RX Family

How to implement OTA by using Microsoft Azure Services

R01AN6928EJ0201
Rev.2.01
July.01.24

Important Notice:

On November 21, 2023, Microsoft announced that they have decided to contribute Azure RTOS to Open Source

under the stewardship of the Eclipse foundation and Azure RTOS becomes Eclipse ThreadX.

For detailed information, please refer to the announcement titled at Microsoft Contributes Azure RTOS to Open Source.

The support strategy scheme for Eclipse ThreadX will be determined and communicated at a later date.

Microsoft will discontinue the Azure RTOS and Azure RTOS Middleware under the existing agreement LICENSED-HARDWARE.txt.

It's important to note that updates for Azure RTOS on these hardware will no longer be provided.

Introduction

This document describes how to create an environment that enables deployment of over-the-air (OTA) updating of IoT devices using Microsoft Azure. OTA updates employ an Azure service called Device Update for IoT Hub. This functionality is referred to as ADU in this document, and a step-by-step guide is presented.

In addition, by using QE for OTA, it is possible to simplify the process required to build an ADU project.

Note that the information presented in this document is subject to change without notice.

Target Devices

- CK-RX65N (Ethernet)
- Renesas Starter Kit+ for RX65N-2MB (Ethernet)
- RX65N Cloud Kit (Wi-Fi)
- RX72N Envision Kit (Ethernet)
- RX671 RSK (Ethernet)

Note: The descriptions in this document use the CK-RX65N as an example.

Development Environment Used

Integrated development environment (IDE)	e2 studio 2024-04
Compiler	Renesas C/C++ Compiler for RX Family CC-RX V3.06.00 GCC for Renesas 8.3.0.202311-GNURXGCC
Driver package (RDP)	RX Driver Package V1.41
Firmware update module	Firmware Update module V1.06 Firmware Update module V2.01*1
Azure RTOS	6.4.0_rel-rx-1.0.0
Flash programming tool	Renesas Flash Programmer V3.15
MOT file conversion tool*1	Renesas Image Generator v3.03 (bundled with the sample project)
	Renesas Secure Flash Programmer (RX MCUs mot file converter 2.0.2) (Installation procedure described separately.)
Python runtime environment	Python 3.12.0 (The installation procedure is described separately.)
Key generation tool	Win32/Win64 OpenSSL v3.1.3 Light (Installation procedure described separately.)

Note: 1. The sample project in this document comes in different versions, each of which is applicable to a specific combination of a supported target board and firmware update module. The MOT file conversion tool to be used also differs depending on the version of the firmware update module you use.
For valid combinations of these items, see the table below.

Table 1-1 Combinations of Target Boards and Firmware Update Modules

Supported target board	Firmware update module	MOT file conversion tool
CK-RX65N	v2.01	Renesas Image Generator
Renesas Starter Kit+ for RX65N-2MB RX65N Cloud Kit RX72N Envision Kit RX671 RSK	v1.06	Renesas Secure Flash Programmer

Note: For the version number of the firmware update module shown in this document, “v2” or “v1” generically refers to all its variations with different subversion numbers (that is, “v2.*xx*” or “v1.*xx*”).

Contents

1. Allocation of Memory Areas and Operations on Them for Using ADU 5
1.1 Firmware Memory Area Allocation 5
1.1.1 Operations Performed by Firmware Update Module v1 5
1.1.2 Operations Performed by Firmware Update Module v2 7
1.2 Area Allocation in Data Flash Memory 8

2. Creating Sample Projects 9
2.1 Creating a Workspace 9
2.2 Creating the Sample Projects 9
2.2.1 Creating a New ADU Sample Project 9
2.2.2 Creating a New Bootloader Sample Project 16
2.3 Changing Project Settings 17
2.3.1 About the Background Operation (BGO) Mode 17
2.3.2 Integrating Components 17
2.3.3 Settings for Code Generation During Building 21
2.4 Creating Key Information 23
2.4.1 Installing OpenSSL 23
2.4.2 Generating a Key Pair for ECC in OpenSSL 23
2.4.3 Entering a Public Key 25
2.5 Building the bootloader Project 25
2.6 Connection Information Macro Settings 26
2.7 Checking the Initial Firmware Version 26
2.8 Building the adu_sample Project (for the Initial Firmware) 26
2.9 Creating the Initial Firmware 27
2.9.1 If Using Firmware Update Module v1 27
2.9.2 If Using Firmware Update Module v2 30
2.10 Installing the Flash Programming Tool 32
2.11 Programming the Initial Firmware 32
2.12 Executing the Initial Firmware 34
2.13 Modifying the Code of the Updated Firmware 35
2.14 Building the adu_sample Project (for the Update Firmware) 35
2.15 Creating the Updated Firmware 36
2.15.1 If Using Firmware Update Module v1 36
2.15.2 If Using Firmware Update Module v2 38
2.15.3 If Using Firmware Update Module v2 in the Bootloader v1 Environment 39

3. Operations on Microsoft Azure Portal 42
3.1 IoT Hub and Device Registration 42
3.2 Creating a Device Update Account and Instance 42
3.3 Creating a Storage Account Container 48

3.4 Preparing the Updated Firmware 53
3.4.1 Building the Updated Firmware 53
3.4.2 Creating a Manifest File 53
3.5 Uploading the Firmware Update to the Storage Container 54
3.6 Registering the Firmware Update 56
3.7 Creating an ADU Group 60
3.8 Updating the Firmware 61
3.8.1 Execution on the Target Board 61
3.8.2 Deploying the Firmware Update 62

4. Appendix 66
4.1 Control of Commands in Azure ADU and the Sample Project 66
4.2 Firmware Debugging Method 67
4.2.1 Debugging the Initial Firmware 67
4.2.2 Debugging the Updated Firmware 70

Revision History 73

1. Allocation of Memory Areas and Operations on Them for Using ADU

1.1 Firmware Memory Area Allocation

The initial firmware and updated firmware each occupy a 1 MB area of memory by the ADU sample project for RX65N.

Firmware update modules v1 and v2 share the same basic structure but have some portions that operate differently in ADU processing.

The following describes allocation of memory areas and operations on them.

1.1.1 Operations Performed by Firmware Update Module v1

The following figure shows an overview of the operations performed by firmware update module v1.

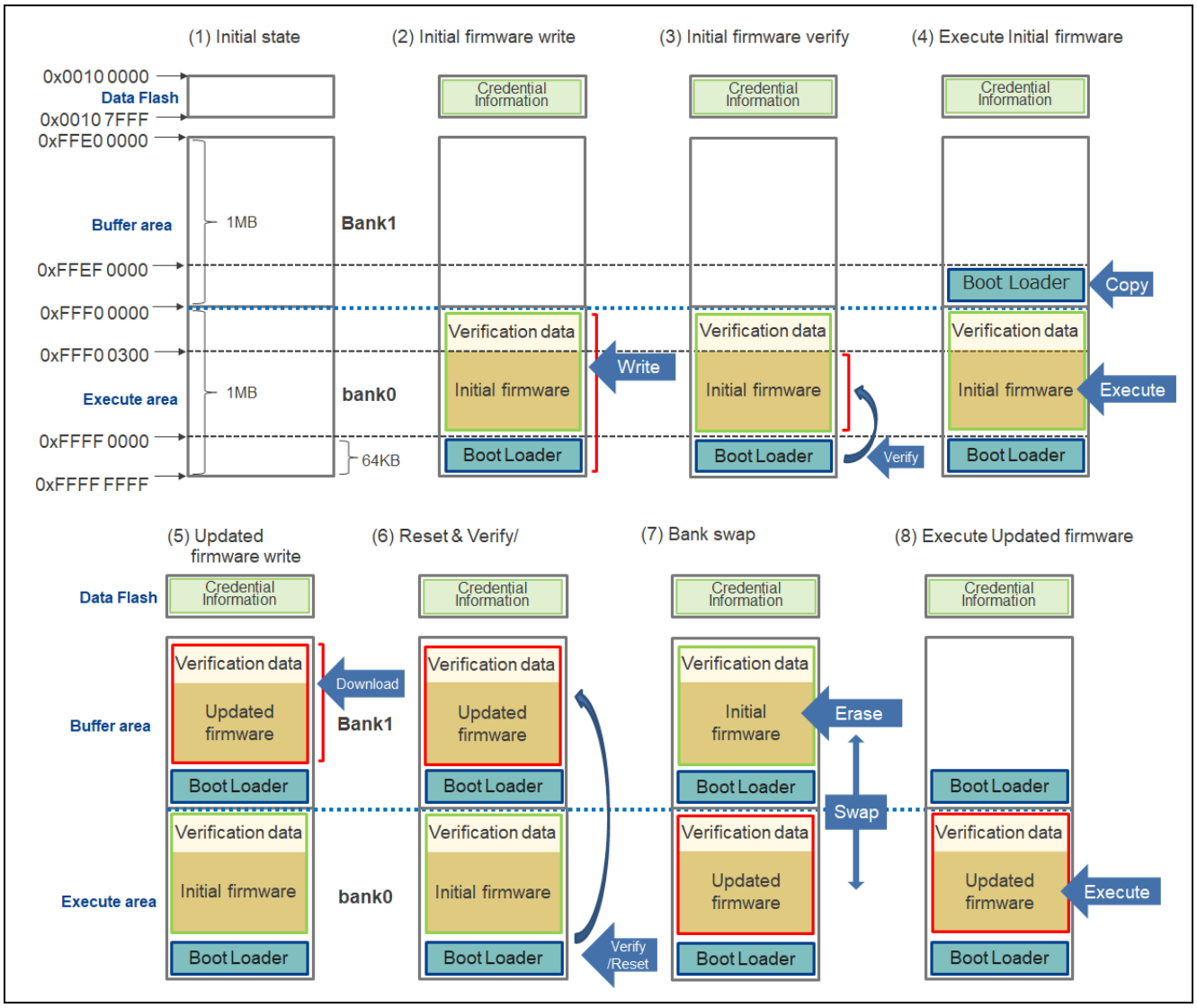

Figure 1.1 Memory Allocation for ADU (If the Version is v1)

1. In the initial state, a 1 MB area is secured for both the initial firmware and update firmware.
   In addition, a 64 KB area is secured for the secure bootloader.
   The following shows the address of each type of data in the memory.
   0xFFF0 0000 to 0xFFF0 02FF: Verification data
   0xFFF0 0300 to 0xFFFE FFFF: Firmware
   0xFFFF 0000 to 0xFFFF FFFF: Secure bootloader (Boot Loader)

2. The initial firmware and secure bootloader are written to the “execute area” by using a Renesas flash programmer.
   Note that credential information, including the information for connecting to IoT Hub, is written to data flash memory.
3. When the secure bootloader is executed, it uses the verification data (written in the address range from 0xFFF00000 to 0xFFF002FF) to verify that the written initial firmware has not been falsified.
4. If the verification is successful, the initial firmware is executed, and then the secure bootloader is copied to the buffer area.
5. The update firmware is downloaded from Azure, and then the downloaded firmware is written to the buffer area (0xFFE0000 to 0xFFEEFFFF).
6. After the download is complete, the system is reset. After the system reset is complete, the bootloader uses the verification data in the buffer area to verify that the update firmware has not been falsified.
7. When the verification is successful, the bank swapping functionality is used to swap the memory areas for the initial firmware and update firmware. After the bank swapping is complete, the bootloader deletes the initial firmware from the buffer area.
8. The update firmware in the execute area is executed.
   Utilizing the bank swapping functionality makes it possible for the addresses referenced by the application to remain unchanged after the firmware update.

1.1.2 Operations Performed by Firmware Update Module v2

The following figure shows an overview of the operations performed by firmware update module v2.

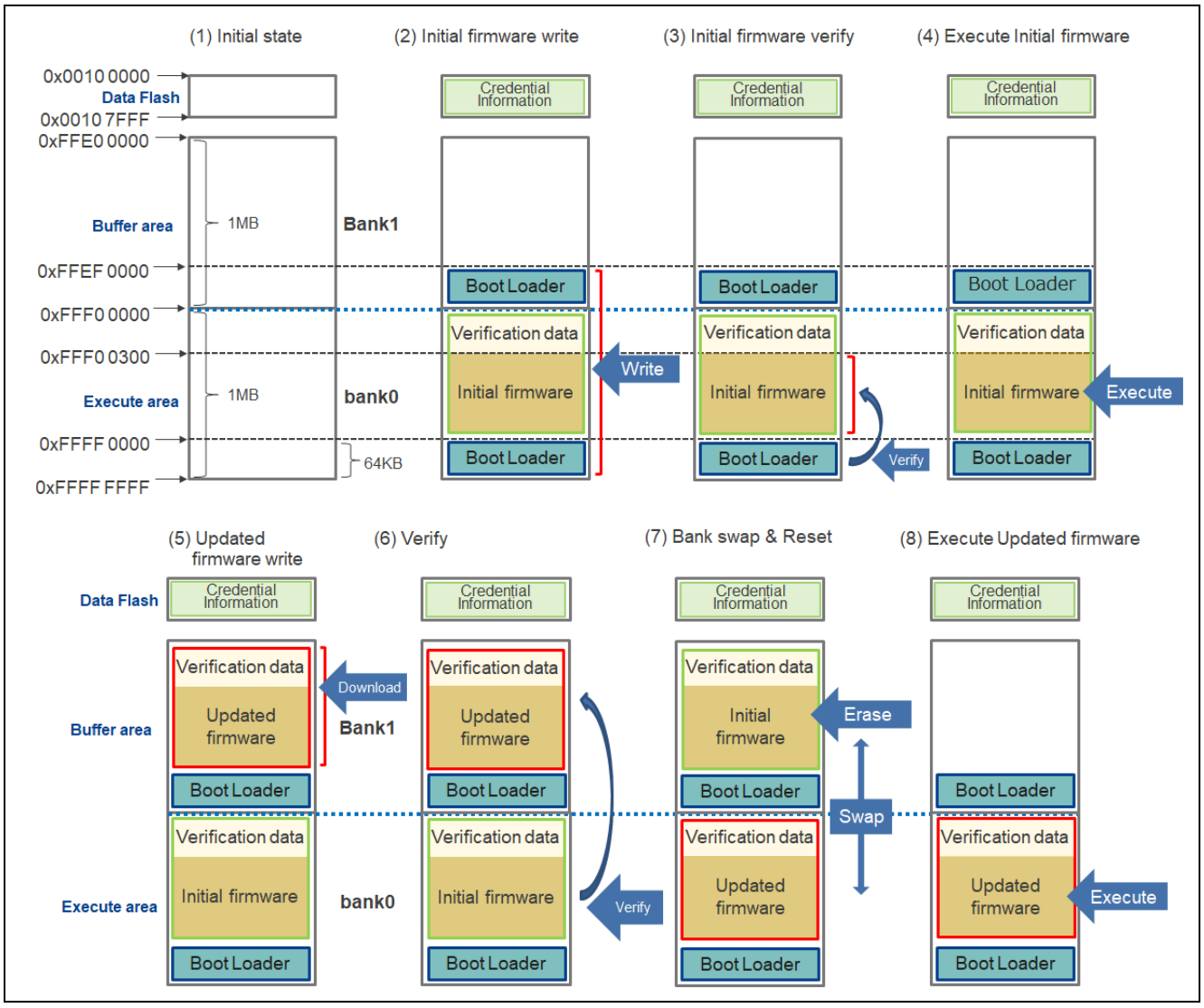

Figure 1.2 Memory Allocation for ADU (If the Version is v2)

1. In the initial state, a 1 MB area is secured for both the initial firmware and update firmware.
   In addition, a 64 KB area is secured for the secure bootloader.
   The following shows the address of each type of data in the memory.
   0xFFF0 0000 to 0xFFF0 02FF: Verification data
   0xFFF0 0300 to 0xFFFE FFFF: Firmware
   0xFFFF 0000 to 0xFFFF FFFF: Secure bootloader
2. The initial firmware and secure bootloader are written to the “execute area” by using a Renesas flash programmer.
   Note that credential information, including the information for connecting to IoT Hub, is written to data flash memory. At the same time, the bootloader is written also in the buffer area.
3. When the secure bootloader is executed, it uses the verification data (written in the address range from 0xFFF00000 to 0xFFF002FF) to verify that the written initial firmware has not been falsified.

4. If the verification is successful, the initial firmware is executed.
5. The update firmware is downloaded from Azure, and then the downloaded firmware is written to the buffer area (0xFFE0000 to 0xFFEEFFFF).
6. After the download is complete, the initial firmware uses the verification data in the buffer area to verify that the update firmware has not been falsified.
7. When the verification is successful, the bank swapping functionality is used to swap the memory areas for the initial firmware and update firmware, and then the system is reset. After the reset is complete, the bootloader starts and deletes the initial firmware from the buffer area.
8. The update firmware in the execute area is executed.
   Because the bank swapping functionality is used, the addresses referenced by the application can be used as is (even after the firmware is updated).

1.2 Area Allocation in Data Flash Memory

In this sample project, credential information, including the information required for connection to Azure, is written to data flash memory.
The credential information includes the connection destination and MAC address of Azure.

Note that the area in which the credential information is written (area in the address range from 0x00100000 to 0x00100EFF) is reserved by the system.
Therefore, if the user application needs to use data flash memory, make sure that it uses a memory area in the range from 0x00100F00 to 0x00107FFF.

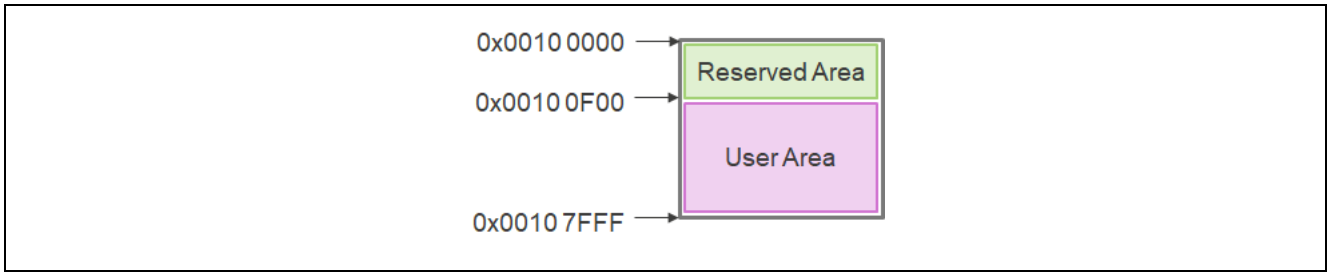

Figure 1.3 Area Allocation in Data Flash Memory

Note: In this sample project, writing of credential information to data flash memory occurs only when the initial firmware is written.
After the initial firmware has been created, therefore, do not make any changes that alter the credential information on the data flash memory. If the credential information is altered, rewrite the initial firmware.
The area allocation on data flash memory may differ depending on the build settings such as the optimization settings. Therefore, make sure that the initial firmware and update firmware are built using the same compiler and the same settings.

2. Creating Sample Projects

This section describes how to create projects that implement ADU.
ADU uses the secure boot that is RX security features. Therefore the following two sample projects are used for ADU operation.

- Azure Device Update (ADU) sample project
- Secure bootloader sample project

Follow the steps described in this section to create the two sample projects. It will be necessary to make changes to the settings, memory allocation, and source code of the newly created projects, and how to make these changes is described as well.

2.1 Creating a Workspace

Launch e² studio and create a new workspace. Keep the names of the workspace and project file as short as possible. If the total length of the full file path exceeds 256 bytes, an error will occur when you build the project.

Example: Creating a workspace in location C:\workspace

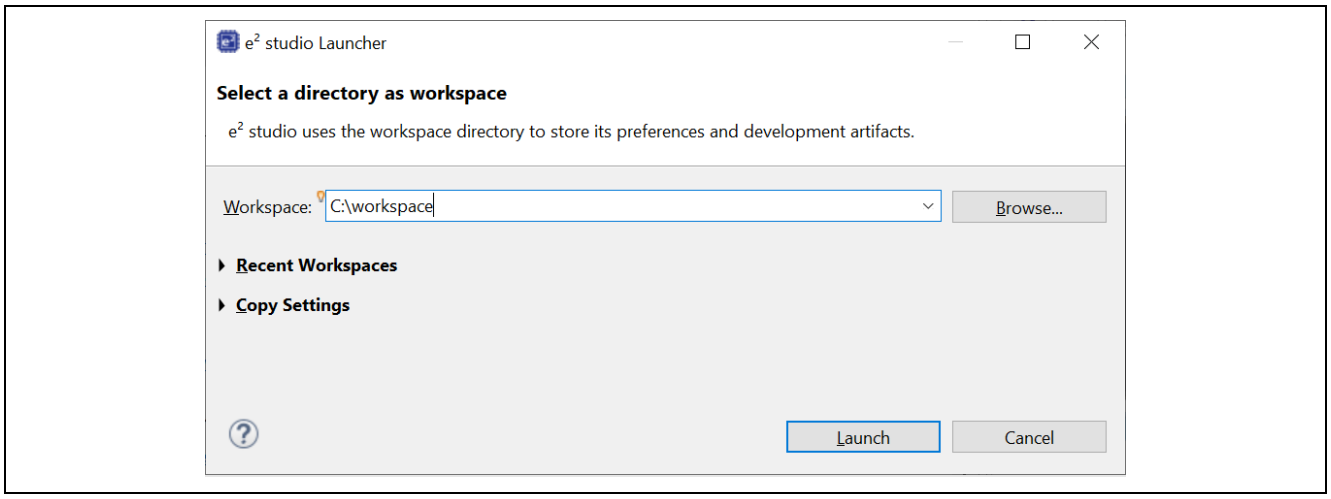

Figure 2.1 Workspace Creation Window

2.2 Creating the Sample Projects

2.2.1 Creating a New ADU Sample Project

After launching e² studio, from the **File** menu select **New** → **Renesas C/C++ Project** → **Renesas RX** to open the **New C/C++ Project** dialog box.

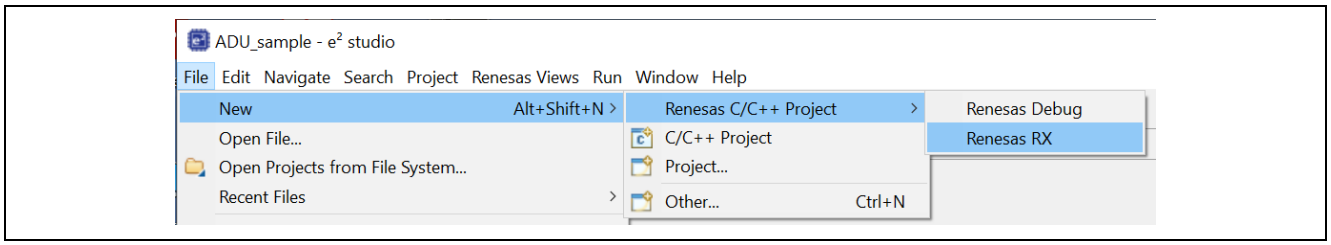

Figure 2.2 Menu Selection to Create a New Project

In the **New C/C++ Project** dialog box you will select the type of project to be created. Here, select **All** at the left, followed by **Renesas CC-RX C/C++ Executable Project**, then click the **Next >** button. A dialog box for the project type you selected (New Renesas CC-RX Executable Project) appears. To use GCC, you would select **GCC for Renesas RX C/C++ Execute Project**.

