RX ファミリ

RX26T OTA デュアルバンク機能を用いたファームウェアアップデート デモ

要旨

本アプリケーションノートはルネサス製マイクロコントローラを使用し、OTA（Over The Air）を利用したファームウェアアップデートを行うソフトウェアサンプルプログラムについて説明することを目的としています。本アプリケーションノートでは、「永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）」に同梱の「2 シャント抵抗電流検出方式」と「1 シャント抵抗電流検出方式」を OTA によるファームウェアアップデートで切り替えます。

本アプリケーションノート対象ソフトウェアはあくまで参考用途であり、弊社がこの動作を保証するものではありません。本アプリケーションノート対象ソフトウェアを使用する場合、適切な環境で十分な評価をしたうえで使用してください。

動作確認デバイス

本アプリケーションノートの対象ソフトウェアの動作確認は下記のデバイスで行っております。

- 使用 MCU：
  RX26T

目次

1. 概要 5
1.1 本アプリケーションノートの OTA について 5
1.2 AWS 環境 5

2. 動作確認環境 6
2.1 ハードウェア仕様 8
2.1.1 ハードウェア構成図 8
2.1.2 ボードユーザインタフェース 9
2.2 ボードの接続方法 9

3. クイックスタートガイド 10
3.1 本サンプルプログラムの OTA 概要 10
3.2 初期イメージと更新イメージ 10
3.2.1 プロジェクトのビルド 11
3.2.2 初期イメージ作成方法 11
3.2.3 更新イメージの作成方法 12
3.2.4 Amazon S3 のファイル構成 12
3.3 モータ制御 UI の設定 12
3.4 AWS の準備 12

4. AWS 環境構築 13
4.1 デバイスを AWS IoT に登録する 13
4.2 Lambda の準備 13
4.2.1 関数を作成する 13
4.2.2 トリガの追加 14
4.2.3 ロールの設定 15
4.2.4 Lambda のコードを登録する 17
4.2.5 OTA 用 Lambda コード 18
4.2.6 プロジェクトのソースコード修正 1 20
4.2.7 プロジェクトのソースコード修正 2 21

5. サンプルプログラム 23
5.1 システム構成 23
5.2 プロジェクト構成 23
5.3 セクション設定 24
5.4 ソフトウェア構成 24
5.4.1 MCU（RX26T）のソフトウェア構成 24
5.4.2 MCU（RX26T）のソフトウェア構成詳細（ファイル構成詳細） 25
5.4.3 AWS のソフトウェア構成 26
5.5 サンプルプログラムの構成 27
5.5.1 概要 27
5.5.2 使用端子一覧 27
5.5.3 使用する周辺機能 27
5.5.4 周辺機能の設定 27
5.5.5 ファイル構成 30

5.5.6 変数一覧 31
5.5.6.1 定数一覧 31
5.5.7 グローバル関数一覧 32
5.5.8 関数仕様 32
5.5.9 サンプルプログラムのフローチャート 38
5.5.10 ブートローダ 42
5.6 ハードウェアの準備 42
5.6.1 Pmod-USBUART の接続方法 42
5.7 ログ確認用 Terminal ソフトの準備 43
5.7.1 Tera Term を起動する 43
5.7.2 Tera Term の Terminal を設定する 43
5.7.3 Tera Term の Serial を設定する 44
5.8 サンプルプログラムの動作概要 45
5.9 サンプルプログラムの動作詳細 46

6. プロジェクトのビルド手順 50
6.1 e2 studio での手順 50
6.1.1 e2 studio でのインポート方法 50
6.1.2 FIT の確認 51
6.1.3 ビルドオプションの設定 51
6.1.4 プロジェクトのビルド 51
6.2 CS+ での手順 52
6.2.1 プロジェクトのビルド 52

7. トラブルシューティング 53
7.1 AWS に接続できない 53
7.1.1 DA16600 Pmod™ Board のログを確認する方法 53
7.1.1.1 必要部品 53
7.1.1.2 接続方法 53
7.1.1.3 Tera Term を起動する 54
7.1.1.4 Tera Term の Terminal を設定する 54
7.1.1.5 Tera Term の Serial を設定する 55
7.1.1.6 動作を確認する 55
7.1.2 DA16600 Pmod™ Board のファームウェアバージョンを更新する 56
7.1.2.1 ファームウェアをダウンロードする 56
7.1.2.2 ダウンロードしたファイルを解凍する 57
7.1.2.3 Pmod-USBUART を接続する 57
7.1.2.4 電源を供給する 57
7.1.2.5 Tera Term を起動する 58
7.1.2.6 Tera Term の Terminal を設定する 58
7.1.2.7 Tera Term の Serial を設定する 58
7.1.2.8 ファームウェアを更新する 59
7.1.2.9 バージョンを確認する 60
7.1.3 Fail to establish tls-sess(0x7200)が表示される場合 61
7.1.3.1 MQTT 用バッファを再設定する 61
7.1.3.2 設定結果の確認する 62

8. 参考資料 63

改訂記録 64

1. 概要

本アプリケーションノートはルネサス製マイクロコントローラを使用し、OTA（Over The Air）を利用したファームウェアアップデートを行うソフトウェアサンプルプログラムについて説明することを目的としています。本サンプルプログラムでは RX26T の OTA を確認することができます。

本サンプルプログラムではクラウドに Amazon Web Service（AWS）を使用します。Amazon S3 にソフトウェアアップデート用のプログラムを配置し、クラウド上の複雑な操作は AWS Lambda で行います。AWS との接続は IoT Core を利用します。更新プログラムは Wi-Fi 経由でインターネット接続された DA16600 Pmod™ Board を経由し、RX26T にダウンロードします。ダウンロードした更新プログラムは「RX ファミリ ファームウェアアップデートモジュール Firmware Integration Technology（R01AN6850）」を使用してセルフプログラミングを行います。本アプリケーションノートで使用する RX26T はデュアルバンク機能を搭載しているため、ファームウェアアップデート前のプログラムを実行しながら OTA を実行できます。本アプリケーションノートでは FreeRTOS を使用しないベアメタル構成で OTA を実現する例を示します。

1.1 本アプリケーションノートの OTA について

ルネサスでは「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）」に記載のとおり、セキュリティリスクの観点から OTA を実装する際にはブートローダを使用します。ブートローダには起動前にプログラム領域をベリファイし改竄されていないことを確認する機能や、ユーザプログラムが存在しない場合にファームウェアをインストールする機能などプログラムが安全に機能するための仕組みが搭載されています。今回使用しているファームウェアアップデート FIT にも、これらのセキュリティ機能が搭載されています。しかし、これらのセキュリティに関する機能は、本来の OTA に対してかなり複雑な制御を要求されます。たとえばセキュリティリスクがない LAN 内での OTA では、セキュリティチェックは不要なオーバヘッドとなります。

本アプリケーションノートではセキュリティなどの OTA とは直接関係しない機能を排除し、ファームウェアアップデート FIT の最低限の機能で OTA を実現しています。

たとえば通常のブートローダは前述のとおり起動前にプログラム面をベリファイしますが、本アプリケーションノートのブートローダはメインプログラムを起動する処理しか入っていません。

1.2 AWS 環境

AWS 環境はお客様ご自身のアカウントで構築する必要があります。アカウントの作成方法については「4 AWS 環境構築」を参照してください。

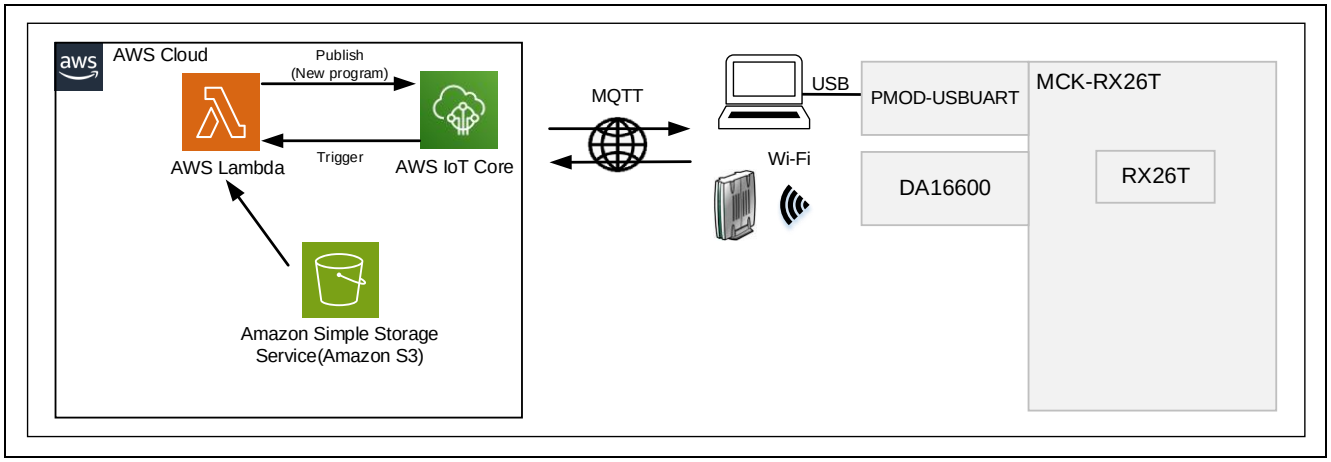

図 1-1 OTA デモ構成図

2. 動作確認環境

本アプリケーションノート対象ソフトウェアの開発環境を表 2-1、表 2-2 に示します。

表 2-1 ハードウェアの開発環境

分類	使用製品
マイコン／CPU ボード型名	RX26T（R5F526TFCDFP）/ RTK0EMXE70C00000B
インバータボード	Renesas Flexible Motor Control Kit 同梱 48V 10A BLDC 用インバータボード（RTK0EMXE70S00020BJ）
Wi-Fi モジュール	DA16600 Pmod™ Board （US159-DA16600EVZ）
モータ	R42BLD30L3（MOONS’製）
USBUART 変換モジュール	Pmod-USBUART（DIGILENT 製）

表 2-2 ソフトウェアの開発環境

IDE バージョン	RX スマート・コンフィグレータ	ツールチェーンバージョン
CS+：V8.10.00	バージョン 2.19.0	CC-RX：V3.05.00
e^{2}studio：2023-10	e^{2}studio プラグイン版	

表 2-3　動作確認条件（DA16600 Pmod™ Board）

項目	内容
使用ボード	US159-DA16600EVZ Pmod Board
ファームウェア	DA 16600 v3.2.8.0 • Wi-Fi への接続や AWS への TLS 通信などは本ファームウェアで実行 • バージョンが異なる場合は「7.1.2 DA16600 Pmod™ Board のファームウェアバージョンを更新する」を参照してください

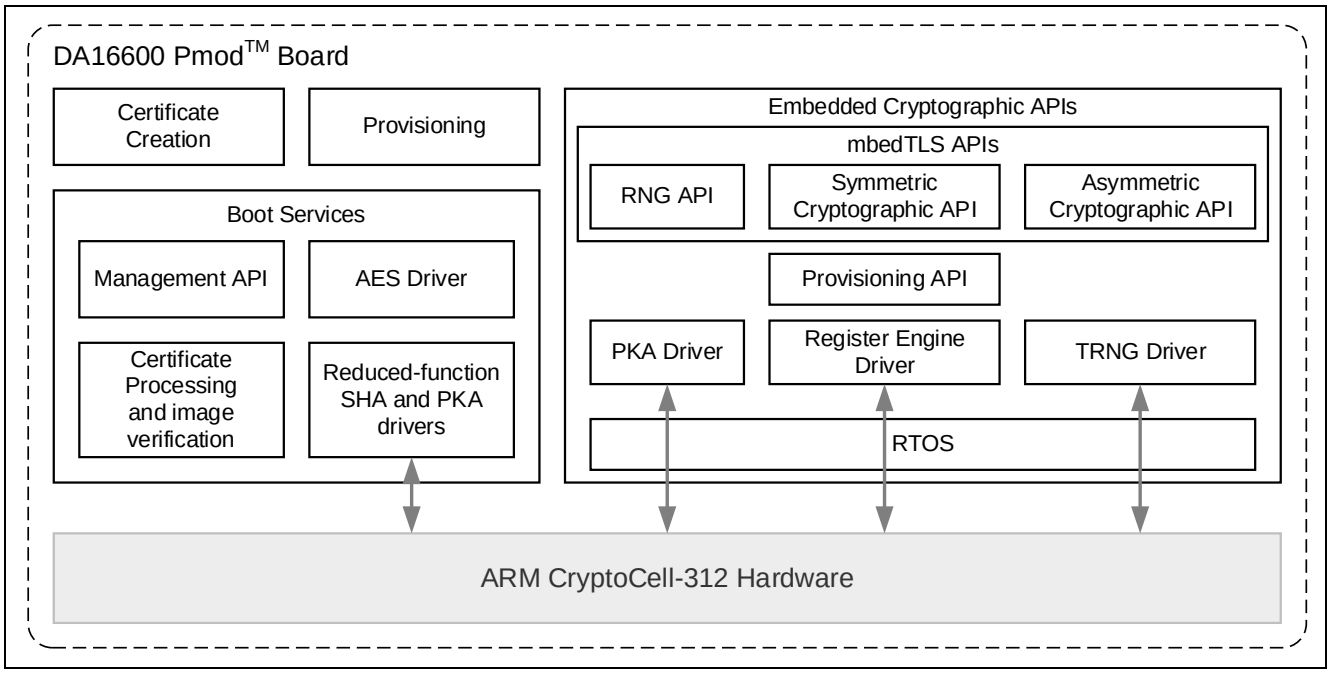

図 2-1　DA16600（DA16200）ソフトウェアスタック

購入、技術サポートにつきましては、弊社営業および代理店にお問い合わせください。

2.1 ハードウェア仕様

2.1.1 ハードウェア構成図

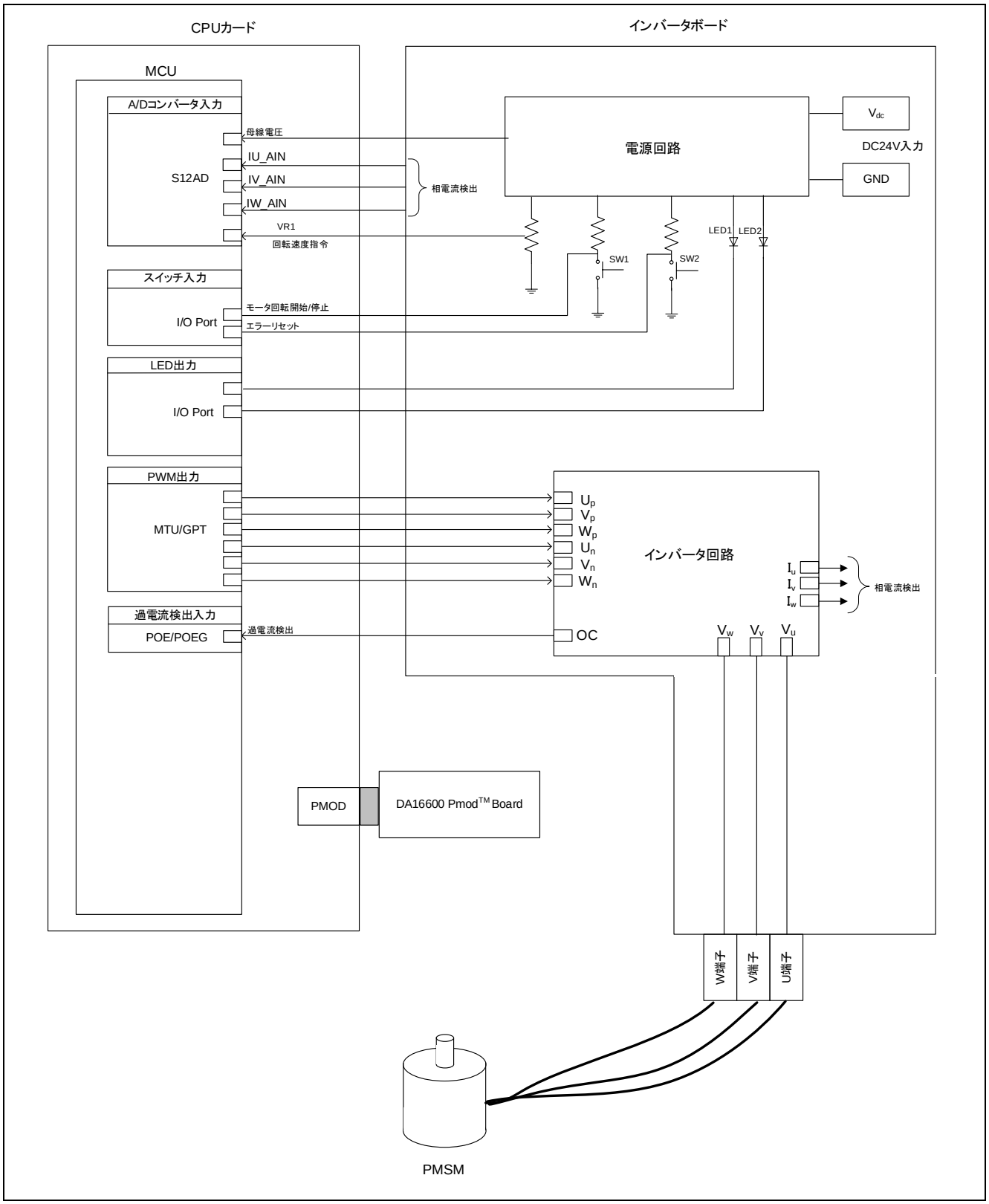

図 2-2 ハードウェア構成図

2.1.2 ボードユーザインタフェース

本システムのボードユーザ

表 2-4 ボードユーザインタフェース

項 目	インタフェース部品	機 能
回転速度	ボリューム （VR1：インバータボード）	回転速度指令値入力
START/STOP	トグルスイッチ （SW1：インバータボード）	モータ回転開始/停止指令
ERROR RESET	プッシュスイッチ（SW2：インバータボード）	エラー状態からの復帰指令
LED1	オレンジ色 LED （CPU ボード／インバータボード）	・モータ回転時：点灯 ・停止時：消灯
LED2	オレンジ色 LED （CPU ボード／インバータボード）	・エラー検出時：点灯 ・通常動作時：消灯
RESET	プッシュスイッチ（SW1：CPU ボード）	システムリセット

2.2 ボードの接続方法

接続イメージを示します。

CPU ボードにはプログラム書き込み時にオープンにする必要がある JP11 があります。インバータボードには 1Shunt 電流検出／2Shunt 電流検出を切り替えるための JP8/JP11 があります。

なお、Pmod-USBUART は PMOD のタイプが異なるため、PMOD に直接接続することができないので注意してください。Pmod-USBUART の接続方法は「5.6.1 Pmod-USBUART の接続方法」を参照してください。

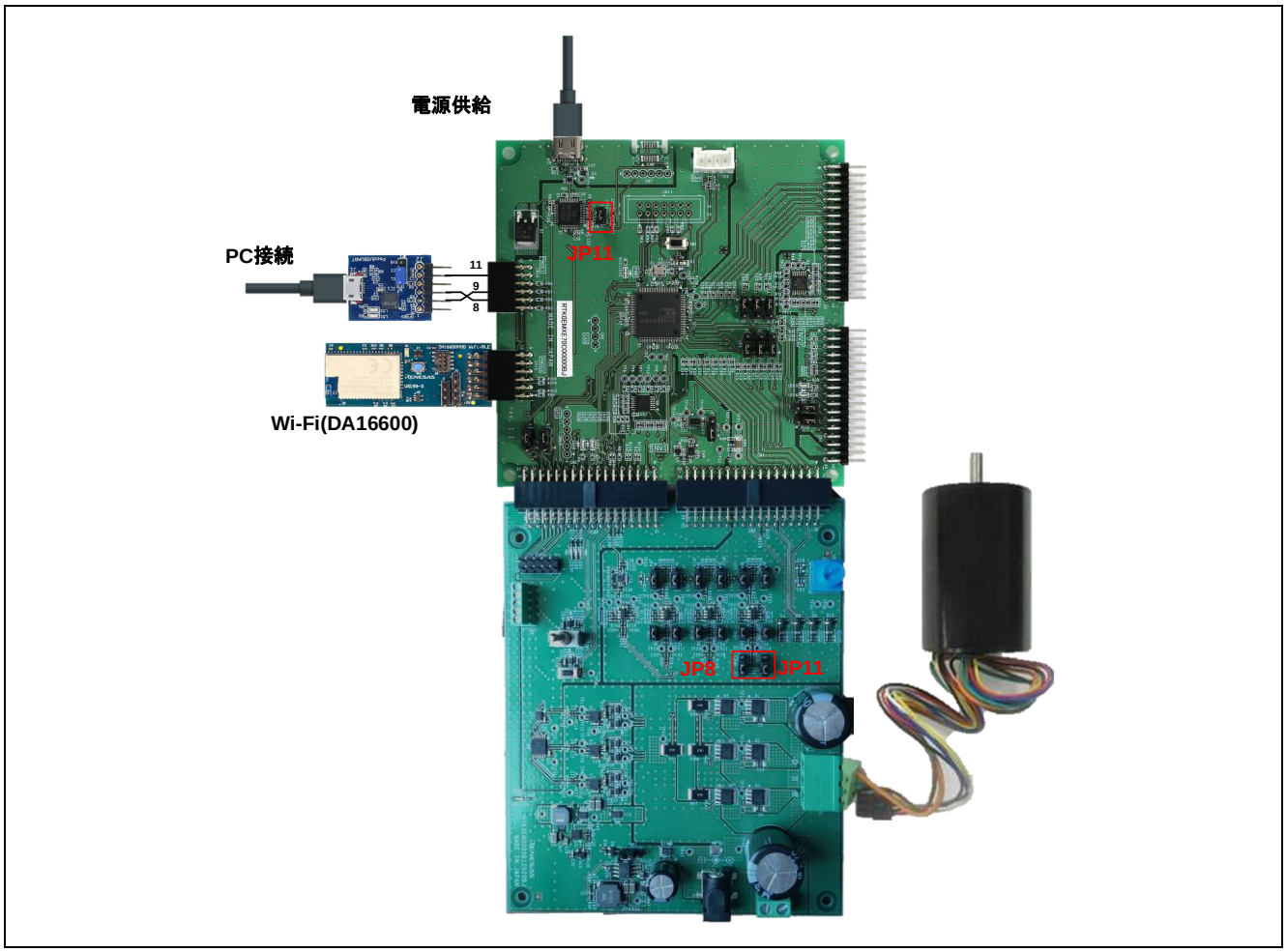

図 2-3 接続イメージ

3. クイックスタートガイド

本章は Renesas Flexible Motor Control Kit とサンプルプログラムを使用して OTA を体感いただくためのクイックスタートガイドです。モータを駆動する方法の詳細については「永久磁石同期モータのセンサレスベクトル制御 - MCK 用（R01AN6858）」を参照してください。また Renesas Flexible Motor Control Kit のボード設定や接続の詳細に関しては「MCK-RX26T ユーザーズマニュアル（R12UZ0111）」を参照ください。

本アプリケーションノートでのボードの接続方法は「5.6 ハードウェアの準備」を参照してください。

なお PMOD Type 6A モジュール接続用コネクタ（CN10）に Pmod-USBUART を接続しますが、直接接続できないことに注意してください（接続方法は「5.6.1 Pmod-USBUART の接続方法」を参照してください）。

3.1 本サンプルプログラムの OTA 概要

本サンプルプログラムは「RX ファミリ ファームウェアアップデートモジュール Firmware Integration Technology（R01AN6850）」の「1.3.1 デュアルバンク方式」をもとに最低限の実装で OTA を実現しています。「図 3-1 OTA のイメージ」に OTA のイメージを示します。

まず初期イメージとしてブートローダとサンプルプログラムをマージしたモトローラ S レコードファイル（拡張子.mot、図中(1)xxxx.mot）を作成し、MCU（RX26T）に書き込み、プログラムを実行します（Before OTA）。

次に PC からの指示により OTA を開始します（OTA in progress）。OTA 実行中、RX26T は AWS から更新イメージ（図中(3)xxxx.rsu）をダウンロードしながら、RX26T のバッファ面に更新イメージを書き込みます。この間メイン面のプログラムは OTA と並行して実行されます。バッファ面のファームウェアアップデートが完了し PC からのリセット指示があると、RX26T はバンクスワップ（Swap bank）を実行し、自己リセットを行い、更新イメージの実行を開始します（After OTA）。

