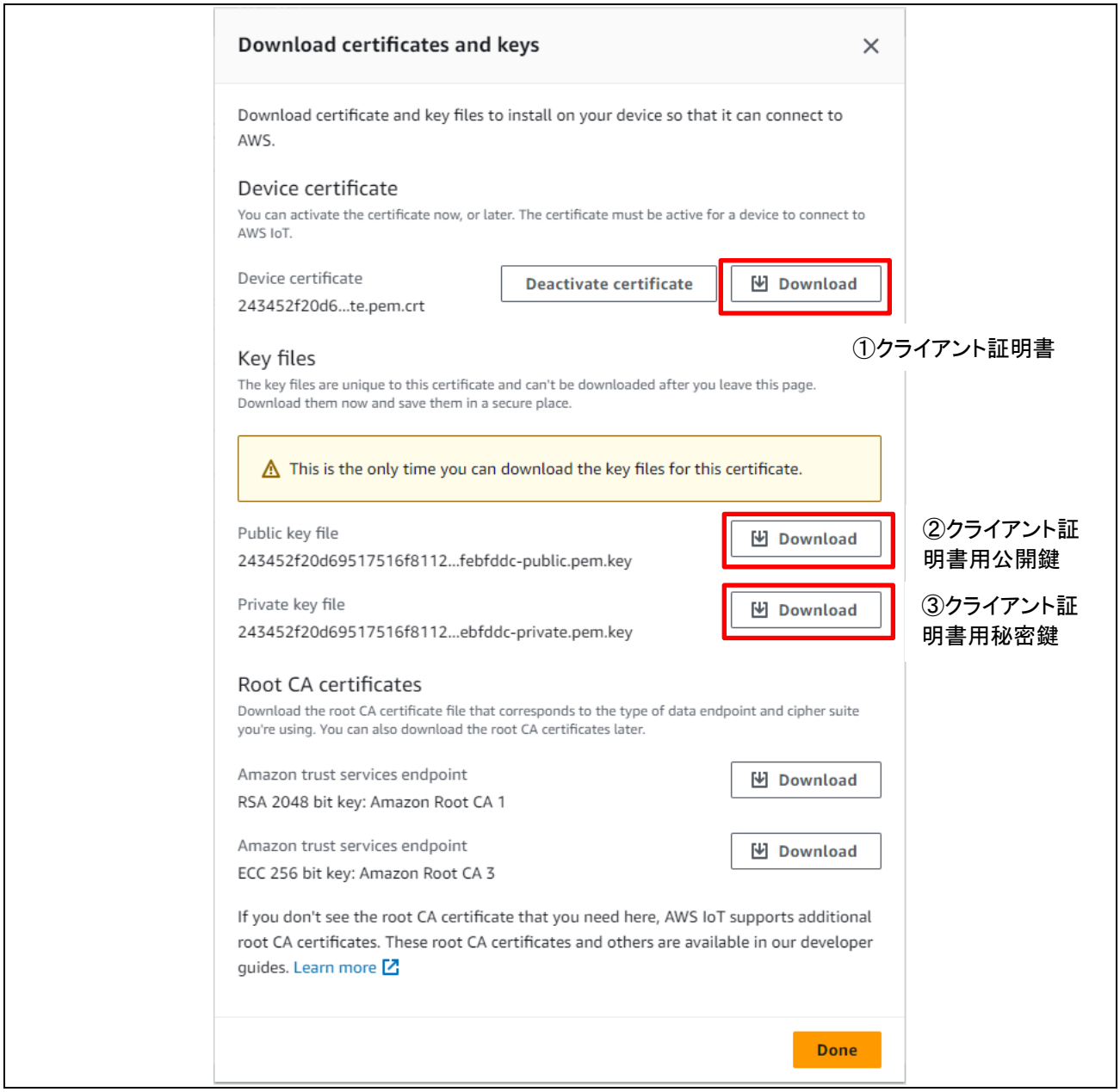


図 5-6 鍵・証明書のダウンロード

以下の 3 つのファイルをダウンロードすることができます。

表 5-2 ダウンロードするファイル一覧

番号	名称	ファイル名	本アプリケーションノートでの表記
①	Device certificate （デバイス証明書）	xxx-certificate.pem.crt	クライアント証明書
②	Public Key file （パブリックキーファイル）	xxx-public.pem.key	クライアント証明書用公開鍵
③	Private Key file （プライベートキーファイル）	xxx-private.pem.key	クライアント証明書用秘密鍵

【注】 1. 公開鍵・秘密鍵ファイルは AWS コンソールでモノの作成時にのみダウンロード可能なため注意してください。

【注】 2. xxx は任意の文字列となります。

ダウンロードした証明書と鍵は、プロジェクトの key_crt_sig_generator フォルダに以下のように配置してください。

```
key_crt_sig_generator
|-- ca
|   |-- AmazonRootCA1.pem
|-- ca-sign-keypair-rsa2048
|-- client-rsa2048
|   |-- xxx-certificate.pem.crt
|   |-- xxx-public.pem.key
|   |-- xxx-private.pem.key
|-- output
|-- 1_rsa2048_convertCrt.sh
|-- convertCrt.sh
```

5.1.4 ルート CA 証明書の署名生成

ダウンロードしたルート CA 証明書を使用して、プロジェクト（ソースコード）へ登録を行います。本項の作業は「図 5-1」の以下の赤で囲んだ部分に相当します。

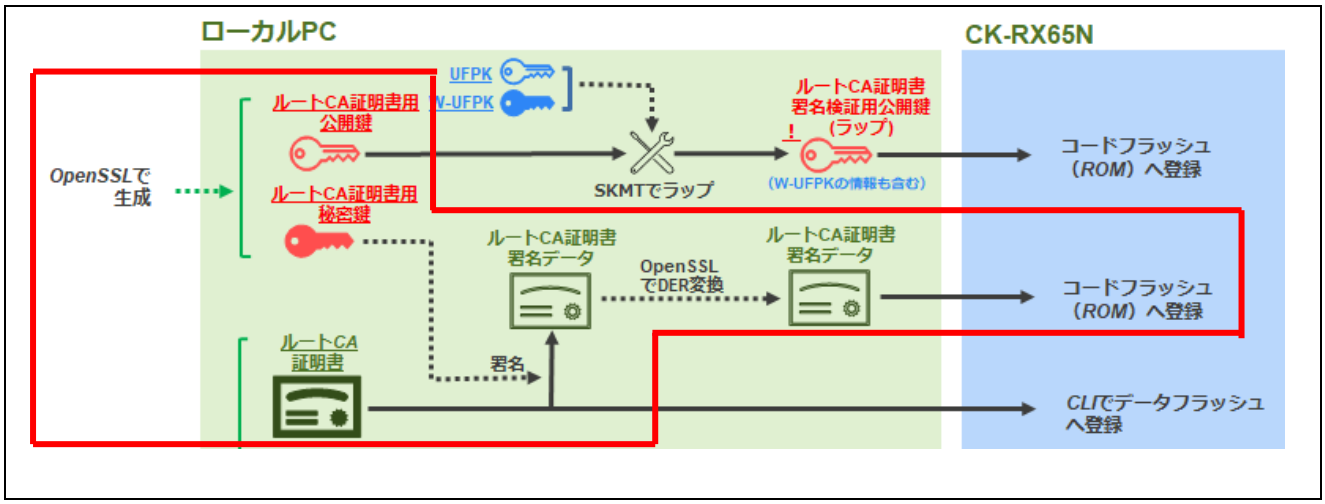


図 5-7 ルート CA 証明書の署名作成

以下の手順にて証明書の変換を行ってください。
これらの変換にはプロジェクトフォルダの key_crt_sig_generator 内にスクリプトファイルを用意しています。スクリプトファイルはシェルスクリプト（bash）にて実行してください。
本アプリケーションノートではシェルスクリプトの実行環境として Cygwin を使用した例で説明します。
「2.3」節を参照して Cygwin を実行できる準備を行ってください。

また、スクリプト実行の前に「5.1.2」「5.1.3」項の手順にて各鍵のファイルを所定のフォルダに配置しておいてください。

【注】スクリプトを実行した場合は、プログラム実行時に必ず「6.1.5(6)」を参照しデータフラッシュの削除を行ってください。また「5.1.4」から「5.1.5」の一連の作業は連続して実施して下さい。部分的に手順を実施すると、生成されたデータの整合が取れなくなる場合があります。

(1) RSA ルート CA 証明書の変換

本プロジェクトに添付されているスプリクトを実行し、RSA ルート CA 証明書を DER 形式に変換します。
その後、スプリクトではルート CA 証明書の署名生成と署名検証に使用する RSA-2048bit の鍵ペア（ルート CA 証明書用公開鍵・秘密鍵）を生成し、生成した鍵ペアの秘密鍵（ルート CA 証明書用秘密鍵）を使用して、ルート CA 証明書に対する署名データを生成します。
変換後の署名データは、プロジェクトのソースコードに登録できるように C 言語の配列形式に変換されます。

Cygwin を実行して以下のコマンドを入力し、プロジェクトの key_crt_sig_generator フォルダへ移動してください。

```
cd /cygdrive/c/workspace/key_crt_sig_generator
```

【注】 上記は「workspace/key_crt_sig_generator」フォルダにスクリプトファイルを配置した場合の例です。以下のコマンドを入力し、スクリプトを実行してください

```
./1_rsa2048_convertCrt.sh
```

図 5-8 スプリクトの実行

(2) 変換ファイルのプロジェクトへの登録

スクリプトの実行後、key_crt_sig_generator /output フォルダに以下の青・赤字の 6 つのファイルが生成されます。

```
key_crt_sig_generator
|-- ca
|-- ca-sign-keypair-rsa2048
|   |-- rsa2048-private.pem
|   |-- rsa2048-public.pem
|-- client-rsa2048
|-- output
|   |-- AmazonRootCA1_crt.der
|   |-- AmazonRootCA1_crt_array.txt
|   |-- AmazonRootCA1_sig.sig
|   |-- AmazonRootCA1_sig_array.txt
|--1_rsa2048_convertCrt.sh
|--convertCrt.sh
```

上記ファイルの詳細は以下の表のとおりです。

表 5-3 変換ファイルの詳細

名　称	ファイル名
ルート CA 証明書用秘密鍵ファイル（PEM 形式）	rsa2048-private.pem
ルート CA 証明書用公開鍵ファイル（PEM 形式）	rsa2048-public.pem
ルート CA 証明書データ（DER 形式）	AmazonRootCA1_crt.der
ルート CA 証明書データ （C 言語の uint8_t 型配列の形式で記述）	AmazonRootCA1_crt_array.txt
秘密鍵にて署名されたルート CA 証明書署名データ	AmazonRootCA1_sig.sig
秘密鍵にて署名されたルート CA 証明書署名データ （C 言語の uint8_t 型配列の形式で記述）	AmazonRootCA1_sig_array.txt

上記の出力ファイルのうち「**AmazonRootCA1_sig_array.txt**」が、秘密鍵にて署名されたルート CA 証明書署名データとなります。
このファイルは、生成されたバイナリデータを C 言語の uint8_t 型配列の形式で記述したものです。生成された「**AmazonRootCA1_sig_array.txt**」をプロジェクトの以下フォルダに上書き保存します。

```
\iot-reference-rx\
Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\src\userdata_tsip
```

また、「key_crt_sig_generator /ca-sign-keypair-rsa2048」フォルダに、ルート CA 証明書用公開鍵ファイル（rsa2048-public.pem）と秘密鍵ファイル（rsa2048-private.pem）が生成されます。
公開鍵ファイルは、ルート CA 証明書署名検証用公開鍵生成のために「5.1.5」項の処理で使用されます。

5.1.5　鍵のラップとプロジェクトへの登録

生成したルート CA 証明書用公開鍵とクライアント証明書用公開鍵・秘密鍵をラップしてプロジェクト（ソースコード）へ登録を行います。

この 3 つの鍵はユーザー鍵となります。

本項では「5.1.4」で生成したルート CA 証明書用公開鍵ファイルと、「5.1.3」で AWS よりダウンロードしたクライアント証明書用公開鍵・秘密鍵ファイルを使用します。

5.1.5.1　鍵のラップの概要

TSIP ドライバは、平文のユーザー鍵を入力として受け付けないため、鍵をラップして TSIP ドライバが受け付け可能な形式に変換する必要があります。

TLS で使用する鍵も証明書と同様、一般に PEM 形式で提供されます。TSIP ドライバで使用するためには PEM 形式の鍵ファイルから鍵データを抽出します。その後、Renesas Key Wrap サービス（Renesas DLM サーバを使用）および Security Key Management Tool を使用して鍵をラップします。Renesas Key Wrap サービスでの鍵データのやり取りは、OpenPGP 暗号化が必要です。

手順の概要は以下のようになります。詳細は次項以降で手順を示します。

①Security Key Management Tool を使用して平文の UFPK を作成
②Kleopatra を使用し、UFPK を PGP 暗号化する（Renesas Key Wrap サービスでやり取りするため）
③Renesas Key Wrap サービスを使用して、PGP 暗号化した UFPK をルネサスへ送信
④PGP 暗号化された UFPK を Renesas DLM サーバで暗号化する（ルネサス内で HRK が使用される）
⑤ルネサスより暗号化された UFPK（送信用に PGP 暗号化されたもの）を受信する
⑥Kleopatra を使用し PGP 暗号化を復号し、暗号化された UFPK（W-UFPK）を得る

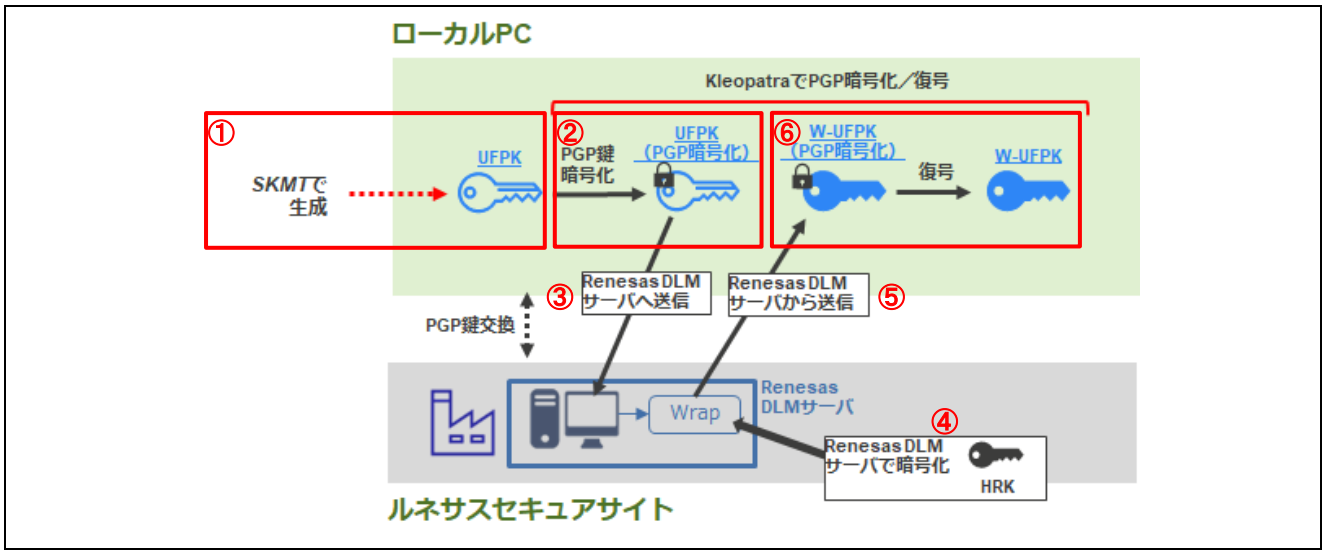


図 5-9 UFPK/W-UFPK 生成手順

⑦Security Key Management Tool を使用してユーザー鍵（ルート CA 証明書用公開鍵とクライアント証明書用公開鍵・秘密鍵）を UFPK でラップする（Encrypted key の生成）。Encrypted key には W-UFPK を結合し、ラップ後の鍵とする。
⑧ラップした暗号化鍵ファイルをソースコードに登録する。

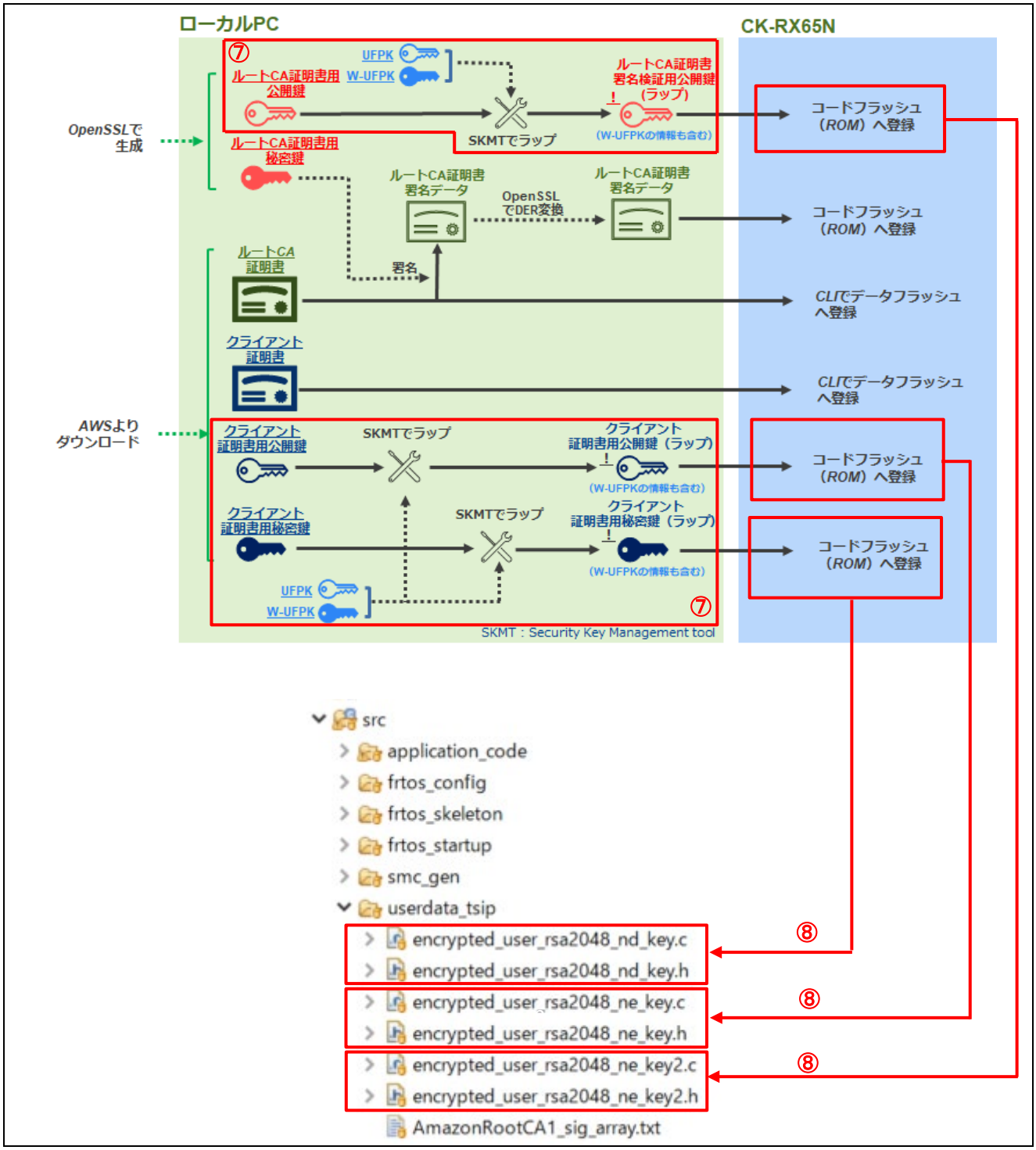


図 5-10 鍵情報のラップ手順

3 つの Encrypted key（ルート CA 署名検証用公開鍵／クライアント証明書用公開鍵・秘密鍵）は TSIP 内部処理で、HRK と HUK を使用して、チップ固有の鍵に暗号化された Wrapped key として生成されます。Wrapped key は TSIP 内部処理で復号され AWS 接続時に使用されます。

5.1.5.2 UFPK と W-UFPK の作成

鍵のラップに使用する User Factory Programming Key (UFPK)と W-UFPK（Renesas Key Wrapping サービスでラップされた UFPK）を生成します。本項の作業は「図 5-2」の部分に相当します。

任意の UFPK を生成し、DLM サーバにアップロードして W-UFPK を生成します。UFPK は署名検証用の公開鍵をラップするために使用する鍵です。
Security Key Management Tool を使用して、DLM サーバが受け付けるフォーマットの UFPK(注)ファイルを作成することができます。

UFPK と W-UFPK の作成は初回のみ実施します。

【注】 本項ではラップで使用する UFPK および暗号化された UFPK（W-UFPK）の生成手順を説明します。
ただし、ここの手順で生成する鍵情報はサンプルのため、実製品では使用することはできません。
量産などに適用する場合は独自の鍵を生成する必要があり、それらの詳細が書かれたアプリケーションノートを別途ご用意しています。
ルネサスマイコンをご採用／ご採用予定のお客様に提供させていただいていますので、お取引のあるルネサスエレクトロニクス営業窓口にお問合せください。https://www.renesas.com/contact/

(1)　ツールの準備

鍵のラップを実行するために以下のツールを使用します。各ツールのセットアップ手順を参照して使用できる状態としてください。

- Gpg4win（Kleopatra）　　：「2.1」節参照
- Renesas Key Wrap サービス：「2.2」節参照
- Security Management Tool　：「2.4」節参照

(2)　Security Key Management Tool（SKMT）の設定

インストールした Security Key Management Tool を起動し、[Overview（概要）]タブをクリックして、「Select MCU/MPU and Security engine（MCU/MPU と暗号エンジン）」の項目で、「RX Family, TSIP」を選択します。

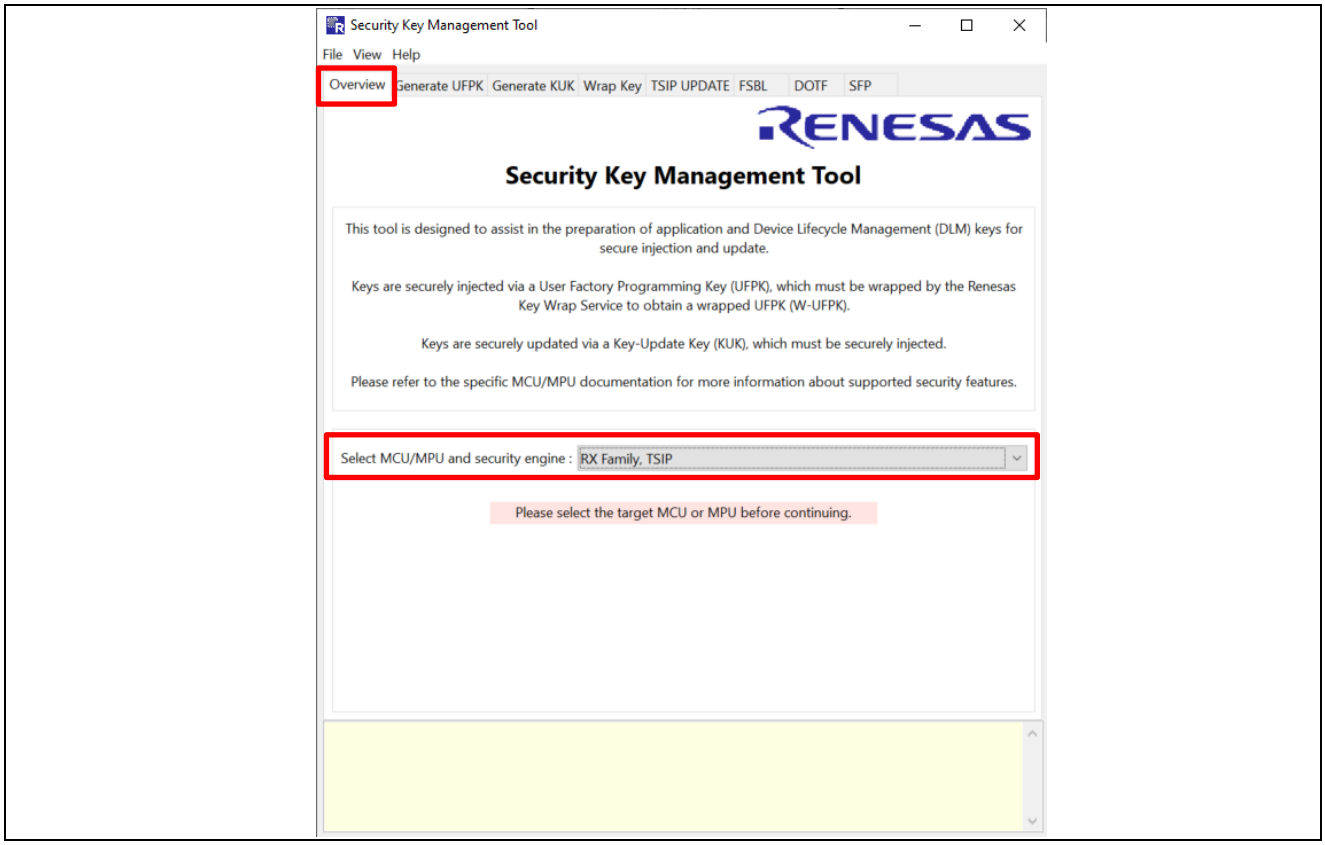


図 5-11 MCU と暗号エンジンの選択

(3) 平文の UFPK を作成

UFPK を作成します。本項の作業は「図 5-2」の以下の赤で囲んだ部分に相当します。

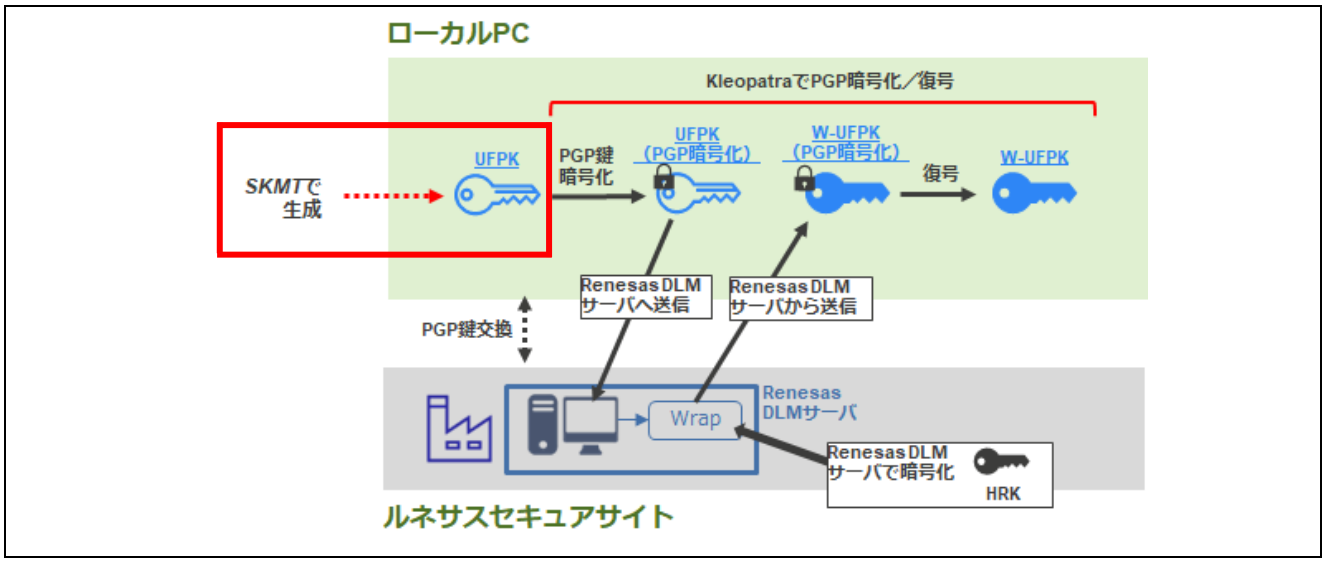


図 5-12 平文の UFPK の作成

Security Key Management Tool アプリの[Generate UFPK（UFPK 生成）]タブをクリックして、以下の項目の設定を行ってください。

- User Factory Programming Key　[Generate random value（乱数生成機能を使用する）]を選択
- Output file（出力ファイル）　出力する UFPK ファイルの任意のパスを設定、ファイル名を「sample.key」と設定

設定が完了したら[Generate UFPK key file（UFPK ファイルを生成する）]ボタンをクリックして、任意のフォルダに UFPK ファイルを出力してください。

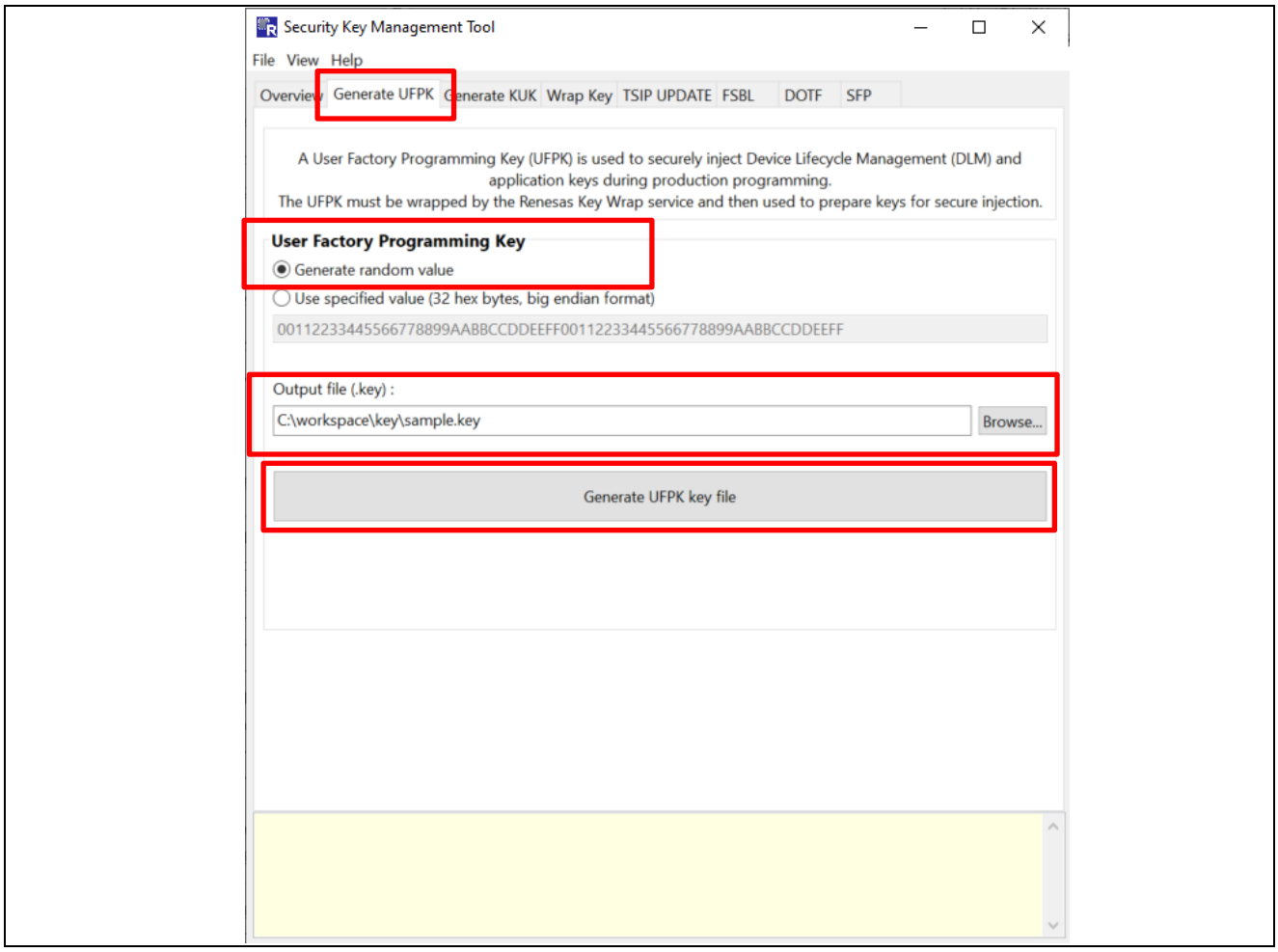


図 5-13 Security Key Management Tool による UFPK の生成

画面下部の処理結果表示部に以下のように表示されたら生成完了です。指定したフォルダに「sample.key」ファイルが出力されていることを確認してください。

```
Output File: C:\workspace\key\sample.key
OPERATION SUCCESSFUL
```

