RX Family

Demonstration of an OTA Firmware Update on the RX26T Using the Dual Bank Feature

Introduction

This application note describes the software sample program that updates firmware via OTA (Over The Air) using a Renesas microcontroller (MCU). The sample program described in this application note demonstrates an OTA firmware update to switch between current detection programs that use the 2-shunt and 1-shunt methods. These programs are bundled with the following application note: “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

The target software for this application note is for reference only; operations cannot be guaranteed. Only use the target software of this application note after conducting thorough evaluation in an appropriate environment.

Target Device

Operations of the target software of this application note are checked by using the following device.

MCU used:

RX26T

Contents

1. Overview

This application note describes the software sample program that updates firmware via OTA (Over The Air) using a Renesas microcontroller (MCU). This sample program can be used to demonstrate an OTA update for the RX26T.

This sample program uses Amazon Web Service (AWS) as a cloud service. With two software update programs uploaded to Amazon S3, cloud-related complicated operations are performed via AWS Lambda. AWS IoT Core is used as a gateway for connection to AWS. One of two update programs is downloaded to the RX26T through the DA16600 Pmod™ board that is connected to the internet via Wi-Fi. The downloaded update program uses its firmware update FIT module (FIT: Firmware Integration Technology) to program itself. For details on this module, refer to the following application note: “RX Family Firmware Update Module Using Firmware Integration Technology” (R01AN6850). The RX26T used in this application note has the dual bank feature. Therefore, you can perform an OTA firmware update while the existing program is running. This application note shows an example in which an OTA update is implemented in a bare metal configuration that does not use FreeRTOS.

1.1 About an OTA Update in This Application Note

We, Renesas, have a policy to use a bootloader when implementing an OTA update from the viewpoint of security risks as described in “Renesas MCU Firmware Update Design Policy” (R01AN5548). This is because the bootloader has mechanisms ensuring safe program operation, such as a function that verifies that the program area is not falsified before startup and a function that installs firmware where no user programs exist. The firmware update FIT module used in this application note also has such security functions. Implementing an OTA update with these security functions requires much more complicated control than when implementing an ordinary OTA update. The complicated control may include security checking. However, security checking is unnecessary overhead for an OTA update in a LAN environment that has no security risks.

In this application note, an OTA update is implemented with minimum necessary functions of the firmware update FIT module without using security-related functions and other functions that have no direct relation to the OTA update.

For example, the bootloader used in this application note provides only the processing that starts the main program, although the normal bootloader verifies the program area before startup as mentioned above.

1.2 AWS Environment

The AWS environment must be built with the customer's own account. For details on how to create an account, refer to “4. Building the AWS Environment”.

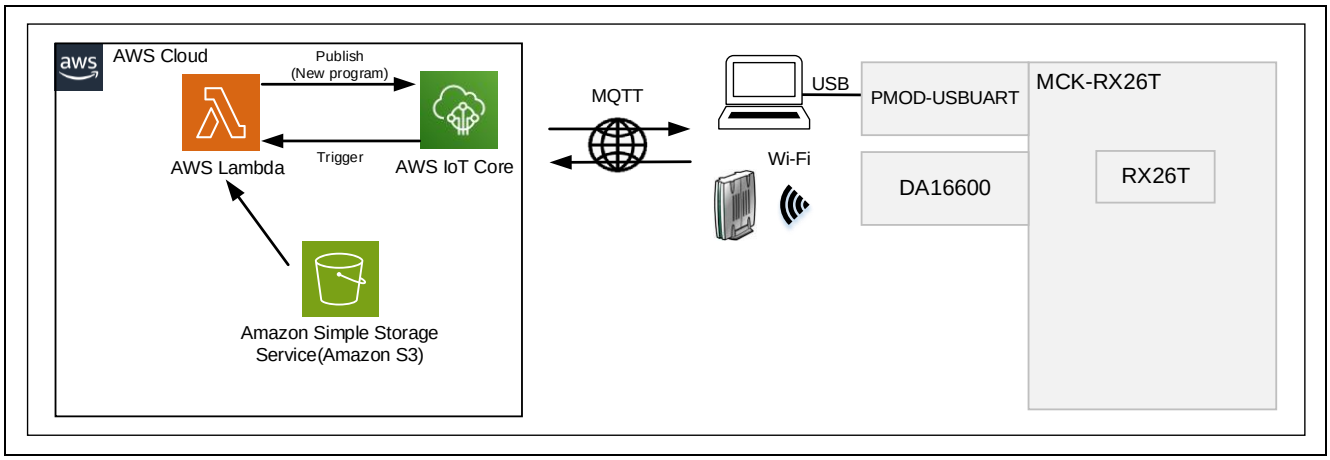

Figure 1-1 System Configuration for OTA Update Demonstration

2. Environment in Which Operation Was Verified

Table 2-1 and Table 2-2 show the environments in which the target software for this application note was developed.

Table 2-1 Hardware Development Environment

Category	Product used
MCU/CPU board model	RX26T (R5F526TFCDFP) / RTK0EMXE70C00000B
Inverter board	Inverter board (for a 48 V 10 A BLDC motor) bundled with the Renesas Flexible Motor Control Kit (RTK0EMXE70S00020BJ)
Wi-Fi module	DA16600 Pmod™ Board (US159-DA16600EVZ)
Motor	R42BLD30L3 (MOONS')
USB-to-UART converter module	Pmod USBUART (Digilent)

Table 2-2 Software Development Environment

IDE version	RX Smart Configurator	Toolchain version
CS+: V8.10.00	Version 2.19.0	CC-RX: V3.05.00
e^{2}studio: 2023-10	e^{2}studio plug-in	

Table 2-3 Conditions Under Which Operation Was Verified (DA16600 Pmod™ Board)

Item	Description
Board used	US159-DA16600EVZ Pmod Board
Firmware	DA 16600 v3.2.8.0 • This firmware performs Wi-Fi connection and TLS communication with AWS. • If the version of the firmware you have is different from the indicated version, refer to “7.1.2 Updating the Version of the Firmware of the DA16600 Pmod™ Board”.

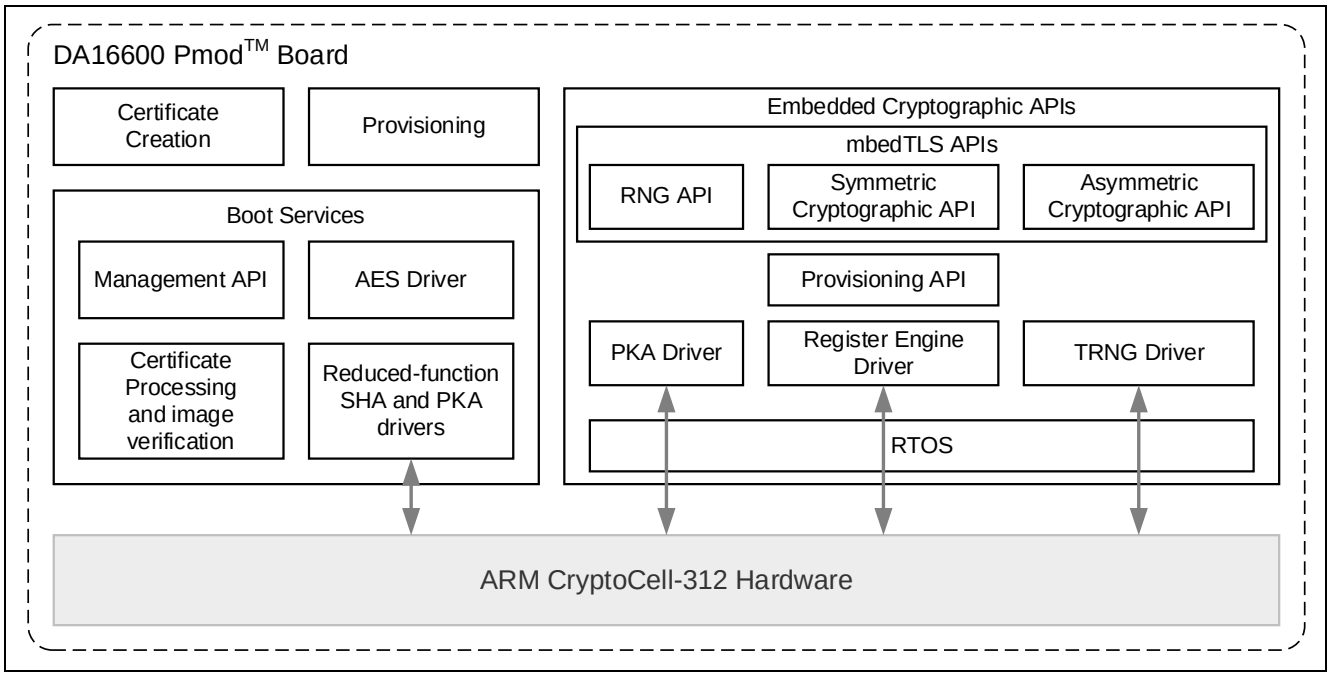

Figure 2-1 DA16600 (DA16200) Software Stack

For the purchase or technical support of this system, contact a Renesas Electronics Corporation sales representative or an authorized Renesas Electronics Corporation product distributor.

2.1 Hardware Specifications

2.1.1 Hardware Configuration Diagram

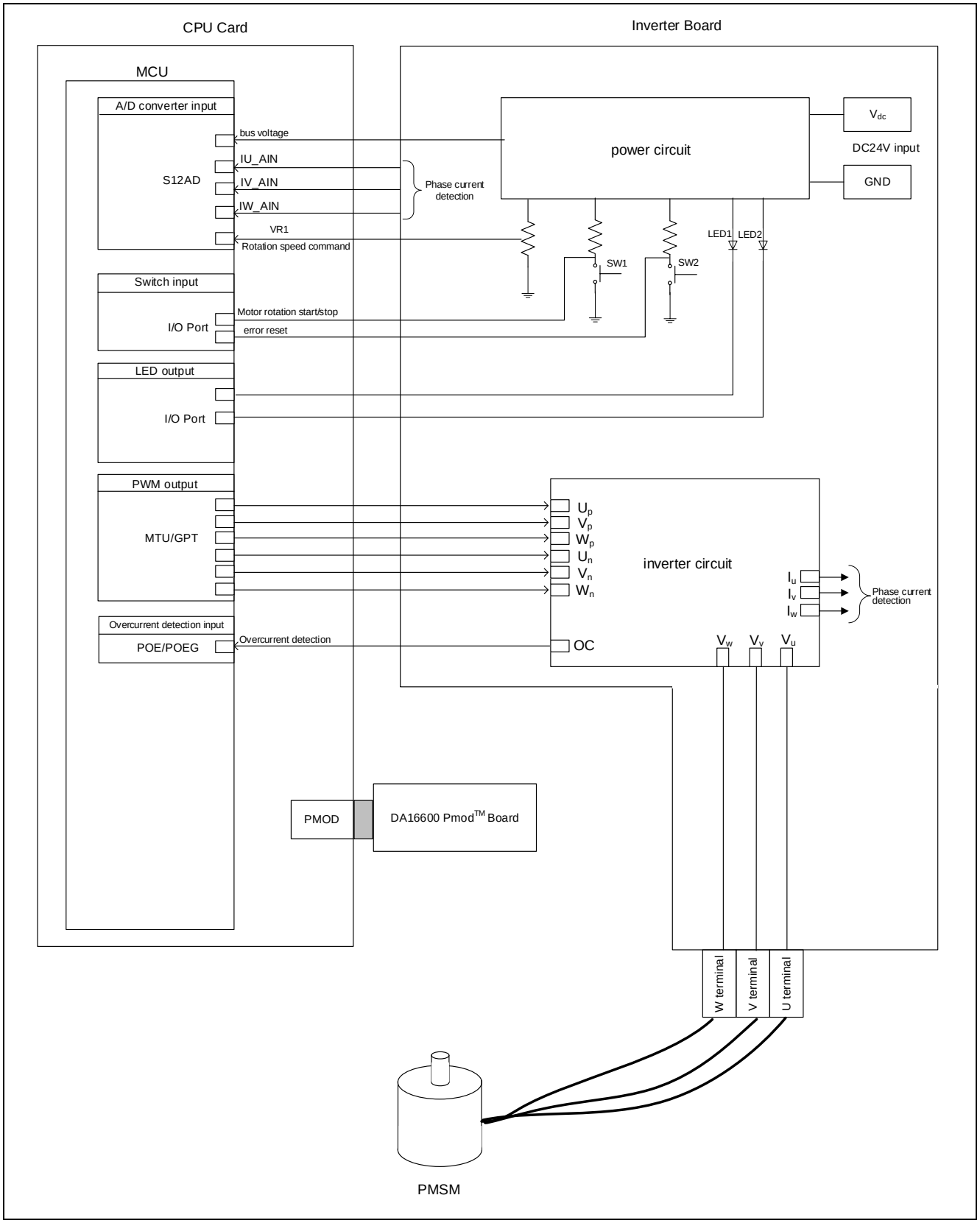

Figure 2-2 Hardware Configuration Diagram

2.1.2 Board User Interface

Table 2-4 lists the components of the board user interface in this system.

Table 2-4 Board User Interface

Item	Interface component	Function
Rotation speed	Variable resistor (VR1 on the inverter board)	Inputs the rotation speed command value.
START/STOP	Toggle switch (SW1 on the inverter board)	Instructs start or stop of motor rotation.
ERROR RESET	Push switch (SW2 on the inverter board)	Instructs recovery from an error state.
LED1	Orange LED (CPU board/inverter board)	• On: The motor is rotating. • Off: The motor is stopped.
LED2	Orange LED (CPU board/inverter board)	• On: An error was detected. • Off: The system is operating normally.
RESET	Push switch (SW1 on the CPU board)	System reset

2.2 Board Connections

The figure below shows the boards that have been connected.

The CPU board has a JP11 jumper pin that must be opened when writing programs. The inverter board has JP8 and JP11 jumper pins that are used to switch between the current detection programs that use the 1-shunt and 2-shunt methods.

Note that a Pmod USBUART cannot be directly connected to the PMOD connector because their PMOD types are different. For details on how to connect the Pmod USBUART, refer to “5.6.1 Connecting a Pmod USBUART”.

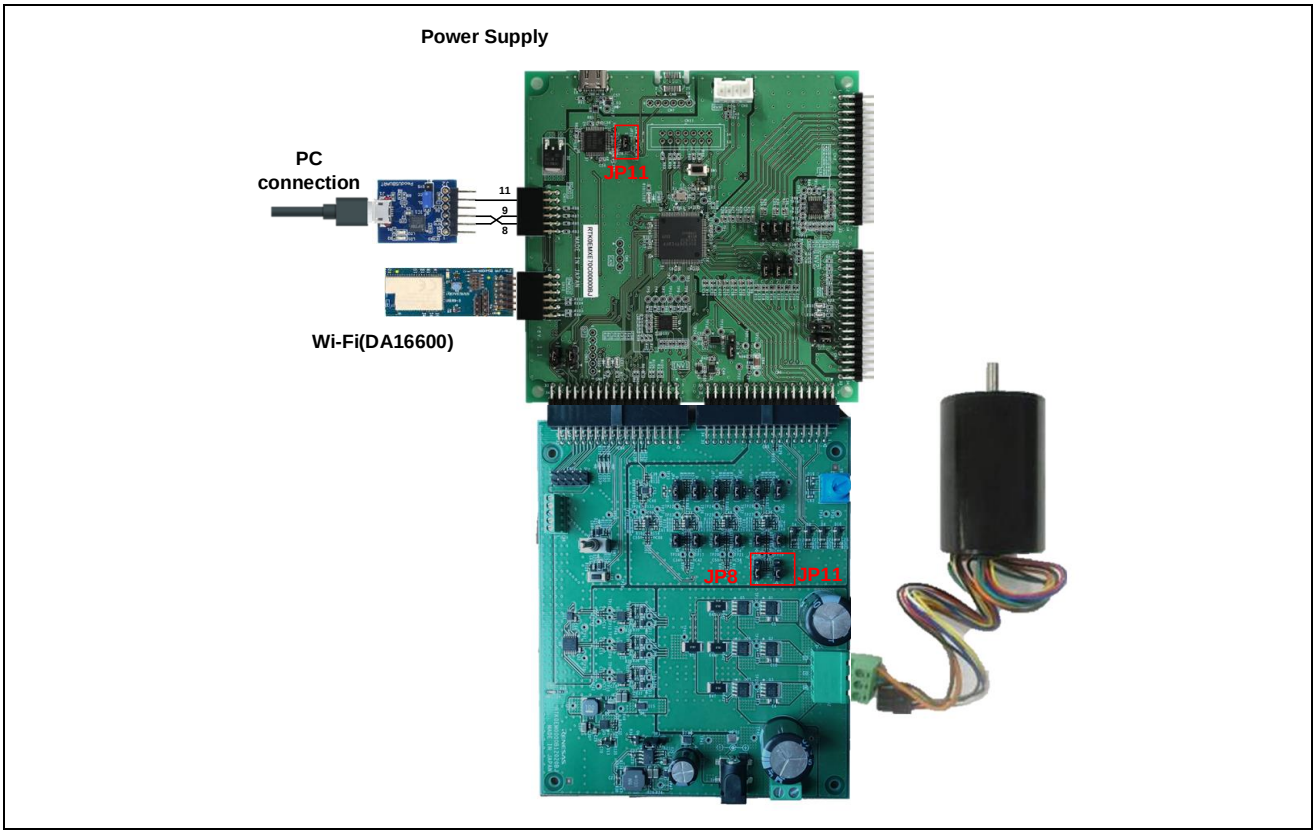

Figure 2-3 Connected Boards

3. Quick Start Guide

This chapter provides a quick start guide for you to experience OTA technology by using the Renesas Flexible Motor Control Kit and the sample program provided by this application note. For details on how to drive a motor, refer to “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858). For details on the settings and connections of the boards bundled with the Renesas Flexible Motor Control Kit, refer to the “MCK-RX26T User's Manual” (R12UZ0111).

For details on how to connect the boards for use with this application note, refer to “5.6 Hardware Preparation”.

Note that you cannot directly connect a Pmod USBUART to the CN10, which is a connector for a PMOD Type 6A module. Connect them by referring to “5.6.1 Connecting a Pmod USBUART”.

3.1 Overview of an OTA Update in the Sample Program

This sample program achieves an OTA update by implementing the minimum functionality based on “1.3.1 Dual-Bank Method” in “RX Family Firmware Update Module Using Firmware Integration Technology” (R01AN6850). “Figure 3-1 Overview of an OTA Update” shows an overview of an OTA update.

“Before OTA” phase: A bootloader and a main program are merged into a Motorola S-record file (“xxxx.mot” at (1) in the figure), which is used as an initial image. Then, the file is written to the MCU (RX26T) and then the program is executed.

“OTA in progress” phase: An OTA update is started by an instruction from the PC. While the OTA update is in progress, the RX26T simultaneously downloads an update image (“xxxx.rsu” at (3) in the figure) from AWS and writes it to a buffer. At this time, the program in the main area also runs simultaneously with the OTA update.

“Swap bank” phase: When the firmware update in the buffer is completed and a reset instruction is sent from the PC, the RX26T performs a bank swap.

“After OTA” phase: The RX26T then resets itself and starts executing the update image.

For details on the initial and update images, refer to “3.2 Initial and Update Images”.

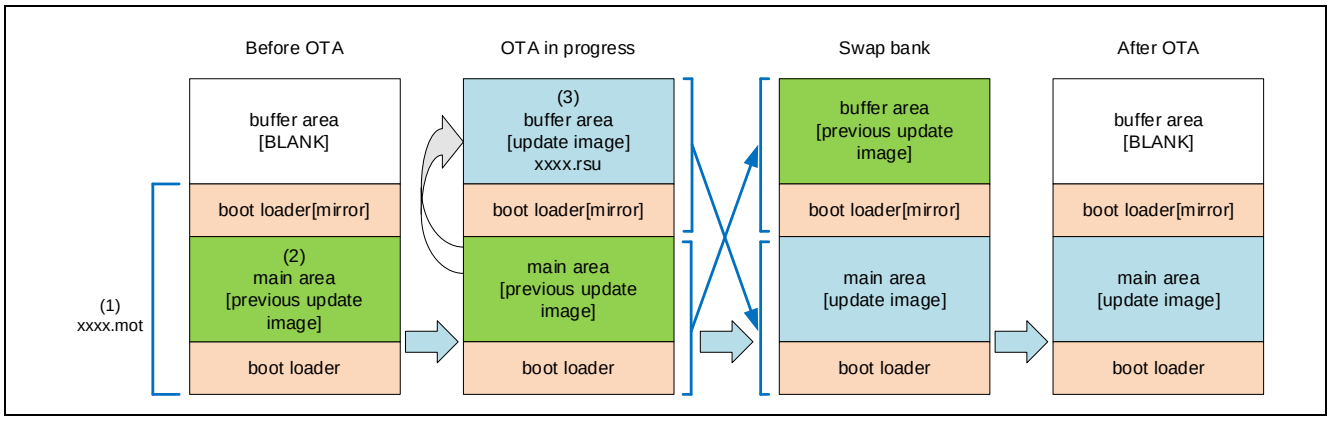

Figure 3-1 Overview of an OTA Update

3.2 Initial and Update Images

This sample program is designed to work as a combination of a bootloader and main program. Each project does not work independently. For the sample program to operate, you create one initial image that can run on the MCU (RX26T) and two update images to be uploaded to Amazon S3. To create initial and update images, you must prepare Motorola S-record files (*.mot) from which to generate the images. These Motorola S-record files are generated by building the projects that make up this sample program.

3.2.1 Building Projects

This sample program consists of three projects. Build these projects to prepare Motorola S-record files. For details on how to build a project, refer to “6. Procedure for Building a Project”. If you use e^2 studio to build projects, the build result will vary depending on whether the necessary FIT modules are present. Make sure that all necessary FIT modules have been downloaded by referring to “6. Procedure for Building a Project”.

Table 3-1 Projects

Project name	Description
r01an7203_rx26t_boot_loader	Bootloader
r01an7203_rx26t_fwup1	Main program Updates the firmware while performing the following processing: RX26T_MCBA_MCILV1_SPM_LESS_FOC_1SHUNT_E2S_V110
r01an7203_rx26t_fwup2	Main program Updates the firmware while performing the following processing: RX26T_MCBA_MCILV1_SPM_LESS_FOC_E2S_V110

3.2.2 Creating an Initial Image

An initial image is created as a combination of a bootloader and a main program. In the figure in “3.1 Overview of an OTA Update in the Sample Program”, the portion indicated by “(1) xxxx.mot” is an initial image. The items listed in “Table 3-1 Projects” are used to create an initial image. One of two initial images can be created as a combination shown in “Table 3-2 Combinations of a Bootloader and a Main Program to Generate an Initial Image”. Refer to “(2) main area” in the figure in “3.1 Overview of an OTA Update in the Sample Program”. The main program “r01an7203_rx26t_fwup1” or “r01an7203_rx26t_fwup2” copied to the main area and a bootloader are combined into an initial image that can run on the MCU (RX26T).

Use Renesas Image Generator (the “image-gen.py” program) to generate an initial image. For details on this program, refer to “4. Renesas Image Generator” in “RX Family Firmware Update Module Using Firmware Integration Technology” (R01AN6850). Do not specify the “-vt ecdsa” option. (Because SHA-256 is selected instead of ECDSA, you do not need to specify a key file necessary for signature generation or verification.)

Example of the command line:

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

Table 3-2 Combinations of a Bootloader and a Main Program to Generate an Initial Image

—	Combination
Initial image 1	“r01an7203_rx26t_boot_loader.mot” + “r01an7203_rx26t_fwup1.mot”
Initial image 2	“r01an7203_rx26t_boot_loader.mot” + “r01an7203_rx26t_fwup2.mot”

3.2.3 Creating an Update Image

In the figure in “3.1 Overview of an OTA Update in the Sample Program”, the portion indicated by “(3) xxxx.rsu” is an update image. Create two update images with “r01an7203_rx26t_fwup1.mot” and “r01an7203_rx26t_fwup2.mot”. In the same way as when creating an initial image, use Renesas Image Generator (the “image-gen.py” program) to generate an update image. For details on this program, refer to “4. Renesas Image Generator” in “RX Family Firmware Update Module Using Firmware Integration Technology” (R01AN6850). Do not specify the “-vt ecdsa” option. (Because SHA-256 is selected instead of ECDSA, you do not need to specify a key file necessary for signature generation or verification.)

Upload the update images created here to Amazon S3.

Example of the command line:

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

3.2.4 Structure of Files on Amazon S3

The following shows an example of the structure of files on Amazon S3.

Table 3-3 Structure of Files on Amazon S3

File structure on Amazon S3	Description
Amazon S3	—
└ rx26t_Sample	Bucket name
├ RX26T_1Shunt.rsu	.rsu file (update image 1)
└ RX26T_2Shunt.rsu	.rsu file (update image 2)

3.3 Motor Control UI Settings

The sample program of this application note has been created on the basis of an existing sample program. In the sample program of this application note, the motor control UI has been changed to MAIN_UI_BOARD.

This UI allows you to control the motor with on-board switches and an on-board variable resistor.

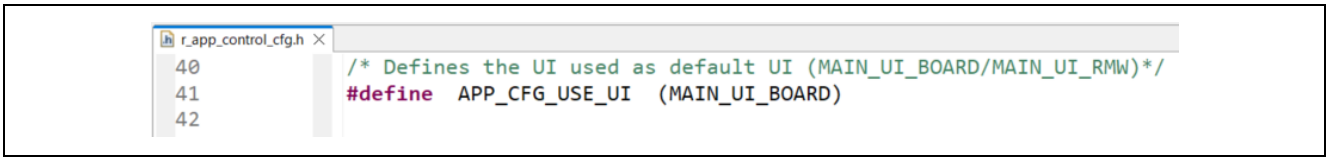

Figure 3-2 Section Where the Motor Control UI Was Changed

3.4 Preparation on the AWS Side

Before you can perform an OTA update by using AWS, you must make preparations on the AWS side. You need to obtain your own AWS account. After obtaining your AWS account, you need to complete some procedures such as registration of a device (thing) and issuance of a certificate. For details on these procedures, refer to “4. Building the AWS Environment”.