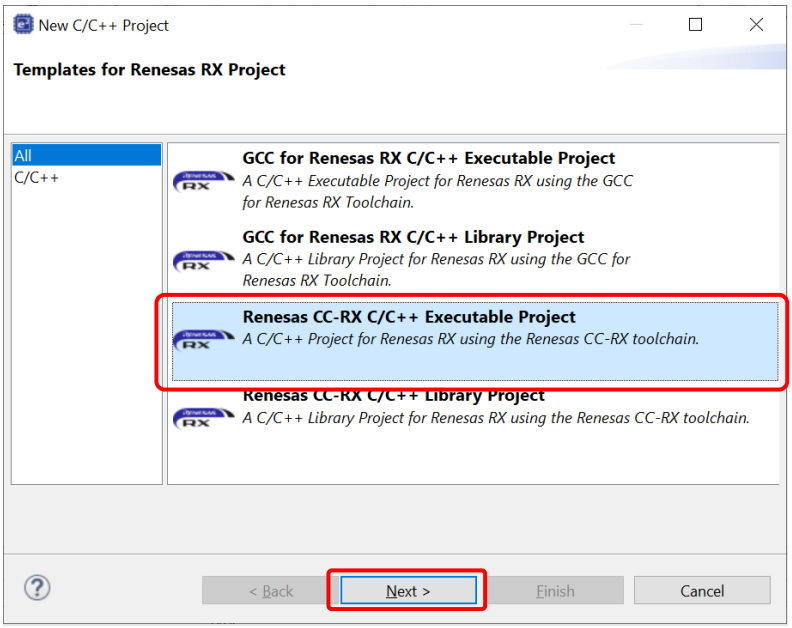

Figure 2.3 Project Type Selection Window

Next, specify a name for the project. For **Project name:** enter **adu_sample**, then click the **Next >** button. The **Select toolchain, device & debug settings** dialog box opens.

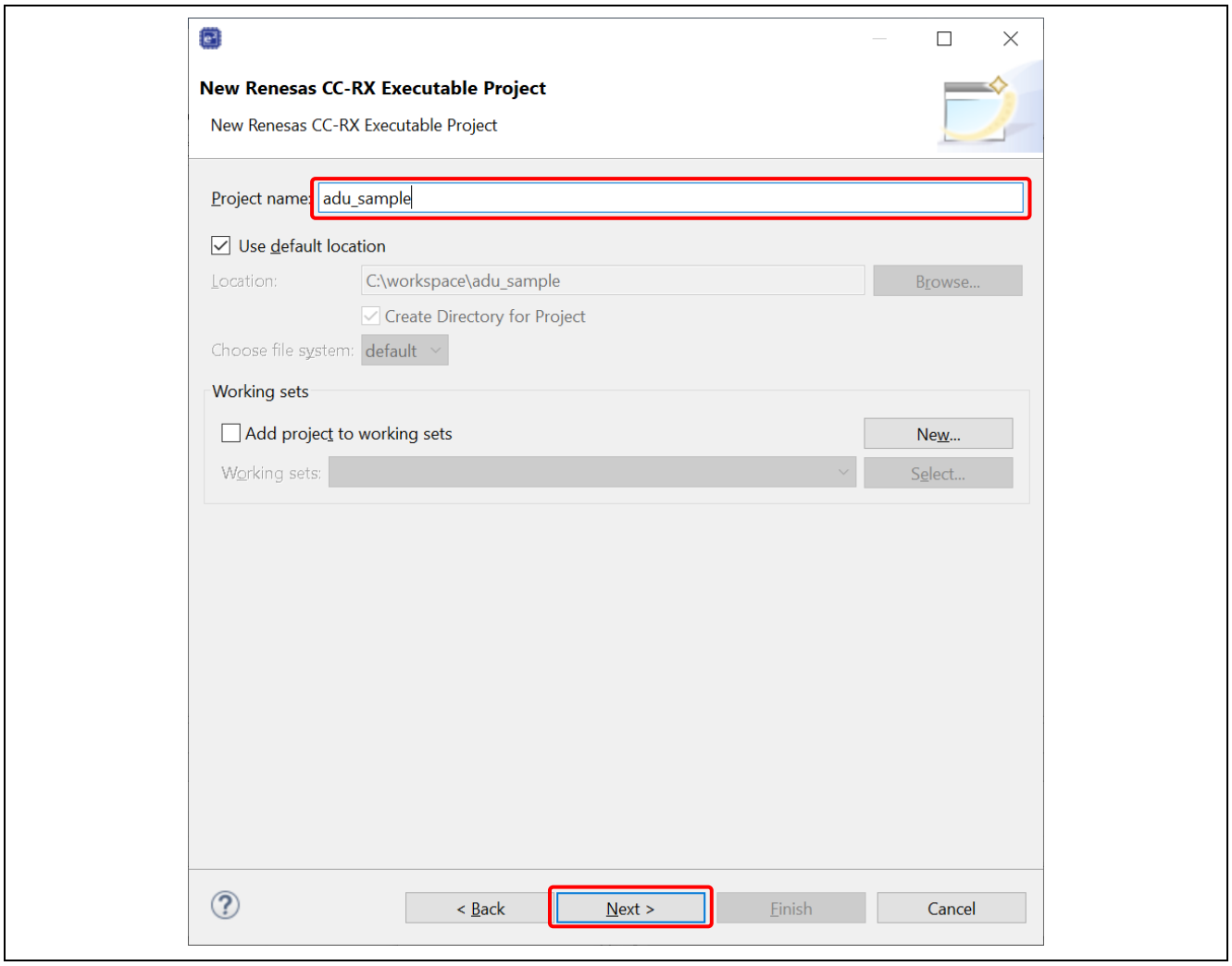

Figure 2.4 Project Name Setting Window

Select the toolchain, device, and debug settings to use for the project. The **Toolchain:** item is set based on the project type you selected earlier. To change the toolchain version, click the down arrow next to **Toolchain Version:** and select the version of your choice.

For **RTOS:** select **Azure RTOS**, and for **RTOS Version:** select **6.4.0_rel-rx-1.0.0**. When you use e^2 studio for the first time or when the desired version does not appear in the list, click **Manage RTOS Versions...**. In the RTOS Module Download dialog box that appears, select the desired version, and then download it by clicking the **Download** button.

For **Target Board:** select **CK-RX65N(DUAL)**. (**Target Device:** is selected automatically.)
For **Bank Mode:** select **Dual Bank**. Before you can use ADU, you must enable dual mode, in which the code flash memory on the device is treated as two bank areas.

When all the settings have been configured, click the **Next >** button.

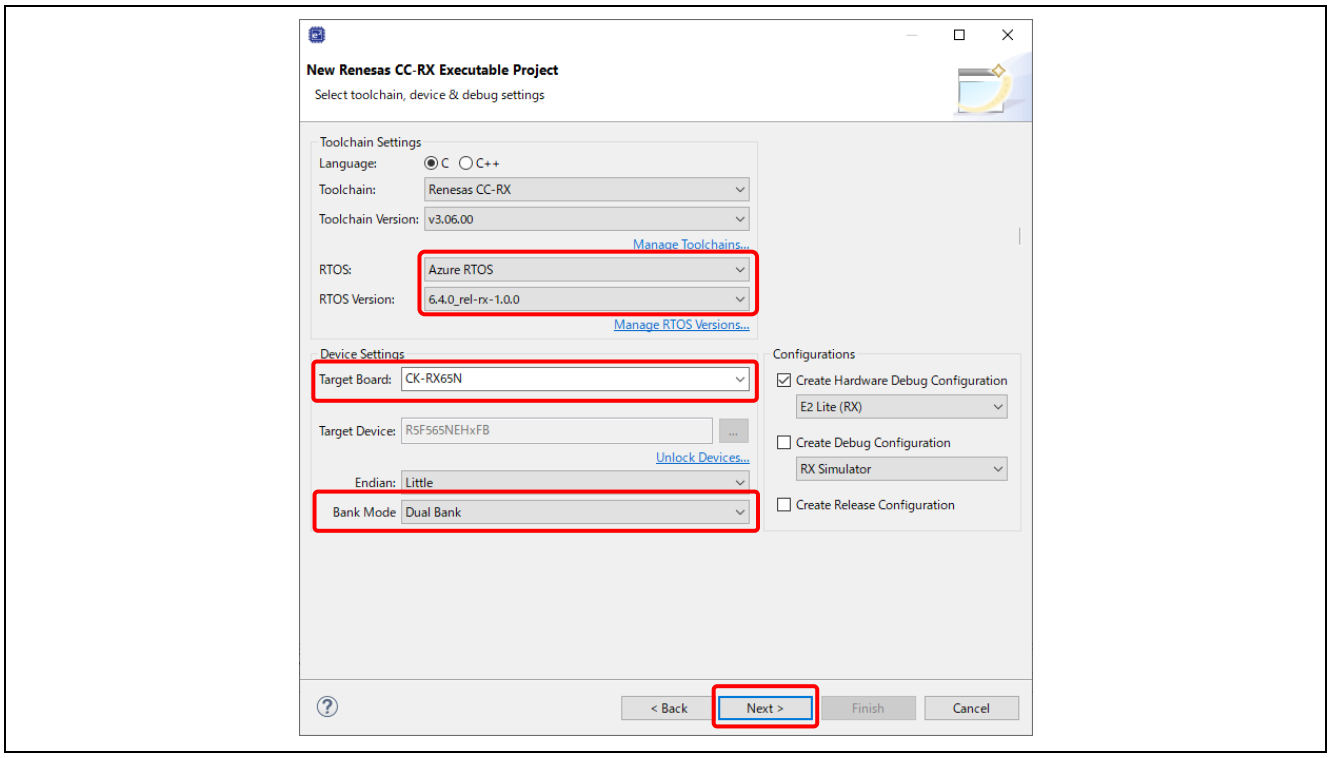

Figure 2.5 Select toolchain, device & debug settings Window

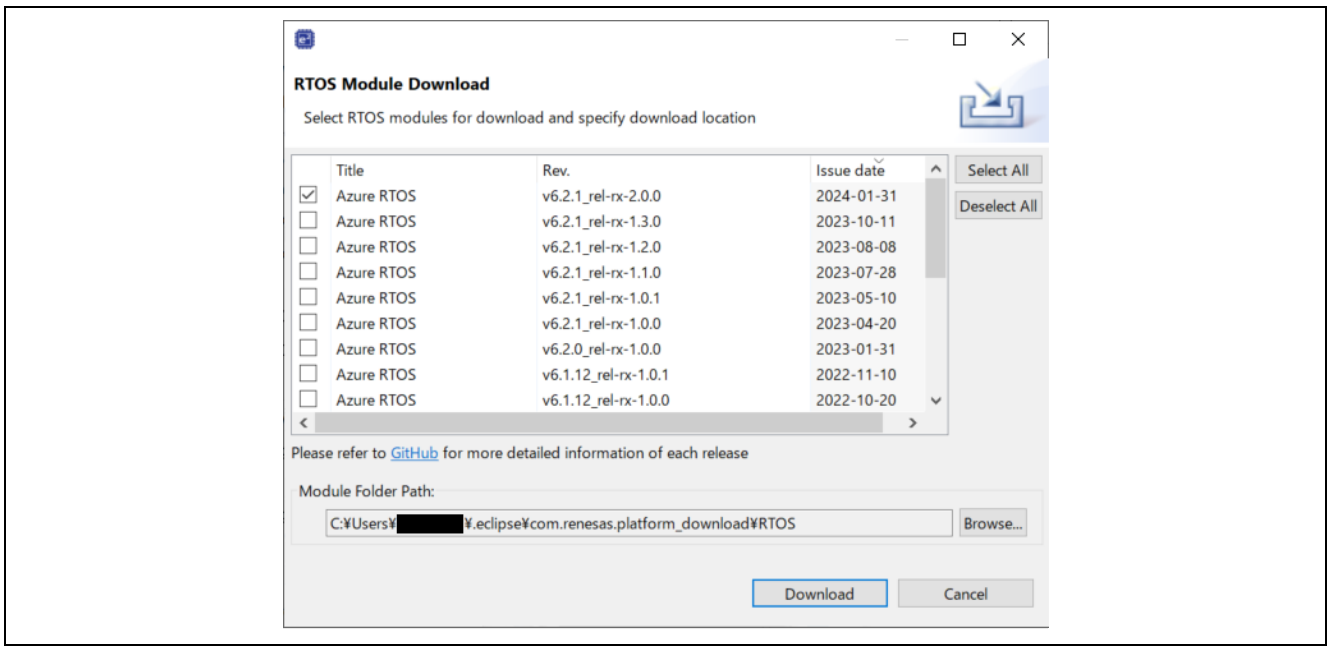

Figure 2.6 RTOS Module Download Window

The **Select Coding Assistant settings** dialog box appears. Click the **Next >** button without making any changes.

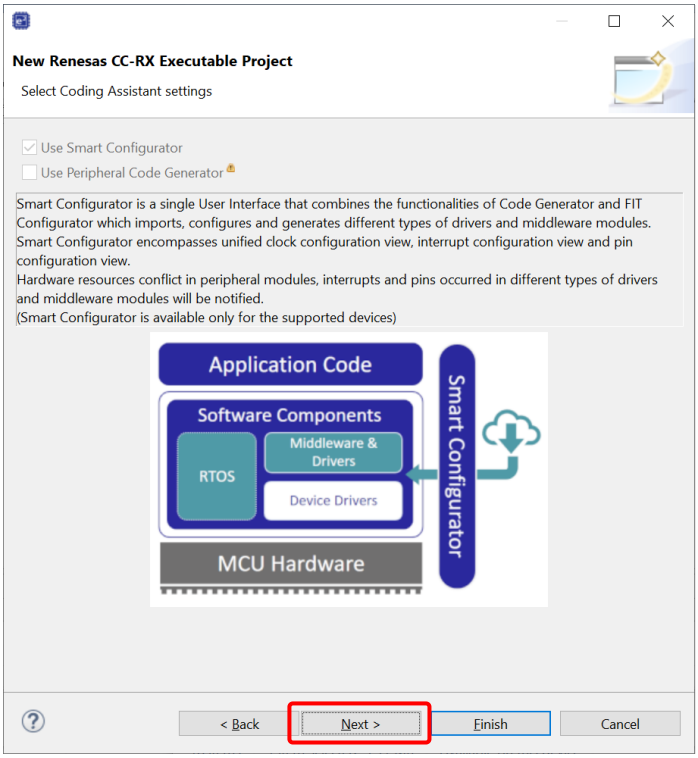

Figure 2.7 Select Coding Assistant settings Window

In the **Select RTOS Project Settings** dialog box a list of sample projects is displayed. Use the scroll bar to scroll down the list, select **Azure Device Update (ADU) sample project**, and click the **Next >** button.

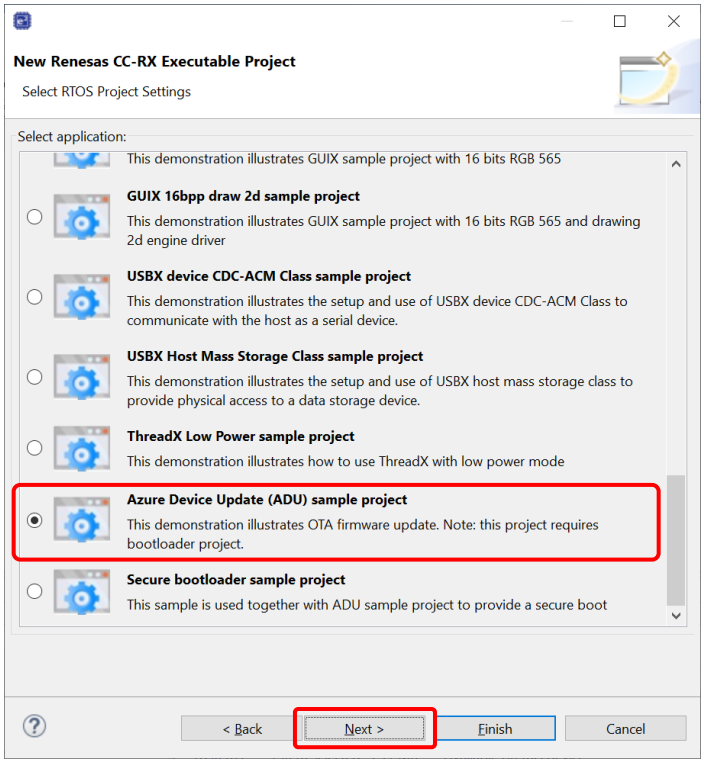

Figure 2.8 Select RTOS Project Settings Window

The **Settings The Contents of Files to be Generated** dialog box appears. Click the **Next >** button without making any changes.

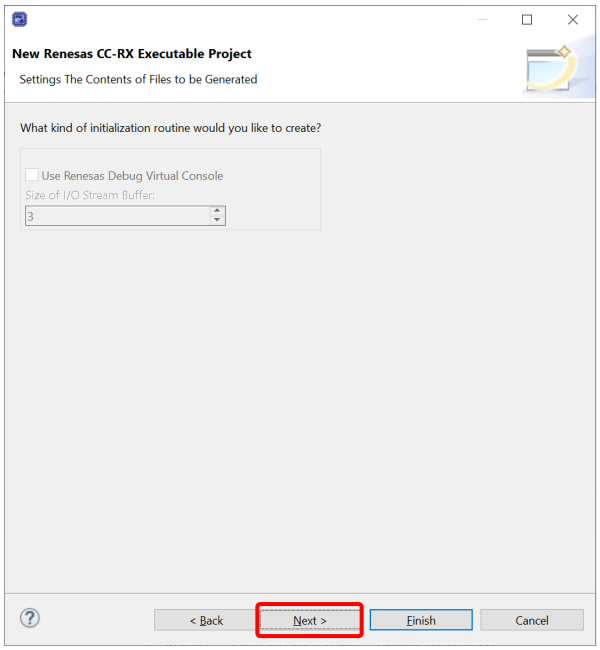

Figure 2.9 Settings The Contents of Files to be Generated Window

A dialog box appears indicating that preparation for creation of the project is complete. If there are no problems, click the **Finish** button.

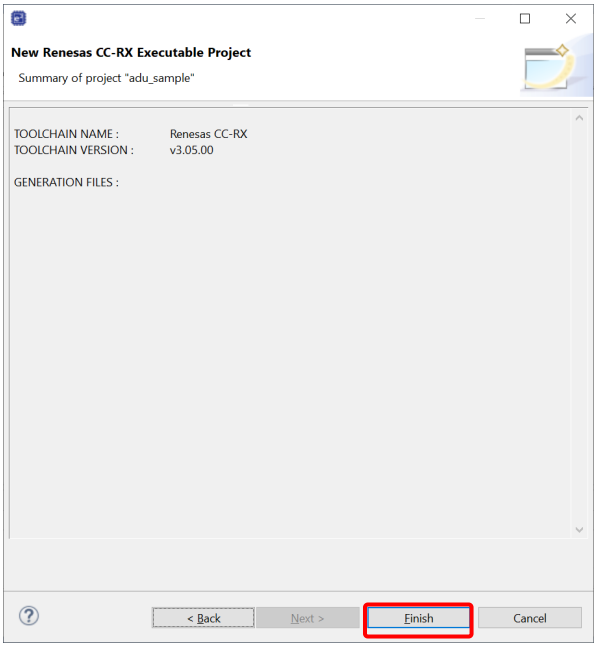

Figure 2.10 Window Indicating Completion of Preparation for Project Creation

If the **Editors available on the Marketplace** dialog box appears, click the **Cancel** button to dismiss it.

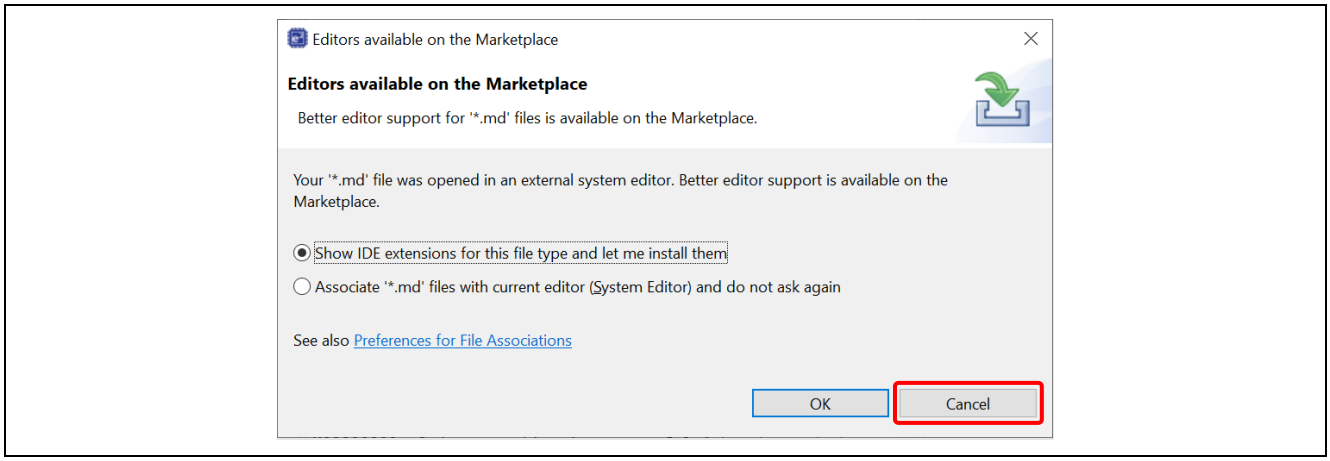

Figure 2.11 Editors available in the Marketplace Window

The project is created in e^2 studio as shown below. If the Project Explorer view is not shown, click the **C/C++** button at the top right of the window and select **Window** → **Show View** → **Project Explorer** from the menu.