図 5-14 出力結果

(4) UFPK の PGP 暗号化

「5.1.5.2 (3)」で作成した UFPK（sample.key）を、Kleopatra を使用して作成した OpenPGP 鍵ペアとルネサスと交換した PGP 公開鍵で PGP 暗号化します。本項の作業は「図 5-2」の以下の赤で囲んだ部分に相当します。

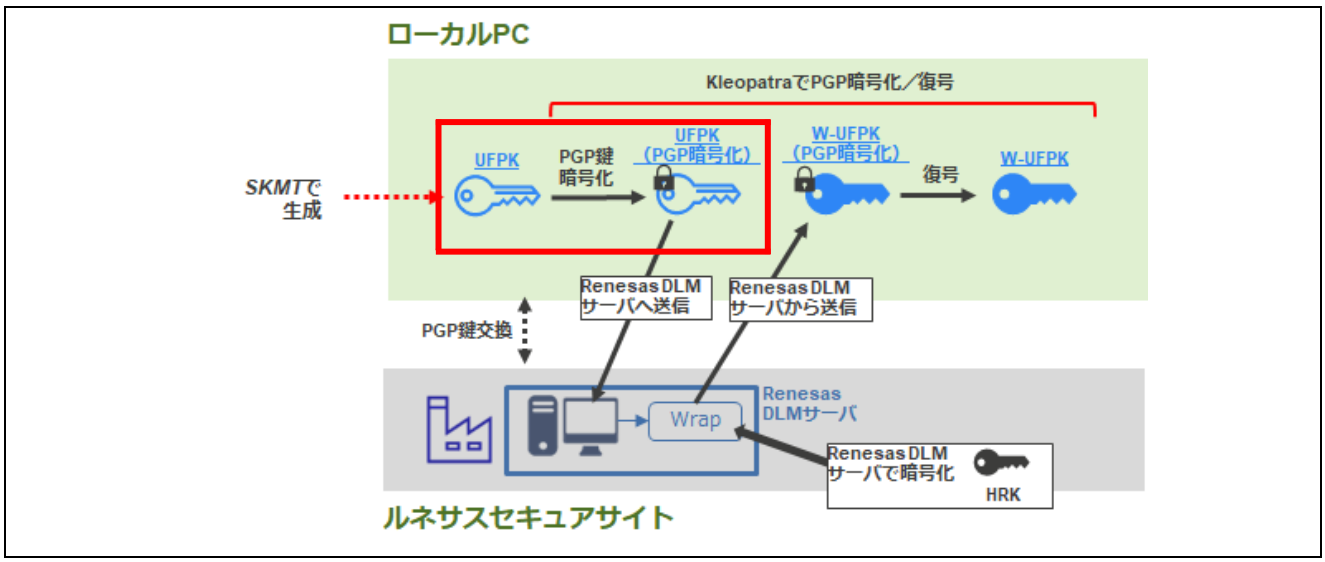


図 5-15 UFPK の PGP 暗号化

Kleopatra の画面で、[Sign/Encrypt（署名/暗号化）]ボタンをクリックしてください。ファイルの選択ダイアログが表示されるので、「sample.key」を指定してください。

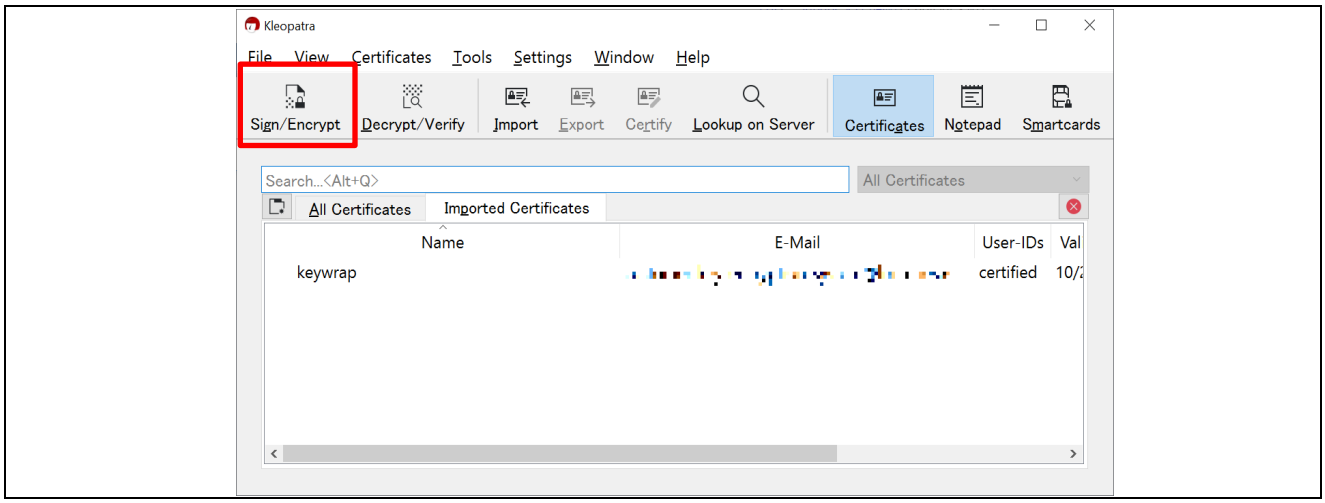


図 5-16 sample.key の暗号化

選択したファイルに対して署名/暗号化の確認画面が出るので、以下のように「Sign/Encrypt Files（署名と暗号化）」の設定画面で作成した鍵を指定します。

- Sign as（署名）：「2.2(2)」項で作成した自分の OpenPGP 鍵ペアを指定
- Encrypt for me（自分自身のために暗号化）：
  「2.2(2)」項で作成した自分の OpenPGP 鍵ペアを指定
- Encrypt for others（他の方のために暗号化）：
  「2.2(4)」項で登録したルネサス PGP 公開鍵（keywrap）を指定

「Output files/folder（出力先フォルダ）」へ出力先フォルダを設定したら[Sign/Encrypt（署名／暗号化）]ボタンをクリックしてください。指定したフォルダに PGP 暗号化された UFPK（W-UFPK）「**sample.key.gpg**」ファイルが作成されます。

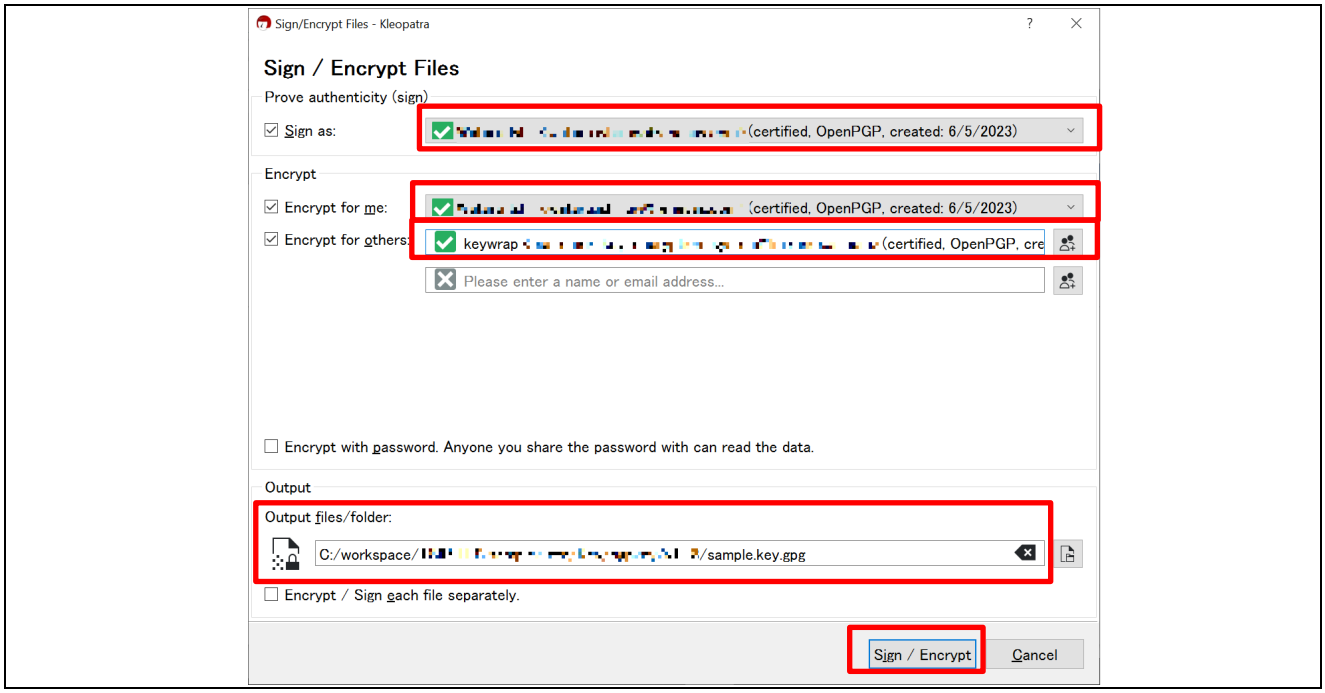


図 5-17 sample.key を PGP 暗号化

(5) PGP 暗号化した UFPK を DLM サーバで暗号化

PGP 暗号化した UFPK を Renesas Key Wrap サービスを使用して暗号化します。本項の作業は「図 5-2」の以下の赤で囲んだ部分に相当します。DLM サーバ内では HRK を使用して暗号化されます。

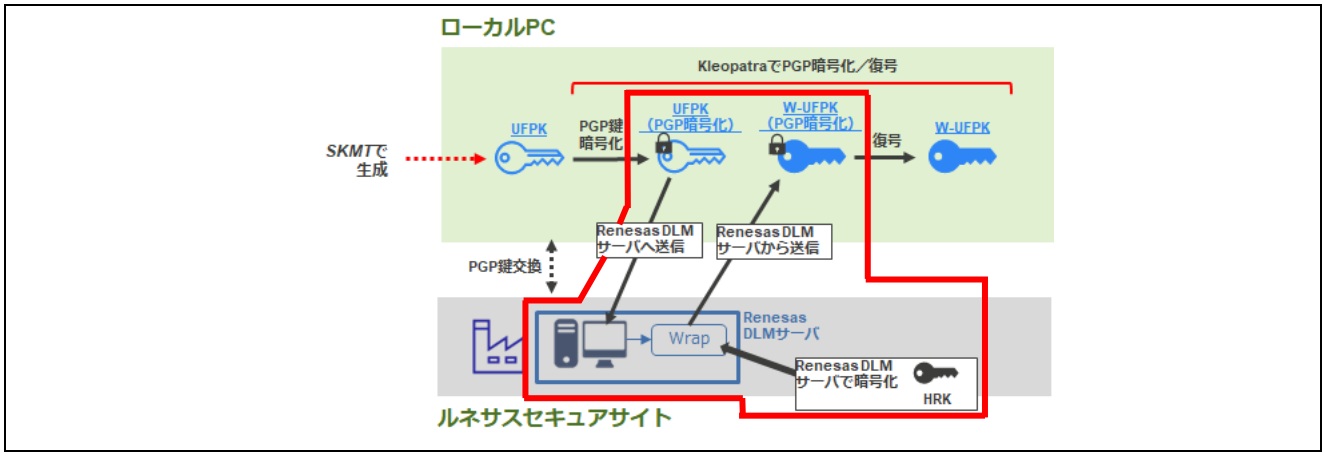


図 5-18 UFPK の暗号化のためのアップロードと暗号化

Renesas Key Wrap サービスの Web サイトにログインし、トップ画面から[RENESAS RX]をクリックします。

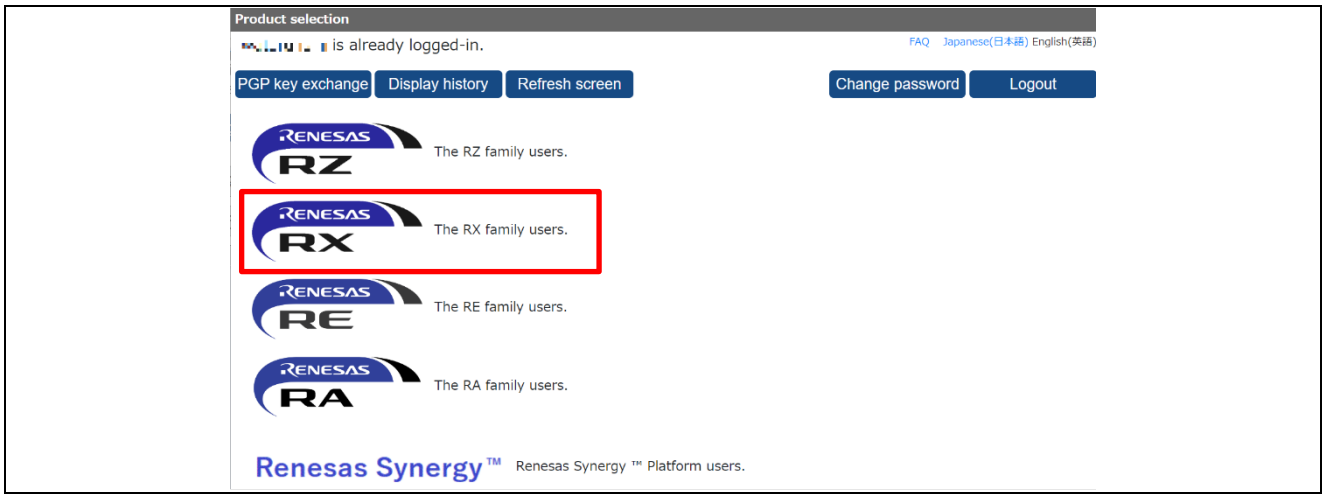


図 5-19 暗号化するファミリの選択

以下のデバイス選択画面が表示されるので、[RX65N/RX651 Encryption of customer's data（お客様データ暗号化）] をクリックします。

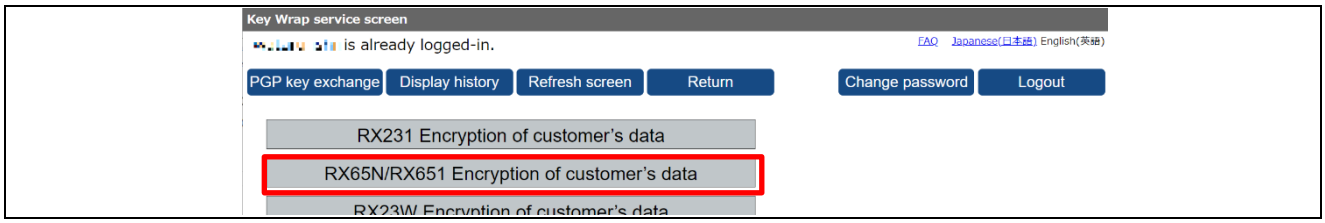


図 5-20 デバイスグループの選択

[Encryption service for products（製品用暗号化サービス）]のリンクをクリックします。

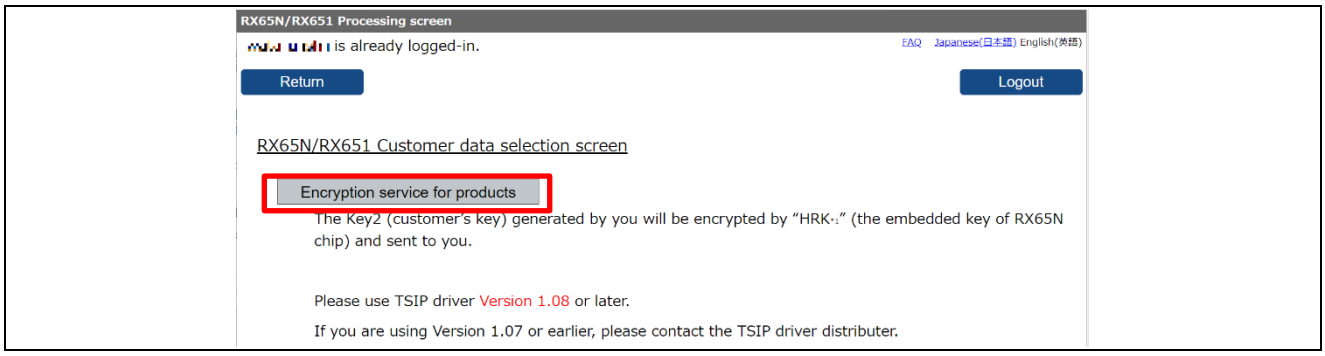


図 5-21 暗号化サービスの選択

アップロード画面が表示されるので[Reference（参照）]ボタンをクリックして、「5.1.5.2(4)」で作成した「sample.key.gpg」を指定し、[Settle（確定）]ボタンをクリックしてください。

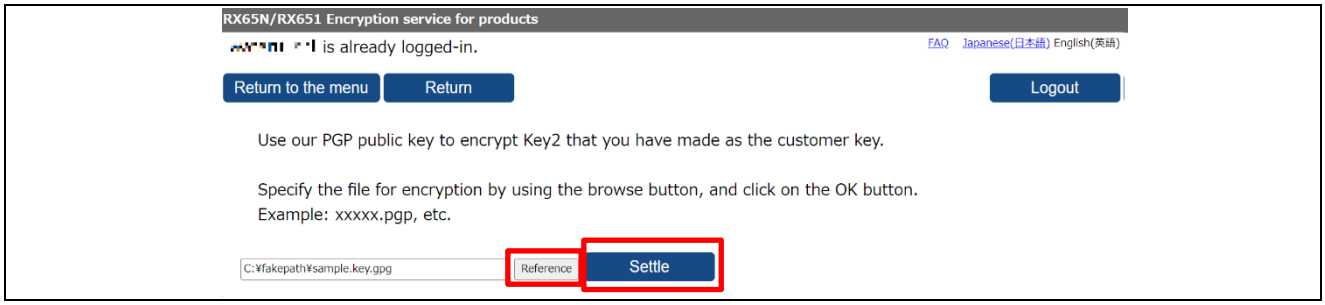


図 5-22 UFPK のアップロード

アップロードが完了すると以下の画面が表示され、Renesas DLM サーバにて暗号化が実行されます。暗号化が完了すると、ルネサスよりメールにて「sample.key_enc.key.pgp」が送られてきますので、任意のフォルダに保存してください。

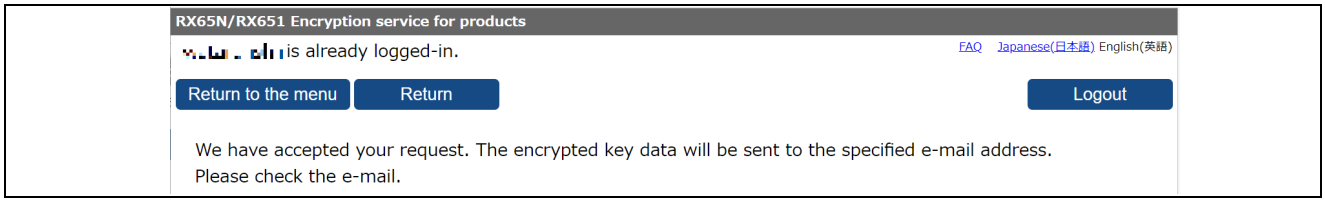


図 5-23 DLM サーバへアップロード完了

(6) sample.key_enc.key.pgp の OpenPGP 復号

ルネサスより送られてきた sample.key_enc.key.pgp（W-UFPK(PGP 暗号化)）を自身の OpenPGP 鍵ペアで復号して、W-UFPK（暗号化された UFPK）を作成します。本項の作業は「図 5-2」の以下の赤で囲んだ部分に相当します。

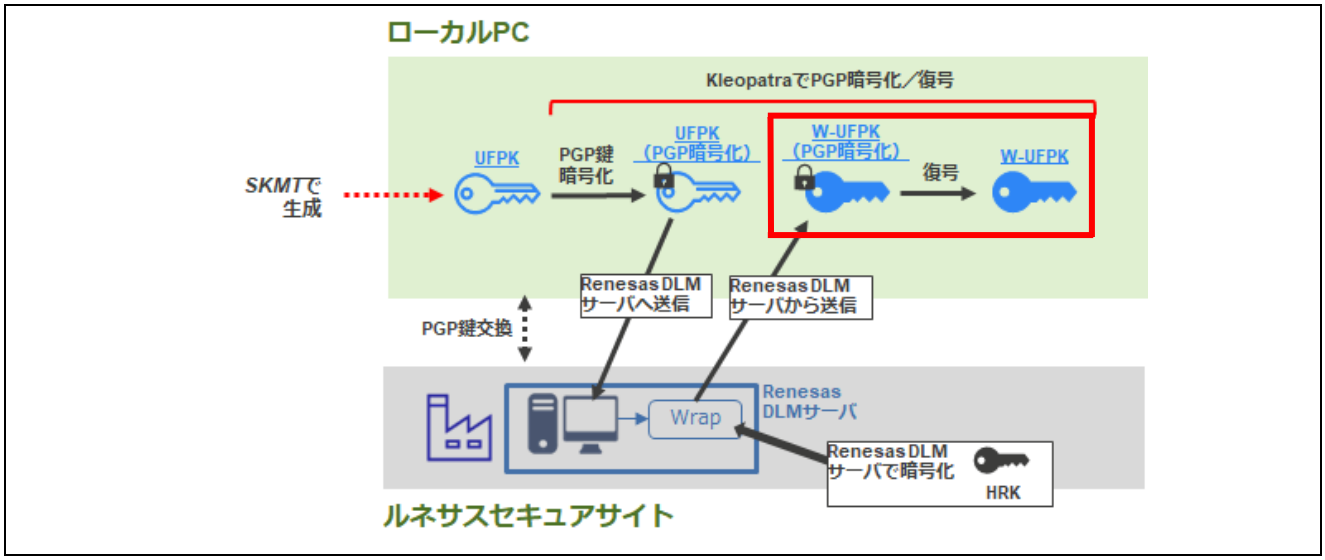


図 5-24 W-UFPK の作成

Kleopatra で[Decrypt/Verify（復号/検証）]ボタンをクリックして、ファイルの選択画面で「sample.key_enc.key.pgp」を選択してください。復号が開始されます。
復号が完了したら完了画面が表示されるので[Save All（すべて保存）]ボタンをクリックして、復号した鍵を保存してください。復号前のファイルと同じフォルダに復号された、「sample.key_enc.key」が出力されます。
以上で、sample.key の暗号化は完了です。sample.key_enc.key が W-UFPK となります。

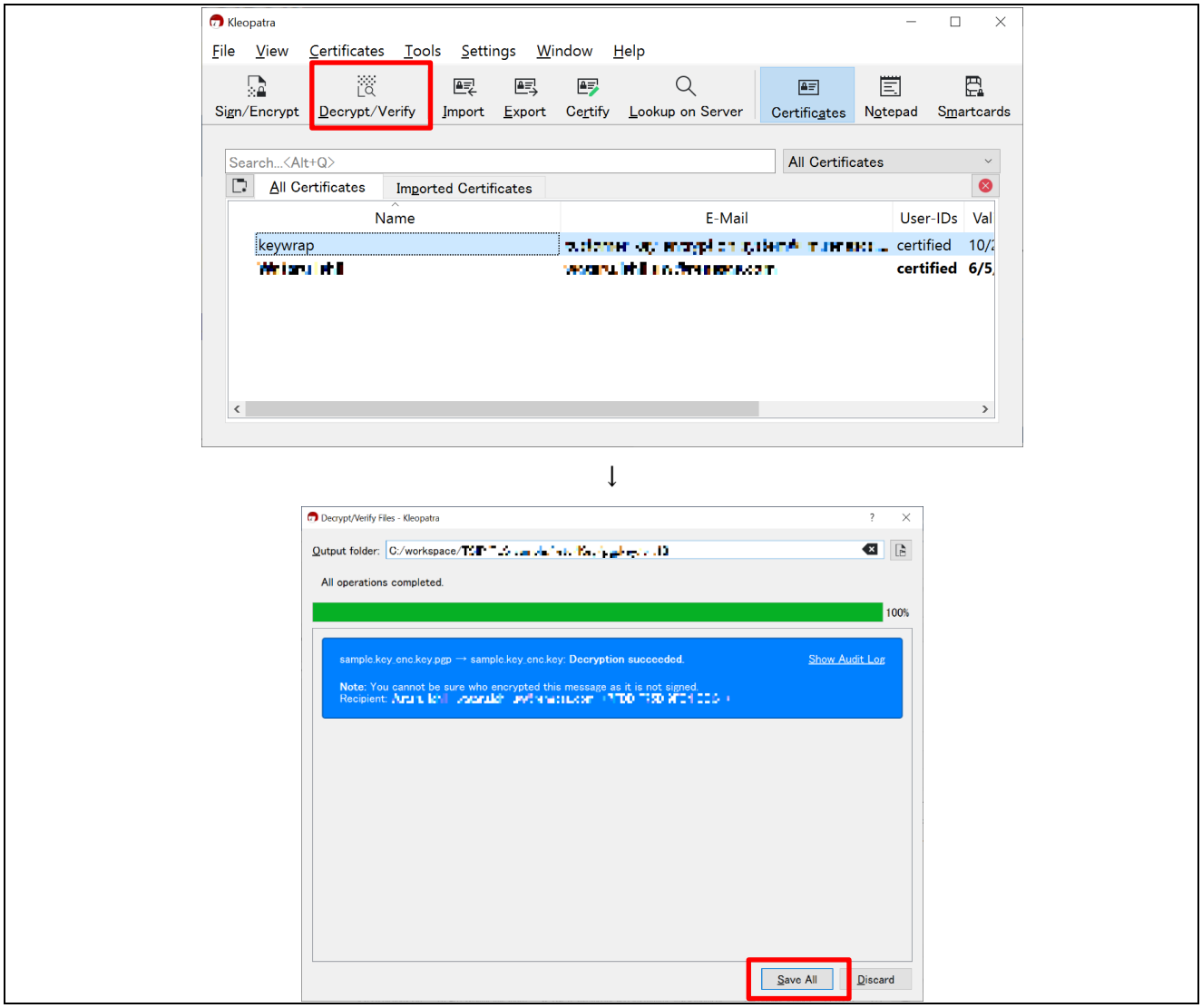


図 5-25 PGP 復号による W-UFPK の作成

5.1.5.3 鍵データのラップ

ルート CA 証明書用公開鍵データ（ルート CA 証明書署名検証用公開鍵）およびクライアント証明書用公開鍵・秘密鍵データを「5.1.5.2 UFPK と W-UFPK の作成」で作成した sample.key（UFPK）と sample.key_enc.key（W-UFPK）を使用して、プロジェクトへ登録するデータへ変換します。
変換データは上記の 3 種の鍵をそれぞれ UFPK でラップして、W-UFPK と結合したものとなります。

(1) 使用する鍵データ

「5.1.4 ルート CA 証明書の署名生成」で生成したルート CA 証明書用公開鍵と「5.1.3 RSA の鍵ペアとクライアント証明書の入手」で AWS より入手した PEM 形式のクライアント証明書用公開鍵・秘密鍵ファイルから、ルート CA 証明書署名検証用公開鍵とクライアント証明書公開鍵・秘密鍵データを抽出します。これらは、以下の 3 つの鍵ファイルとなります。

- ルート CA 証明書用公開鍵ファイル(PEM)
  /key_crt_sig_generator /ca-sign-keypair-rsa2048 /rsa2048-public.pem
- クライアント証明書用公開鍵ファイル(PEM)
  /key_crt_sig_generator /client-rsa2048 /xxxx-public.pem.key
  【注】 xxxx は任意の文字となります
- クライアント証明書用秘密鍵ファイル(PEM)
  /key_crt_sig_generator /client-rsa2048 /xxxx-private.pem.key

【注】1. xxxx は任意の文字となります
【注】2. 「5.1.5.2」項で作成した以下の 2 つの鍵を用いて上記鍵のラップを行います。
- UFPK（sample.key）
- W-UFPK（sample.key_enc.key）

ここで使用する鍵は以下の赤枠で囲んだ鍵となります。
ルート CA 証明書署名検証用公開鍵とクライアント証明書用公開鍵・秘密鍵のラッピングに使用する UFPK ／W-UFPK は同じものです。

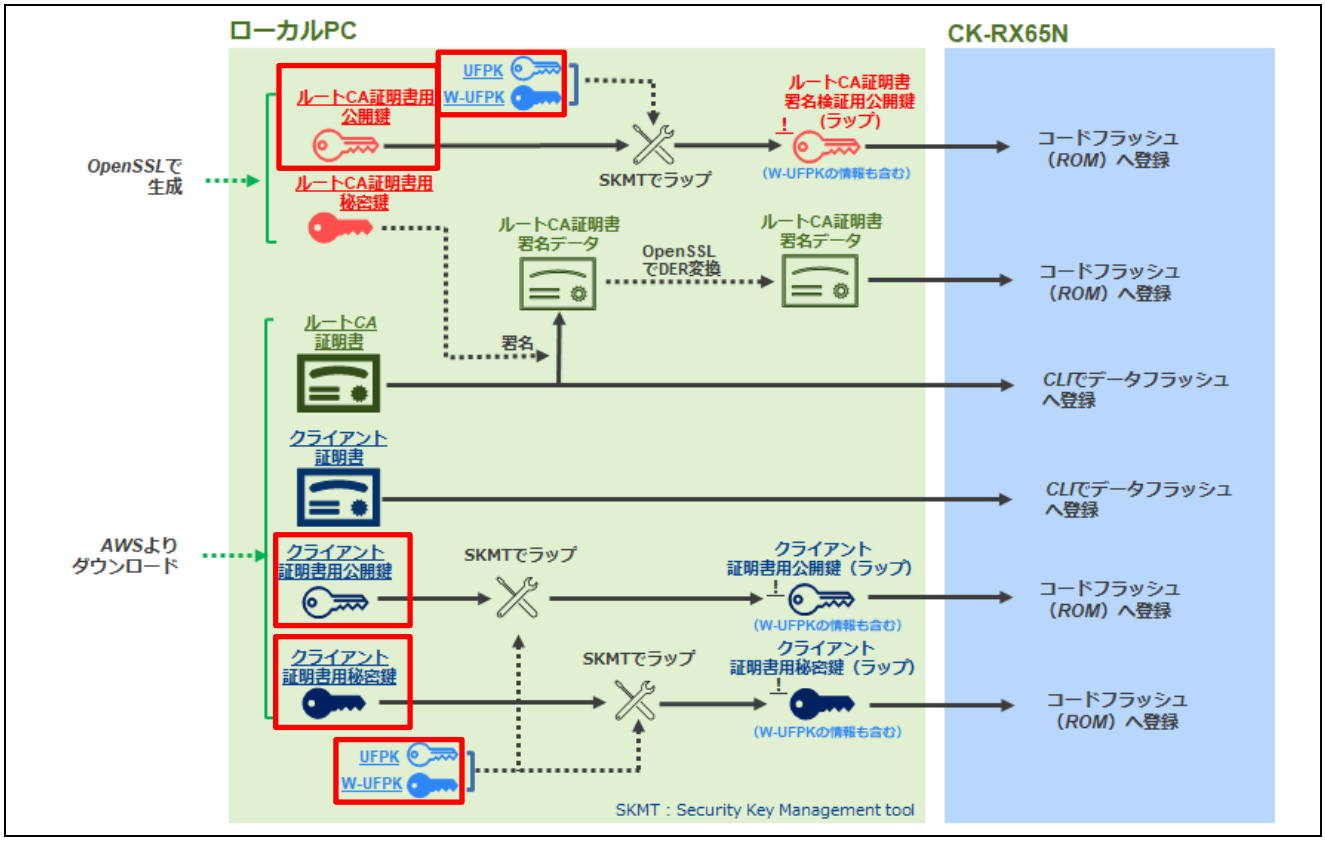


図 5-26 本項で使用する鍵データ

これらの鍵データを Security Key Management Tool に入力し、ラップされた鍵ファイルを生成します。ここで生成する鍵ファイルは、UFPK でラップされた鍵（encrypted key）に W-UFPK を付加してプロジェクトに組み込み可能なソースコードとして出力されます。
W-UFPK は TSIP 内部の処理でラップを解除するために使用されます。