4. Building the AWS Environment

4.1 Registering a Device with AWS IoT

Specify the AWS settings by referring to “Register your device (thing) with AWS IoT” in the following tutorial.

- Specify the settings up to “Check AWS IoT endpoints”.
- When you download a certificate, also download a public key file, private key file, and a root CA for AWS IoT (Amazon root CA 1).

Register your device (thing) with AWS IoT: renesas/amazon-freertos Wiki · GitHub

4.2 Preparation for Lambda

4.2.1 Creating a Function

In the AWS Lambda console, in [Functions], click [Create function].

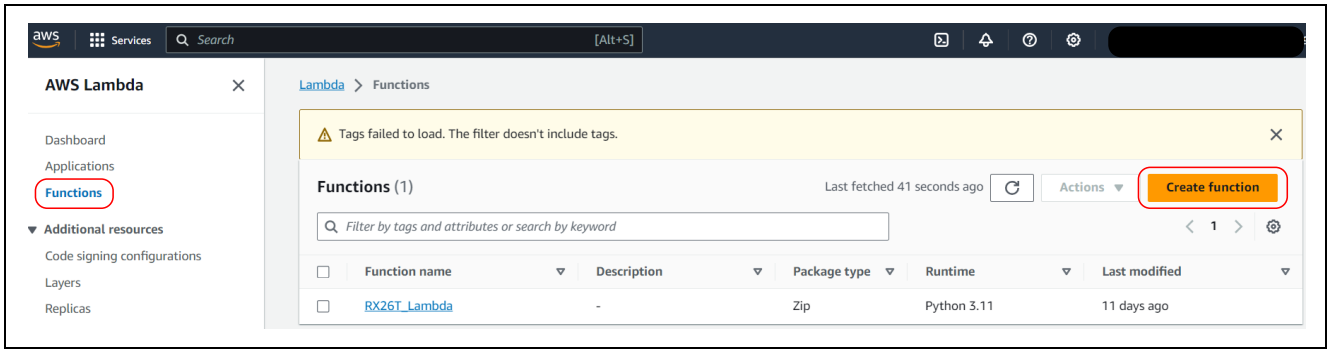

Figure 4-1 Starting Creation of an AWS Lambda Function

1. Select [Author from scratch].
2. Enter a function name. (This document assumes that you enter “RX26T_Lambda” here.)
3. From the drop-down list, select “Python3.11”.
4 Click [Create function]. The function is created.

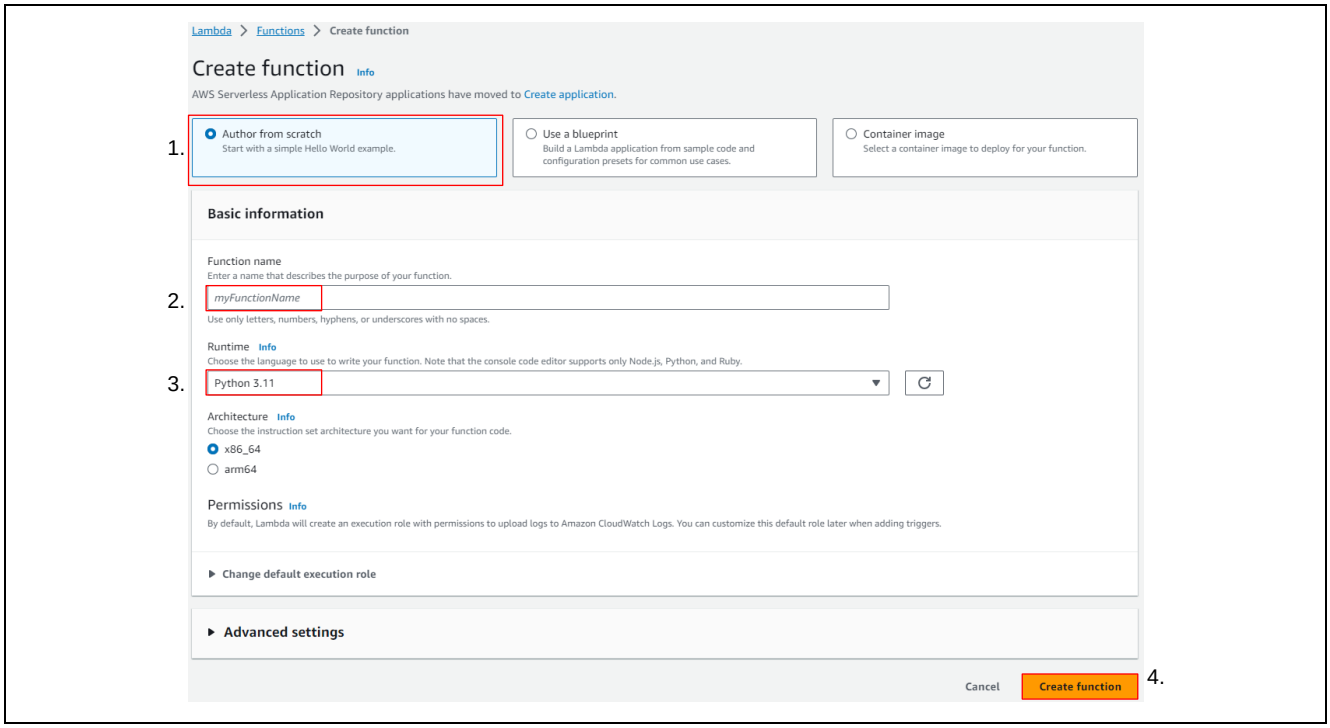

Figure 4-2 Creating an AWS Lambda

4.2.2 Adding a Trigger

Click [Add trigger].

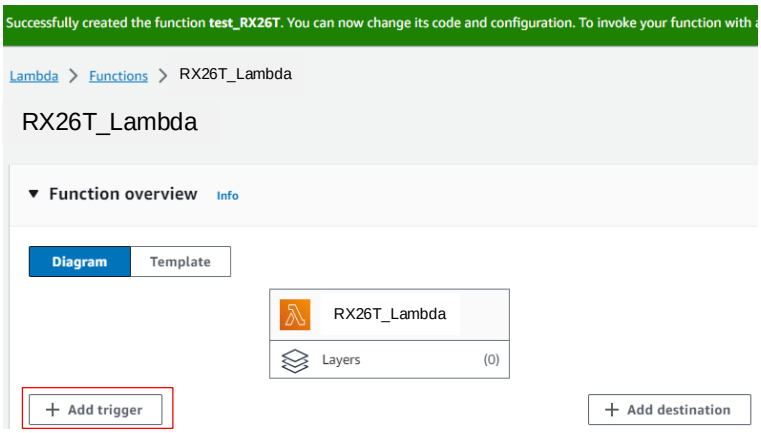

Figure 4-3 Adding a Trigger

1. Select [AWS IoT].
2. Select the [Custom IoT rule] radio button.
3. Select the [Create a new rule] radio button.
4. Enter a trigger name. (This document assumes that you enter “RX26T_Lambda_Trigger” here.)
5. Enter a rule. (This document assumes that you enter “SELECT * FROM ‘RX26T_Topic/FROM_EDGES’” here.) AWS IoT follows this rule when sending MQTT messages to Lambda.
6. Click [Add].

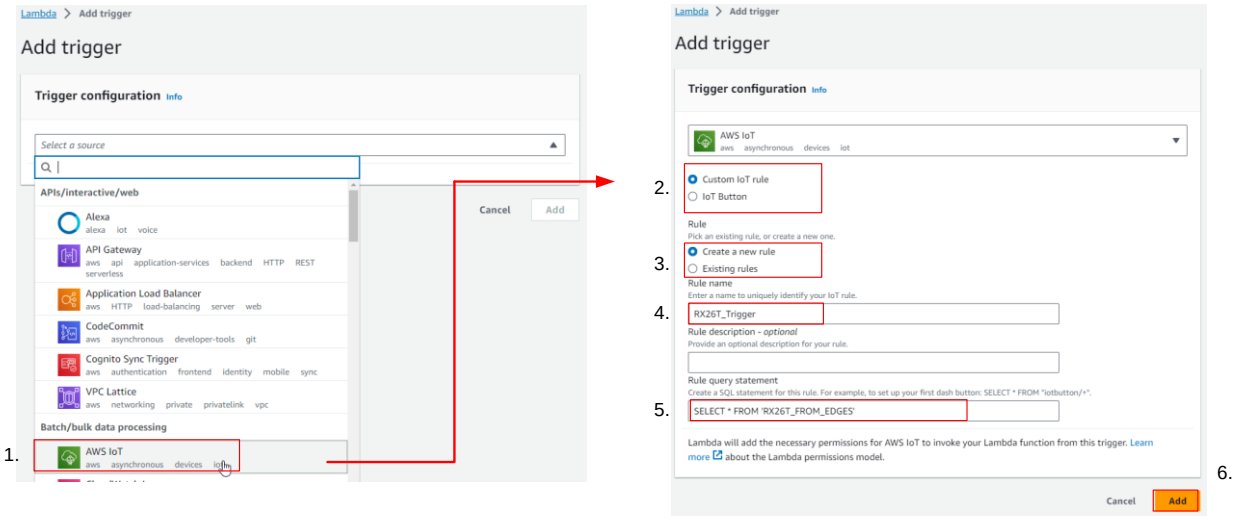

Figure 4-4 Procedure for Adding a Trigger

4.2.3 Setting a Role

Set a role.

1. Click [Configuration].
2. Click [Permissions].
3. Click a role name.

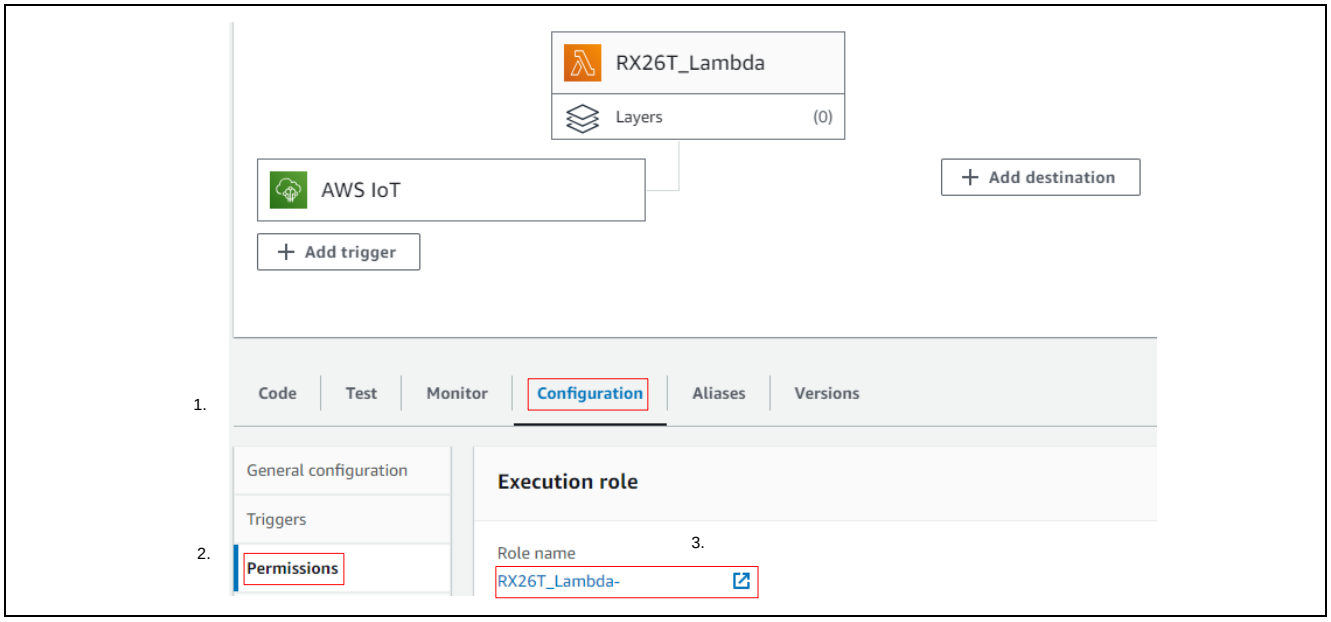

Figure 4-5 Setting a Role

From [Add permissions], select [Attach policies].

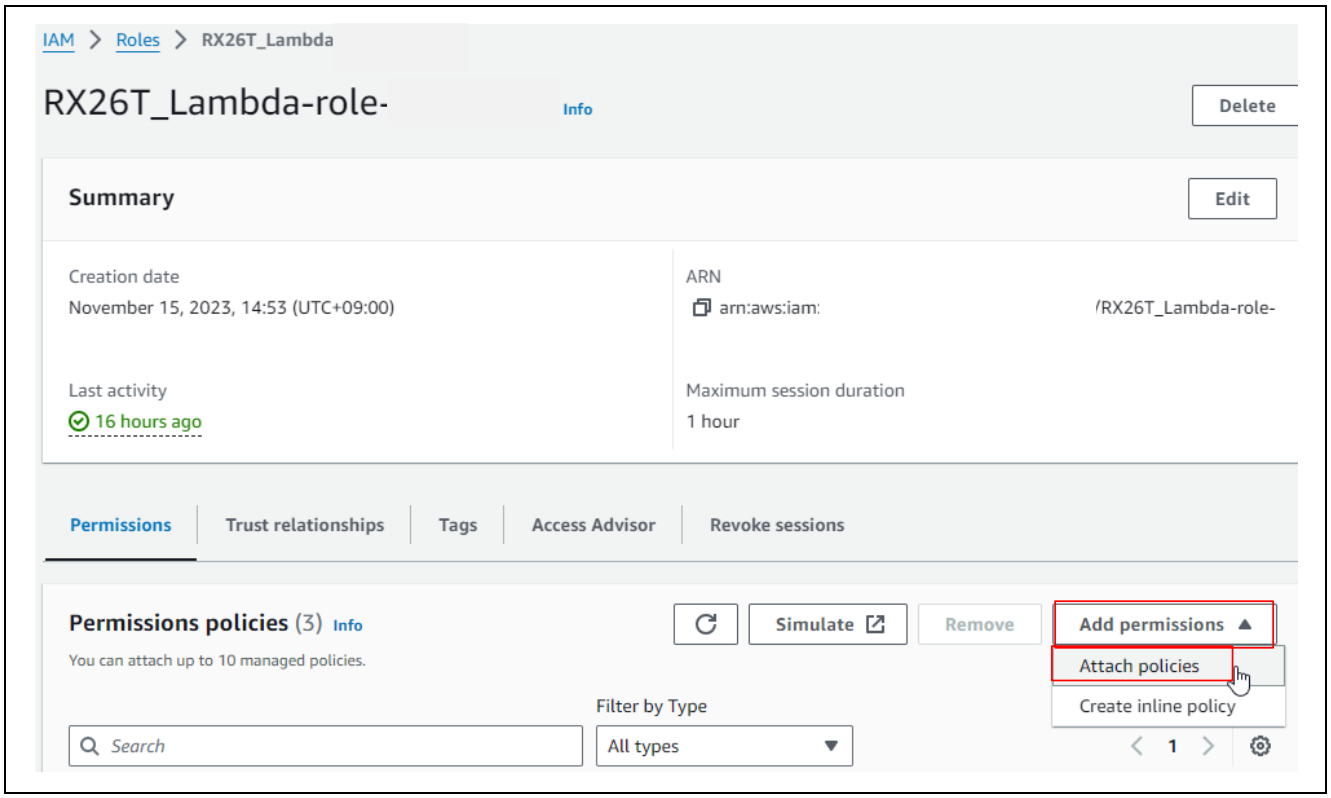

Figure 4-6 Attaching Policies to the Role (First Time)

1. In the search bar, enter “S3” to narrow down the search results.
2. Select the [AmazonS3FullAccess] check box.
3. Click [Add permissions].

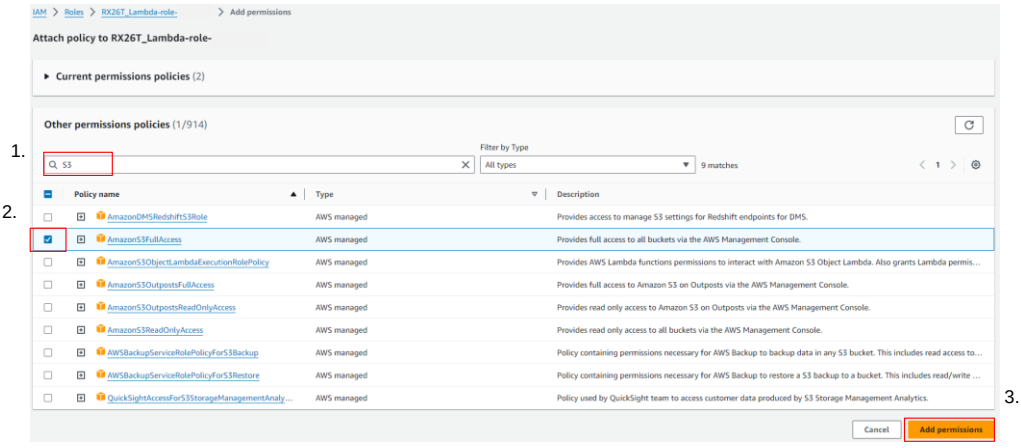

Figure 4-7 Adding Permissions for S3

From [Add permissions], select [Attach policies] again.

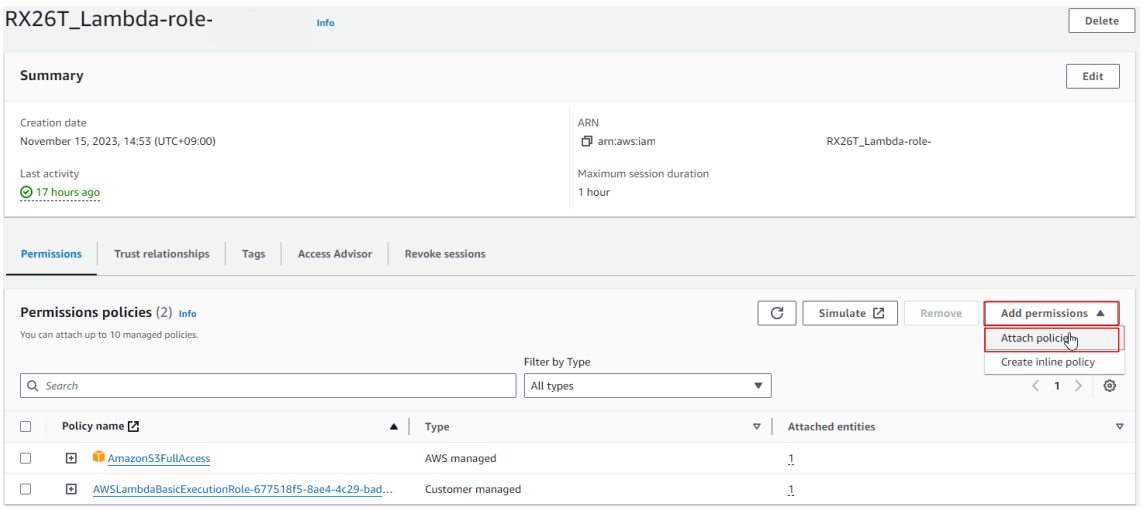

Figure 4-8 Attaching Policies to the Role (Second Time)

1. In the search bar, enter “IoTData” to narrow down the search results.
2. Select the [AWSIoTDataAccess] check box.
3. Click [Add permissions].

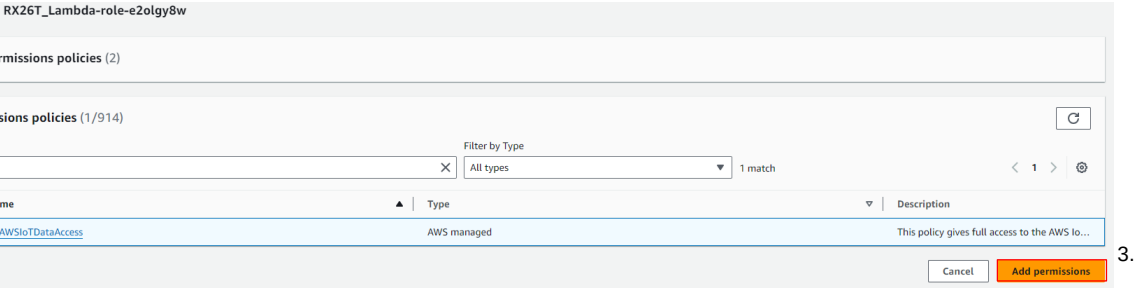

Figure 4-9 Adding Permissions for AWS IoT

4.2.4 Registering the Lambda Code

Click [Code], and then double-click [Lambda Function.py] to open the [lambda_function] tab. Replace the code in the tab with the code shown in “4.2.5 Lambda Code for an OTA Update”. After replacement, click [Deploy] to deploy the new Lambda code.

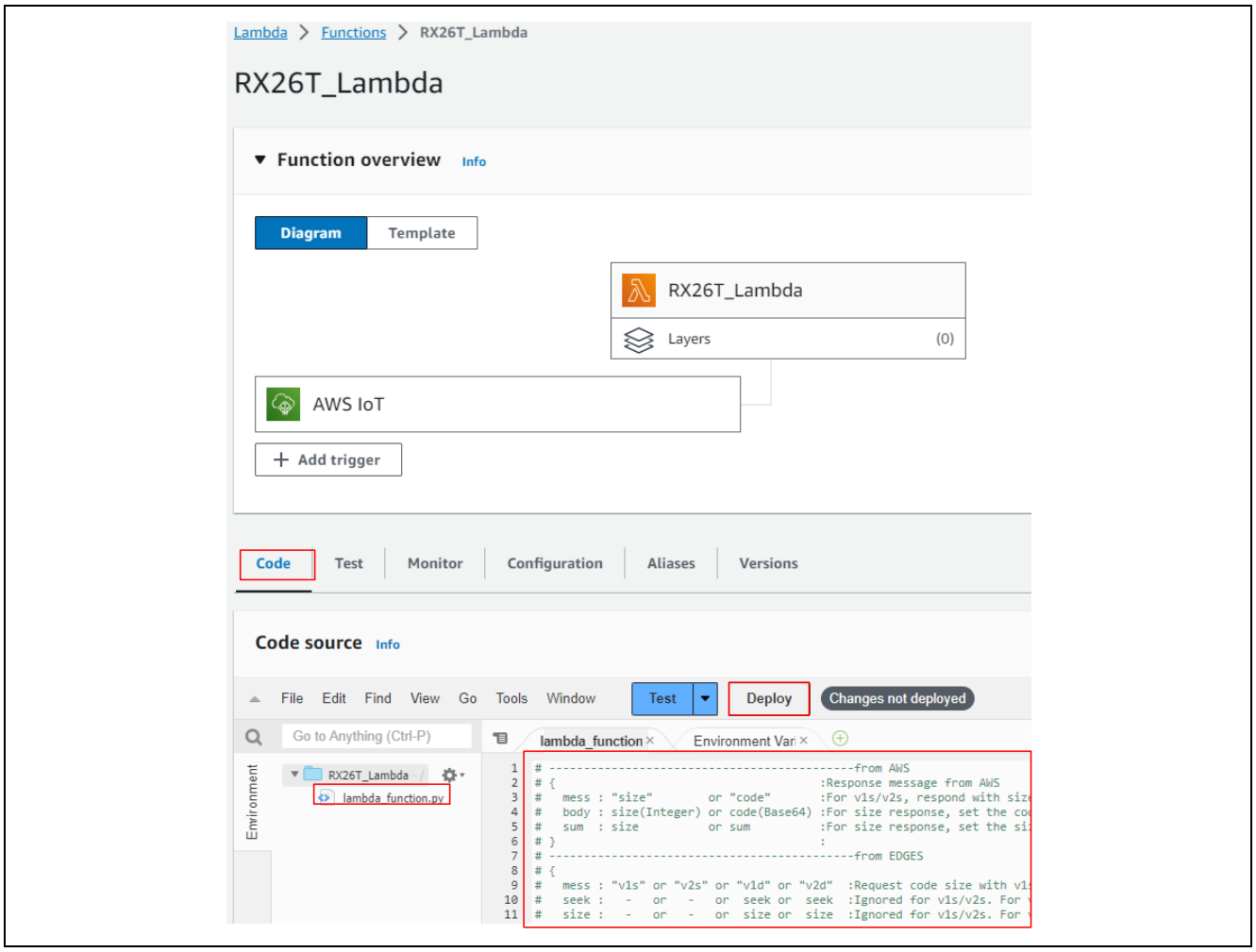

Figure 4-10 Setting the Lambda Code

4.2.5 Lambda Code for an OTA Update

An example of the Lambda code for an OTA update is shown below.
In the code example, change the values on lines in red font according to the AWS environment.