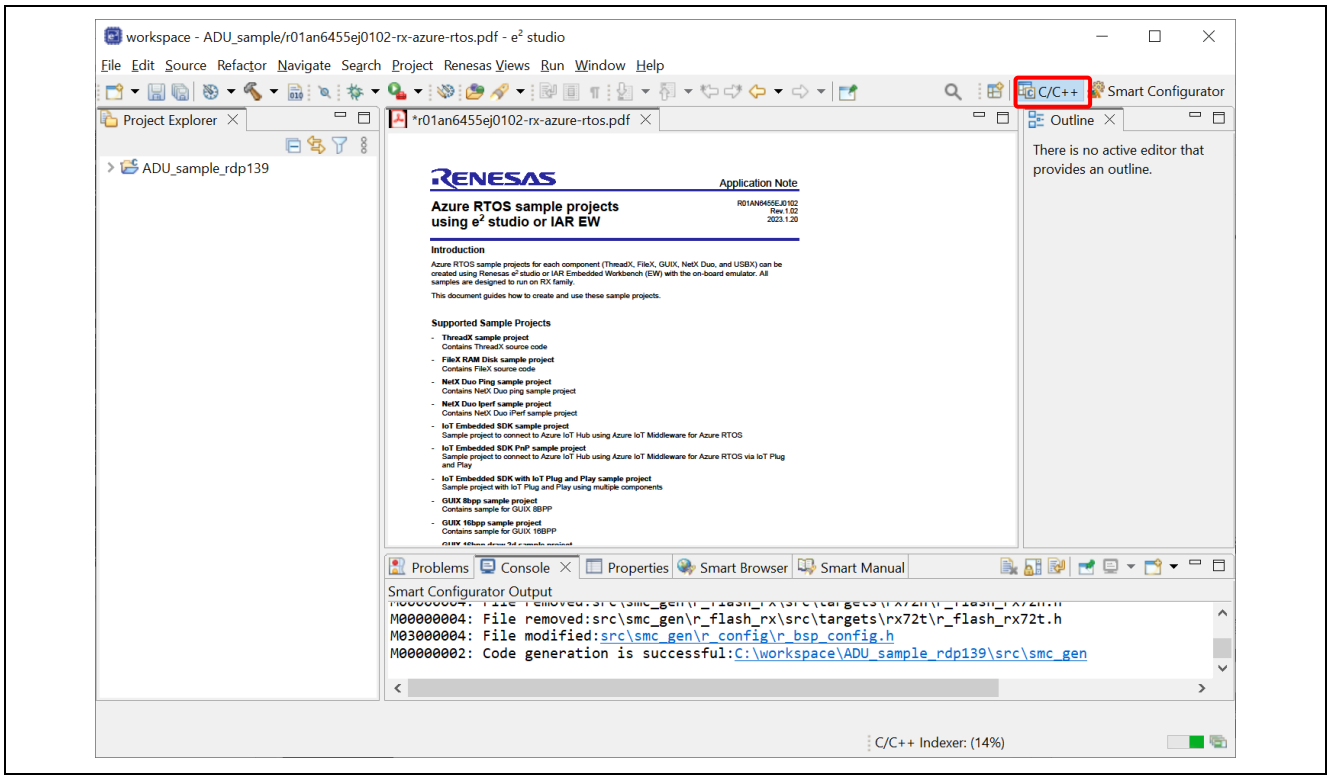

Figure 2.12 Window after Creation of ADU Sample Project

2.2.2 Creating a New Bootloader Sample Project

Follow the same procedure as that used to create the ADU sample project to create a bootloader sample project. The basic steps are the same as those for the ADU sample project. For **Project name:** enter **bootloader**, and in the **Select RTOS Project Settings** dialog box select **Secure bootloader sample project**. All other settings are the same as those for the ADU sample project.

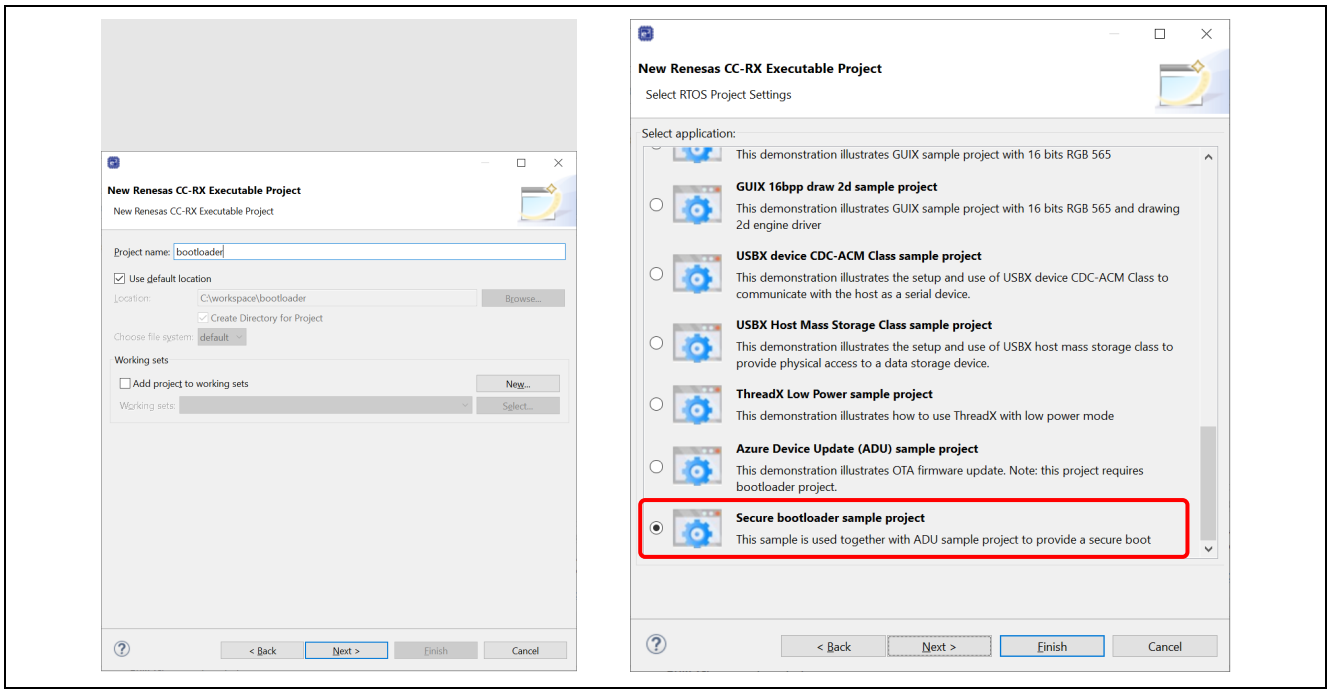

Figure 2.13 Settings for Creating Bootloader Project

2.3 Changing Project Settings

It is necessary to change the settings of the newly created projects in order to implement ADU. Note that the steps described in this section must be performed on both the ADU sample project and the bootloader sample project. The description of the steps below mainly uses the bootloader sample project as an example.

2.3.1 About the Background Operation (BGO) Mode

The MCU of the RX72N, RX671, or RX65N Group mounted on this target board supports BGO (Back Ground Operation) mode in which the content of flash memory can be rewritten in the background.

In the sample project described in this document, BGO is enabled if firmware update module v1 is used, and BGO is disabled if firmware update module v2 is used. (Firmware update module v2 supports only the CK-RX65N.)

If BGO is enabled, writing to flash memory is performed in non-blocking mode. If BGO is disabled, writing to flash memory is performed in blocking mode. For details on BGO mode, refer to “RX Family Flash Module Using Firmware Integration Technology RX Driver Package” (R01AN2184) or “RX65N Group, RX651 Group Flash Memory User’s Manual: Hardware Interface” (R01UH0602).

2.3.2 Integrating Components

When it is first set up, the e² studio environment may not include certain components. In the description below, the firmware update module (FIT) required for ADU is used as an example.

In Project Explorer, expand the **bootloader** project tree and double-click **bootloader.scfg** to open the Smart Configurator perspective for the **bootloader** project. In the Smart Configurator window, select the **Components** tab to open the **Software component configuration** window.

On the left of the window a tree of components that need to be integrated is displayed. The firmware update module corresponds to the **r_fwup** item in the tree.

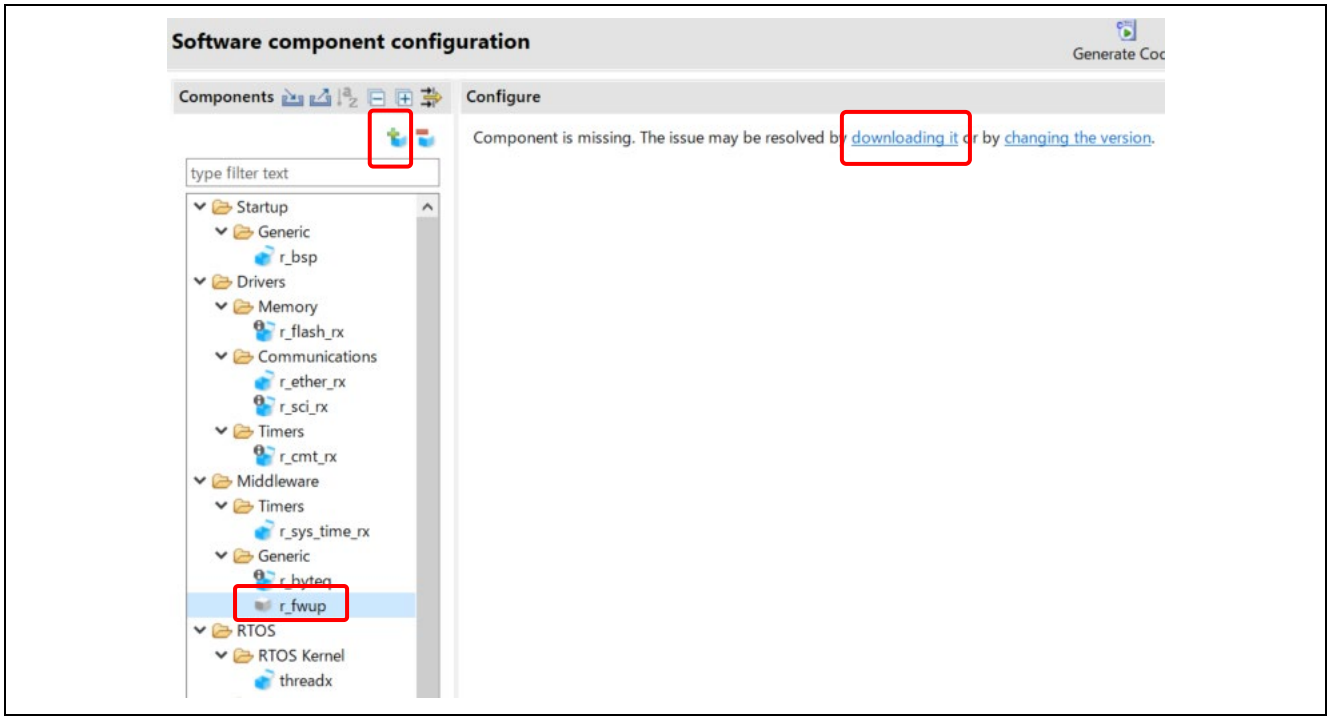

Figure 2.14 Software component configuration Window

If the icon of the desired component is grayed out, the component has not been downloaded to e² studio. Use the procedure described below to download it.

Alternatively, you can also download all missing components from **downloading it**, which is displayed by clicking a component icon that is grayed out.

1. Click the **+** button above the component tree. When the **Software Component Selection** window appears, select **FWUP Library** in the **Components** listing and click the **Finish** button. The gray overlay disappears from the blue icon in the component tree. (In the case of a component that was not originally shown in the tree, a new icon is added to the tree.)
   The version of FWUP Library to be specified differs depending on the target board you use. Specify the appropriate version by referring to the following table.

Table 2-1 Version Compatibility of the Firmware Update Module

Target board	FWUP Library (Firmware Update module)
CK-RX65N	v2.01
Renesas Starter Kit+ for RX65N-2MB RX65N Cloud Kit RX72N Envision Kit RX671 RSK	v1.06

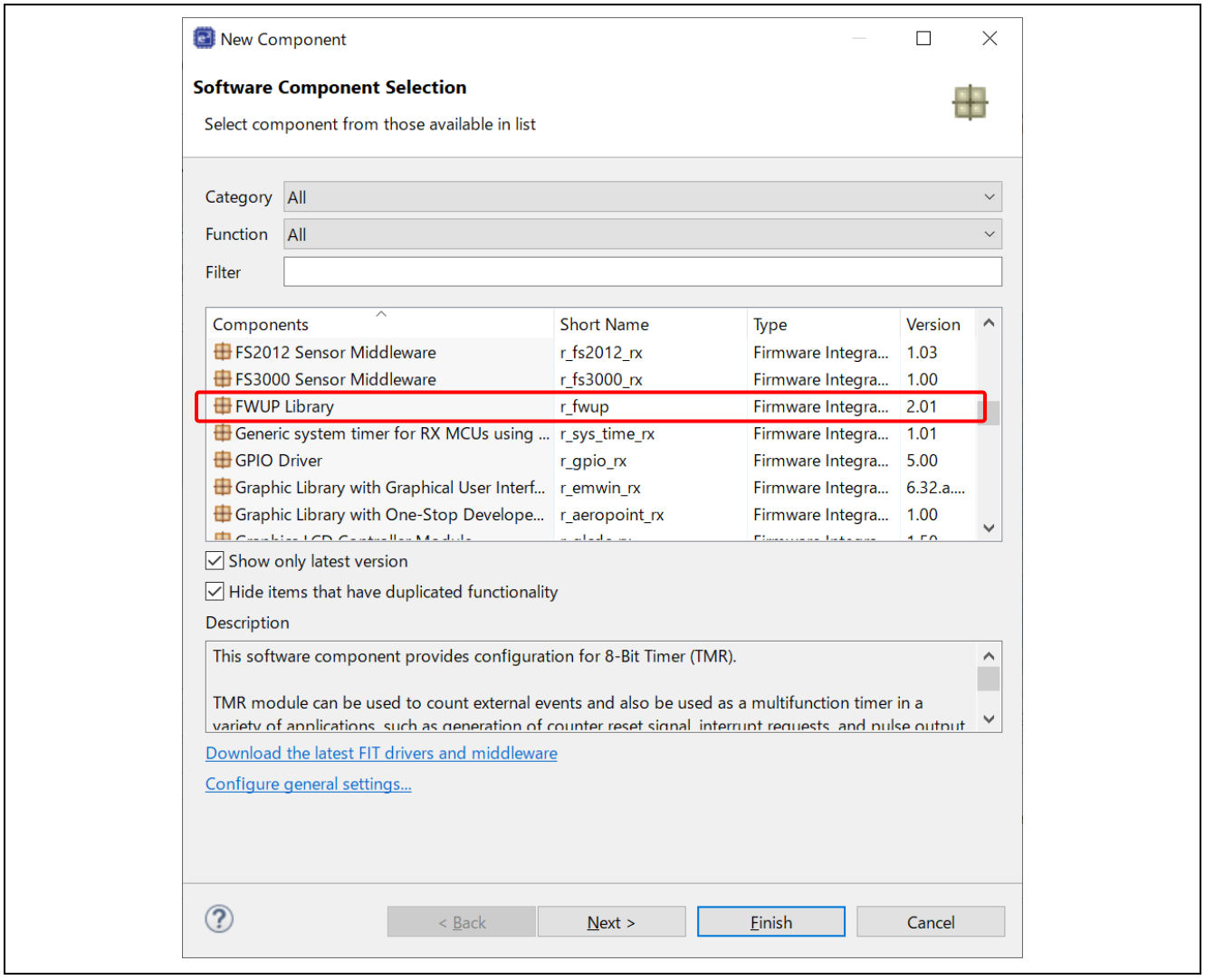

Figure 2.15 Software Component Selection Window

The latest version of the desired component may not appear in the software component list shown in the above figure. In such a case, the desired component can be downloaded by using procedure (1) or (2) shown later.
After the component has been downloaded, you can register it by using the dialog box shown in the above figure.
If the version of the firmware update module you use is v2, the component downloaded by using procedure (1) may not appear in the list. In such a case, use procedure (2).

(1) Using the FIT Module Download window of Smart Configurator to download the components
Click the **+** button above the component tree. In the Software Component Selection window that appears, click **Download the latest FIT drivers and middleware** at the bottom of the window.

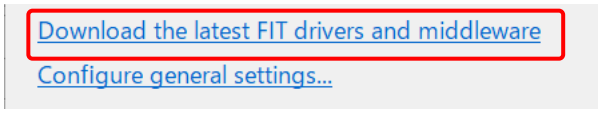

Figure 2.16 Downloading the Latest Versions of FIT Drivers

When the FIT Module Download window appears, clear the **Show RX Driver Package only** check box. Select the check boxes of the necessary modules in the FIT module list, and then click the **Download** button. The selected modules are downloaded and saved in the folder specified in **Module Folder Path**. These modules can be added to the sample project by using the **Software component configuration** window.

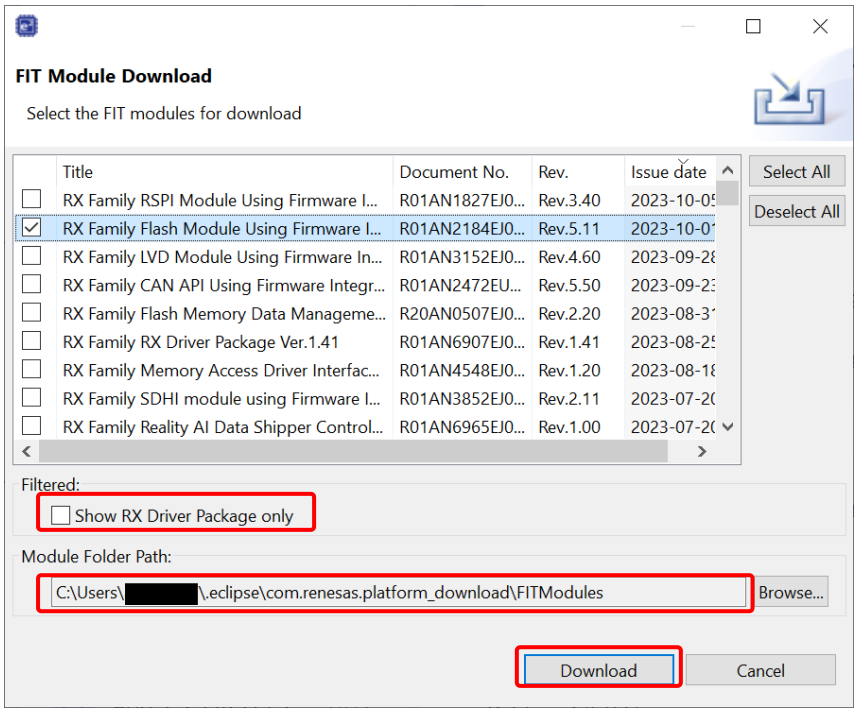

Figure 2.17 FIT Module Download window

(2) Using the Renesas website to download and register the components

If there are modules whose versions are too new to appear in the FIT Module Download window, you can download them from the Renesas website and register them.

The following shows an example procedure applicable when firmware update module V2.01 is used.

a) From the Renesas website, download the sample code for the firmware update module.

b) Uncompress the downloaded ZIP file, and then copy the three files (shown in the following figure) from the FITModules folder to the folder indicated in **Module Folder Path** at the bottom of the FIT Module Download window (see (1) above).
Restart e^2 studio, and the downloaded FIT modules become usable.

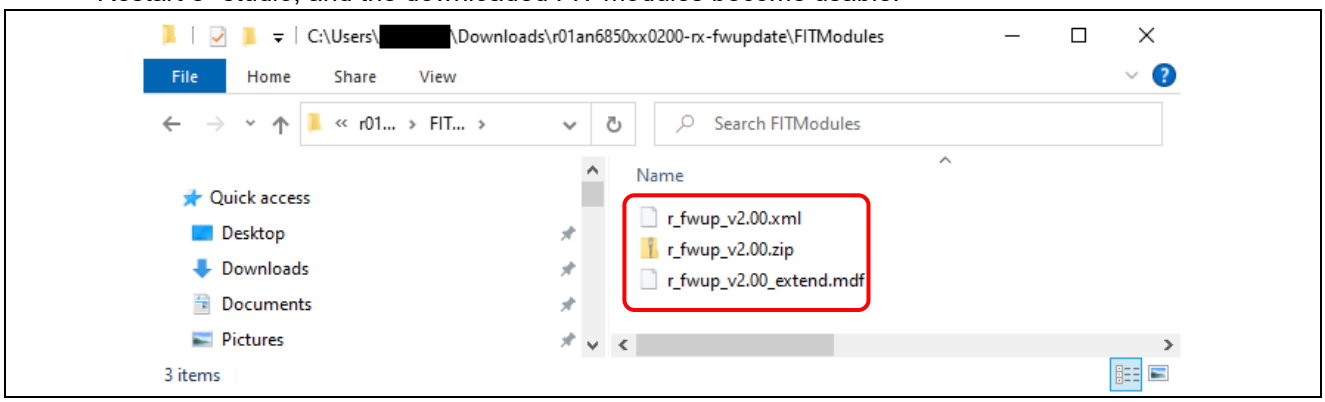

Figure 2.18 Downloaded FIT Module Files

2. After the component has been configured, generate component code.*1 Click the **Generate Code** button at the top right of the **Software component configuration** window. The generated code is stored in the **\src\smc_gen** folder in the project folder.

Figure 2.19 Generate Code Button

If there are other components with a gray overlay on their icons, repeat the above steps for each of them.

Note: 1. After changing settings in Smart Configurator, make sure to generate code as the final step.

Make sure that you also perform the above procedure for the “adu_sample” project.

2.3.3 Settings for Code Generation During Building

In this sample project, the settings additionally specified after code generation may be lost when code generation is performed again.

To prevent all source files from being re-created when you build or clean the project, perform the procedures in 1 to 3 below.

1. In the Project Explorer of e^2 studio, right-click the **bootloader** project, and then, in the context menu that appears, select **Properties**.
   Make sure that you also perform this procedure for the “adu_sample” project.

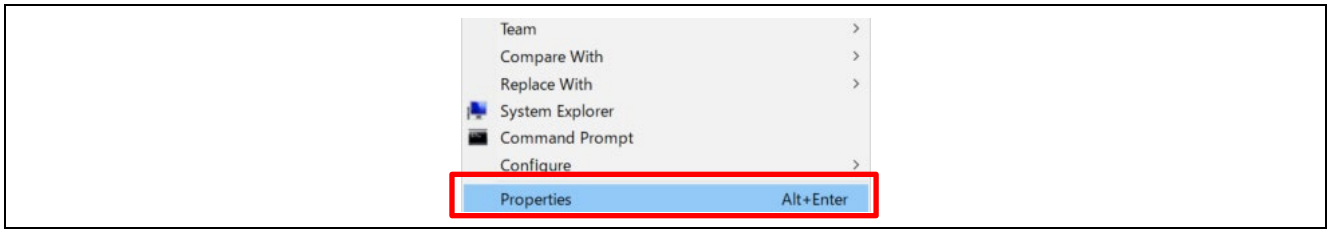

Figure 2.20 Context Menu for a Project

2. In the menu of the Properties dialog box, select **Builders**, select the **SC Code Generation Builder** check box, and then click **Edit**. The window for configuring the builder appears.