上記鍵ファイルは AWS に作成したデバイス（モノ）に対応しているため、異なるデバイスへ接続する場合は、鍵ファイルのラップはその都度実施する必要があります。

次に、鍵ファイルのラップ手順を示します。

(2) ルート CA 証明書署名検証用公開鍵の生成

ルート CA 証明書署名検証用公開鍵を生成します。本項の作業は「図 5-1」の以下の赤で囲んだ部分に相当します。

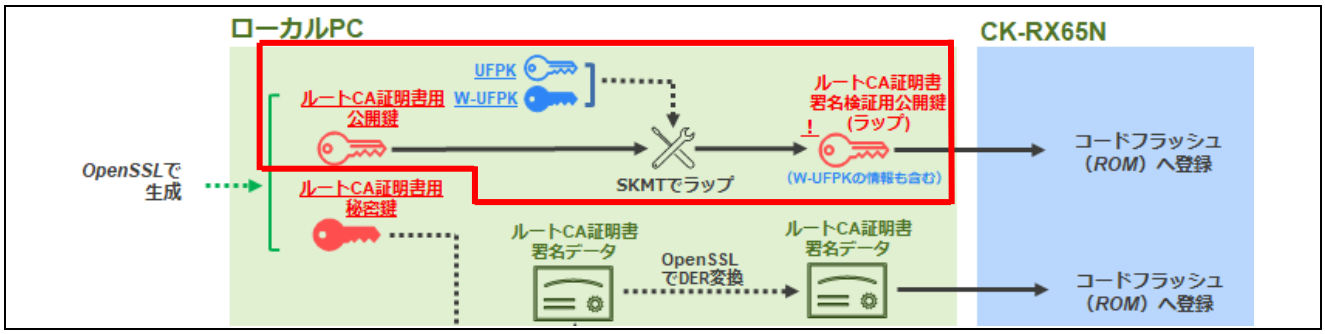


図 5-27 ルート CA 証明書署名検証用公開鍵の生成

以下のルート CA 証明書用公開鍵ファイルをラップして、プロジェクトに組み込むルート CA 証明書署名検証用公開鍵を作成します。

/key_crt_sig_generator /ca-sign-keypair-rsa2048 /rsa2048-public.pem

① Security Key Management Tool アプリの[Wrap key（鍵のラッピング）]タブをクリックして、さらに[Key Type（鍵の種類）]タブをクリックし、鍵の種類より[RSA]のラジオボタンを選択後、リストより「2048bits, public」を選択してください。

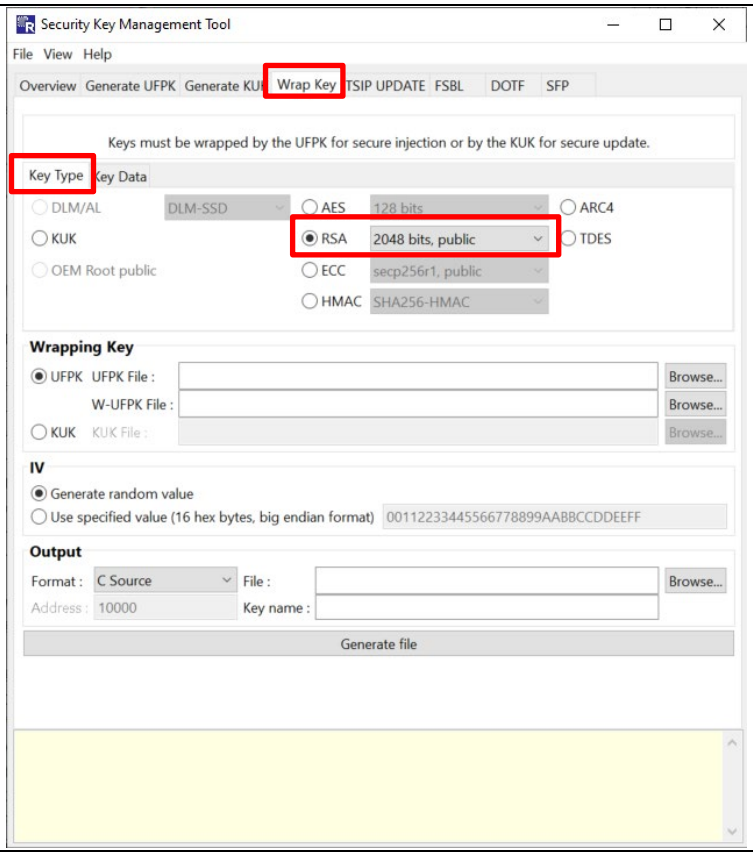


図 5-28 Security Key Management Tool を使用した公開鍵のラップ

② UFPK と W-UFPK の登録を行います。Wrapping Key（ラッピング鍵）の欄で[UFPK]を選択し、以下の項目を設定してください。
「UFPK File（ファイル）」と「W-UFPK File（ファイル）」の[Browse（参照）]ボタンをそれぞれクリックして、「5.1.5.2」項で作成した UFPK（sample.key）と W-UFPK（sample.key_enc.key）を指定します。
IV 値は[Generate random Value（乱数生成機能を使用する）]を選択してください。

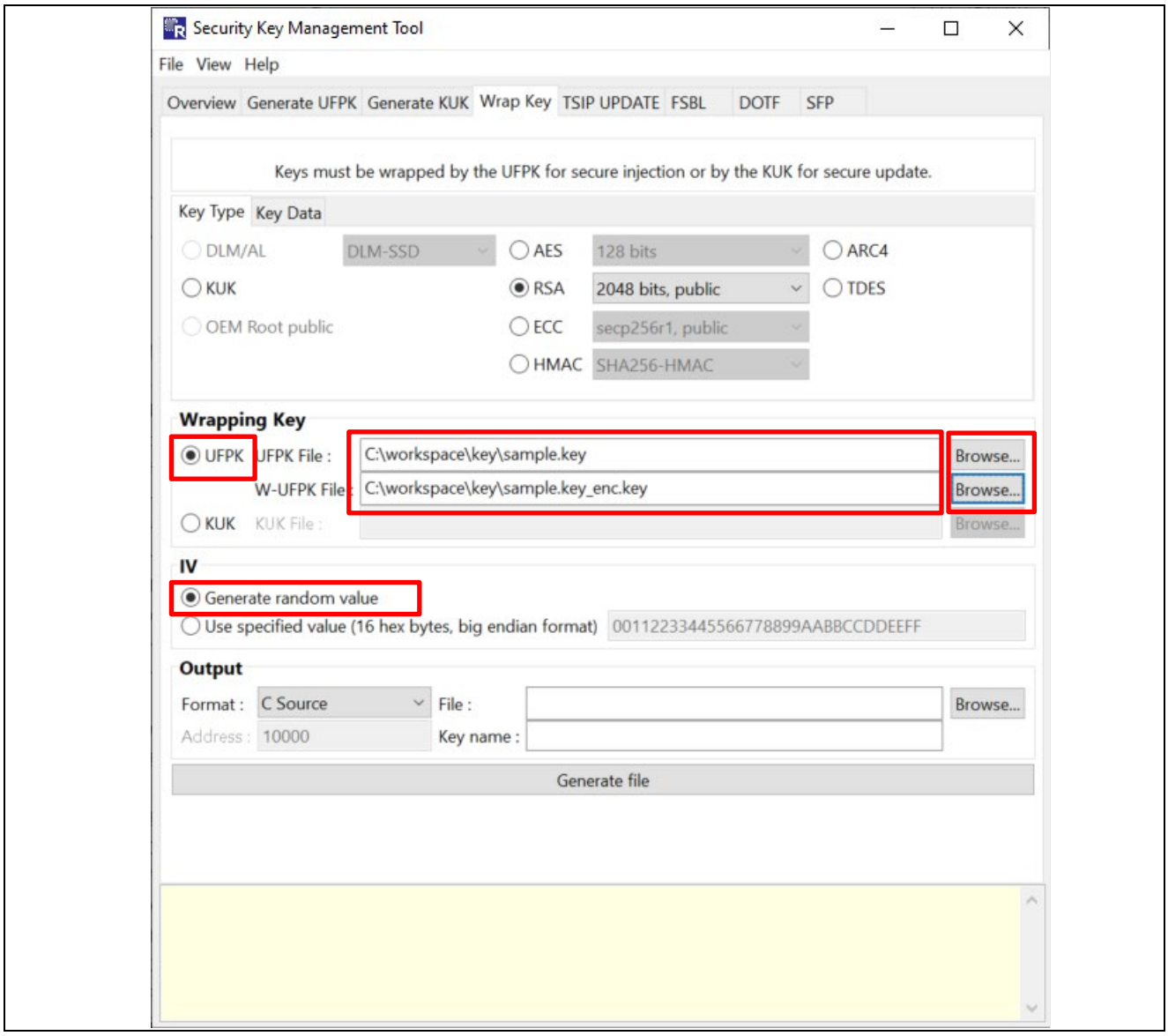


図 5-29 UFPK と W-UFPK の指定

③ [Key Data（鍵データ）]タブをクリックします。
[File（ファイル）]を選択し、[Browse（参照）]ボタンをクリックしてください。

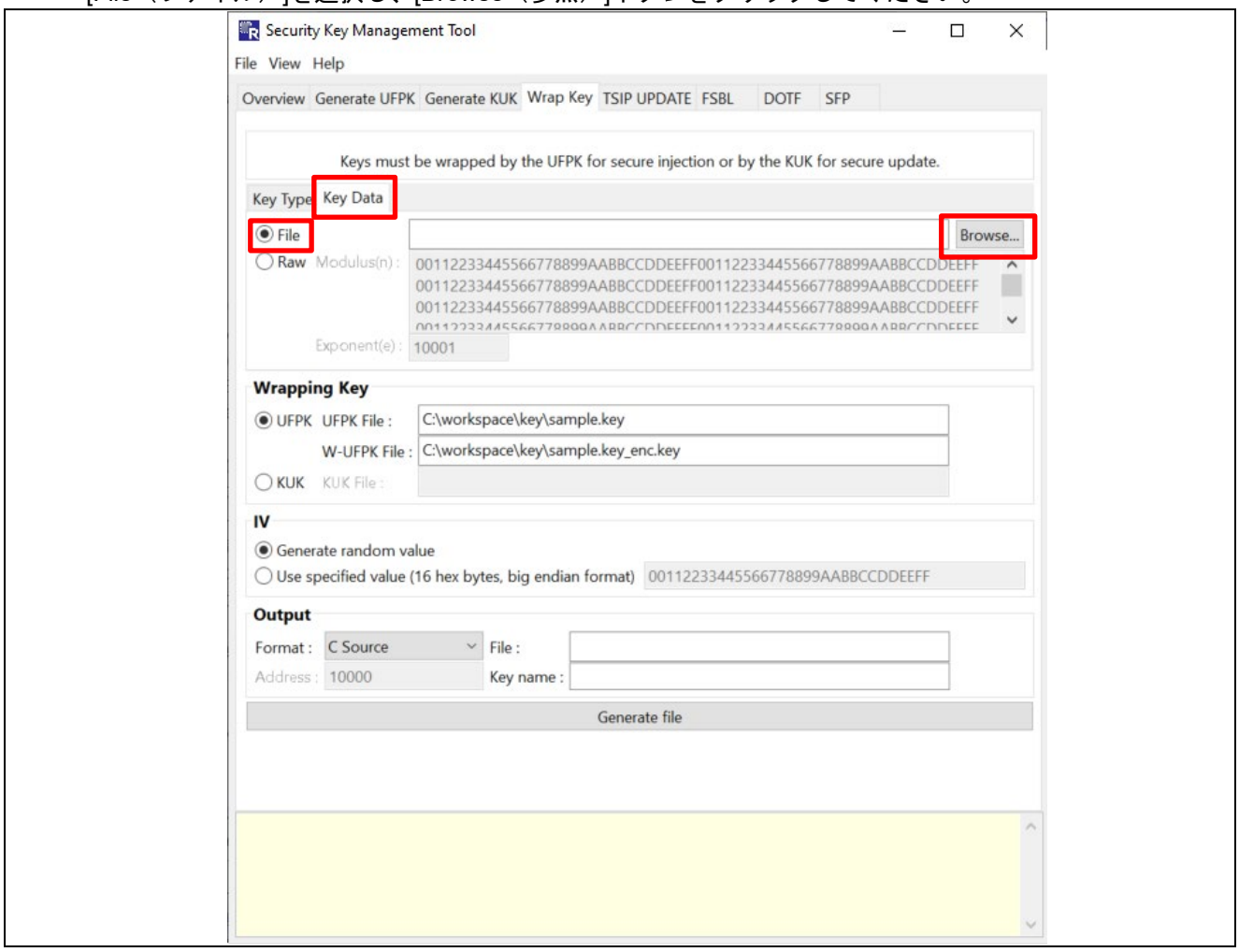


図 5-30 ルート CA 証明書用公開鍵の選択

Key data File（鍵データファイル）の選択ダイアログが開くので、ファイルの種類のリストより[PEM 鍵データ(*.pem)]を選択してください。ルート CA 証明書用公開鍵ファイル(PEM) が表示できるようになるので、/key_crt_sig_generator/ca-sign-keypair-rsa2048/rsa2048-public.pem を選択して、[Open（開く）]ボタンをクリックします。

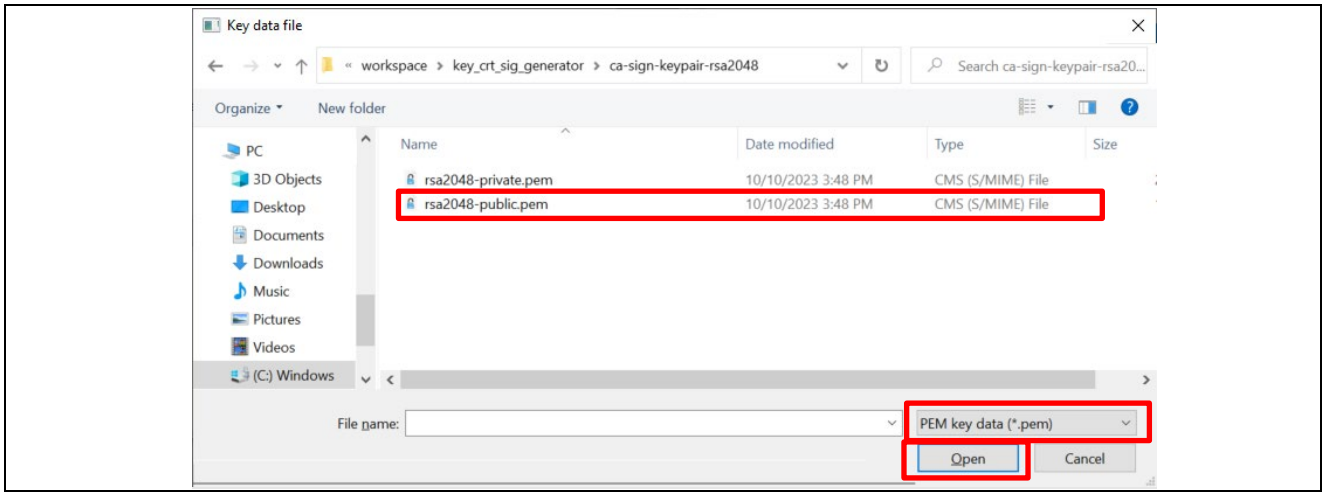


図 5-31 ファイル選択ダイアログ

④ Output（出力）の項目の「Format（フォーマット）」を「C Source（C ソース）」に設定します。
「File（ファイル）」に出力先の任意のフォルダを選択しファイル名に
「**encrypted_user_rsa2048_ne_key.c**」を入力してください。
また、「Key name」のテキストボックスには「**encrypted_user_rsa2048_ne_key**」と入力してください。
入力が完了したら、[Generate file（ファイルを生成する）]をクリックして、ルート CA 証明書署名検証用公開鍵データを生成します。

【注】 ファイル名および Key name に入力した「encrypted_user_rsa2048_ne_key」はソースコード内で使用しているため、変更しないようにしてください。

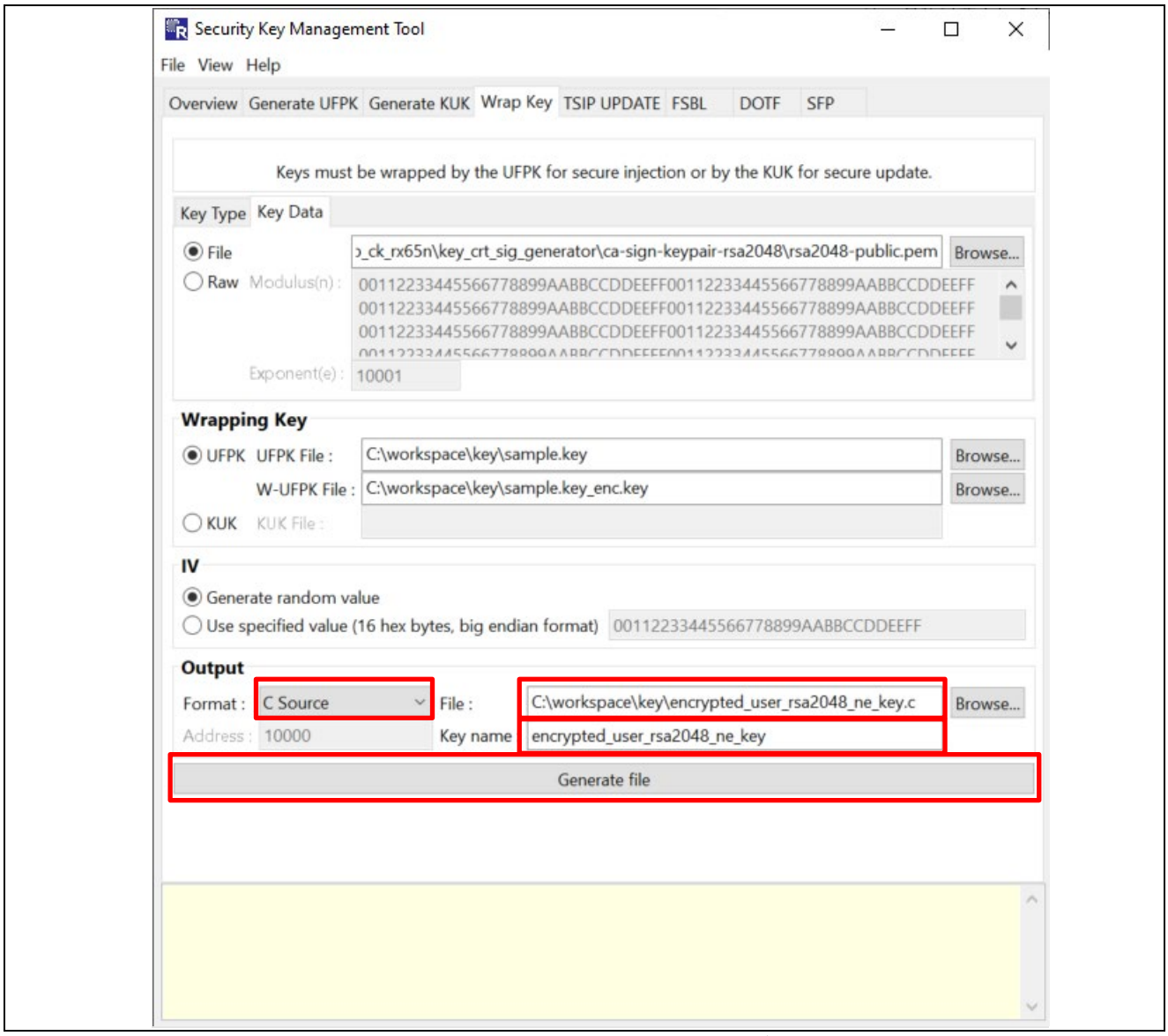


図 5-32 ルート CA 証明書署名検証用公開鍵データの生成

画面下部の処理結果表示部に以下のように表示されたら生成完了です。指定したフォルダに「encrypted_user_rsa2048_ne_key.c/h」ファイルが出力されていることを確認してください。

ここで生成したルート CA 証明書用公開鍵は、プロジェクトで「ルート CA 証明書署名検証用公開鍵」として使用されます。

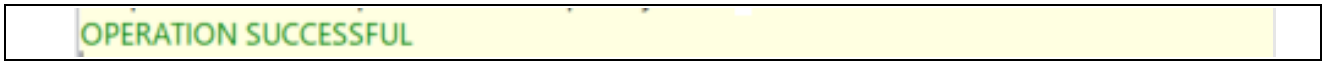


図 5-33 出力結果

(3) クライアント証明書用公開鍵の生成

クライアント証明書用公開鍵を生成します。本項の作業は「図 5-1」の以下の赤で囲んだ部分に相当します。

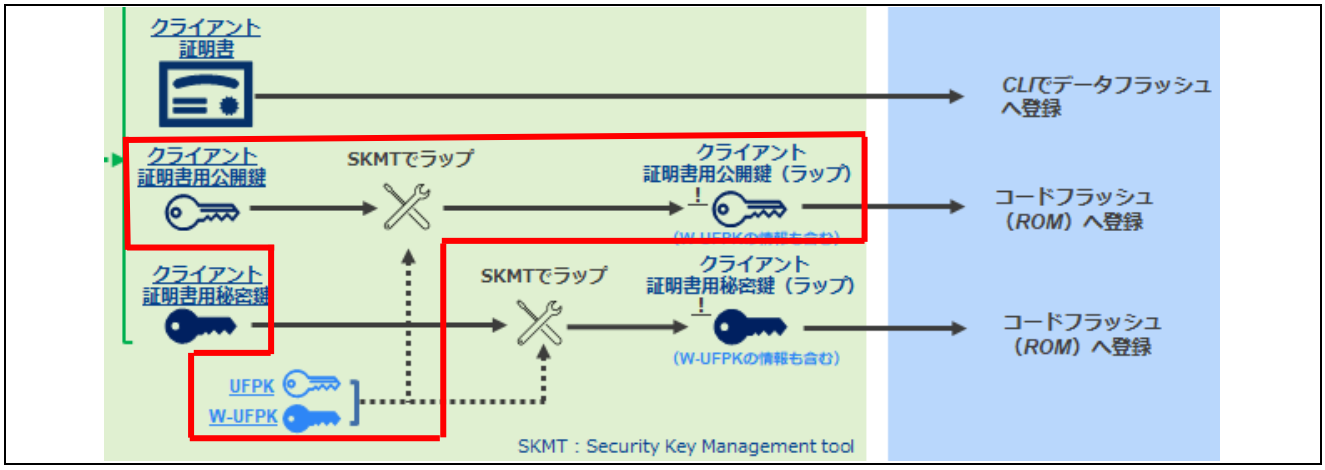


図 5-34 クライアント証明書用公開鍵の生成

以下の鍵ファイルをラップして、プロジェクトに組み込むためのクライアント証明書用の公開鍵を作成します。

/key_crt_sig_generator / client-rsa2048 / xxxx-public.pem.key

【注】 AWS からダウンロードしたものです。xxxx は任意の文字列となります。

① クライアント証明書用公開鍵を生成します。
あらかじめ、xxxx-public.pem.key を「xxxx-public.pem」にリネームしておいてください。
次に[Key Type（鍵の種類）]タブに移動し、(2)と同様に[RSA] の「2048bits, public」が選択されていることを確認します。「Wrapping Key（ラッピング鍵）」、「IV」についても設定は同じとなります。

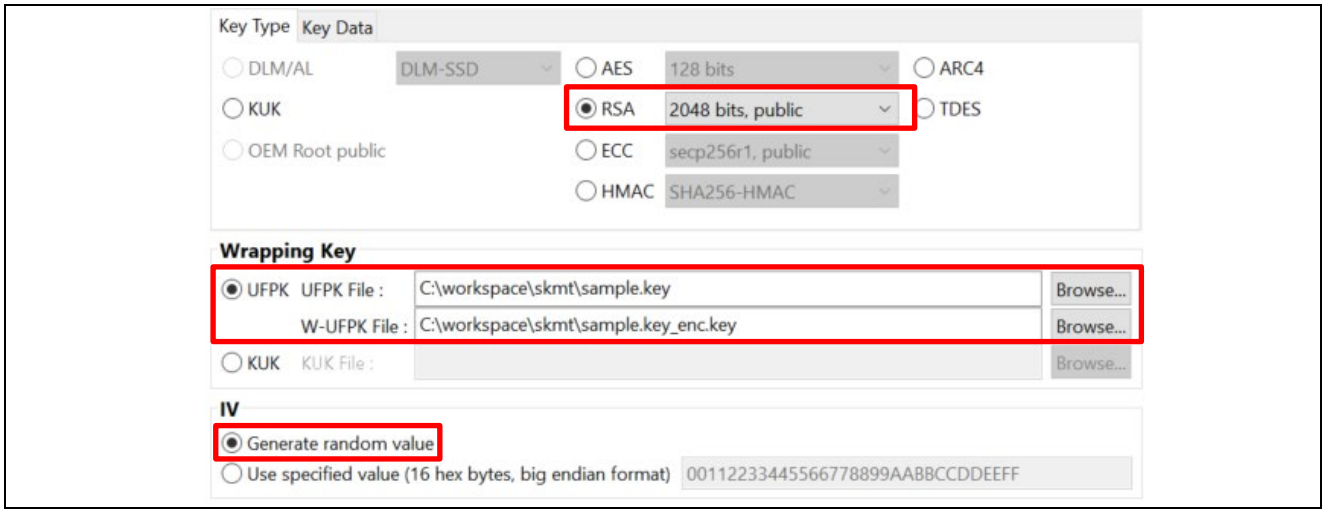


図 5-35 Key Type（鍵の種類）設定確認

② 再び[Key Data]タブに移動します。
[Key Data（鍵データ）]タブをクリックします。
[File（ファイル）]を選択し、[Browse（参照）]ボタンをクリックしてください。
Key data File（鍵データファイル）の選択ダイアログが開くので、ファイルの種類のリストより[PEM 鍵データ(*.pem)]を選択してください。クライアント証明書用公開鍵ファイル(PEM) が表示できるようになるので、/key_crt_sig_generator / client-rsa2048 / xxxx-public.pem を選択して、[Open（開く）]ボタンをクリックします。

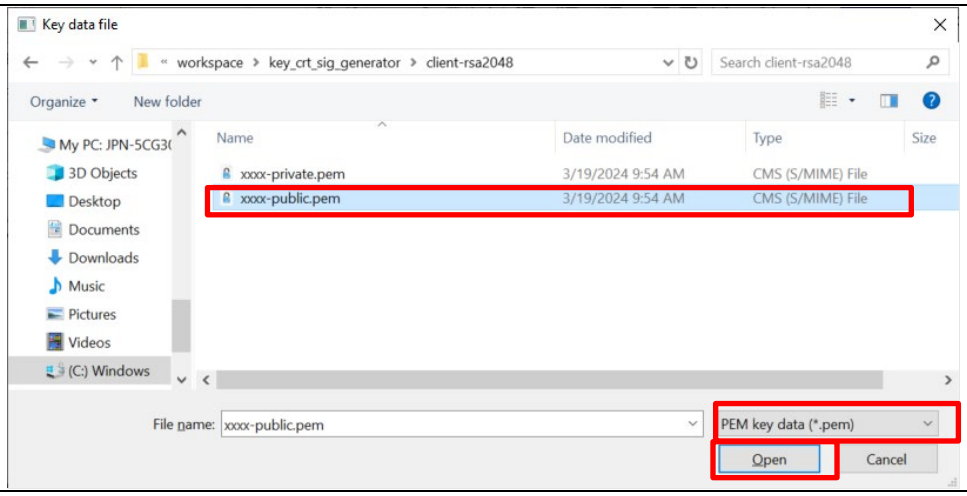


図 5-36 ファイル選択ダイアログ

③ Output（出力）の項目の「Format（フォーマット）」を「C Source（C ソース）」に設定します。
「File（ファイル）」に出力先の任意のフォルダを選択しファイル名に
「**encrypted_user_rsa2048_ne_key2.c**」を入力してください。
また、「Key name」のテキストボックスには「**encrypted_user_rsa2048_ne_key2**」と入力してください。
入力が完了したら、[Generate file（ファイルを生成する）]をクリックして、クライアント証明書用公開鍵データを生成します。

【注】 ファイル名および Key name に入力した「encrypted_user_rsa2048_ne_key2」はソースコード内で使用しているため、変更しないようにしてください。

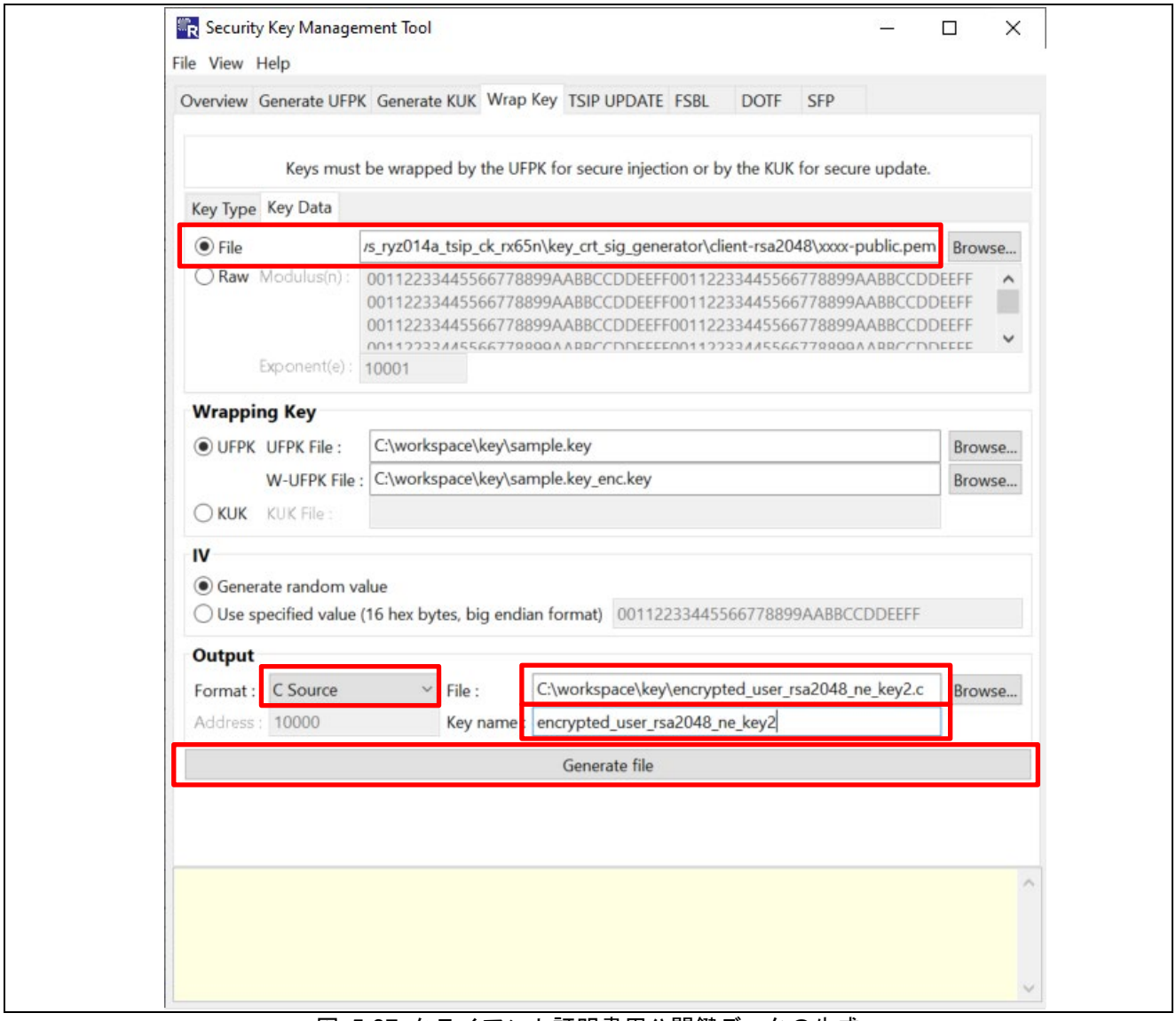


図 5-37 クライアント証明書用公開鍵データの生成

画面下部の処理結果表示部に以下のように表示されたら生成完了です。指定したフォルダに「encrypted_user_rsa2048_ne_key2.c/h」ファイルが出力されていることを確認してください。