- topic:
  Specify the topic name with which to perform Publish operations.
- BUCKET_NAME:
  Specify the name of an Amazon S3 bucket.
- OBJECT_KEY_NAME1:
  Specify the name of the first update image that you created in “3.2.3 Creating an Update Image”.
  Note: This is the name of a file deployed on Amazon S3.
- OBJECT_KEY_NAME2:
  Specify the name of the second update image that you created in “3.2.3 Creating an Update Image”.
  Note: This is the name of a file deployed on Amazon S3.

```
#-------------------------------------------from AWS
#{                                      : Response message from AWS
#  mess : "size"        or "code"       : For v1s/v2s, response is "size".
#                                        For v1d/v2d, response is "code".
#  body : size(Integer) or code(Base64) : In case of "size", the code size is set.
#                                        In case of "code", the code content is set.
#  sum  : size          or sum          : In case of "size", the size is set as is.
#                                        In case of "code", the code checksum
#                                        (16 bits) is set.
#}
#-------------------------------------------from EDGES
#{
#  mess : "v1s" or "v2s" or "v1d" or "v2d"  : In case of v1s/v2s, the code size is
#                                             requested.
#                                           In case of v1d/v2d, the code content is
#                                           requested.
#  seek :   -   or   -   or  seek or  seek  : In case of v1s/v2s, this is ignored.
#                                           In case of v1d/v2d, specify the seek
#                                           position.
#  size :   -   or   -   or  size or  size  : In case of v1s/v2s, this is ignored.
#                                           In case of v1d/v2d, specify the required
#                                           size (in bytes).
#}
import json
import boto3
import base64
def lambda_handler(event, context):                 # This is called from IotCore.
    ### IoT Core Topic send settings                #
    topic           = 'RX26T_Topic/FROM_AWS'        # Specify the topic name.
    BUCKET_NAME     = 'rx26t_Sample'                # Specify the S3 bucket name.
    OBJECT_KEY_NAME1= 'RX26T_1Shunt.rsu'            # Set the file name of the 1st
                                                    # update image.
    OBJECT_KEY_NAME2= 'RX26T_2Shunt.rsu'            # Set the file name of the 2nd
                                                    # update image.
    mess = event['mess']                            # Obtains the JSON message in
                                                    # FROM_EDGES.
    seek = int(event['seek'])                       # Obtains the JSON message seek
                                                    # position in FROM_EDGES.
    size = int(event['size'])                       # Obtains the JSON message size in
                                                    # FROM_EDGES.
    iot  = boto3.client('iot-data')                 # IoTData
    s3   = boto3.client('s3')                       # S3
    sum  = 0                                        # Initializes the checksum.
```

```
    if(mess=='v1s'): # In case of v1s, the size of OBJECT_KEY_NAME1 on S3 is
                     # returned.
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME1)
        mess     = "size"                          # mess is size.
        body     = response["ContentLength"]       # Obtains the value of ContentLength
                                                   # (size) on S3.
        sum      = body                            # body is size.
    if(mess=='v2s'): # In case of v2s, the size of OBJECT_KEY_NAME2 on S3 is
                     # returned.
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME2)
        mess     = "size"                          # mess is size.
        body     = response["ContentLength"]       # Obtains the value of ContentLength
                                                   # (size) on S3.
        sum      = body                            # body is size.
    if(mess=='v1d'): # In case of v1d, the value of OBJECT_KEY_NAME1 on S3 is
                     # returned.
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME1,
Range='bytes={}-{}'.format(seek, seek+(size-1)))
        mess     = "code"                          # The seek position is specified on
                                                   # the above line.
                                                   # The data of the specified size (in
                                                   # bytes) is read.
        body     = response["Body"].read()         # The read data is set in body.
        body_len = len(body)                       # Obtains the length of body for sum
                                                   # calculation.
        for ii in range(body_len):                 # Calculates the checksum.
            sum += body[ii]                        #
        sum = sum & 0xFFFF                         # 16 bits is set.
        body     = base64.b64encode(body).decode('utf-8') # base64 encoding
    if(mess=='v2d'): # In case of v2d, the value of OBJECT_KEY_NAME2 on S3 is
                     # returned.
        response = s3.get_object(Bucket=BUCKET_NAME, Key=OBJECT_KEY_NAME2,
Range='bytes={}-{}'.format(seek, seek+(size-1)))
        mess     = "code"                          # The seek position is specified on
                                                   # the above line.
                                                   # The data of the specified size (in
                                                   # bytes) is read.
        body     = response["Body"].read()         # The read data is set in body.
        body_len = len(body)                       # Obtains the length of body for sum
                                                   # calculation.
        for ii in range(body_len):                 # Calculates the checksum.
            sum += body[ii]                        #
        sum = sum & 0xFFFF                         # 16 bits is set.
        body     = base64.b64encode(body).decode('utf-8') # base64 encoding

    payload = { "mess" : mess , "body" : body , "sum" : sum } # List of response JSON
                                                              # messages

    iot.publish(                                   # Requests Publish.
        topic = topic,                             # Destination
        qos = 1,                                   #
        payload = json.dumps(payload)              # Message body
    )
```

4.2.6 Project Source Code - Modification 1

Specify the Wi-Fi connection information and AWS connection information in the project source code.

In “src/sample_ota_add_on/sample_mqtt_config.c”, set the following five variables:

- g_mqtt_broker_endpoint[] → Endpoint name that you confirmed in “4. Building the AWS Environment”
- g_mqtt_subscriber[] → Topic name that you set in “4.2.5 Lambda Code for an OTA Update”
- g_mqtt_publisher[] → Topic name that you sent for the rule in “4.2.2 Adding a Trigger”
- g_wifi_ssid → SSID of the access point to be used
- g_wifi_password → Password of the access point to be used

Note: For the above variables, specify their values in double quotation marks.

```
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);
```

Figure 4-11 sample_mqtt_config.c

4.2.7 Project Source Code - Modification 2

Prepare the Amazon root CA 1 certificate, device certificate, and private key that you downloaded in “4.1 Registering a Device with AWS IoT”, and set their content to the source code.

In “src/sample_ota_add_on/sample_mqtt_config.c”, set the following three variables:

- g_mqtt_root_cetificate_pem[] → Set the content of the Amazon root CA 1 certificate as in Figure 4-12.
- g_mqtt_cetificate_pem_cert[] → Set the content of the device certificate as in Figure 4-13.
- g_mqtt_private_pem_key[] → Set the content of the private key as in Figure 4-14.

Note: Note the following points.

- All lines except the last line must end with “\n”.
- Each line must be enclosed in double quotation marks.
- The right quotation marks on all lines except the last line must be followed by a back slash (\).

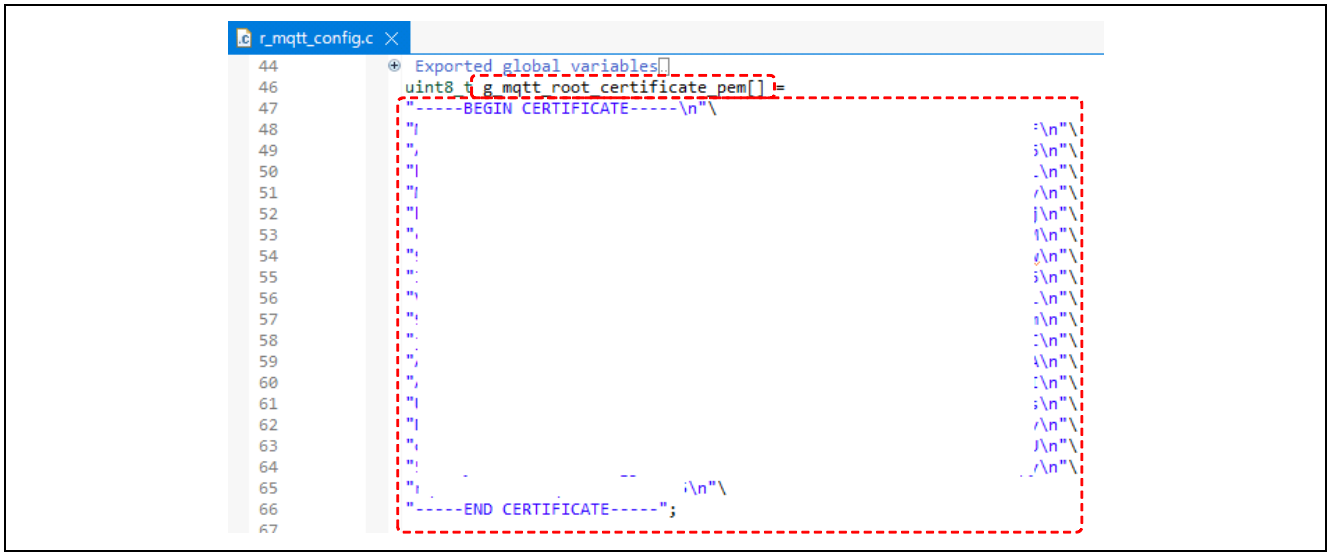

Figure 4-12 Setting the Content of the Root CA Certificate

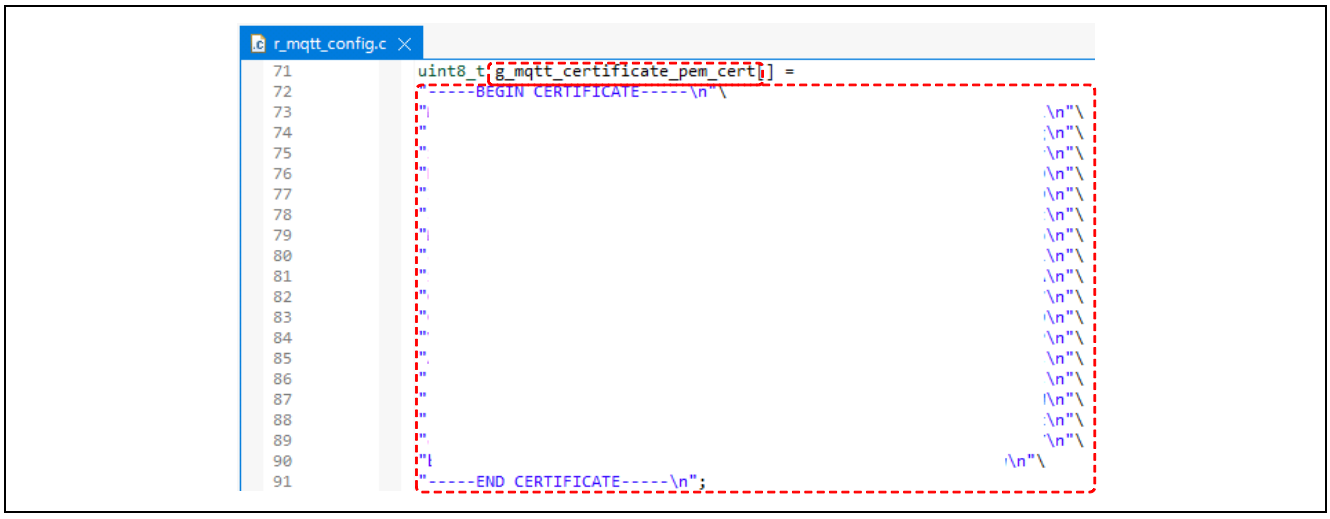

Figure 4-13 Setting the Content of the Device Certificate

Figure 4-14 Setting the Content of the Private Key

5. Sample Program

5.1 System Configuration

“Figure 5-1 System Configuration Including AWS for Demonstration” shows the configuration of the system in which this sample program is used.

The two update images must be stored on Amazon S3 of AWS beforehand. All instructions related to an OTA update are sent from the PC to the MCK-RX26T. When an update instruction (FWUP1 or FWUP2) is entered from the PC, the MCU (RX26T) issues a request for downloading the corresponding update image through MQTT to AWS IoT Core. Then, AWS IoT Core forwards the request to AWS Lambda. Then, AWS Lambda reads the corresponding update image from Amazon S3 through MQTT, and then publishes the update image.

Then, the MCU performs Subscribe through MQTT to obtain the update image. The obtained update image is written to the buffer of the flash ROM on the MCU. When the update image is completely written to the flash ROM, the MCU begins to wait for a reset instruction from the PC. Before a reset instruction (RESET) is entered from the PC, switch the jumper connection according to the desired update image. When a reset instruction (RESET) is entered from the PC, the MCU swaps the banks, and then resets itself. After that, the update image (new program) starts operation.

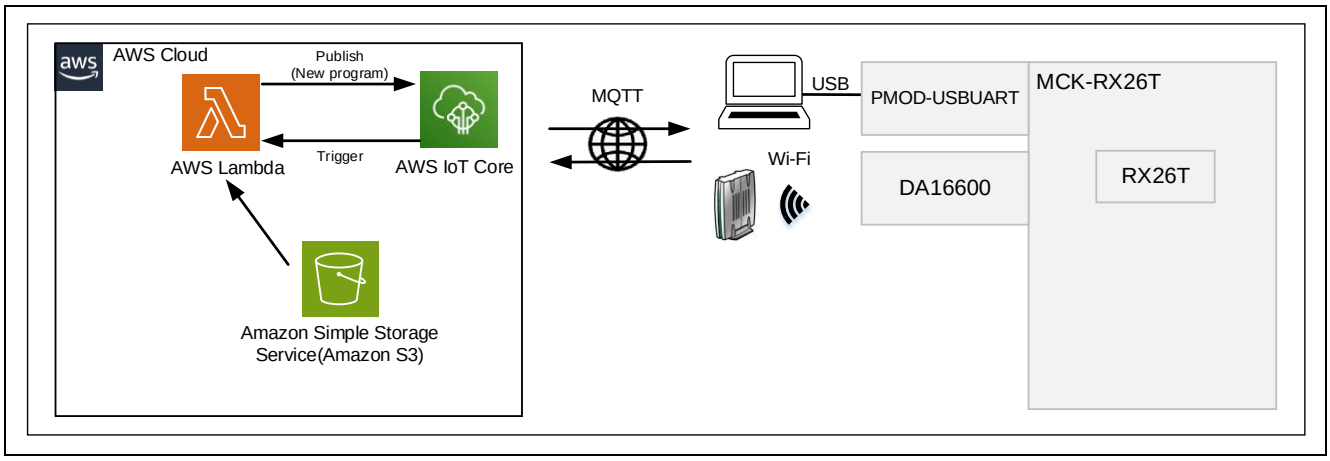

Figure 5-1 System Configuration Including AWS for Demonstration

5.2 Project Configuration

This sample program consists of three projects. The first project is a bootloader. This bootloader has a function that only starts the program in the main area. The other two projects are main programs. Each of these projects was created by adding an OTA update add-on function (sample_ota_add_on) to the project described in “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858). For details on configuration of the add-on function, refer to “5.5.9 Flowcharts of the Sample Program”.

These projects are designed to work in combination with a bootloader, and therefore cannot operate independently. For details on how to combine one of these projects with a bootloader, refer to “3.2 Initial and Update Images”. Generate the initial and update images by using Motorola S files generated by building the projects in “Table 5-1 List of Projects”.

Table 5-1 List of Projects

Project name	Description
r01an7203_rx26t_boot_loader	Bootloader
r01an7203_rx26t_fwup1	Main program Updates the firmware while performing the following processing: RX26T_MCBA_MCILV1_SPM_LESS_FOC_1SHUNT_E2S_V110
r01an7203_rx26t_fwup2	Main program Updates the firmware while performing the following processing: RX26T_MCBA_MCILV1_SPM_LESS_FOC_E2S_V110

5.3 Section Setup

In this sample program, section setup is already completed in each project so that a bootloader and main program can be used in combination. Sections have been set according to “6.2.2.1 Memory map of dual bank method demo project” in “RX Family Firmware Update Module Using Firmware Integration Technology” (R01AN6850).

5.4 Software Configuration

This section shows the software configuration of this sample program. This sample software handles two kinds of software: software related to the MCU (RX26T) and the software related to AWS.

5.4.1 Software Related to the MCU (RX26T)

“Figure 5-2 Configuration of the Software Related to the MCU (RX26T)” shows the configuration of the RX26T-related software.

The RX26T-related software consists of two major programs. One is the main program, which is a user program. The other is an OTA update program named “sample_ota_add_on”. The OTA update program has been created by using “RX Family Firmware Update Module Using Firmware Integration Technology” (R01AN6850), and uses AT commands to control the DA16600 Pmod™ board via serial communication interfaces (SCIs). The OTA update program is an add-on for the main program.

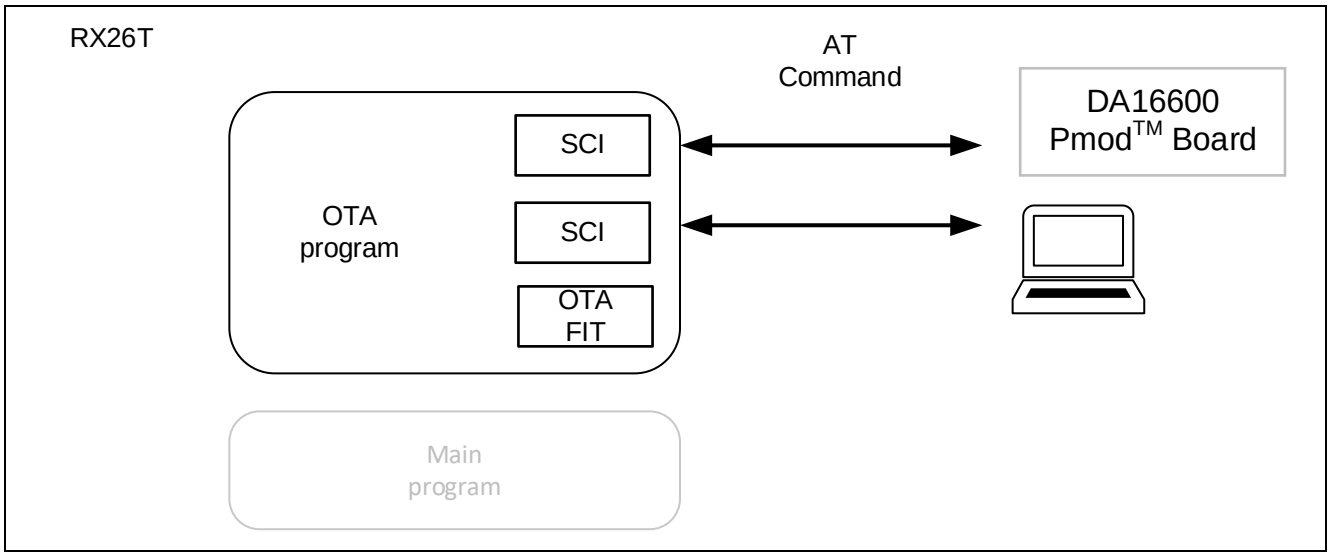

Figure 5-2 Configuration of the Software Related to the MCU (RX26T)

5.4.2 Configuration of the Software Related to the MCU (RX26T): Details of the File Organization

The OTA update program (sample_ota_add_on) consists of seven “.c” files and their corresponding “.h” files in pairs.

“Figure 5-3 Details of the Configuration of the Software Related to the MCU (RX26T)” shows the dependencies among these files.

- sample_ota_add_on.c/.h:
  These are top-level files that contain a function that is equivalent to the main routine of the OTA update program.
- sample_aws.c/.h:
  These files are used to control AWS. All AWS-related functions are defined in these files.
- sample_da16600_uart.c/.h:
  These files enable UART communication with the DA16600 board. RSCI11 is used as a hardware resource.
- sample_mqtt_config.c/.h:
  These files are used to manage AWS certificates, Wi-Fi SSIDs and passwords, and other information.
- base64_decode.c/.h
  These files are used to implement Base64 decoding.
- sample_fwup.c/h:
  These files use the firmware update FIT module to update the firmware.
- sample_pc_uart.c/.h:
  These files enable UART communication with the PC. RSCI8 is used as a hardware resource.

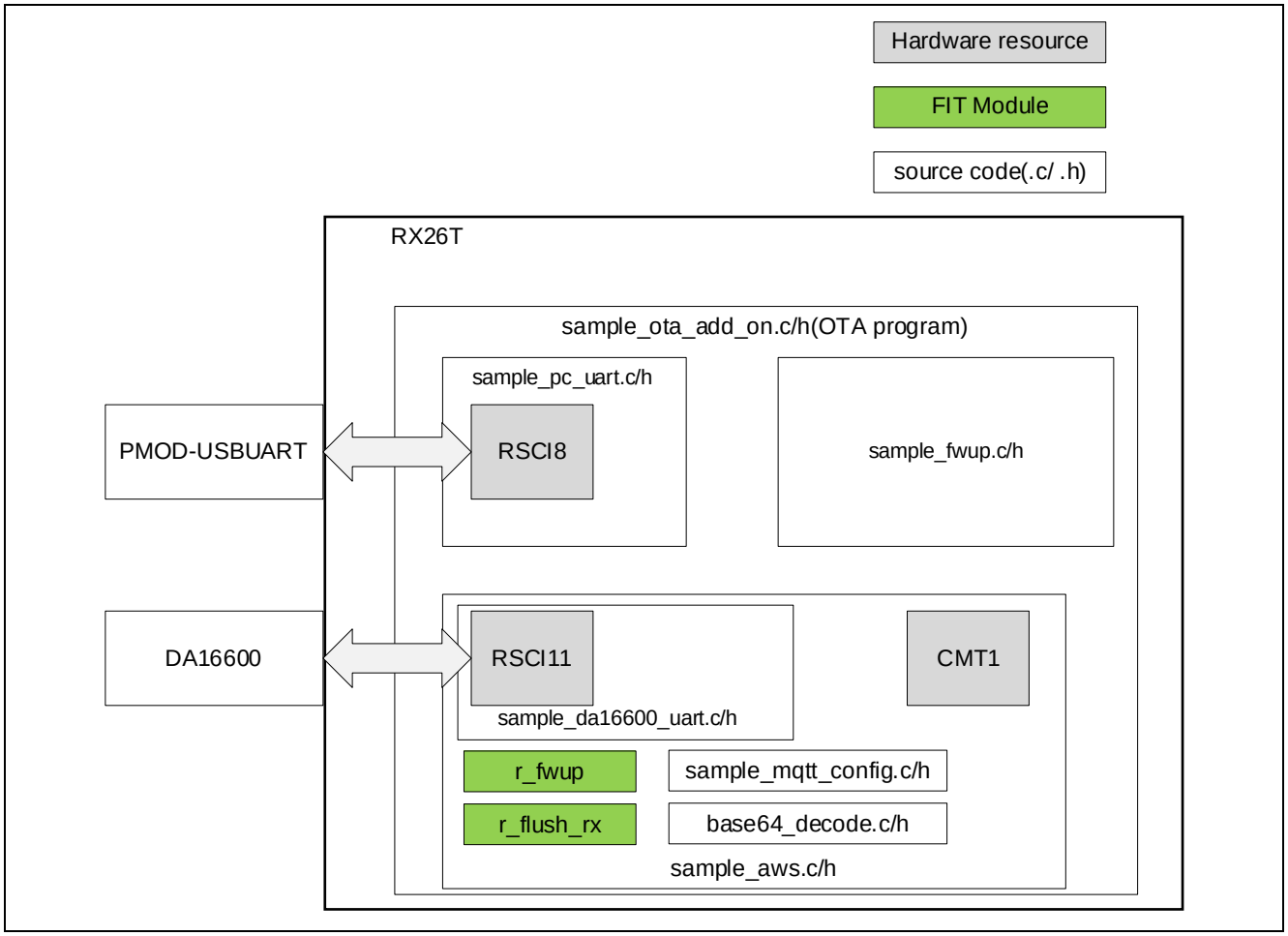

Figure 5-3 Details of the Configuration of the Software Related to the MCU (RX26T)

5.4.3 Software Related to AWS

The software related to AWS is configured by using AWS Lambda (Python 3.11). In AWS, the Boto3 library that can control Amazon S3 and AWS IoT Core is used to set up the AWS environment.

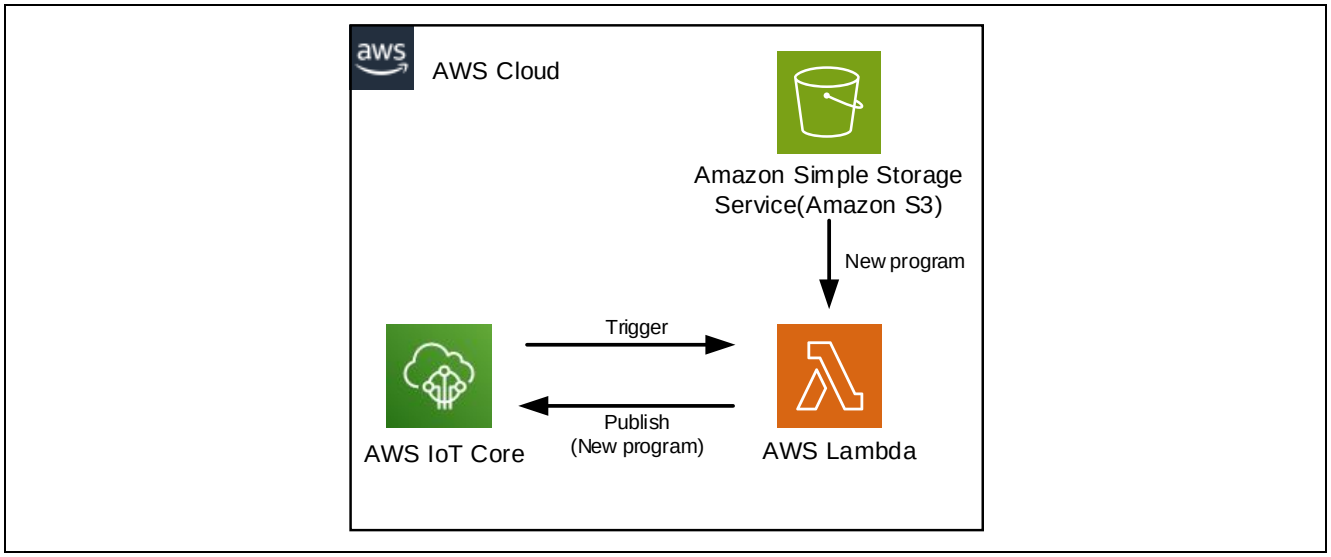

Figure 5-4 Configuration of the Software Related to AWS

5.5 Configuration of the Sample Program

5.5.1 Overview

This sample program is designed to work in combination with a bootloader as mentioned in “3.2 Initial and Update Images”. The firmware update functionality is implemented as an add-on for the main function of the sample program for “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

5.5.2 List of Pins Used

The following table lists the pins on the RX26T used by the add-on that provides the OTA update functionality for this sample program.