初期イメージと更新イメージについては「3.2 初期イメージと更新イメージ」で詳しく説明します。

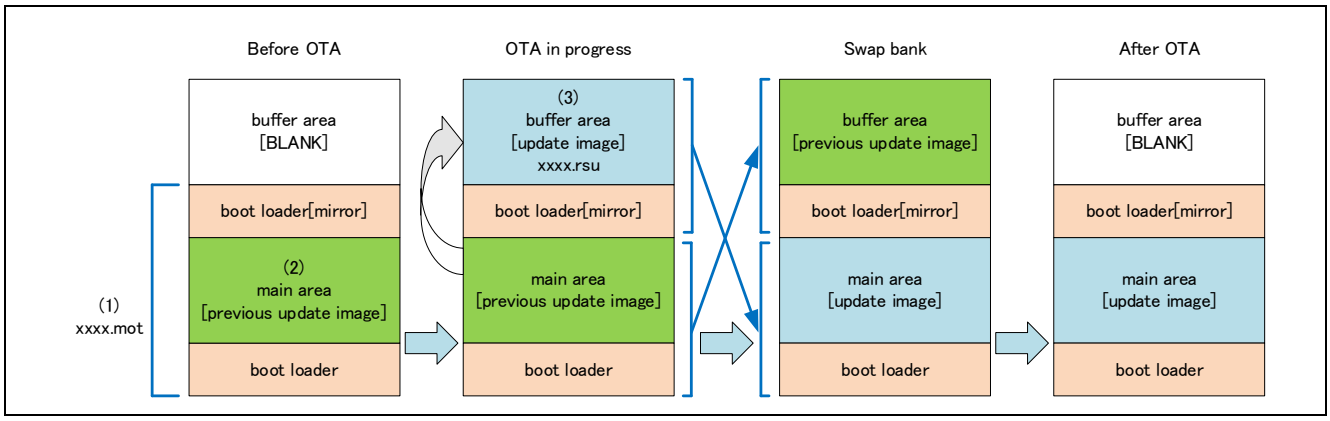

図 3-1 OTA のイメージ

3.2 初期イメージと更新イメージ

本サンプルプログラムはブートローダと組み合わせて動作するように作られているため、プロジェクト単体では動作しません。プログラムを動作させるためには MCU（RX26T）上で動作可能な初期イメージおよび Amazon S3 にアップロードする更新イメージを作成する必要があります。初期イメージと更新イメージを作成するためにモトローラ S レコードファイル（拡張子 mot）が必要です。モトローラ S レコードファイルは本サンプルプログラムをビルドして生成します。

3.2.1 プロジェクトのビルド

本サンプルプログラムは3つのプロジェクトで構成します。各プロジェクトをビルドしてモトローラ S レコードファイルを準備してください。ビルド方法は「6 プロジェクトのビルド手順」を参照してください。なお、e2studio でビルドを行う場合は、FIT モジュールの有無によりビルド結果が変わりますので、必ず「6 プロジェクトのビルド手順」に従い、FIT がダウンロードされていることを確認してからビルドを行ってください。

表 3-1 プロジェクト

プロジェクト名	内容
r01an7203_rx26t_boot_loader	ブートローダ
r01an7203_rx26t_fwup1	メインプログラム RX26T_MCBA_MCILV1_SPM_LESS_FOC_1SHUNT_E2S_V110 の処理を実行しながらファームウェアアップデートを実行
r01an7203_rx26t_fwup2	メインプログラム RX26T_MCBA_MCILV1_SPM_LESS_FOC_E2S_V110 の処理を実行しながらファームウェアアップデートを実行

3.2.2 初期イメージ作成方法

初期イメージはブートローダとメインプログラムの組み合わせで作成します。初期イメージの構成は「3.1 本サンプルプログラムの OTA 概要」の「(1)xxxx.mot」のとおりで、「表 3-1 プロジェクト」の中から「表 3-2 初期イメージの組み合わせ」の組み合わせで作成します。「3.1 本サンプルプログラムの OTA 概要」の「(2)main_area」に「r01an7203_rx26t_fwup1」、または「r01an7203_rx26t_fwup2」が配置され、MCU（RX26T）で実行可能な初期イメージとなります。

初期イメージは「RX ファミリ ファームウェアアップデートモジュール Firmware Integration Technology（R01AN6850）」の「4. Renesas Image Generator」に記載の専用プログラム（image-gen.py）で生成します。なお専用プログラムのオプション（-vt ecdsa）は不要です（sha256 が選択されるため署名生成／検証用鍵は不要です）。

コマンド入力例

```
>python image-gen.py -iup r01an7203_rx26t_fwup1.mot -ip
RX26T_DualBank_ImageGenerator_PRM.csv -o initial_firm -ibp
r01an7203_rx26t_boot_loader.mot
```

表 3-2 初期イメージの組み合わせ

ー	組み合わせ
初期イメージ 1	「r01an7203_rx26t_boot_loader.mot」＋「r01an7203_rx26t_fwup1.mot」
初期イメージ 2	「r01an7203_rx26t_boot_loader.mot」＋「r01an7203_rx26t_fwup2.mot」

3.2.3 更新イメージの作成方法

更新イメージ（拡張子 rsu）の構成は「3.1 本サンプルプログラムの OTA 概要」の「(3)xxxx.rsu」のとおりであり、「r01an7203_rx26t_fwup1.mot」、または、「r01an7203_rx26t_fwup2.mot」から作成します。更新イメージも初期イメージと同様「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（R01AN6850）」の「4. Renesas Image Generator」に記載の専用プログラム（image-gen.py）で生成します。専用プログラムのオプション（-vt ecdsa）は不要です（sha256 が選択されるため署名生成／検証用鍵は不要です）。

ここで作成した更新イメージを Amazon S3 に配置してください。

コマンド入力例

```
>python image-gen.py -iup r01an7203_rx26t_fwup1.mot -ip
RX26T_DualBank_ImageGenerator_PRM.csv -o RX26T_1Shunt
```

3.2.4 Amazon S3 のファイル構成

Amazon S3 のファイル構成の例を以下に示します。

表 3-3 Amazon S3 のファイル構成

Amazon S3 構成	説明
Amazon S3	―
└ rx26t_Sample	バケット名
├ RX26T_1Shunt.rsu	rsu ファイル（更新イメージ 1）
└ RX26T_2Shunt.rsu	rsu ファイル（更新イメージ 2）

3.3 モータ制御 UI の設定

本サンプルプログラムのモータ制御 UI は流用元から MAIN_UI_BOARD に変更しています。
モータの制御はボードのスイッチやボリュームコントロールで行うことが出来ます。

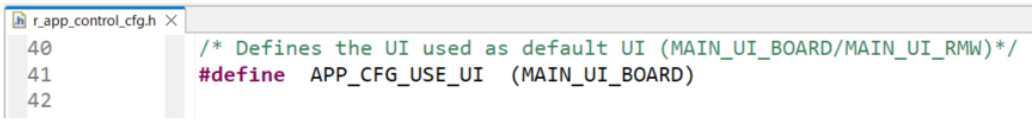

図 3-2 モータ制御 UI の変更方法

3.4 AWS の準備

AWS を使用して OTA を行うためには、AWS での事前準備が必要です。AWS のアカウントは、お客様ご自身で準備いただく必要があります。AWS のアカウントを準備いただいた後、「モノの登録」や、「証明書の発行」など、いくつかの手続きが必要になります。これらの手続きについては「4 AWS 環境構築」を参照してください。

4. AWS 環境構築

4.1 デバイスを AWS IoT に登録する

以下のチュートリアル（デバイスを AWS IoT に登録する）を参考に AWS の設定をしてください。
・「AWS IoT のエンドポイントを確認する」まで行ってください。
・証明書をダウンロードする際は「デバイス証明書、パブリックキーファイル、プライベートキーファイルに加えルート CA 証明書（Amazon ルート CA 1）もダウンロードしてください。
デバイスを AWS IoT に登録する · renesas/amazon-freertos Wiki · GitHub

4.2 Lambda の準備

4.2.1 関数を作成する

AWS Lambda のコンソールで、「関数」➡「関数の作成」をクリックします。

図 4-1 AWS Lambda 関数の作成開始

1. 「一から作成」にチェックを入れます。
2. 関数名を入力します（本書では、RX26T_Lambda と入力した前提で説明します）。
3. プルダウンメニューから、「Python3.11」を選択します。
4. 「関数の作成」をクリックすると、関数が作成されます。

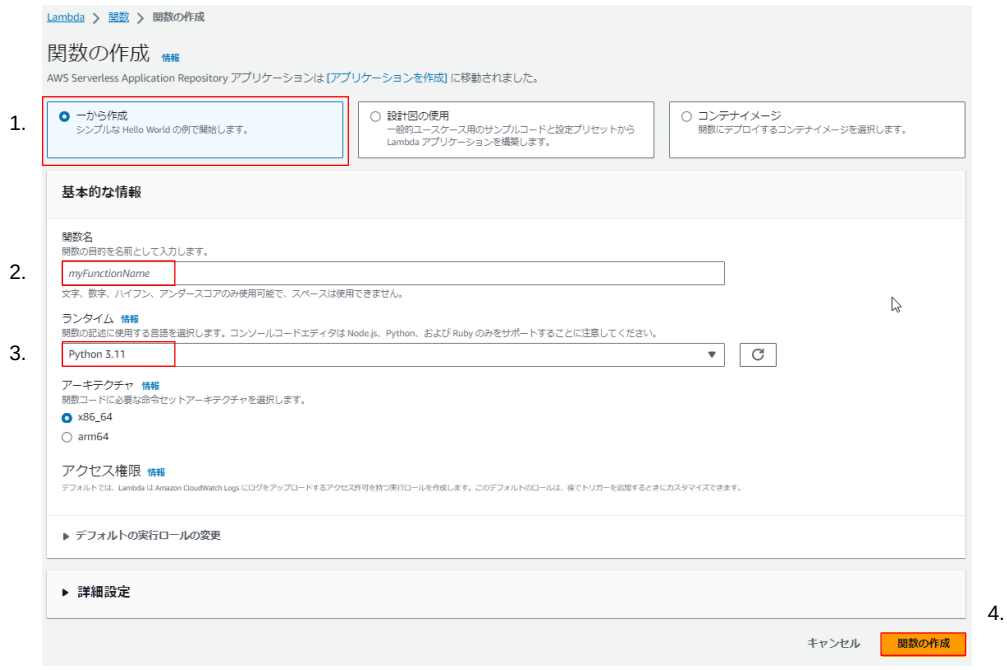

図 4-2 AWS Lambda 関数の作成

4.2.2 トリガの追加

「トリガの追加」をクリックします。

図 4-3 トリガ追加

1. 「AWS IoT」を選択します。
2. 「Custom IoT rule」にチェックを入れます。
3. 「Create a new rule」にチェックを入れます。
4. トリガ名称を入力します（本書では、RX26T_Lambda_Trigger と入力した前提で説明します）。
5. ルールを入力します（本書では、「SELECT * FROM ‘RX26T_Topic/FROM_EDGES’」を入力した前提で説明します）。AWS IoT は、このルールに従い MQTT メッセージを Lambda に伝えます。
6. 「追加」をクリックします。

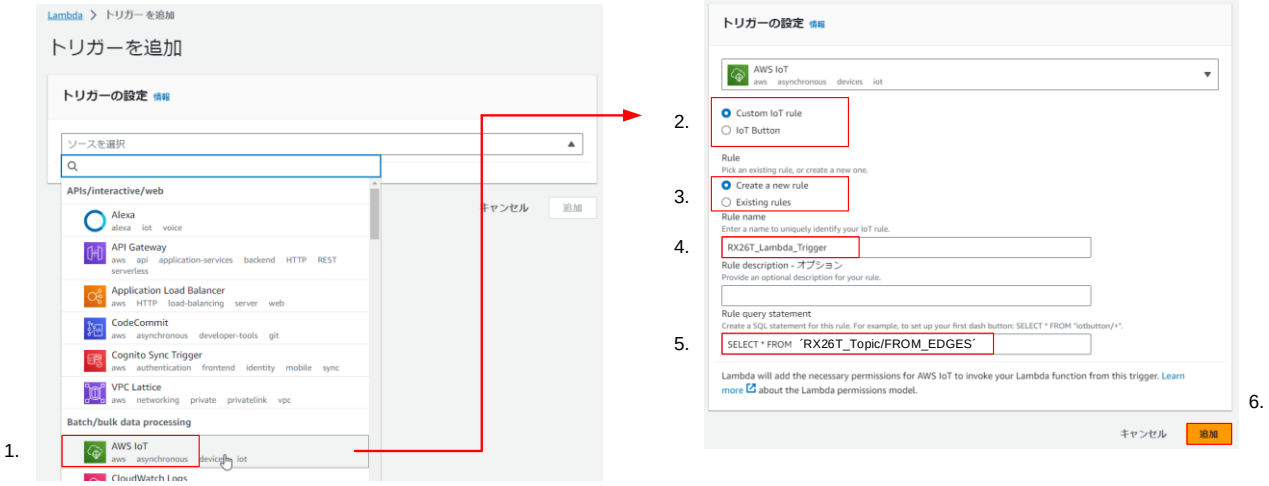

図 4-4 トリガ追加手順

4.2.3 ロールの設定

ロールを設定します。

1. 設定をクリックします。
2. アクセス権限をクリックします。
3. ロール名をクリックします。

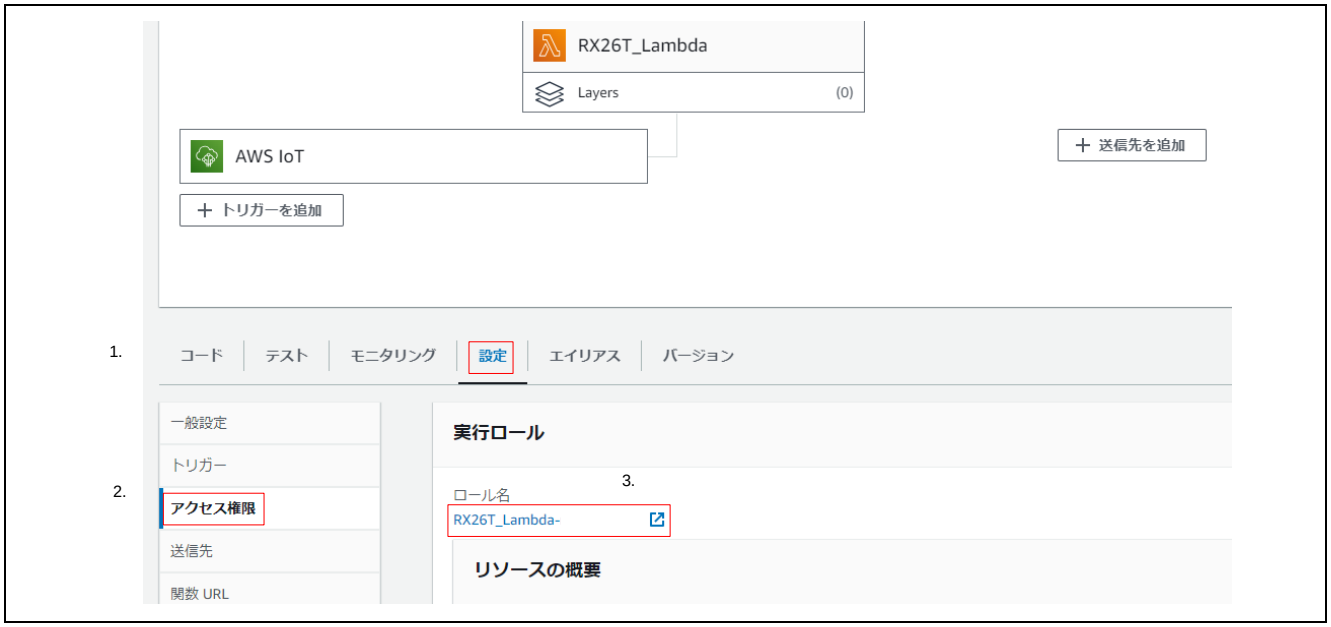

図 4-5 ロールの設定

「許可を追加」➡「ポリシーをアタッチ」をクリックします。

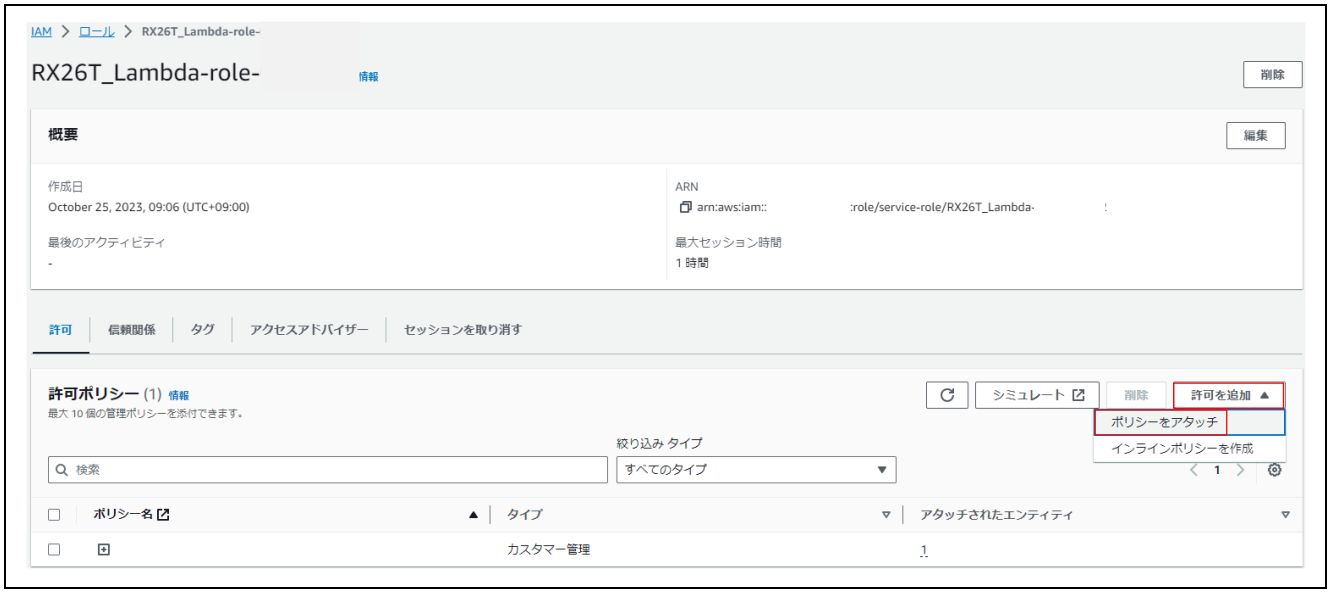

図 4-6 ロールにポリシーをアタッチ（初回）

1. 検索ウインドウに「S3」と入力し項目を絞り込みます。
2. 「AmazonS3FullAccess」にチェックを入れます。
3. 「許可を追加」をクリックします。

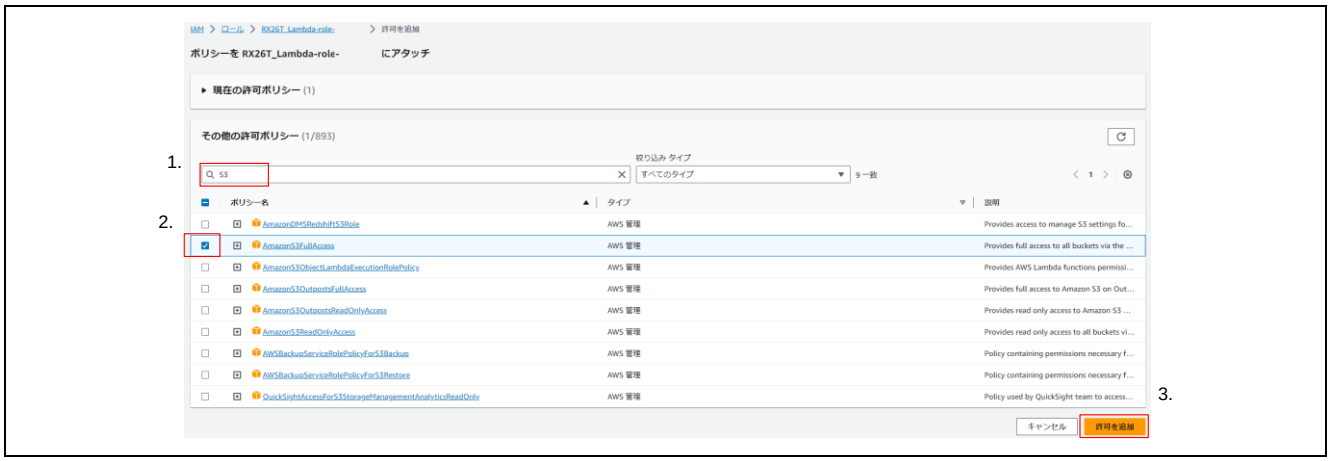

図 4-7　S3 のアクセス権を追加

再度、「許可を追加」➡「ポリシーをアタッチ」をクリックします。

図 4-8　ロールにポリシーをアタッチ（2 回目）

1. 検索ウインドウに「IoTData」と入力し項目を絞り込みます。
2. 「AWSIoTDataAccess」にチェックを入れます。
3. 「許可を追加」をクリックします。

図 4-9　AWS IoT のアクセス権を追加

4.2.4 Lambda のコードを登録する

「コード」をクリック➡「Lambda Function.py」をダブルクリックし lambda_function のタブを開きます。lambda_function タブのコードを「4.2.5 OTA 用 Lambda コード」に置き換えます。置き換え後は「Deploy」をクリックし、Lambda コードを Deploy してください。

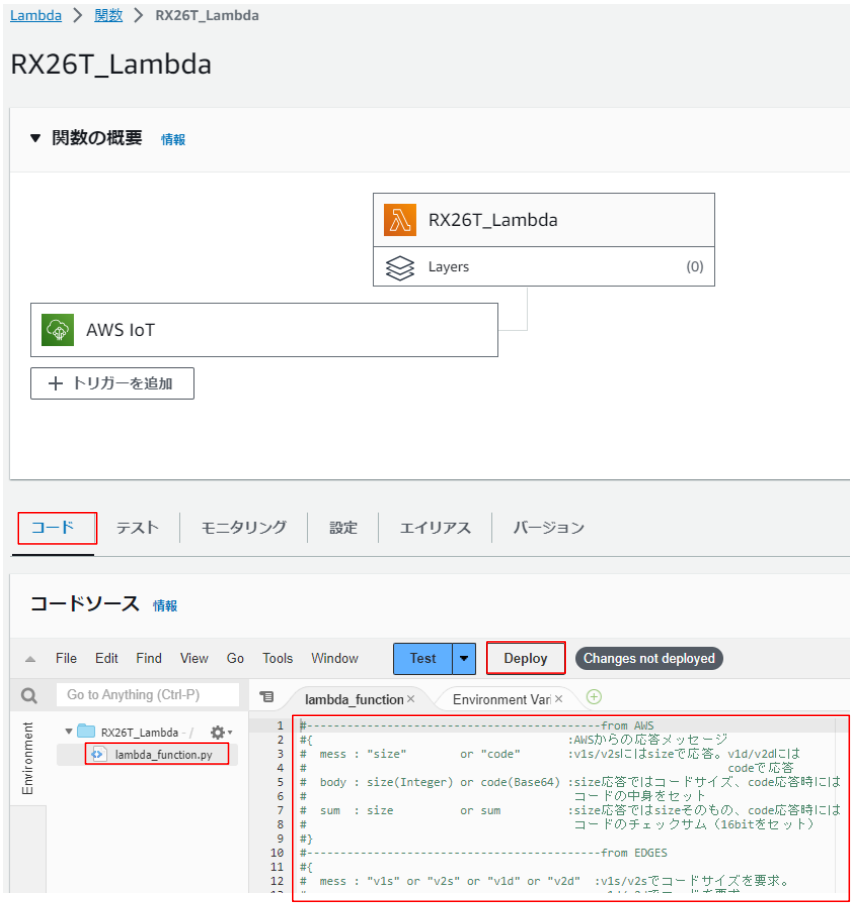

図 4-10 Lambda コードの設定

4.2.5 OTA 用 Lambda コード

OTA 用の Lambda コードの例を示します。

赤文字の部分については AWS 環境に合わせた名前を設定してください。

- topic：
  AWS から Publish する際のトピック名を指定します。
- BUCKET_NAME：
  Amazon S3 のバケット名を指定します。
- OBJECT_KEY_NAME1：
  「3. 2. 3 更新イメージの作成方法」で作成した一つ目の更新イメージの名前を指定します。
  ※Amazon S3 に配置したファイル名
- OBJECT_KEY_NAME2：
  「3. 2. 3 更新イメージの作成方法」で作成した二つ目の更新イメージの名前を指定します。
  ※Amazon S3 に配置したファイル名

```
#-------------------------------------------from AWS
#{                                      :AWS からの応答メッセージ
#  mess : "size"        or "code"       :v1s/v2s には size で応答。v1d/v2d には
#                                                  code で応答
#  body : size(Integer) or code(Base64) :size 応答ではコードサイズ、code 応答時には
#                                        コードの中身をセット
#  sum  : size          or sum          :size 応答では size そのもの、code 応答時には
#                                        コードのチェックサム（16bit をセット）
#}
#-------------------------------------------from EDGES
#{
#  mess : "v1s" or "v2s" or "v1d" or "v2d"  :v1s/v2s でコードサイズを要求。
#                                           v1d/v2d でコードを要求
#  seek :   -   or   -   or  seek or  seek  :v1s/v2s 時は無視される
#                                           v1d/v2d 時は seek 位置
#  size :   -   or   -   or  size or  size  :v1s/v2s 時は無視され、v1d/v2d 時は
#                                           要求するサイズを指定（バイト）
#}
import json
import boto3
import base64
def lambda_handler(event, context):                 #IotCore からコールされる
    ### IoT Core Topic send settings                #
    topic           = 'RX26T_Topic/FROM_AWS'        #トピック名を指定
    BUCKET_NAME     = 'rx26t_Sample'                #S3 のバケット名を指定
    OBJECT_KEY_NAME1= 'RX26T_1Shunt.rsu'            #更新イメージ 1 のファイル名 1 を設定
    OBJECT_KEY_NAME2= 'RX26T_2Shunt.rsu'            #更新イメージ 2 のファイル名 2 を設定
    mess = event['mess']                            #FROM_EDGES の JSON の mess を取得
    seek = int(event['seek'])                       #FROM_EDGES の JSON の seek を取得
    size = int(event['size'])                       #FROM_EDGES の JSON の size を取得
    iot  = boto3.client('iot-data')                 #IoTData
    s3   = boto3.client('s3')                       #S3
    sum  = 0                                        #チェックサムを初期化しておく
```

```
    if(mess=='v1s'): #v1s の場合、S3 の OBJECT_KEY_NAME1 のサイズを返す
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME1)
        mess     = "size"                          #mess は size
        body     = response["ContentLength"]       #S3 の ContentLength(サイズ)を取得
        sum      = body                            #body はサイズ
    if(mess=='v2s'): #v2s の場合、S3 の OBJECT_KEY_NAME2 のサイズを返す
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME2)
        mess     = "size"                          #mess は size
        body     = response["ContentLength"]       #S3 の ContentLength(サイズ)を取得
        sum      = body                            #body はサイズ
    if(mess=='v1d'): #v1d の場合、S3 の OBJECT_KEY_NAME1 を返す
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME1,
Range='bytes={}-{}'.format(seek, seek+(size-1)))
        mess     = "code"                          #↑指定された seek 位置から、
                                                   #  指定されたバイトを読み出す
        body     = response["Body"].read()         #読み出した内容を body にセット
        body_len = len(body)                       #sum の計算用に body の長さを取得
        for ii in range(body_len):                 #チェックサムを計算
            sum += body[ii]                        #
        sum = sum & 0xFFFF                         #16bit にする
        body     = base64.b64encode(body).decode('utf-8') #base64 エンコード
    if(mess=='v2d'): #v2d の場合、S3 の OBJECT_KEY_NAME2 を返す
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME2,
Range='bytes={}-{}'.format(seek, seek+(size-1)))
        mess     = "code"                          #↑指定された seek 位置から、
                                                   #  指定されたバイトを読み出す
        body     = response["Body"].read()         #読み出した内容を body にセット
        body_len = len(body)                       #sum の計算用に body の長さを取得
        for ii in range(body_len):                 #チェックサムを計算
            sum += body[ii]                        #
        sum = sum & 0xFFFF                         #16bit にする
        body     = base64.b64encode(body).decode('utf-8') #base64 エンコード

    payload = { "mess" : mess , "body" : body , "sum" : sum } #応答 JSON 用リスト

    iot.publish(                                   #Publish 要求
        topic = topic,                             #送り先
        qos = 1,                                   #
        payload = json.dumps(payload)              #本文
    )
```