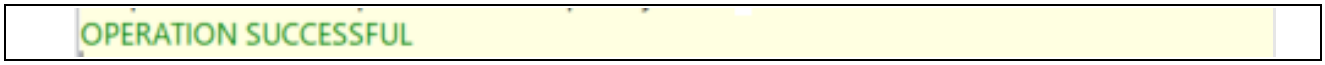


図 5-38 出力結果

(4) クライアント証明書用秘密鍵の生成

クライアント証明書用秘密鍵を生成します。本項の作業は「図 5-1」の以下の赤で囲んだ部分に相当します。

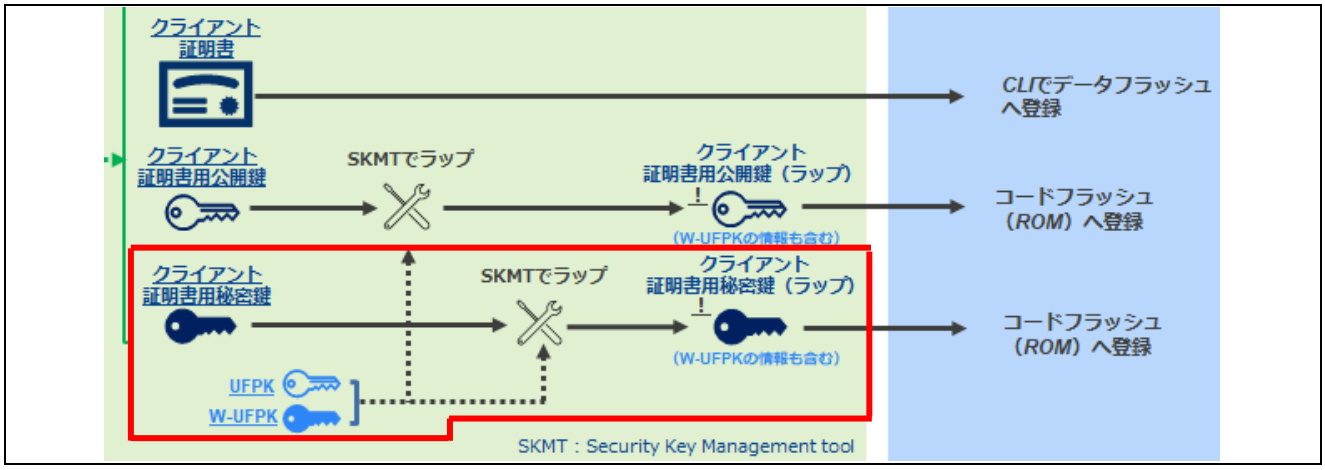


図 5-39 クライアント証明書用秘密鍵の生成

以下の鍵ファイルをラップして、プロジェクトに組み込むクライアント証明書用秘密鍵を作成します。

/key_crt_sig_generator / client-rsa2048 / xxxx-private.pem.key

【注】 AWS からダウンロードしたものです。xxxx は任意の文字列となります。

① クライアント証明書用秘密鍵を生成します。
あらかじめ、xxxx-private.pem.key を「xxxx-private.pem」にリネームしておいてください。
次に[Key Type（鍵の種類）]タブに移動し、[RSA] の「2048bits, private」を選択してください。
Wrapping Key（ラッピング鍵）、IV については「(2)」と同様です。

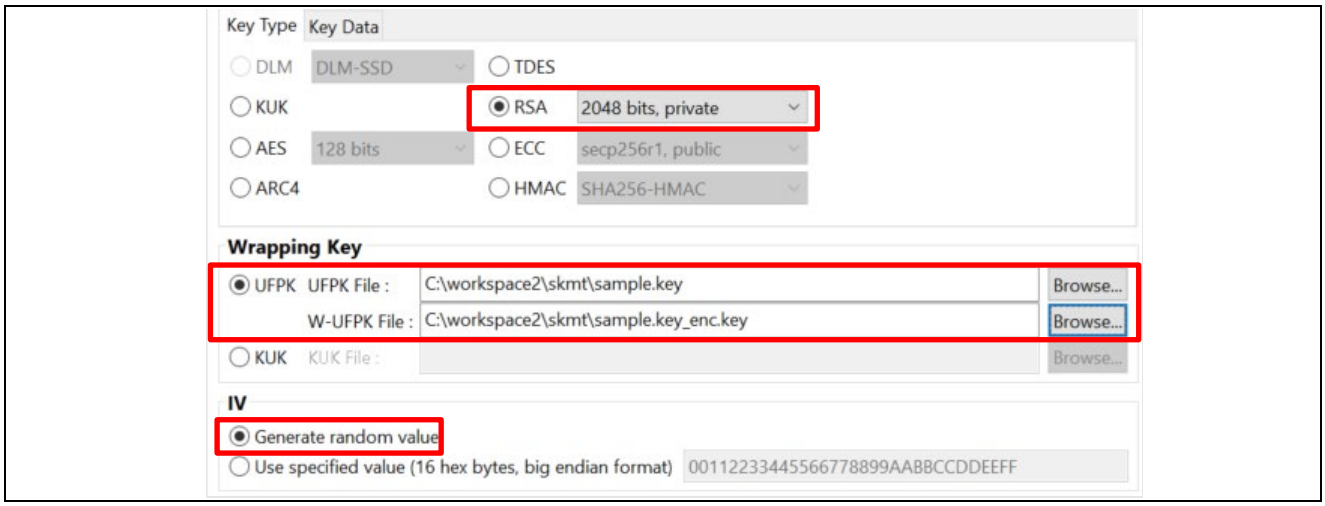


図 5-40 Key Type（鍵の種類）設定確認

② 再び[Key Data]タブに移動します。
[Key Data（鍵データ）]タブをクリックします。
[File（ファイル）]を選択し、[Browse（参照）]ボタンをクリックしてください。
Key data File（鍵データファイル）の選択ダイアログが開くので、ファイルの種類のリストより[PEM 鍵データ(*.pem)]を選択してください。クライアント用秘密鍵ファイル(PEM) が表示できるようになるので、/key_crt_sig_generator / client-rsa2048 / xxxx-private.pem を選択して、[Open（開く）]ボタンをクリックします。

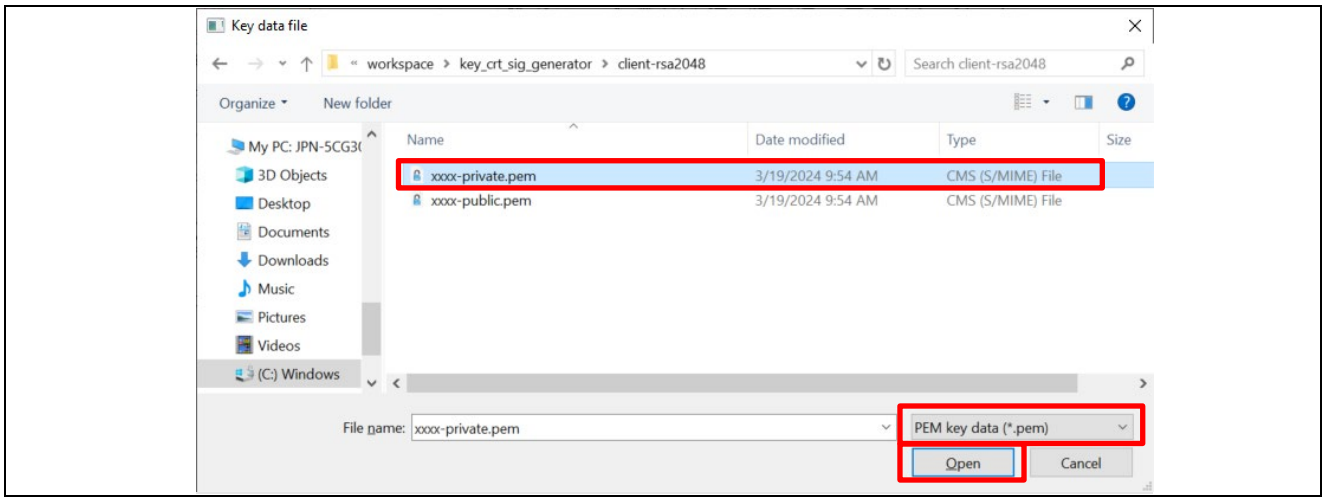


図 5-41 ファイル選択ダイアログ

③ Output（出力）の項目の「Format（フォーマット）」を「C Source（C ソース）」に設定します。
「File（ファイル）」に出力先の任意のフォルダを選択しファイル名に
「**encrypted_user_rsa2048_nd_key.c**」を入力してください。
また、「Key name」のテキストボックスには「**encrypted_user_rsa2048_nd_key**」と入力してください。
入力が完了したら、[Generate file（ファイルを生成する）]をクリックして、クライアント証明書用秘密鍵データを生成します。

【注】 ファイル名および Key name に入力した「encrypted_user_rsa2048_nd_key」はソースコード内で使用しているため、変更しないようにしてください。

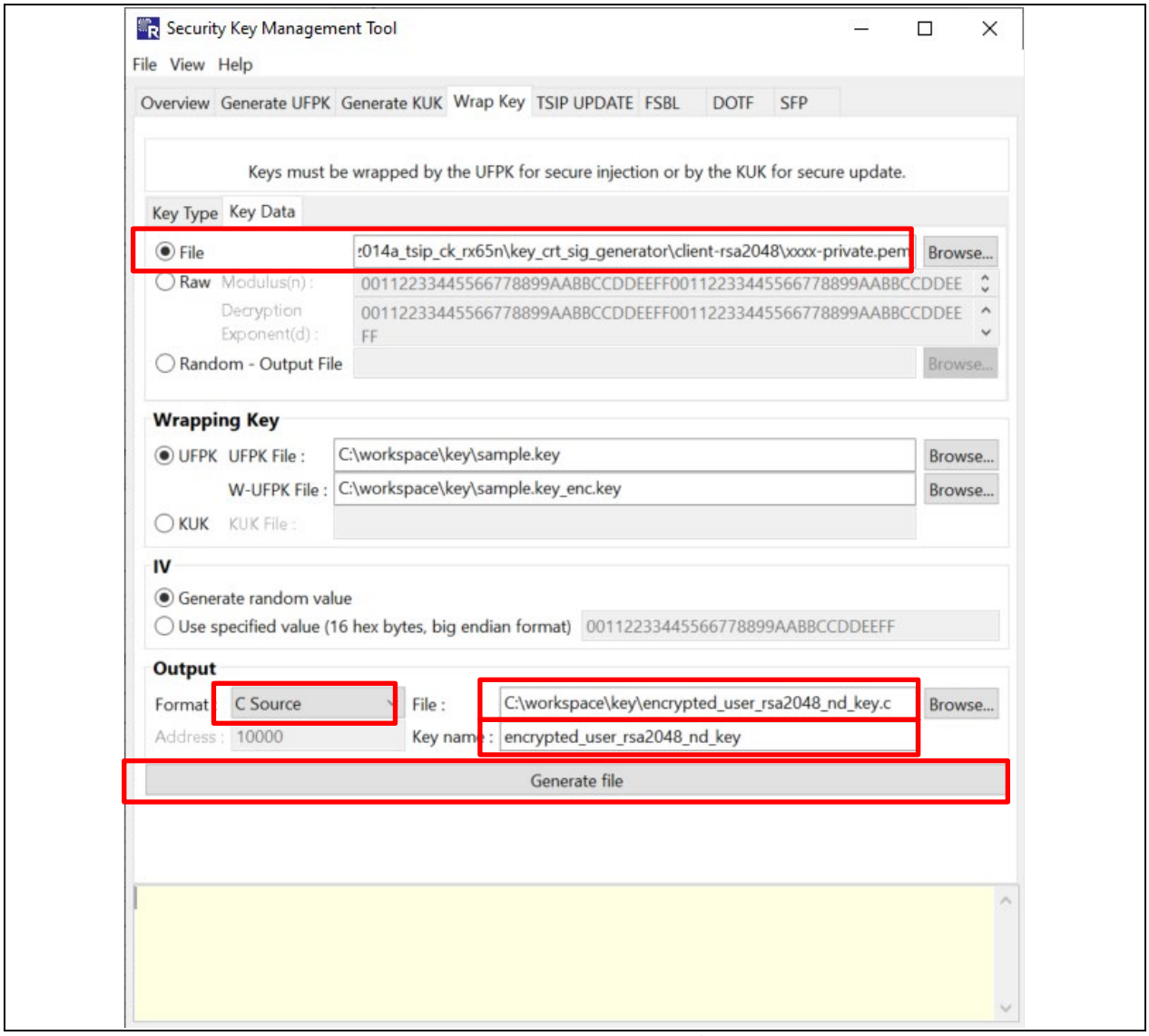


図 5-42 クライアント証明書用秘密鍵データの生成

画面下部の処理結果表示部に以下のように表示されたら生成完了です。指定したフォルダに「encrypted_user_rsa2048_nd_key.c/h」ファイルが出力されていることを確認してください。

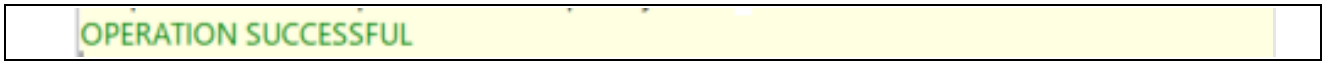


図 5-43 出力結果

「(2)～(4)」の手順で以下のラップされた鍵ファイルのソースコードへの登録を行います。本項の作業は「図 5-1」の以下の赤で囲んだ部分に相当します。

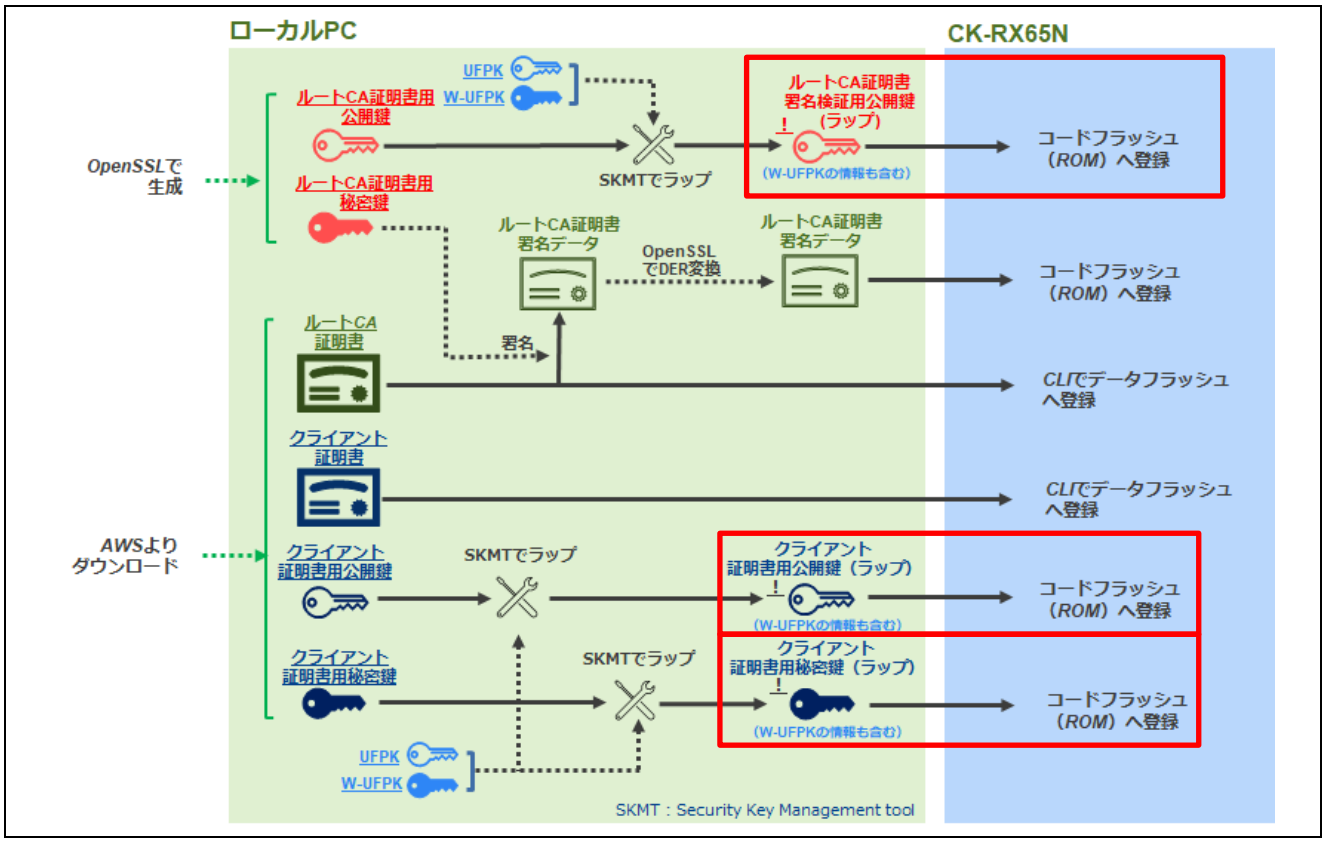


図 5-44 生成した鍵データのソースコードへの登録

以下表に生成された鍵データの一覧を示します。

表 5-4 ラップを行った鍵ファイルの一覧

名称	ファイル名
ルート CA 証明書署名検証用公開鍵	・encrypted_user_rsa2048_ne_key.c ・encrypted_user_rsa2048_ne_key.h
クライアント証明書用公開鍵	・encrypted_user_rsa2048_ne_key2.c ・encrypted_user_rsa2048_ne_key2.h
クライアント証明書用秘密鍵	・encrypted_user_rsa2048_nd_key.c ・encrypted_user_rsa2048_nd_key.h

これら 6 種のファイルは以下のプロジェクトのユーザーデータフォルダに上書きコピーしてください。

```
\iot-reference-rx\Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\src\userdata_tsip
```

5.2 OTA 用鍵ペアと証明書の生成

OTA 実行には、使用するファームウェアが改ざんされていないか検証するために証明書と鍵ペアを使用します。

アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「4.1 鍵ペアと証明書の生成」の手順を実行し、ECDSA の証明書と鍵を生成してください。

ここで作成した以下の ECDSA の証明書（鍵ペア証明書）と公開鍵、秘密鍵をプロジェクトおよび AWS の OTA ジョブ作成時の設定に使用します。

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

6. プロジェクトの構築

「4.デモプロジェクトの準備」で作成したプロジェクトと「5.鍵と証明書の作成」で作成した各種証明書・鍵データを使用し、TSIP を用いた OTA 実行を行うためのプロジェクトを作成します。

本章では OTA 実行に必要な 2 つのファームウェアをプロジェクトから作成します。

- 初期ファームウェア
- 更新ファームウェア

6.1 初期バージョンのファームウェア構築と実行

初期バージョンのファームウェアの構築を行います。

6.1.1 プロジェクトのインポート

「4.デモプロジェクトの準備」の章を参照し、以下の二つのプロジェクトを e² studio にインポートし初期設定を行ってください。

- ブートローダ：boot_loader_ck_rx65n
- デモアプリケーション（Cellular 版）aws_ryz014a_tsip_ck_rx65n

また、「5.鍵と証明書の作成」で作成した以下の 7 つの証明書・鍵ファイルをユーザーデータフォルダに格納してください。

①	AmazonRootCA1_sig_array.txt	：ルート CA 証明書署名データ
②	encrypted_user_rsa2048_ne_key.c	：ルート CA 証明書署名検証用公開鍵（ソースファイル）
③	encrypted_user_rsa2048_ne_key.h	：〃（ヘッダファイル）
④	encrypted_user_rsa2048_ne_key2.c	：クライアント証明書用公開鍵（ソースファイル）
⑤	encrypted_user_rsa2048_ne_key2.h	：〃（ヘッダファイル）
⑥	encrypted_user_rsa2048_nd_key.c	：クライアント証明書用秘密鍵（ソースファイル）
⑦	encrypted_user_rsa2048_nd_key.h	：〃（ヘッダファイル）

ユーザーデータフォルダ

```
\iot-reference-
rx\Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\src\userdata_tsip
```

これらの証明書・鍵データはデモアプリケーションビルド時にプログラムに組み込まれます。

また、②～⑦の鍵ファイルはプログラム実行時に鍵注入に使用される Index データに変換され、初回のみデータフラッシュに書き込まれます。

【注】 1. 2 回目以降の実行時はデータフラッシュへの書き込みは行わず、データフラッシュに保管されている鍵 Index 情報を読み込んで使用します。
また、2 回目以降のプログラム実行でも、データフラッシュがクリアされた場合はプログラム実行時に鍵 Index 情報の再書き込みが行われます。

【注】 2. 鍵ファイルを変更した場合は、プログラム実行時に必ず「6.1.5(6)」を参照しデータフラッシュの削除を行ってください。

6.1.2 プロジェクト設定とビルド

インポートしたプロジェクトに OTA を実行するための設定を行い、ビルドを実行します。

(1) ブートローダへ公開鍵を設定する

「5.2」節で作成した ECDSA 公開鍵「secp256r1.publickey」をテキストエディタなどで開いて、内容をコピーし、ブートローダの以下ファイルに定義されている「CODE_SIGNER_PUBLIC_KEY_PEM」に張り付けます。

```
boot_loader_ck_rx65n\src¥key¥code_signer_public_key.h
```

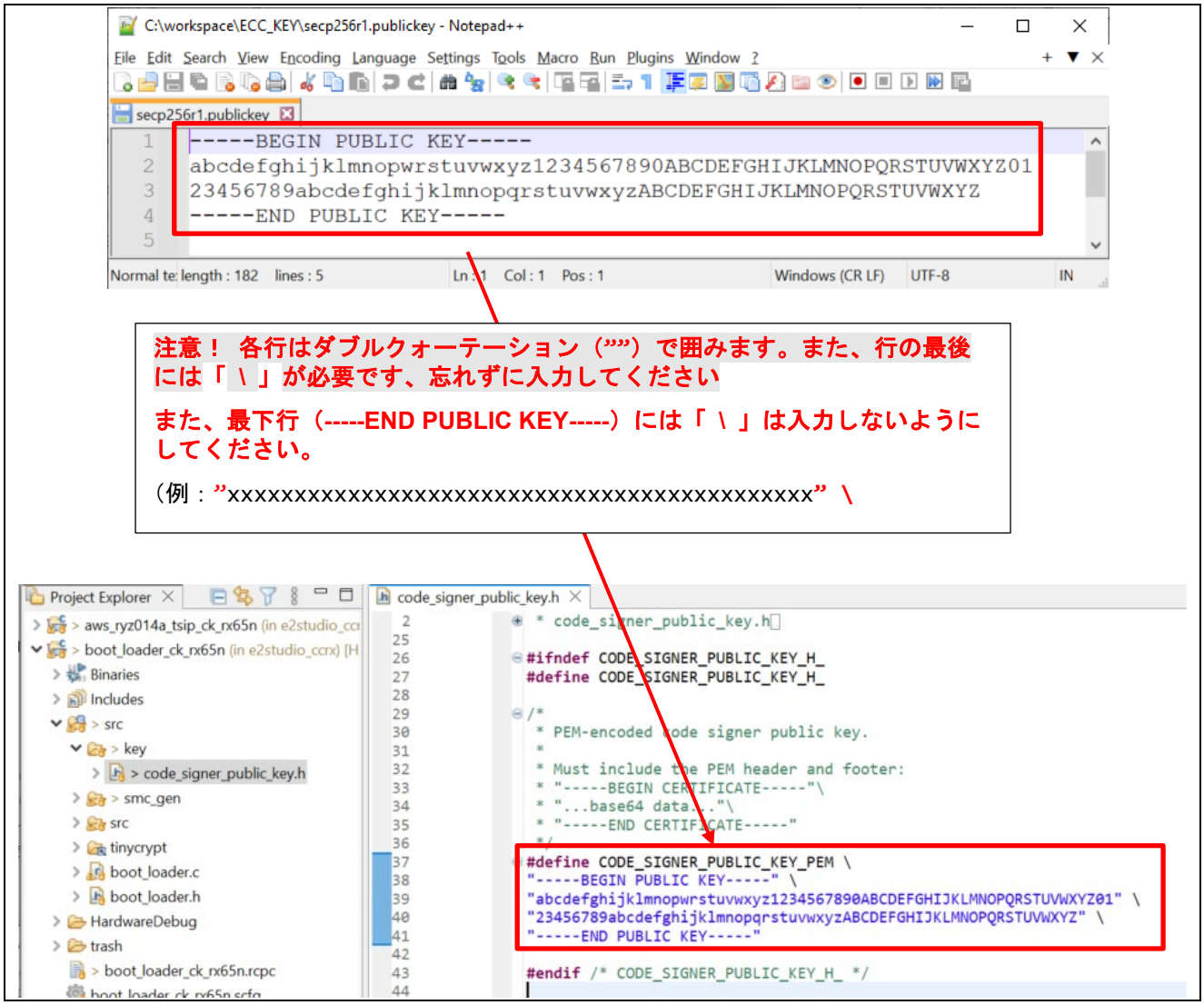


図 6-1 ブートローダへ公開鍵の設定

(2) デモアプリケーションへ OTA アップデートデモの定義を許可にする

aws_ryz014a_tsip_ck_rx65n\src\frtos_confit\demo_config.h にある ENABLE_OTA_UPDATE_DEMO 定義を 1 (許可)に設定してください。（デフォルト 0）

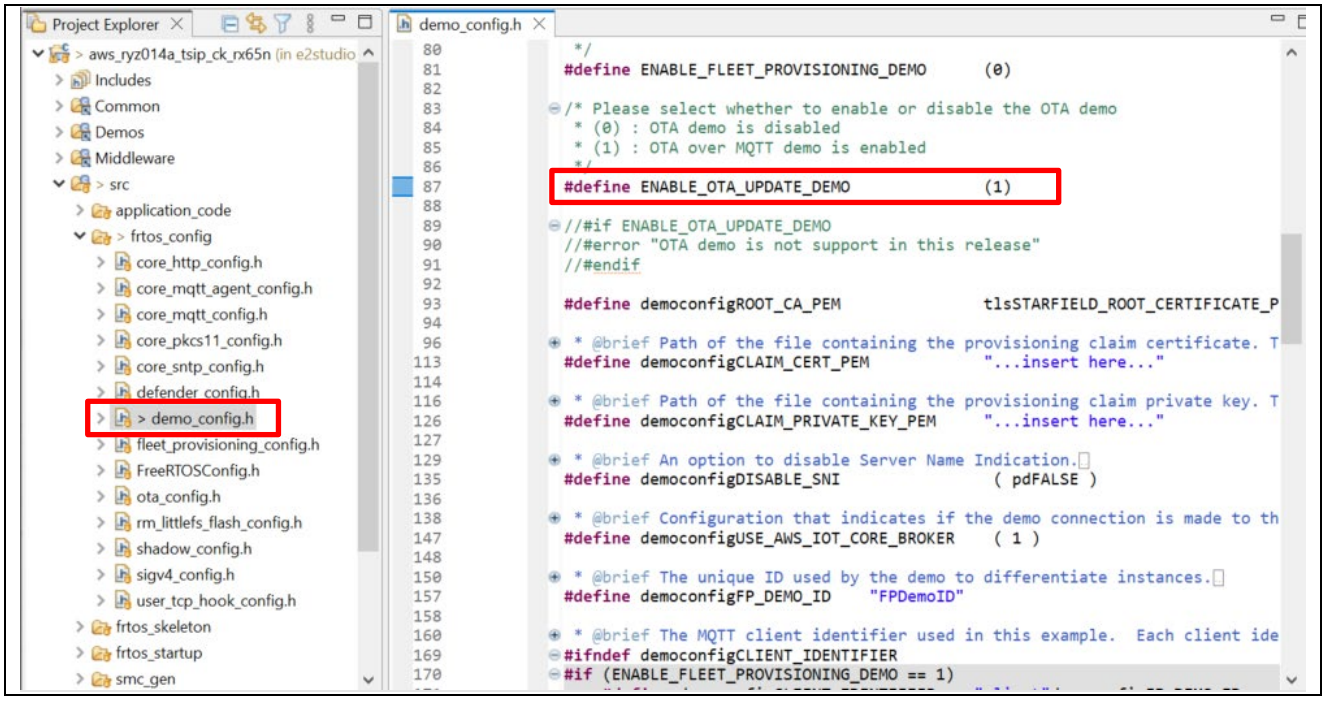


図 6-2 OTA アップデートデモの定義

(3) デモアプリケーションの初期バージョンが 0.92 であることを確認する

aws_ryz014a_tsip _ck_rx65n\src\frtos_config\ demo_config.h のバージョン定義が以下になっていることを確認します。

- APP_VERSION_MAJOR 0
- APP_VERSION_MINOR 9
- APP_VERSION_BUILD 2

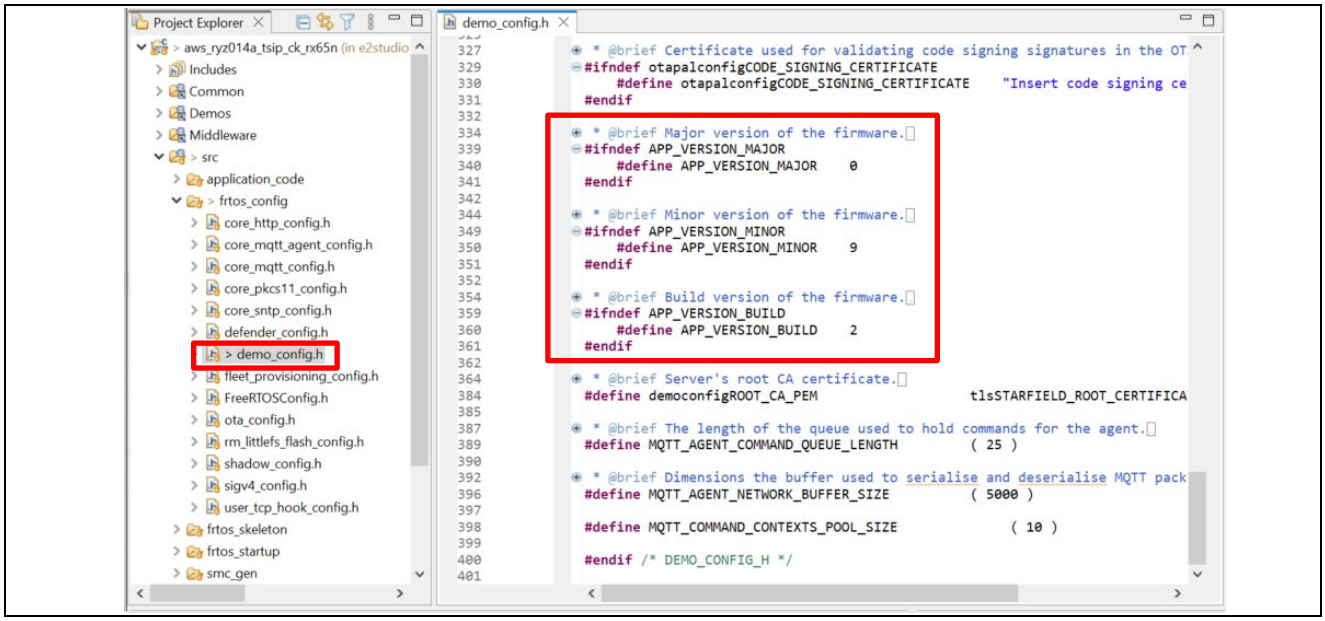


図 6-3 バージョンの設定

(4) RYZ014A Cellular モジュール制御 FIT モジュール(r_cellular)の設定

スマート・コンフィグレータ aws_ryz014a_tsip_ck_rx65n.scfg を開き、Components（コンポーネント）タブを選択します。

r_cellular の Access point name、Access point login ID、Access point password、Authenication protocol type を SIM カードに合わせて設定してください。また、Debug log output level を「3」に設定してください。