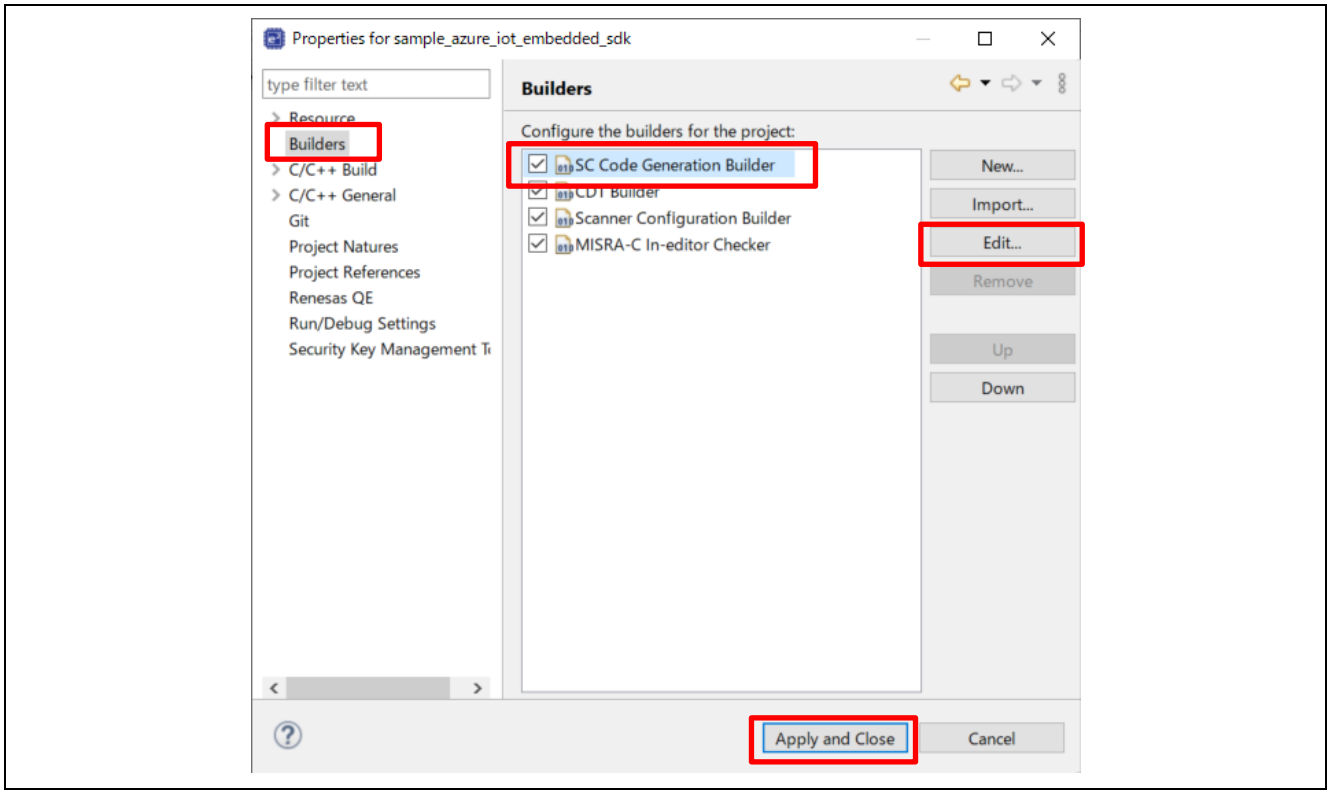

Figure 2.21 Properties Dialog Box for the Project

3. In the Configure Builder dialog box, clear all check boxes, and then click **OK**. Then, in the Properties dialog box, click **Apply and Close**.
   The settings specified here will prevent unexpected code generation. *1

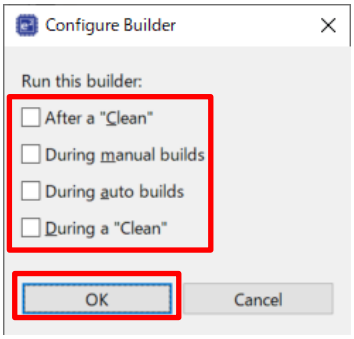

Figure 2.22 Configure Builder Dialog Box

Note: 1. If you want to generate code, click the **Generate Code** button in the Smart Configurator.

2.4 Creating Key Information

This section describes how to generate key information used for settings in the sample project. OpenSSL is used to generate key information.

2.4.1 Installing OpenSSL

Access the Win32/Win64 OpenSSL download site, download the OpenSSL installer that matches your OS version, and run it to install the software.

After installation is complete, add the OpenSSL installation folder to the Path system variable of Windows.

64-bit version: C:\Program Files\OpenSSL-Win64\bin

In Windows 10, to access the system variable, select:
Settings > **System** > **About** > **Advanced system settings** > **System Properties** > **Advanced** tab > **Environment variables** > **System variables**

To execute commands of OpenSSL, from the Windows Start menu, start the Win64 OpenSSL Command Prompt application.

Note: The name of the application to start differs depending on the Windows architecture. This is an example in the 64-bit edition.

2.4.2 Generating a Key Pair for ECC in OpenSSL

When creating firmware, use public and private keys for ECC. You can use OpenSSL to generate these keys. Enter the character strings shown below in blue, substituting appropriate values of your choice for the input values shown, at the command prompt.

Some commands require you to input settings. Enter the character strings shown below in blue.

If you just want to generate the ECC public and private keys, steps B, E, and F are sufficient.

A. Create a CA Certificate

```
openssl ecparam -genkey -name secp256r1 -out ca.key
using curve name prime256v1 instead of secp256r1

openssl req -x509 -sha256 -new -nodes -key ca.key -days 3650 -out ca.crt
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
- - -
Country Name (2 letter code) [AU]:JP
State or Province Name (full name) [Some-State]:Tokyo
Locality Name (eg, city) []:Kodaira
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Renesas Electronics
Organizational Unit Name (eg, section) []:Software Development Division
Common Name (e.g. server FQDN or YOUR name) []:Renesas Tarou
Email Address []:Tarou.Renesas@sample.com
```

B. Generate a Key Pair for Elliptic Curve Cryptography (Parameter: secp256r1)

```
openssl ecparam -genkey -name secp256r1 -out secp256r1.keypair
using curve name prime256v1 instead of secp256r1
```

C. Create a Certificate for the Key Pair

```
openssl req -new -sha256 -key secp256r1.keypair > secp256r1.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
- - -
Country Name (2 letter code) [AU]:JP
State or Province Name (full name) [Some-State]:Tokyo
Locality Name (eg, city) []:Kodaira
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Renesas Electronics
Organizational Unit Name (eg, section) []:Software Development Division
Common Name (e.g. server FQDN or YOUR name) []:Renesas Tarou
Email Address []:Tarou.Renesas@sample.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

D. Use the CA Certificate to Create a Certificate for the Key Pair

```
openssl x509 -req -sha256 -days 3650 -in secp256r1.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
secp256r1.crt
Certificate request self-signature ok
subject=/C=JP/ST=Tokyo/L=Kodaira/O=Renesas Electronics/OU=Software Development
Division/CN= Renesas Tarou/emailAddress= Tarou.Renesas@sample.com
```

E. Extract a Private Key for Elliptic Curve Cryptography (Parameter: secp256r1)

```
openssl ec -in secp256r1.keypair -outform PEM -out secp256r1.privatekey
read EC key
writing EC key
```

F. Extract a Public Key for Elliptic Curve Cryptography (Parameter: secp256r1)

```
openssl ec -in secp256r1.keypair -outform PEM -pubout -out secp256r1.publickey
read EC key
writing EC key
```

2.4.3 Entering a Public Key

Open **\src\key\code_signer_public_key.h** from the **bootloader** project and the **secp256r1.publickey** file generated by OpenSSL in a text editor. Copy the contents of **secp256r1.publickey** to **CODE_SIGNENR_PUBLIC_KEY_PEM**.

Note that each line must be enclosed in straight quotes ("") and end with the backslash character (\).

If you are using firmware update module v2, you must also perform the same key information entry task for “\src\key\code_signer_public_key.h” of the “adu_sample” project.

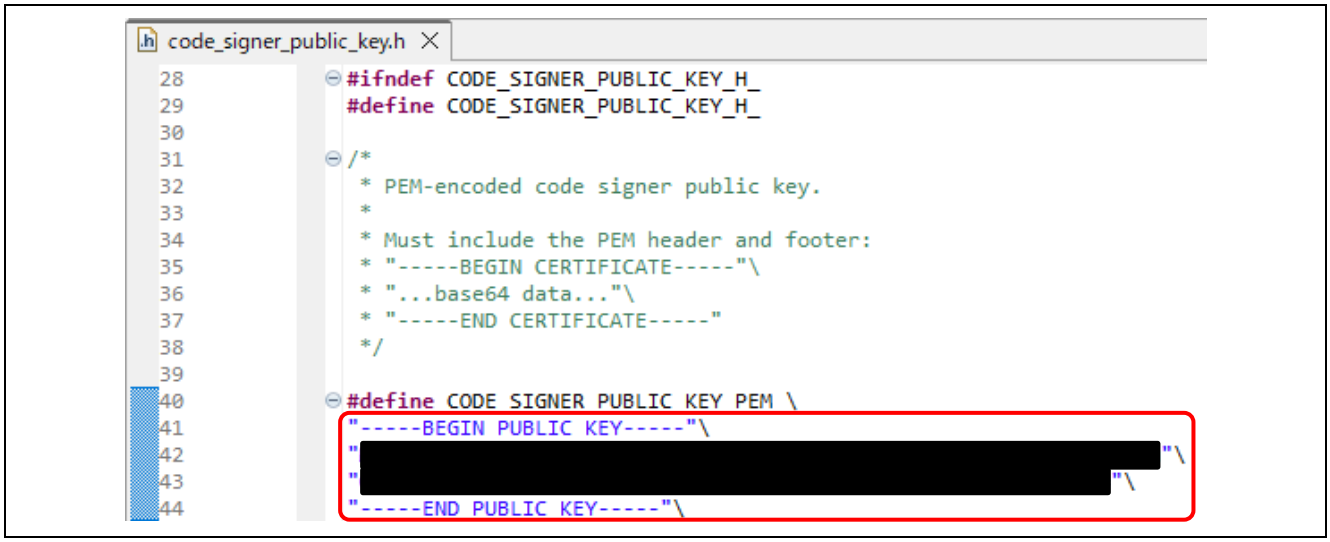

Figure 2.23 Public Key Information Setting

2.5 Building the bootloader Project

Build the **bootloader** project and create a **bootloader.mot** file.

The MOT file is created in the following folder.

\bootloader\HardwareDebug/

2.6 Connection Information Macro Settings

Open **\src\sample_config.h** from the **adu_sample** project and specify setting values for **HOST_NAME**, **DEVICE_ID**, and **DEVICE_SYMMETRIC_KEY**. For the setting values, refer to the parameters configured on the Azure portal as described in 3.1, IoT Hub and Device Registration.

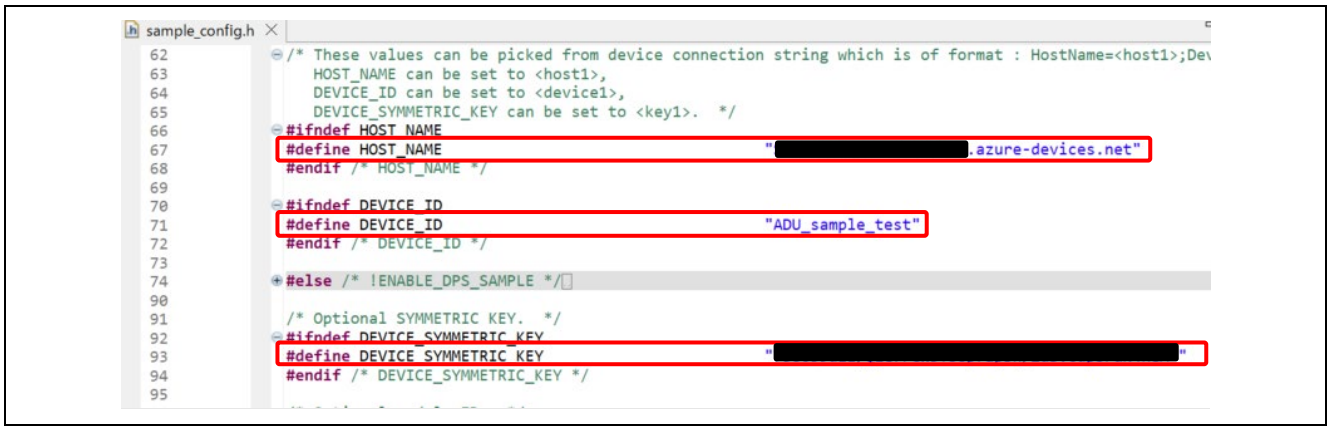

Figure 2.24 Azure Connection Information Settings

It is necessary to configure Wi-Fi connection settings in order to use RX65N Cloud Kit. Open **\src\hardware_setup.h** from the **adu_sample** project and specify setting values for **WIFI_SSID** and **WIFI_PASSWORD**. For the setting values, refer to the SSID and password of the Wi-Fi access point you wish to connect to.

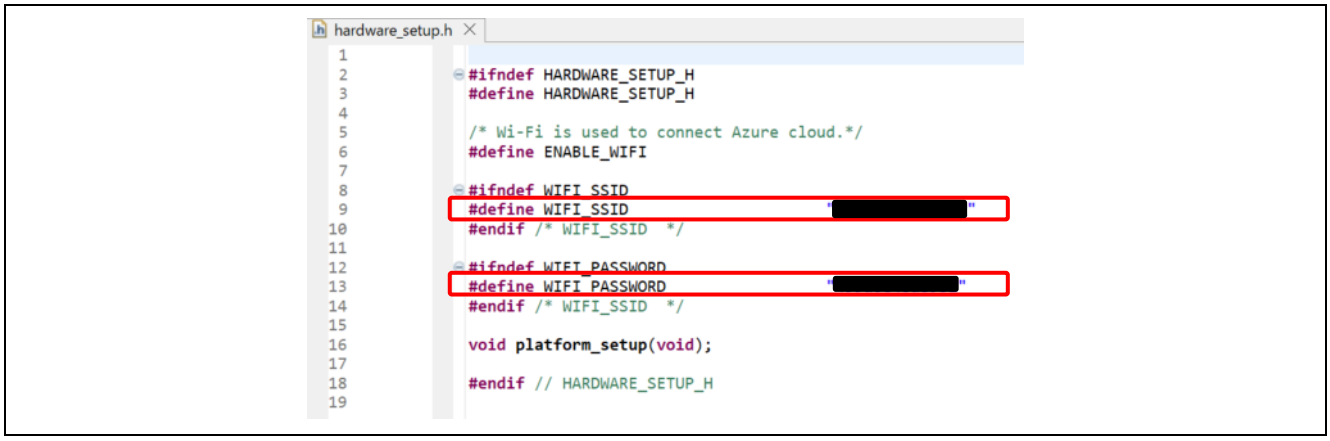

Figure 2.25 Wi-Fi Connection Information Settings

2.7 Checking the Initial Firmware Version

Confirm that the initial firmware version is 1.0.0. Open **\src\sample_azure_iot_embedded_sdk_adu.c** from the **adu_sample** project and check to confirm that the value of **SAMPLE_DEVICE_INSTALLED_CRITERIA** is **1.0.0**.

Figure 2.26 Initial Firmware Version

2.8 Building the adu_sample Project (for the Initial Firmware)

Build the **adu_sample** project and create a **adu_sample.mot** file.

The MOT file is created in the following folder.

\adu_sample\HardwareDebug\

2.9 Creating the Initial Firmware

Create the initial firmware to be downloaded to the target board.

When creating firmware, the creation procedure and the MOT file conversion tool to be used differ depending on whether the version of the firmware update module you use is v1 or v2. Refer to the procedure appropriate for the version of the firmware update module you use.

2.9.1 If Using Firmware Update Module v1

By using the MOT file conversion tool, merge the MOT files of “bootloader” and “adu_sample” to create an initial firmware image.
Use Renesas Secure Flash Programmer as the MOT file conversion tool.

Access the Renesas Secure Flash Programmer (RX MCUs mot file converter 2.0.2), download the **Source Code(zip)** and unzip it at a folder of your choice.

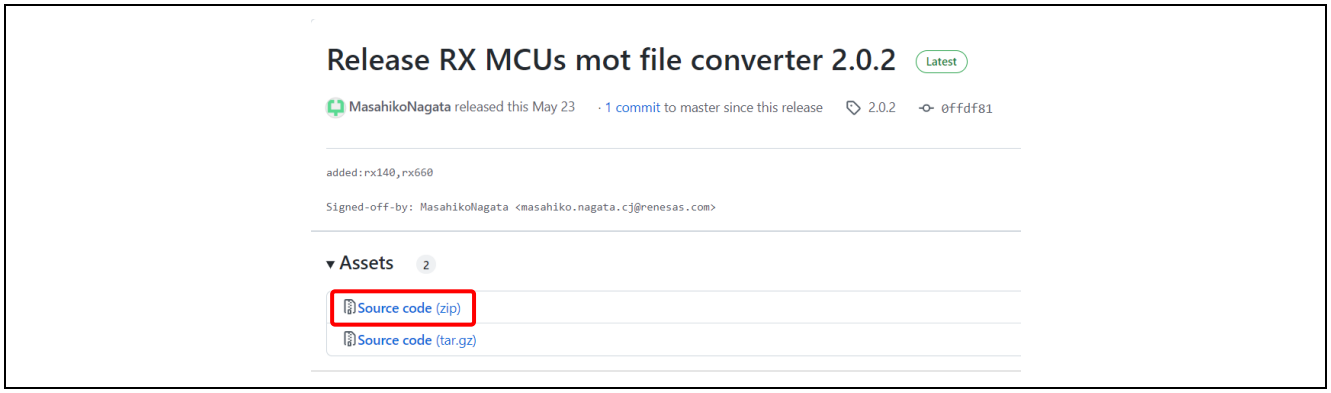

Figure 2.27 Renesas Secure Flash Programmer Download Page

After the download completes, double-click the file **mot-file-converter-2.0.2\Renesas Secure Flash Programmer\bin\Debug\Renesas Secure Flash Programmer.exe** to launch the program.

When you start Renesas Secure Flash Programmer, the following window appears:

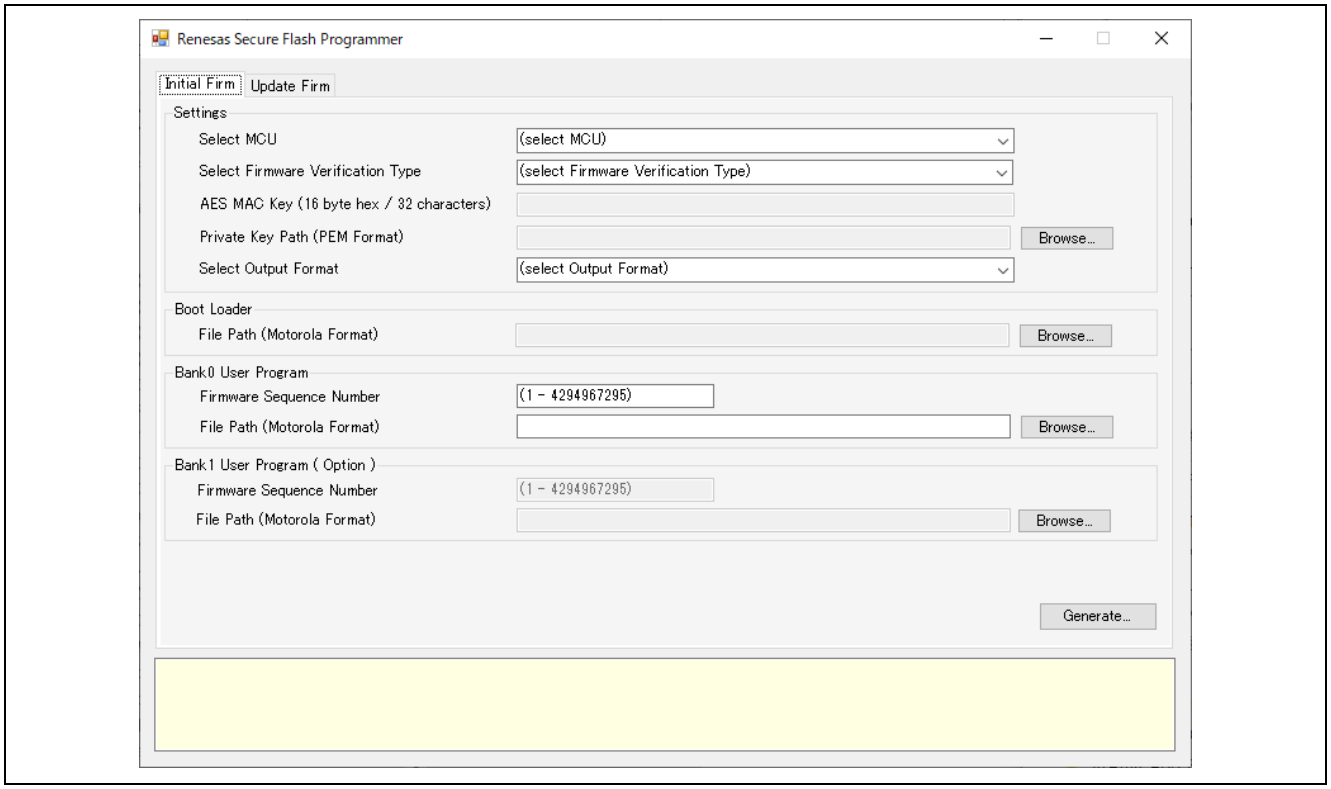

Figure 2.28 Renesas Secure Flash Programmer Window

In this window, click the **Initial Firm** tab, and then specify the settings as follows. The values to be specified are shown in blue font.

- Select MCU: RX65N Flash(Code=2MB, Data=32KB)/Secure Bootloader=64KB
- Select Firmware Verification Type: sig-sha256-ecdsa
- Private Key Path (PEM format): secp256r1.privatekey generated by OpenSSL in step E
- Select Output Format: Bank 0 User Program + Boot Loader (Motorola S Format)
- Boot Loader File Path (Motorola Format): \bootloader\HardwareDebug\bootloader.mot
- Firmware Sequence Number: 1
- Bank 0 User Program File Path (Motorola format): \adu_sample\HardwareDebug\adu_sample.mot

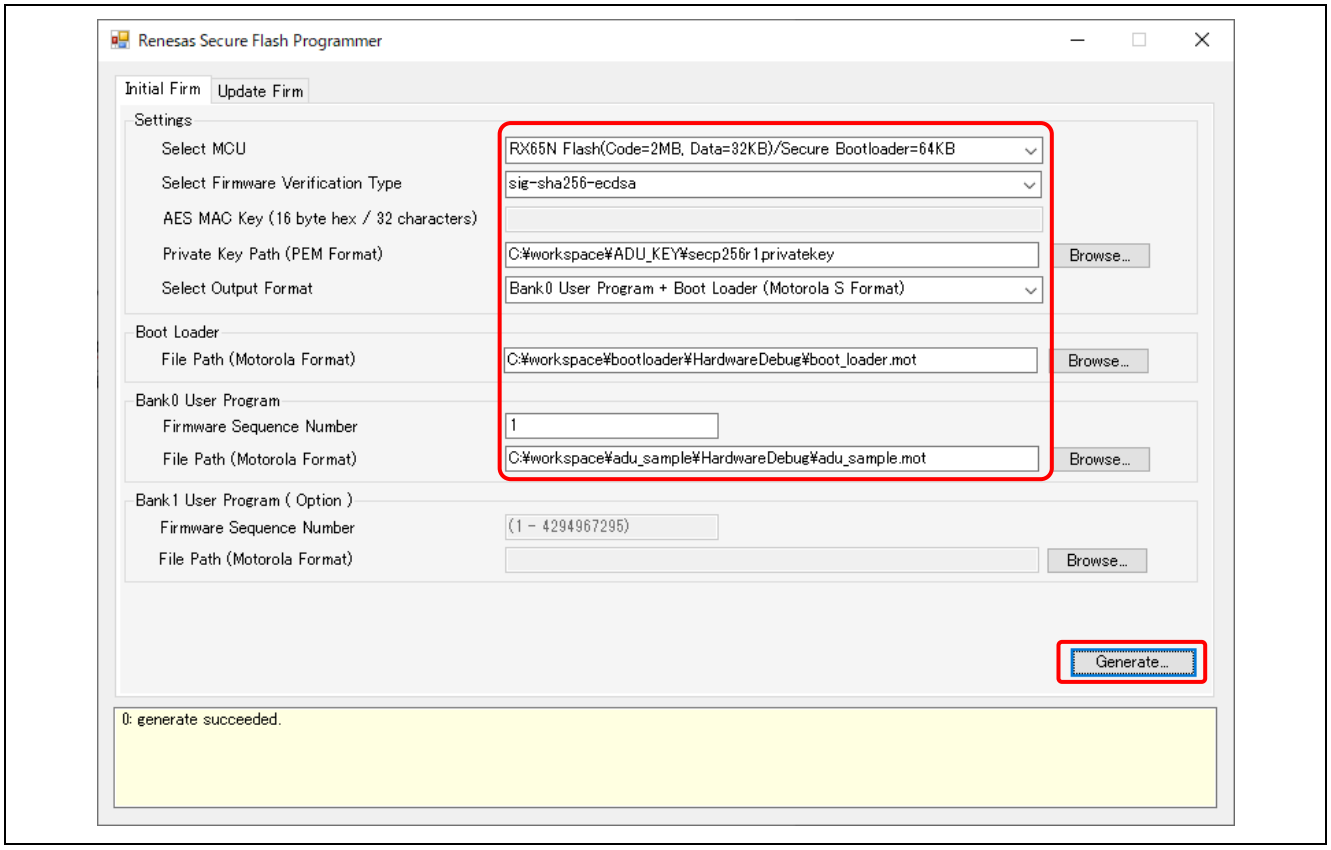

Figure 2.29 Initial Firmware Creation Window

Next, click the **Generate...** button, and then, in the following window that appears, set any output destination folder of your choice, and enter the name of the output file. In this example, **userprog.mot** is entered as the output file name.

Click the **Save** button. The MOT file of the initial firmware is output to the specified folder.
The process is complete when the message “generate succeeded.” appears at the bottom of the window.

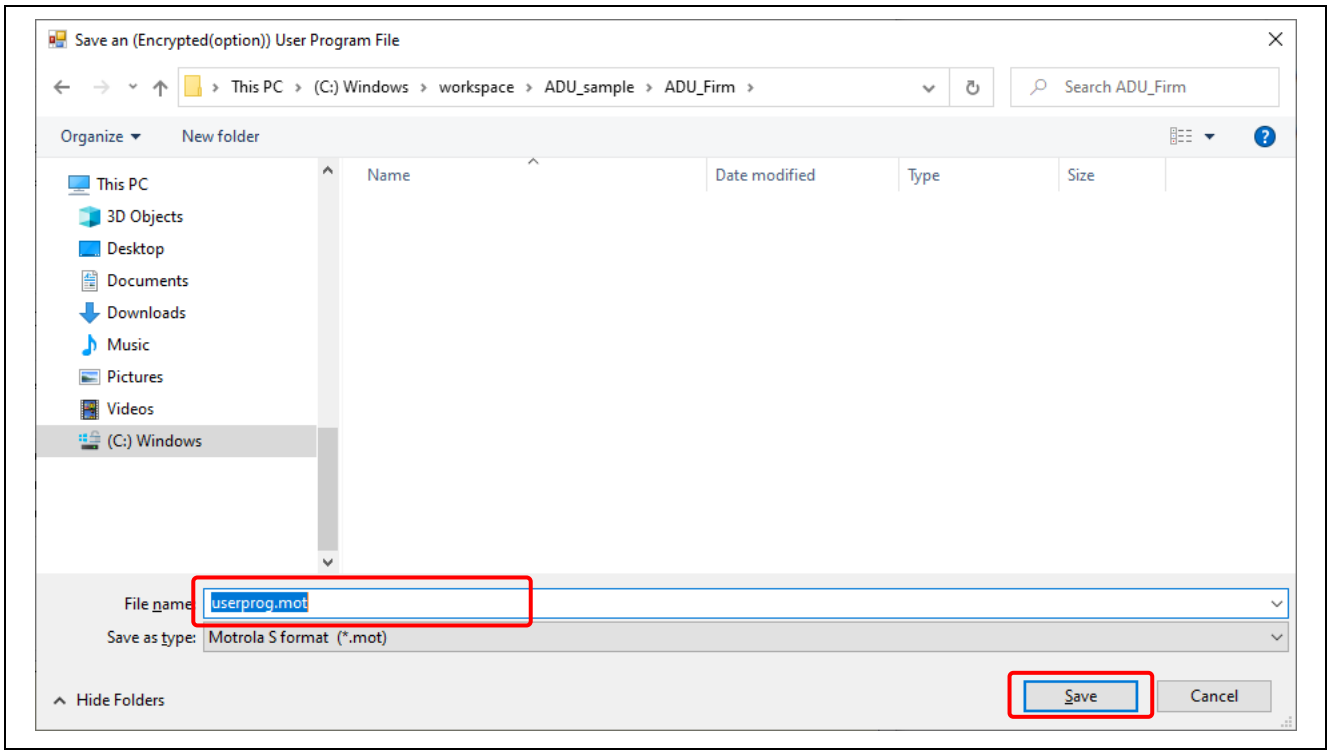

Figure 2.30 Outputting the MOT File of the Initial Firmware

The following figure shows the configuration of the “userprog.mot” file generated by Renesas Secure Flash Programmer.
The verification data, initial firmware, and bootloader have been merged into this MOT file.
The verification data contains the firmware address and other information required for verification.

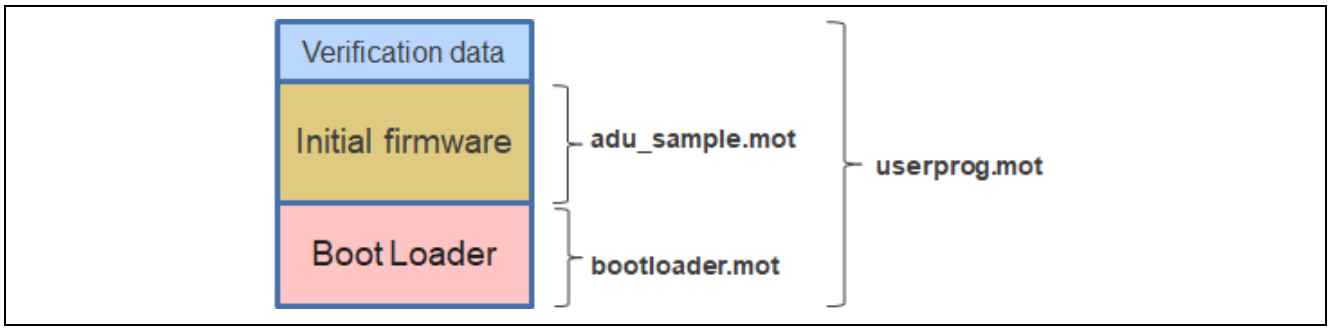

Figure 2.31 File Configuration of userprog.mot (for Firmware Update Module v1)

2.9.2 If Using Firmware Update Module v2

By using the MOT file conversion tool, merge the MOT files of **bootloader** and **adu_sample** to create an initial firmware image.

Use Renesas Image Generator as the MOT file conversion tool.

1. Install Python.

Before you can use Renesas Image Generator, you must prepare the Python runtime environment. Download Python from the Python download website, and then install Python. We have confirmed correct operation with Python 3.12.0.

After installation is complete, add the Python installation folder path (shown below) to the Path system variable of Windows.
If the version of Python is v3.12.0, the path to be added is as follows:
C:\Users*xxxx*\AppData\Local\Programs\Python\Python312

Note: Replace *xxxx* by your Windows user name. The location of the installation folder may differ depending on the installation environment.

In Windows 10, to access the system variable, select:
Settings > **System** > **About** > **Advanced system settings** > **System Properties** > **Advanced** tab > **Environment variables** > **System variables**

When you have installed Python, install the Python encryption library (pycryptodome).
The encryption library is downloaded and then installed by entering the following command from the command line:

```
> python -m pip install pycryptodome
```

When installation is complete, the message “Successfully installed pycryptodome-x.xx.x” appears.

2. Create an initial firmware image.

Create an image file of the initial firmware by using the following procedure:

(1) Open the “RenesasImageGenerator” folder that is bundled with the sample project. This folder is located in the “tools” folder of the sample project.
This folder contains three Renesas Image Generator files for ADU:

```
tools
|--RenesasImageGenerator
   |-- image-gen.py
   |-- RX65N_DualBank_Initial_PRM.csv
   |-- RX65N_DualBank_Update_PRM.csv
```

The table below provides a brief explanation of each file contained in the folder.

File name	Explanation
image-gen.py	Renesas Image Generator application (Python code)
RX65N_DualBank_Initial_PRM.csv	File containing the parameters for creating the initial firmware
RX65N_DualBank_Update_PRM.csv	File containing the parameters for creating an update firmware

(2) Copy the following three files to the “RenesasImageGenerator” folder:

Private key	secp256r1.privatekey	Created in section 2.4.2, Generating a Key Pair for ECC in OpenSSL
Bootloader	bootloader.mot	Created in section 2.5, Building the bootloader Project
Initial firmware	adu_sample.mot	Created in section 2.8, Building the adu_sample Project (for the Initial Firmware)

Make sure that you copy the files to the “RenesasImageGenerator” folder in the following structure. The files in red font are the files you copy.

```
RenesasImageGenerator
|-- image-gen.py
|-- RX65N_DualBank_Initial_PRM.csv
|-- RX65N_DualBank_Update_PRM.csv
|-- secp256r1.privatekey
|-- bootloader.mot
|-- adu_sample.mot
```

(3) Open the Command Prompt window, and then change the current directory to the “RenesasImageGenerator” folder.

(4) Execute the following command from the command prompt. An image file of the initial firmware will be created.

```
> python image-gen.py -iup adu_sample.mot -ip RX65N_DualBank_Initial_PRM.csv -o
userprog -ibp bootloader.mot -vt ecdsa
```

Creation takes some time.
When the message “Successfully generated the userprog.mot file” appears, creation is complete.
The initial firmware image “userprog.mot” is output to the “RenesasImageGenerator” folder.

The following figure shows the configuration of the “userprog.mot” file generated by Renesas Image Generator.

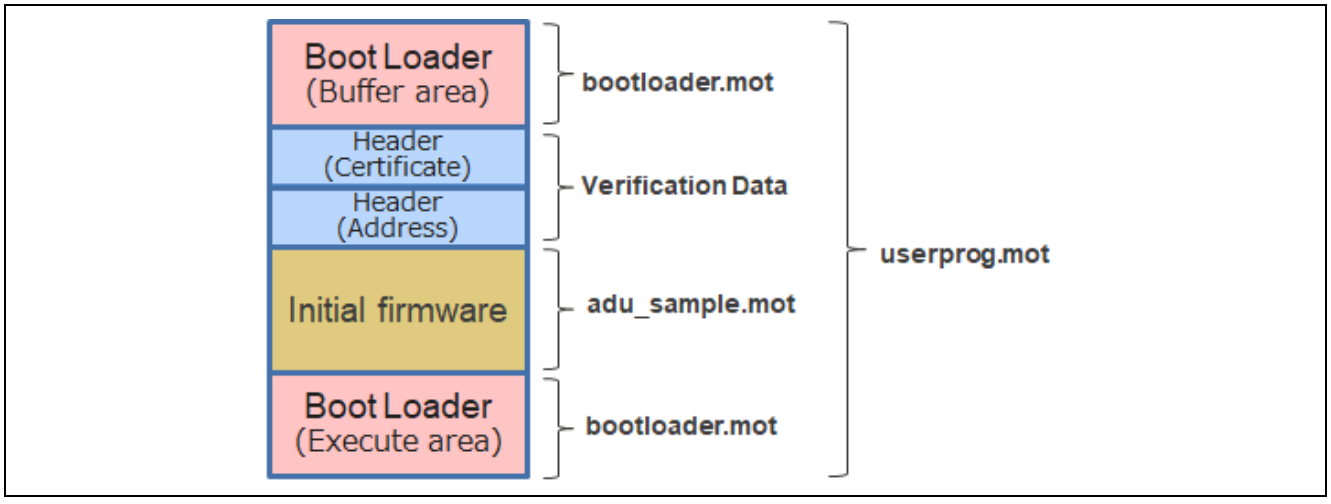

Figure 2.32 File Configuration of userprog.mot (for Firmware Update Module v2)

The initial firmware image “userprog.mot” is a MOT file that contains headers, the initial firmware, and bootloaders.
The “Header (Certificate)” area stores the information for verifying the firmware and the “Header (Address)” area stores the address information of the firmware.

Note that this file also contains the bootloader in the buffer area.

2.10 Installing the Flash Programming Tool

Access the Renesas Flash Programmer download site, download the Renesas Flash Programmer V3.11.02 Windows installer, and run it to install the tool.

2.11 Programming the Initial Firmware

A) Flash Programming of Initial Firmware (Linear Mode → Dual Mode)

Program the initial firmware on the target board.
In this procedure, you change the mode of the RX65 board to dual mode from the (initial) linear mode, and then program the firmware.
First, launch **flash_project.rpj**, which is located in the **\adu_sample\tools\Flash_Project\CKRX65N_ADU_Write** folder. Next, on the **Operation** tab specify **userprog.mot** for **Program File** and click the **Start** button to program the previously generated initial firmware file **userprog.mot** to the RX65N.
Programming is complete when the message “Operation completed.” appears at the bottom of the window.

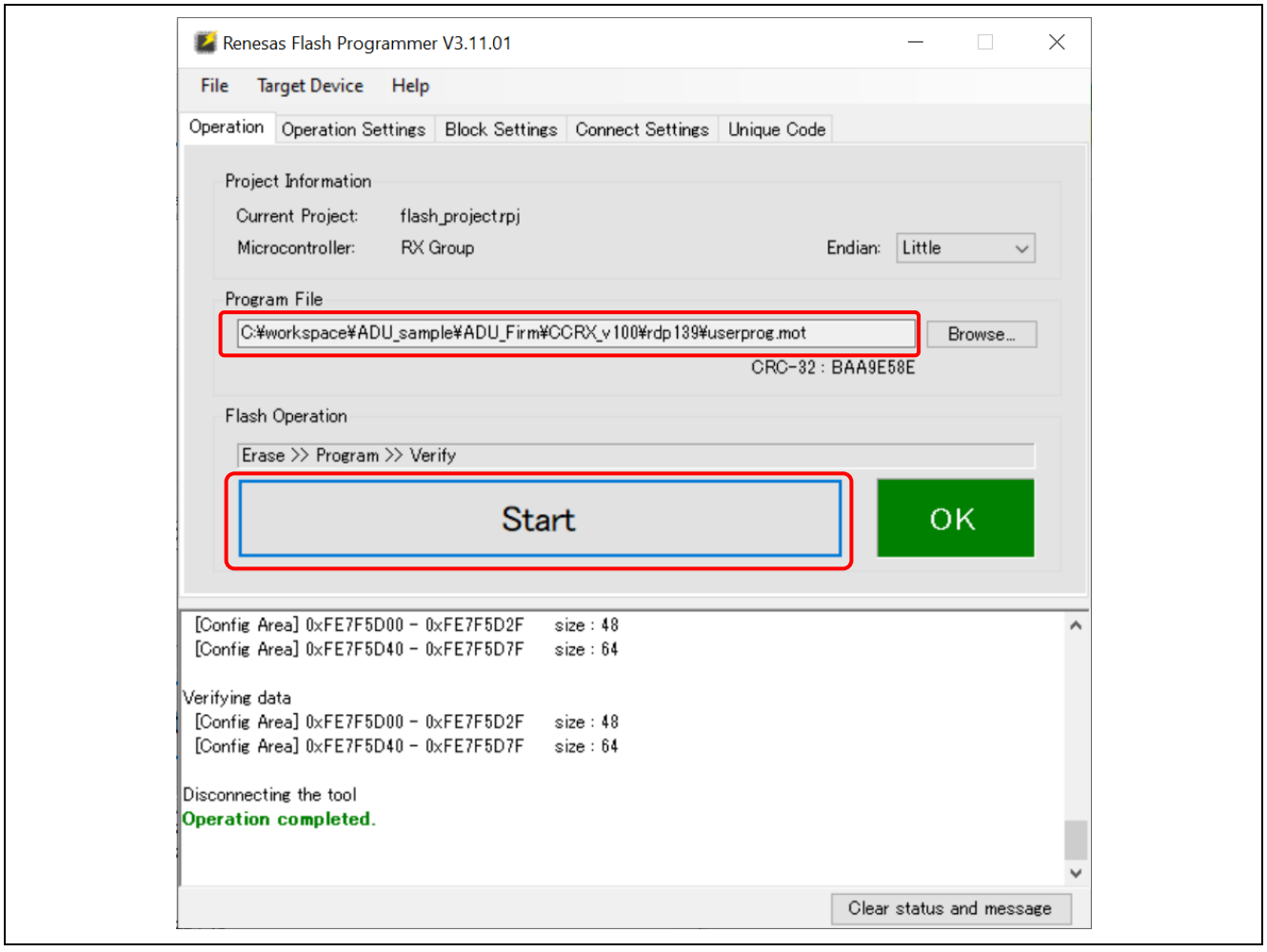

Figure 2.33 Programming the Initial Firmware to the Device

Running the initial firmware transitions the MCU from linear mode(normal mode) to dual mode (a mode in which the code flash memory is divided into two banks). Subsequently, if it is necessary once again to program the initial firmware to the flash memory, first perform B) to erase the flash memory and change to linear mode, then A) to program the flash memory.

B) Flash Erasing (Dual Mode → Linear Mode)
The ROM is initialized by "Flash Erasing", and the RX65N changes from dual mode to linear mode. Launch erase_project.rpj in the **CKRX65N_ADU_Erase** folder in tools/Flash_Project and click the Start button on the Operation tab to erase the chip.

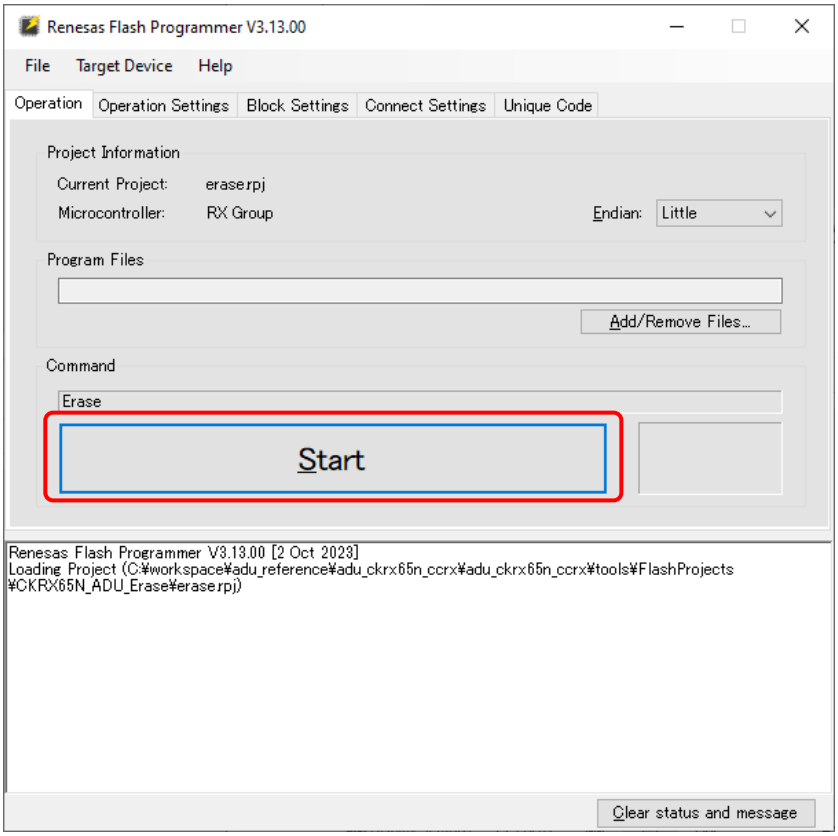

Figure 2.34 Erasing the Flash Memory and Changing the Mode to Linear Mode

Renesas Flash Programmer recognizes linear mode and dual mode as separate MCUs. In procedure A, the programming of flash memory starts after the device recognizes that it is placed in linear mode. In procedure B, erasure of flash memory starts after the device recognizes that it is placed in dual mode. Therefore, if either procedure A or B is repeated in succession, the device information becomes inconsistent, resulting in the E3000107 error (the device and connection information do not agree).

2.12 Executing the Initial Firmware

After programming of the initial firmware is complete, the program operates on the RX65N.
For the actual operation procedure, refer to section 3.8.1, Execution on the Target Board.
The following shows the program startup sequence from the programming.

1. The program operates after programming of the initial firmware is complete.
2. After the program runs, the bootloader runs and decrypts the encrypted hash value using the public key that was programmed to the verification data area.
3. The program calculates the hash value for the entire firmware, and checks whether it is identical to the decrypted hash value.
   If the values match, it launches the firmware.
   If you are using firmware update module v1, the bootloader is copied to the buffer area after verification of the firmware is complete.

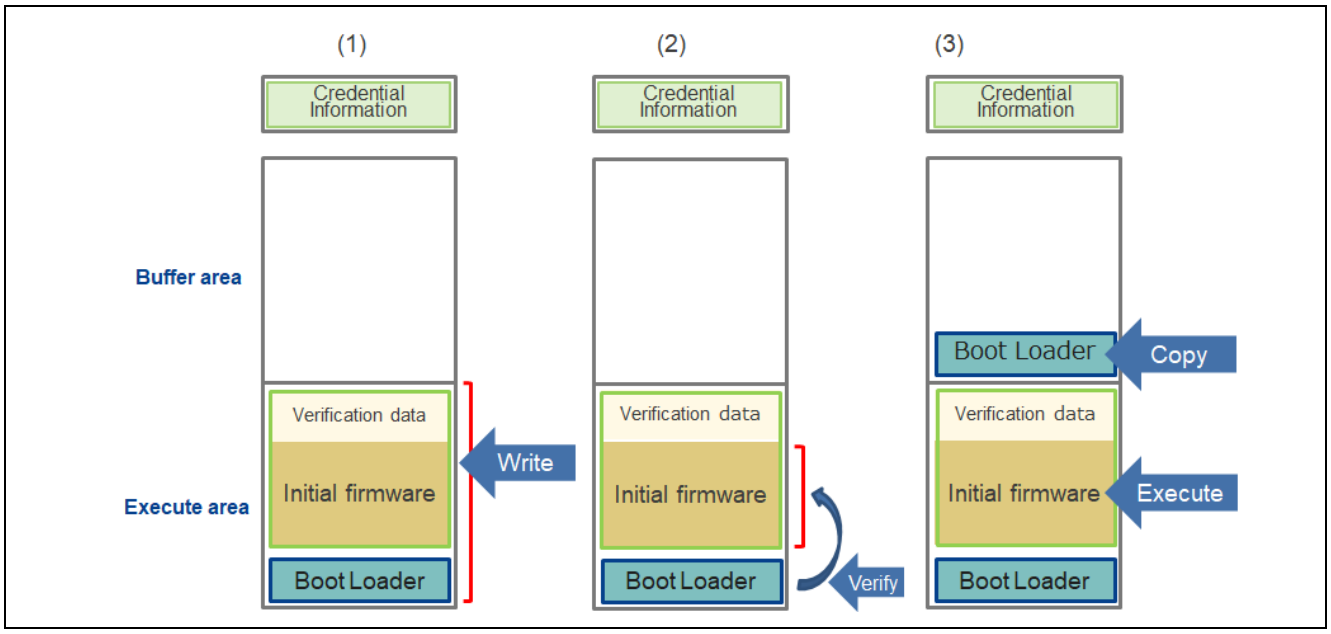

Figure 2.35 Operation of the Bootloader (for Firmware Update Module v1)

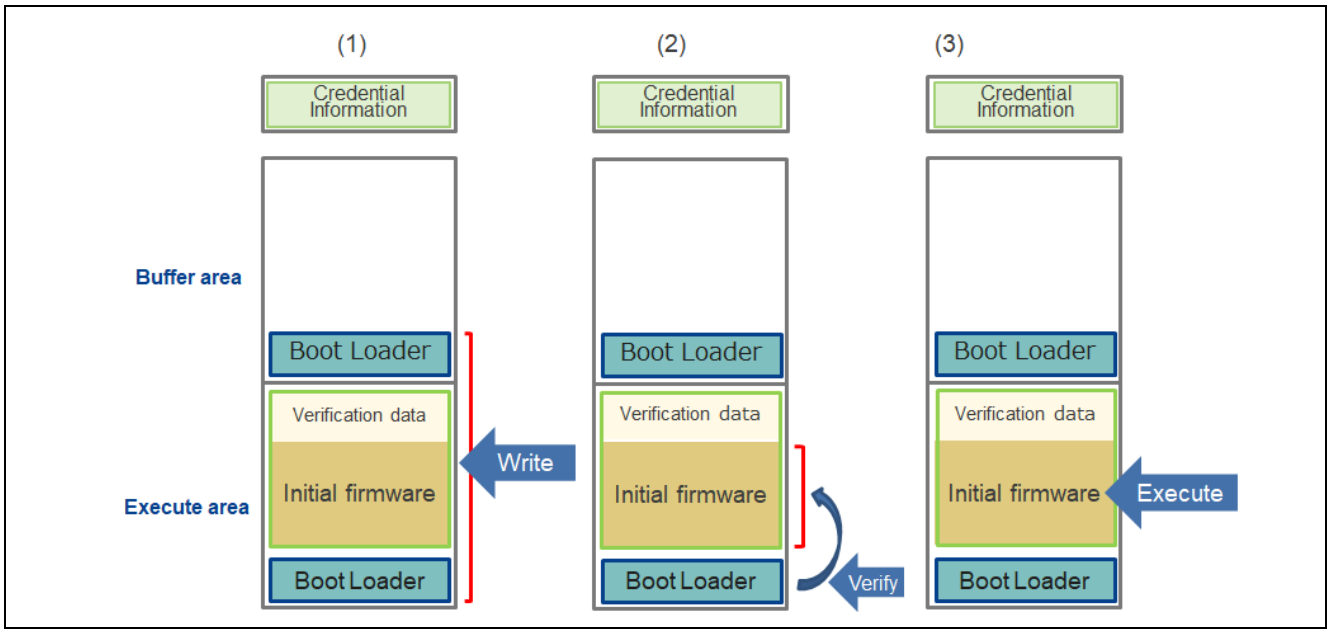

Figure 2.36 Operation of the Bootloader (for Firmware Update Module v2)

2.13 Modifying the Code of the Updated Firmware

Open **\src\sample_azure_iot_embedded_sdk_adu.c** from the **adu_sample** project and change the value of **SAMPLE_DEVICE_INSTALLED_CRITERIA** to **1.1.0**.*[1]

Note: 1. If the firmware version set in the Azure IoT Hub is already present, set the value to a different version number.

If necessary, add any needed update processing.