4.2.6 プロジェクトのソースコード修正 1

プロジェクトのソースコードに Wi-Fi 接続情報と AWS 接続情報を入力します。
「src/sample_ota_add_on/sample_mqtt_config.c」にある 5 つの変数を設定してください。

- g_mqtt_broker_endpoint[] ➡「4 AWS 環境構築」で確認した endpoint の名前
- g_mqtt_subscriber[] ➡「4.2.5OTA 用 Lambda コード」で設定した Topic 名
- g_mqtt_publisher[] ➡「4.2.2 トリガの追加」でルールに指定した Topic 名
- g_wifi_ssid ➡ 接続するアクセスポイントの SSID
- g_wifi_password ➡ 接続するアクセスポイントのパスワード

※上記の変数は下図のように”” の中に入力してください

```
uint8_t  g_mqtt_broker_endpoint[] = "[redacted].ap-northeast-1.amazonaws.com";
uint16_t g_broker_endpoint_size   = sizeof(g_mqtt_broker_endpoint);

uint8_t  g_mqtt_subscriber[] = "RX26T_Topic/FROM_AWS";
uint16_t g_subscriber_size   = sizeof(g_mqtt_subscriber);

uint8_t  g_mqtt_publisher[] = "RX26T_Topic/FROM_EDGES";
uint16_t g_publisher_size   = sizeof(g_mqtt_publisher);

uint8_t  g_wifi_ssid[] = "[redacted]";
uint16_t g_wifi_size   = sizeof(g_wifi_ssid);

uint8_t  g_wifi_password[] = "[redacted]";
uint16_t g_password_size   = sizeof(g_wifi_password);
```

図 4-11 sample_mqtt_config.c

4.2.7 プロジェクトのソースコード修正 2

「4.1 デバイスを AWS IoT に登録する」でダウンロードした Amazon ルート CA 1 証明書とデバイス証明書およびプライベートキーをソースコードに反映します。

「src/sample_ota_add_on/sample_mqtt_config.c」にある 3 つの変数を以下のように反映してください。

- g_mqtt_root_cetificate_pem[] ➡図 4-12：Amazon ルート CA 1 証明書
- g_mqtt_cetificate_pem_cert[] ➡図 4-13：デバイス証明書
- g_mqtt_private_pem_key[] ➡図 4-14：プライベートキー

注：以下に注意してください。
- 最終行を除く各行の最後に「\n」が必要であること。
- 各行はダブルクォーテーションで囲われること。
- 最終行を除く各行の最終がバックスラッシュで終わっていること。

図 4-12 ルート CA 証明書の反映

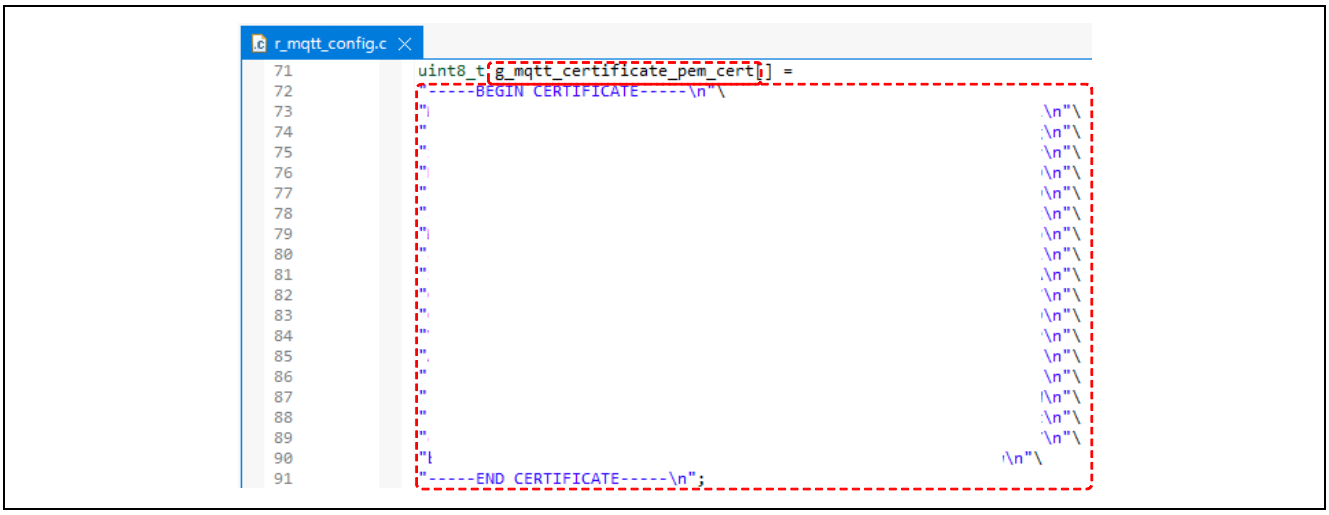

図 4-13 デバイス証明書の反映

図 4-14　プライベートキーの反映

5. サンプルプログラム

5.1 システム構成

本サンプルプログラムのシステム構成を「図 5-1 AWS デモのシステム構成」に示します。

更新イメージは、あらかじめ AWS の Amazon S3 格納しておきます。OTA に関する命令は PC から MCK-RX26T に対して行います。PC から更新命令（FWUP1 or FWUP2）を入力すると MCU（RX26T）は MQTT を使用して AWS IoT Core に対し、更新イメージのダウンロード要求を行います。AWS IoT Core は、この要求を AWS Lambda に転送します。AWS Lambda は要求に応じた更新イメージを Amazon S3 から読み出し MQTT を使用して Publish を行います。

MCU は MQTT を使用して Subscribe を行い、更新イメージを取得します。取得した更新イメージは MCU の Flash ROM のバッファ面に書き込みます。Flash ROM へのすべての更新イメージの書き込みが完了すると、MCU は PC からのリセット命令を待ちます。PC からリセット命令（RESET）を入力する前にボード上のジャンパを更新イメージに合わせて切り替えます。PC からリセット命令（RESET）を入力すると MCU はバンクを切り替えた後に自己リセットを行います。これにより更新イメージ（新しいプログラム）が動作を開始します。

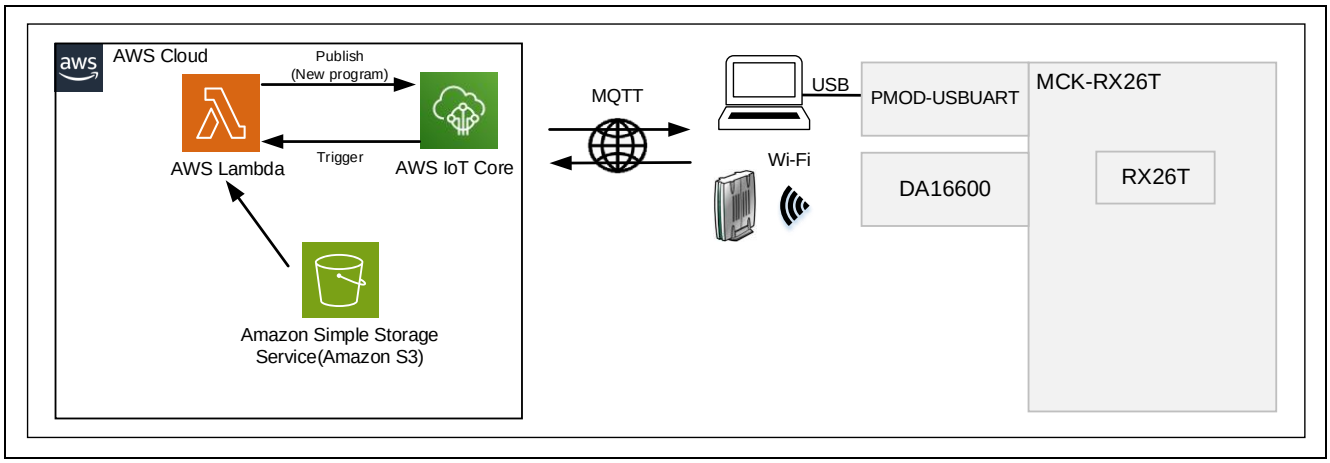

図 5-1 AWS デモのシステム構成

5.2 プロジェクト構成

本サンプルプログラムは 3 つのプロジェクトで構成します。1 つ目はブートローダです。本ブートローダはメイン面を起動する機能しか実装していません。残りの 2 つはメインプログラムで「永久磁石同期モータのセンサレスベクトル制御 - MCK 用（R01AN6858）」のプロジェクトに OTA（sample_ota_add_on 関数）をアドオンしたプロジェクトです。アドオン構成については「5.5.9 サンプルプログラムのフローチャート」を参照してください。

これらのプロジェクトはブートローダとセットで使用するように作られているため単体では動作しません。ブートローダと組み合わせる方法については「3.2 初期イメージと更新イメージ」を参照してください。初期イメージと更新イメージは、「表 5-1 プロジェクト一覧」のプロジェクトをビルドして生成したモトローラ S ファイルを使用して生成します。

表 5-1　プロジェクト一覧

プロジェクト名	内容
r01an7203_rx26t_boot_loader	ブートローダ
r01an7203_rx26t_fwup1	メインプログラム RX26T_MCBA_MCILV1_SPM_LESS_FOC_1SHUNT_E2S_V110 の処理を実行しながらファームウェアアップデートを実行
r01an7203_rx26t_fwup2	メインプログラム RX26T_MCBA_MCILV1_SPM_LESS_FOC_E2S_V110 の処理を実行しながらファームウェアアップデートを実行

5.3　セクション設定

本プロジェクトはブートローダとメインプログラムを組み合わせて使用可能にするために、それぞれのプロジェクトにセクションを設定済みです。セクション設定は「RX ファミリ ファームウェアアップデートモジュール Firmware Integration Technology（R01AN6850）」の「6.2.2.1 デュアルバンク方式のデモプロジェクトのメモリマップ」に従い設定しています。

5.4　ソフトウェア構成

本サンプルプログラムのソフトウェア構成を示します。本サンプルソフトウエアは MCU（RX26T）と AWS の 2 つで構成されています。

5.4.1　MCU（RX26T）のソフトウェア構成

RX26T のソフトウェア構成を「図 5-2　MCU（RX26T）のソフトウェア構成」に示します。

RX26T のソフトウェアは大きく 2 つのプログラムで構成しています。1 つ目はユーザプログラムである Main program です。2 つ目は OTA 用のプログラムである OTA program（sample_ota_add_on）です。OTA program は OTA FIT（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（ R01AN6850））を使用して構成しており SCI を使用して AT Command により DA16600 Pmod™ Board を制御します。OTA program は Main program にアドオンする形で構成しています。

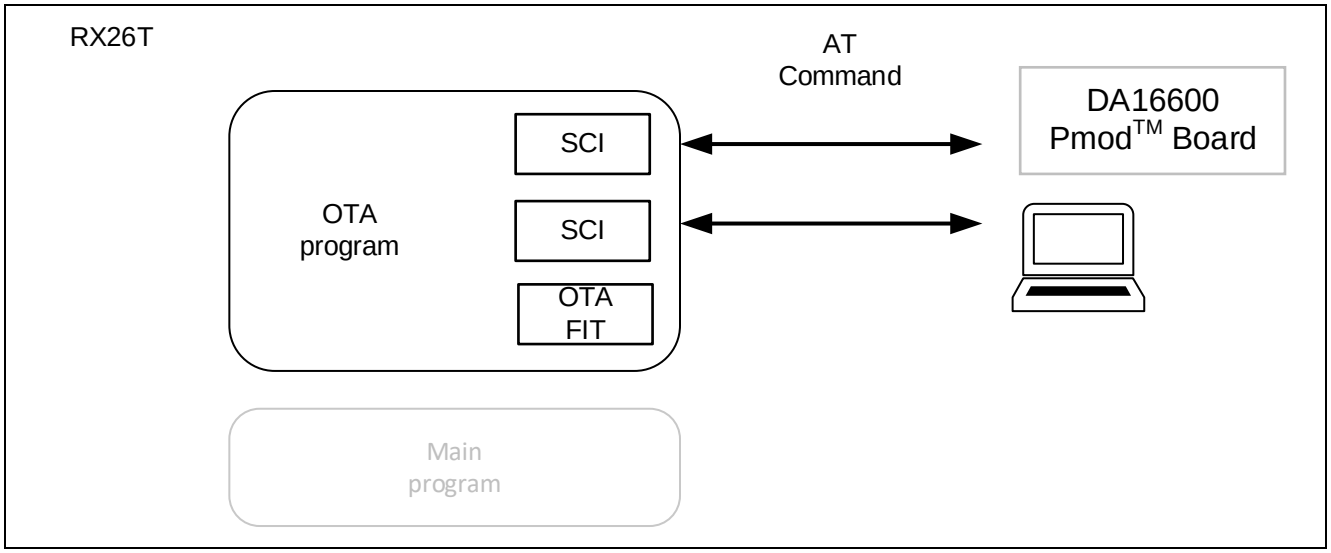

図 5-2　MCU（RX26T）のソフトウェア構成

5.4.2 MCU（RX26T）のソフトウェア構成詳細（ファイル構成詳細）

OTA program（sample_ota_add_on）は、7 種類の.c 、.h のペアで構成しています。
それぞれの依存関係を「図 5-3 MCU（RX26T）のソフトウェア構成詳細」に示します。

- sample_ota_add_on.c/.h：
  OTA program のメインルーチンに相当する関数を含む TOP 階層のファイルです。
- sample_aws.c/.h：
  AWS 制御を行います。AWS に関係する関数はすべてこのファイルに定義しています。
- sample_da16600_uart.c/.h：
  da16600 との UART 通信を行います。ハードウェアリソースとして RSCI11 を使用します。
- sample_mqtt_config.c/.h：
  AWS の証明書や Wi-Fi SSID/Password などを管理するファイルです。
- base64_decode.c/.h
  base64_decode 処理を行います。
- sample_fwup.c/h：
  ファームウェアアップデート FIT を使用しファームウェアアップデートを行います。
- sample_pc_uart.c/.h：
  PC との UART 通信を行います。ハードウェアリソースとして RSCI8 を使用します。

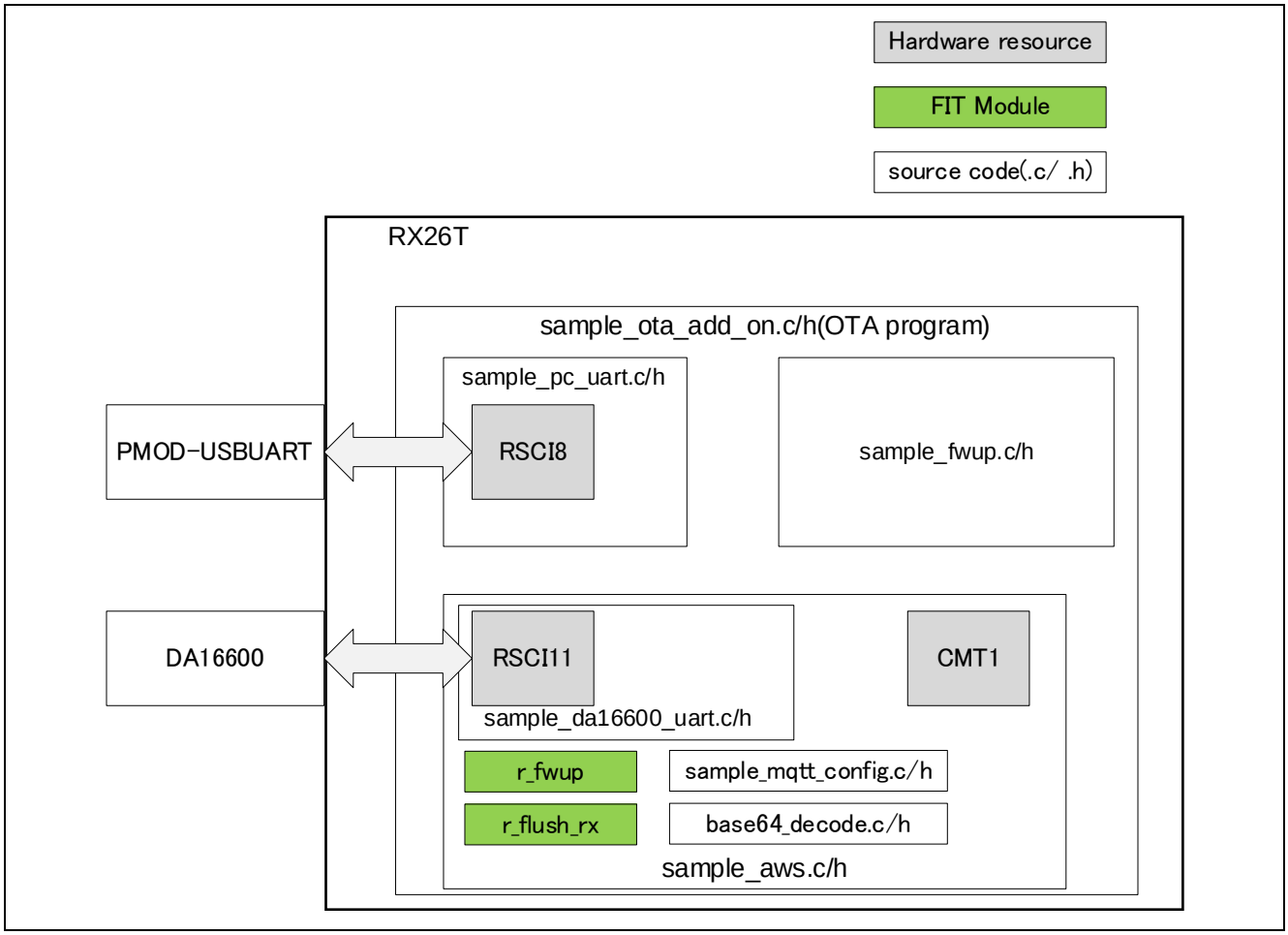

図 5-3 MCU（RX26T）のソフトウェア構成詳細

5.4.3 AWS のソフトウェア構成

AWS のソフトウェアは、AWS Lambda（Python3.11）で構成します。AWS では Amazon S3 や AWS IoT Core を制御することができる boto3 というライブラリを使用して環境を構築します。

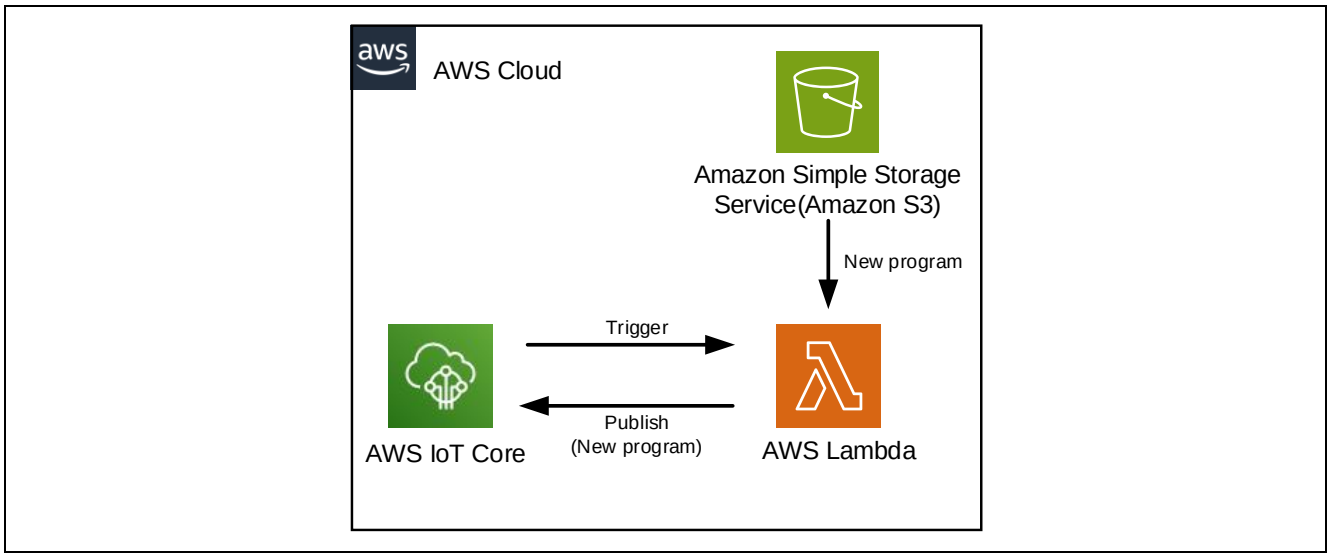

図 5-4 AWS のソフトウェア構成

5.5 サンプルプログラムの構成

5.5.1 概要

本サンプルプログラムは「3.2 初期イメージと更新イメージ」に記載のとおりブートローダと組み合わせて動作するように作られています。またファームウェア アップデートの機能は「永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）」のサンプルプログラムの main 関数にアドオンする形で構成しています。

5.5.2 使用端子一覧

本サンプルプログラムの OTA 用にアドオンした RX26T の使用端子一覧を以下に示します。
OTA 以外の使用端子については「永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）」を参照してください。

表 5-2 使用端子一覧

端子名	入出力	用途
PD5/TXD11	出力	DA16600 Pmod™ Board の GPIOC7_TXD_HOST に接続
PB6/RXD11	入力	DA16600 Pmod™ Board の GPIOC6_RXD_HOST に接続
PD0/TXD8	出力	Pmod-USBUART の RX に接続
PD1/RXD8	入力	Pmod-USBUART の TX に接続
VCC	―	DA16600 Pmod™ Board に 3.3V を供給
VSS	―	DA16600 Pmod™ Board の VSS に接続

5.5.3 使用する周辺機能

本サンプルプログラムで使用する周辺機能を以下に示します。
OTA 以外の使用端子については「永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）」を参照してください。

表 5-3 使用する周辺機能一覧

周辺機能	用途
RSCI11	DA16600 Pmod™ Board との UART 通信
RSCI8	PC との UART 通信
CMT1	MQTT 通信のインターバル制御

5.5.4 周辺機能の設定

本サンプルプログラムで使用している周辺機能の設定はスマート・コンフィグレータ（V2.19.0）のコード生成機能を用いています。スマート・コンフィグレータの設定条件を以下に示します。ベースとする「永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）」から追加した周辺機能を記載します。

表 5-4　Config_RSCI11 の設定（DA16600 Pmod™ Board 接続用）

項目	設定
シリアル通信方式	調歩同期式
スタートビット検出設定	RXD11 端子の Low レベル
データ・ビット長	8 ビット
パリティ設定	禁止
ストップビット設定	1 ビット
データ転送方向設定	LSB ファースト
転送速度設定	• 転送クロック：内部クロック • ビットレート：115200bps • ビットレートモジュレーション機能有効 • SCK11 端子機能：SCK11 を使用しない
ノイズフィルタ設定	ノイズフィルタ無効
ハードウェアフロー制御設定	ハードウェアフロー制御設定：禁止
データ処理設定	送信データ処理：割り込みサービスルーチンで処理 受信データ処理：割り込みサービスルーチンで処理
割り込み設定	受信エラー割り込み許可 優先順位：レベル 15
コールバック機能設定	使用しない
入出力端子	• 出力：TXD11（PB5） • 入力：RXD11（PB6）

表 5-5　Config_RSCI8 の設定（PC 接続用）

項目	設定
シリアル通信方式	調歩同期式
スタートビット検出設定	RXD8 端子の Low レベル
データ・ビット長	8 ビット
パリティ設定	禁止
ストップビット設定	1 ビット
データ転送方向設定	LSB ファースト
転送速度設定	• 転送クロック：内部クロック • ビットレート：115200bps • ビットレートモジュレーション機能有効 • SCK8 端子機能：SCK8 を使用しない
ノイズフィルタ設定	ノイズフィルタ無効
ハードウェアフロー制御設定	ハードウェアフロー制御設定：禁止
データ処理設定	送信データ処理：割り込みサービスルーチンで処理 受信データ処理：割り込みサービスルーチンで処理
割り込み設定	受信エラー割り込み許可 優先順位：レベル 15
コールバック機能設定	使用しない
入出力端子	• 出力：TXD8（PD0） • 入力：RXD8（PD1）

表 5-6　r_flash_rx（FIT v5.10）の設定

項目	設定
Parameter_check	Enable_parameter_check
Enable code flash programming	Includes code to ROM area
Enable BGO/Non-blocking data flash operations	Forces data flash API function to block until completed.
Enable BGO/Non-blocking code flash operations	Forces ROM API function to block until completed.
Enable code flash self-programming	Programming code flash while executing in RAM.