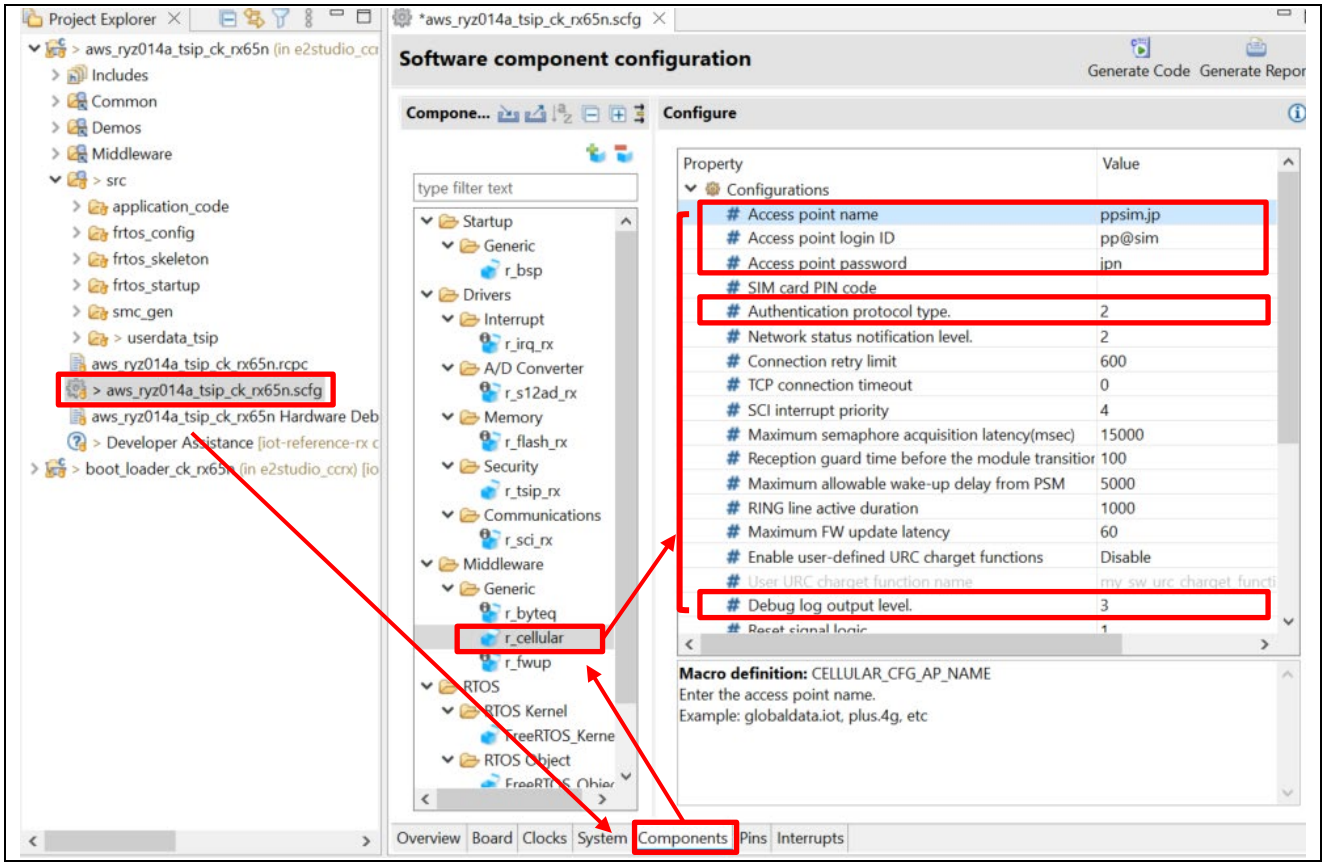


図 6-4 セルラーモジュール（r_cellular）の設定

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ください。スマート・コンフィグレータの変更内容がコードに適用されます。

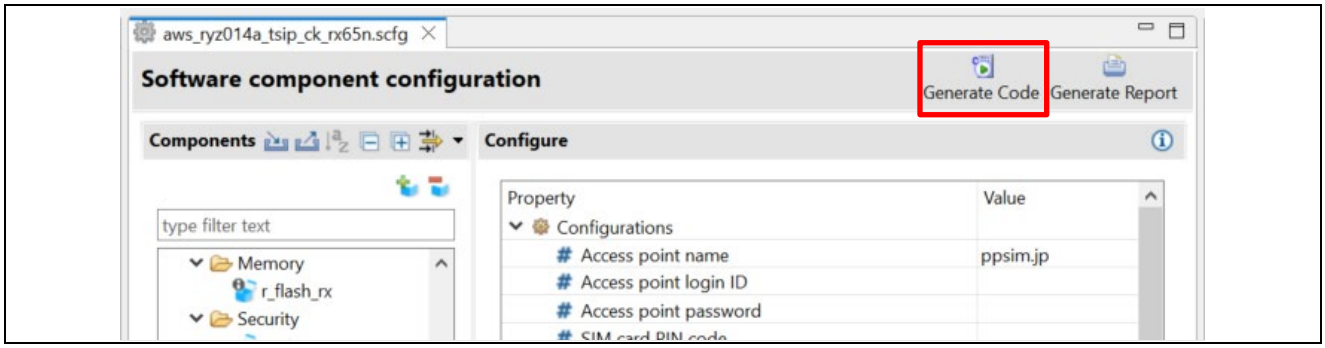


図 6-5 コードの生成

CK-RX65N 付属の SIM カードを使用する場合は、以下アプリケーションノートの 「4.1.5 Activating SIM card」を参照し、SIM カードのアクティベーションを行ってください。

SIM activation, Creating the trial account and using Dashboard with RYZ014A or Ethernet Application for AWS - Getting Started Guide (R01QS0064)

(5) ファームウェアのデバイス設定

① スマート・コンフィグレータ aws_ryz014a_tsip_ck_rx65n.scfg を開き、[Board（ボード）]タブを選択し、Device（デバイス）が「R5F565NEHxFB DUAL」となっていることを確認してください。

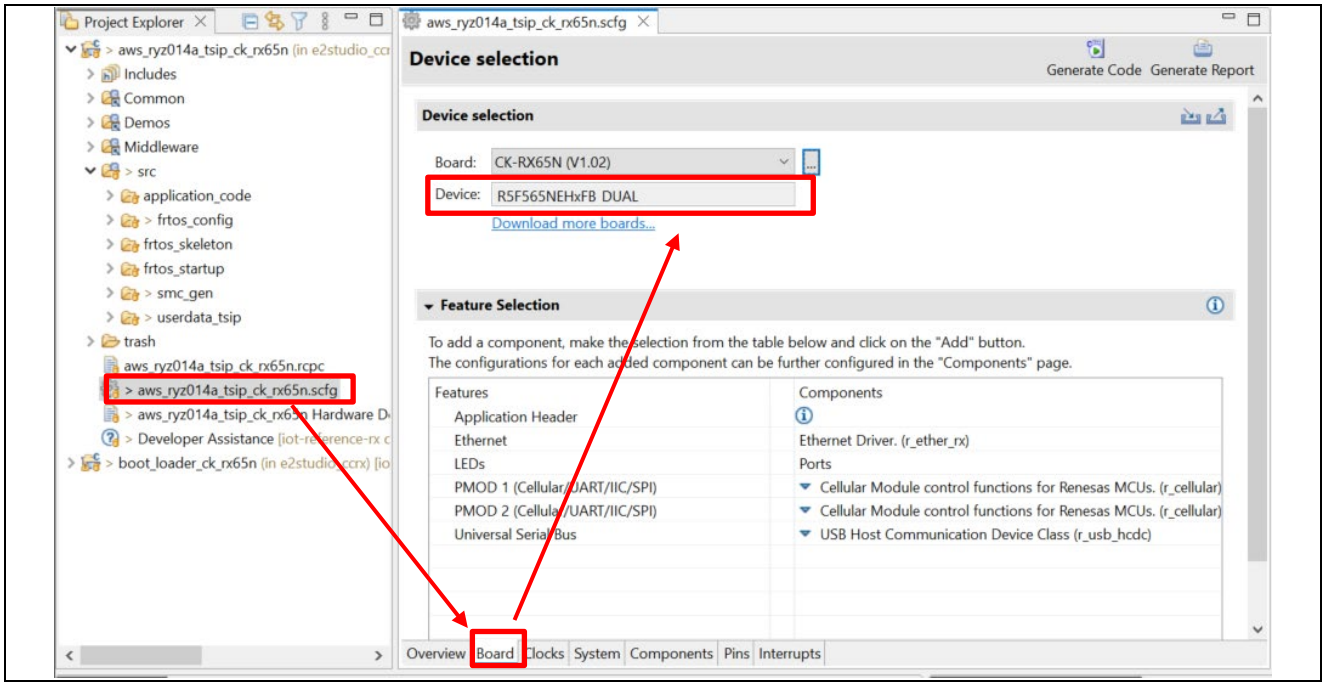


図 6-6 デバイス設定の確認

もし、デバイスが「R5F565NEHxFB」となっている場合は次の手順でデバイスの設定を行ってください。

② Board 選択の「…」をクリックしてください。

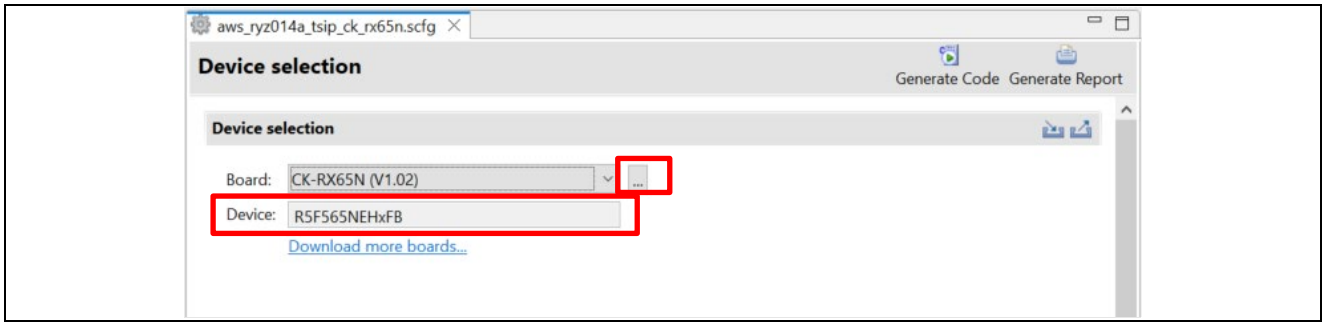


図 6-7 デバイスの変更 1

③ Change Device 画面が表示されます。
ターゲットボードのリストボックスをクリックし、ターゲットボードの一覧から「CK-RX65N(DUAL)」を選択して、[Next（次へ）]ボタンをクリックしてください。

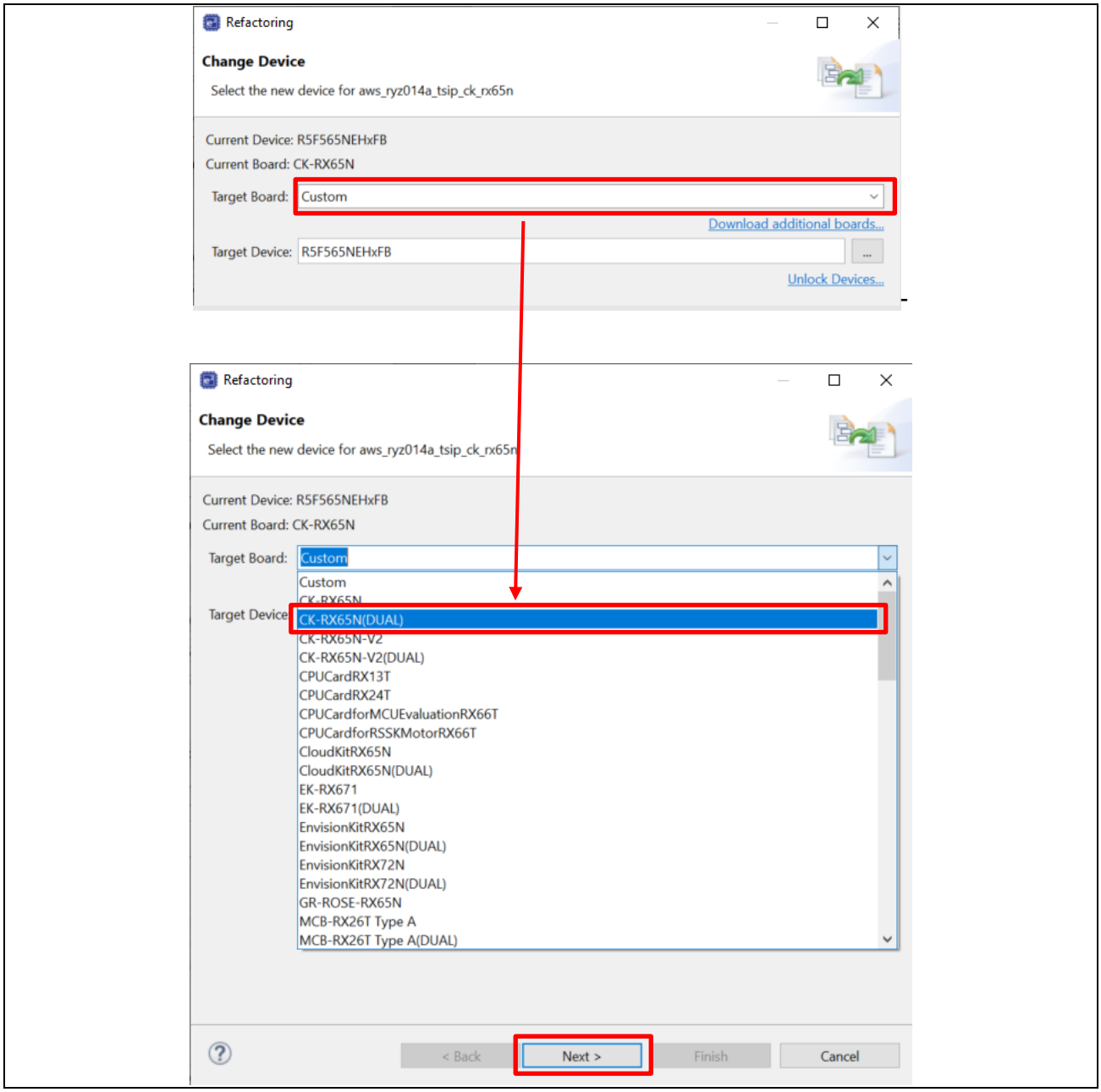


図 6-8 Change Device 画面

④ デバイスの変更を行うと以下の Found problems（検出された問題）の画面になるため[Next（次へ）]をクリックします。

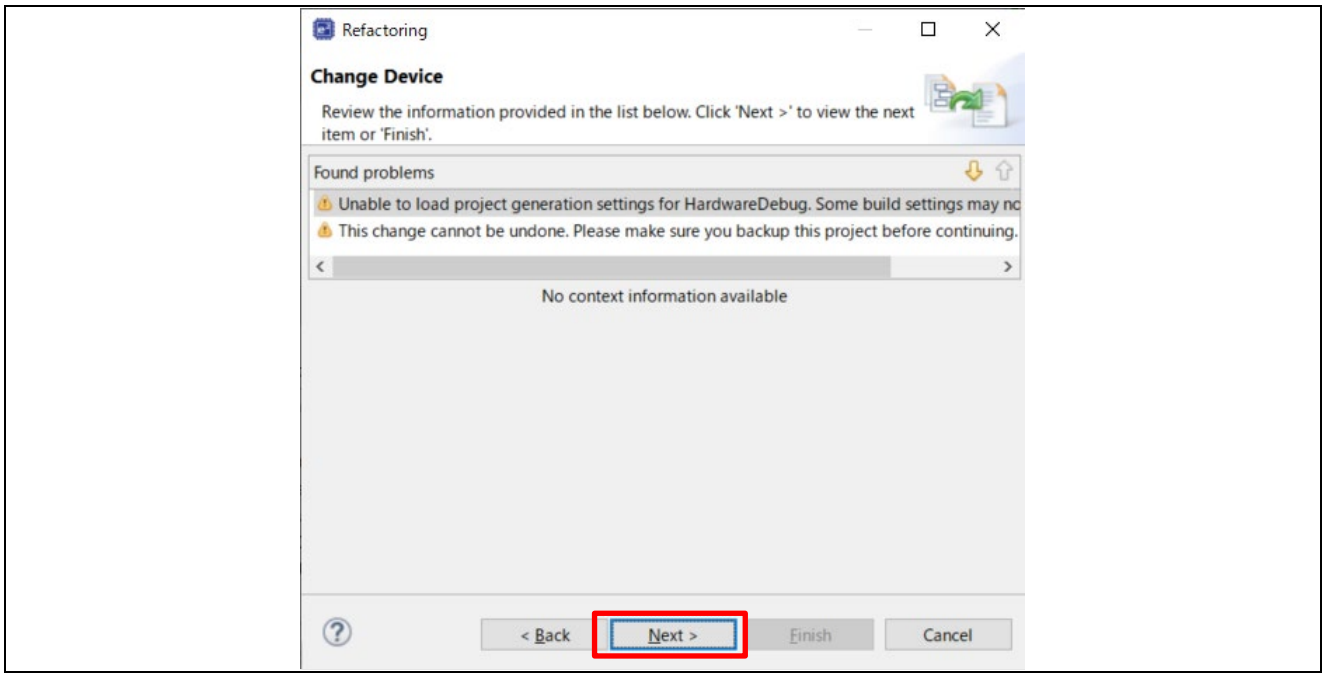


図 6-9 検出された問題画面

⑤ 以下の Change to be performed（実行される変更）画面が表示されるため、[Build Settings]>[HardwareDebug]>[Toolchain Settings]にある、[ROM to RAM mapped section(-rom)（ROM から RAM へマップするセクション）]と[Sections(-start)（セクション）]のチェックを外し、[Finish（終了）]をクリックしてください。

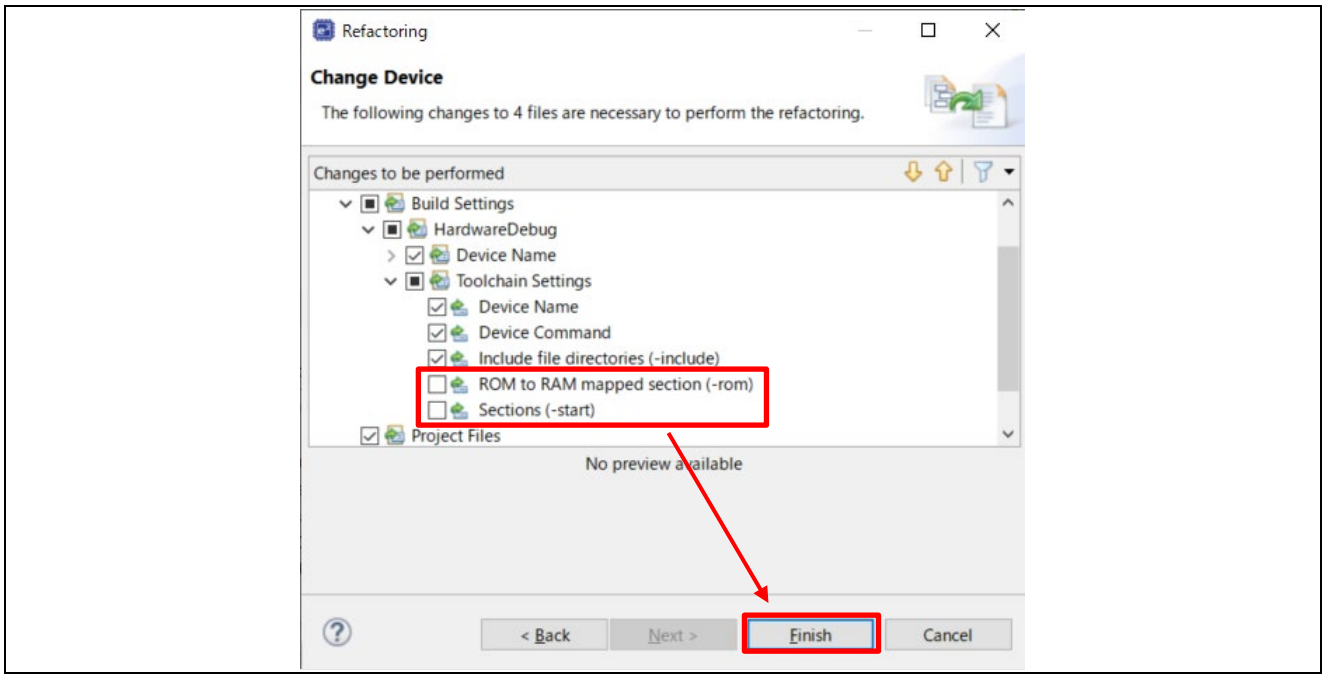


図 6-10 実行される変更画面

ターゲットデバイス変更時に、以下のようにターゲットボードを「CK-RX65N(V1.02)のままにしますか？」の確認ダイアログ画面が表示された場合は[Yes（はい）]ボタンをクリックしてください。

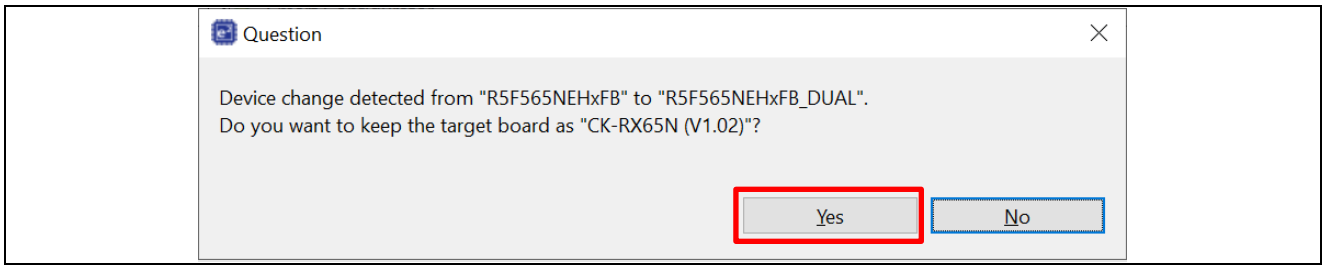


図 6-11 ターゲットボード変更の確認画面

⑥ デバイス設定が「R5F565NEHxFB DUAL」に変更されます。

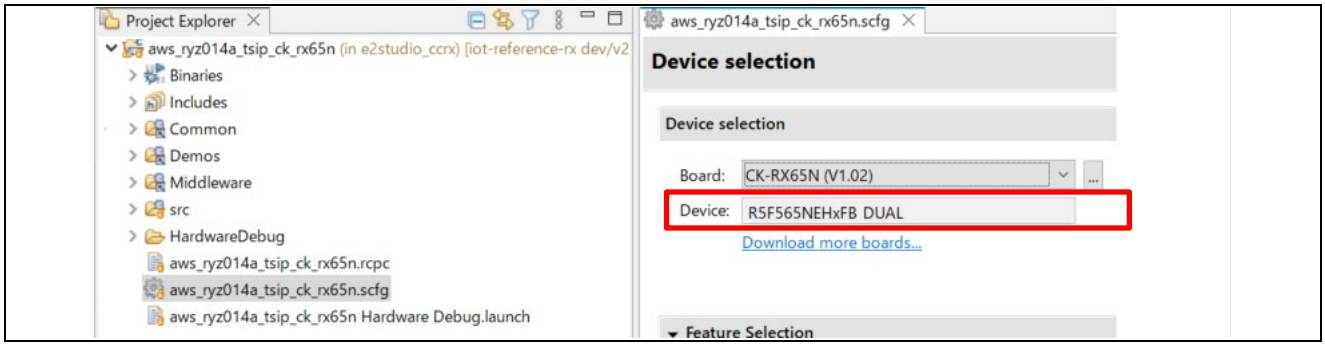


図 6-12 デバイスの変更

(6) ブートローダのデバイス確認

boot_loader_ck_rx65n.scfg を開き、Board タブを選択してください。Device が「(5)」と同様に「R5F565NEHxFB_DUAL」になっていることを確認してください。

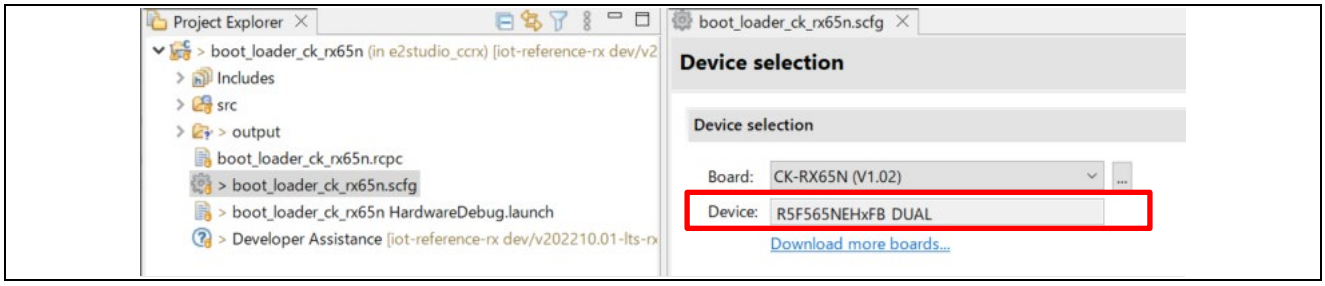


図 6-13 ブートローダのデバイス設定

(7)　ファームウェア(aws_ryz014a_tsip_ck_rx65n)のベクタテーブルのアドレス変更

aws_ryz014a_tsip_ck_rx65n のプロジェクトから、[Project（プロジェクト）] > [Properties（プロパティ）]を選択しプロパティ画面から、
C/C++Build（C/C++ビルド）> Settings（設定）から、Tool Settings（ツール設定）をクリックします。

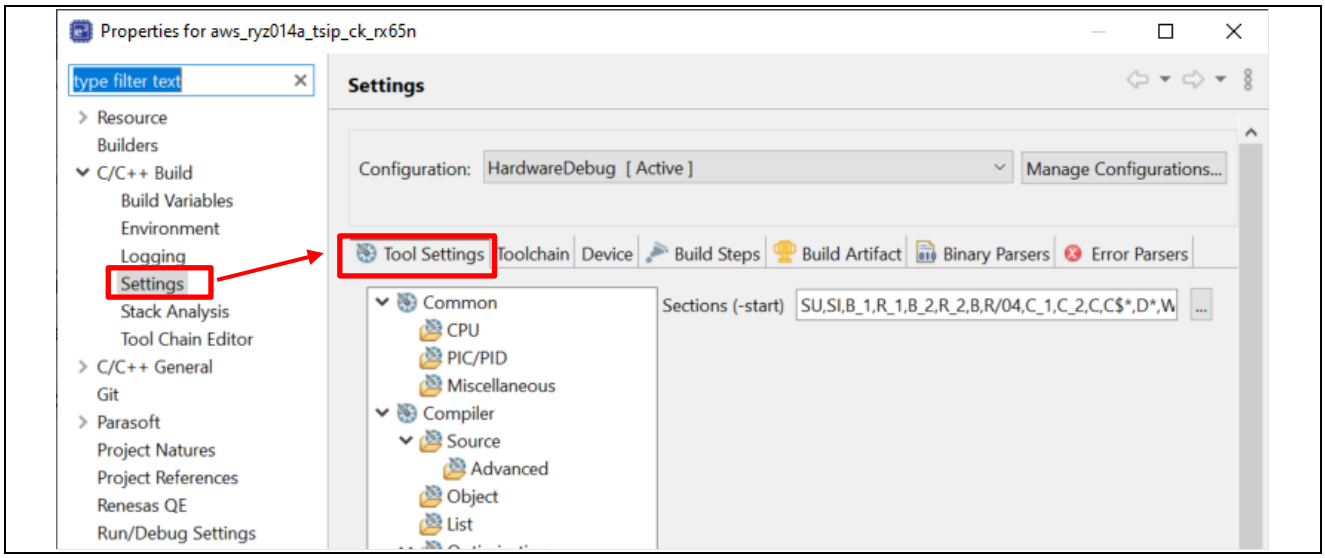


図 6-14 ツール設定画面

設定のツリーから Linker > Section（セクション）をクリックして、Sections の「...」をクリックすると Section Viewer（セクションビューワー）が開きます。