For details on configuration of the pins other than the pins used for an OTA update, refer to “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

Table 5-2 List of Pins Used

Pin name	Input or output	Purpose
PD5/TXD11	Output	Connection to GPIOC7_TXD_HOST on the DA16600 Pmod™ board
PB6/RXD11	Input	Connection to GPIOC6_RXD_HOST on the DA16600 Pmod™ board
PD0/TXD8	Output	Connection to RX on Pmod USBUART
PD1/RXD8	Input	Connection to TX on Pmod USBUART
VCC	—	Supplying 3.3 V to the DA16600 Pmod™ board
VSS	—	Connection to VSS on the DA16600 Pmod™ board

5.5.3 Peripheral Functions Used

This section shows the peripheral functions used by this sample program.

For details on configuration of the pins other than the pins used for an OTA update, refer to “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

Table 5-3 List of Peripheral Functions Used

Peripheral function	Purpose
RSCI11	UART communication with the DA16600 Pmod™ board
RSCI8	UART communication with the PC
CMT1	Interval control for MQTT communication

5.5.4 Peripheral Function Settings

The peripheral function settings used in this sample program were generated by using Smart Configurator (V2.19.0). The following tables show the settings specified by using Smart Configurator. Note that these tables show only the peripheral function settings added to the settings for the base sample program that was provided by the following application note: “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

Table 5-4 Settings for Config_RSCI11 (for Connection to the DA16600 Pmod™ Board)

Item	Setting
Serial communication method	Asynchronous mode
Start bit detection setting	Low level on the RXD11 pin
Data bit length	8 bits
Parity setting	Disabled
Stop bit setting	1 bit
Data transfer direction setting	LSB first
Transfer rate setting	• Transfer clock: Internal clock • Bit rate: 115,200 bps • Bit rate modulation function enabled • SCK11 pin function: Do not use the SCK11
Noise filter setting	Noise filter disabled
Hardware flow control setting	Hardware flow control setting: Disabled
Data processing setting	Transmit data processing: Interrupt service routine used Receive data processing: Interrupt service routine used
Interrupt setting	Receive error interrupt enabled Priority: Level 15
Callback function setting	Do not use
Input/output pins	• Output: TXD11 (PB5) • Input: RXD11 (PB6)

Table 5-5 Settings for Config_RSCI8 (for Connection to the PC)

Item	Setting
Serial communication method	Asynchronous mode
Start bit detection setting	Low level on the RXD8 pin
Data bit length	8 bits
Parity setting	Disabled
Stop bit setting	1 bit
Data transfer direction setting	LSB first
Transfer rate setting	• Transfer clock: Internal clock • Bit rate: 115,200 bps • Bit rate modulation function enabled • SCK8 pin function: Do not use the SCK8
Noise filter setting	Noise filter disabled
Hardware flow control setting	Hardware flow control setting: Disabled
Data processing setting	Transmit data processing: Interrupt service routine used Receive data processing: Interrupt service routine used
Interrupt setting	Receive error interrupt enabled Priority: Level 15
Callback function setting	Do not use
Input/output pins	• Output: TXD8 (PD0) • Input: RXD8 (PD1)

Table 5-6 Settings for r_flash_rx (FIT v5.10)

Item	Setting
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.

Table 5-7 Settings for r_fwup (FIT v2.01)

Item	Setting
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)

Table 5-8 Settings for CMT1

Item	Setting
Clock setting	PCLKB/32
Compare match setting	• Interval time: 10 ms • Compare match interrupt enabled (CMI1) • Priority: Level 15 (interrupt disabled)

Table 5-9 Settings for r_bsp

Item	Setting
Enable user studio charput function	Use user chaput function
User studio charput function name	my_sw_charput_function

Note: Other settings for r_bsp can be specified optionally.

5.5.5 File Organization

The following table shows the organization of the files of this sample program.

Table 5-10 File Organization

Folder and file names	Description
src	Program storage folder
├ app	Main processing
│ ├ board_ui (folder)	Definitions related to the board UI (refer to the application note R01AN6858)
│ ├ cfg (folder)	Configuration definitions for the application layer (refer to the application note R01AN6858)
│ ├ main	Program that processes table searches and linear interpolation
│ ├ └ main.c/h*1	User main function
│ └ rmw(folder)	Definition of functions related to the Analyzer UI of RMW (refer to the application note R01AN6858)
├ motor_module (folder)	Definitions related to the motor control module (refer to the application note R01AN6858)
├ QE_Motor	Files generated by QE for Motor (refer to the application note R01AN6858)
└ src	src folder
├ sample_ota_add_on*1	OTA-related program
│ ├ base64_decode.c/h*1	Base64 decoding
│ ├ sample_aws.c/h*1	AWS control program
│ ├ sample_da16600_uart.c/h*1	DA16600 control program
│ ├ sample_fwup.c/h*1	Firmware update control program
│ ├ sample_mqtt_config.c/h*1	MQTT connection settings
│ ├ sample_ota_add_on.c/h*1	Main program of ota_add_on
│ └ sample_pc_uart.c/h*1	File related to communication with the PC
└ smc_gen	Files generated by Smart Configurator
├ Config_CMT0	
├ Config_CMT1*1	
├ Config_IWDT	
├ Config_MTU3_MTU4	
├ Config_POE	
├ Config_PORT	
├ Config_RSCI8*1	
├ Config_RSCI11*1	
├ Config_S12AD0	
├ Config_S12AD2	
├ general	
├ r_bsp	
├ r_config	
├ r_flash_rx	
├ r_fwup*2	
└ r_pincfg	

Notes: 1. These are files that have been added to, or changed from, the base sample program that was provided by “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858). For details on the files and folders to which the *1 footnote number is not added, refer to the application note R01AN6858.

2. In the “r_fwup/src/r_fwup_wrap_flash.c” file, lines 129 and 155 have been commented out.

5.5.6 List of Variables

The following table lists the variables used in this sample program.

Note that this table shows only the variables added to the variables for the base sample program that was provided by the following application note: “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858). For details on the other variables, refer to the application note R01AN6858.

Table 5-11 List of Variables Used in the Sample Program

Variable name	Type	Description
g_sample_aws_code_buf[260]	uint8_t	Downloaded program code
g_sample_aws_code_size	uint16_t	Size of the downloaded code
g_sample_da16600_uart	struct	Status of communication with the DA16600
g_mqtt_root_certificate_pem	uint8_t	Root certificate
g_mqtt_certificate_pem_cert	uint8_t	Code signature certificate
g_mqtt_private_pem_key	uint8_t	Private key
g_mqtt_broker_endpoint	uint8_t	AWS endpoint
g_mqtt_broker_port	uint8_t	MQTT broker port number
g_mqtt_subscriber	uint8_t	AWS “thing” name
g_mqtt_publisher	uint8_t	MQTT topic
g_wifi_ssid	uint8_t	SSID of the access point
g_wifi_password	uint8_t	Password for the access point
g_root_certificate_pem_size	uint16_t	Size of the root certificate (in characters)
g_certificate_pem_cert_size	uint16_t	Size of the code signature certificate (in characters)
g_private_pem_key_size	uint16_t	Size of the private key (in characters)
g_broker_endpoint_size	uint16_t	Size of the AWS endpoint (in characters)
g_broker_port_size	uint16_t	Size of the MQTT broker port number (in characters)
g_subscriber_size	uint16_t	Size of the AWS “thing” name (in characters)
g_publisher_size	uint16_t	Size of the MQTT topic (in characters)
g_wifi_size	uint16_t	Size of the SSID of the access point (in characters)
g_password_size	uint16_t	Size of the password for the access point (in characters)
g_sample_pc_uart	struct	Status of communication with the PC

5.5.6.1 List of Constants

There are no constants that have been added to the base sample program provided by the following application note: “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858). For details on the constants, refer to the application note R01AN6858.

5.5.7 List of Global Functions

The following table lists the global functions used in this sample program.
Note that this table shows only the functions added to, or changed from, the base sample program that was provided by the following application note: “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

Table 5-12 List of Global Functions Used in the Sample Program

Function name	Summary	
main	Function created by adding a call to “sample_ota_add_on()” to the main loop of the main function of the base sample program	Changed
base64_decode	Performs Base64 decoding.	Added
sample_aws_send_clieant_credential	Sets a certificate for the DA16600.	Added
sample_aws_mqtt_start	Establishes a connection for MQTT communication with AWS.	Added
sample_aws_cmt_callback	Callback function for the CMT	Added
sample_aws_init	Specifies the initial settings of sample_aws.	Added
sample_aws_publish_size	Sends an inquiry to AWS about the size of the update image.	Added
sample_aws_subscribe_size	Receives the size of the update image from AWS.	Added
sample_aws_publish_rsu	Requests AWS to return the update image.	Added
sample_aws_subscribe_rsu	Receives the update image from AWS.	Added
sample_aws_init_state	Initializes the AWS processing state.	Added
sample_da16600_uart_send_at_command	Sends AT commands to the DA16600.	Added
sample_da16600_uart_init	Opens or starts the RSCI11.	Added
sample_da16600_uart_read_buf	Transfers the content of the internal ring buffer to the read buffer.	Added
sample_da16600_uart_recv_callback	Performs a callback when reception over RSCI11 is completed	Added
sample_da16600_uart_analyze_buf	Analyzes the data received from the DA16600.	Added
sample_fwup_init	Opens the firmware update FIT module.	Added
sample_fwup_bunkswap	Swaps the banks.	Added
sample_write_image	Performs self-programming in the buffer.	Added
sample_ota_add_on	Main program for an OTA update	Added
sample_pc_uart_init	Opens or starts the RSCI8.	Added
sample_pc_uart_read_buf	Transfers the content of the internal ring buffer to the read buffer.	Added
sample_pc_uart_recv_callback	Performs a callback when reception over RSCI8 is completed.	Added

5.5.8 Specifications of Functions

This section describes the specifications of each function used in this sample program.

Function name: main

Summary	Main processing
Header	None
Declaration	void main (void)
Description	This function was created by adding a call to “sample_ota_add_on()” to the main loop of the main function of the base sample program. The “sample_ota_add_on()” function communicates with the PC, controls the DA16600 Pmod™ board, and performs an OTA update.
Argument	None
Return value	None
Remarks	None

Function name: base64_decode

Summary	Function that performs Base64 decoding
Header	base64_decode.h
Declaration	uint32_t base64_decode(uint8_t *source, uint8_t *result, uint32_t size);
Description	This function performs Base64 decoding.
Argument	uint8_t * source: Address of the data to be decoded uint8_t * result: Address of the decoded data uint32_t size: Size of the decoded data
Return value	Size of the decoding result data (in bytes)
Remarks	None

Function name: sample_aws_send_clieant_credential

Summary	Function that sends a root certificate, device certificate, and private key
Header	sample_aws.h
Declaration	void sample_aws_send_clieant_credential (uint8_t * st);
Description	This function uses AT commands to send certificates to the DA16600 Pmod™ board.
Argument	uint8_t * st: Specifies the address of “ota_st”.
Return value	None
Remarks	None

Function name: sample_aws_mqtt_start

Summary	Function that establishes connections to a Wi-Fi module and AWS.
Header	sample_aws.h
Declaration	void sample_aws_mqtt_start (uint8_t * st);
Description	This function controls the DA16600 to establish a Wi-Fi connection. After establishing a Wi-Fi connection, it establishes a connection to AWS through MQTT.
Argument	uint8_t * st: Specifies the address of “ota_st”.
Return value	None
Remarks	None

Function name: sample_aws_cmt_callback

Summary	Callback function for the CMT1
Header	sample_aws.h
Declaration	void sample_aws_cmt_callback (void);
Description	This function increments the value of the “millis_cnt” internal variable by 10 each time a compare match succeeds.
Argument	None
Return value	None
Remarks	None

Function name: sample_aws_init

Summary	Initialization function
Header	sample_aws.h
Declaration	void sample_aws_init (void);
Description	This function opens or starts the CMT1.
Argument	None
Return value	None
Remarks	None

Function name: sample_aws_publish_size

Summary	Function that sends an inquiry to AWS about the size of the update image
Header	sample_aws.h
Declaration	void sample_aws_publish_size (uint8_t * upper_st, uint8_t no);
Description	This function sends an inquiry to AWS about the size of the update image.
Argument	uint8_t * upper_st: Specifies the address of “ota_st”. uint8_t * no: Specifies the update image number (1 or 2).
Return value	None
Remarks	None

Function name: sample_aws_subscribe_size

Summary	Function that receives the size of the update image from AWS
Header	sample_aws.h
Declaration	void sample_aws_subscribe_size (uint8_t * upper_st);
Description	This function receives the size of the update image from AWS.
Argument	None
Return value	None
Remarks	None

Function name: sample_aws_publish_rsu

Summary	Function that requests AWS to return the update image.
Header	sample_aws.h
Declaration	void sample_aws_publish_rsu (uint8_t * upper_st, uint8_t no);
Description	This function requests AWS to return the update image.
Argument	uint8_t * upper_st: Specifies the address of “ota_st”. uint8_t * no: Specifies the update image number (1 or 2).
Return value	None
Remarks	None

Function name: sample_aws_subscribe_rsu

Summary	Function that receives the update image from AWS
Header	sample_aws.h
Declaration	sample_aws_subscribe_rsu (uint8_t * upper_st, uint8_t * success);
Description	This function receives the update image from AWS.
Argument	uint8_t * upper_st: Specifies the address of “ota_st”. uint8_t * no: Specifies the update image number (1 or 2).
Return value	None
Remarks	None

Function name: sample_aws_init_state

Summary	Function that initializes the AWS processing state.
Header	sample_aws.h
Declaration	void sample_aws_init_state (void);
Description	This function initializes the AWS processing state.
Argument	None
Return value	None
Remarks	None

Function name: sample_da16600_uart_send_at_command

Summary	Function that sends AT commands to the DA16600
Header	sample_da16600_uart.h
Declaration	void sample_da16600_uart_send_at_command (uint8_t * const com, uint16_t len);
Description	This function sends AT commands to the DA16600.
Argument	uint8_t * const com: Specifies the address of the AT command to be sent. uint16_t len: Specifies the length of the AT command to be sent.
Return value	None
Remarks	None

Function name: sample_da16600_uart_init

Summary	Function that opens or starts the RSCI11
Header	sample_da16600_uart.h
Declaration	void sample_da16600_uart_init (void);
Description	This function opens or starts the RSCI11.
Argument	None
Return value	None
Remarks	None

Function name: sample_da16600_uart_read_buf

Summary	Function that sets receive data in the read buffer
Header	sample_da16600_uart.h
Declaration	void sample_da16600_uart_read_buf (void);
Description	This function transfers data from the internal ring buffer to the read buffer.
Argument	None
Return value	None
Remarks	None

Function name: sample_da16600_uart_recv_callback

Summary	Function that performs a callback when reception over RSCI11 is completed
Header	sample_da16600_uart.h
Declaration	void sample_da16600_uart_read_buf (void);
Description	This function copies the data received from the DA16600 to the internal ring buffer.
Argument	None
Return value	None
Remarks	None

Function name: sample_da16600_uart_analyze_buf

Summary	Function that analyzes the data received from the DA16600
Header	sample_da16600_uart.h
Declaration	uint8_t sample_da16600_uart_analyze_buf(uint8_t * p_resp)
Description	This function analyzes the data received from the DA16600.
Argument	uint8_t * p_resp: Specifies the type of the response to be analyzed.
Return value	uint8_t: 0 = The specified response does not exist; 1 = The specified response exists.
Remarks	None

Function name: sample_fwup_init

Summary	Function that opens the firmware update FIT module
Header	sample_fwup.h
Declaration	void sample_fwup_init (void);
Description	This function opens the firmware update FIT module.
Argument	None
Return value	None
Remarks	None

Function name: sample_fwup_bunkswap

Summary	Function that swaps the banks
Header	sample_fwup.h
Declaration	void sample_fwup_bunkswap(void)
Description	This function calls “R_FWUP_ActivateImage” of the firmware update FIT module, performs a self-reset, and starts the update image (new update program).
Argument	None
Return value	None
Remarks	None

Function name: sample_write_image

Summary	Function that writes data to the flash ROM
Header	sample_fwup.h
Declaration	e_fwup_err_t sample_write_image(e_fwup_area_t area, uint8_t * rsu, uint16_t size)
Description	This function calls “R_FWUP_WriteImage” of the firmware update FIT module to perform self-programming.
Argument	e_fwup_area_t area: Specifies the update-target area (only the buffer area can be specified). uint8_t * rsu: Specifies the address of the area that stores the update image. uint16_t size: Specifies the size of the update image (in units of 256 bytes).
Return value	None
Remarks	None

Function name: sample_ota_add_on

Summary	Main routine of the OTA update program
Header	sample_ota_add_on.h
Declaration	void sample_ota_add_on (void);
Description	This function is the main routine of an OTA update. An OTA update is achieved during execution of the main program by making repeated calls to this function from the main() function.
Argument	None
Return value	None
Remarks	None

Function name: sample_pc_uart_init

Summary	Function that opens or starts the RSCI8
Header	sample_pc_uart.h
Declaration	void sample_pc_uart_init (void);
Description	This function opens or starts the RSCI8.
Argument	None
Return value	None
Remarks	None

Function name: sample_pc_uart_read_buf

Summary	Function that sets receive data in the read buffer
Header	sample_pc_uart.h
Declaration	void sample_pc_uart_read_buf (void);
Description	This function transfers data from the internal ring buffer to the read buffer.
Argument	None
Return value	None
Remarks	None

Function name: sample_pc_uart_recv_callback

Summary	Function that performs a callback when reception over RSCI8 is completed
Header	sample_da16600_uart.h
Declaration	void sample_pc_uart_read_buf (void);
Description	This function copies the data received from the PC to the internal ring buffer.
Argument	None
Return value	None
Remarks	None

5.5.9 Flowcharts of the Sample Program

The following shows the flowchart of the “main” function.

This function was created by adding the main routine of the OTA update program (sample_ota_add_on) to the main function of the base sample program provided by the following application note: “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

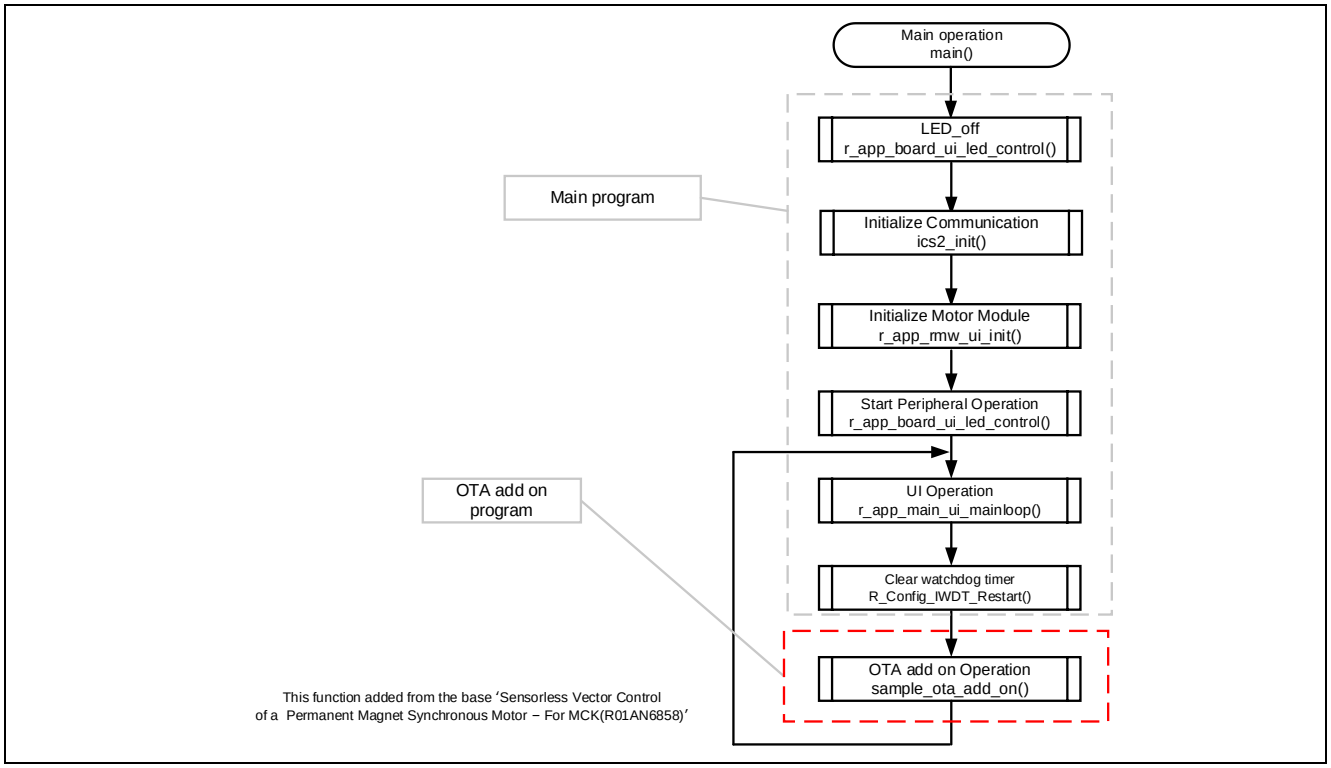

Figure 5-5 Flowchart of the “main” Function

The following shows the flowchart of the “sample_ota_add_on” function. The “sample_ota_add_on” function is the main routine of the OTA update program. This function calls the following four static functions: “ota_init”, “pc_comm”, “aws_comm”, and “ota_ope”.

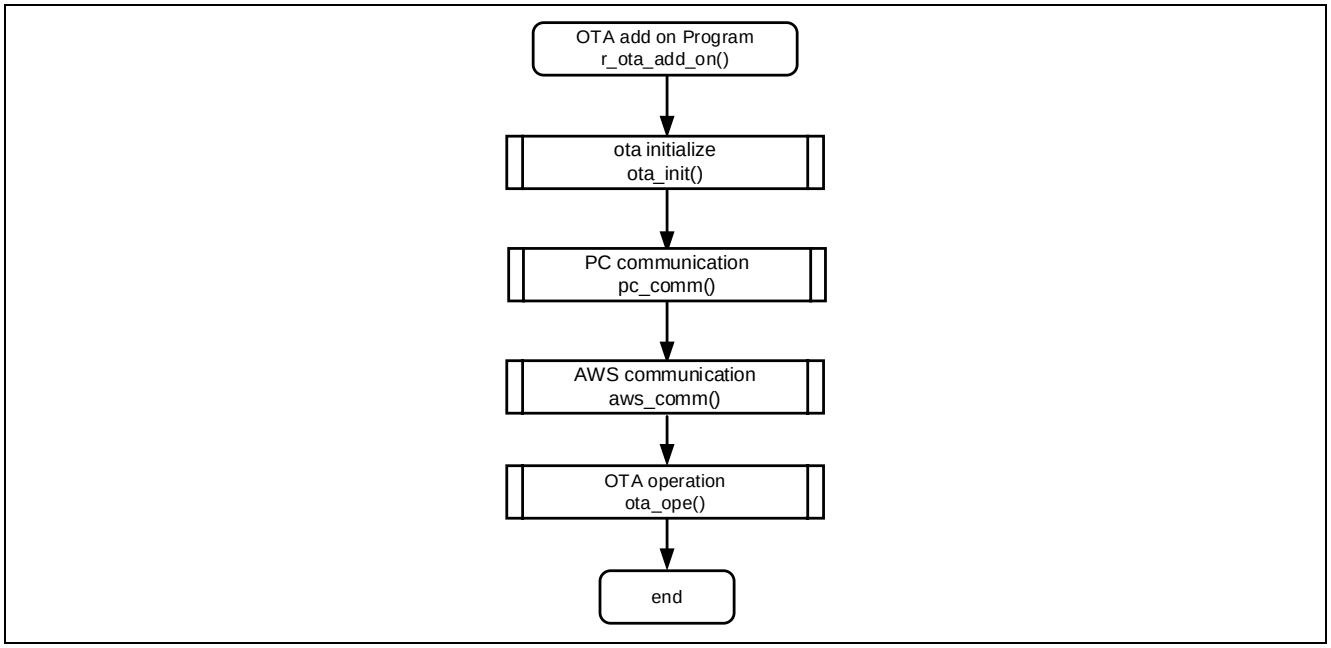

Figure 5-6 Flowchart of the “sample_ota_add_on” Function

The following shows the flowchart of the “ota_init” function. The “ota_init” function establishes a connection to AWS through MQTT and sends certificates. Immediately after a reset is performed (“ota_st” = 0), this function calls the “send_client_credential” function to send certificates and other credentials, and then sets “ota_st” to 1. When “ota_st” is 1, this function calls the “mqtt_start” function to establish a connection to MQTT. When the connection is established, “ota_st” is set to 2.