表 5-7　r_fwup（FIT v2.01）の設定

項目	設定
Select the update mode	Dual bank
Select the function mode	use the User program
Main area start address	0xFFFC0000
Buffer area start address	0xFFF80000
Install area size	0x38000
Code flash block size	0x4000
Code flash write unit size	128
External flash Buffer area start address	0x0000
External flash block(Sector) size	4096
Data flash start address	0x00100000
Data flash block size	64
Data flash blocks	256
FWUP v1 compatibility Setting	Disable(default)
Select the algorithm Setting	SHA256
Disable Printf Output Setting	Enable(default)

表 5-8　CMT1 の設定

項目	設定
クロック設定	PCLKB/32
コンペアマッチ設定	• インターバル時間：10ms • コンペアマッチ割り込みを許可（CMI1） • 優先順位：レベル 15（割り込み禁止）

表 5-9　r_bsp の設定

項目	設定
Enable user studio charput function	Use user chaput function
User studio charput function name	my_sw_charput_function

※r_bsp のその他の設定は任意です。

5.5.5 ファイル構成

本サンプルプログラムのファイル構成を以下に示します。

表 5-10 ファイル構成

フォルダ名、ファイル名	説明
src	プログラム格納用フォルダ
├ app	メイン処理
│ ├ board_ui(folder)	ボード UI 関連定義（R01AN6858 参照）
│ ├ cfg(folder)	アプリ層の Configuration 定義（R01AN6858 参照）
│ ├ main	テーブル検索、直線補間処理プログラム
│ ├ └ main.c/h 注 1	ユーザメイン関数
│ └ rmw(folder)	RMW の Analyzer UI 関連関数定義（R01AN6858 参照）
├ motor_module(folder)	モータ制御モジュール関連定義（R01AN6858 参照）
├ QE_Motor	QE for Motor 生成ファイル（R01AN6858 参照）
└ src	src フォルダ
├ sample_ota_add_on 注 1	OTA 関連プログラム
│ ├ base64_decode.c/h 注 1	Base64 デコード
│ ├ sample_aws.c/h 注 1	AWS 制御プログラム
│ ├ sample_da16600_uart.c/h 注 1	DA16600 制御プログラム
│ ├ sample_fwup.c/h 注 1	ファームウェアアップデート制御プログラム
│ ├ sample_mqtt_config.c/h 注 1	MQTT 接続設定
│ ├ sample_ota_add_on.c/h 注 1	ota_add_on メインプログラム
│ └ sample_pc_uart.c/h 注 1	PC 通信関連
└ smc_gen	スマート・コンフィグレータ生成
├ Config_CMT0	
├ Config_CMT1 注 1	
├ Config_IWDT	
├ Config_MTU3_MTU4	
├ Config_POE	
├ Config_PORT	
├ Config_RSCI8 注 1	
├ Config_RSCI11 注 1	
├ Config_S12AD0	
├ Config_S12AD2	
├ general	
├ r_bsp	
├ r_config	
├ r_flash_rx	
├ r_fwup 注 2	
└ r_pincfg	

注 1. ベースとする「永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）から追加、変更を加えたファイルです。注 1 が付いていないファイルやフォルダについては前記 R01AN6858 を参照してください。

注 2. r_fwup/src/r_fwup_wrap_flash.c の 129 行目および 155 行目をコメントアウトしています。

5.5.6 変数一覧

本サンプルプログラムで使用する変数一覧を以下に示します。

ベースとする「永久磁石同期モータのセンサレスベクトル制御 － MCK 用（R01AN6858）」から追加した変数を記載します。追加した変数以外は R01AN6858 を参照ください。

表 5-11 サンプルプログラムで使用する変数一覧

変数名	型	内容
g_sample_aws_code_buf[260]	uint8_t	ダウンロードしたプログラムコード
g_sample_aws_code_size	uint16_t	ダウンロードしたコードサイズ
g_sample_da16600_uart	struct	da16600 通信ステータス
g_mqtt_root_certificate_pem	uint8_t	ルート証明書
g_mqtt_certificate_pem_cert	uint8_t	コード署名証明書
g_mqtt_private_pem_key	uint8_t	プライベートキー
g_mqtt_broker_endpoint	uint8_t	AWS エンドポイント
g_mqtt_broker_port	uint8_t	MQTT broker ポート番号
g_mqtt_subscriber	uint8_t	AWS モノの名前
g_mqtt_publisher	uint8_t	MQTT トピック
g_wifi_ssid	uint8_t	アクセスポイントの SSID
g_wifi_password	uint8_t	アクセスポイントのパスワード
g_root_certificate_pem_size	uint16_t	ルート証明書の文字数
g_certificate_pem_cert_size	uint16_t	コード署名証明書の文字数
g_private_pem_key_size	uint16_t	プライベートキーの文字数
g_broker_endpoint_size	uint16_t	AWS エンドポイントの文字数
g_broker_port_size	uint16_t	MQTT broker ポート番号の文字数
g_subscriber_size	uint16_t	AWS モノの名前の文字数
g_publisher_size	uint16_t	MQTT トピックの文字数
g_wifi_size	uint16_t	アクセスポイントの SSID の文字数
g_password_size	uint16_t	アクセスポイントのパスワードの文字数
g_sample_pc_uart	struct	PC 通信ステータス

5.5.6.1 定数一覧

ベースとする「永久磁石同期モータのセンサレスベクトル制御 － MCK 用（R01AN6858）」から追加した定数はありません。定数一覧は R01AN6858 を参照ください。

5.5.7 グローバル関数一覧

本サンプルプログラムで使用するグローバル関数一覧を以下に示します。

ベースとする「永久磁石同期モータのセンサレスベクトル制御 － MCK 用（R01AN6858）」から追加、変更した関数を記載します。

表 5-12 サンプルプログラムで使用するグローバル関数一覧

関数名	概要	
main	メインループに sample_ota_add_on()を追加	追加
base64_decode	base64 のデコードを行います。	新規
sample_aws_send_clieant_credential	DA16600 に証明書を設定します。	新規
sample_aws_mqtt_start	AWS との MQTT 通信のための接続を行います。	新規
sample_aws_cmt_callback	CMT 用コールバック関数です。	新規
sample_aws_init	sample_aws の初期設定を行います。	新規
sample_aws_publish_size	AWS に更新イメージのサイズを問い合わせます。	新規
sample_aws_subscribe_size	AWS から更新イメージのサイズを受け取ります。	新規
sample_aws_publish_rsu	AWS に更新イメージを要求します。	新規
sample_aws_subscribe_rsu	AWS から更新イメージを受け取ります。	新規
sample_aws_init_state	AWS 処理ステートを初期化します。	新規
sample_da16600_uart_send_at_command	DA16600 に AT コマンドを送信します。	新規
sample_da16600_uart_init	RSCI11 の Open/Start を行います。	新規
sample_da16600_uart_read_buf	内部 Ring buffer を Read buffer に転送します。	新規
sample_da16600_uart_recv_callback	RSCI11 の受信完了コールバック関数です。	新規
sample_da16600_uart_analyze_buf	DA16600 から受信した内容を解析します。	新規
sample_fwup_init	ファームウェアアップデート FIT を Open します。	新規
sample_fwup_bunkswap	バンクスワップを行います。	新規
sample_write_image	バッファ面にセルフプログラミングを行います。	新規
sample_ota_add_on	OTA 用のメインプログラムです。	新規
sample_pc_uart_init	RSCI8 の Open/Start を行います。	新規
sample_pc_uart_read_buf	内部 Ring buffer を Read buffer に転送します。	新規
sample_pc_uart_recv_callback	RSCI8 の受信完了コールバック関数です。	新規

5.5.8 関数仕様

本サンプルプログラムの関数仕様を以下に示します。

[関数名]main

概要	メイン処理
ヘッダ	なし
宣言	void main (void)
説明	メイン関数のメインループに sample_ota_add_on()関数を追加します。 sample_ota_add_on()関数は PC との通信、DA16600 Pmod™ Board の制御および OTA を行います。
引数	なし
戻り値	なし
備考	なし

[関数名] base64_decode

概要	Base64 のデコードを行います。
ヘッダ	base64_decode.h
宣言	uint32_t base64_decode(uint8_t *source, uint8_t *result, uint32_t size);
説明	base64 デコードを行います。
引数	uint8_t * source：base64 アドレス uint8_t * result ：base64 デコード後アドレス uint32_t size ：デコードサイズ
戻り値	デコード後のサイズ（バイト数）
備考	なし

[関数名] sample_aws_send_clieant_credential

概要	root 証明書、デバイス証明書および private_key を送信します。
ヘッダ	sample_aws.h
宣言	void sample_aws_send_clieant_credential (uint8_t * st);
説明	DA16600 Pmod™ Board に AT コマンドで証明書を送信します。
引数	uint8_t * st：ota_st のアドレスを指定します。
戻り値	なし
備考	なし

[関数名] sample_aws_mqtt_start

概要	Wi-Fi 接続および AWS との接続を行います。
ヘッダ	sample_aws.h
宣言	void sample_aws_mqtt_start (uint8_t * st);
説明	DA16600 を制御し Wi-Fi に接続します。Wi-Fi 接続後 AWS との接続を行い、 MQTT 通信を確立します。
引数	uint8_t * st：ota_st のアドレスを指定します。
戻り値	なし
備考	なし

[関数名] sample_aws_cmt_callback

概要	CMT1 用コールバック関数です。
ヘッダ	sample_aws.h
宣言	void sample_aws_cmt_callback (void);
説明	コンペアマッチするごとに内部変数（millis_cnt）を+10 します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_aws_init

概要	初期化関数です。
ヘッダ	sample_aws.h
宣言	void sample_aws_init (void);
説明	CMT1 の Open/Start を行います。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_aws_publish_size

概要	AWS に更新イメージのサイズを問い合わせます。
ヘッダ	sample_aws.h
宣言	void sample_aws_publish_size (uint8_t * upper_st, uint8_t no);
説明	AWS に更新イメージのサイズを問い合わせます。
引数	uint8_t * upper_st ：ota_st のアドレスを指定します。 uint8_t * no ：更新イメージの番号（1 or 2）を指定します。
戻り値	なし
備考	なし

[関数名] sample_aws_subscribe_size

概要	AWS から更新イメージのサイズを受け取ります。
ヘッダ	sample_aws.h
宣言	void sample_aws_subscribe_size (uint8_t * upper_st);
説明	AWS から更新イメージのサイズを受け取ります。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_aws_publish_rsu

概要	AWS に更新イメージを要求します。
ヘッダ	sample_aws.h
宣言	void sample_aws_publish_rsu (uint8_t * upper_st, uint8_t no);
説明	AWS に更新イメージを要求します。
引数	uint8_t * upper_st ：ota_st のアドレスを指定します。 uint8_t * no ：更新イメージの番号（1 or 2）を指定します。
戻り値	なし
備考	なし

[関数名] sample_aws_subscribe_rsu

概要	AWS から更新イメージを受け取ります。
ヘッダ	sample_aws.h
宣言	sample_aws_subscribe_rsu (uint8_t * upper_st, uint8_t * success);
説明	AWS から更新イメージを受け取ります。
引数	uint8_t * upper_st ：ota_st のアドレスを指定します。 uint8_t * no ：更新イメージの番号（1 or 2）を指定します。
戻り値	なし
備考	なし

[関数名] sample_aws_init_state

概要	AWS 処理ステートを初期化します。
ヘッダ	sample_aws.h
宣言	void sample_aws_init_state (void);
説明	AWS 処理ステートを初期化します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_da16600_uart_send_at_command

概要	DA16600 に AT コマンドを送信します。
ヘッダ	sample_da16600_uart.h
宣言	void sample_da16600_uart_send_at_command (uint8_t * const com, uint16_t len);
説明	DA16600 に AT コマンドを送信します。
引数	uint8_t * const com：送信する AT コマンドのアドレスを指定します。 uint16_t len　　：送信する AT コマンドの長さを指定します。
戻り値	なし
備考	なし

[関数名] sample_da16600_uart_init

概要	RSCI11 を Open/Start します。
ヘッダ	sample_da16600_uart.h
宣言	void sample_da16600_uart_init (void);
説明	RSCI11 を Open/Start します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_da16600_uart_read_buf

概要	リードバッファに受信データをセットします。
ヘッダ	sample_da16600_uart.h
宣言	void sample_da16600_uart_read_buf (void);
説明	内蔵 Ring buffer からリードバッファにデータを転送します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_da16600_uart_recv_callback

概要	RSCI11 の受信完了時のコールバック関数です。
ヘッダ	sample_da16600_uart.h
宣言	void sample_da16600_uart_read_buf (void);
説明	DA16600 からの受信データを内蔵 Ring buffer にコピーします。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_da16600_uart_analyze_buf

概要	DA16600 からの受信データを解析します。
ヘッダ	sample_da16600_uart.h
宣言	uint8_t sample_da16600_uart_analyze_buf(uint8_t * p_resp)
説明	DA16600 からの受信データを、内蔵 Ring buffer にコピーします。
引数	uint8_t * p_resp：解析したいレスポンスの種類を指定します。
戻り値	uint8_t　　：0＝指定レスポンスなし、1＝指定レスポンスあり。
備考	なし

[関数名] sample_fwup_init

概要	ファームウェアアップデート FIT を Open します。
ヘッダ	sample_fwup.h
宣言	void sample_fwup_init (void);
説明	ファームウェアアップデート FIT を Open します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_fwup_bunkswap

概要	バンクスワップを行います。
ヘッダ	sample_fwup.h
宣言	void sample_fwup_bunkswap(void)
説明	ファームウェアアップデート FIT の R_FWUP_ActivateImage をコールし自己リセットを行い、更新イメージ（新しい更新プログラム）をスタートします。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_write_image

概要	Flash ROM への書き込みを行います。
ヘッダ	sample_fwup.h
宣言	e_fwup_err_t sample_write_image(e_fwup_area_t area, uint8_t * rsu, uint16_t size)
説明	ファームウェアアップデート FIT の R_FWUP_WriteImage をコールしセルフプログラミングを行います。
引数	e_fwup_area_t area : 更新面を指定します（バッファ面 Only です）。 uint8_t * rsu : 更新イメージを格納しているアドレスを指定します。 uint16_t size : 更新イメージのサイズを指定します（256 バイト単位で書き込みます）。
戻り値	なし
備考	なし

[関数名] sample_ota_add_on

概要	OTA program のメインルーチンです。
ヘッダ	sample_ota_add_on.h
宣言	void sample_ota_add_on (void);
説明	OTA のメインルーチンです。この関数を main()関数から繰り返しコールすることで、メインプログラムを実行中に OTA を実現します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_pc_uart_init

概要	RSCI8 を Open/Start します。
ヘッダ	sample_pc_uart.h
宣言	void sample_pc_uart_init (void);
説明	RSCI8 を Open/Start します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_pc_uart_read_buf

概要	リードバッファに受信データをセットします。
ヘッダ	sample_pc_uart.h
宣言	void sample_pc_uart_read_buf (void);
説明	内蔵 Ring buffer からリードバッファにデータを転送します。
引数	なし
戻り値	なし
備考	なし

[関数名] sample_pc_uart_recv_callback

概要	RSCI8 の受信完了時のコールバック関数です。
ヘッダ	sample_da16600_uart.h
宣言	void sample_pc_uart_read_buf (void);
説明	PC からの受信データを内蔵 Ring buffer にコピーします。
引数	なし
戻り値	なし
備考	なし

5.5.9 サンプルプログラムのフローチャート

main 関数のフローチャートを示します。
ベースとする「永久磁石同期モータのセンサレスベクトル制御 － MCK 用（R01AN6858）」に OTA program のメインルーチン（sample_ota_add_on）を追加しています。

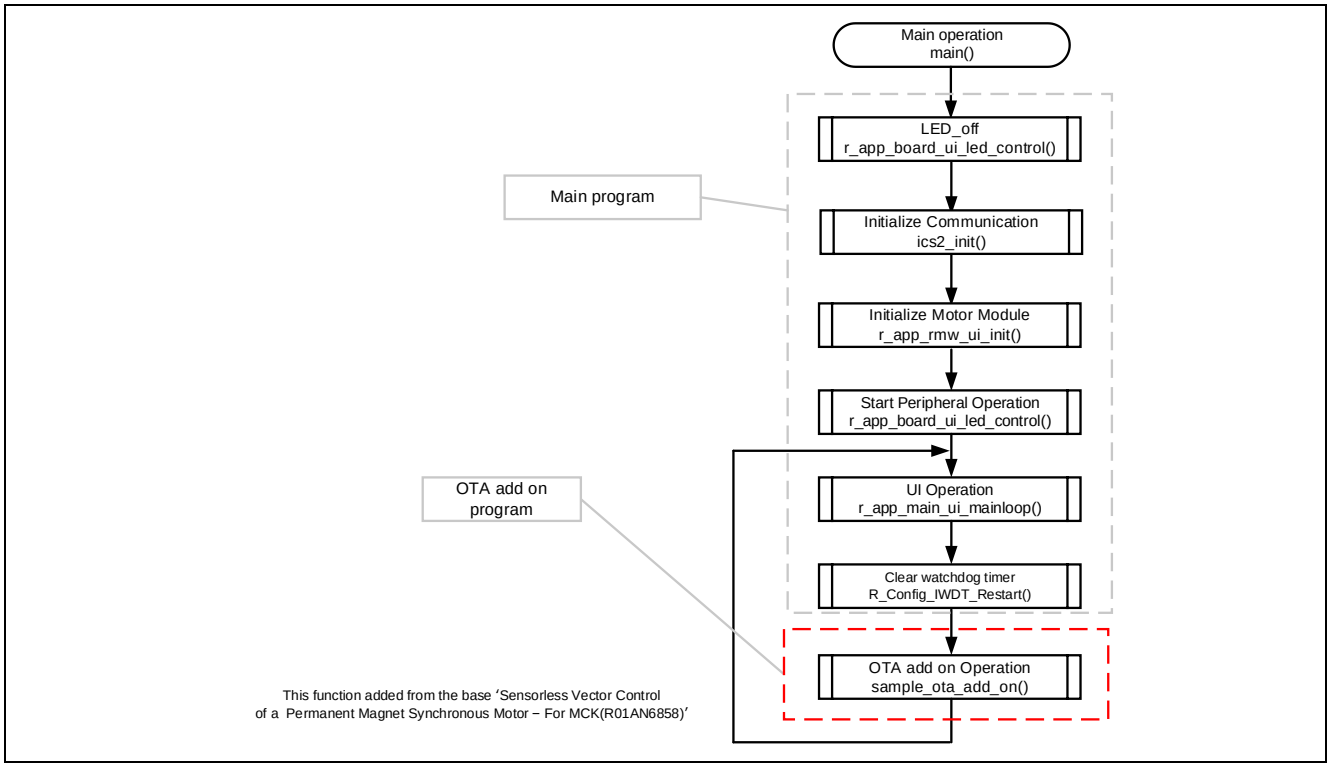

図 5-5　main 関数フローチャート

sample_ota_add_on 関数のフローチャートを示します。sample_ota_add_on 関数は OTA program のメインルーチンに相当します。sample_ota_add_on()からコールする 4 つの関数（ota_init、pc_comm、aws_comm、ota_ope）はすべて static 関数です。

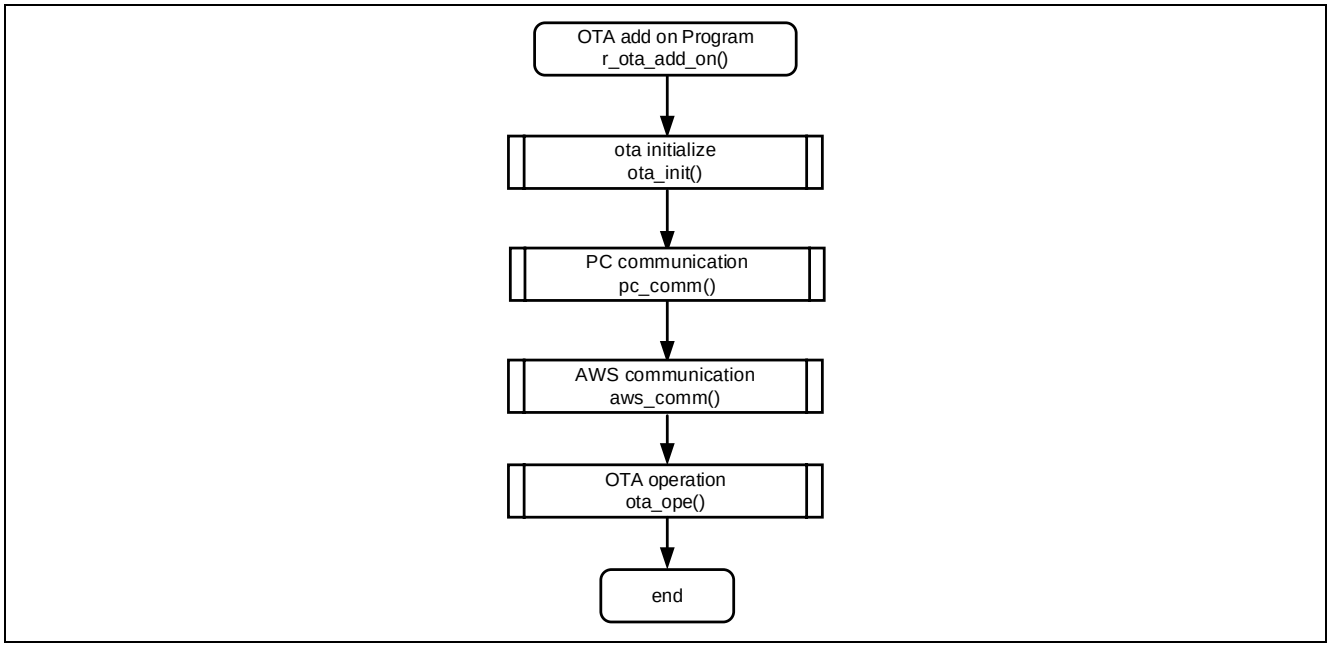

図 5-6　sample_ota_add_on 関数フローチャート

ota_init 関数のフローチャートを示します。ota_init 関数は AWS との MQTT 接続の確立および証明書の送信を行います。リセット直後（ota_st が 0 のとき）は send_client_credential 関数をコールし証明書などを送信して ota_st を 1 にします。ota_st が 1 のときは mqtt_start 関数をコールし MQTT 接続を行います。接続が確立すると ota_st を 2 にします。