```
#ifndef SAMPLE_DEVICE_INSTALLED_CRITERIA
#define SAMPLE_DEVICE_INSTALLED_CRITERIA                    "1.1.0"
#endif /* SAMPLE_DEVICE_INSTALLED_CRITERIA */

#if (NX_AZURE_IOT_ADU_AGENT_PROXY_UPDATE_COUNT >= 1)
/* Device properties.  */
#ifndef SAMPLE_LEAF_DEVICE_MANUFACTURER
#define SAMPLE_LEAF_DEVICE_MANUFACTURER                     "Contoso"
#endif /* SAMPLE_LEAF_DEVICE_MANUFACTURER*/

#ifndef SAMPLE_LEAF_DEVICE_MODEL
#define SAMPLE_LEAF_DEVICE_MODEL                            "IoTDevice-Leaf"
#endif /* SAMPLE_LEAF_DEVICE_MODEL */

#ifndef SAMPLE_LEAF_DEVICE_INSTALLED_CRITERIA
#define SAMPLE_LEAF_DEVICE_INSTALLED_CRITERIA               "1.0.0"
#endif /* SAMPLE_LEAF_DEVICE_INSTALLED_CRITERIA */
#endif /* NX_AZURE_IOT_ADU_AGENT_PROXY_UPDATE_COUNT */
```

Figure 2.37 Updated Firmware Setting

2.14 Building the adu_sample Project (for the Update Firmware)

Build the **adu_sample** project and create an **adu_sample.mot** file.
The MOT file is generated in the following folder:
\adu_sample\HardwareDebug\

2.15 Creating the Updated Firmware

Convert the updated firmware to RSU format.*1

The procedure for creating firmware differs depending on whether the version of the firmware update module is v1 or v2.
Refer to the procedure applicable to the version you use.
Note that a specific procedure also applies in the case where you use firmware update module v2 with a bootloader created by using firmware update module v1. In this case, refer to the applicable procedure.

Note: 1. The MOT file format (the data format generally used for firmware) provides no mechanism for storing data other than the actual data to be programmed to the device (for example, verification data such as hash values). In addition, MOT files consist of text data, so they tend to be around twice as large as files consisting of binary data.
To avoid these limitations, Renesas Secure Update (RSU), a binary file format exclusive to Renesas that allows storage of verification data alongside the actual firmware data, is used for ADU releases.

2.15.1 If Using Firmware Update Module v1

Create an RSU file of the update firmware by adding verification data to the firmware **adu_sample.mot**.
Use Renesas Secure Flash Programmer to create the update firmware.
Double-click the file **mot-file-converter-2.0.2\Renesas Secure Flash Programmer\bin\Debug\Renesas Secure Flash Programmer.exe** to launch the program. Click the **Update Firm** tab, and then specify the settings as follows. The values to be specified are shown in blue font.

- Select MCU: RX65N Flash(Code=2MB, Data=32KB)/Secure Bootloader=64KB
- Select Firmware Verification Type: sig-sha256-ecdsa
- Private Key Path (PEM format): secp256r1.privatekey generated by OpenSSL in step E
- Firmware Sequence Number: 1
- File Path (Motorola format): \adu_sample\HardwareDebug\adu_sample.mot

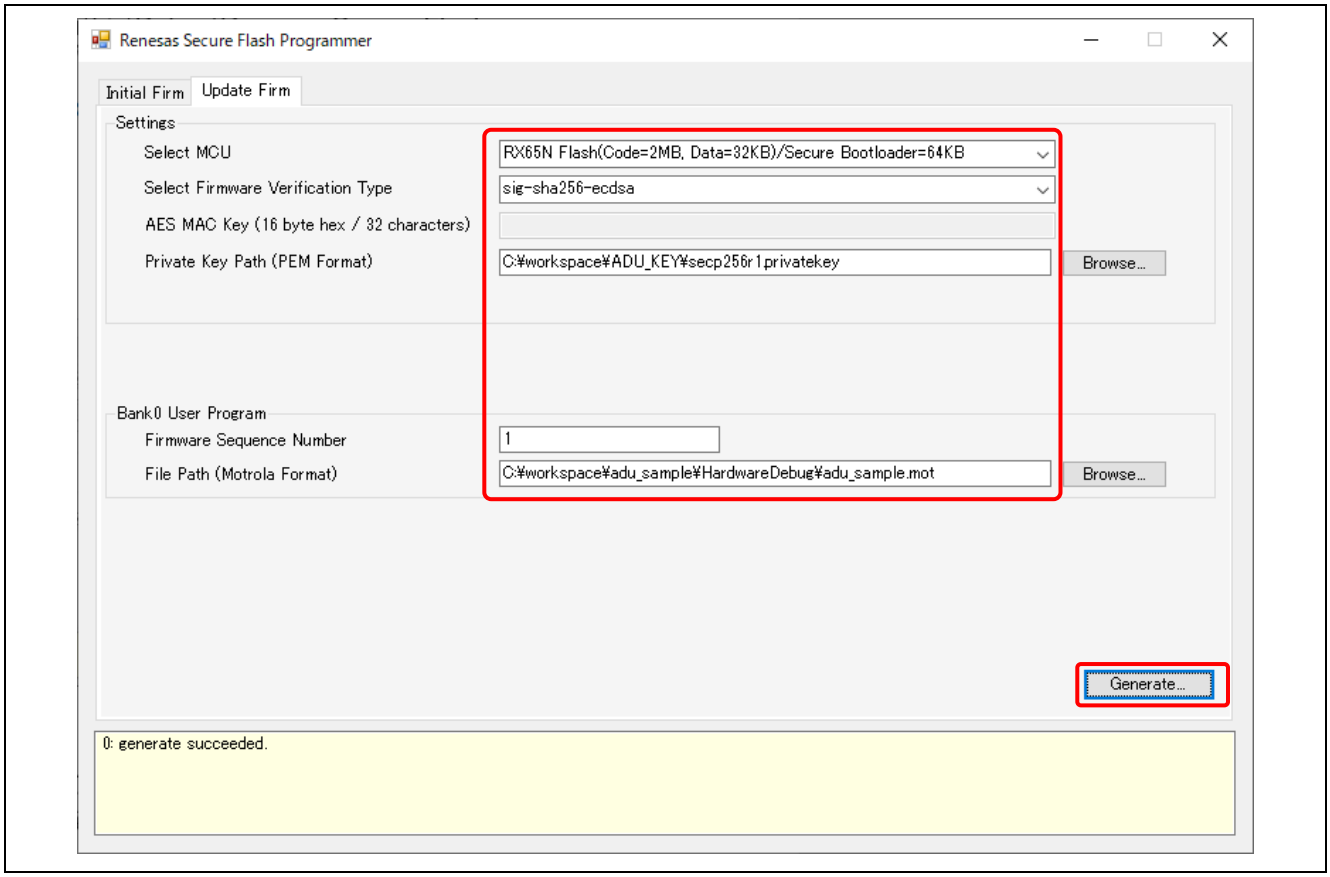

Figure 2.38 Updated Firmware Creation Window

Finally, click the **Generate...** button, and then, in the following window that appears, set any output destination folder of your choice, and enter the name of the output file. In this example, **firmware_1.1.0.rsu** is entered as the output file name.*1
Click the **Save** button. The RSU file of the update firmware is output to the specified folder.
The process is complete when the message “generate succeeded.” appears at the bottom of the window.

Note: 1. The “1.1.0” portion in the file name differs depending on the version of the update firmware that you set.

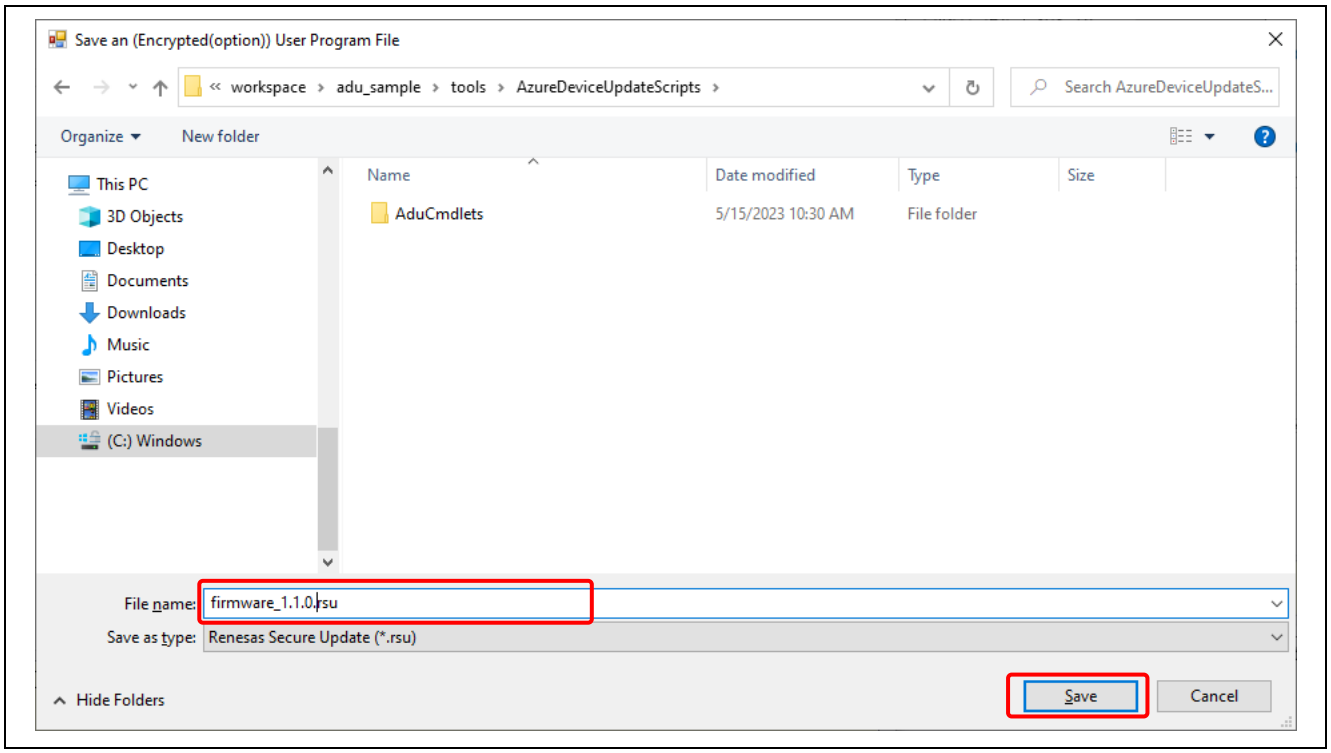

Figure 2.39 Outputting an RSU File of the Update Firmware

When the message “generate succeeded” appears, processing is complete.

The following figure shows the configuration of the RSU file of the update firmware generated by Renesas Secure Flash Programmer.

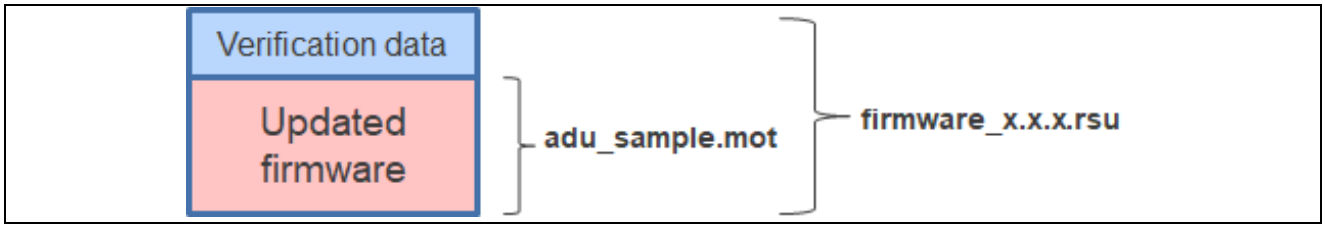

Figure 2.40 Configuration of the RSU File of the Update Firmware (for Firmware Update Module v1)

The update firmware image “firmware_1.1.0.rsu” is a binary file in which verification data and the update firmware (MOT file) have been merged.

Refer to section 7.1, Download Data Format, in the application note Renesas MCU Firmware Update Design Policy for details on RSU files.

Also, this item provides details of the verification data (0x00000000 to 0x000002FF).

Once the updated firmware has been created, refer to section 3, Operations on Microsoft Azure Portal, and follow the instructions to register the updated firmware on the IoT Hub and update the firmware.

2.15.2 If Using Firmware Update Module v2

Create an RSU file of the update firmware by adding header data to the firmware “adu_sample.mot”.

Use Renesas Image Generator to create the update firmware.
Create an image file of the update firmware by using the following procedure:

1. Open the “RenesasImageGenerator” folder that is bundled with the sample project.
   This folder contains the following three files:

```
RenesasImageGenerator
|-- image-gen.py
|-- RX65N_DualBank_Initial_PRM.csv
|-- RX65N_DualBank_Update_PRM.csv
```

2. Copy the following two files to the “RenesasImageGenerator” folder:

Private key	secp256r1.privatekey	Created in section 2.4.2, Generating a Key Pair for ECC in OpenSSL
Update firmware	adu_sample.mot	Created in section 2.14, Building the adu_sample Project (for the Update Firmware)

   Make sure that you copy the files to the “RenesasImageGenerator” folder in the following structure. The files in red font are the files you copy.

```
RenesasImageGenerator
|-- image-gen.py
|-- RX65N_DualBank_Initial_PRM.csv
|-- RX65N_DualBank_Update_PRM.csv
|-- secp256r1.privatekey
|-- adu_sample.mot
```

3. Open the Command Prompt window, and then change the current directory to the “RenesasImageGenerator” folder.
4. Execute the following command from the command prompt. An image file of the update firmware will be created.

```
> python image-gen.py -iup adu_sample.mot -ip RX65N_DualBank_Update_PRM.csv -o
firmware_1.1.0 -vt ecdsa
```

The string in red font in the above command line is the file name of the update firmware image.
In the file name, the “1.1.0” portion is the version number of example update firmware. Change the value of this portion according to the version of the update firmware that you set.

Creation of firmware takes some time.
When the message “Successfully generated the firmware_1.1.0.rsu file” appears, creation is complete.
The RSU file of the update firmware image “firmware_1.1.0.rsu” is output to the “RenesasImageGenerator” folder.

The following figure shows the configuration of the RSU file of the update firmware generated by Renesas Image Generator.

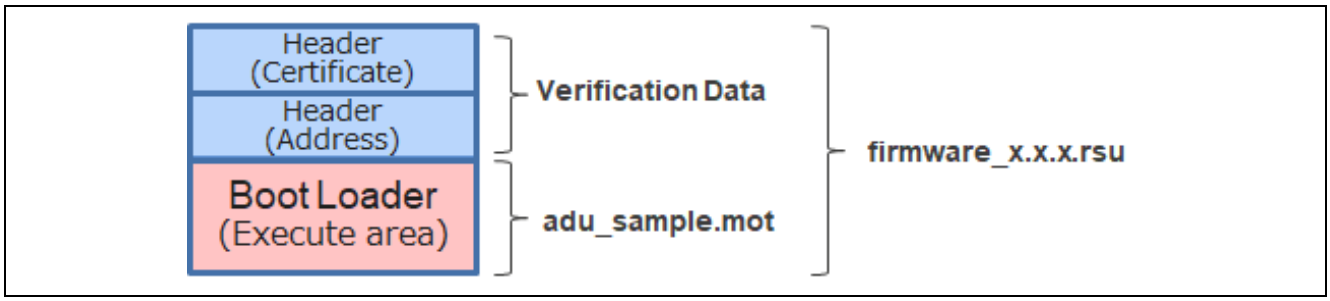

Figure 2.41 Configuration of the RSU File of the Update Firmware (for Firmware Update Module v2)

The update firmware image “firmware_1.1.0.rsu” is a binary file in which verification data and the update firmware (MOT file) have been merged.
The verification data is located in two areas. The “Header (Certificate)” area stores the information for verifying the firmware, and the “Header (Address)” area stores the address information of the firmware.

For details on RSU files, refer to section 4.2.1, Update Image File, in the application note “Firmware Update Module Using Firmware Integration Technology (Rev2.01)”.
Note that this section also describes the RSU header format (Verification data: 0x00000000 to 0x000002FF) in detail.

Once the updated firmware has been created, refer to section 3, Operations on Microsoft Azure Portal, and follow the instructions to register the updated firmware on the IoT Hub and update the firmware.

2.15.3 If Using Firmware Update Module v2 in the Bootloader v1 Environment

The RSU files created by firmware update modules v1 and v2 have different formats.
Therefore, the bootloader created by firmware update module v1 cannot handle raw update firmware created by firmware update module v2.

For this bootloader to handle such update firmware, create an update firmware image by using the procedure shown below.*1

Note: 1. The update firmware image created by using the procedure shown below cannot be used for a bootloader created by firmware update module v2. If you use a bootloader created by firmware update module v2, use the procedure described in section 2.15.2, If Using Firmware Update Module v2.

1. When you build update firmware, specify the following settings.
   Select **r_fwup** in the component tree of the Smart Configurator, set **Enabled** for the **FWUP v1 compatible Setting** property, and then build “adu_sample”.
   After you have changed the settings in the Smart Configurator, click the **Generate Code** button to generate code.

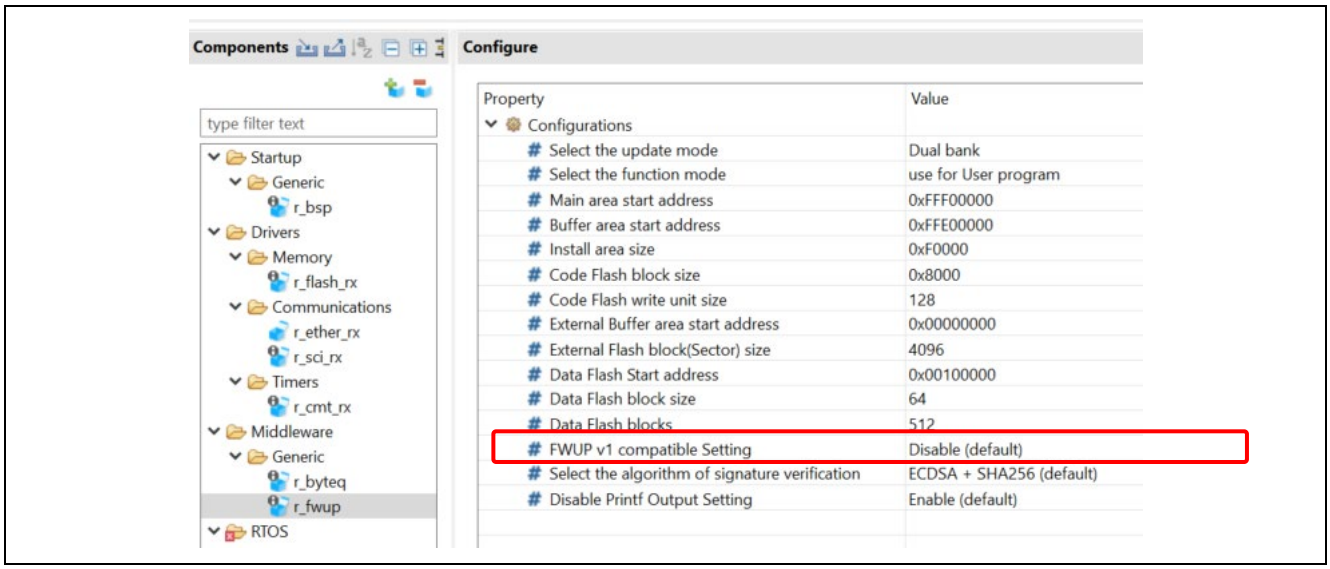

Figure 2.42 Settings of Firmware Update Module v1 to Ensure Compatibility with v2

2. Open the “RenesasImageGenerator” folder that is bundled with the sample project.
   This folder contains the following three files:

```
RenesasImageGenerator
|-- image-gen.py
|-- RX65N_DualBank_Initial_PRM.csv
|-- RX65N_DualBank_Update_PRM.csv
```

3. Copy the following two files to the “RenesasImageGenerator” folder:

Private key	secp256r1.privatekey	Created in section 2.4.2, Generating a Key Pair for ECC in OpenSSL
Update firmware	adu_sample.mot	Created in section 2.14, Building the adu_sample Project (for the Update Firmware) Built with **FWUP v1 compatible Setting** enabled

   RenesasImageGenerator is deployed in the folder as follows. The files in red font are the files you copy.