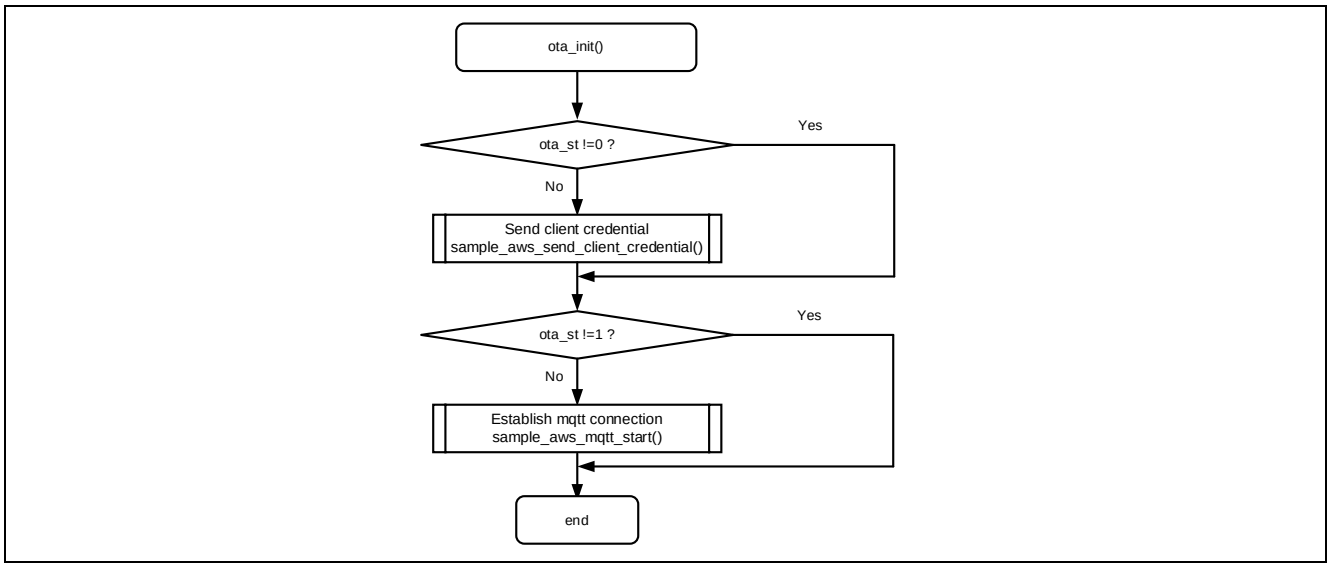

Figure 5-7 Flowchart of the “ota_init” Function

The following shows the flowchart of the “send_client_credential” function. The “send_client_credential” function sends a root certificate, device certificate, and a private key to AWS.

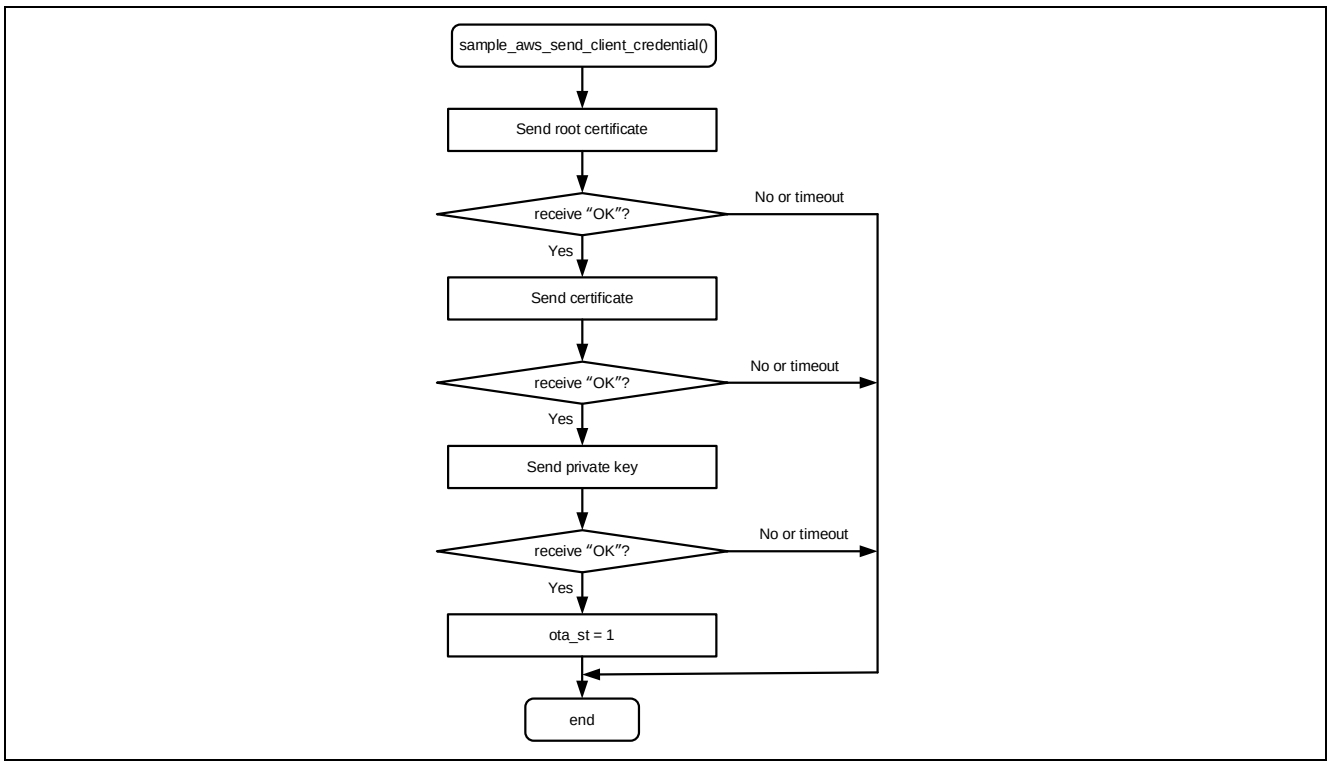

Figure 5-8 Flowchart of the “send_client_credential” Function

The following shows the flowchart of the “mqtt_start” function. The “mqtt_start” function establishes a connection to AWS through MQTT.

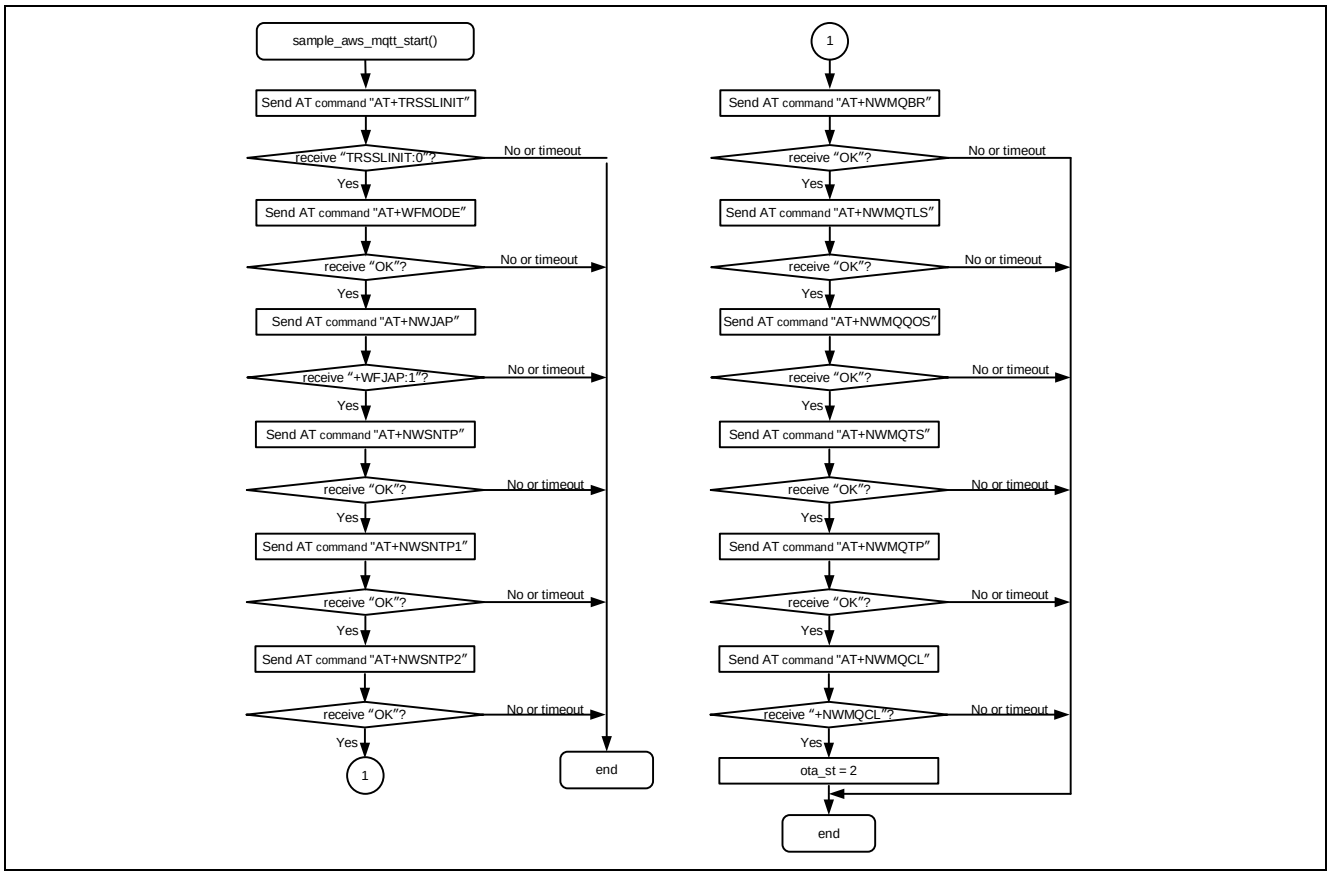

Figure 5-9 Flowchart of the “mqtt_start” Function

The following shows the flowchart of the “pc_comm” function. The “pc_comm” function receives instructions to start an OTA update and a reset from the PC.

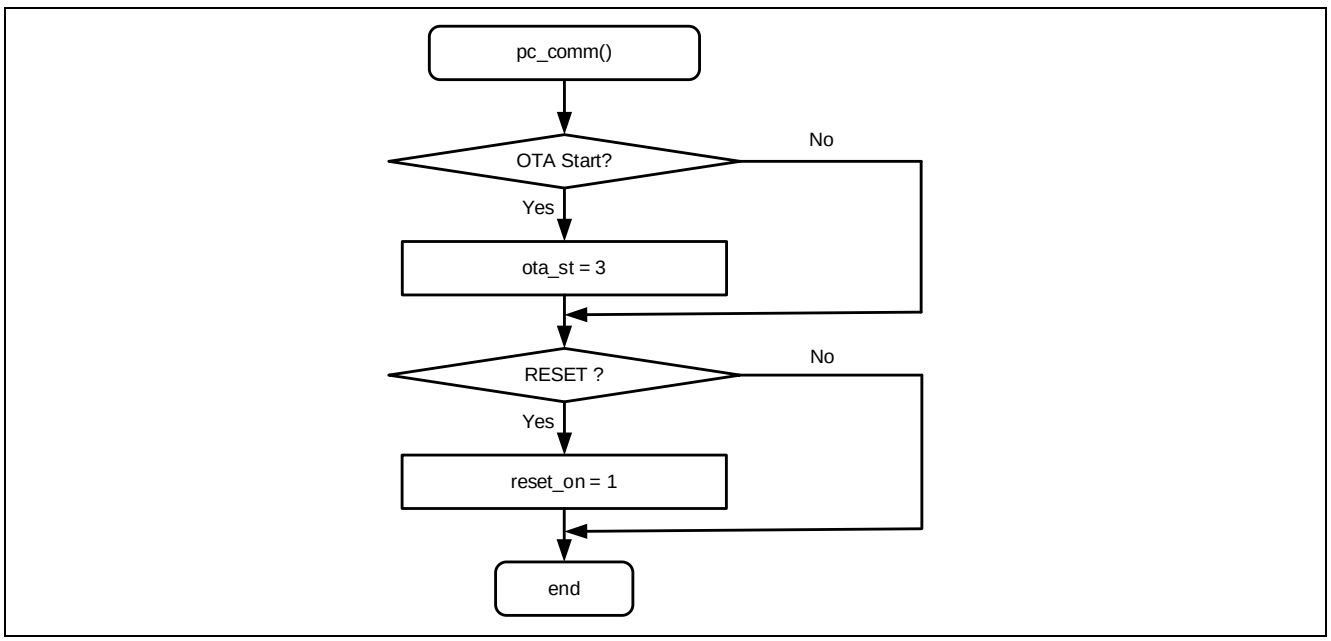

Figure 5-10 Flowchart of the “pc_comm” Function

The following shows the flowchart of the “aws_comm” function. The “aws_comm” function downloads an update image. When “ota_st” is 3, this function calls the “publish_size” function to send an inquiry about the size of the update image to be downloaded, and then sets “ota_st” to 4. When “ota_st” is 4, this function calls the “subscribe_size” function to receive the size of the update image to be downloaded. When the size is received, “ota_st” is set to 5. When “ota_st” is 5, this function calls the “publish_rsu” function to send a request to download the update image, and then sets “ota_st” to 6. When “ota_st” is 6, this function calls the “subscribe_rsu” function to receive the update image. Note that only 256 bytes can be downloaded at one time, the value of “ota_st” toggles between 5 and 6 until all data of the update image is downloaded.

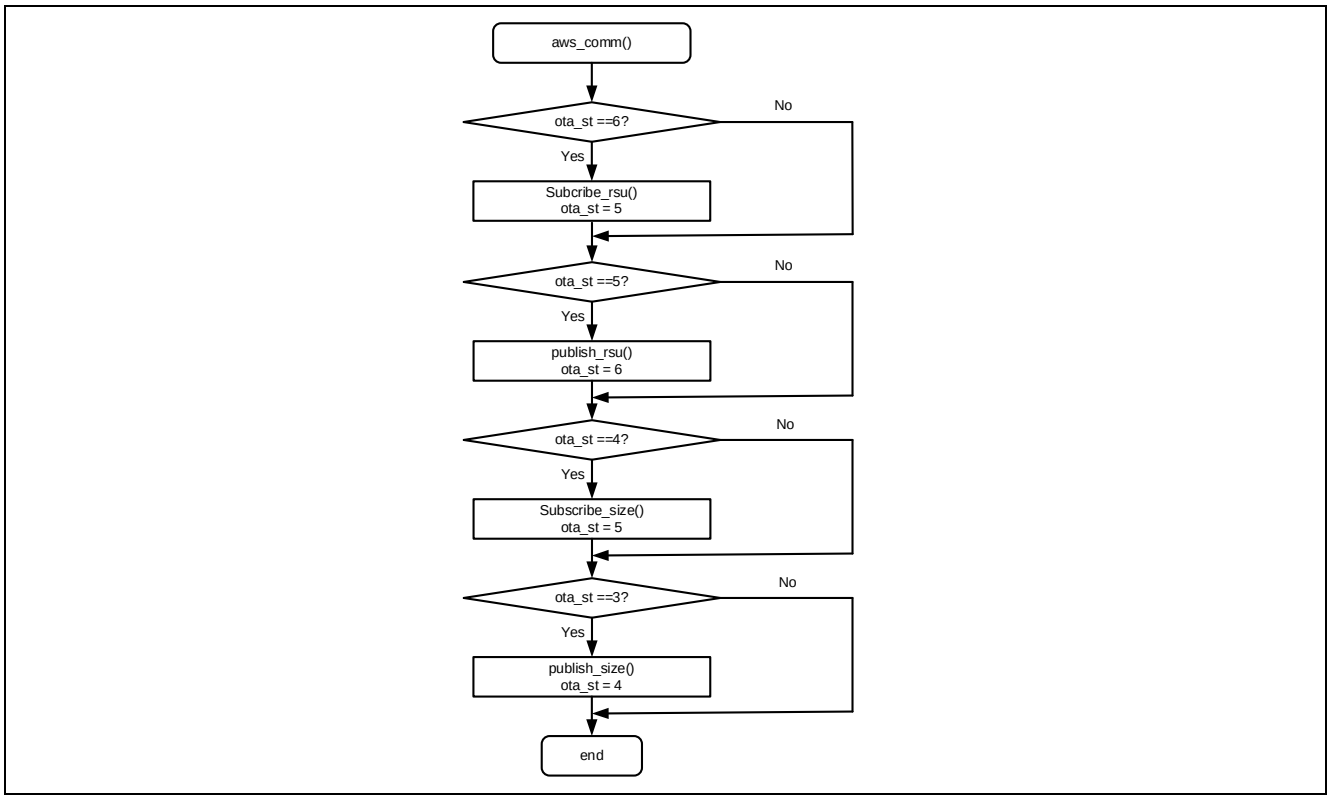

Figure 5-11 Flowchart of the “aws_comm” Function

The following shows the flowchart of the “ota_ope” function.

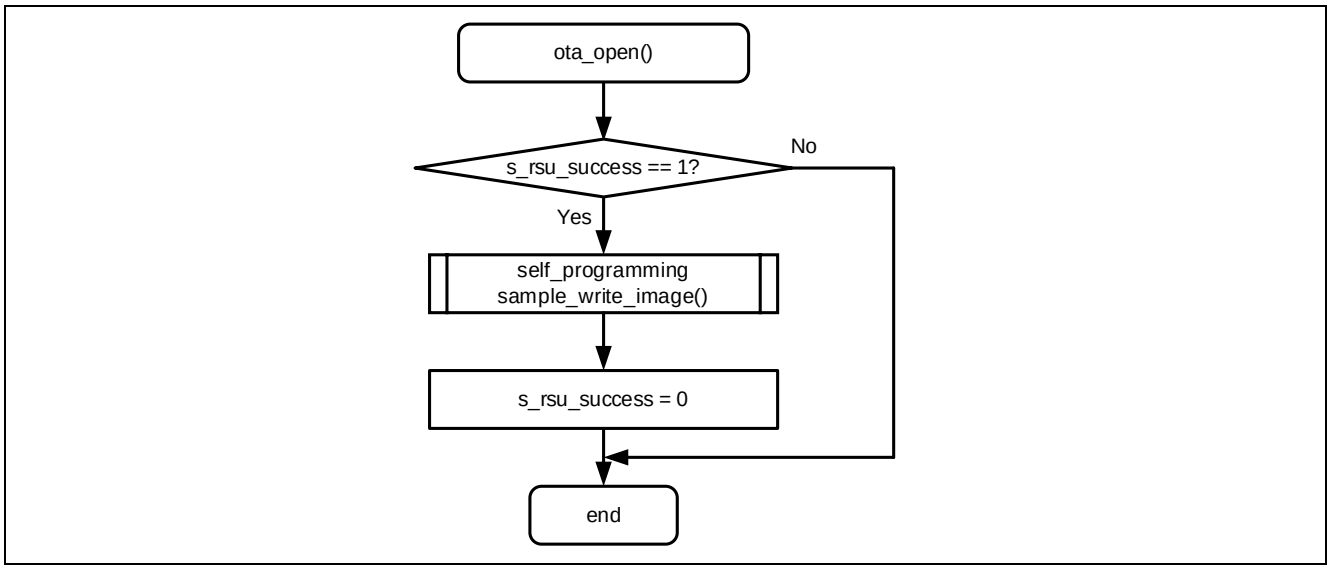

Figure 5-12 Flowchart of the “ota_ope” Function

5.5.10 Bootloader

This sample program uses the bootloader for dual bank mode as described in “RX Family Firmware Update Module Using Firmware Integration Technology” (R01AN6850). This bootloader has only a function that starts the main program. For the addresses of the bootloader and main program, refer to the sections that describe the settings for the r_fwup modules in “5.5.4 Peripheral Function Settings”.

5.6 Hardware Preparation

Connect the MCK-RX26T, inverter board, and a brushless DC motor as shown in “2.2 Board Connections”. Also, connect the DA16600 Pmod™ board to the CN12 (PMOD Type 3A module connector) on the MCK-RX26T.

5.6.1 Connecting a Pmod USBUART

Connect a Pmod USBUART to the CN10 (PMOD Type 6A module connector) on the MCK-RX26T.

As shown in the following figure, connect RXD, TXD, and GND of the Pmod USBUART to pins 9, 8, and 11 of the CN10, respectively. For the pin layout of the CN10, refer to the silkscreen marking inside the red frame in the following figure. Note that pins 7 to 12 of the CN10 are located on the lower row.

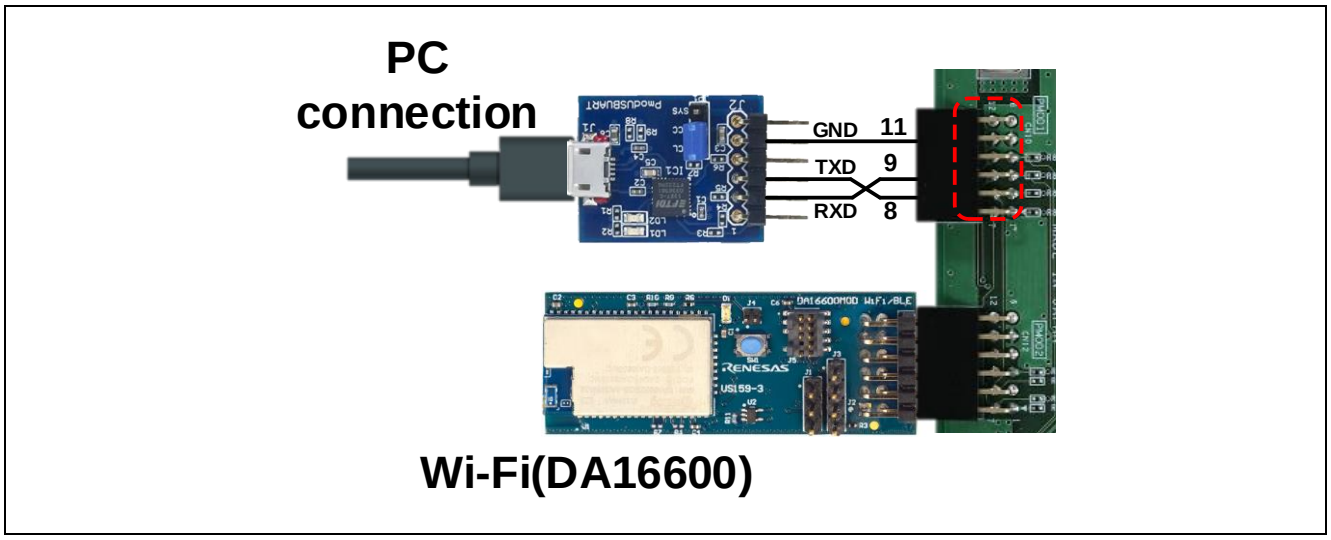

Figure 5-13 Connecting a Pmod USBUART

5.7 Preparation for the Terminal Software for Viewing a Log

This sample program outputs log data through the Pmod USBUART. You can view the log by using terminal software such as Tera Term. This application note describes the setup procedure in the case when you use Tera Term.

5.7.1 Starting Tera Term

The log data output through the Pmod USBUART can be viewed by using terminal software such as Tera Term. Prepare a PC on which Tera Term is installed, and connect the PC to the Pmod USBUART by USB. Then, start Tera Term. Select the [Serial] radio button, and select the COM port to which the Pmod USBUART is connected.

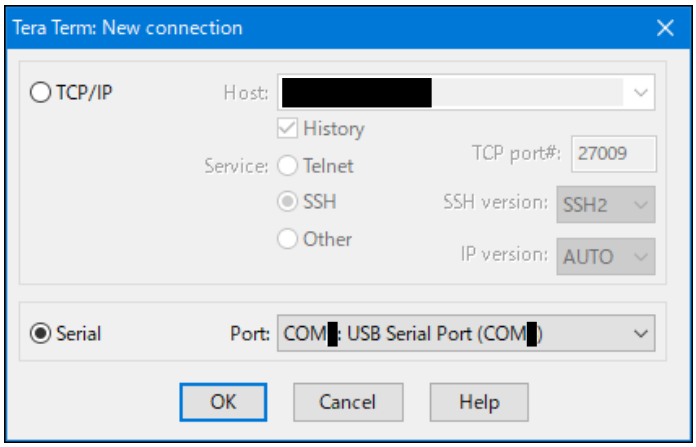

Figure 5-14 Starting Tera Term

5.7.2 Tera Term Terminal Setup

In the Tera Term window, select [Setup] → [Terminal]. The [Terminal setup] dialog box opens. Specify the settings as shown in the following figure.

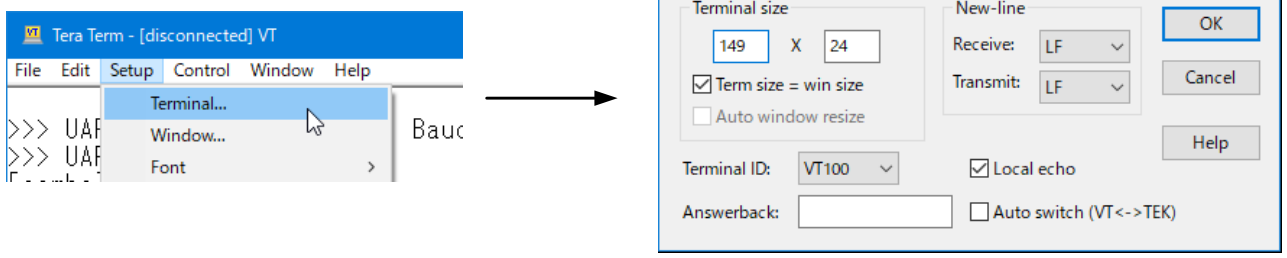

Figure 5-15 [Terminal setup] Dialog Box of Tera Term

5.7.3 Tera Term Serial Port Setup

In the Tera Term window, select [Setup] → [Serial port setup and connection]. The [Serial port setup and connection] dialog box opens. Note that the baud rate must be set according to the Pmod USBUART. Specify the settings as shown in the following figure.