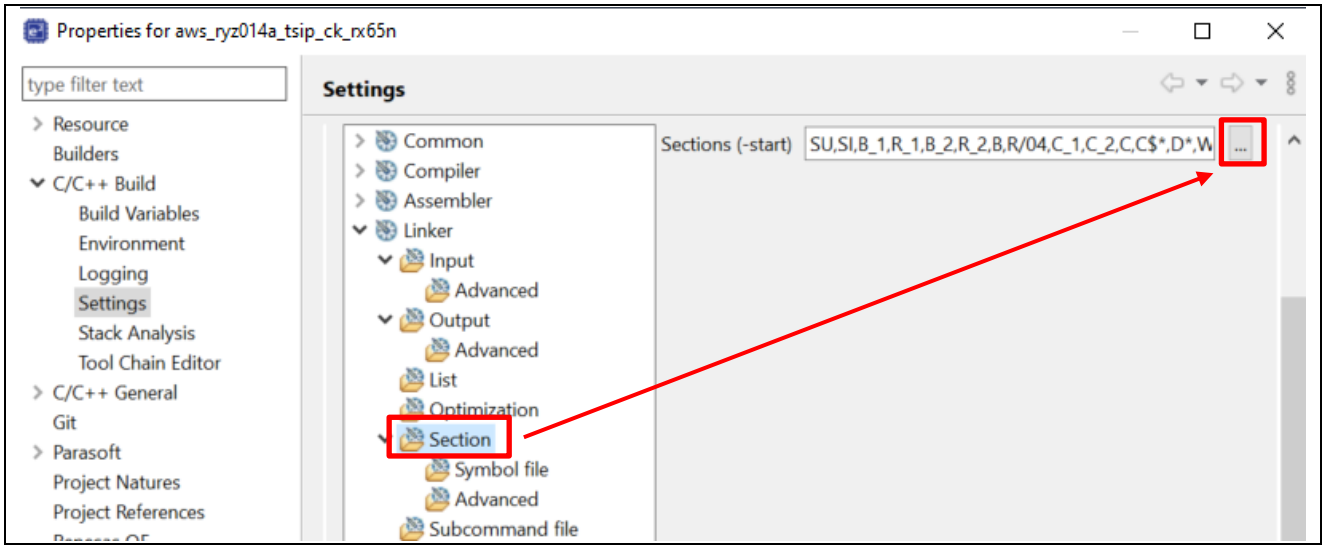


図 6-15 設定のツリー画面

セクションビューワー画面で以下の設定を変更してください。

- EXCEPTVECT　　: 0xFFFFFF80　→　0xFFF**E**FF80
- RESETVECT　　: 0xFFFFFFFC　→　0xFFF**E**FFFC

設定が完了したら[OK]ボタンをクリックしてください。

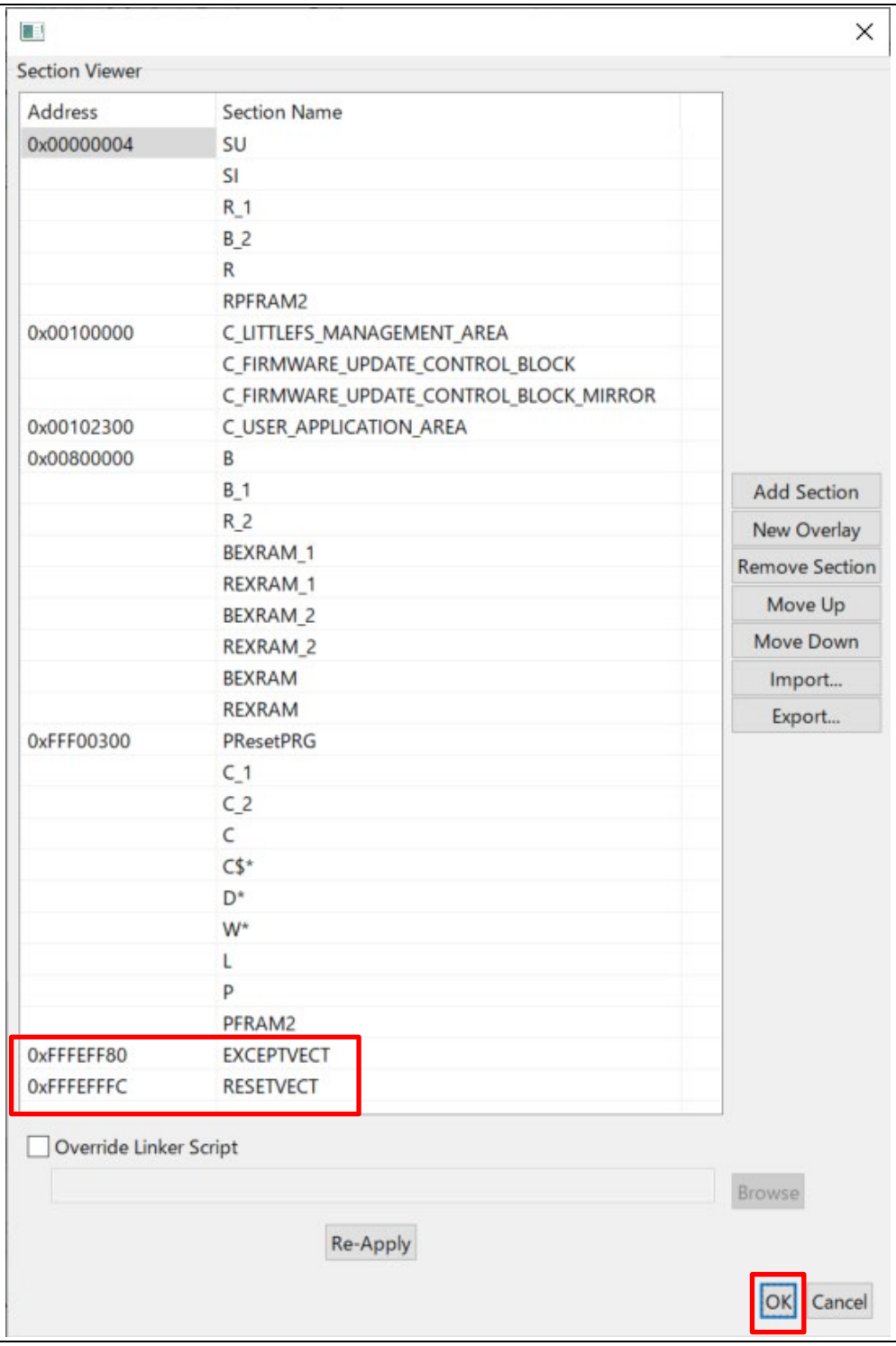


図 6-16 セクションビューワー

(8) プロジェクトのビルド

各設定が完了したらブートローダとデモアプリケーションをビルドし、エラーが出ないことを確認します。ビルドした結果は以下フォルダに格納されます。フォルダ内にはビルドした mot ファイルが生成されます。

- ブートローダ
  `\iot-reference-rx\Projects\boot_loader_ck_rx65n\e2studio_ccrx HardwareDebug`
  mot ファイル：boot_loader_ck_rx65n.mot
- デモアプリケーション
  `\iot-reference-rx\Projects\aws_ryz014a_tsip_ck_rx65n\e2studio_ccrx\HardwareDebug`
  mot ファイル：aws_ryz014a_tsip_ck_rx65n.mot

6.1.3 初期ファームウェアの作成

ブートローダ(boot_loader_ck_rx65n)とファームウェア(aws_ryz014a_tsip_ck_rx65n)を結合して初期ファームウェアを作成します。

ファームウェアの作成には Renesas Image Generator を使用します。アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.2」「2.4」節を参照し、Python と Renesas Image Generator をインストールしてください。

また、ファームウェアのターゲットボードへの書き込みは Renesas Flash Programmer を使用します。

(1) Renesas Image Generator を使用して初期ファームウェアを生成

Renesas Image Generator をインストールしたフォルダに以下のファイルを格納します。

- 「6.1.2」で作成したデモアプリケーションビルド実施結果　aws_ryz014a_tsip _ck_rx65n.mot
- 「6.1.2」で作成したブートローダのビルド実施結果　boot_loader_ck_rx65n.mot
- 「5.2」で作成した ECDSA 秘密鍵　secp256r1.privatekey

コマンドプロンプトを起動し、RenesasImageGenerator ディレクトリへ移動して以下のコマンドを実行すると、userprog.mot ファイルが生成されます。

```
python image-gen.py -iup aws_ryz014a_tsip_ck_rx65n.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o userprog -ibp
boot_loader_ck_rx65n.mot -key secp256r1.privatekey -vt ecdsa -ff RTOS
```

生成にはしばらく時間がかかります。

図 6-17 初期ファームウェアの作成

コマンドラインに「Successfully generated the userprog.mot file.」と表示されたら作成完了です。

初期ファームウェアは以下のファイル名で作成されます。

- userprog.mot

(2) Renesas Flash Programmer を使用し、作成した初期ファームウェアをターゲットボード（CK-RX65N）へ書き込みます。

① フラッシュ書き込みツール(Renesas Flash Programmer)のインストールを行います。
フラッシュ書き込みツールのダウンロードサイトにアクセスし、「Renesas Flash Programmer V3.14.00 Windows」をダウンロードしてインストールしてください。

② アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.5」章を参照して、PC と CK-RX65N v1 の接続を行ってください。

③ Renesas Flash Programmer を起動し、デバイスイレーズプロジェクト「erase.rpj」を開きます。
「erase.rpj」プロジェクトは、サンプルプログラムの以下フォルダに格納しています。

```
\Projects\aws_ryz014a_tsip_ck_rx65n\flash_project\erase_from_bank1
```

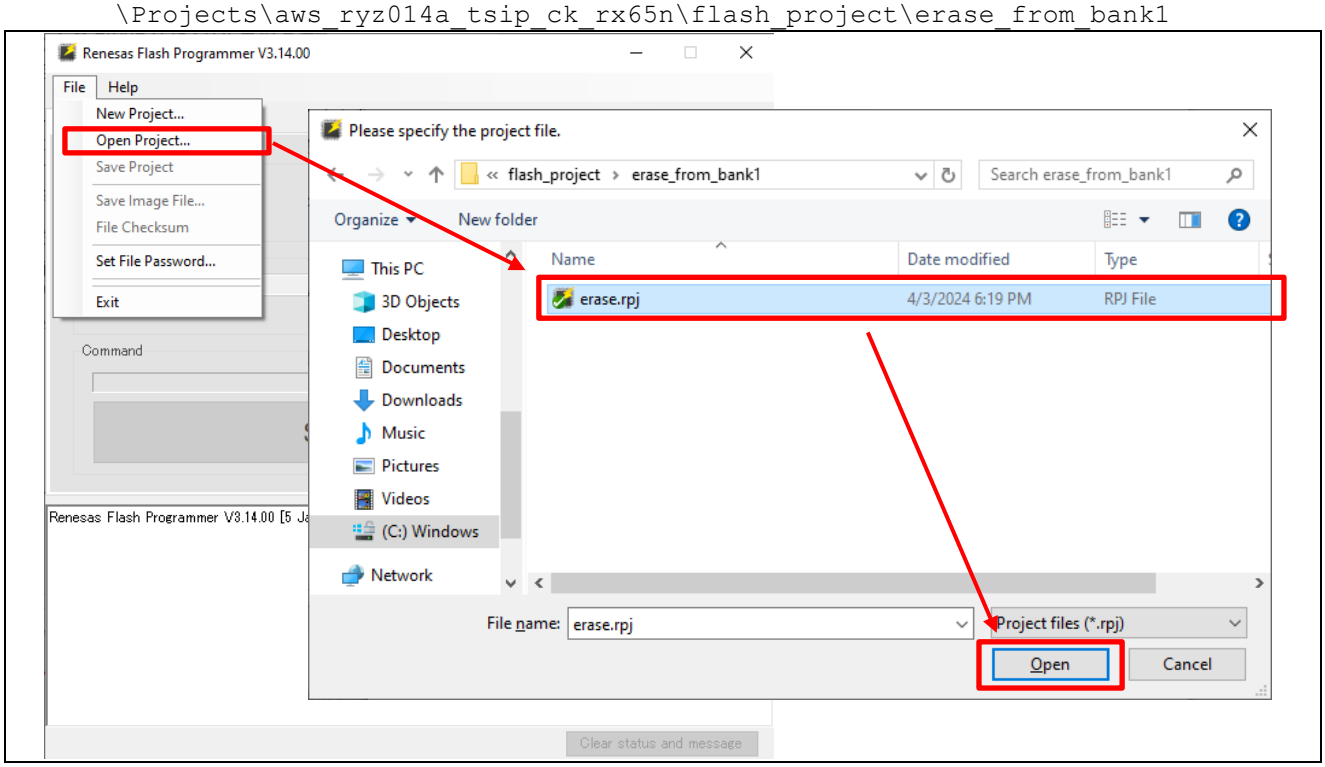


図 6-18 erase.rpj のオープン

④　[Start（スタート）]ボタンをクリックし、デバイスのイレーズを実施します。

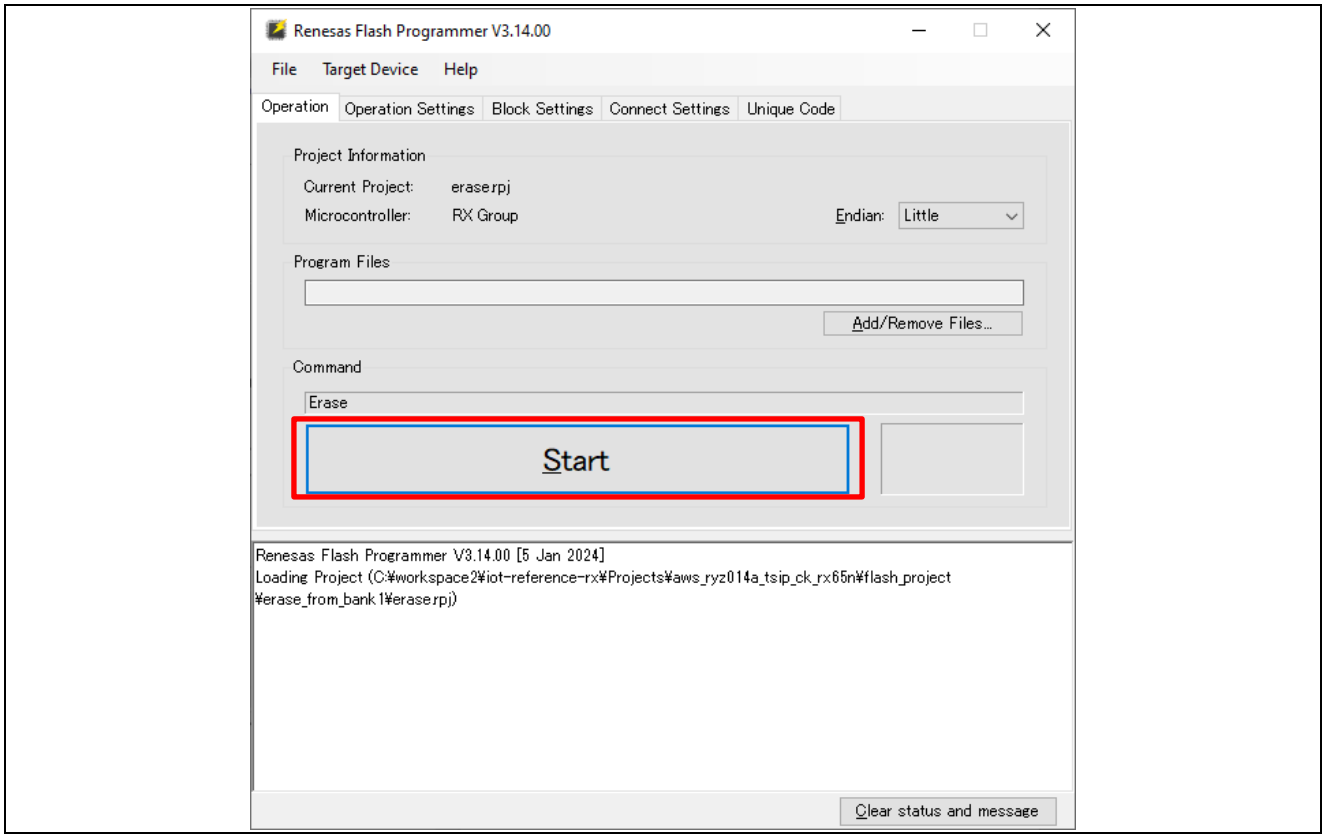


図 6-19 デバイスのイレーズの開始

なお、「エラー(E3000107): デバイスが接続情報と一致しません」が出力された場合は、次の⑤へ進んでください。

⑤　フラッシュ書き込みプロジェクト「flash_project.rpj」プロジェクトを開きます。
「flash_project.rpj」プロジェクトは、サンプルプログラムの以下フォルダにあります。

```
\Projects\aws_ryz014a_tsip_ck_rx65n\flash_project\
```

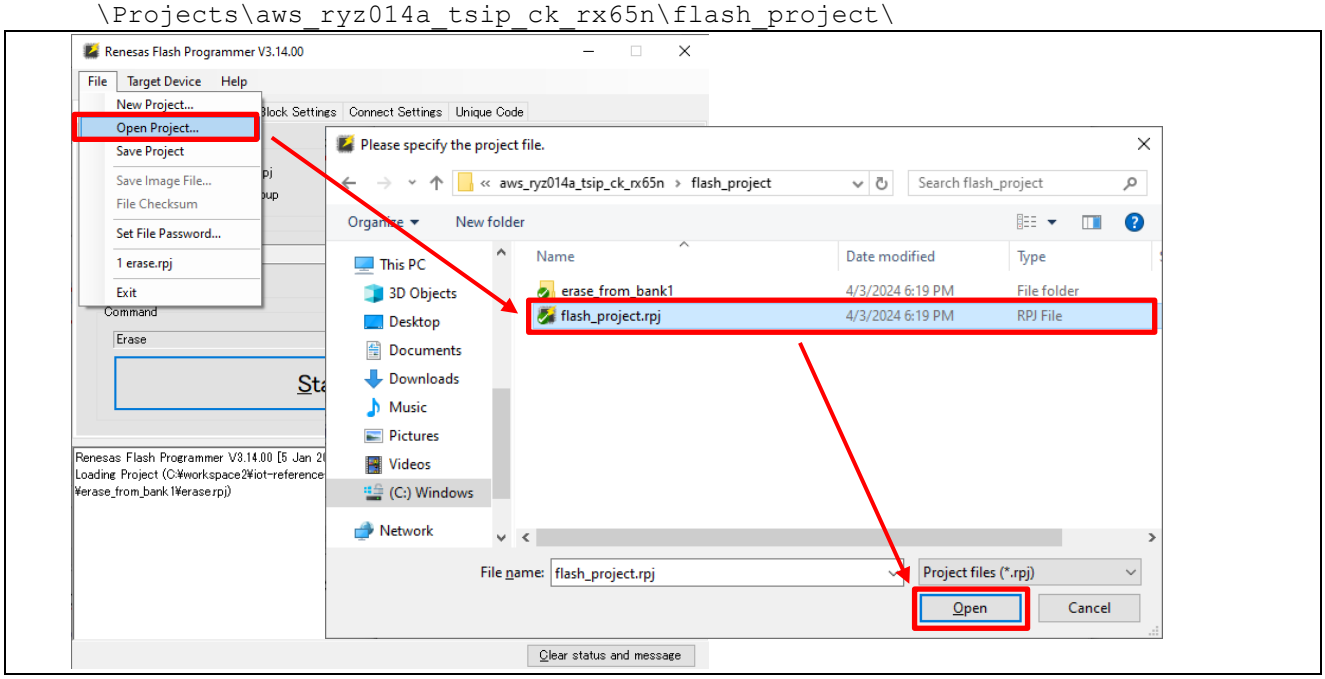


図 6-20 flash_project.rpj のオープン

⑥ 書き込みを行うファームウェアを選択します。
[Add/Remove files（ファイルの追加と削除）]ボタン > [Add Files（ファイルの追加）]をクリックしてファイル選択ダイアログで「6.1.3(1)」で作成した初期ファームウェア(userprog.mot)を選択してください。

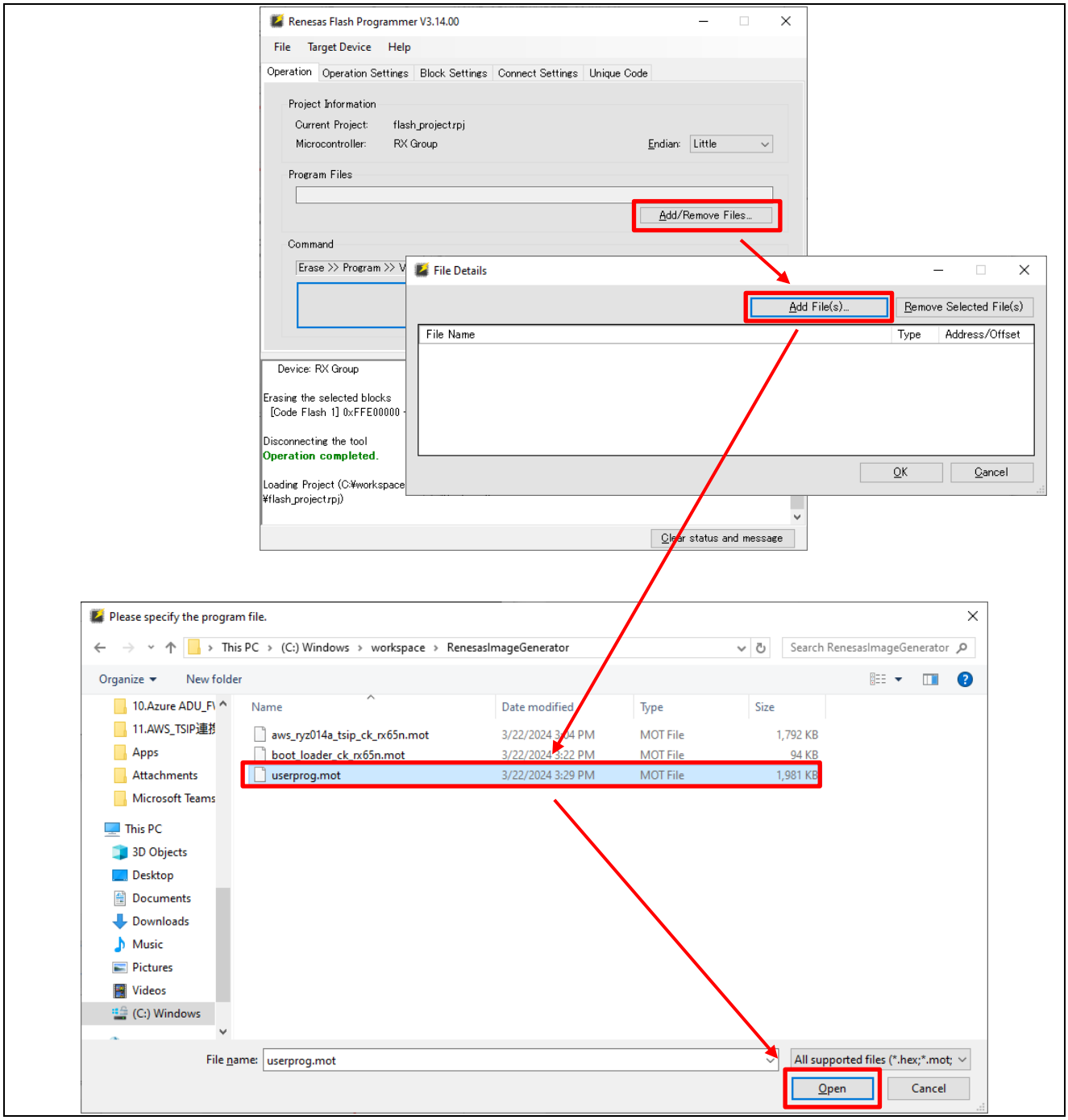


図 6-21 初期ファームウェアの指定

⑦　[Start（スタート）]ボタンをクリックし、ファームウェアの書き込みを行います。

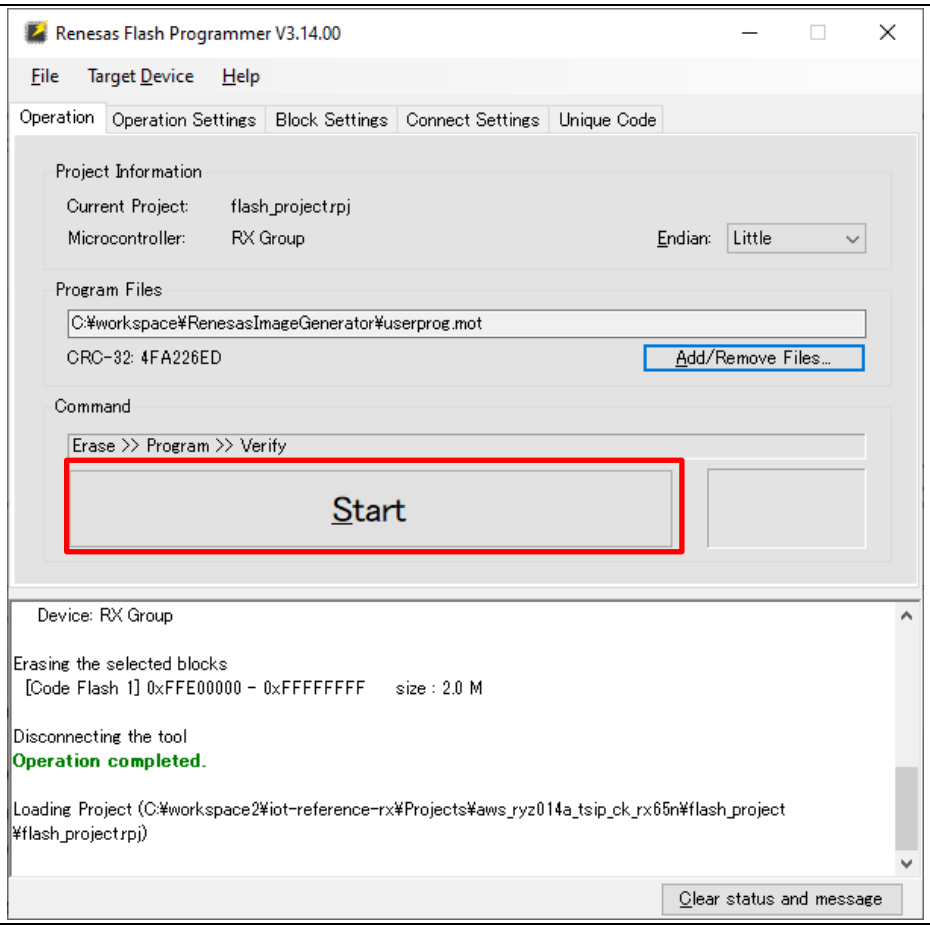


図 6-22 ファームウェアの書き込み

書き込み開始時に以下の「Authentication（認証）」画面が表示された場合は、設定した ID コードを入力して[OK]ボタンをクリックしてください。
ID コードが未設定の場合は初期値のままとしてください。

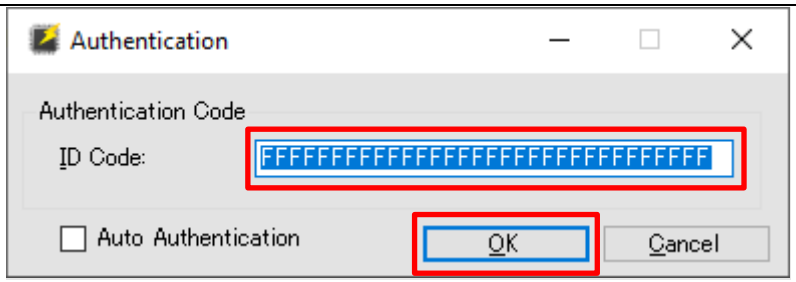


図 6-23 ID コード認証画面

ファームウェアの書き込みが開始されます。画面下部に以下のように「Operation completed（操作が完了しました）」と表示されたら書き込みは完了です。

```
[Config Area 1] 0xFE7F5D00 - 0xFE7F5D2F    size : 48
[Config Area 1] 0xFE7F5D40 - 0xFE7F5D7F    size : 64

Verifying data
[Config Area 1] 0xFE7F5D00 - 0xFE7F5D2F    size : 48
[Config Area 1] 0xFE7F5D40 - 0xFE7F5D7F    size : 64

Disconnecting the tool
Operation completed.
```

図 6-24 ファームウェア書き込み完了

また、書き込みの際に以下のような接続時に通信エラーが出る場合があります。この場合は「Start（スタート）」ボタンをクリックして再書き込みを実施してください。再書き込みしてうまく接続できない場合は、一度ターゲットボードの USB ケーブルを外して再接続してください。

```
Disconnecting the tool
Error(E4000004): A framing error occurred while receiving data. (BFW: 0354)
Operation failed.
```

図 6-25 通信エラーの例（フレーミングエラー）

ターゲットボード（CK-RX65N）の J16 を[RUN]側に接続しデモを実行した後で、再度初期ファームウェアを Renesas Flash Programmer から書き込む際に、CK-RX65N の J16 を[DEBUG]側に切り替えを忘れていると以下のエラーが表示されます。

Renesas Flash Programmer にてファームウェアの書き込みを行う際は、J16 を[DEBUG]に設定してから行ってください。

```
Connecting the tool
Error(E3000201): Cannot find the specified tool.
Operation failed.
```

図 6-26 接続エラーの例

6.1.4　初期ファームウェアの実行

AWS IoT の情報は aws_ryz014a_tsip_ck_rx65n を動作させて、ターミナルソフト 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。アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.1」節を参照し、Tera Term のインストールを行ってください。

(1) Tera Term を起動して、メニューの File > New Connection…から、Serial を選択して OK をクリックします。

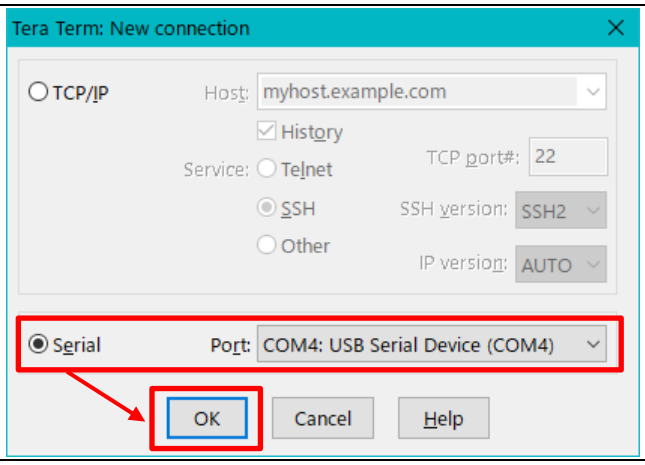


図 6-27 シリアルの設定

(2) メニューの Setup（設定） > Terminal（端末）を開き、New-line（改行コード）の Receive（受信）を Auto、Transmit（送信）を CR+LF を選択して OK をクリックします。

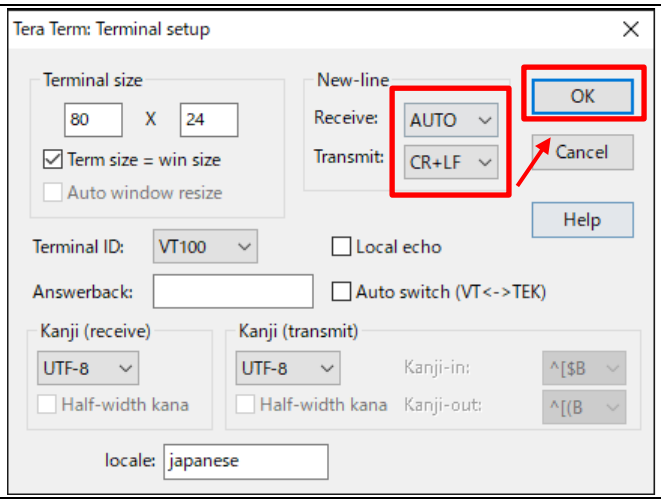


図 6-28 ターミナルの設定

(3) メニューの Setup（設定） > Serial port（シリアルポート）を開いて Speed（速度）を 115200 に設定して New setting（現在の接続を再設定）をクリックします。その他の設定は初期値のままとしてください。

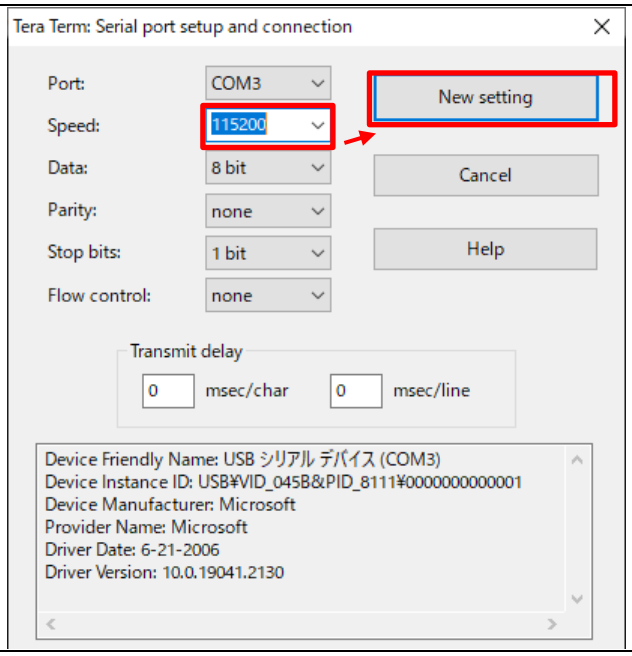


図 6-29 通信速度の設定

(4) CK-RX65N の J16 を RUN 側に接続し、RESET SW を押します。ハードウェアリセット後プログラムが実行されます。
プログラム実行後は動作状況が Tera Term の画面に表示されます。