```
RenesasImageGenerator
|-- image-gen.py
|-- RX65N_DualBank_Initial_PRM.csv
|-- RX65N_DualBank_Update_PRM.csv
|-- secp256r1.privatekey
|-- adu_sample.mot
```

4. Open the Command Prompt window, and then change the current directory to the “RenesasImageGenerator” folder.

5. Execute the following command from the command prompt. An image file of the update firmware will be created.

```
> python image-gen.py -iup adu_sample.mot -ip RX65N_DualBank_Update_PRM.csv -o
firmware_1.1.0 -vt ecdsa -ff BAREMETAL_FWUP_V2_V1_DATA
```

The string in red font in the above command line is the file name of the update firmware.
In the file name, the “1.1.0” portion is the version number of example update firmware. Change the value of this portion according to the version of the update firmware that you set.

Creation takes some time.
When the message “Successfully generated the firmware_1.1.0.rsu file” appears, creation is complete.
The RSU file of the update firmware image “firmware_1.1.0.rsu” is output to the “RenesasImageGenerator” folder.

Once the updated firmware has been created, refer to section 3, Operations on Microsoft Azure Portal, and follow the instructions to register the updated firmware on the IoT Hub and update the firmware.

3. Operations on Microsoft Azure Portal

The Microsoft Azure operation procedure for implementing ADU is described below.

3.1 IoT Hub and Device Registration

Create an IoT Hub and device on the Azure portal.*1 This process is described in section 3.1 of the application note Visualization of Sensor Data using RX65N Cloud Kit and Azure RTOS.

Note: 1. The screenshots of windows at the link may be modified.
In order to implement ADU it is necessary to select the Standard tier and edition type S1 in the pricing options for the IoT Hub. Note that it is not possible to implement ADU using the Free edition type.
You can set S1 as the edition level by setting **Tier** to **Standard** and **Daily message limit** to **400,000** in the IoT Hub creation window shown in the following figure.

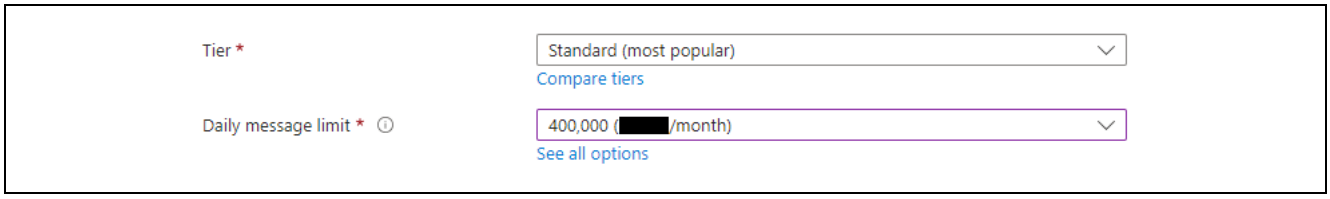

Figure 3.1 Pricing Option Settings

The IoT hub created here has the following three parameters. Specify these parameters in the source code of the ADU sample project by referring to section 2.6, Connection Information Macro Settings.

- Host name: HOST_NAME
- Device ID: DEVICE_ID
- Primary key: DEVICE_SYMMETRIC_KEY

3.2 Creating a Device Update Account and Instance

To execute ADU, a device update resource for IoT Hub is required. In this section, you create this resource. Be aware that a paid account is required in order to use Device Update.

1. In the home page of the Azure portal site, click **Create a resource**. The **Create a resource** page opens.

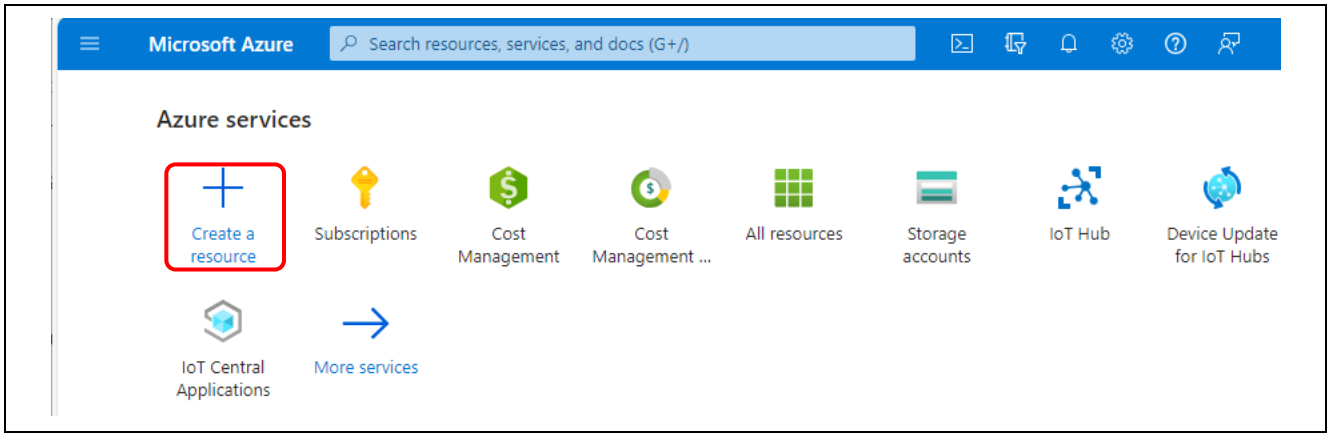

Figure 3.2 Azure Home Page

2. Create a Device Update account for IoT Hub. In the search box on the **Create a resource** page, enter **device update**, and then perform a search. **Device Update for IoT Hub** is displayed as the search result on the **Marketplace** page.

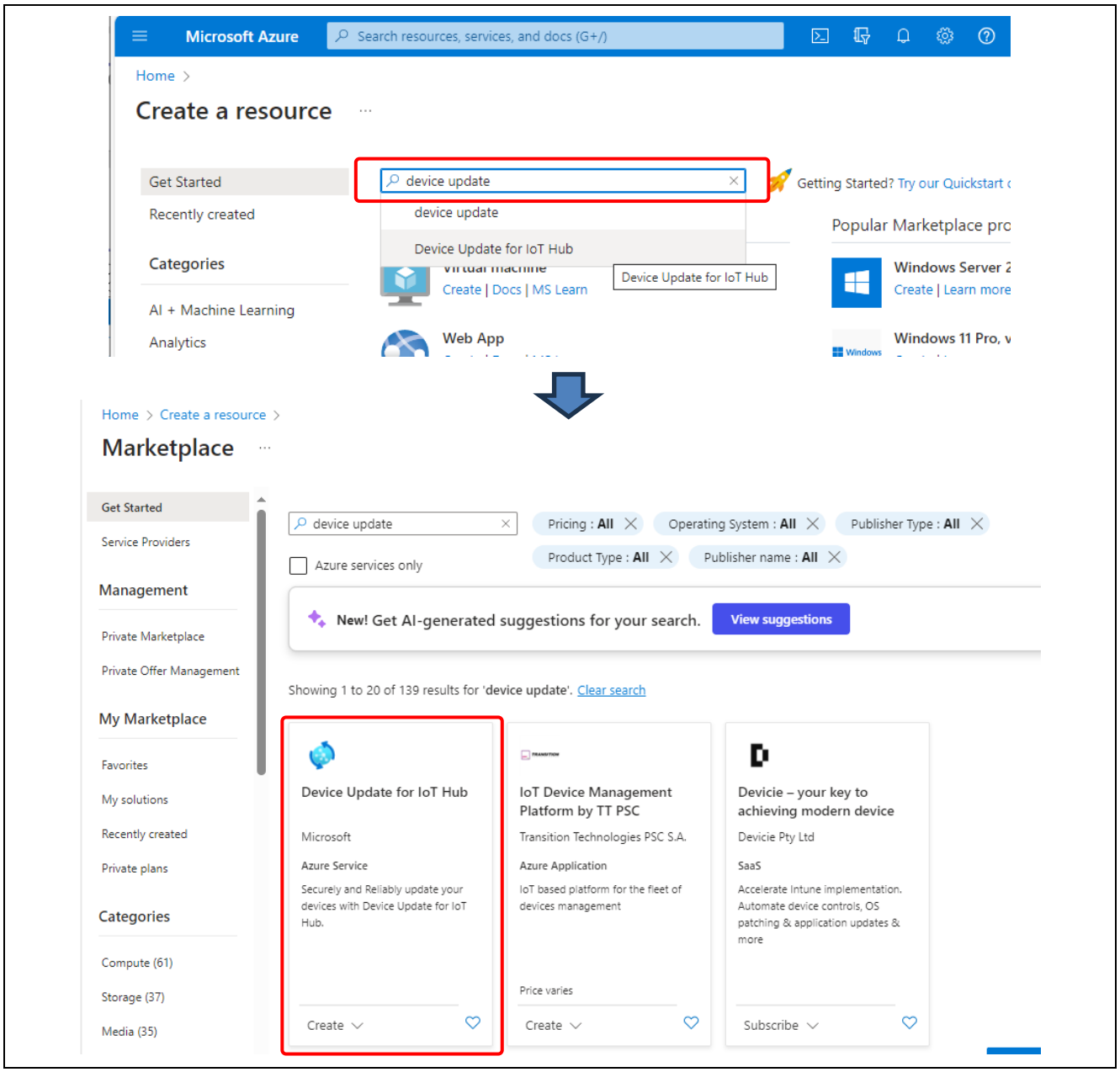

Figure 3.3 [Create a resource] page

3. From the **Create** drop-down list at the bottom of the **Device Update for IoT Hub** area, select **Device Update for IoT Hub**. The **Create Device Update** page appears.

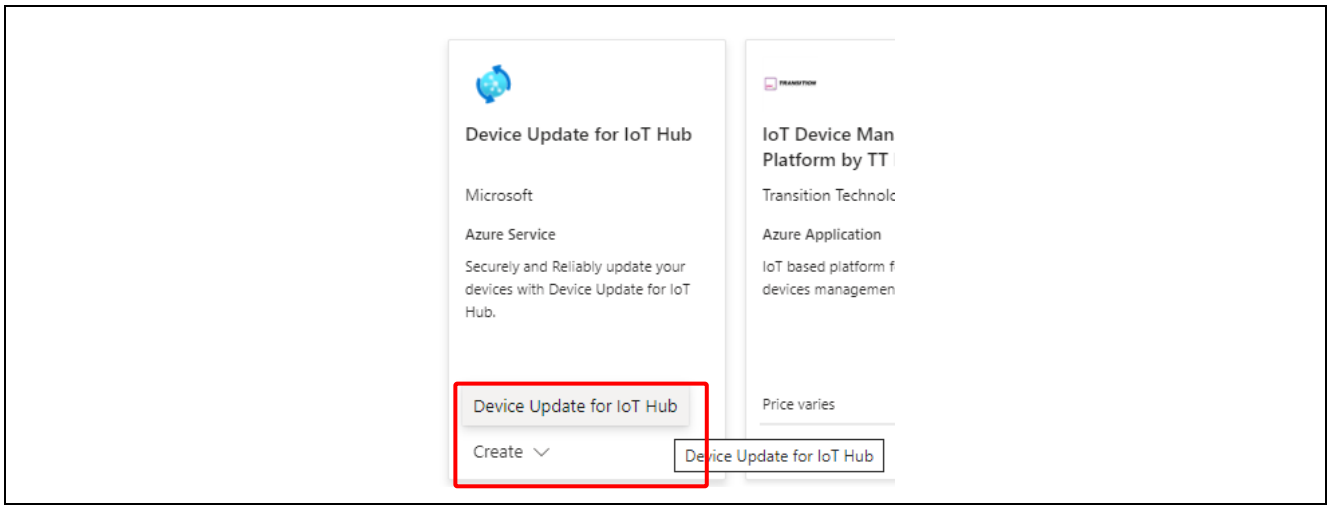

Figure 3.4 [Device Update] Page for IoT Hub

4. On the **Create Device Update** page, specify the following settings. After completing the settings, click the **Next: Diagnostics** button at the bottom of the page.

(1)	**Subscription**	Specify the subscription ID that you used when you created the IoT hub.
(2)	**Resource group**	Specify the resource group that you used when you created the IoT hub.
(3)	**Name** (account details)	Specify any character string of your choice as the name of the Device Update account.
(4)	**Location** (account details)	Specify the Azure region in which the Device Update account will be deployed.
(5)	**SKU**	Do not change the initial setting (**Standard**).
(6)	**Grant Access to Account**	This check box is used to determine whether to assign administrator privileges. Leave the check box selected.
(7)	**Instance Name**	Do not change the displayed value because the account name is set as is.
(8)	**IoT Hub Name**	Specify the IoT hub that you created in section 3.1.

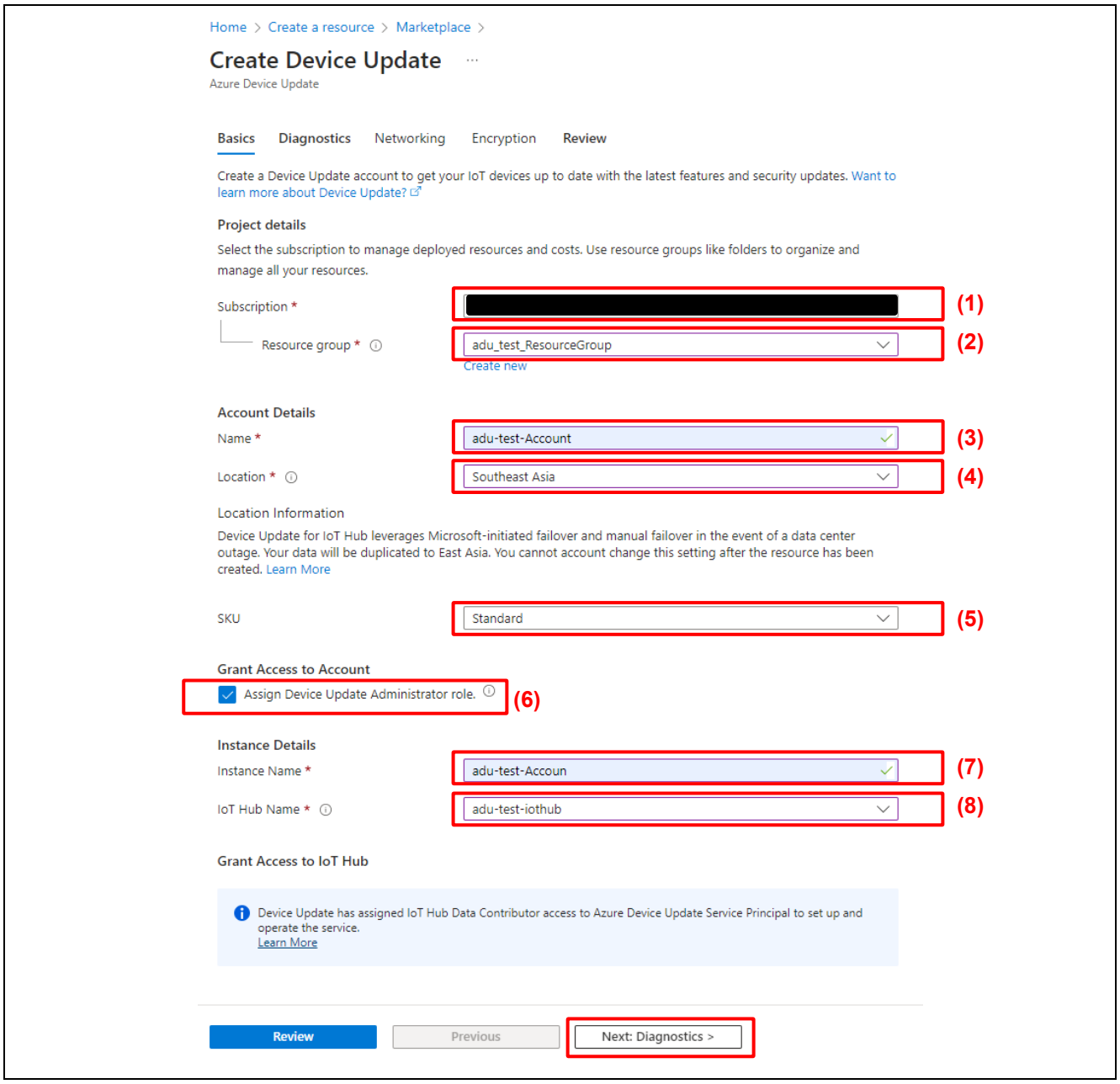

Figure 3.5 [Create Device Update] Page