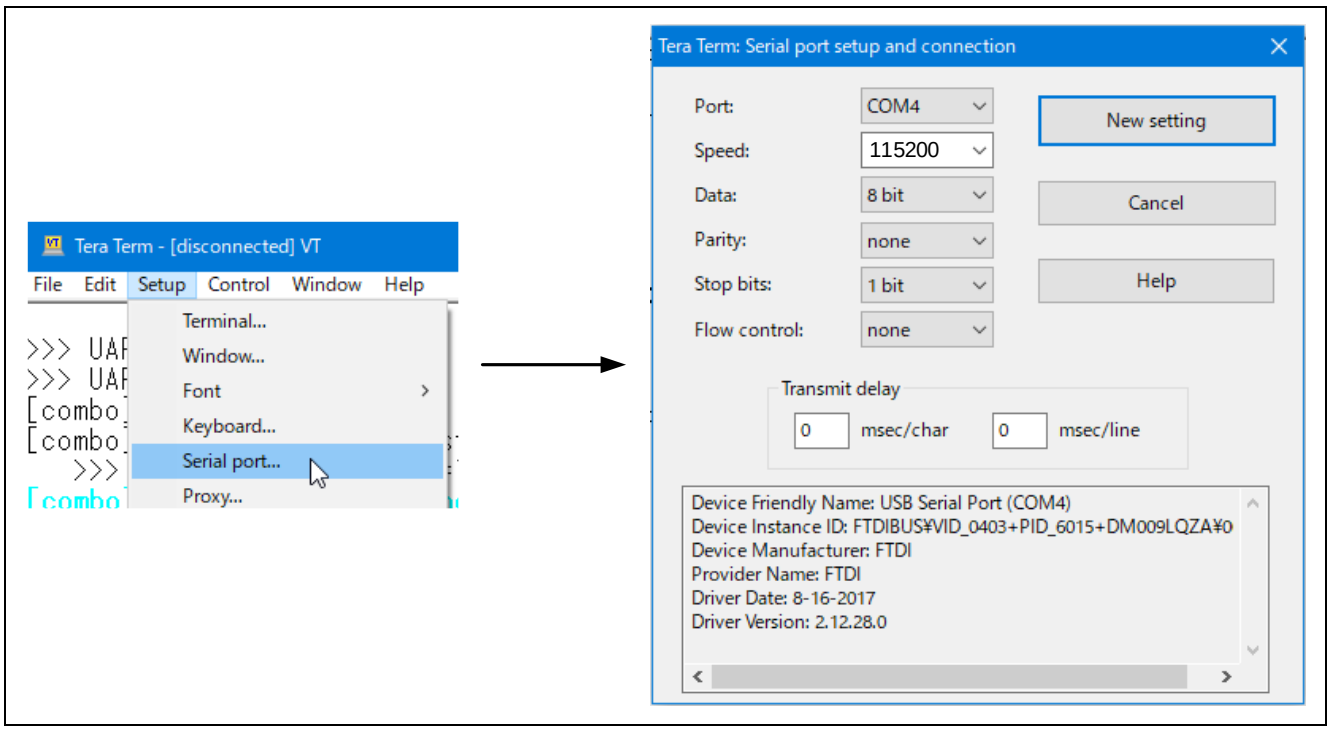

Figure 5-16 [Serial port setup and connection] Dialog Box of Tera Term

5.8 Overview of the Operation of the Sample Program

This section provides an overview of the operation of the sample program. For details on the procedure for executing the sample program, refer to “5.9 Details of the Operation of the Sample Program”.

(1) Power On:
Renesas Flash Programmer is used to write the initial image and start the program.
(2) Initialization:
The MCK-RX26T is automatically initialized.
(3) Storage of Information:
The MCK-RX26T automatically places the DA16600 Pmod™ board in STA (Station) mode. The MCK-RX26T also automatically writes credentials (certificates and a private key) to the DA16600 Pmod™ board.
(4) Wi-Fi Connection:
After the MCK-RX26T writes the credentials, it automatically requests the DA16600 Pmod™ board to establish a Wi-Fi connection.
(5) AWS Connection:
The MCK-RX26T confirms that a Wi-Fi connection is established, and then requests the DA16600 Pmod™ board to establish a connection to AWS.
(6) MQTT Connection:
The MCK-RX26T confirms that a connection to AWS is established, and then requests the DA16600 Pmod™ board to establish a connection to MQTT.

When the above procedure finishes, preparations for an OTA update are completed, and the “OTA ready” message appears on the PC. The MCK-RX26T issues a subscribe request to subscribe MQTT topics. An OTA update starts when an update instruction (FWUP1 or FWUP2) is entered from the PC.

While the above procedure is being performed, the main program operates without interruption. The main program is the sample program of the following application note: “RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK” (R01AN6858).

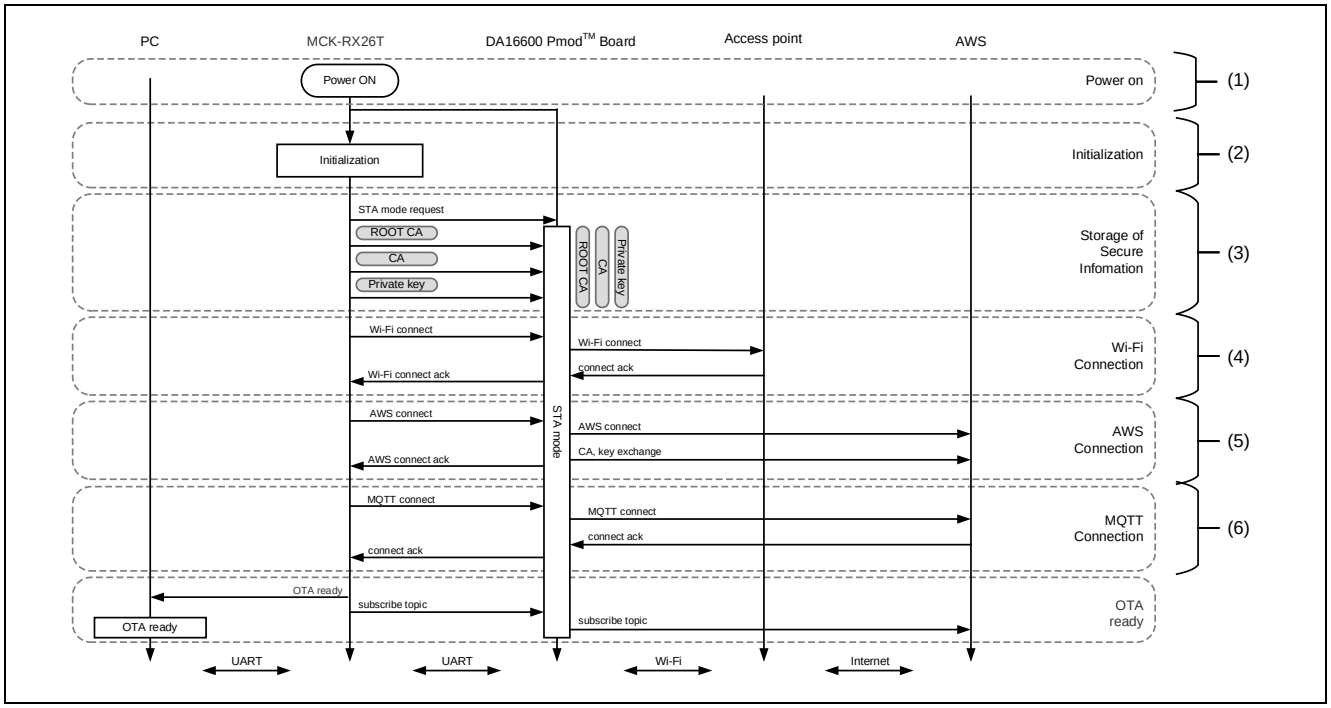

Figure 5-17 Overview of the Operation of the Sample Program

5.9 Details of the Operation of the Sample Program

This section describes the procedure for executing the sample program. By using the procedure, you can demonstrate a firmware update.

Note that the DA16600 Pmod™ board holds some settings on an NVRAM. If the board was previously used for another demonstration, the previous demonstration may be reproduced. In this case, perform power-off and power-on operations several times. Then, the conditions for the demonstration in this application note are written to the NVRAM on the DA16600. If the board does not operate as expected even after repeated power-off and power-on operations, perform a factory reset for the DA16600 Pmod™ board.

Procedure for Executing the Sample Program

(1) Use Renesas Flash Programmer to write the initial image (described in “3.2 Initial and Update Images”) to the RX26T. Before writing a program, make sure that the JP11 jumper pin on the CPU board is open. After writing the program and closing the JP11, when power is supplied from the USB port, the power lamp (LED5) is turned on.

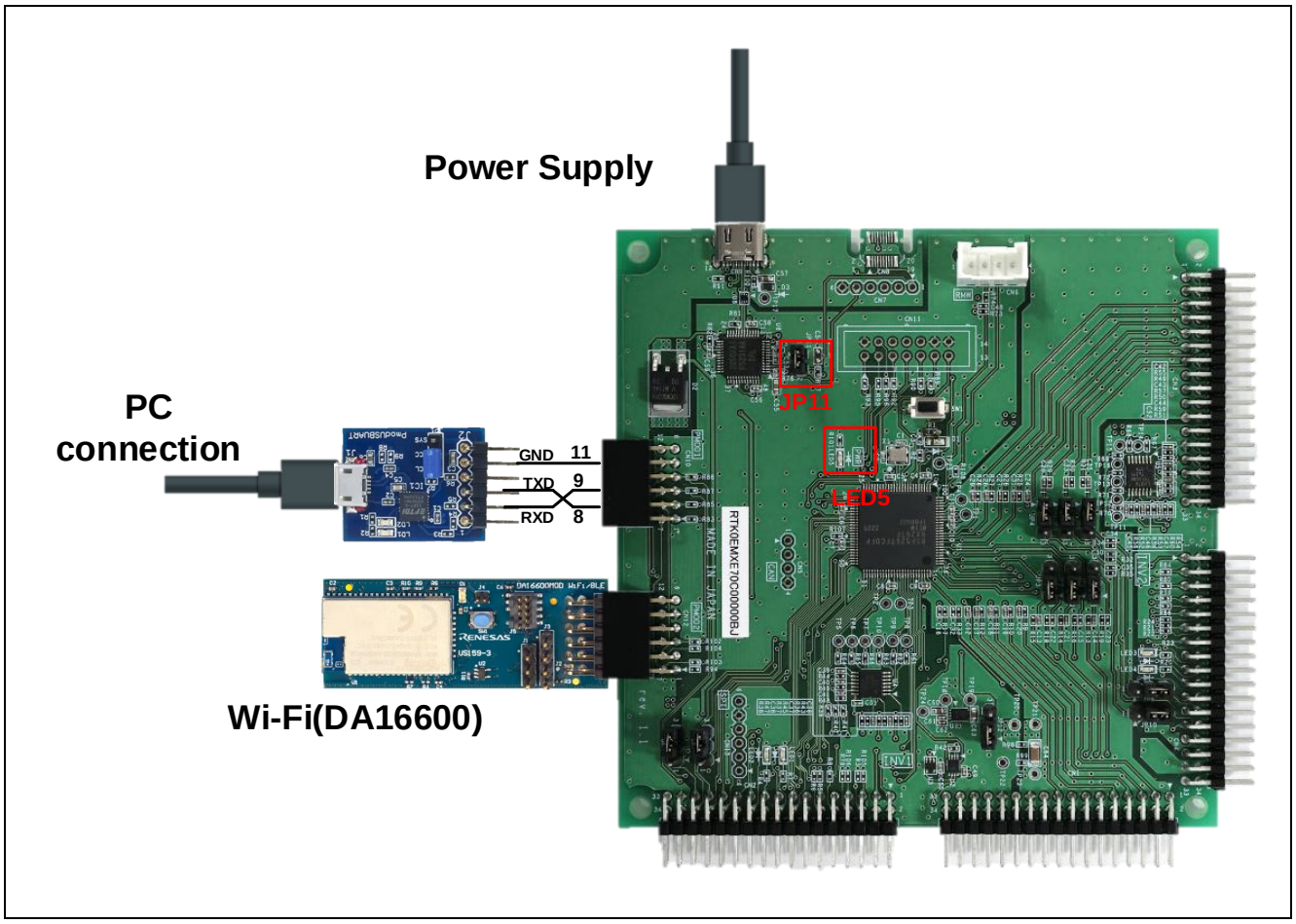

Figure 5-18 Location of the Power Lamp (LED5)

(2) Initialization:
The MCK-RX26T is automatically initialized.
When this step is completed, the MCK-RX26T automatically performs steps (3) to (6). After that, an OTA update becomes ready in step (7).

(3) Storage of Information:
The MCK-RX26T automatically places the DA16600 Pmod™ board in STA mode.
Then, the MCK-RX26T automatically writes certificates and a private key to the DA16600 Pmod™ board.

```
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↓
```

Figure 5-19 Information Logged in Step (3)

(4) Wi-Fi Connection:
The MCK-RX26T automatically requests the DA16600 Pmod™ board to establish a Wi-Fi connection.

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

Figure 5-20 Information Logged in Step (4)

(5) The MCK-RX26T automatically requests the DA16600 Pmod™ board to establish a connection to AWS.

(6) The MCK-RX26T automatically requests the DA16600 Pmod™ board to establish a connection to MQTT.

(7) OTA Ready:
When a connection to AWS is established and an OTA update is ready, the “OTA ready” message appears on the PC.

```
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)
```

Figure 5-21 Information Logged in Steps (5) and (6)

When the above procedure finishes, preparations for an OTA update are completed, and the “OTA ready” message appears. If the procedure does not proceed to step (5), the state described in “7.1.3 Troubleshooting a Problem That Displays the “Failed to establish tls-sess (0x7200)” Message” may apply.

(8) Publish Size:
When an OTA update start instruction (FWUP1 or FWUP2 followed by a carriage return) is entered from the PC, the MCK-RX26T calls the sample_aws_publish_size function to publish a request to AWS in order to make an inquiry about the size of the update image (*.rsu) corresponding to the entered instruction.

(9) Subscribe Size:
When the request is accepted and the size of the update image is published in AWS (Lambda), AWS returns the size through an already subscribed topic (this processing is not logged).

(10) Publish rsu:
The MCK-RX26T calls the sample_aws_publish_rsu function to publish a request to AWS in order to issue a request to download the update image from AWS.

(11) Subscribe rsu:
When the request is accepted and the update image is published in AWS (Lambda), AWS returns the update image through an already subscribed topic (this processing is not logged). Steps (10) and (11) are repeated some times because the data of the update image is downloaded in units of 256 bytes.
The downloaded update image is written to the buffer area of the flash ROM on the MCU (RX26T).

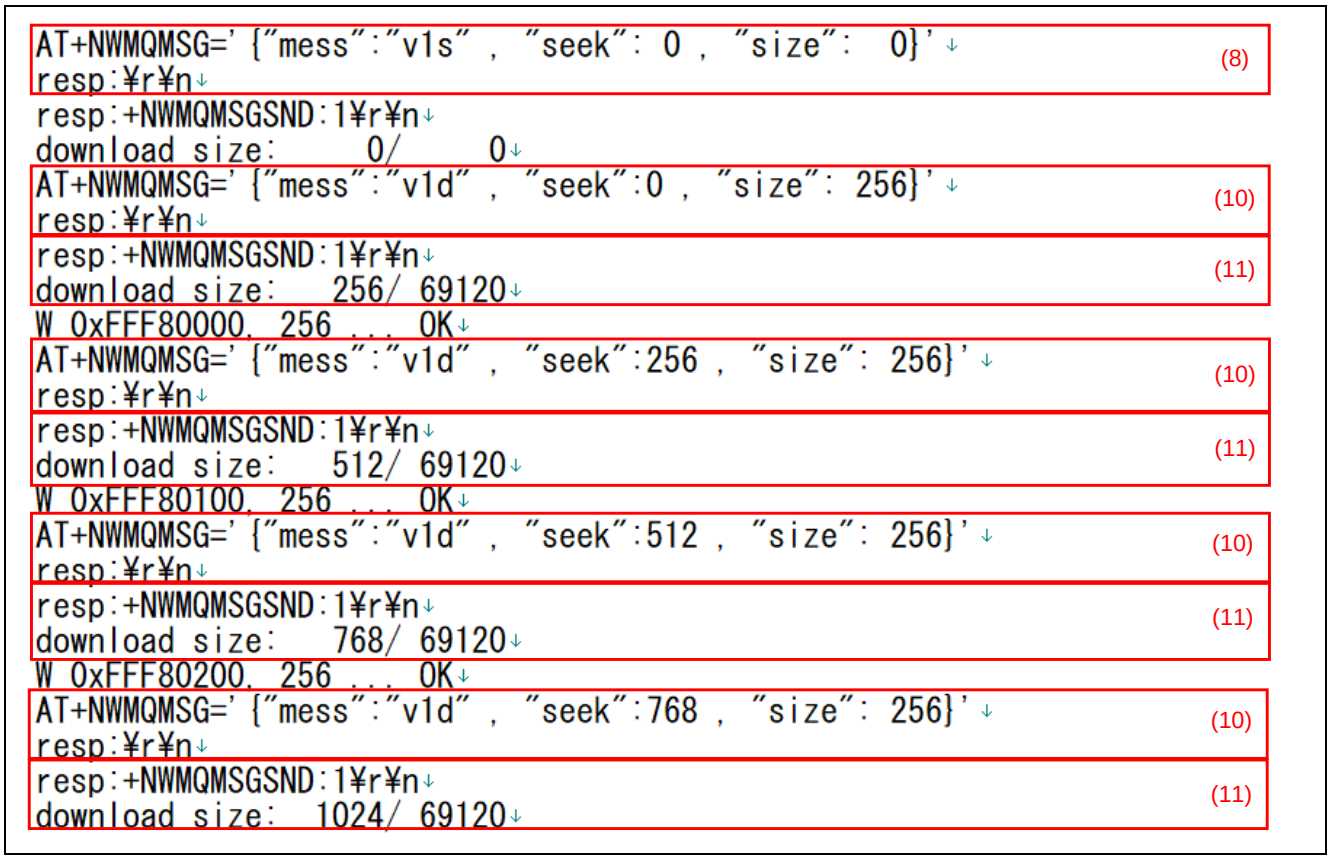

Figure 5-22 Information Logged in Steps (8) to (11)

(12) When all data of the update image is downloaded normally and writing of firmware is completed, screen display stops and the MCU (RX26T) begins to wait for a reset instruction from the PC.

Note: In general firmware update processing, when the MCU is started the next time, a bootloader verifies that a valid program exists in the buffer area, swaps the banks, performs a self-reset, erases the old program, and then starts the update image. However, in this demonstration program, a bank swap and self-reset are performed by an instruction from the PC.

The MCK-RX26T is ready for starting the update image at this time. However, before starting the update image, you must switch between the JP8 and JP11 on the inverter board according to the current detection method (1 shunt or 2-shunt method). Make sure that the motor is stopped and set the JP8 or JP11 according to the current detection method (1 shunt or 2-shunt method) of the update image.