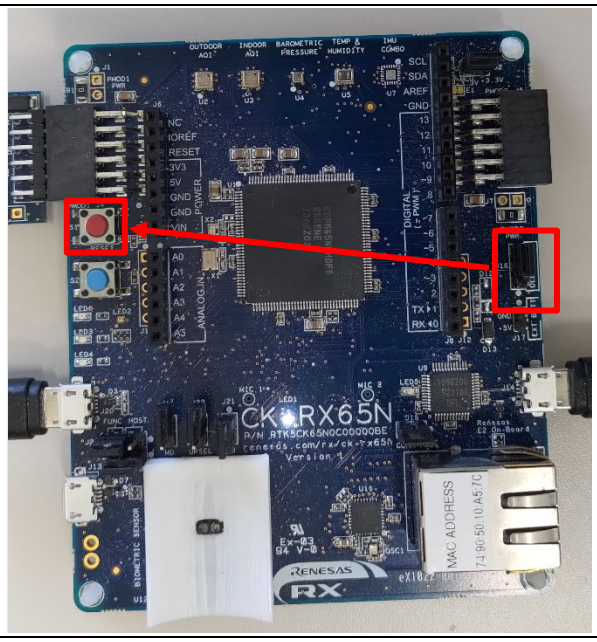


図 6-30 ハードウェアリセットとプログラム実行

(5) ブートローダが起動し、検証後にデモアプリケーションが起動します。
デモアプリケーション起動後にメニューが表示されるので、10 秒以内に「CLI」と入力して Enter キーを押して CLI モードに切り替えてください。CLI モードではコマンド入力でプログラムへ各種情報を登録できるようになります。
また、「CLI」を入力せずに 10 秒以上待つと AWS へ接続するシーケンスに遷移します。

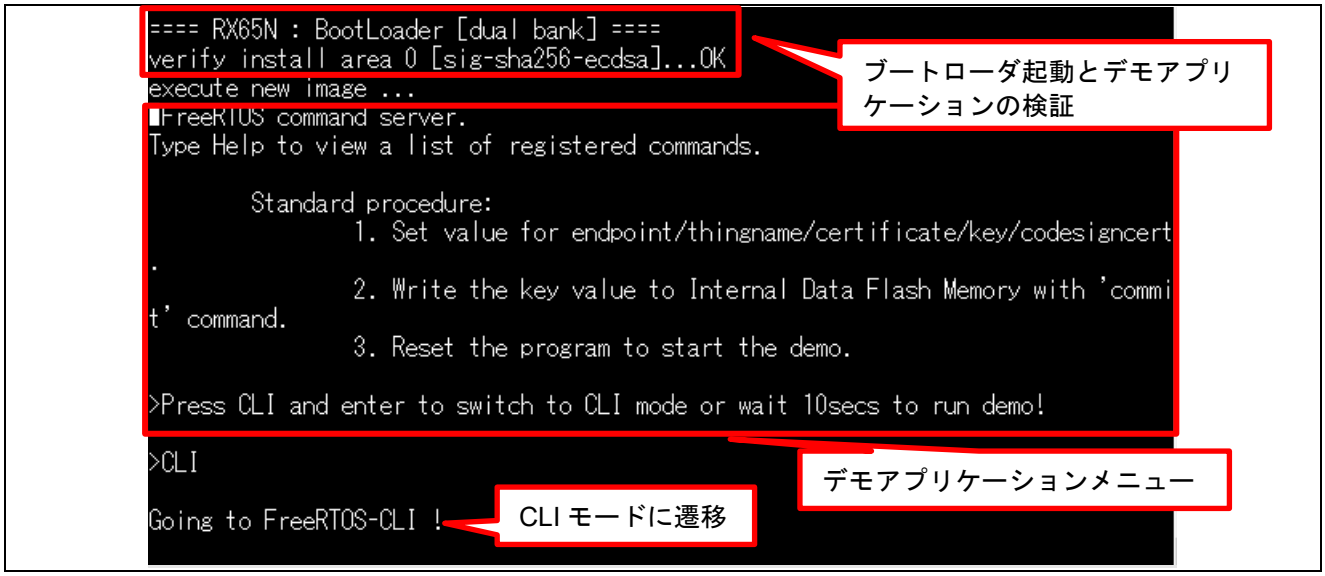


図 6-31 デモアプリケーション実行画面

6.1.5 AWS IoT 情報の登録

AWS へ TSIP を使用した TLS 通信で接続するために必要な各種情報をプログラムに登録します。初期ファームウェアを実行後、CLI モードに入り各種設定を入力します。
CLI モードにて各コマンドを入力することで設定値を入力したり、設定値を確認したりすることができます。
また、CLI モードで入力した設定値はデータフラッシュに保存されるため、ターゲットボードの電源を切っても保持されます。

(1) AWS 接続情報を設定

「3.AWS の設定」で設定したモノの名前、およびエンドポイントを CLI モードで登録します。

Tera Term にて CLI モードに入り、以下のコマンドを実行します。

```
conf set thingname [モノの名前] [enter]
conf set endpoint [エンドポイント名] [enter]
```

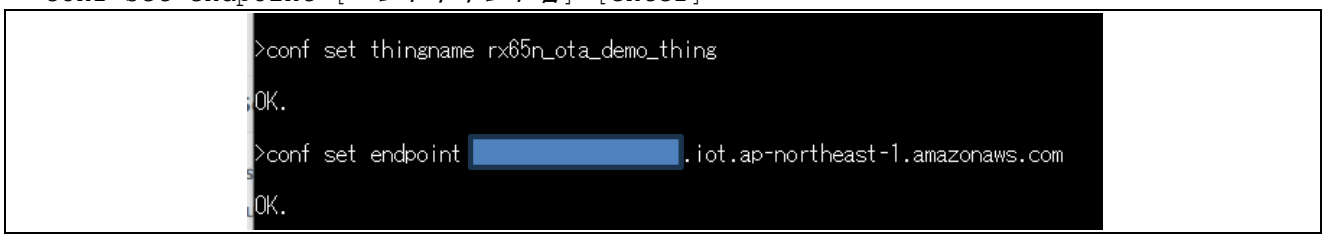


図 6-32 AWS 接続情報の CLI 入力

(2) クライアント証明書の登録

「5.1.3 RSA の鍵ペアとクライアント証明書の入手」で AWS よりダウンロードしたクライアント証明書を CLI モードで登録します。

Tera Term にて「conf set cert 」と入力したのち、クライアント証明書ファイル「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）してください。（cert の後には半角スペースを一文字入力してください）

ファイルドロップ後は Tera Term アプリで「Enter」を押してください。

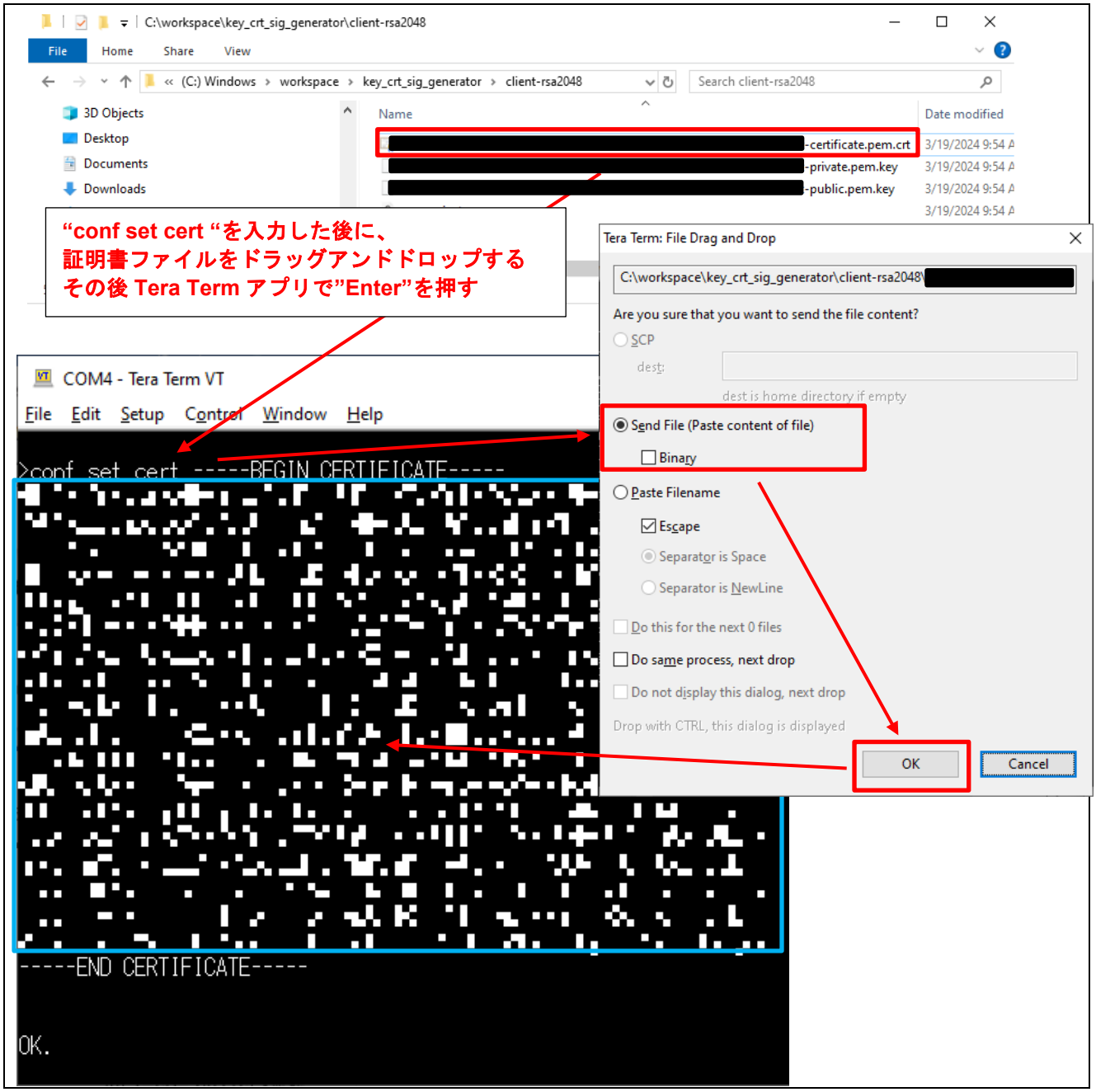


図 6-33 クライアント証明書の入力

(3) ルート CA 証明書の登録

「5.1.2 ルート CA 証明書の入手」で AWS よりダウンロードしたルート CA 証明書を CLI モードで登録します。

Tera Term にて「conf set rootca 」と入力したのち、ルート CA 証明書ファイル「AmazonRootCA1.pem」を Tera Term にドラッグアンドドロップ（ファイル送信）してください。（rootca の後には半角スペースを一文字入力してください）

ファイルドロップ後は Tera Term アプリで「Enter」を押してください。

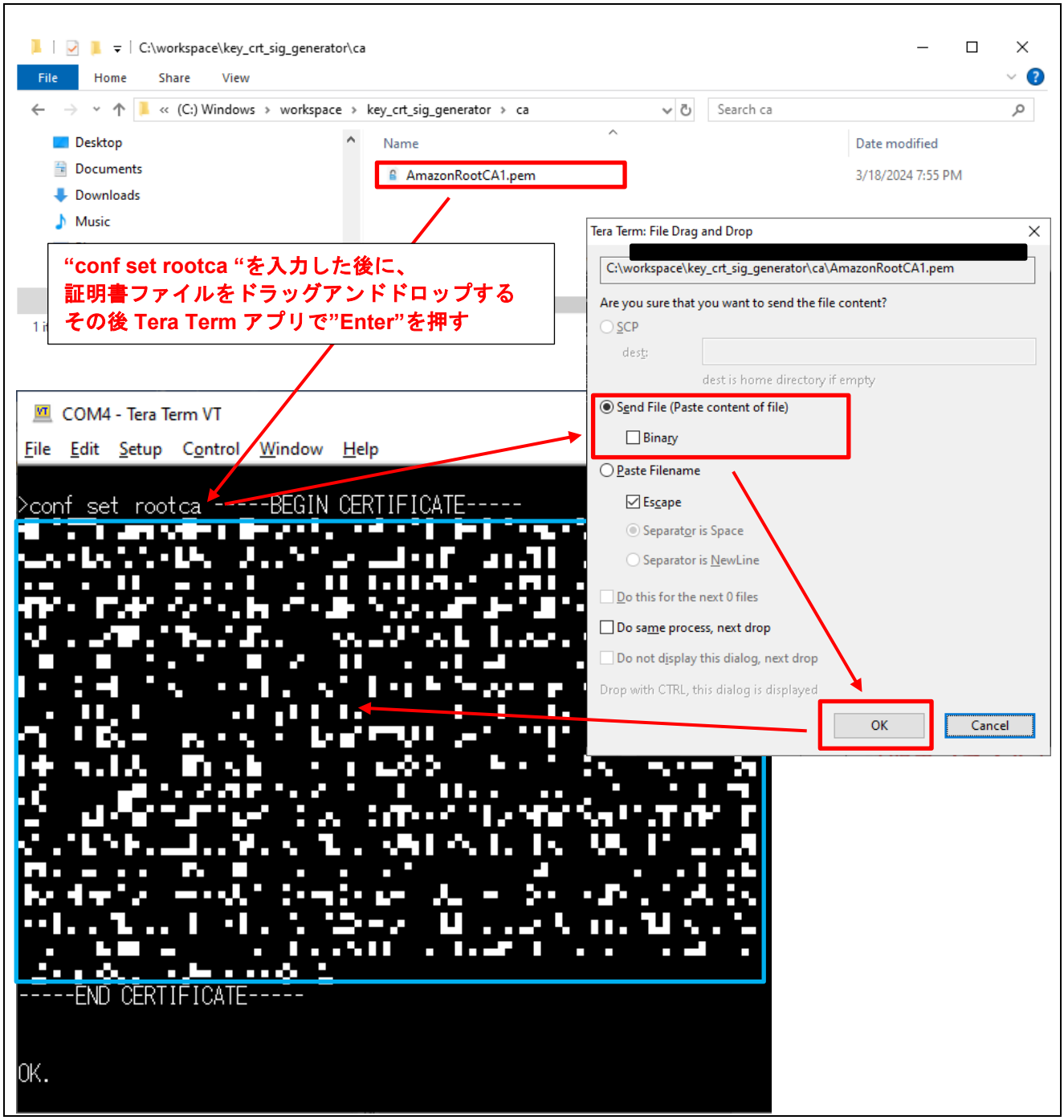


図 6-34 ルート CA 証明書の入力

(4) OTA 用鍵ペア証明書（ECDSA 証明書）の登録

「5.2 OTA 用鍵ペアと証明書の生成 5.1.2 で生成した鍵ペア証明書を CLI モードで登録します。

Tera Term にて「conf set codesigncert 」と入力したのち、鍵ペア証明書ファイル「secp256r1.crt」を Tera Term にドラッグアンドドロップ（ファイル送信）してください。（codesigncert の後には半角スペースを一文字入力してください）

ファイルドロップ後に Tera Term アプリで「Enter」を押してください。

【注】 証明書ファイルの改行コードはテキストエディタなどで LF に変更してから張り付けてください

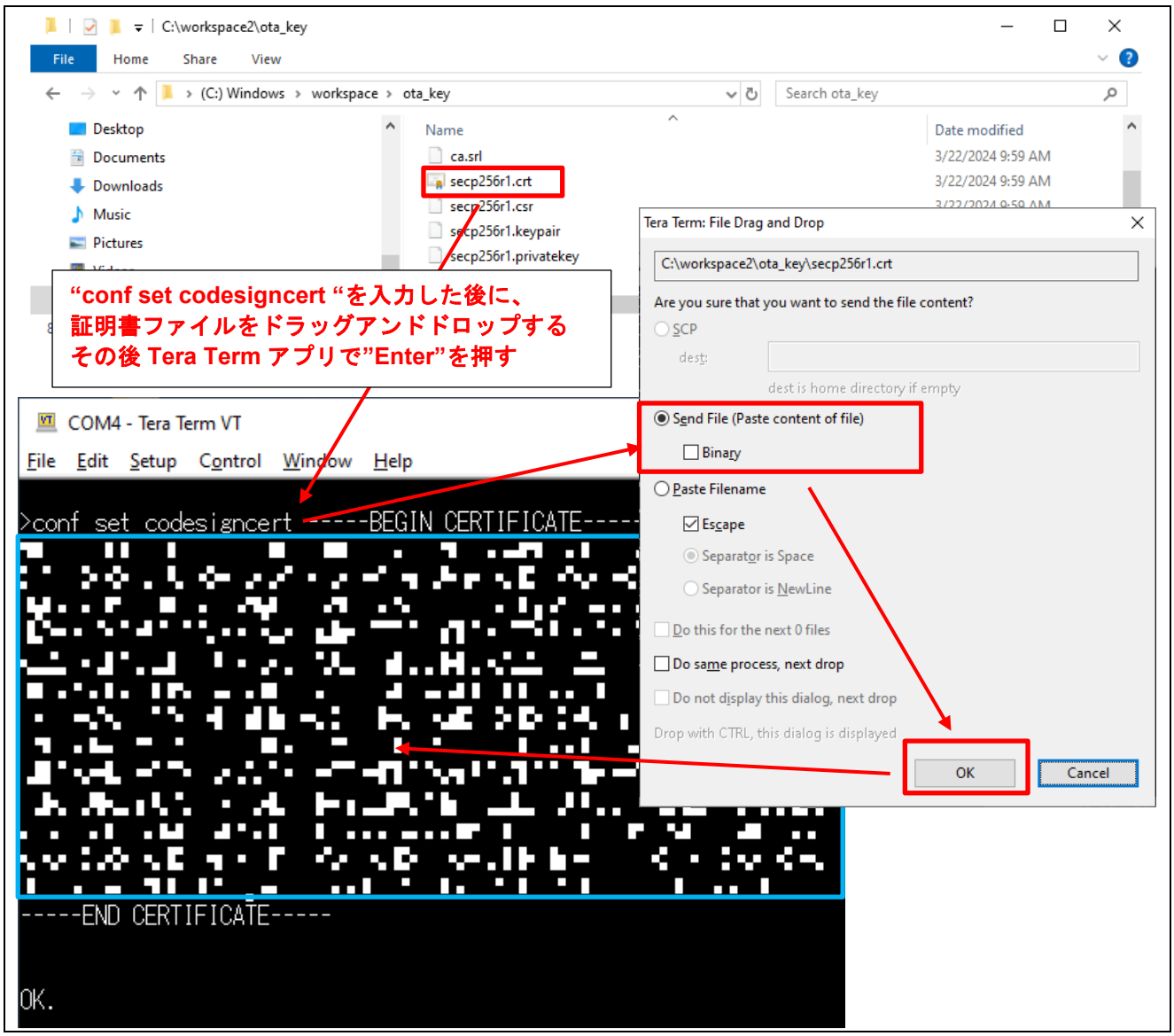


図 6-35 OTA 用鍵ペア証明書の入力

(5) AWS IoT の設定をコミット（データフラッシュに書き込み）

CLI で入力した各設定値をデータフラッシュに書き込みます。コミット操作をするまでは入力した設定値は保存されません。ボードの電源を切る前に必ずコミット操作を行ってください。

コミット操作後、電源を切っても各設定値は保持されます。

Tera Term で以下のコマンドを実行します。

```
conf commit[enter]
```

コミット操作をすると以下のように表示がされます。

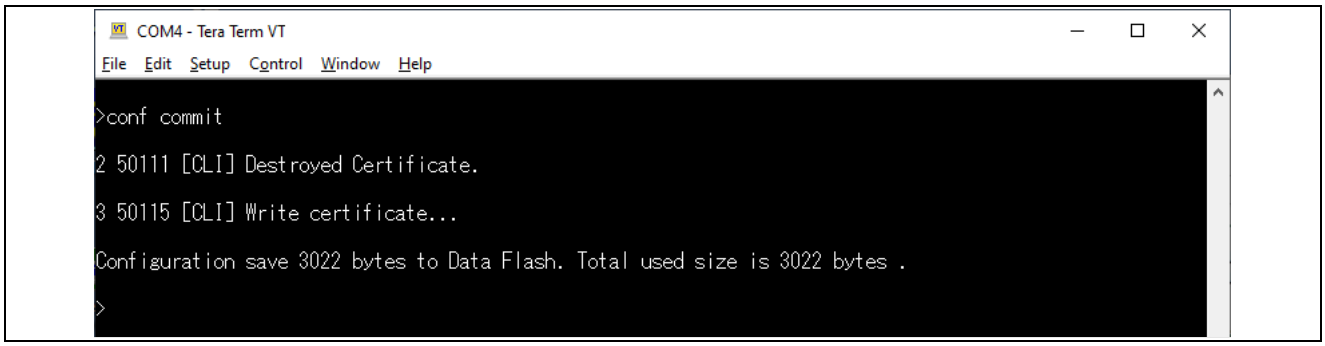


図 6-36 commit によるデータフラッシュへの書き込み

(6) データフラッシュの書き込み設定値の削除

コミットしてデータフラッシュへ書き込まれたデータを削除することができます。
各設定値を更新したい場合、新しい設定値を入力してコミット操作を実行することによって更新（上書き）することができます。よって、削除操作は通常不要ですが、書き込まれたデータをクリアしたい場合に本操作を行ってください。

また、TSIP で使用するルート CA 証明書署名検証用公開鍵とクライアント証明書用公開鍵・秘密鍵の 3 種の鍵はプログラム初回実行時にデータフラッシュへ書き込まれます。
上記の TSIP 用鍵は CLI でアクセスすることはできないため、使用する鍵を変更したいときなどは本操作にてデータフラッシュをクリアしてください。

【注】「5.1.4」のスクリプト再実行や「5.1.5」の 3 種の鍵の再生成を実施した場合は、必ず本操作にてデータフラッシュをクリアして下さい。
また、クリア後は(1)～(4)の操作を再度行ってください。
これにより、プログラム実行時に、再作成した新しい 3 種の鍵がデータフラッシュに書き込まれます。

Tera Term で以下のコマンドを実行します。

```
format[enter]
```

削除操作をすると以下のように表示されます。データフラッシュの各設定値はすべてクリアされるため、再設定を行ってください。

図 6-37 データフラッシュの設定値の削除

(7) リセットと初期ファームウェアの実行

各設定値の入力が終わったら、プログラムをリセットして AWS への接続処理に移行します。
Tera Term で以下のコマンドを実行してください。ソフトウェアリセットが行われプログラムが最初から実行されます。

```
reset[enter]
```

リセット実行後、Tera Term に以下のように通信ログが表示されます。
TSIP による TLS 通信がスタート後、PubSub デモと OTA デモが動作します。PubSub デモは MQTT 通信を確認するデモ、OTA デモはファームウェアアップデートを行うデモです。
以下画面で PubSub デモが実行されて、OTA ジョブ待ちになっていることを確認します。

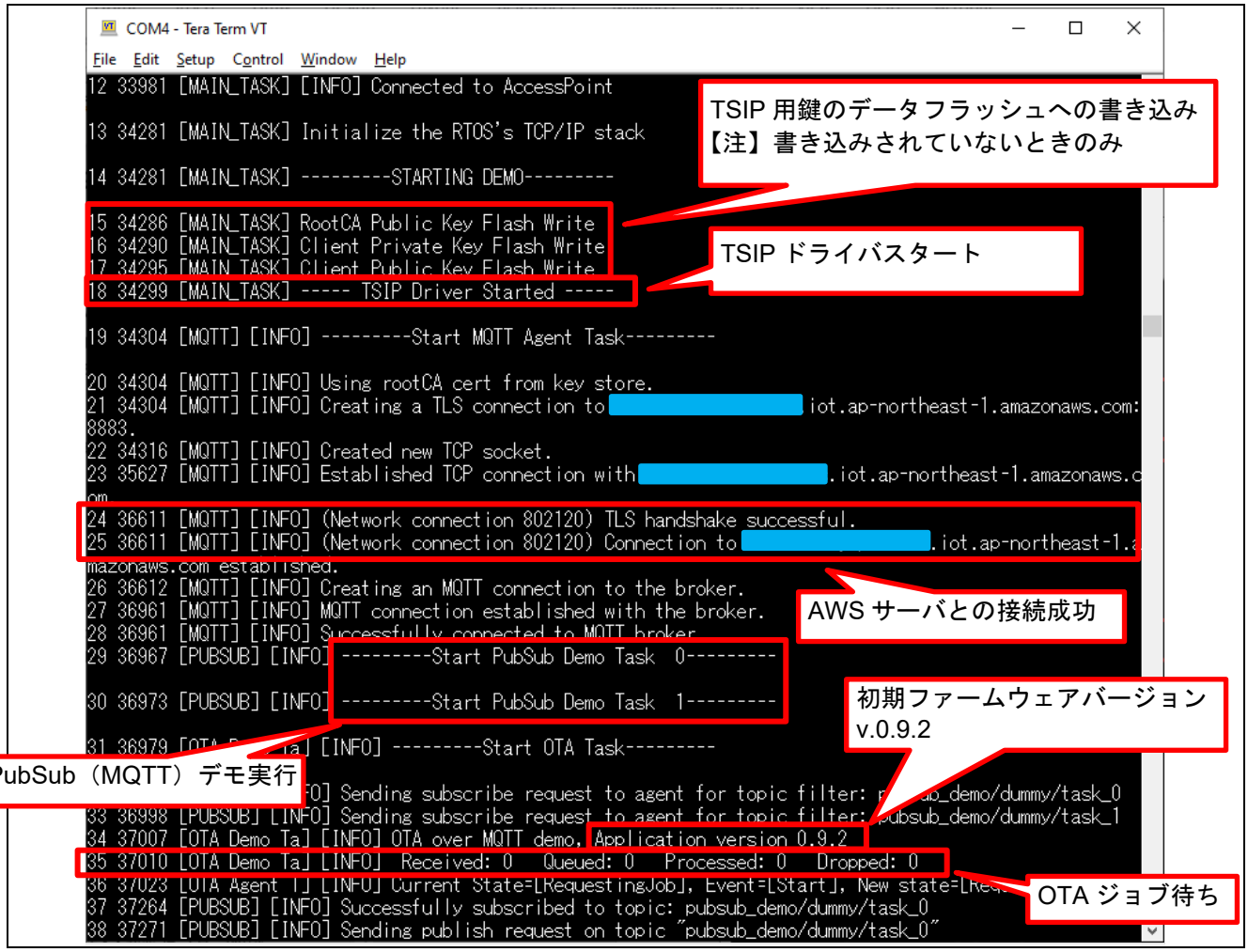


図 6-38 初期ファームウェアの実行

6.1.6 MQTT 通信状況の確認

MQTT 通信の状況を確認します。

MQTT 通信の実行状態は AWS にて確認を行うことができます。プログラム実行前に、以下の手順で AWS のモニター設定を行ってください。

(1) AWS マネジメントコンソールへのサインイン
AWS マネジメントコンソール (https://aws.amazon.com/console/)にサインインし、AWS のメニューより IoT Core の画面を表示してください。IoT Core の左端のメニューから「Test（テスト）」より、[MQTT test client（MQTT テストクライアント）]を選択し、MQTT テストクライアントを開きます。

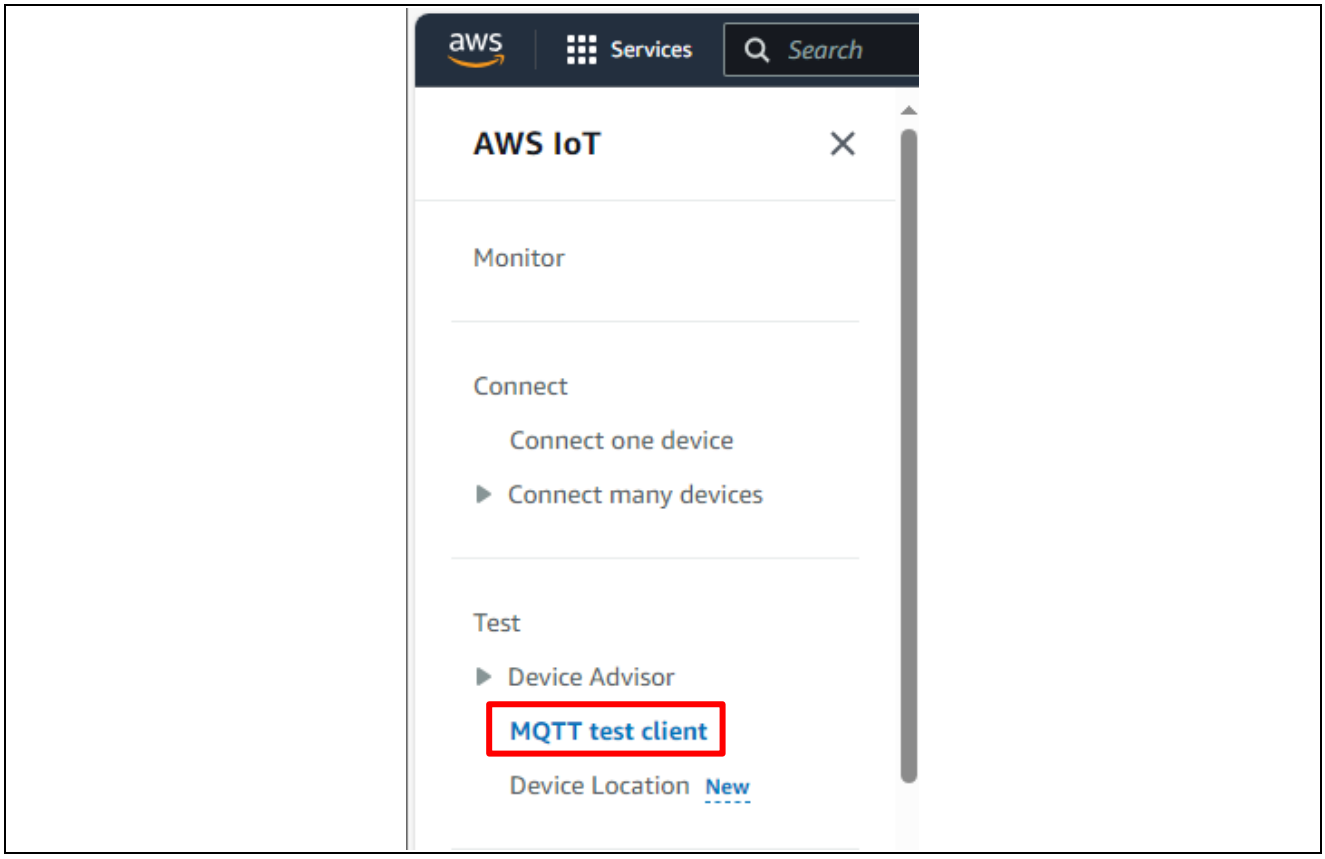


図 6-39 AWS IoT メニュー

(2) トピックをサブスクライブ
[Subscribe to a topic（トピックをサブスクライブする）]のタブをクリックし、[Topic filter（トピックのフィルター）]にワイルドカードの「#」を入力し、[Subscribe（サブスクライブ）]をクリックします。