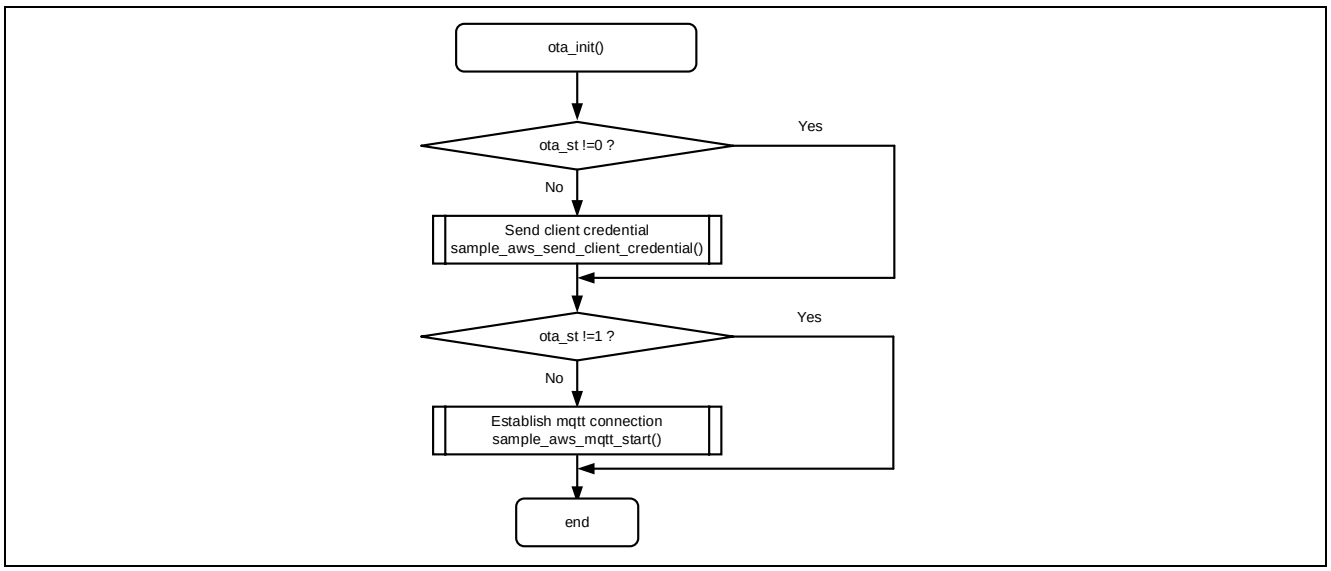

図 5-7　ota_init 関数フローチャート

send_client_credential 関数のフローチャートを示します。send_client_credential 関数はルート証明書、デバイス証明書および private_key を AWS に送信します。

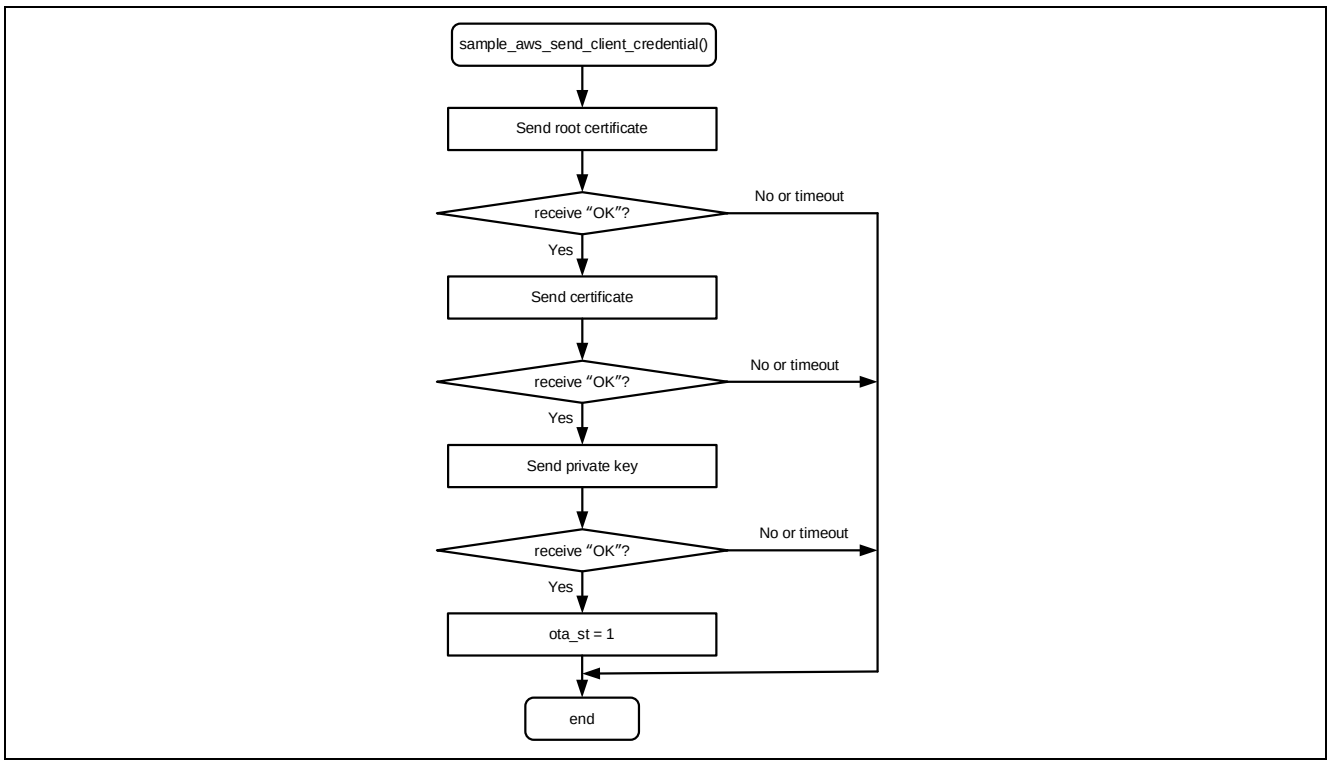

図 5-8　send_client_credential 関数フローチャート

mqtt_start 関数のフローチャートを示します。mqtt_start 関数は AWS との MQTT 接続を確立します。

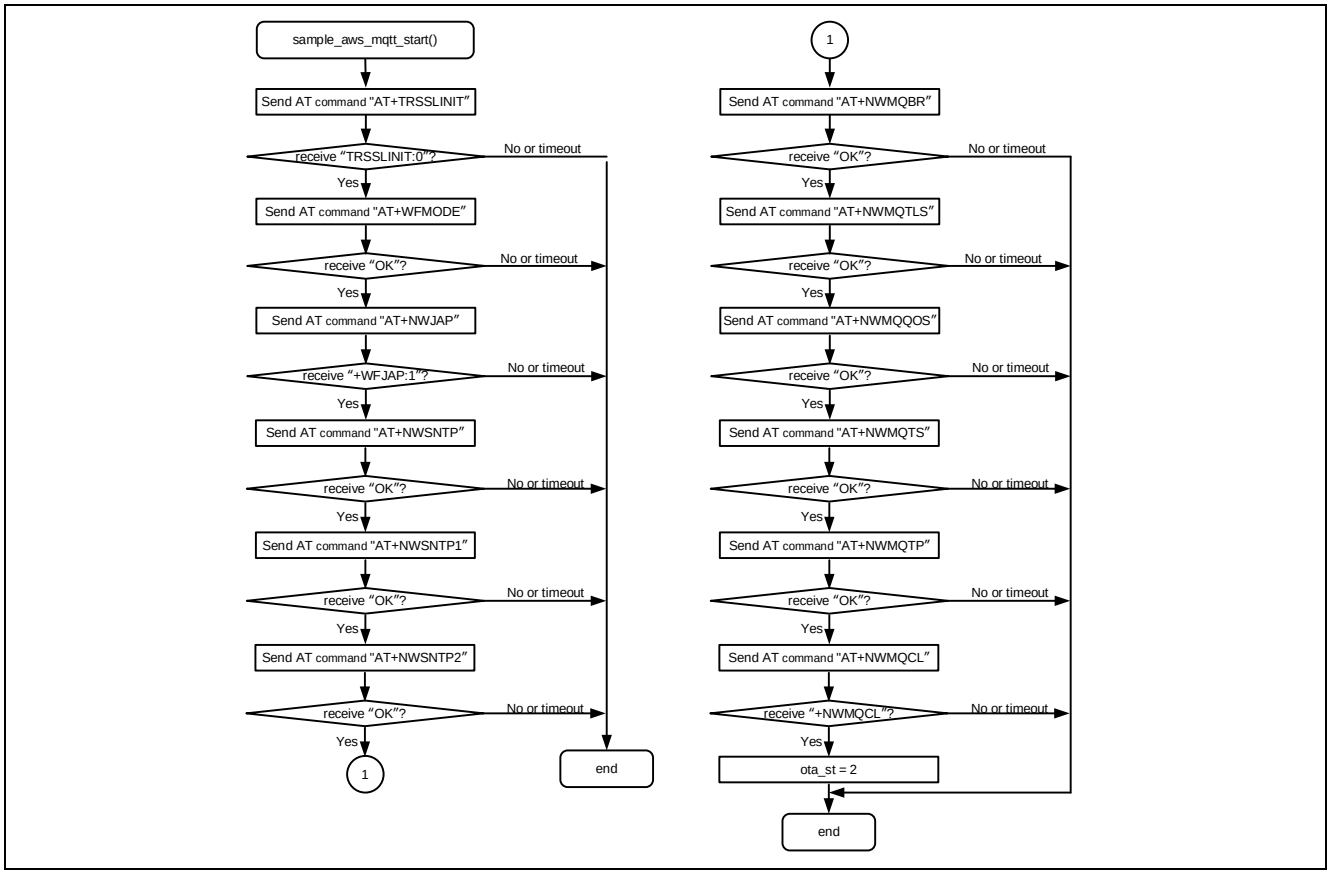

図 5-9　mqtt_start 関数フローチャート

pc_comm 関数フローチャートを示します。pc_comm 関数は pc からの OTA 開始指示やリセット開始指示を受け取ります。

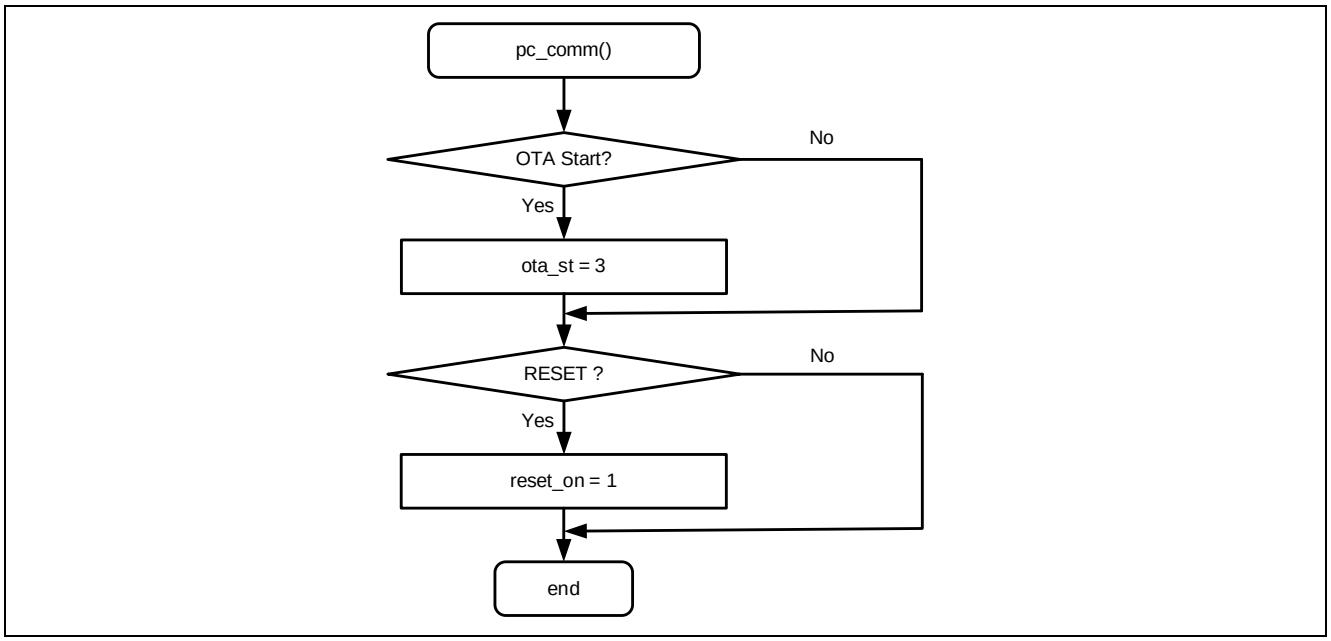

図 5-10　pc_comm 関数フローチャート

aws_comm 関数のフローチャートを示します。aws_comm 関数は更新イメージのダウンロードを行います。ota_st が 3 のときダウンロードする更新イメージのサイズを確認するために publish_size 関数をコールし ota_st を 4 にします。ota_st が 4 のとき subscribe_size 関数で更新イメージのサイズを取得します。サイズが確定したら ota_st を 5 にします。ota_st が 5 のときは publish_rsu 関数を使用して更新イメージのダウンロード要求を行い ota_st を 6 にします。ota_st が 6 のときは subscribe_rsu 関数をコールして更新イメージを取得します。更新イメージのダウンロードは 256 バイト刻みで行うためすべてのダウンロードが完了するまで ota_st=5 と ota_st=6 を繰り返し行います。

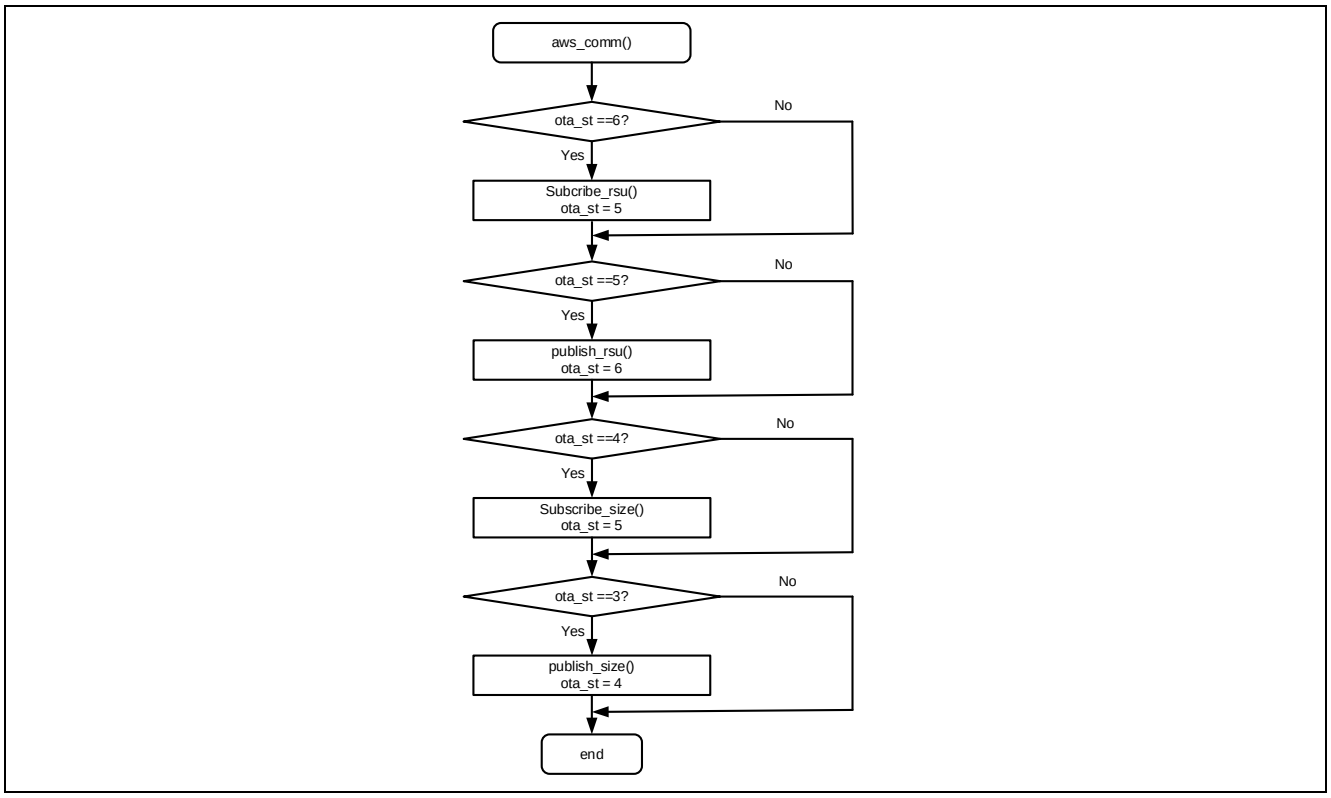

図 5-11　aws_comm 関数フローチャート

ota_ope 関数のフローチャートを示します。

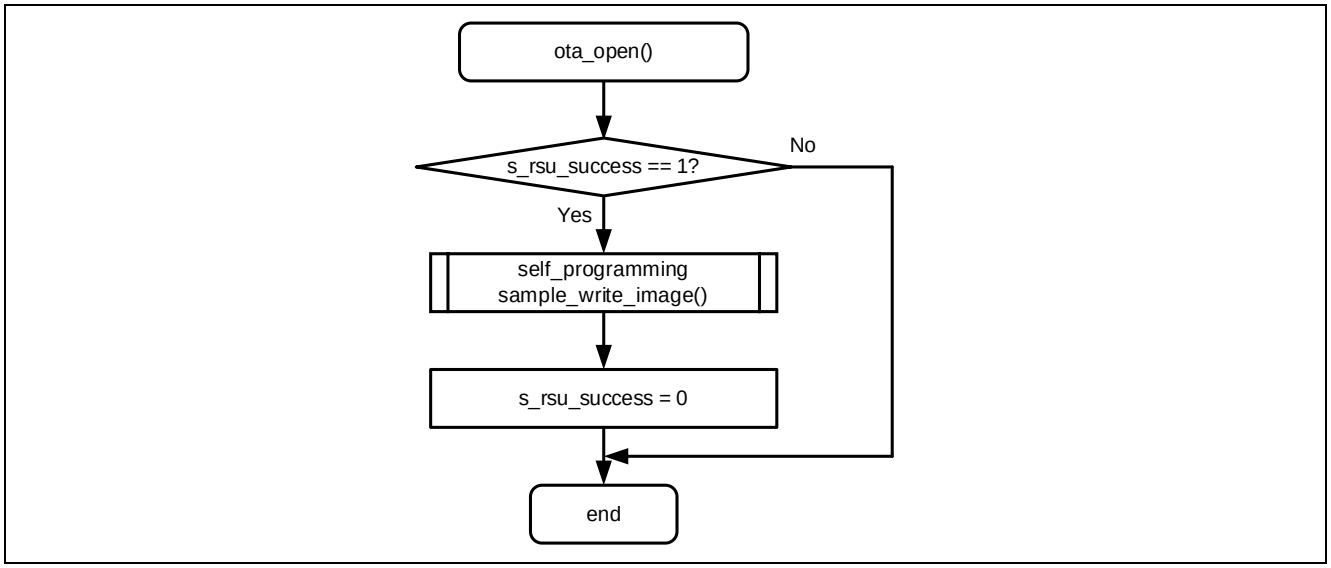

図 5-12　ota_ope 関数フローチャート

5.5.10 ブートローダ

本サンプルプログラムは「RX ファミリ ファームウェアアップデートモジュール Firmware Integration Technology（R01AN6850）」のデュアルバンク構成のブートローダを採用しています。ブートローダとしての機能は搭載しておらずメインプログラムを起動するのみとなっています。ブートローダとメインプログラムのアドレスについては、「5.5.4 周辺機能の設定」の r_flash_rx、r_fwup の設定値を参照してください。

5.6 ハードウェアの準備

「2.2 ボードの接続方法」のとおり MCK-RX26T、インバータボードおよびブラシレス DC モータを接続してください。さらに MCK-RX26T の PMOD Type 3A モジュール接続用コネクタ(CN12)に DA16600 Pmod™ Board を接続します。

5.6.1 Pmod-USBUART の接続方法

MCK-RX26T の PMOD Type 6A モジュール接続用コネクタ(CN10)と Pmod-USBUART を接続します。

以下のとおり Pmod-USBUART の RXD を CN10 の 9 番ピンに、TXD を 8 番ピンに、GND を 11 番ピンに接続します。CN10 のピン配列は下図赤枠内のシルクを参照してください。CN10 の 7 番ピンから 12 番ピンは CN10 の下段側です。

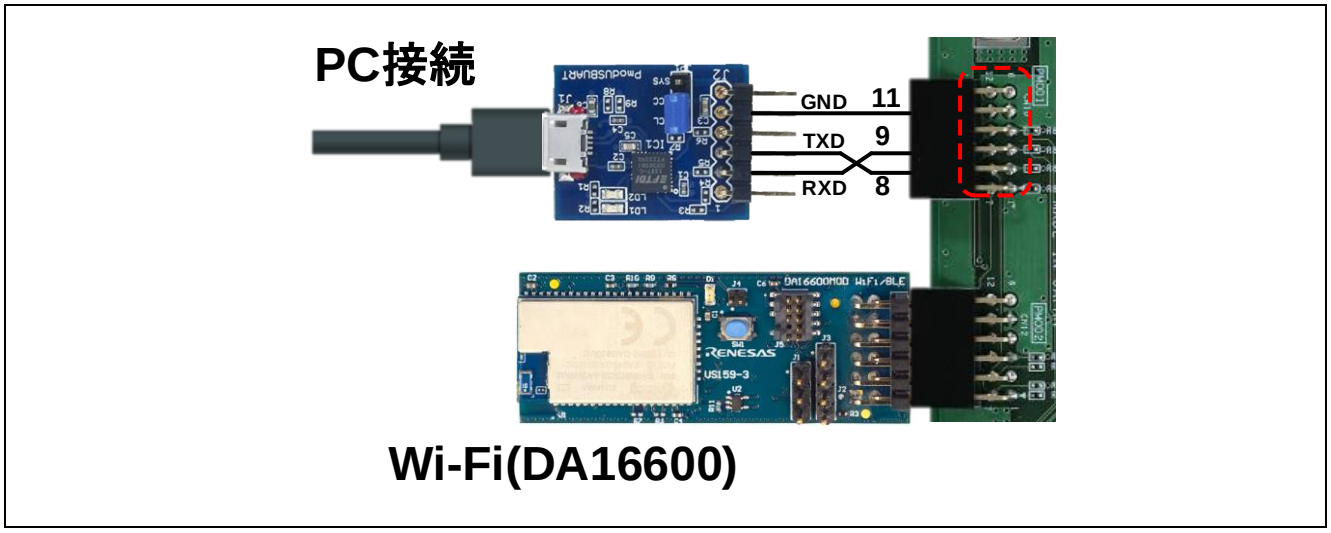

図 5-13 Pmod-USBUART の接続方法

5.7 ログ確認用 Terminal ソフトの準備

本サンプルプログラムのログは Pmod-USBUART 経由で出力されます。TeraTerm などのターミナルソフトを使用し、ログを確認することができます。本アプリケーションノートでは TeraTerm での設定方法を示します。

5.7.1 Tera Term を起動する

ログは Pmod-USBUART から出力されるため TeraTerm などのターミナルソフトを使用して確認します。Pmod-USBUART に USB を接続後 Tera Term を起動します。「Serial」にチェックを入れて Pmod-USBUART が接続されている COM ポートを選択してください。