When you enter “RESET” from the PC, the MCK-RX26T swaps the banks and resets itself. Then, the update image starts.

```
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)

Figure 5-23 Information Logged When the Procedure Ended Normally

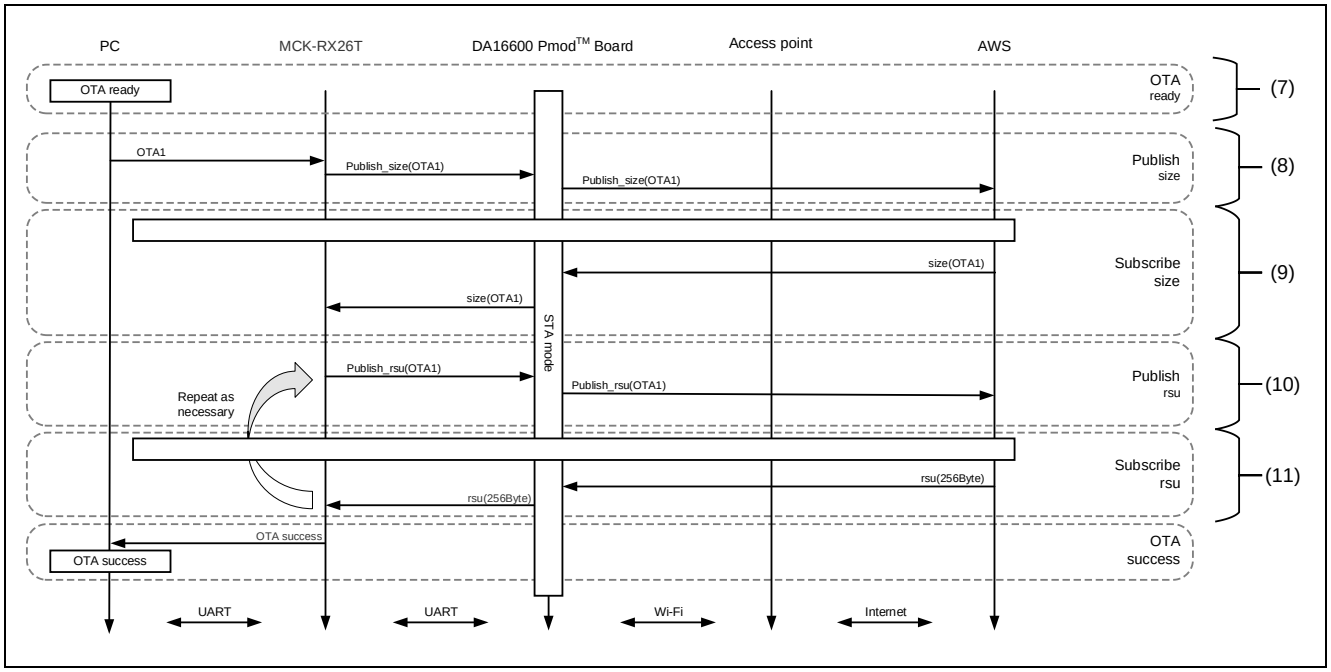

Figure 5-24 Flow of Data During an OTA Update

6. Procedure for Building a Project

The sample program is provided as a suite of e² studio projects. This chapter describes the procedures for importing a project when you use e² studio and when you use CS+. After an import is completed, confirm the build settings. The validity of these procedures was verified under conditions where Smart Configurator V2.19.0, r_fwup v2.01, and r_flash_rx v5.10 were used.

6.1 Procedure When You Use e² studio

6.1.1 Importing a Project into e² studio

If you use e² studio, use the following procedure to import each necessary project into e² studio.

Note that the path name of the folder that stores a project managed by e² studio, including the folder name, must not include white spaces, fullwidth characters, halfwidth Kana characters, and halfwidth symbols ($, #, and % especially). Also note that the screenshots used in this section may slightly differ from the actual screens you see, depending on the version of e² studio.

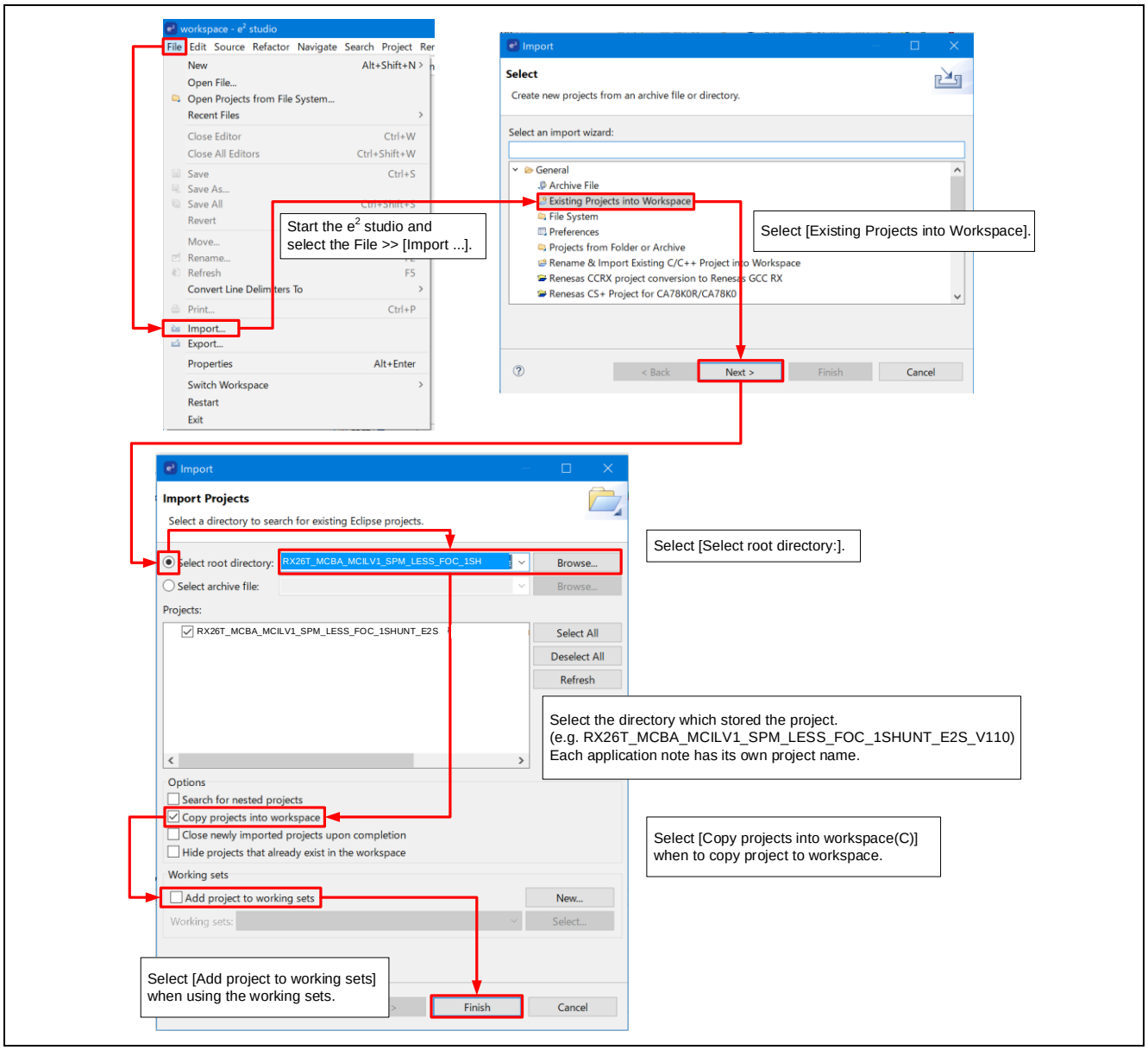

Figure 6-1 How to Import a Project into e² studio

6.1.2 Confirming That You Have All Necessary FIT Modules

Before building a project with e^2 studio, confirm that all necessary FIT modules have been downloaded. If a necessary FIT module is missing, building may fail because necessary code is not generated.

This section shows an example in which the FIT module “r_flash_rx” has been downloaded but “r_fwup” has not been downloaded. If a necessary FIT module has not been downloaded, the module is grayed out and a message appears stating that this component is missing (as shown in the figure). If the “Component is missing” message appears, click the [downloading it] link to download the FIT module.

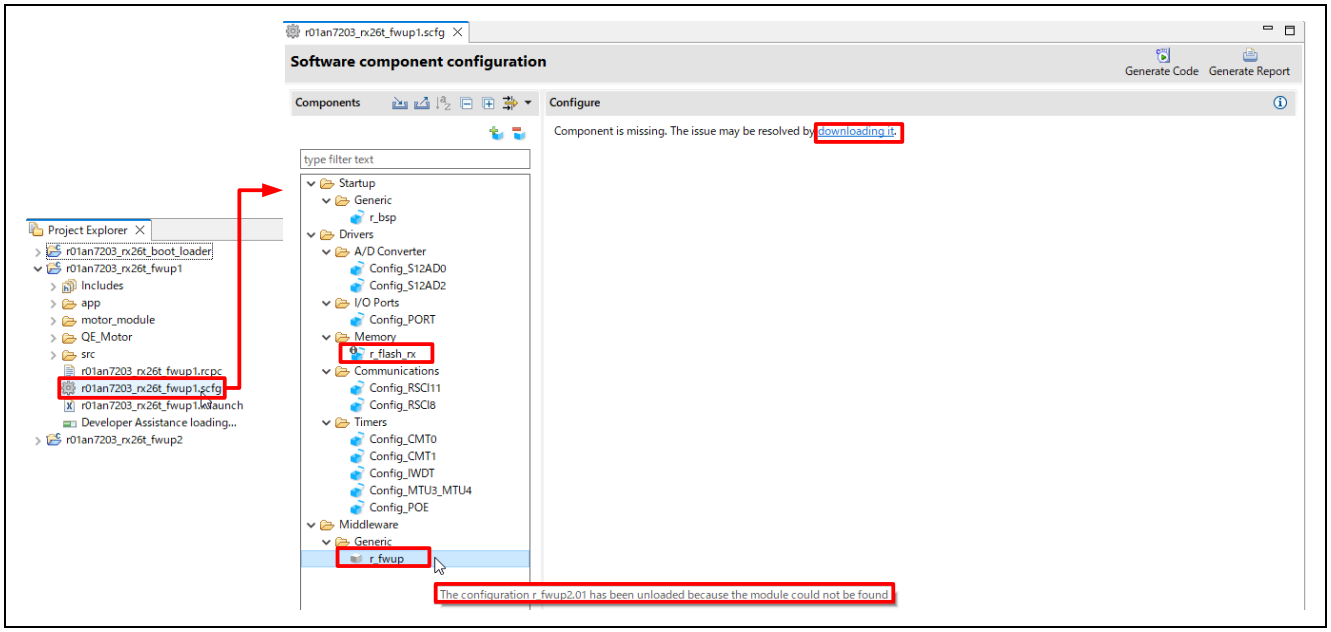

Figure 6-2 Checking the FIT Module

6.1.3 Specifying the Build Options

1. Right-click the name of the target project, and then, in the menu that appears, select [Properties].
2. Select [C/C++ Build] → [Settings], click the [Toolchain] tab, and then check the toolchain and its version.
   - Toolchain: Renesas CC-RX
   - Version: v3.05.00

6.1.4 Building a Project

1. In the Project Explorer, right-click the target project, and then select [Build Project].
2. A build starts and processing progress is displayed in the console view. When the “Build Finished” message appears, the build is complete. A file assigned the project name followed by the extension “.mot” is output to the HardwareDebug folder.

6.2 Procedure When You Use CS+

If you use CS+, use the following procedure to import each necessary project into CS+.

Note that the path name of the folder that stores a project managed by CS+, including the folder name, must not include white spaces, fullwidth characters, halfwidth Kana characters, and halfwidth symbols ($, #, and % especially). Also note that the screenshots used in this section may slightly differ from the actual screens you see, depending on the version of CS+.

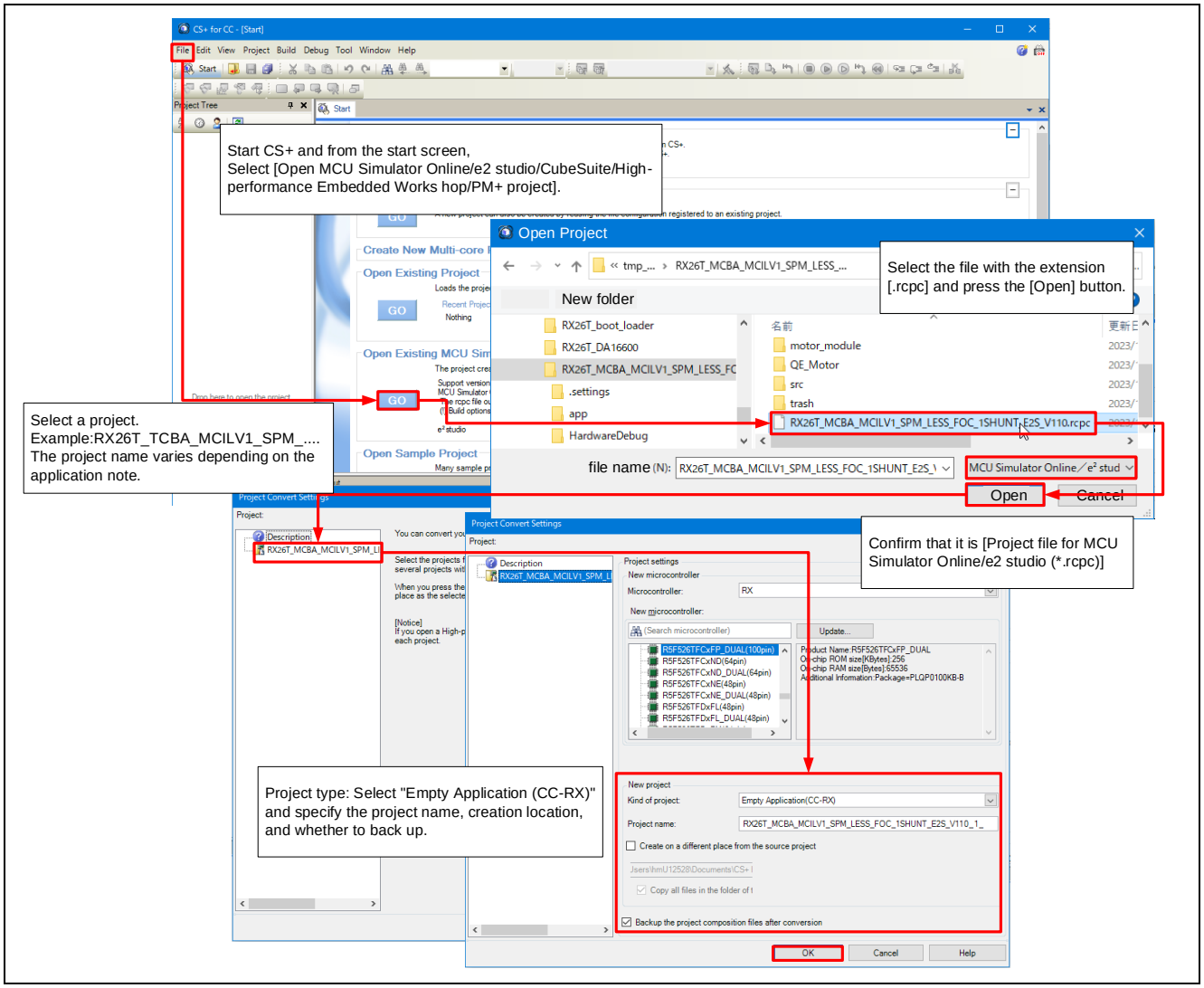

Figure 6-3 How to Import a Project into CS+

6.2.1 Building a Project

1. Select [Build] → [Build Project].
2. A build starts and the processing progress is displayed in the console view. When a message stating that the build ended appears, the build is complete. A file assigned the project name followed by the extension ".mot" is output to the HardwareDebug folder.

7. Troubleshooting

7.1 Cannot Connect to AWS

There are various possible causes that prevent connection to AWS. Typical causes include an incorrect AWS endpoint and an incorrect certificate or private key. To identify the causes, examine the log via the debug port of the DA16600 Pmod™ board. This chapter describes how to examine the log and shows some corrective measures.

7.1.1 Examining the Log of the DA16600 Pmod™ Board

7.1.1.1 Necessary Components

To examine the log of the DA16600 Pmod™ board, the following module is necessary:

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

7.1.1.2 Connecting the Component

Connect a Pmod USBUART to the DA16600 Pmod™ board as shown in the following figure. Do not change the connections to the MCK-RX26T. Before connecting the Pmod USBUART, turn off the power to the MCK-RX26T.

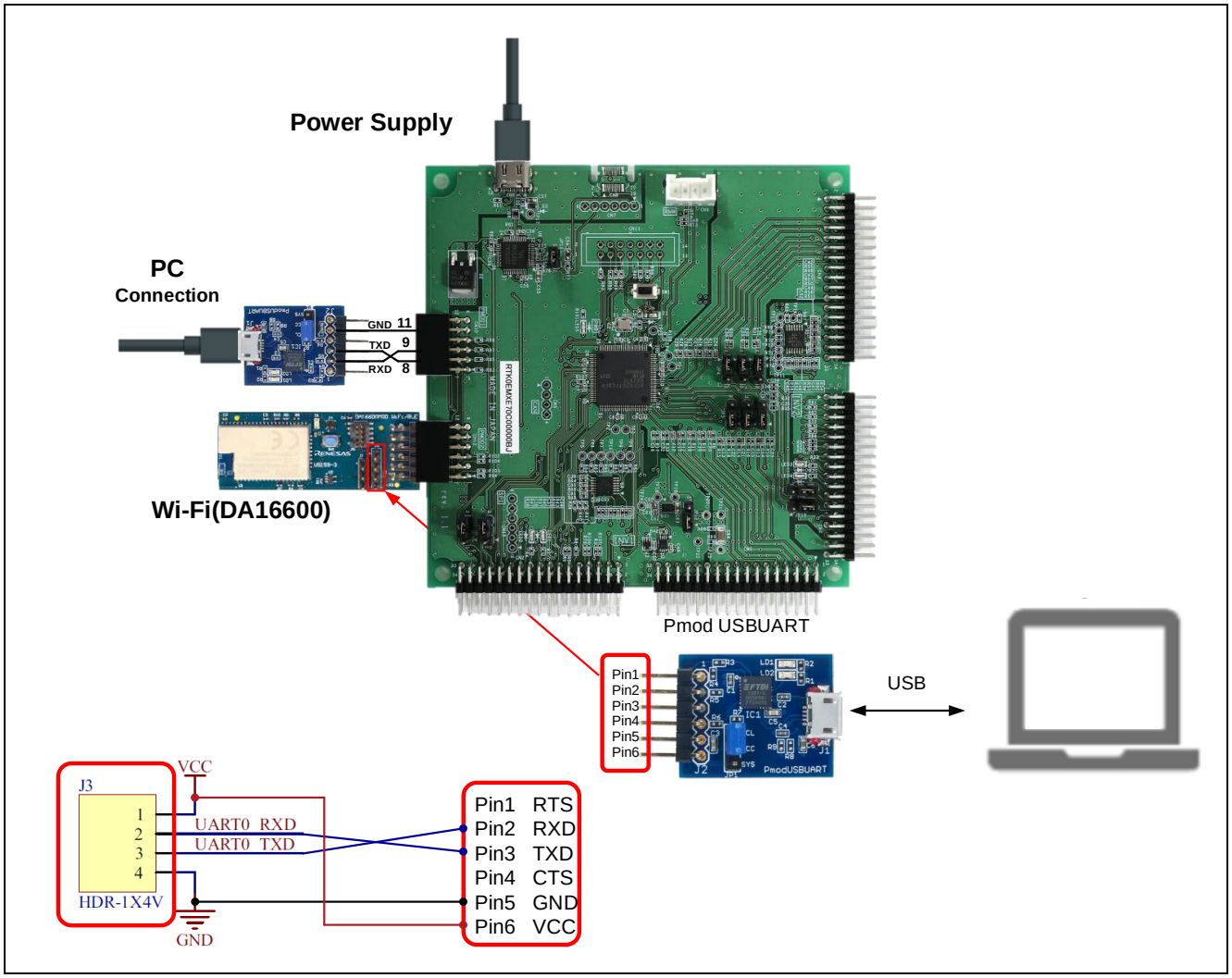

Figure 7-1 Connecting a Pmod USBUART to the DA16600 Pmod™ Board

7.1.1.3 Starting Tera Term

Turn on the power to the MCK-RX26T, and then start Tera Term. Select the [Serial] radio button, and select the COM port to which the Pmod USBUART is connected.

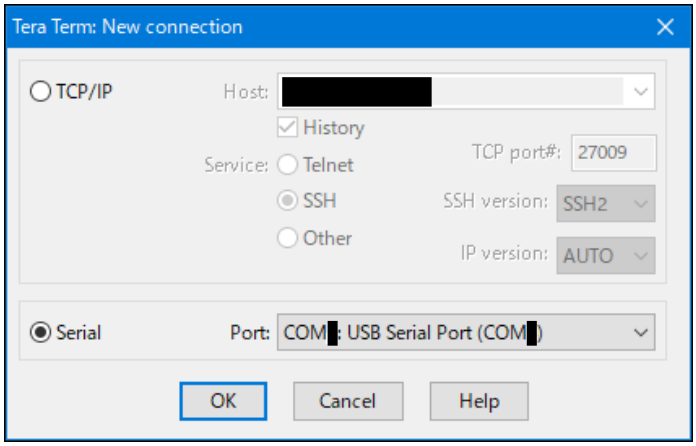

Figure 7-2 Starting Tera Term

7.1.1.4 Tera Term Terminal Setup

In the Tera Term window, select [Setup] → [Terminal]. The [Terminal setup] dialog box opens. Specify the settings as shown in the following figure.

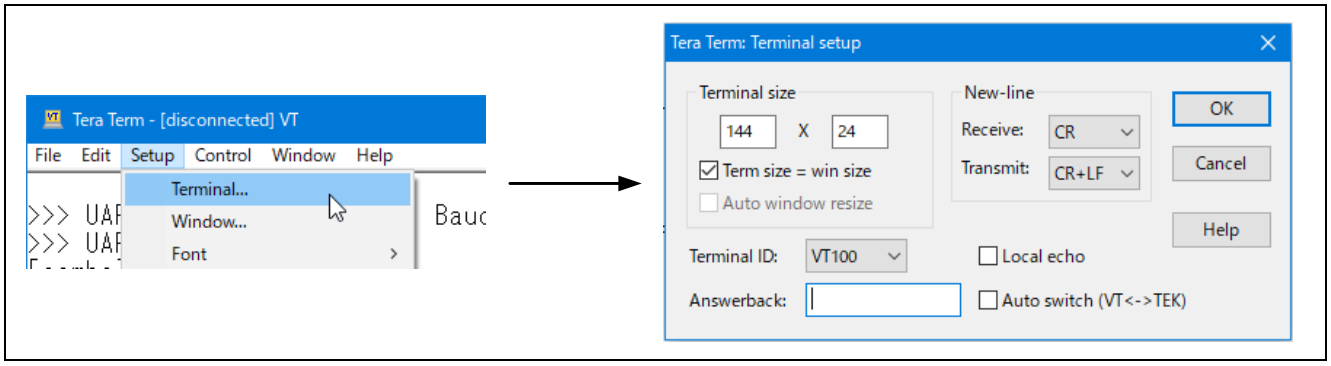

Figure 7-3 [Terminal setup] Dialog Box of Tera Term

7.1.1.5 Tera Term Serial Port Setup

In the Tera Term window, select [Setup] → [Serial port setup and connection]. The [Serial port setup and connection] dialog box opens. Note that the baud rate must be set according to the debug port of the DA16600 Pmod™ board. Specify the settings as shown in the following figure.

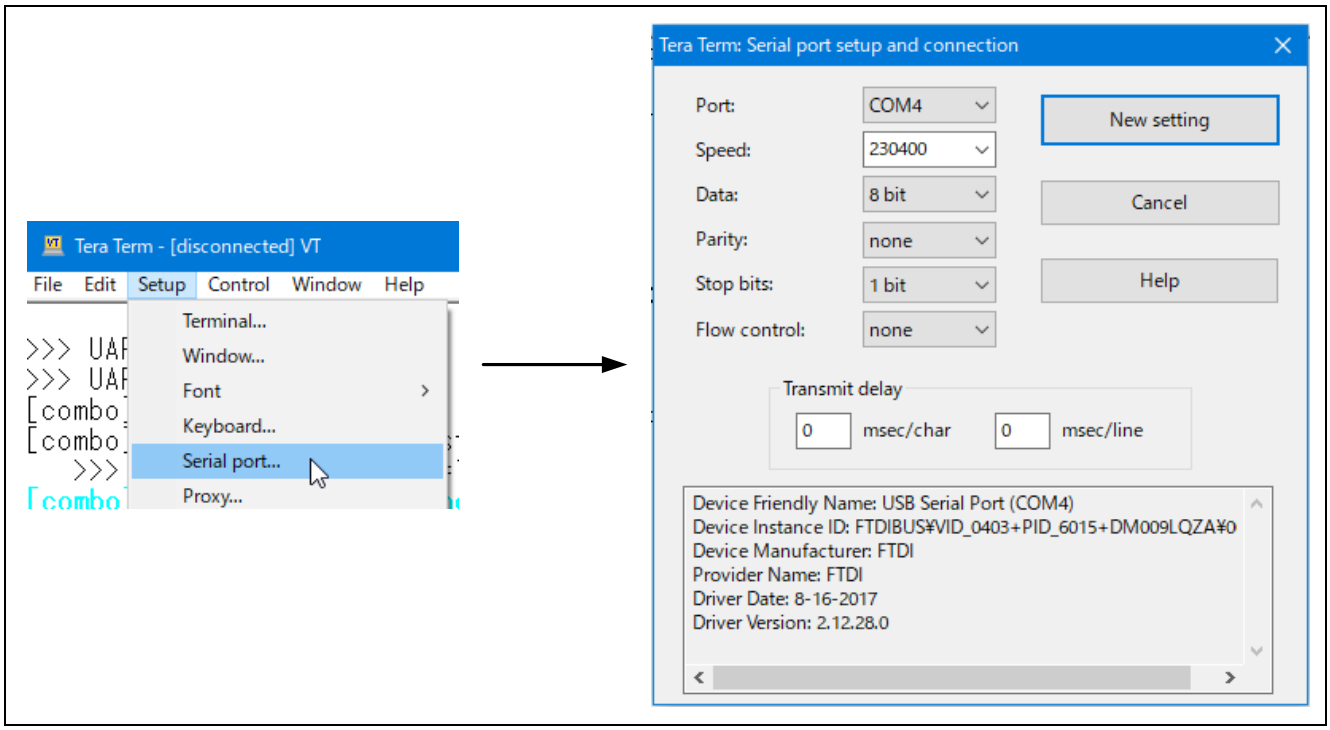

Figure 7-4 [Serial port setup and connection] Dialog Box of Tera Term

7.1.1.6 Checking the Operation

Turn on the power to the MCK-RX26T, and then confirm that the log is displayed correctly. An example of typical information displayed in the log is as follows.

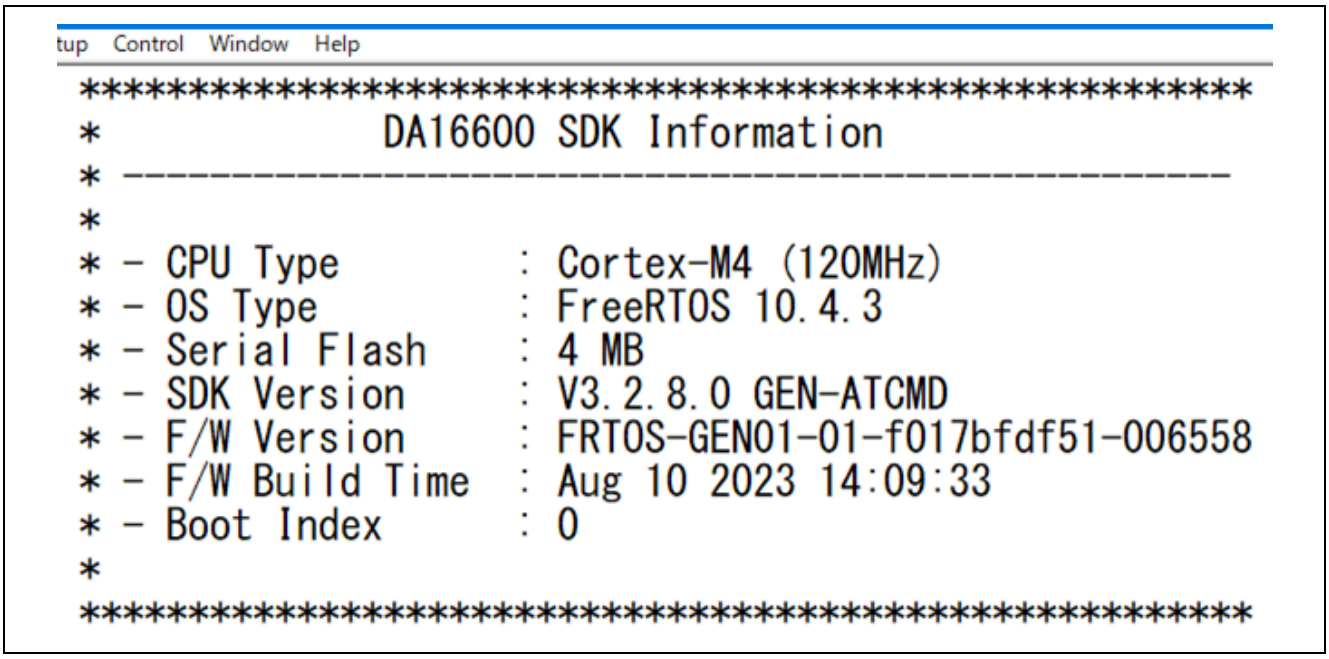

Figure 7-5 Example of Information Displayed in the Log

7.1.2 Updating the Version of the Firmware of the DA16600 Pmod™ Board

Operation of the sample program provided by this application note was verified by using the following firmware: DA16600_IMG_FreeRTOS_ATCMD_UART2_EVK_v3.2.8.0. If the version of the firmware you have is different from this version, update it as described below. This section shows an example in which the version will be updated to v3.2.8.0.

7.1.2.1 Downloading Firmware

Download the firmware DA16200 DA16600 FreeRTOS SDK Image (v3.2.8.0) from the following webpage:

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

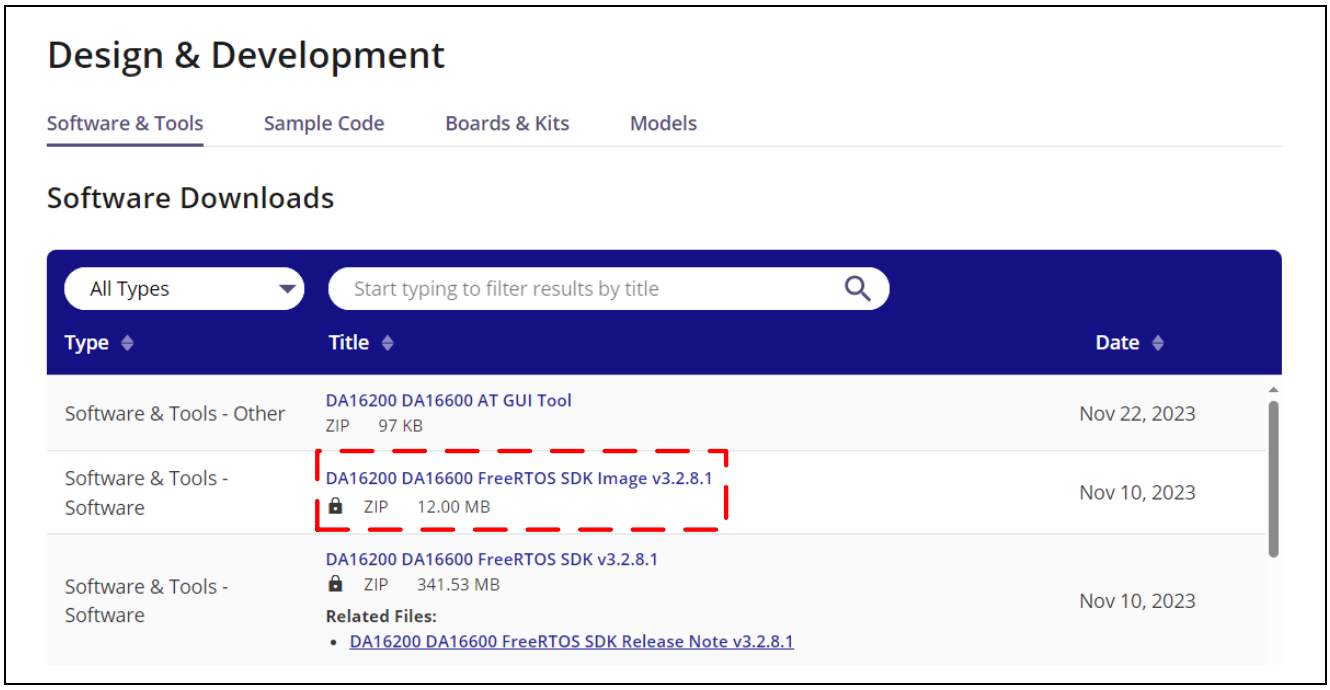

Figure 7-6 Downloading Firmware

The sample program can operate if the version of the firmware is v3.2.8.0 or later.

You may be able to download only firmware whose version is later than v3.2.8.0. If you want to obtain version v3.2.8.0 (for example, because of a version-dependent problem), contact a Renesas Electronics Corporation sales representative or an authorized Renesas Electronics Corporation product distributor.

7.1.2.2 Uncompressing the Downloaded File

The downloaded file will be a ZIP file. Uncompress it in any folder of your choice. After uncompressing the file, also uncompress the following ZIP file:
DA16600_IMG_FreeRTOS_ATCMD_UART2_EVK_v3.2.8.0_4MB.zip.

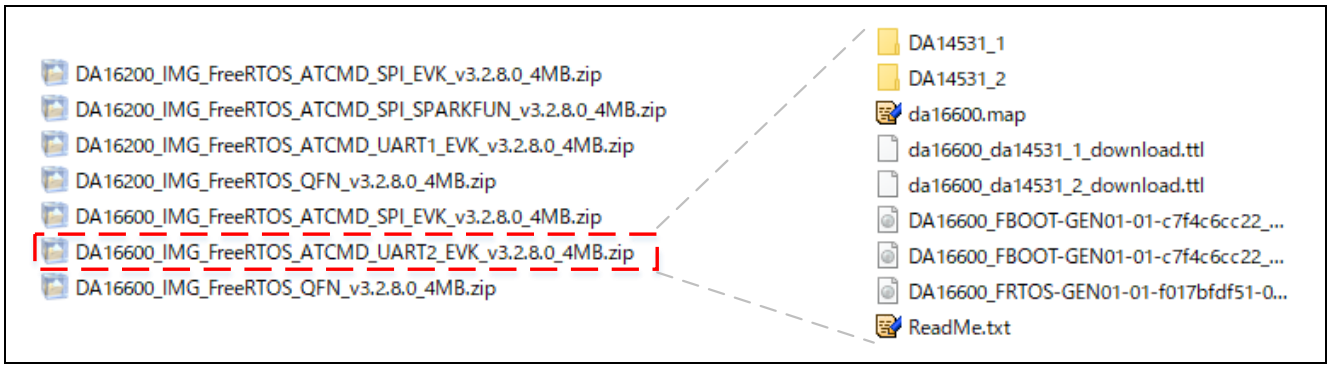

Figure 7-7 Uncompressing the Firmware

7.1.2.3 Connecting a Pmod USBUART

Connect a Pmod USBUART to the DA16600 Pmod™ board as shown in the following figure.

Make sure that all Pmod pins of the DA16600 Pmod™ board are open. Do not connect the Pmod USBUART to the PC before connection between the Pmod USBUART and the DA16600 Pmod™ board is completed.

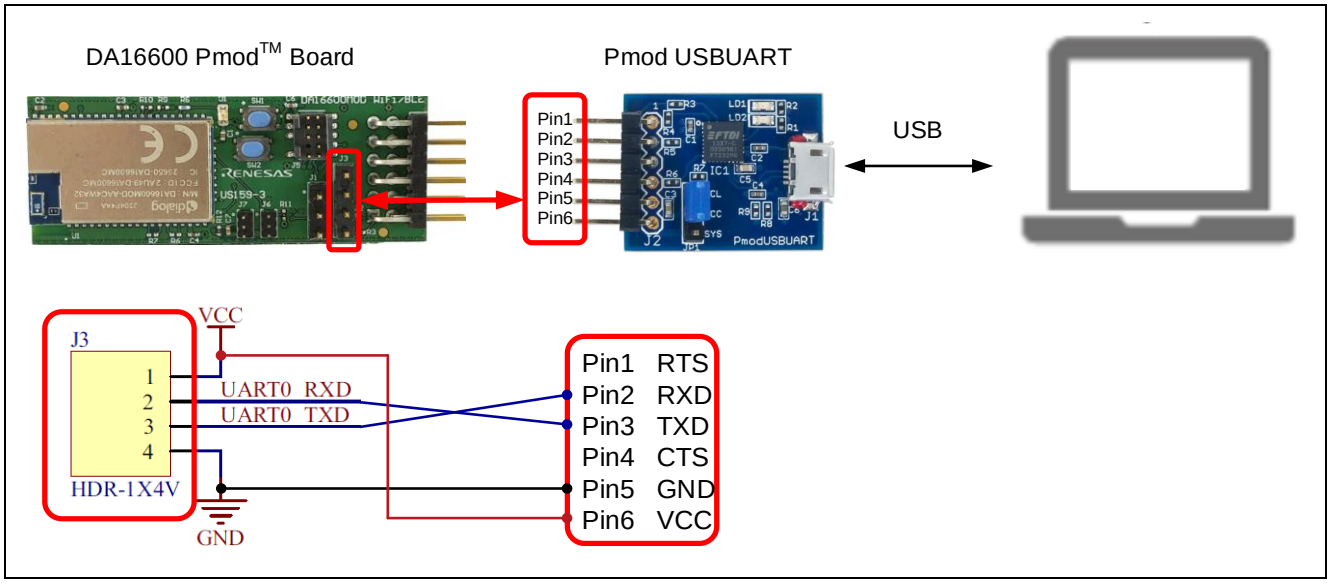

Figure 7-8 Connecting a Pmod USBUART to the DA16600 Pmod™ Board

7.1.2.4 Supplying Power

When you connect the Pmod USBUART to the PC by USB, power is supplied to the Pmod USBUART and the DA16600 Pmod™ board.

7.1.2.5 Starting Tera Term

Turn on the power to the MCK-RX26T, and then start Tera Term. Select the [Serial] radio button, and select the COM port to which the Pmod USBUART is connected.

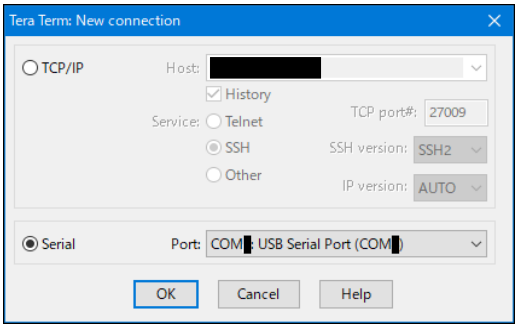

Figure 7-9 Starting Tera Term

7.1.2.6 Tera Term Terminal Setup

In the Tera Term window, select [Setup] → [Terminal]. The [Terminal setup] dialog box opens. Specify the settings as shown in the following figure.

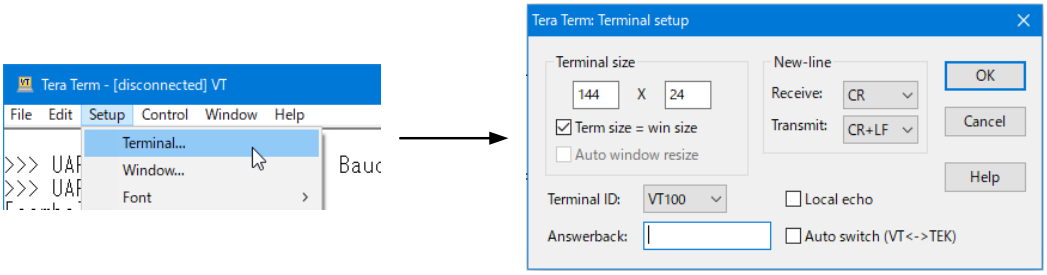

Figure 7-10 [Terminal setup] Dialog Box of Tera Term

7.1.2.7 Tera Term Serial Port Setup

In the Tera Term window, select [Setup] → [Serial port setup and connection]. The [Serial port setup and connection] dialog box opens. Note that the baud rate must be set according to the debug port of the DA16600 Pmod™ board. Specify the settings as shown in the following figure.

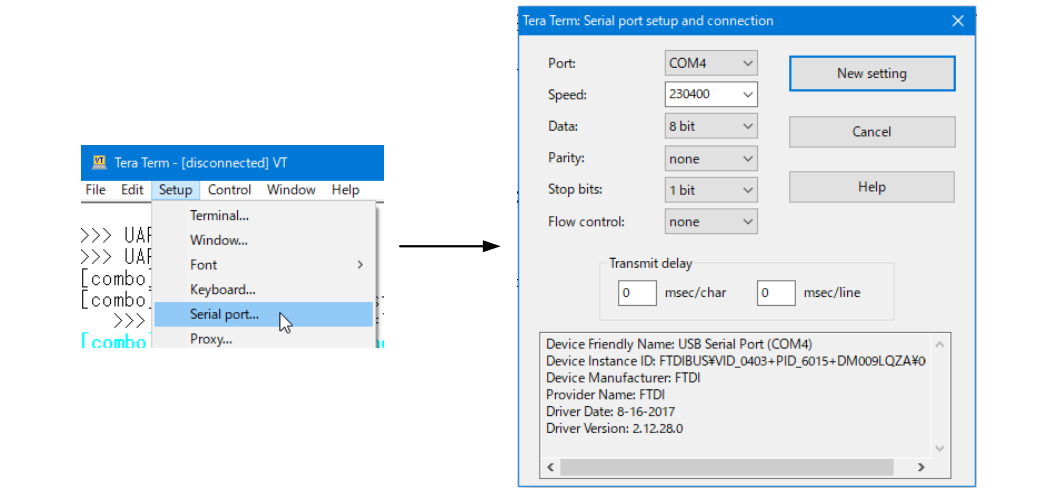

Figure 7-11 [Serial port setup and connection] Dialog Box of Tera Term

7.1.2.8 Updating the Firmware Version

In the Tera Term window, select [Control] → [Macro]. In the folder that was uncompressed in “7.1.2.2 Uncompressing the Downloaded File”, select “da16600_da14531_1_download.ttl”, and then clock [Open]. Then, in the [Confirm] dialog box, select “AT25SL321”, and then click [OK]. The selected command is automatically executed on Tera Term and the firmware version is updated.

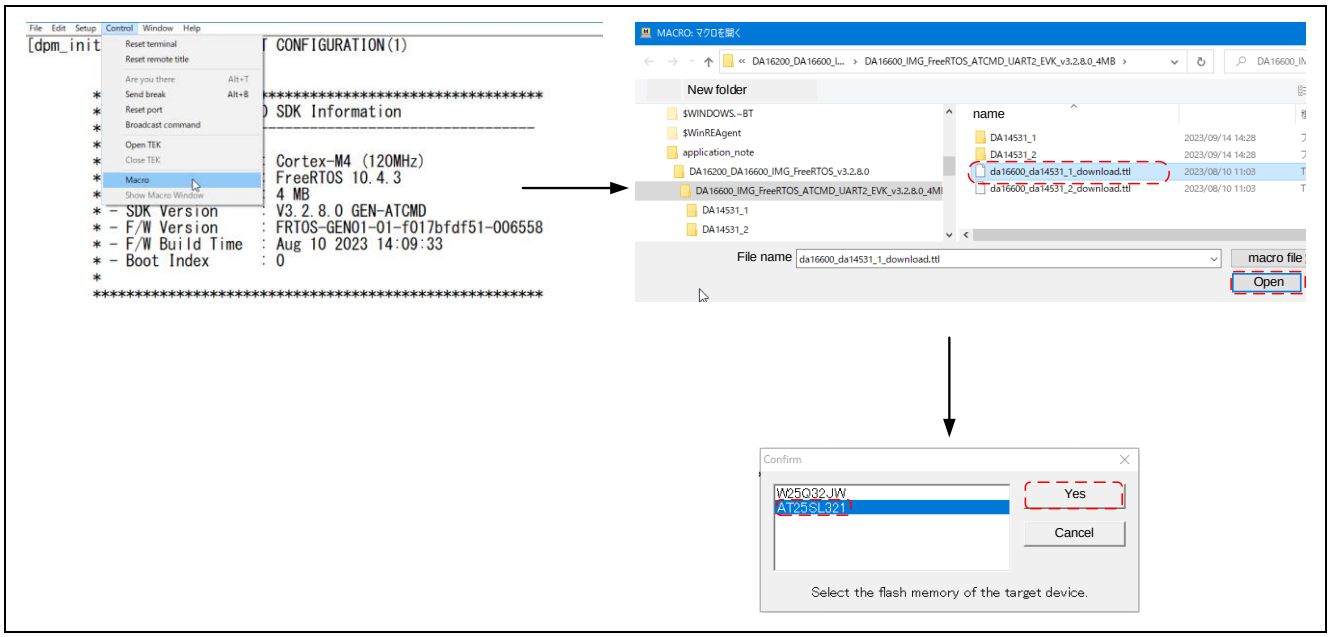

Figure 7-12 Updating the Firmware Version

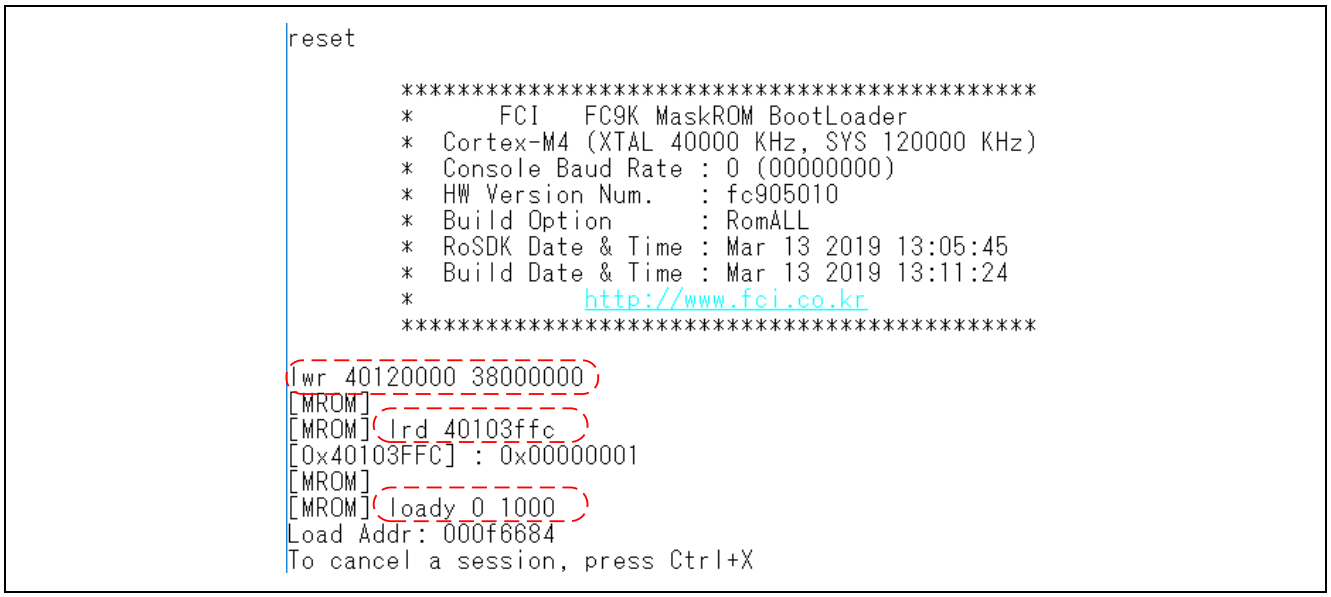

Figure 7-13 Information Displayed in Tera Term During a Firmware Update

7.1.2.9 Confirming the Version

After the firmware update is completed, confirm that its version has been changed to v3.2.8.0.

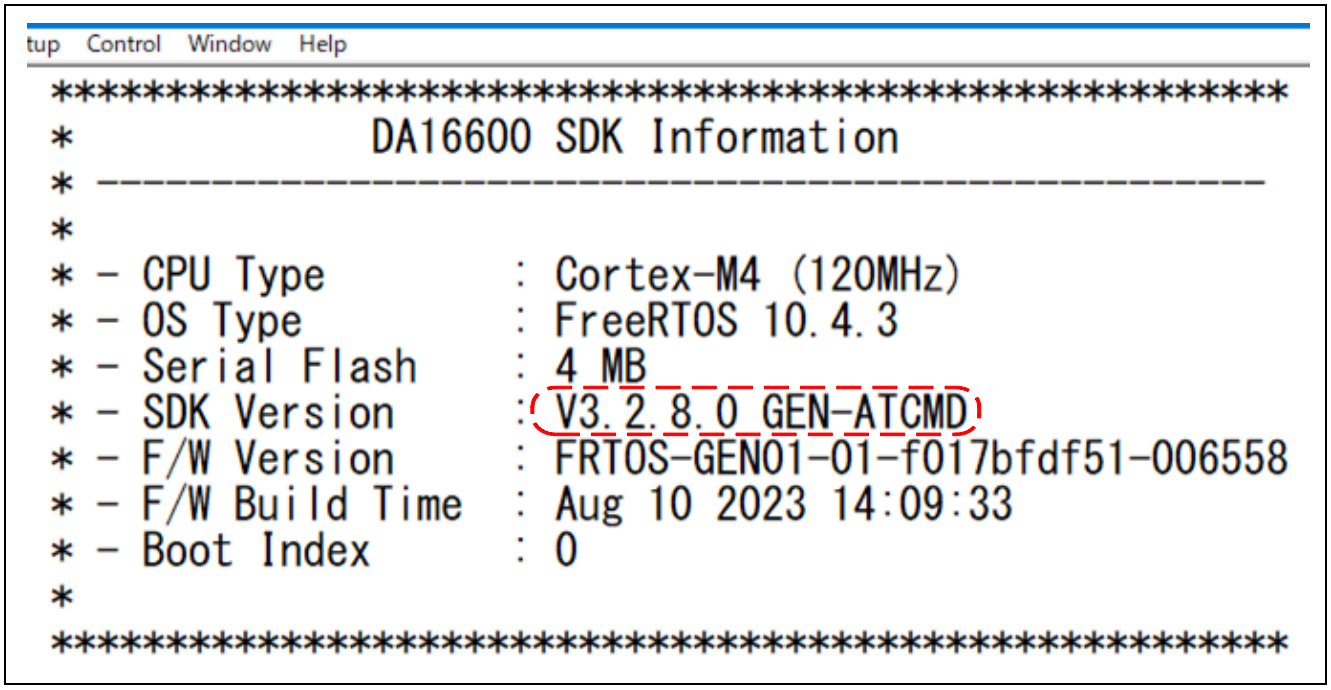

Figure 7-14 Confirming the Firmware Version

7.1.3 Troubleshooting a Problem That Displays the “Failed to establish tls-sess (0x7200)” Message

This section describes how to troubleshoot a problem that displays the “Failed to establish tls-sess (0x7200)” message as shown in the following figure.

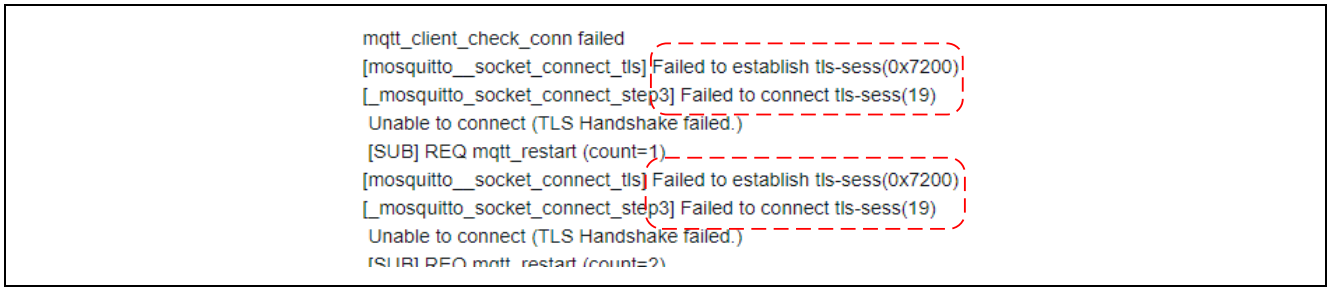

Figure 7-15 “Failed to establish tls-sess (0x7200)” Message

7.1.3.1 Setting the MQTT Buffers Again

If the “Failed to establish tls-sess (0x7200)” message appears, set the MQTT buffers again. Execute the following commands to set the buffers again:

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

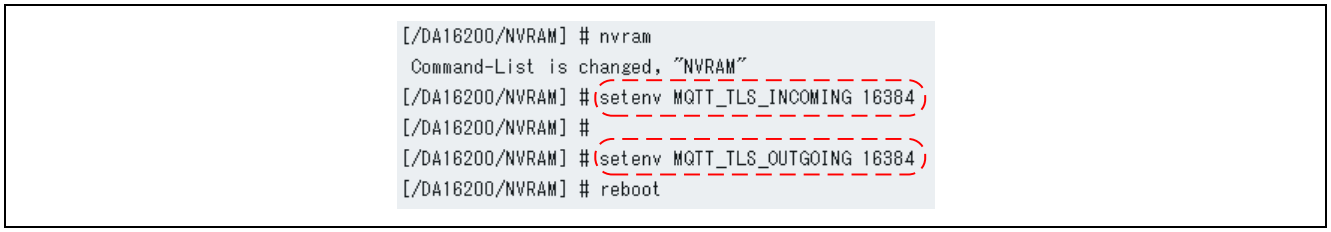

Figure 7-16 Setting the MQTT Buffers Again

7.1.3.2 Confirming the Setting Results

Execute the following command to confirm that the buffers have been set again:

```
mqtt_config status
```

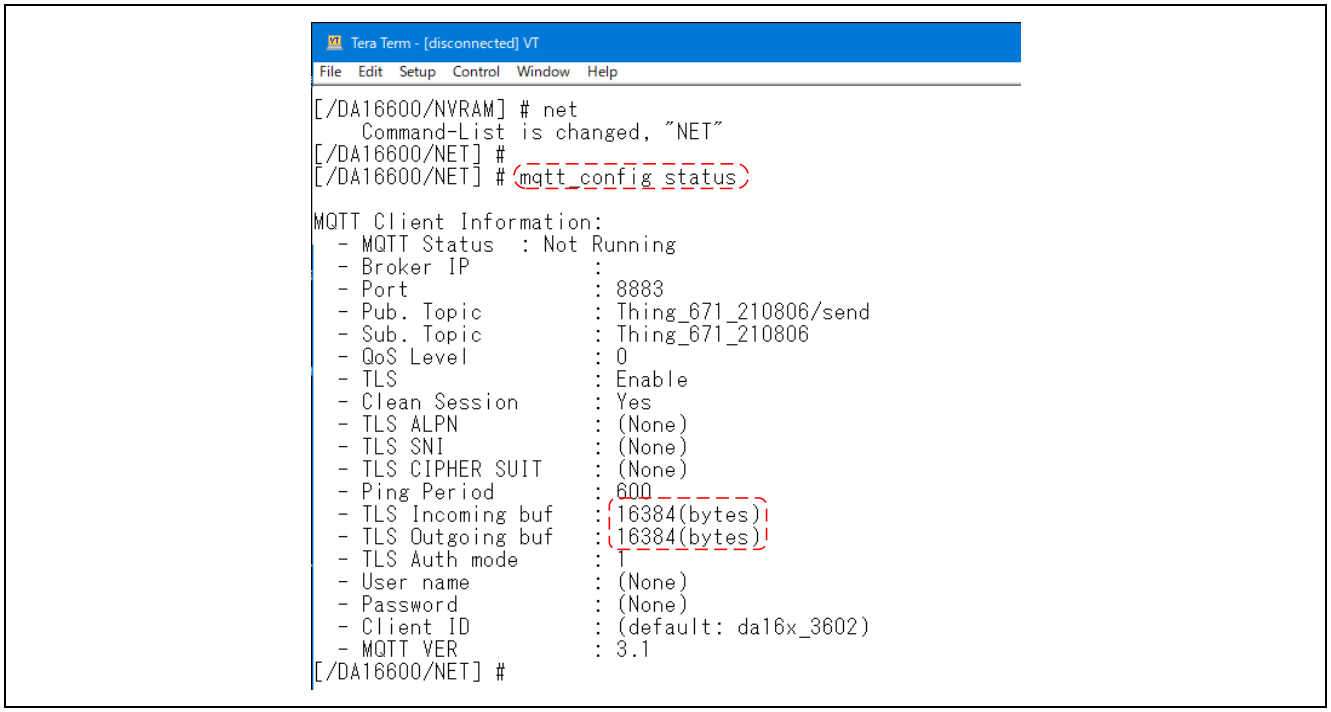

Figure 7-17 Confirming the Results of Setting the MQTT Buffers Again

8. Reference Materials

- US159-DA 16600 EVZ Evaluation Board Manual (R15UZ0006)
- User Manual DA16200 DA16600 Host Interface and AT Command (UM-WI-003)
- RX Family Firmware Update Module Using Firmware Integration Technology (R01AN6850)
- RX Family Sensorless Vector Control of a Permanent Magnet Synchronous Motor - For MCK (R01AN6858)
- MCK-RX26T User's Manual (R12UZ0111)
- Renesas MCU Firmware Update Design Policy (R01AN5548)
- Renesas Motor Workbench User's Manual (R21UZ0004)

The latest version of each document can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jan. 5, 24	—	First edition issued
1.01	Dec.23.24	9	Modified “Figure 2-3 Connected Boards”.
		53	Modified “Figure 7-1 Connecting a Pmod USBUART to the DA16600 Pmod™ Board”.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.