図 6-40 MQTT テストクライアントの設定

(3) 空のコンソールの確認
以下のように画面下部に空のコンソールが表示されることを確認してください。

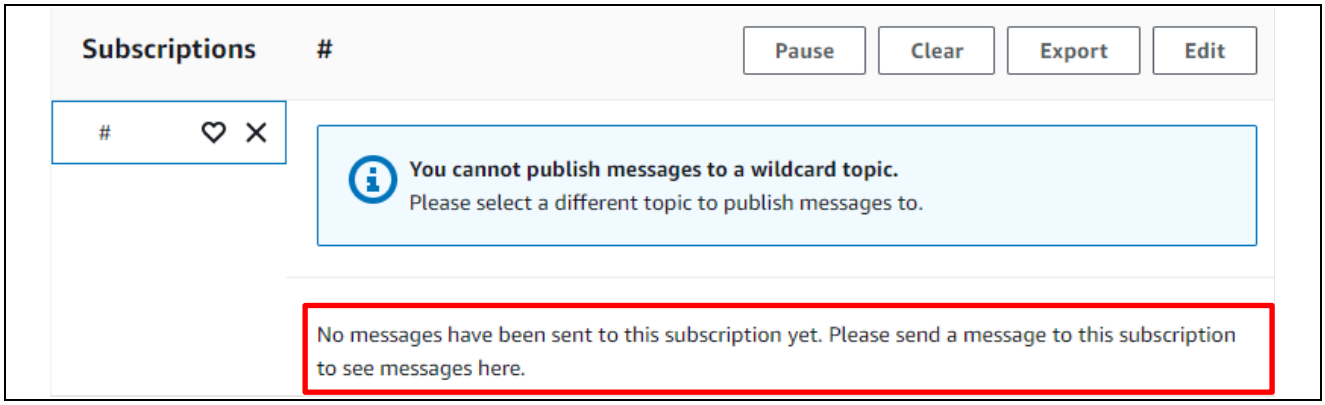


図 6-41 コンソール表示の確認

(4) プログラムを実行

「6.1.4(4)」項を参照してターゲットボードのリセット SW を押下し、プログラムをリセット・実行してください。プログラムが実行されると、以下のように PubSub デモ（MQTT 通信タスク）の実行状態が表示されます。
PubSub デモはタスク 0・タスク 1 の 2 つの MQTT 通信タスクを実行します。
それぞれの PubSub タスクは message 0 から message 9 の 10 回メッセージの送信を行います。

```
30 37415 [OTA Demo Ta] [INFO] OTA over MQTT demo, Application version 0.9.2
31 37418 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
32 37431 [OTA Agent T] [INFO] Current State=[RequestingJob], Event=[Start], New state=[RequestingJob]
33 37700 [PUBSUB] [INFO] Successfully subscribed to topic: pubsub_demo/dummy/task_0
34 37707 [PUBSUB] [INFO] Sending publish request on topic "pubsub_demo/dummy/task_0"
35 38410 [PUBSUB] [INFO] Successfully subscribed to topic: pubsub_demo/dummy/task_1
36 38417 [PUBSUB] [INFO] Sending publish request on topic "pubsub_demo/dummy/task_1"
37 39118 [OTA Agent T] [INFO] Subscribed to topic $aws/things/ckrx65n_wishii_test/jobs/notify-next.


38 39125 [OTA Agent T] [INFO] Subscribed to MQTT topic: $aws/things/ckrx65n_wishii_test/jobs/notify-next
39 39431 [OTA Demo Ta] [INFO]  Received: 0   Queued: 0   Processed: 0   Dropped: 0
40 39568 [MQTT] [INFO] Publishing message to pubsub_demo/dummy/task_0.

41 39604 [PUBSUB] [INFO] Successfully sent QoS 0 publish to topic: pubsub_d
, FailCount:0).
42 39822 [MQTT] [INFO] De-serialized incoming PUBLISH packet: Deserialize
43 39822 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
44 39822 [MQTT] [INFO] Received incoming publish message Task 0 publishing message 0
45 40274 [MQTT] [INFO] Publishing message to pubsub_demo/dummy/task_1.

46 40542 [MQTT] [INFO] Ack packet deserialized with result: MQTTSuccess.
47 40542 [MQTT] [INFO] State record updated. New state=MQTTPublishDone.
48 40549 [PUBSUB] [INFO] Successfully sent QoS 1 publish to topic: pubsub_dem
, FailCount:0).
49 40610 [MQTT] [INFO] De-serialized incoming PUBLISH packet: Deserializer
50 40610 [MQTT] [INFO] State record updated. New state=MQTTPubAckSend.
51 40612 [MQTT] [INFO] Received incoming publish message Task 1 publishing message 0
52 41074 [MQTT] [INFO] Publishing message to $aws/things/ckrx65n_wishii_test/jobs/$next/get.
```

Task0 messege 0 送信完了

Task1 messege 0 送信完了

図 6-42 PubSub デモ実行状態

(5) MQTT テストクライアント画面の確認
PubSub デモ実行中に AWS の MQTT テストクライアント画面を表示すると、通信されたデータを確認することができます。
PubSub デモが AWS との接続に成功すると、MQTT クライアントに通信ログが出力されます。
以下の画面は「Task0」の「message 5」を受信した場合の例となります。

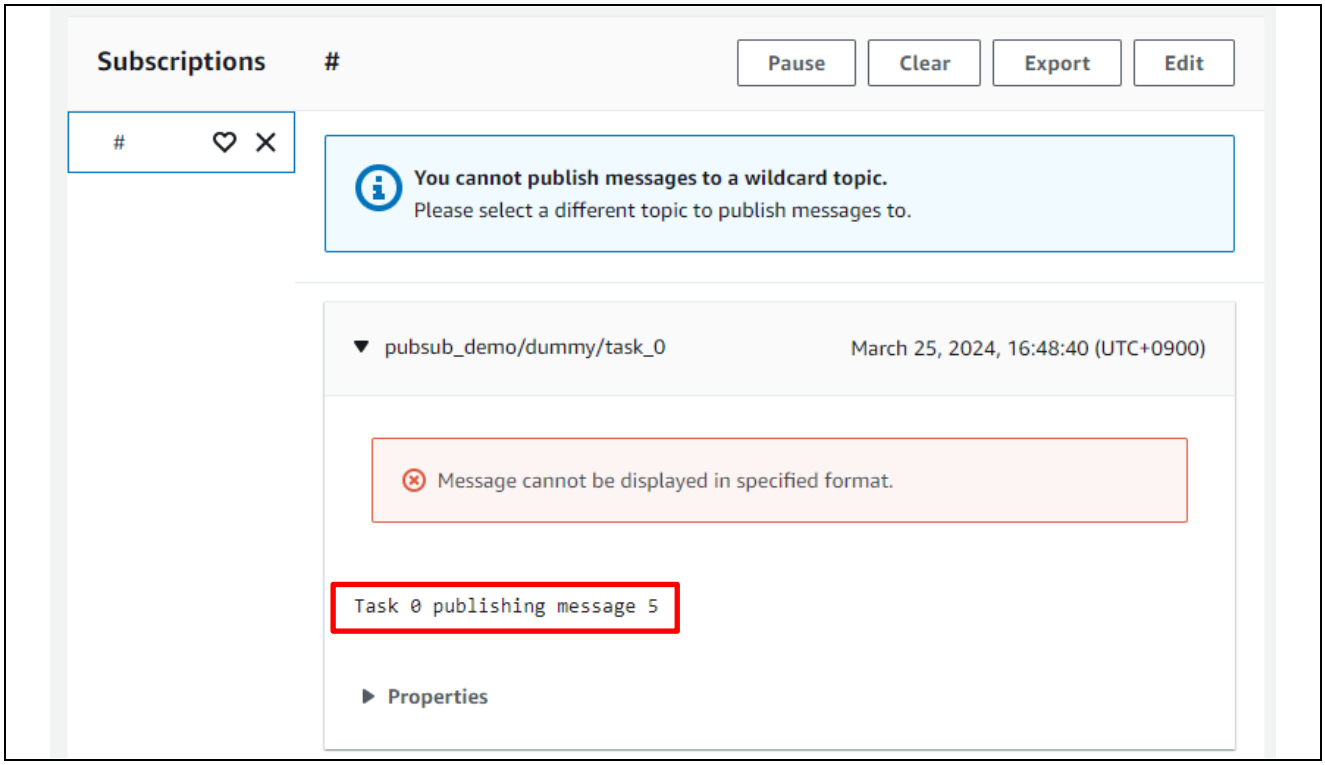


図 6-43 MQTT 通信ログの確認

6.2 更新用ファームウェア構築と実行

OTA にてアップデートされる更新用ファームウェアを作成します。本アプリケーションノートでは「6.1」節で作成した、初期ファームウェアのプロジェクト aws_ryz014a_tsip_ck_rx65n に対してバージョン番号のみ変更したものを作成します。

6.2.1 更新ファームウェアの作成

(1) ファームウェアのバージョンを v0.9.3 に変更

プロジェクト aws_ryz014a_tsip_ck_rx65n の以下箇所を修正します。
aws_ryz014a_tsip_ck_rx65n\src\frtos_config\demo_config.h の APP_VERSION_BUILD 定義を 3 にして、バージョン番号を 0.9.3 に変更してください。変更が完了したらビルドを再実行します。

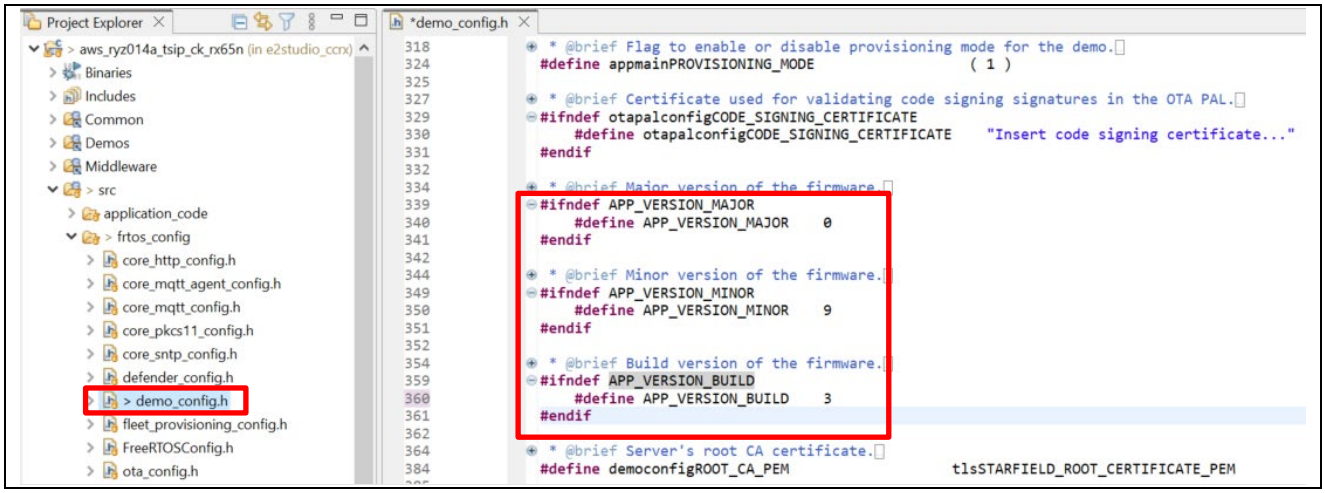


図 6-44 更新ファームウェアのバージョン変更

(2) Renesas Image Generator を使用して更新ファームウェアを生成

6.2.1(1)で再ビルドしたファームウェア(aws_ryz014a_tsip_ck_rx65n.mot)を Renesas Image Generator フォルダに上書きし、コマンドプロンプトで以下コマンドを実行します。

```
python image-gen.py -iup aws_ryz014a_tsip_ck_rx65n.mot -ip
RX65N_DualBank_ImageGenerator_PRM.csv -o user_093 -key secp256r1.privatekey -
vt ecdsa -ff RTOS
```

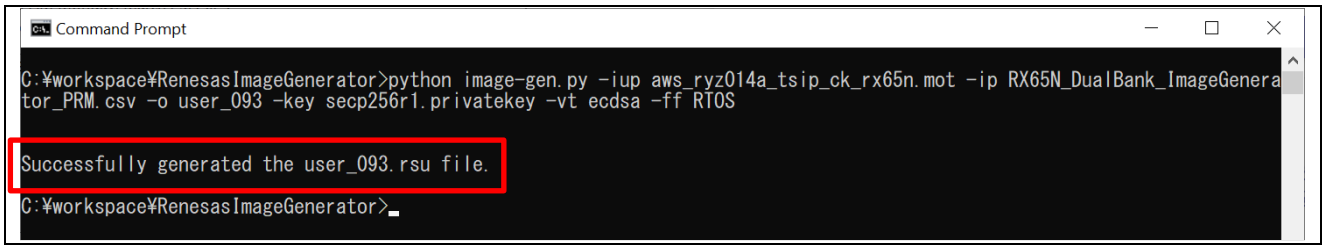


図 6-45 更新ファームウェアの作成

コマンドラインに「Successfully generated the user_093.rsu file.」と表示されたら作成完了です。

更新ファームウェアは以下のファイル名で作成されます。

- user_093.rsu

6.2.2　ファームウェアの更新

AWS にて、ファームウェアの更新を行うための OTA 更新ジョブを作成します。

ジョブ作成前にターゲットボードにて「6.1.5(7)」の手順で初期ファームを実行し、OTA ジョブ待ちの状態としておいてください。

(1) AWS IoT Core へファームウェア更新のジョブ作成
AWS への更新ファームウェア登録の手順の詳細は、別途アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「5.2 ファームウェアの更新」を参照して実施して下さい。

OTA ジョブの作成時には「5.2 OTA 用鍵ペアと証明書の生成」で作成した ECDSA の証明書・鍵を使用します。初期ファームウェア作成時に使用したものと同じ生成データを使用してください。
OTA ジョブを作成すると、OTA が実行されファームウェアのアップデートが実行されます。

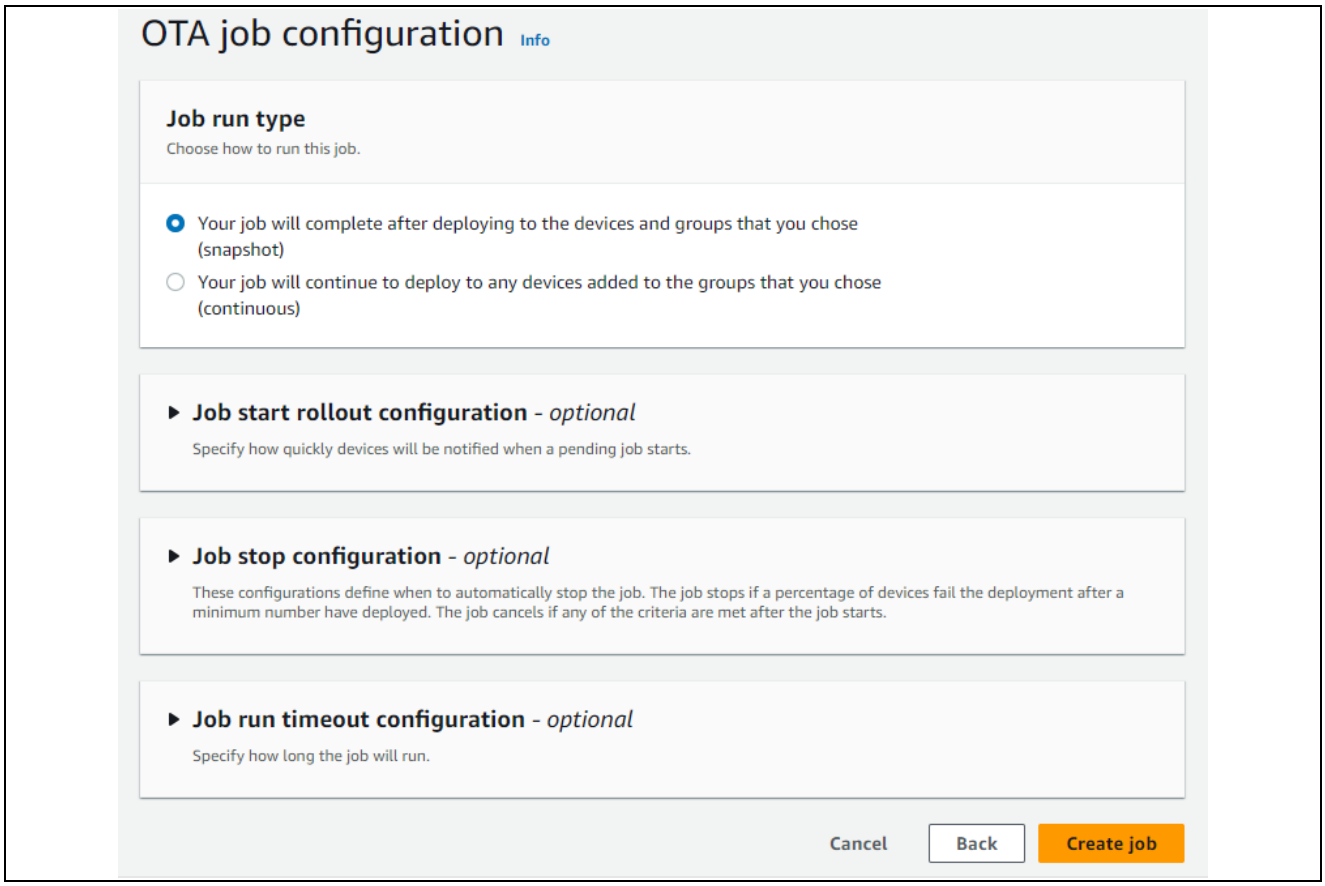


図 6-46 OTA ジョブの実行

(2) ファームウェアの受信

OTA ジョブが開始されると、ターゲットボードでファームウェアの受信が開始されます。
ファームウェアの受信はブロック単位で行われます。受信ごとに Received の回数が増えます。残りのブロックが 0 になると受信完了です。

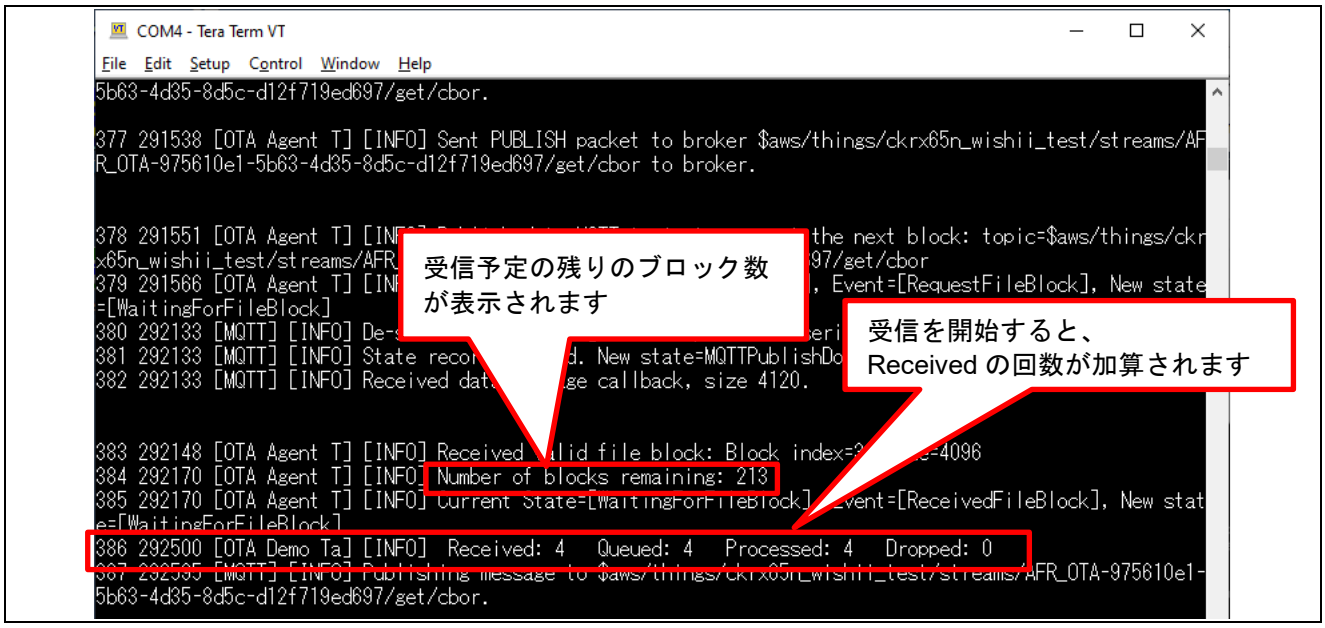


図 6-47 ファームウェアの受信

(3) ファームウェアの受信完了

ファームウェアの受信が完了し、受信したファームウェアの検証に成功したらファームウェアの書き込み後バンクスワップが行われ、更新ファームウェアが実行されます。
正常に更新ファームウェアが実行されると最初のメニューが表示されます。

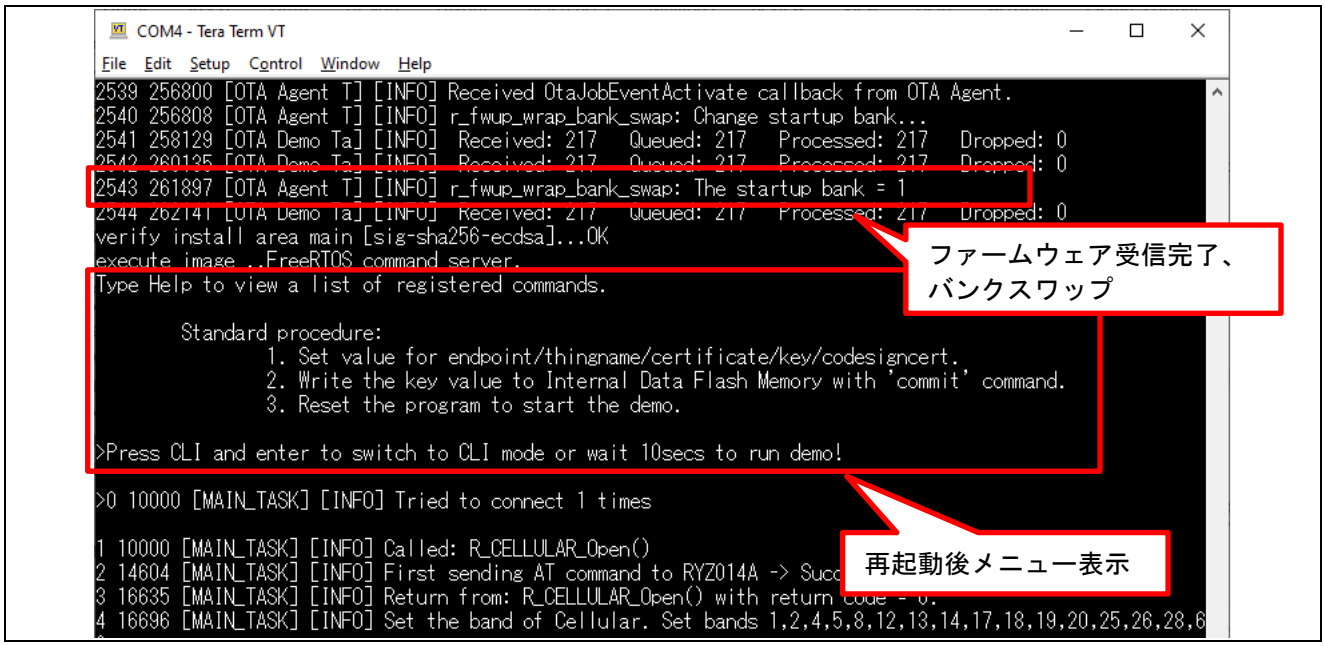


図 6-48 ファームウェア受信完了

(4) ファームウェアのバージョンの確認
ファームウェアのバージョンが更新後の Ver0.9.3 になっていることを確認します。
初期ファームウェアと同様に PubSub デモと OTA デモが動作すればファームウェアアップデートは完了です。

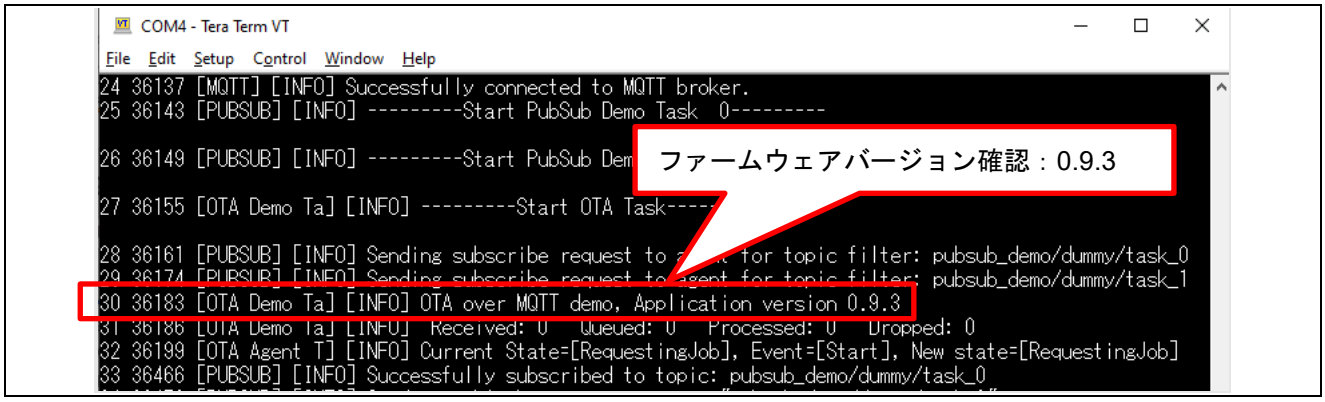


図 6-49 ファームウェアのバージョン確認

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（Ethernet 版）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_tsip_ck_rx65n.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

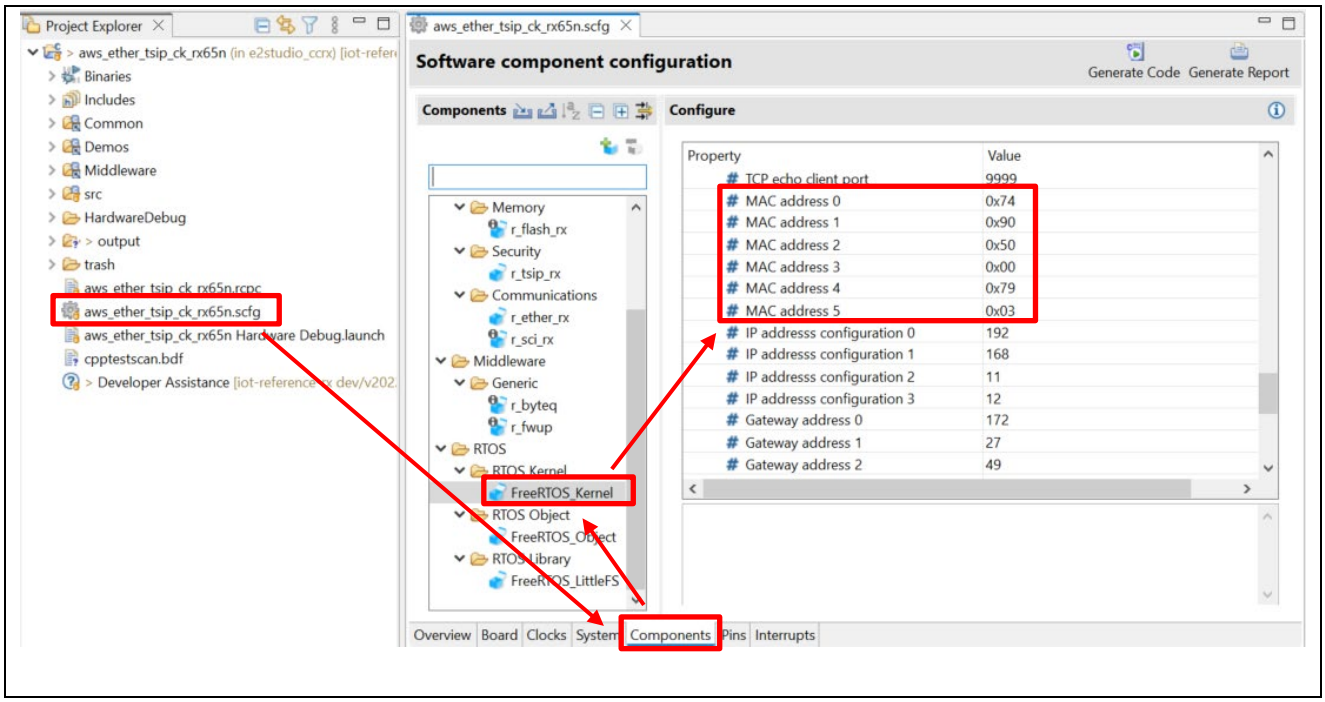


図 7-1 MAC アドレスの設定

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。

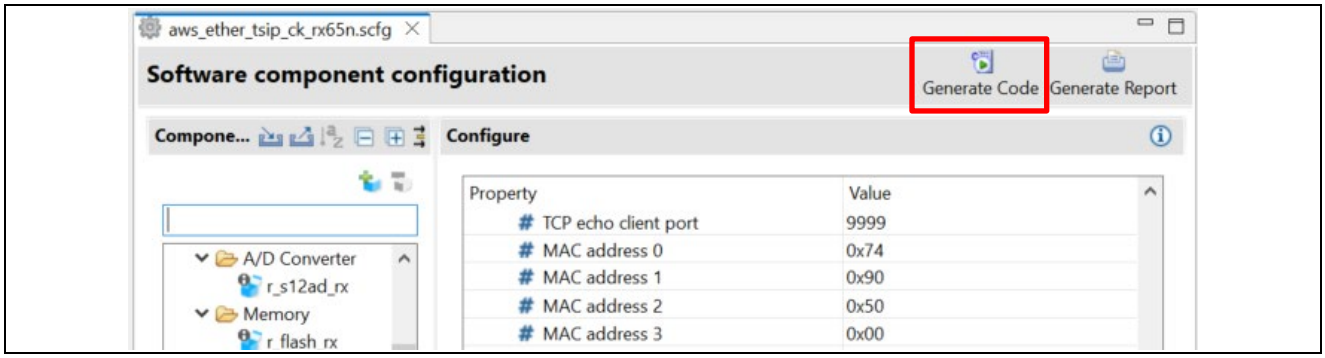


図 7-2 コードの生成

8. トラブルシューティング

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

表 8-1 トラブルシューティング（1）

No	トラブル内容	原因	解決策	参照
1	初期ファームウェアの作成コマンドが失敗する	Python のパスが通っていない。	Python を再インストールしてください。 また、「Add python.exe to PATH」にチェックが入っていることを確認してください。(注 2)	(注 1)
2		暗号化ライブラリがインストールされていない。	暗号化ライブラリをインストールしてください。	(注 2)
3	初期ファームウェアが書き込めない	CK-RX65N がデバッグ設定になっていない。	CK-RX65N の J16 の設定が 1-2 ショート（デバッグ）になっていることを確認してください。	(注 3)
4	初期ファームウェアが起動しない	CK-RX65N が RUN 設定になっていない。	CK-RX65N の J16 の設定が 2-3 ショート（RUN）になっていることを確認してください。	6.1.4(4)
5	セルラー通信が開始できない	RYZ014A PMOD が正しく接続されていない。	RYZ014A PMOD の接続を見直してください。	(注 3)
6		SIM カードが挿入されていない。	SIM カードを挿入してください。	(注 3)
7		SIM カードの設定が正しくされていない。	r_cellular のコンフィグ設定を見直してください。	6.1.2(4)
8		CK-RX65N 付属の SIM カードを使用しており、かつ SIM カードのアクティベーションができていない。	SIM カードのアクティベーションを行ってください。	6.1.2(4)
9	セルラー通信中にエラーが発生する	通信環境が悪い。	RYZ014A PMOD にアンテナおよび電源の接続を行ってください。 また、アンテナを窓際など通信環境の良い場所においてください。	(注 3)

【注】 1. アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.2」参照

【注】 2. 注アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.2(5)」参照

【注】 3. アプリケーションノート「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(v202210.01-LTS-rx-1.1.3 以降対応版)」（R01AN7037）の「2.5」参照

表 8-2 トラブルシューティング（2）

No	トラブル内容	原因	解決策	参照
10	AWS への接続でエラーが発生する	AWS IoT 情報が設定されていない、または間違えている。	再度、AWS IoT 情報の設定を行ってください。	6.1.4
11		証明書・鍵データが正常に作成されていない、または登録されていない。	再度手順に基づき証明書・鍵データの作成を行ってください。	5.1
12	ブートローダ起動後にファームウェアが起動しない	ブートローダに公開鍵が正しく設定がされていない。	ブートローダの公開鍵設定を見直してください。	6.1.2(1)
13	OTA アップデート後にファームウェアが起動しない	ファームウェアに公開鍵が正しく設定されていない。	ファームウェアの公開鍵設定を見直してください。	6.2.1(2)
14		デバイス選択が正しく設定されていない。	ファームウェアおよびブートローダのデバイス設定を見直してください。	6.1.5

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2024.06.14	-	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、ノイズ輻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。

© 2024 Renesas Electronics Corporation. All rights reserved.