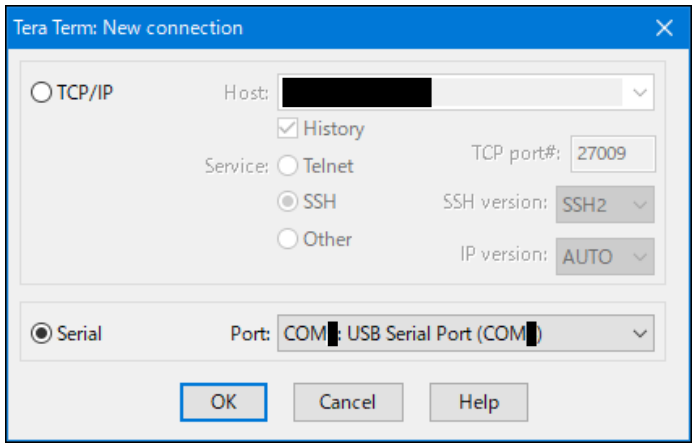

図 5-14 Tera Term の起動方法

5.7.2 Tera Term の Terminal を設定する

Tera Term のウインドウから「Setup」 ➡ 「Terminal setup」を開きます。以下のように設定してください。

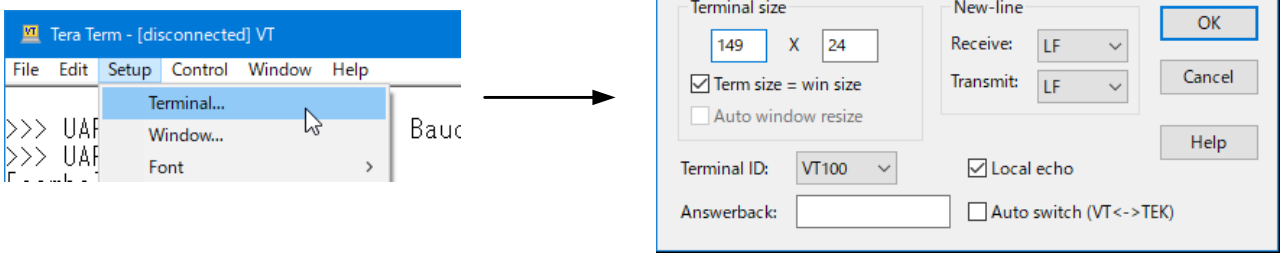

図 5-15 Tera Term の「Terminal setup」画面

5.7.3 Tera Term の Serial を設定する

Tera Term のウインドウから「Setup」 ➡ 「Serial port setup and connection」を開きます。ボーレートは Pmod-USBUART に合わせる必要があります。以下のように設定してください。

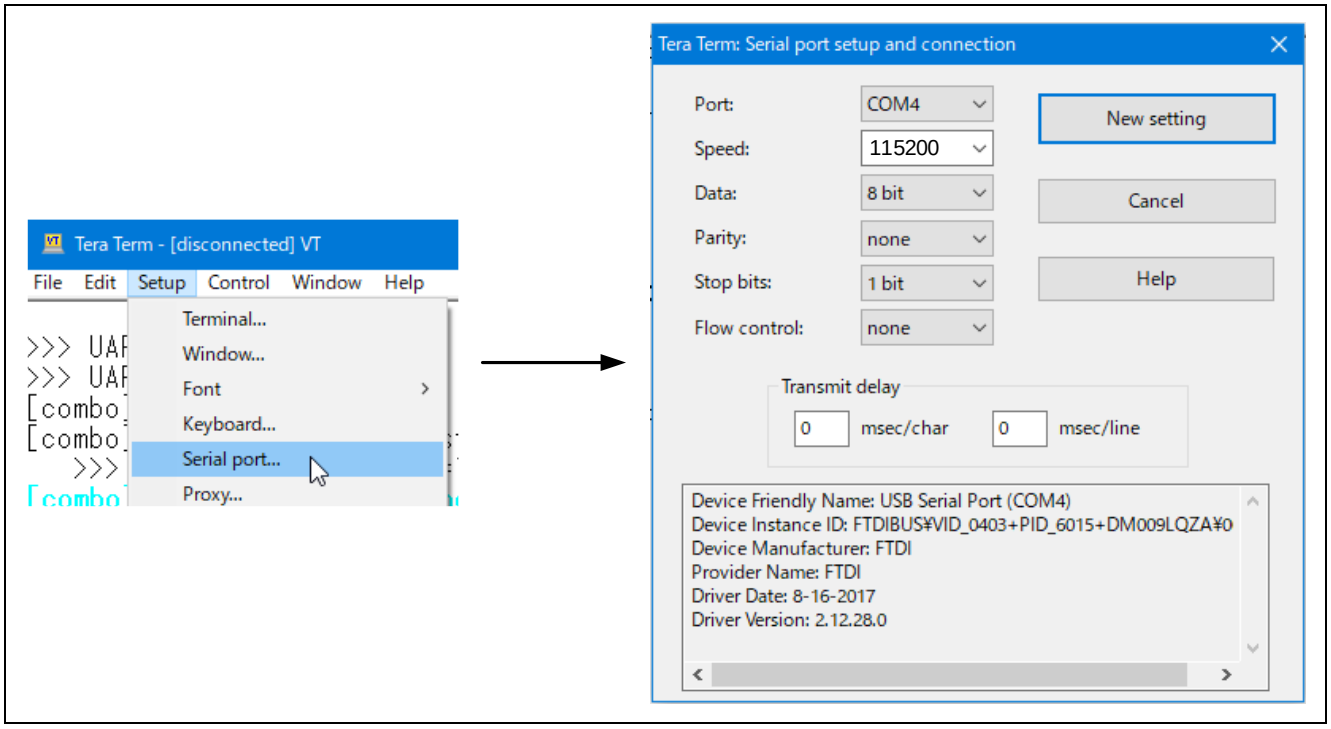

図 5-16 Tera Term の「Serial port setup and connection」画面

5.8 サンプルプログラムの動作概要

サンプルプログラムの動作概要を示します。サンプルプログラムを動作させるための詳細手順は、「5.9 サンプルプログラムの動作詳細」を参照してください。

(1) Start
Renesas Flash Programmer で初期イメージを書き込み、プログラムを実行します。
(2) Initialization
MCK-RX26T は自動的に初期化を行います。
(3) Storage of Information
MCK-RX26T は自動的に DA16600 Pmod™ Board を STA（Station）モードに設定します。また MCK-RX26T は自動的に DA16600 Pmod™ Board に証明書やプライベートキーを書き込みます。
(4) Wi-Fi Connection
MCK-RX26T は証明書などを書き込み後、自動的に DA16600 Pmod™ Board に Wi-Fi 接続要求を行います。
(5) AWS Connection
MCK-RX26T は Wi-Fi に接続されたことを確認後、DA16600 Pmod™ Board に AWS 接続要求を行います。
(6) MQTT Connection
MCK-RX26T は AWS に接続されたことを確認後、DA16600 Pmod™ Board に MQTT 接続要求を行います。

ここまでの処理で OTA の準備が整い PC には、「OTA Ready」の文字が表示されます。また MCK-RX26T は MQTT トピックを購読するために subscribe 要求を発行します。OTA を開始するには、PC 画面から「FWUP1」または「FWUP2」と入力します。

なお、これら一連の処理の間 Main program（「永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）」）は並行して動作を続けています。

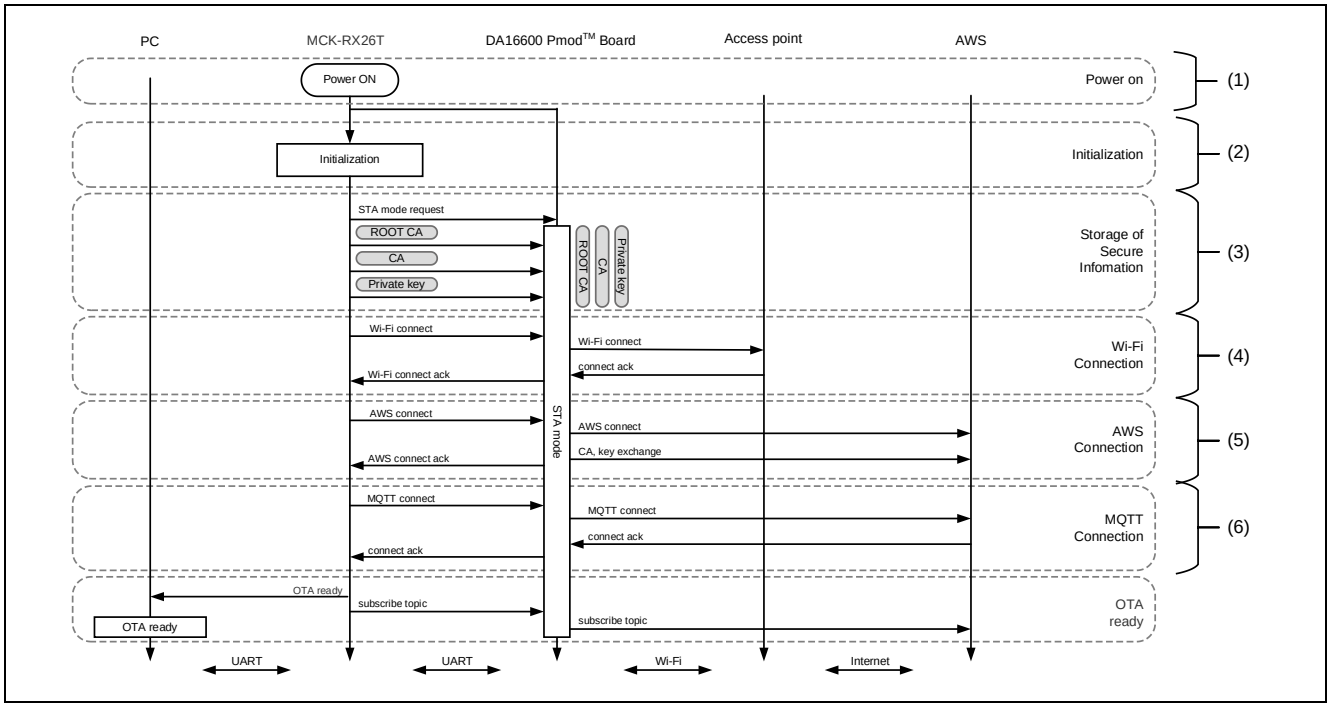

図 5-17 サンプルプログラムの動作概要

5.9 サンプルプログラムの動作詳細

以下の、サンプルプログラムの実行手順に従いデモを実行してください。
なお、DA16600 Pmod™ Board は一部の設定を NVRAM に保持します。そのため以前に別のデモを行っていた場合、前回のデモ動作が継続して実行される可能性があります。この場合、電源 OFF と電源 ON を数回繰り返すと DA16600 の NVRAM に本デモの条件が書き込まれ、正常に動作します。電源 ON を繰り返しても正常に動作しない場合は、DA16600 Pmod™ Board のファクトリーリセットを行ってください。

サンプルプログラムの実行手順

(1) Renesas Flash Programmer を使用し「3.2 初期イメージと更新イメージ」に記載の初期イメージを RX26T に書き込みます。プログラムを書き込む際には CPU ボードの JP11 を開放する必要がありますので注意してください。書き込み完了後 JP11 を閉じて USB から電源を供給すると LED5 が点灯します。

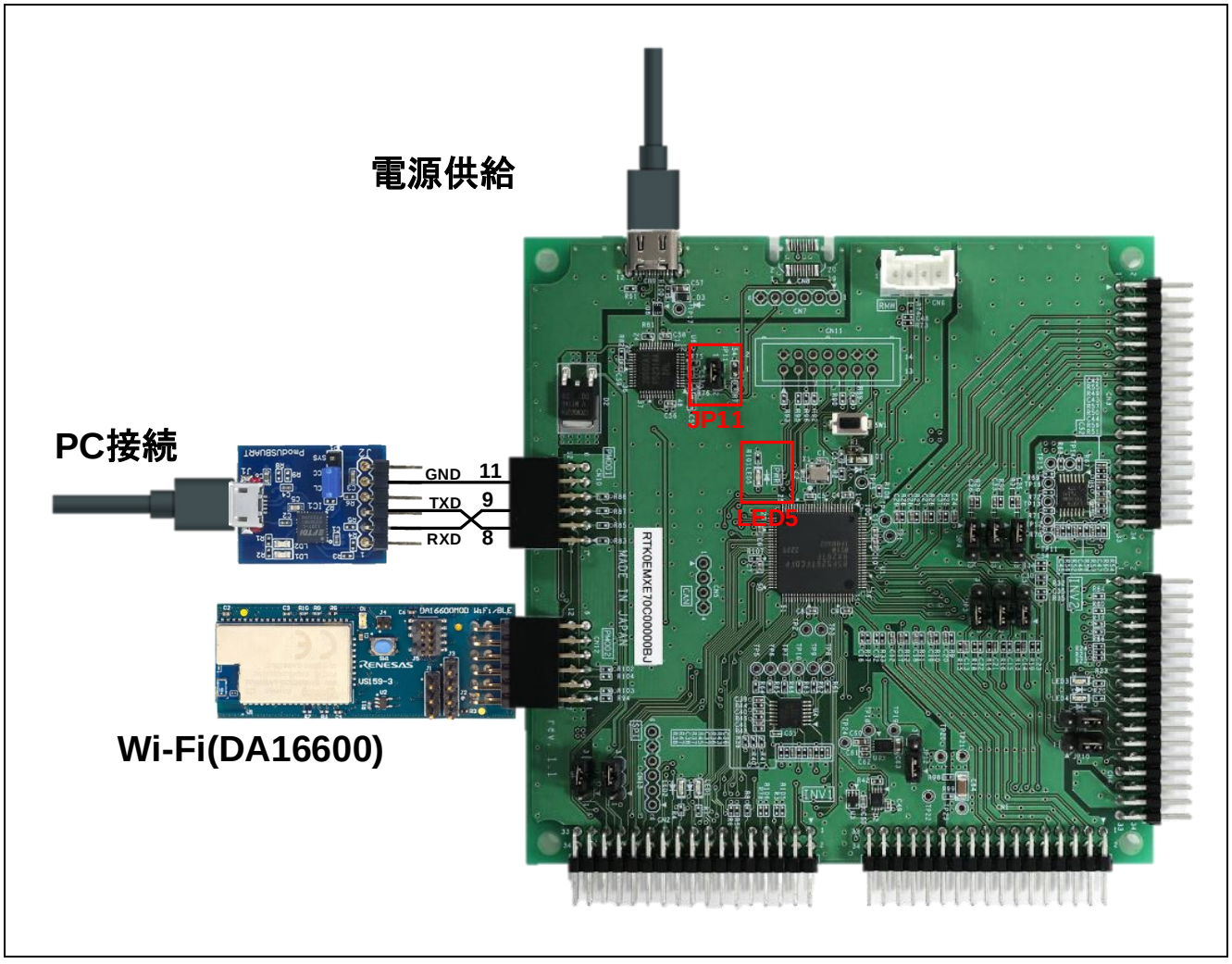

図 5-18 LED5（Power）の位置

(2) Initialization
MCK-RX26T は自動的に初期化を行います。
この後（3）から（6）までのすべての動作は MCK-RX26T が自動で行い、（7）で OTA の準備が整います。

(3) Storage of Information
MCK-RX26T は自動的に DA16600 Pmod™ Board を STA モードに設定します。
その後 MCK-RX26T は自動的に証明書と秘密鍵を DA16600 Pmod™ Board に書き込みます。

```
START 1Shunt!!↓
AT↓
resp:OK¥r¥n↓
C0,-----BEGIN CERTI↓
resp:OK¥r¥n↓
C1,-----BEGIN CERTI↓
resp:OK¥r¥n↓
C2,-----BEGIN RSA P↓
resp:OK¥r¥n↓
```

図 5-19　証明書と秘密鍵のログ

(4) Wi-Fi Connection
MCK-RX26T は自動的に DA16600 Pmod™ Board に Wi-Fi 接続要求を行います。

```
AT+WFJAP=[redacted],4,2,[redacted]
resp:OK¥r¥n↓
resp:+WFJAP:1,'[redacted]',192.168.1.56¥r¥n↓
```

図 5-20　Wi-Fi 要求のログ

(5) MCK-RX26T は自動的に DA16600 Pmod™ Board に AWS 接続要求を行います。
(6) MCK-RX26T は自動的に DA16600 Pmod™ Board に MQTT 接続要求を行います。
(7) OTA ready
AWS との接続が完了し OTA の準備が整うと PC 画面に OTA ready と表示されます。

```
AT+NWMQBR=[redacted].iot.ap-northeast-1.amazonaws.com,8883↓   (5)
resp:OK¥r¥n↓
AT+NWMQTLS=1↓
resp:OK¥r¥n↓
AT+NWMQQOS=0↓
resp:OK¥r¥n↓
AT+NWMQTS=1,RX26T_Topic/FROM_AWS↓                             (6)
resp:OK¥r¥n↓
AT+NWMQTP=RX26T_Topic/FROM_EDGES↓
resp:OK¥r¥n↓
AT+NWMQCL=1↓
resp:+NWMQCL:1¥r¥n↓
OTA ready↓                                                    (7)
```

図 5-21　AWS と MQTT のログ

ここまでの処理で OTA の準備が整い「OTA Ready」の文字が表示されます。（５）から（８）が動作しない場合「7.1.3Fail to establish tls-sess(0x7200)が表示される場合」に該当していないかご確認ください。

(8) Publish size
PC 画面から OTA 開始命令を入力する（FWUP1 または FWUP2 と入力しキャリッジリターンを入力する）と MCK-RX26T は AWS に対し要求に応じた更新イメージ（.rsu）のサイズを問い合わせるために、sample_aws_publish_size 関数をコールし AWS に publish します。

(9) Subscribe size
AWS 内部（Lambda）で要求が受理され更新イメージのサイズが publish されるとすでに購読済みのトピックスを通じてサイズが返信されます（ログには出力しません）。

(10) Publish rsu
MCK-RX26T は受信したサイズをもとに AWS に対し更新イメージのダウンロード要求を行うため sample_aws_publish_rsu 関数をコールし AWS に publish します。

(11) Subscribe rsu
AWS 内部（Lambda）で要求が受理され更新イメージが publish されるとすでに購読済みのトピックスを通じて更新イメージが返信されます（ログには出力しません）。なお、更新イメージは 256Byte 単位でダウンロードするため(10)と(11)は、複数回実行されます。
またダウンロードした更新イメージは MCU（RX26T）の Flash ROM のバッファ面に書き込みます。

```
AT+NWMQMSG='{"mess":"v1s" , "seek": 0 , "size":  0}'↓                    (8)
resp:¥r¥n↓
resp:+NWMQMSGSND:1¥r¥n↓
download size:     0/     0↓
AT+NWMQMSG='{"mess":"v1d" , "seek":0 , "size": 256}'↓                    (10)
resp:¥r¥n↓
resp:+NWMQMSGSND:1¥r¥n↓                                                  (11)
download size:   256/ 69120↓
W 0xFFF80000, 256 ... OK↓
AT+NWMQMSG='{"mess":"v1d" , "seek":256 , "size": 256}'↓                  (10)
resp:¥r¥n↓
resp:+NWMQMSGSND:1¥r¥n↓                                                  (11)
download size:   512/ 69120↓
W 0xFFF80100, 256 ... OK↓
AT+NWMQMSG='{"mess":"v1d" , "seek":512 , "size": 256}'↓                  (10)
resp:¥r¥n↓
resp:+NWMQMSGSND:1¥r¥n↓                                                  (11)
download size:   768/ 69120↓
W 0xFFF80200, 256 ... OK↓
AT+NWMQMSG='{"mess":"v1d" , "seek":768 , "size": 256}'↓                  (10)
resp:¥r¥n↓
resp:+NWMQMSGSND:1¥r¥n↓                                                  (11)
download size:  1024/ 69120↓
```

図 5-22　更新イメージのダウンロードログ

(12) 更新イメージのすべてのデータのダウンロードが正常に完了しファームウェアの書き込みが完了すると画面表示が停止し MCU（RX26T）は PC からのリセット命令を待ちます。
※本来のファームウェアアップデート動作としては次に MCU が再起動したときにブートローダがバッファ面に有効なプログラムがあることを検証しバンクスワップと自己リセットおよび旧プログラムの消去を行い更新イメージを起動します。本デモプログラムでは PC からの命令によりバンクスワップと自己リセットを行います。
この時点で MCK-RX26T は更新イメージを起動する準備を整えていますが 1Shunt/2Shunt でインバータボードの JP8/JP11 の切り替えを行う必要があります。モータを停止させた状態で、JP8/JP11 を更新イメージ（1Shunt/2Shunt）に合わせてください。
PC 画面から「RESET」と入力すると MCK-RX26T はバンクスワップを実行した後ソフトウェア自己リセットを行います。これにより、更新イメージが起動します。

```
AT+NWMQMSG='{"mess":"v1d" , "seek":68608 , "size": 256}'↓
resp:¥r¥n↓
resp:+NWMQMSGSND:1¥r¥n↓
download size: 68864/ 69120↓
W 0xFFF90C00, 256 ... OK↓
AT+NWMQMSG='{"mess":"v1d" , "seek":68864 , "size": 256}'↓
resp:¥r¥n↓
resp:+NWMQMSGSND:1¥r¥n↓
download size: 69120/ 69120↓
W 0xFW 0xFFFB7F80, 128 ... OK↓
```

(12)

図 5-23　正常終了時のログ

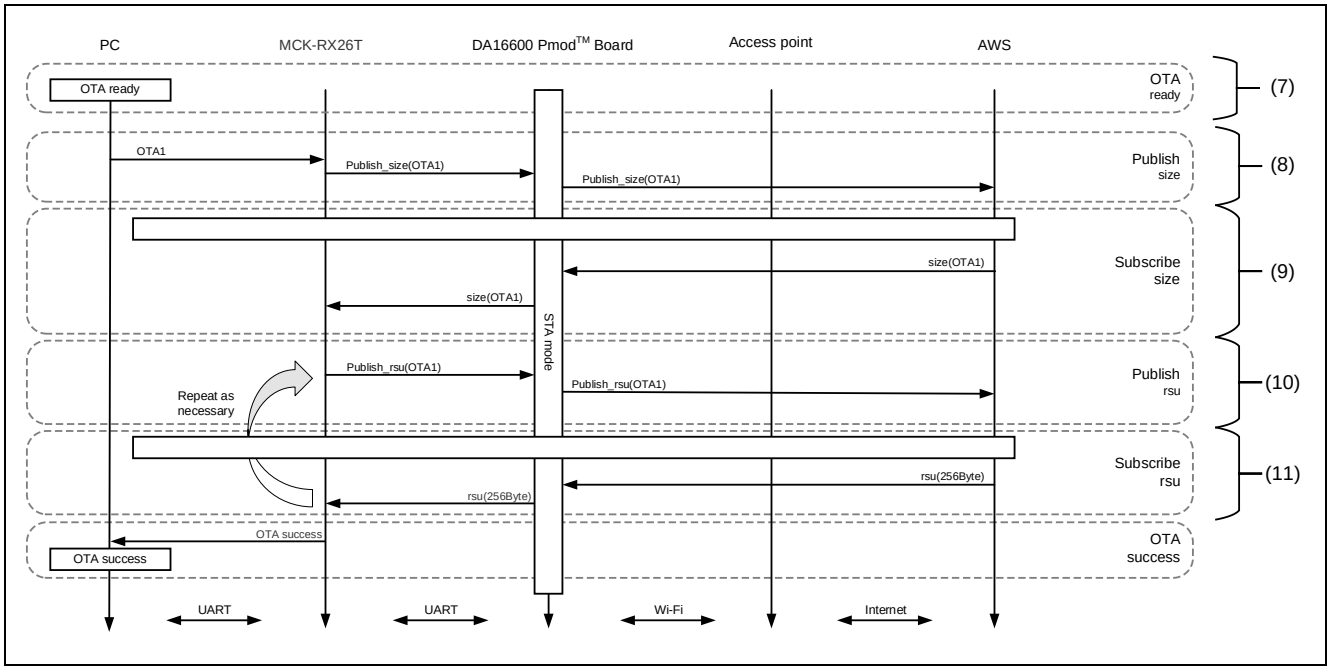

図 5-24　OTA のデータの流れ

6. プロジェクトのビルド手順

サンプルプログラムは e² studio のプロジェクト形式で提供しています。本章では、e² studio および CS+ へプロジェクトをインポートする方法を示します。インポート完了後、ビルドの設定を確認してください。この手順は、スマート・コンフィグレータ（V2.19.0）、r_fwup（v2.01）および r_flash_rx（v5.10）で動作を確認しています。

6.1 e² studio での手順

6.1.1 e² studio でのインポート方法

e² studio で使用する際は以下の手順で e² studio にインポートしてください。

なお、e² studio で管理するプロジェクトのフォルダ名、およびそのフォルダに至るファイルパスには 空白文字の他、半角カナ文字、全角文字、半角記号（特に'$','#','%'）が混じらないようにしてください（使用する e² studio のバージョンによっては画面が異なる場合があります）。

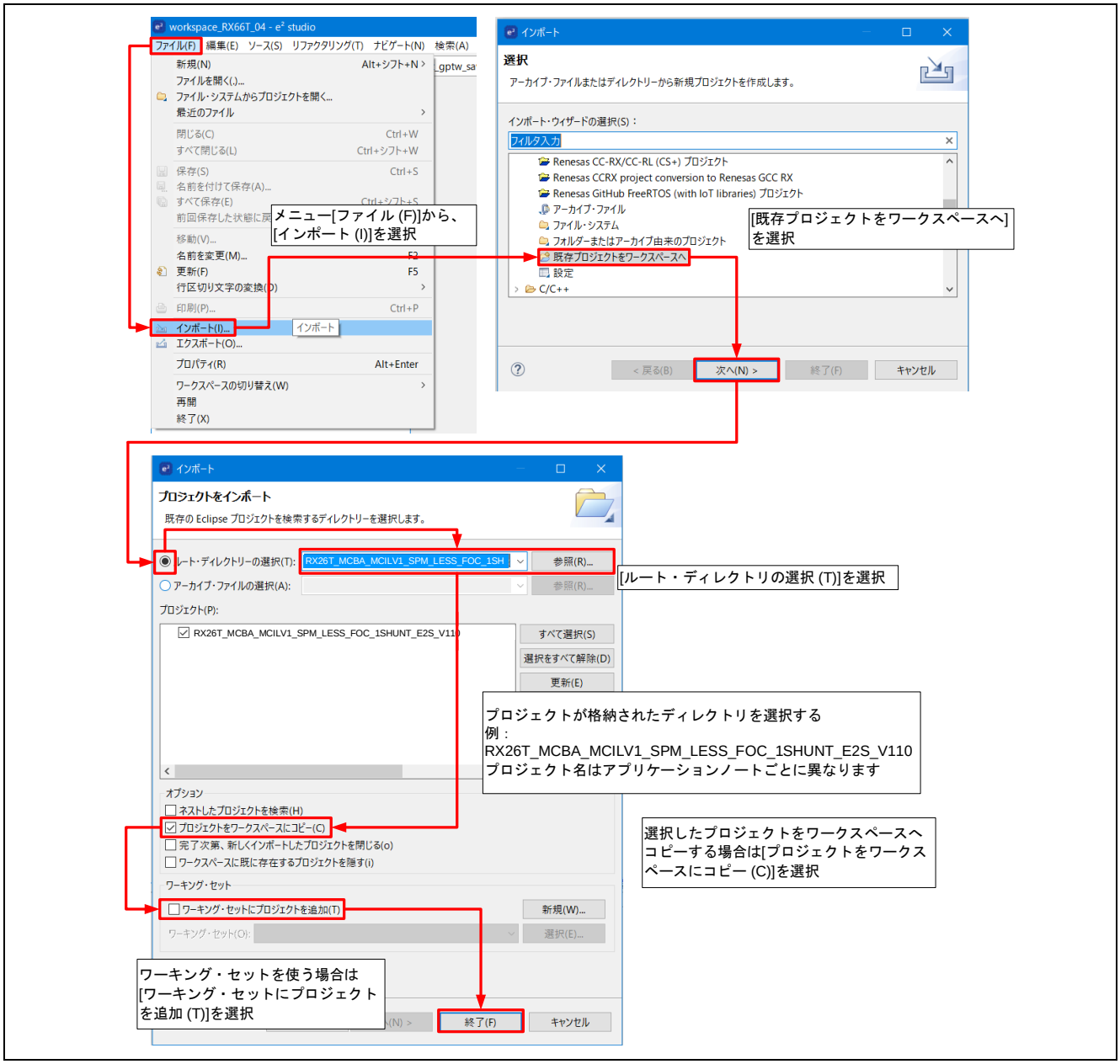

図 6-1 プロジェクトを e² studio にインポートする方法

6.1.2 FIT の確認

e2studio でビルドを行う前に、FIT がダウンロードされていることを確認してください。FIT がダウンロードされていない場合、コード生成の際に必要なコードが生成されずエラーとなる可能性があります。

「6.1.2 FIT の確認」に従い「r_flash_rx」と「r_fwup」のバージョンを確認します。

本例では、「r_flash_rx」の FIT はダウンロード済みですが、「r_fwup」の FIT はダウンロードされていません。FIT がダウンロードされていない場合、本例のようにコンポーネントがグレー表示され、コンポーネントがない旨のメッセージが表示されます。「このコンポーネントがありません」と表示された場合、「コンポーネントをダウンロード」をクリックし、FIT をダウンロードしてください。

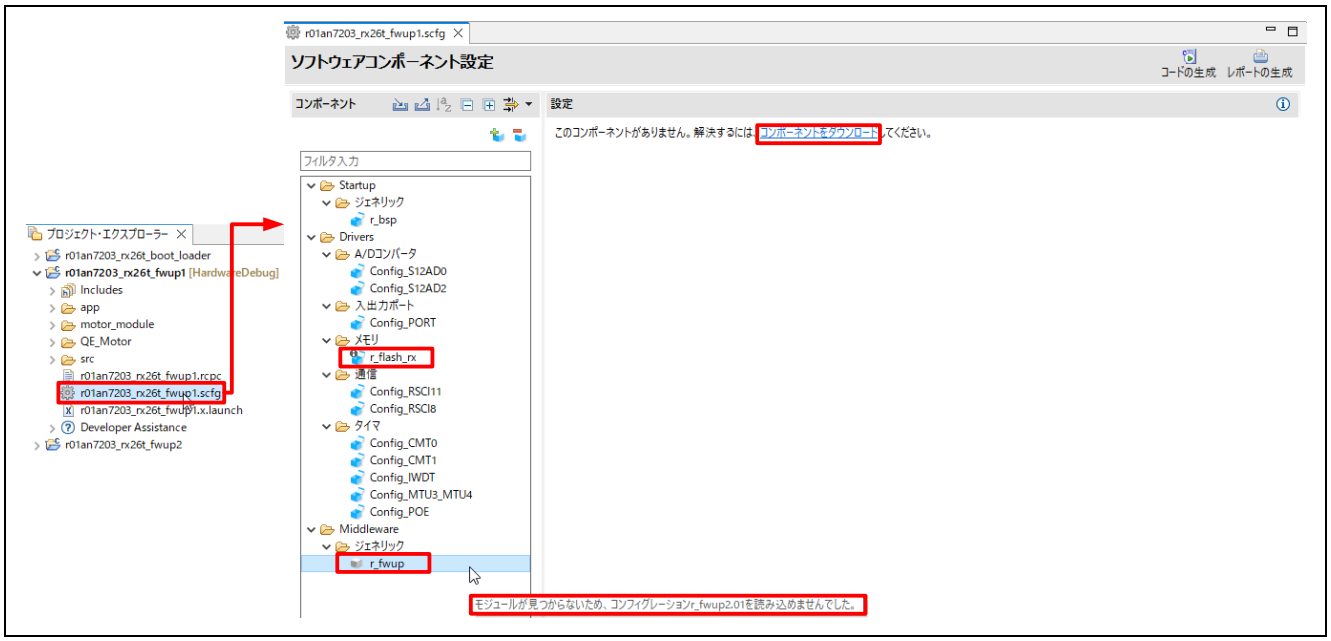

図 6-2 FIT のバージョン確認方法

6.1.3 ビルドオプションの設定

1. プロジェクト名を右クリックし、メニュー表示➡「プロパティ」を選択します。
2. 「C/C++ビルド」➡「設定」の「Toolchain」タブをクリックし、ツールチェーンとバージョンを確認してください。
   ・ツールチェーン：Renesas CC-RX
   ・バージョン　　：v3.05.00

6.1.4 プロジェクトのビルド

1. プロジェクト・エクスプローラでプロジェクトを右クリックし、「プロジェクトのビルド」を選択します。
2. ビルドが開始され「コンソール」にビルドの状況が表示されます。”Build Finished”というメッセージが表示されたらビルド完了です。HardwareDebug フォルダにプロジェクト名.mot ファイルが出力されます。

6.2 CS+ での手順

CS+ で使用する際は以下の手順で CS+ にインポートしてください。

なお CS+で管理するプロジェクトのフォルダ名およびそのフォルダに至るファイルパスには 空白文字の他、半角カナ文字、全角文字、半角記号（特に'$','#','%'）が混じらないようにしてください（使用する CS+ のバージョンによっては画面が異なる場合があります）。

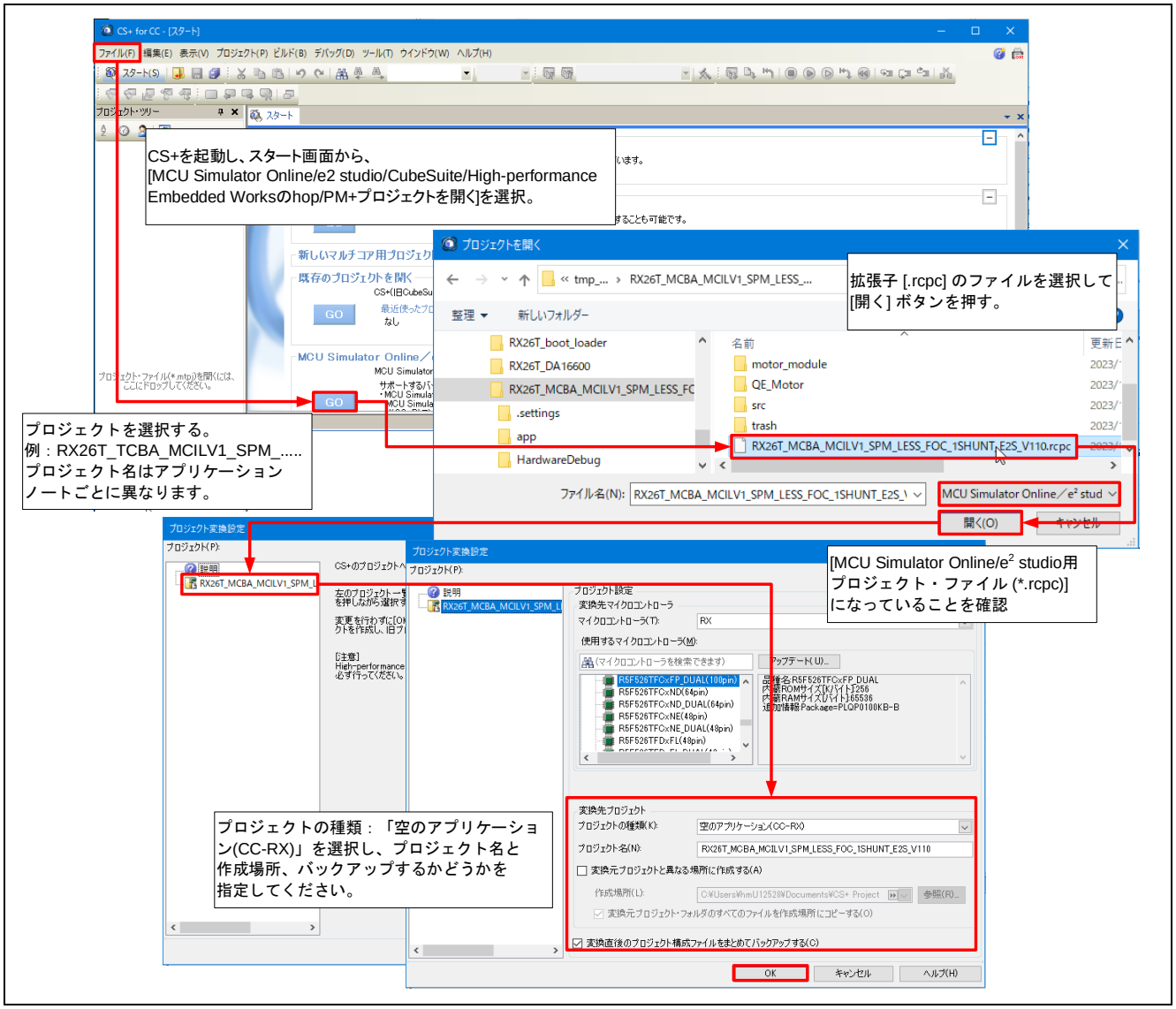

図 6-3 プロジェクトを CS+ にインポートする方法

6.2.1 プロジェクトのビルド

1. 「ビルド」➡「ビルド・プロジェクト」を選択します。
2. ビルドが開始され「コンソール」にビルドの状況が表示されます。”ビルド終了”というメッセージが表示されたらビルド完了です。HardwareDebug フォルダにプロジェクト名.mot ファイルが出力されます。

7. トラブルシューティング

7.1 AWS に接続できない

AWS に接続できない原因は様々です。主な原因は、「AWS のエンドポイントが間違っている」や「証明書やプライベートキーが間違っている」などが挙げられます。これらを正確に確認するためには、DA16600 Pmod™ Board のデバッグポート経由でログを確認します。ここではログの確認方法およびいくつかのトラブルシュートの方法を示します。

7.1.1 DA16600 Pmod™ Board のログを確認する方法

7.1.1.1 必要部品

DA16600 Pmod™ Board のログを確認するためには以下のモジュールが必要です。

- Pmod-USBUART（DIGILENT 製）
  https://digilent.com/reference/pmod/pmodusbuart/start?redirect=1

7.1.1.2 接続方法

Pmod-USBUART と DA16600 Pmod™ Board を以下のように接続してください。MCK-RX26T の接続はそのままにしてください。ただし Pmod-USBUART の接続を行う際は MCK-RX26T の電源は切断してください。

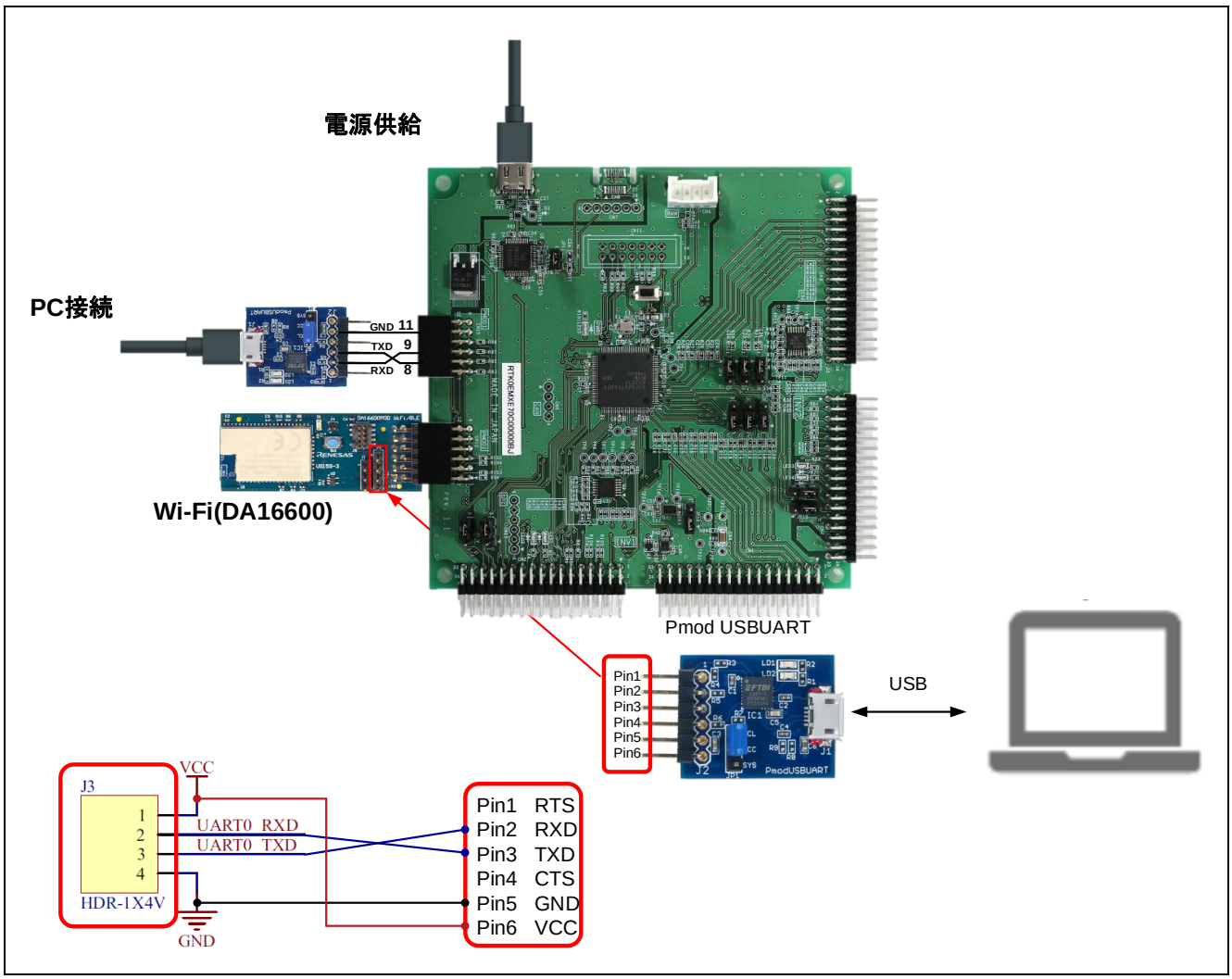

図 7-1 Pmod-USBUART と DA16600 Pmod™ Board の接続方法

7.1.1.3 Tera Term を起動する

MCK-RX26T の電源を投入後 Tera Term を起動します。「Serial」にチェックを入れて Pmod-USBUART が接続されている COM ポートを選択してください。

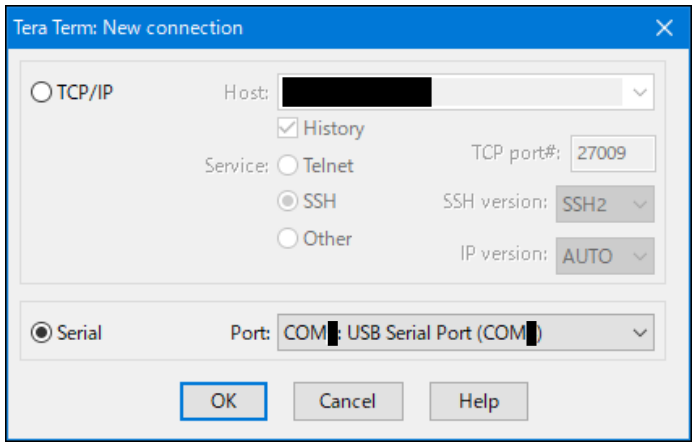

図 7-2 Tera Term の起動方法

7.1.1.4 Tera Term の Terminal を設定する

Tera Term のウインドウから「Setup」 ➡ 「Terminal setup」を開きます。以下のように設定してください。

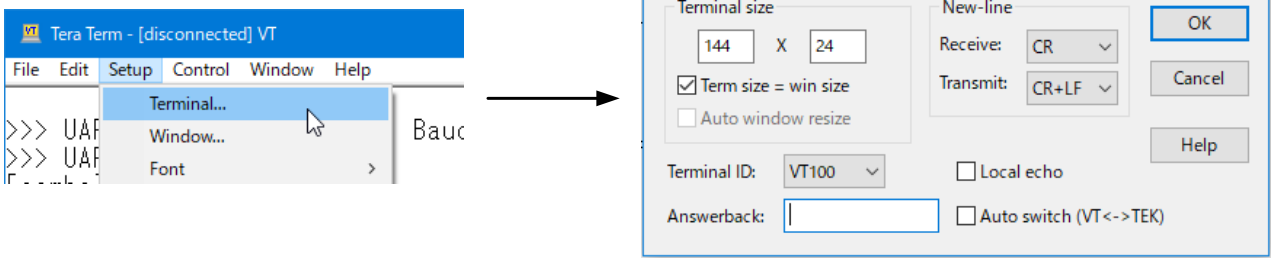

図 7-3 Tera Term の「Terminal setup」画面

7.1.1.5 Tera Term の Serial を設定する

Tera Term のウインドウから「Setup」 ➡ 「Serial port setup and connection」を開きます。ボーレートは DA16600 Pmod™ Board のデバッグポートの設定に合わせる必要があります。以下のように設定してください。

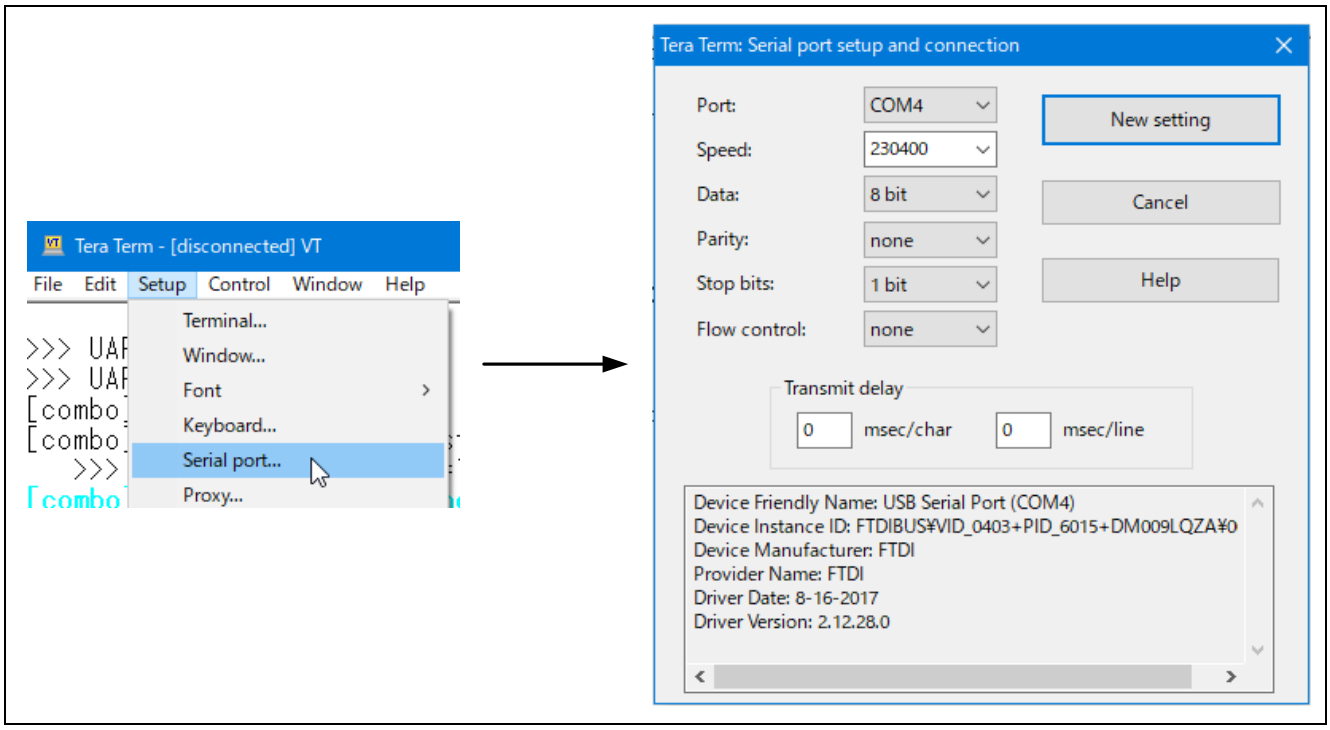

図 7-4 Tera Term の「Serial port setup and connection」画面

7.1.1.6 動作を確認する

MCK-RX26T の電源を入れ正しくログが表示されることを確認してください。代表的なログを以下に示します。

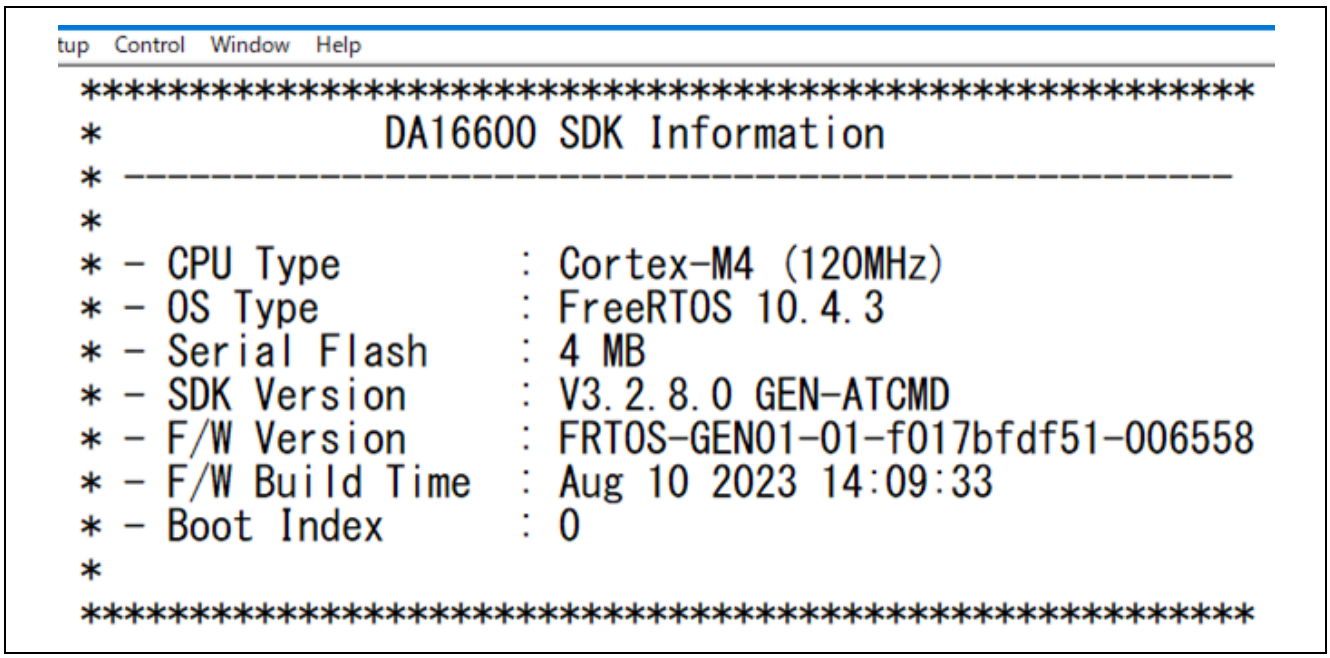

図 7-5 ログの例

7.1.2 DA16600 Pmod™ Board のファームウェアバージョンを更新する

本アプリケーションノートは DA16600_IMG_FreeRTOS_ATCMD_UART2_EVK_v3.2.8.0 で動作を確認しています。ファームウェアのバージョンが異なる場合、以下に従って
ファームウェアを更新してください。本 7.1.2 では v3.2.8.0 を例に動作を示します。

7.1.2.1 ファームウェアをダウンロードする

以下に接続し DA16200 DA16600 FreeRTOS SDK Image（例：v3.2.8.0）をダウンロードします。
https://www.renesas.com/jp/ja/products/interface-connectivity/wireless-communications/wi-fi/low-power-wi-fi/da16600mod-ultra-low-power-wi-fi-bluetooth-low-energy-combo-modules-battery-powered-iot-devices#document

図 7-6 ファームウェアのダウンロード

上記ファームウェアは、v3.2.8.0 以降のバージョンであれば動作します。
バージョン依存で何らかの問題が発生し、v3.2.8.0 の入手を希望される場合は弊社営業および代理店にお問い合わせください。

7.1.2.2 ダウンロードしたファイルを解凍する

ダウンロードした zip ファイルを任意のフォルダに解凍してください。解凍後さらに DA16600_IMG_FreeRTOS_ATCMD_UART2_EVK_v3.2.8.0_4MB.zip を解凍してください。

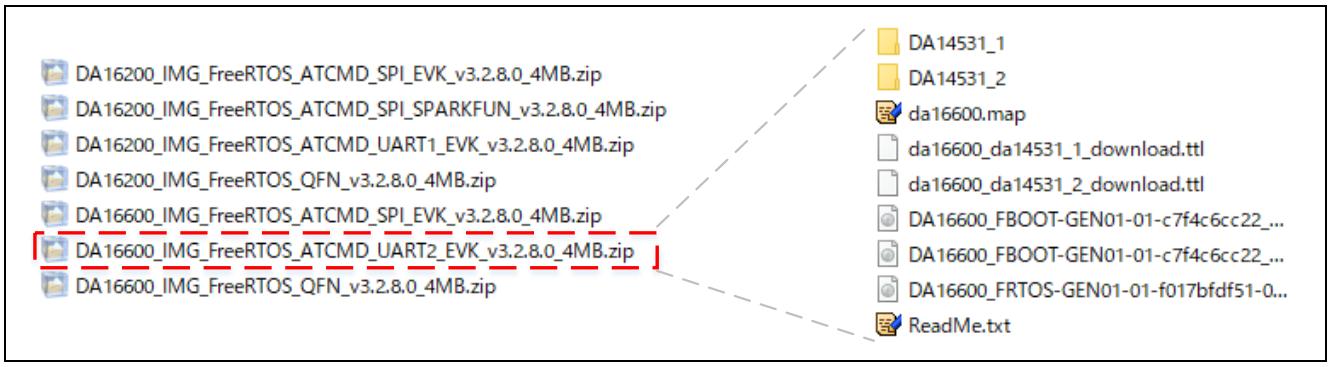

図 7-7 ファームウェアの解凍

7.1.2.3 Pmod-USBUART を接続する

Pmod-USBUART と DA16600 Pmod™ Board を以下のとおり接続してください。

DA16600 Pmod™ Board の Pmod 端子は、すべてオープンにしてください。Pmod-USBUART と PC の接続は最後に行ってください。

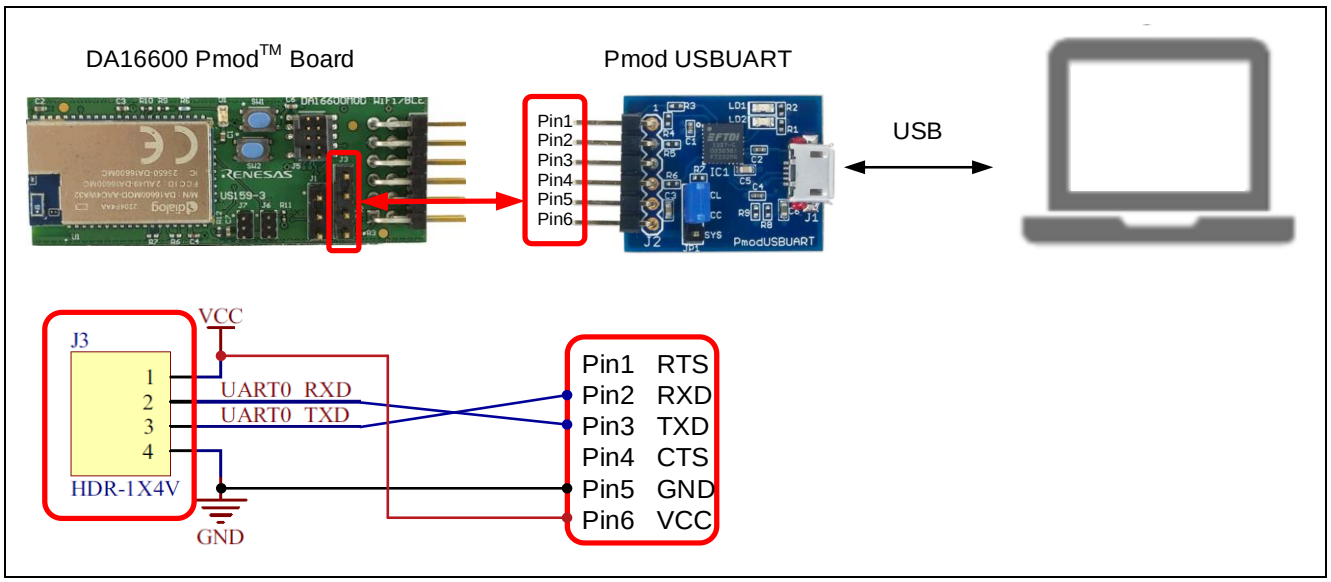

図 7-8 Pmod-USBUART と DA16600 Pmod™ Board の接続方法

7.1.2.4 電源を供給する

Pmod-USBUART と PC を USB で接続すると Pmod-USBUART と DA16600 Pmod™ Board に電源が供給されます。

7.1.2.5 Tera Term を起動する

MCK-RX26T の電源を投入後 Tera Term を起動します。「Serial」にチェックを入れて Pmod-USBUART が接続されている COM ポートを選択してください。

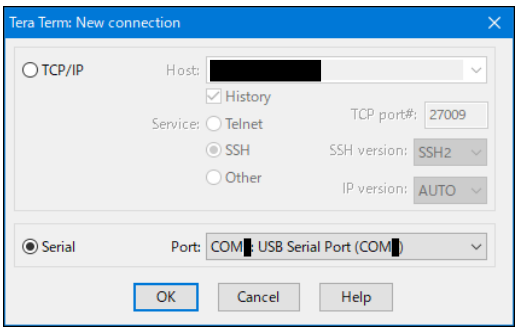

図 7-9 Tera Term の起動方法

7.1.2.6 Tera Term の Terminal を設定する

Tera Term のウインドウから、「Setup」 ➡ 「Terminal setup」を開きます。以下のように設定してください。

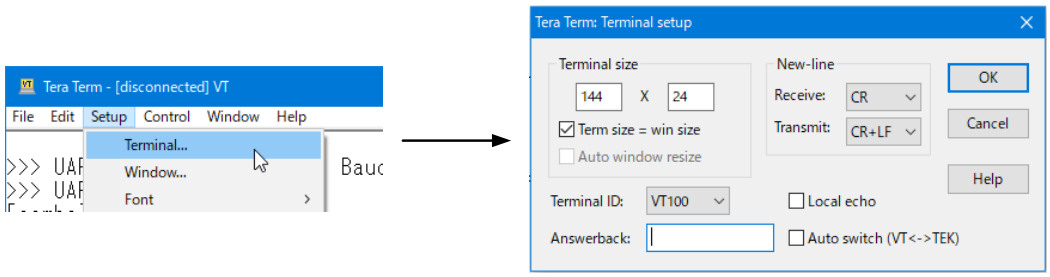

図 7-10 Tera Term の「Terminal setup」画面

7.1.2.7 Tera Term の Serial を設定する

Tera Term のウインドウから「Setup」 ➡ 「Serial port setup and connection」を開きます。ボーレートは DA16600 Pmod™ Board のデバッグポートの設定に合わせる必要があります。以下のように設定してください。

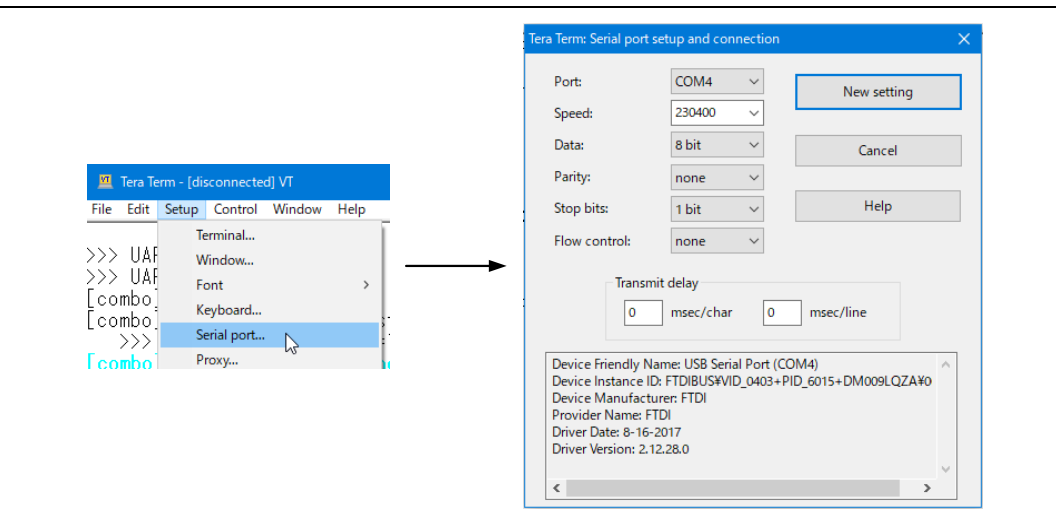

図 7-11 Tera Term の Serial port setup and connection 画面

7.1.2.8 ファームウェアを更新する

Tera Term のウインドウから「Control」 ➡ 「Macro」を開きます。「7.1.2.2 ダウンロードしたファイルを解凍する」で解凍したフォルダの「da16600_da14531_1_download.ttl」を選択し「開く」をクリックします。続いて、「Confirm」画面で、「AT25SL321」を選択し「OK」をクリックすると Tera Term 上で自動的にコマンドが実行されファームウェアのアップデートが行われます。

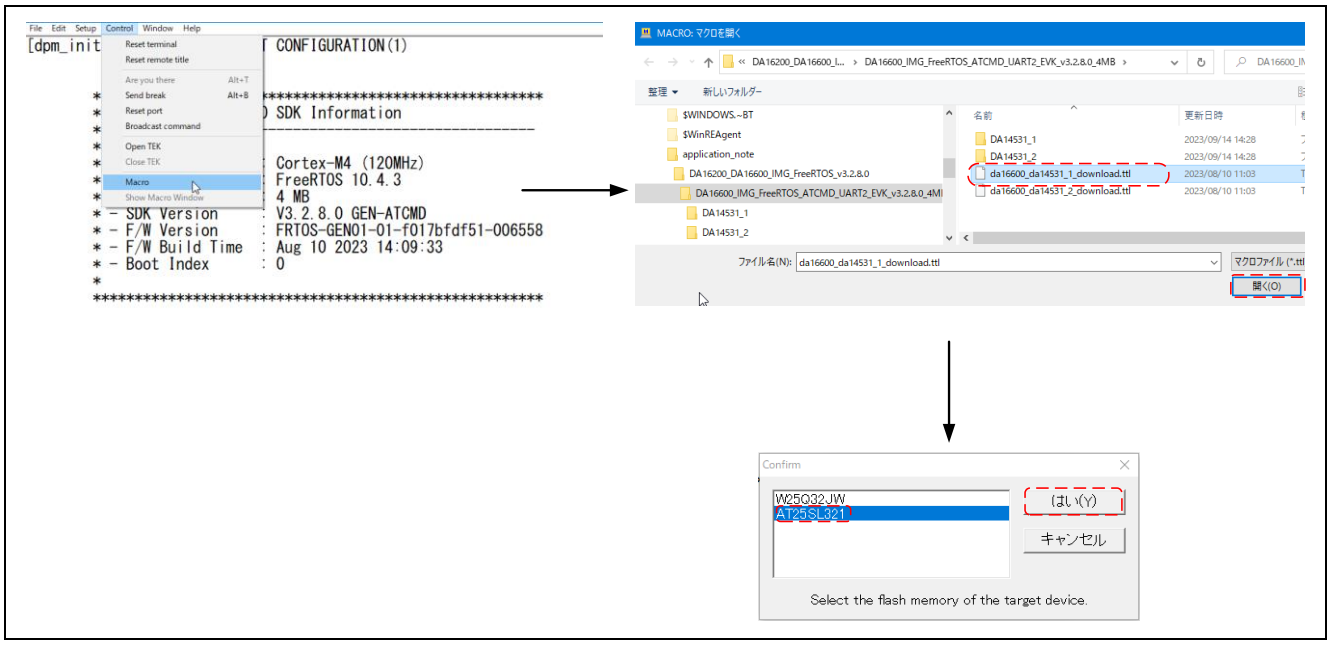

図 7-12　ファームウェアの更新

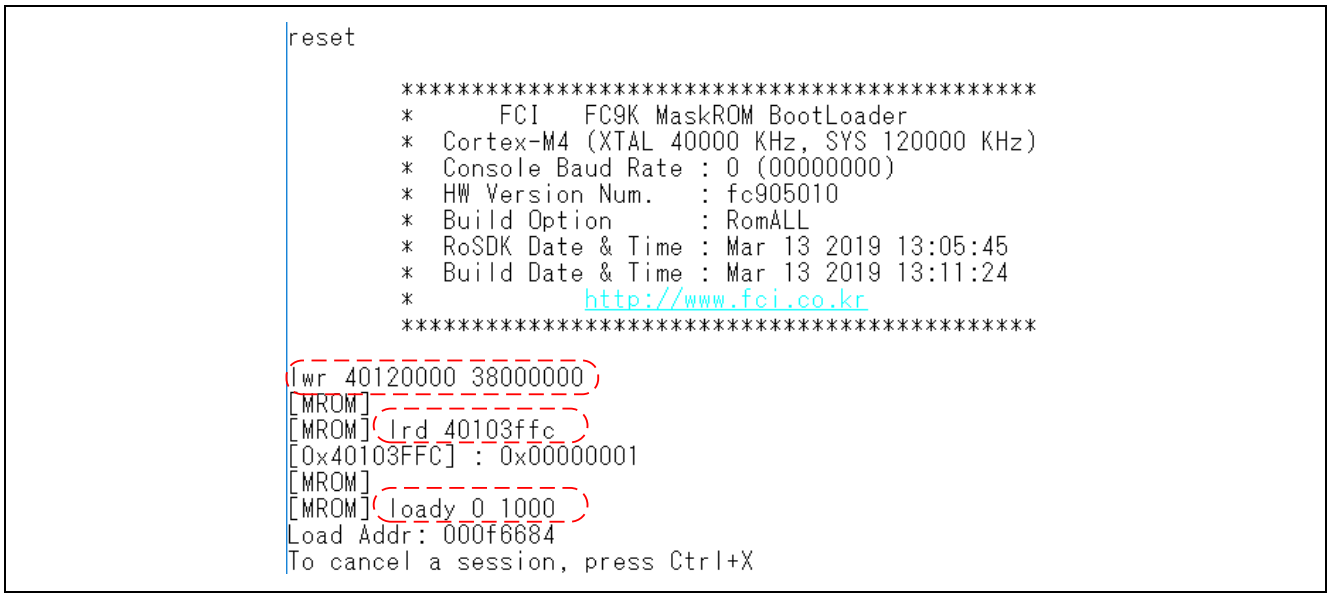

図 7-13　ファームウェアの更新中の Tera Term 画面

7.1.2.9 バージョンを確認する

ファームウェアがアップデートされた後ファームウェアのバージョンが v3.2.8.0 になっていることを確認してください。

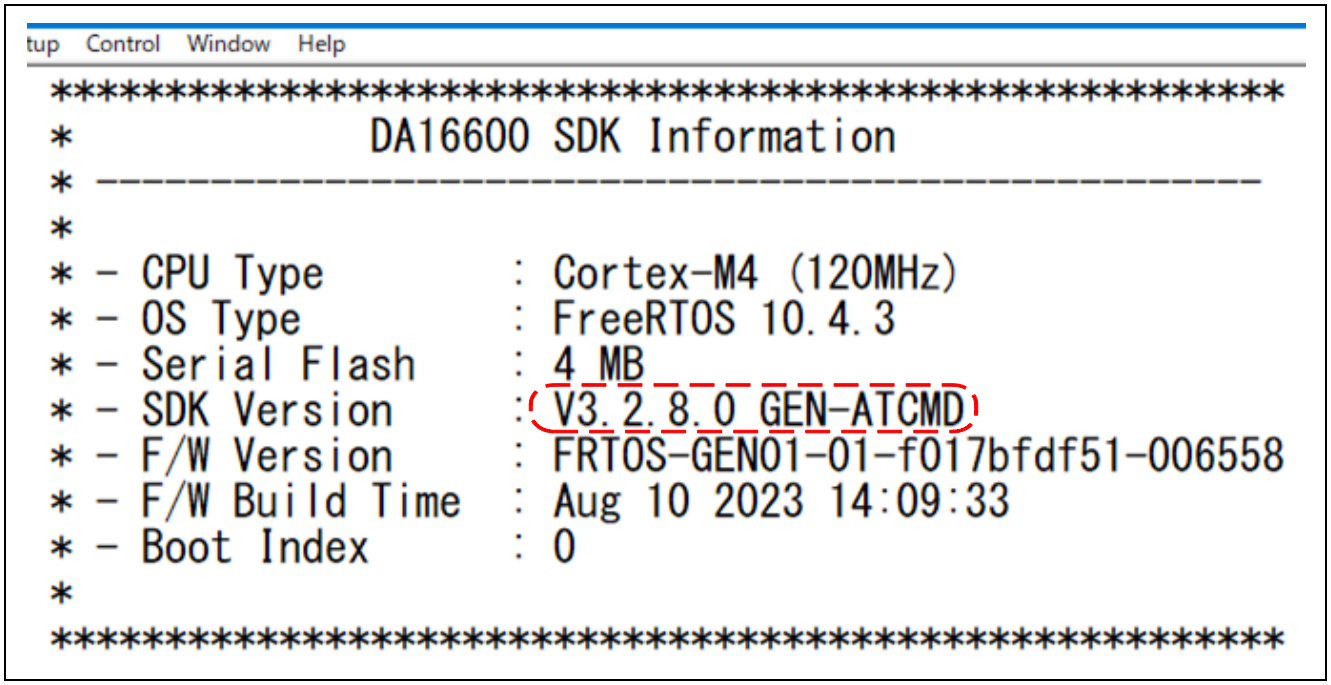

図 7-14 ファームウェアのバージョン確認

7.1.3 Fail to establish tls-sess(0x7200)が表示される場合

以下のように Fail to establish tls-sess(0x7200)が表示される場合について説明します。

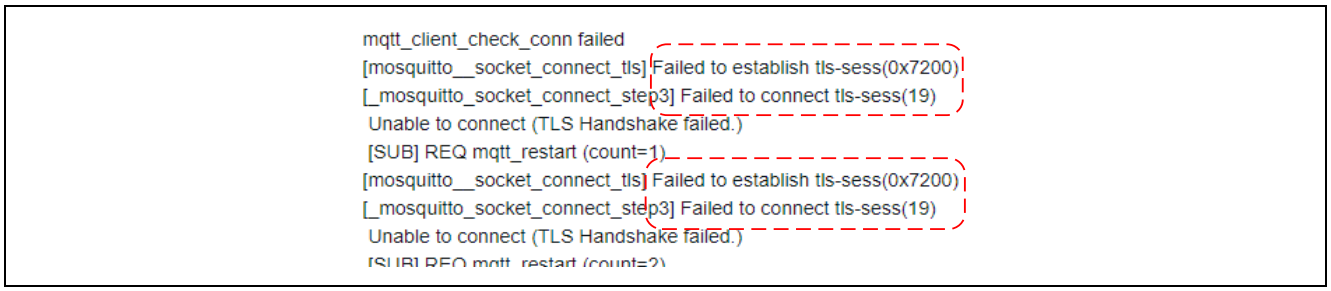

図 7-15 Fail to establish tls-sess(0x7200)が表示される場合

7.1.3.1 MQTT 用バッファを再設定する

Fail to establish tls-sess(0x7200)が表示される場合 MQTT 用のバッファを再設定します。以下のコマンドを実行し再設定を行ってください。

```
setenv MQTT_TLS_INCOMING 16384
setenv MQTT_TLS_OUTGOING 16384
```

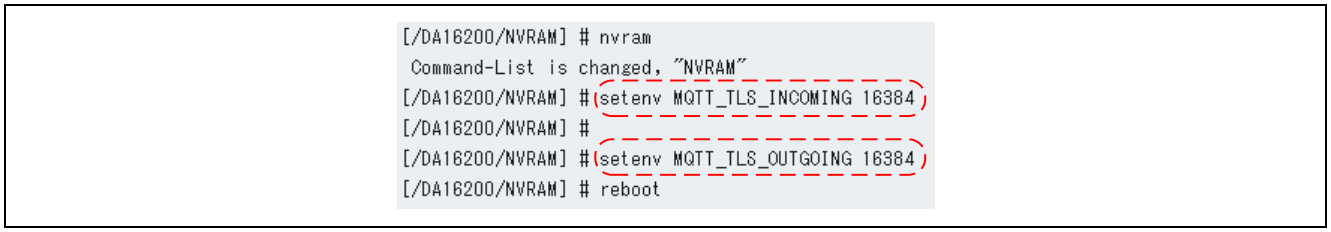

図 7-16 MQTT バッファ再設定

7.1.3.2 設定結果の確認する

以下のコマンドを実行し再設定されていることを確認してください。

```
mqtt_config status
```

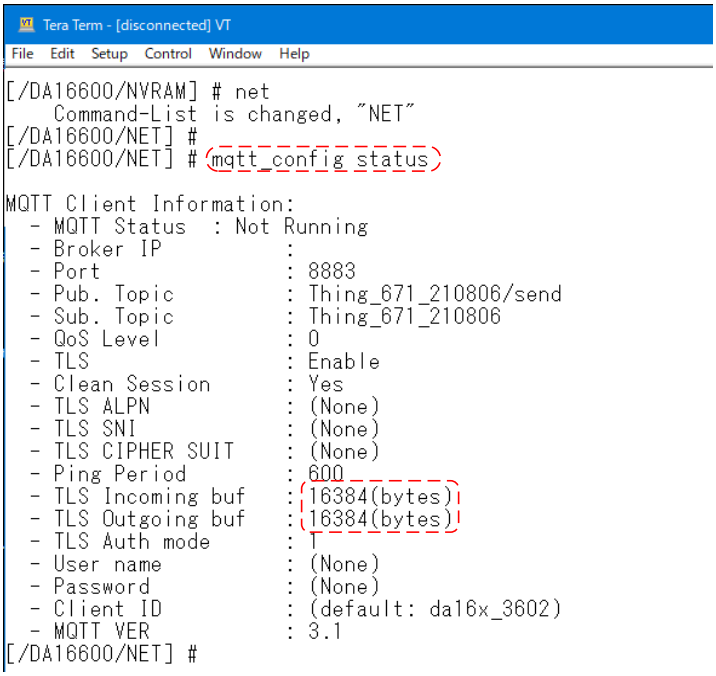

図 7-17　MQTT バッファ再設定確認結果

8. 参考資料

- US159-DA 16600 EVZ Evaluation Board Manual（R15UZ0006）
- User Manual DA16200 DA16600 AT Command（UM-WI-003）
- 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 （ R01AN6850）
- 永久磁石同期モータのセンサレスベクトル制御 – MCK 用（R01AN6858）
- MCK-RX26T ユーザーズマニュアル（R12UZ0111）
- 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）
- Renesas Motor Workbench のユーザーズマニュアル（R21UZ0004）

最新版をルネサスエレクトロニクスホームページから入手してください。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jan.5.24	―	初版
1.01	Dec.23.24	9	図 2-3 接続イメージを修正。
		53	図 7-1 Pmod-USBUART と DA16600 Pmod™ Board の接続方法を修正。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。