5. In the **Diagnostics**, **Networking**, and **Encryption** tabs, leave the initial settings unchanged. Click the **Next** button at the bottom of the page. In the **Encryption** tab, click the **Review + create** button to display the following confirmation page.
   In the confirmation page, the registration information is validated. When validation is complete, a message to that effect appears, and the **Create** button at the bottom of the page is enabled.
   When you have confirmed that the specified settings are correct, click the **Create** button. Creation of a device update account starts.

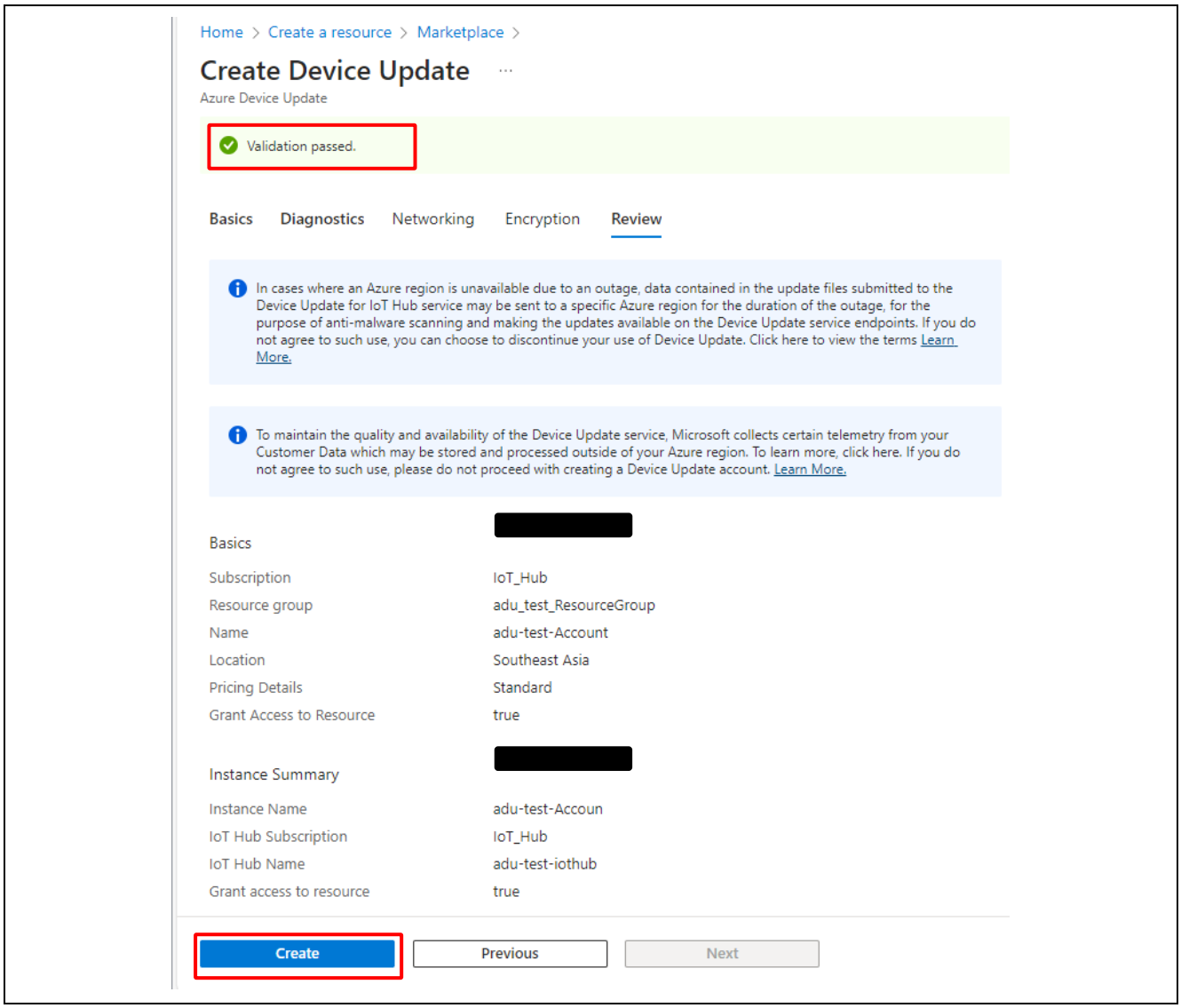

Figure 3.6 Confirmation Page for Creating a Device Update Account

6. Registration (deployment) of a Device Update account may take a long time (approximately 10 to 20 minutes). When registration is complete, click the **Go to resource** button. The device update window for the IoT hub that you created appears.

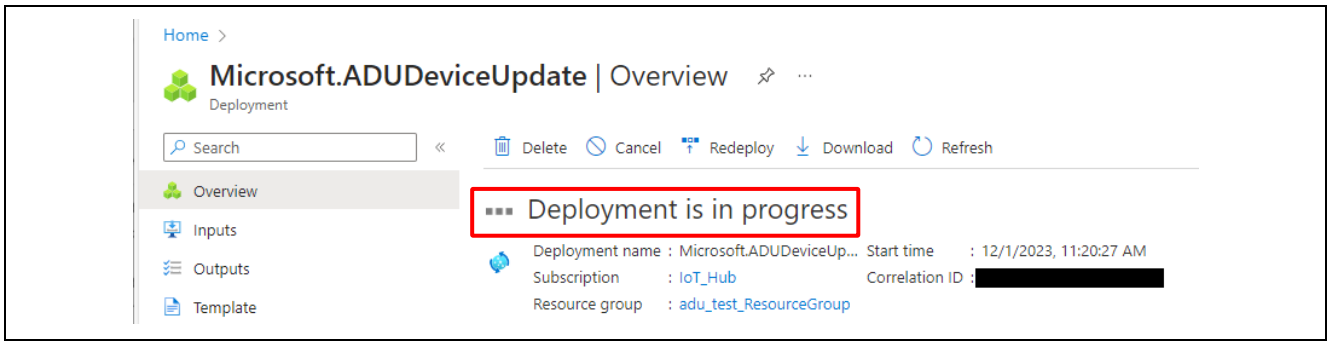

Figure 3.7 Registering a Device Update Account

7. In the Device Update page for the IoT hub that you created, click **View Device Update Instances** or **Instances** under **Instance Management** in the menu on the left of the page. The instance registration status is displayed.
   The IoT hub name linked with the instance name that was set during creation of the Device Update account is displayed. If **Succeeded** appears in the **Provisioning State** column, creation of the Device Update account is complete.

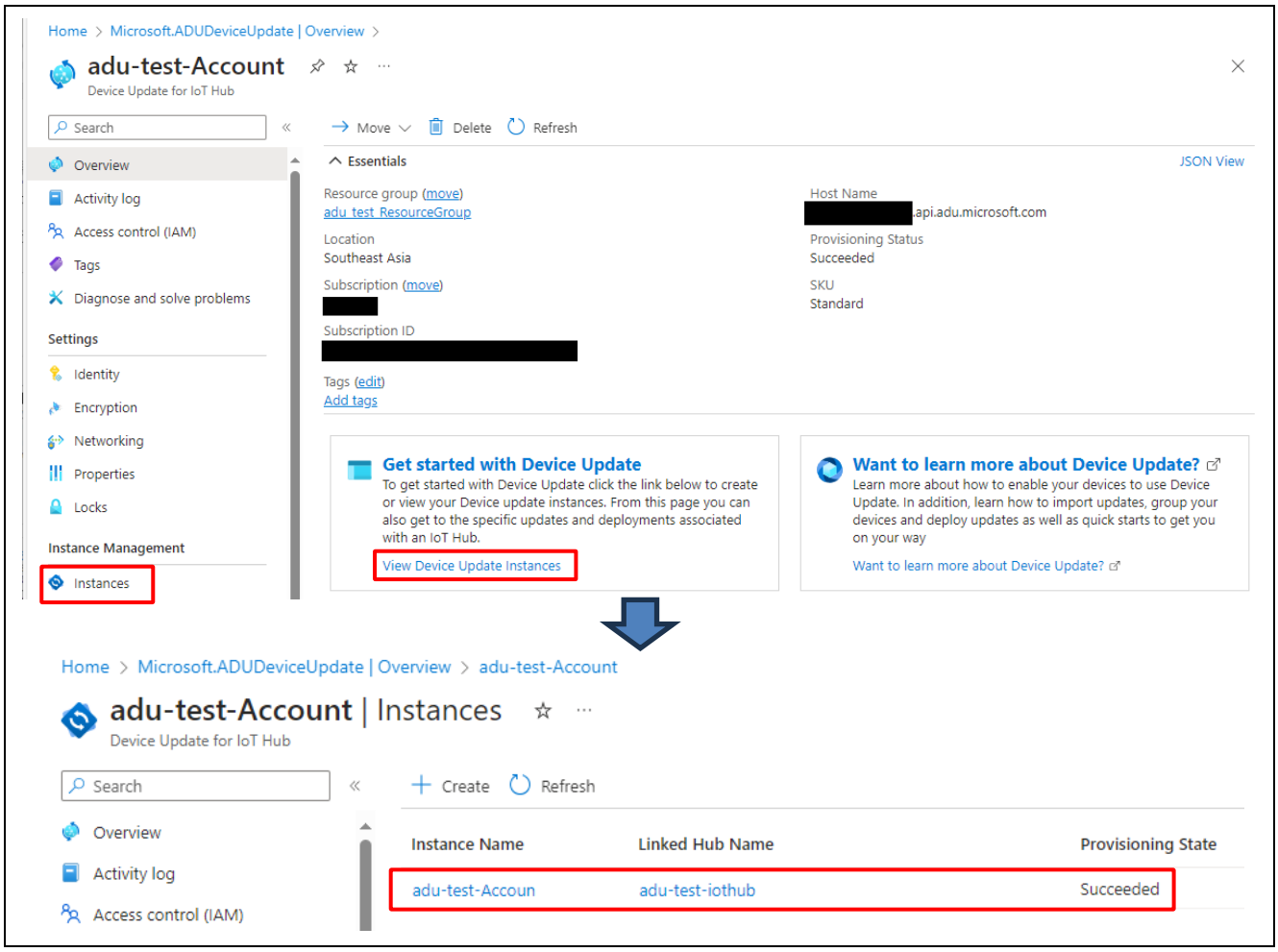

Figure 3.8 [Device Update for IoT Hub] Page

3.3 Creating a Storage Account Container

Create storage for storing update firmware.
Before you can use storage, you must prepare a storage account and container that stores files.

1. As in section 3.2, Creating a Device Update Account and Instance, in the home page of the Azure portal site, click **Create a resource**, and then, use the search box to perform a search for the string **storage**. When **Storage account** appears as the search result, from the **Create** drop-down list, select **Storage account**. The storage account creation page appears.

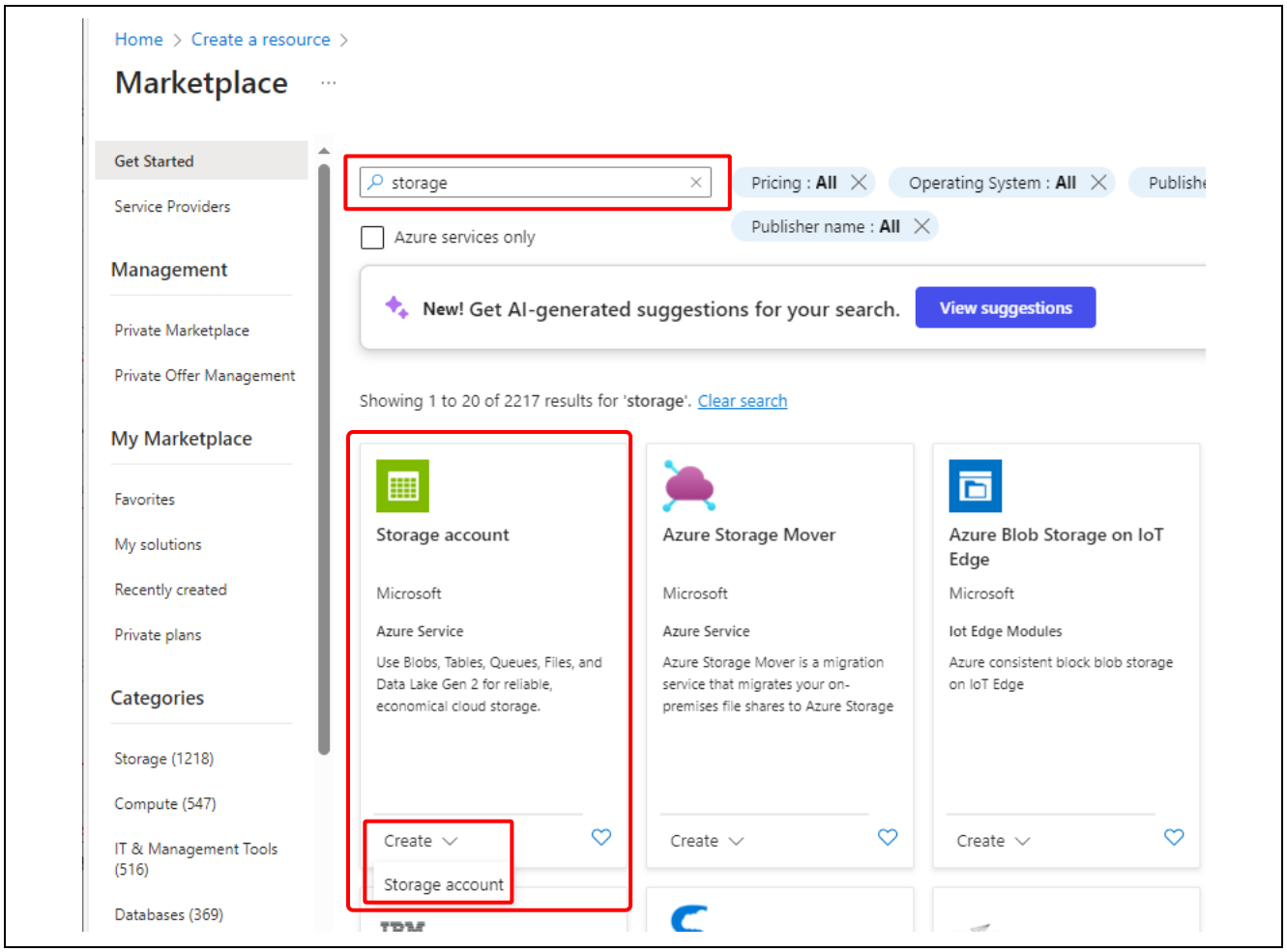

Figure 3.9 Storage Account Creation

2. On the **Create a storage account** page, specify the following settings. After completing the settings, click the **Next: Advanced** button at the bottom of the page.

(1)	**Subscription**	Specify the subscription ID that you used when you created the IoT hub.
(2)	**Resource group**	Specify the resource group that you used when you created the IoT hub.
(3)	**Storage account name**	Specify any character string of your choice as the name of the storage account.
(4)	**Region**	Specify the Azure region in which the storage account will be deployed.

Note: For the settings other than the above, leave the initial values unchanged.

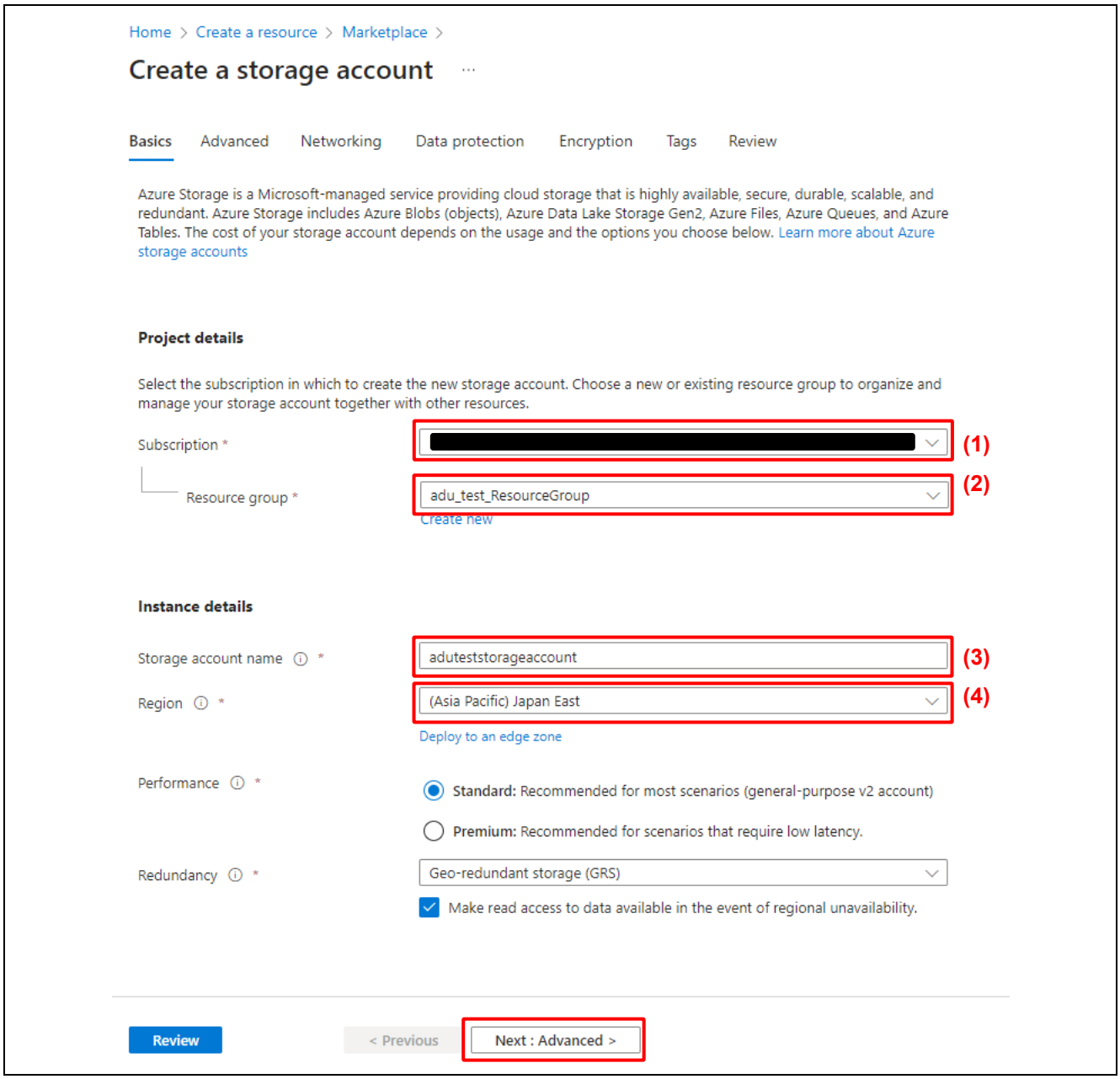

Figure 3.10 Creating a Storage Account

3. Leave the initial values unchanged in the **Advanced**, **Networking**, **Data protection**, **Encryption**, and **Tags** tabs. Click the **Next** button at the bottom of the page. In the **Tags** tab, click the **Next: Review** button to display the following confirmation page.
   In the confirmation page, the registration information is validated. When validation is complete, the **Create** button at the bottom of the page is enabled.
   When you have confirmed that the specified settings are correct, click the **Create** button. Creation of a storage account starts.

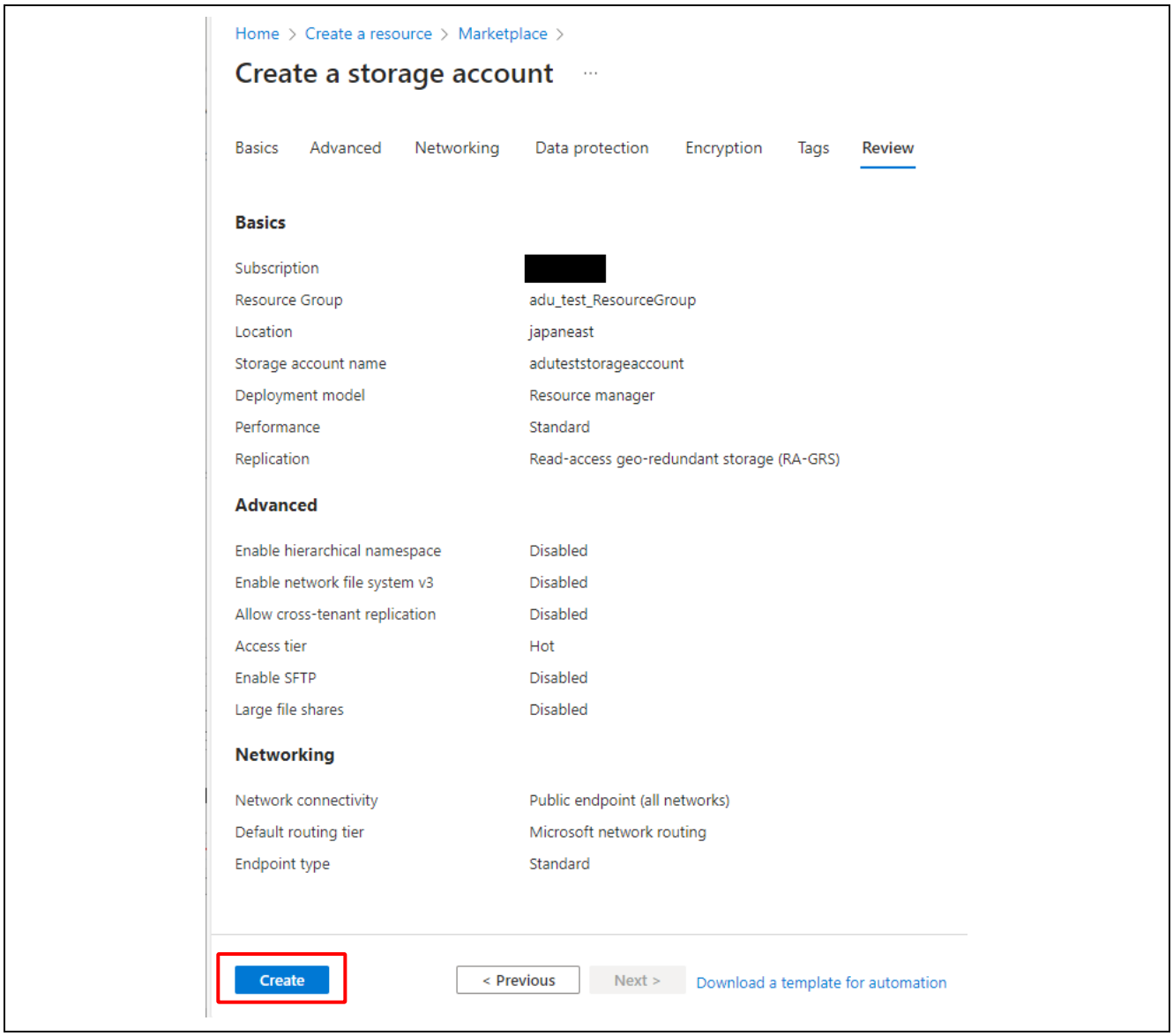

Figure 3.11 Settings Confirmation Page for Storage Account Creation

4. Registration (deployment) of a storage account takes some time. When registration is complete, click the **Go to resource** button. Registration is complete when the page of the storage account that you created appears.

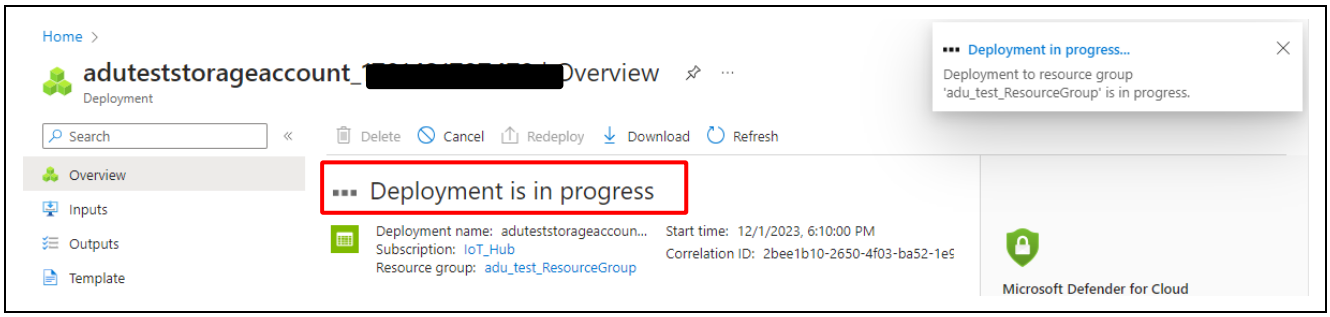

Figure 3.12 Storage Account Registration

5. Next, create a container in which to store update firmware files.
   On the storage account page, in the menu on the left side, click **Container** under **Data storage**. The **Containers** page appears.

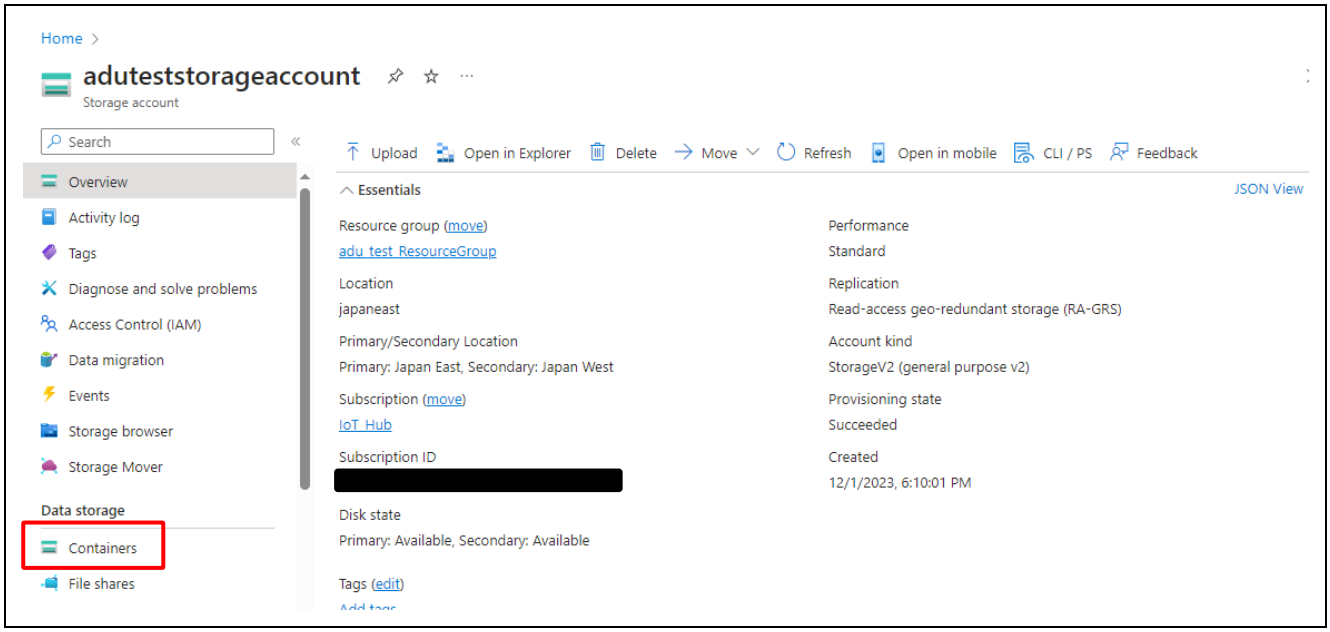

Figure 3.13 Completion of Storage Account Registration

6. On the **Containers** page, click **+Container**. When the page for creating a new container appears, enter any container name of your choice, and then click the **Create** button. For the other settings, leave the initial values unchanged. The processing is complete when the container name that you entered is displayed in the list.

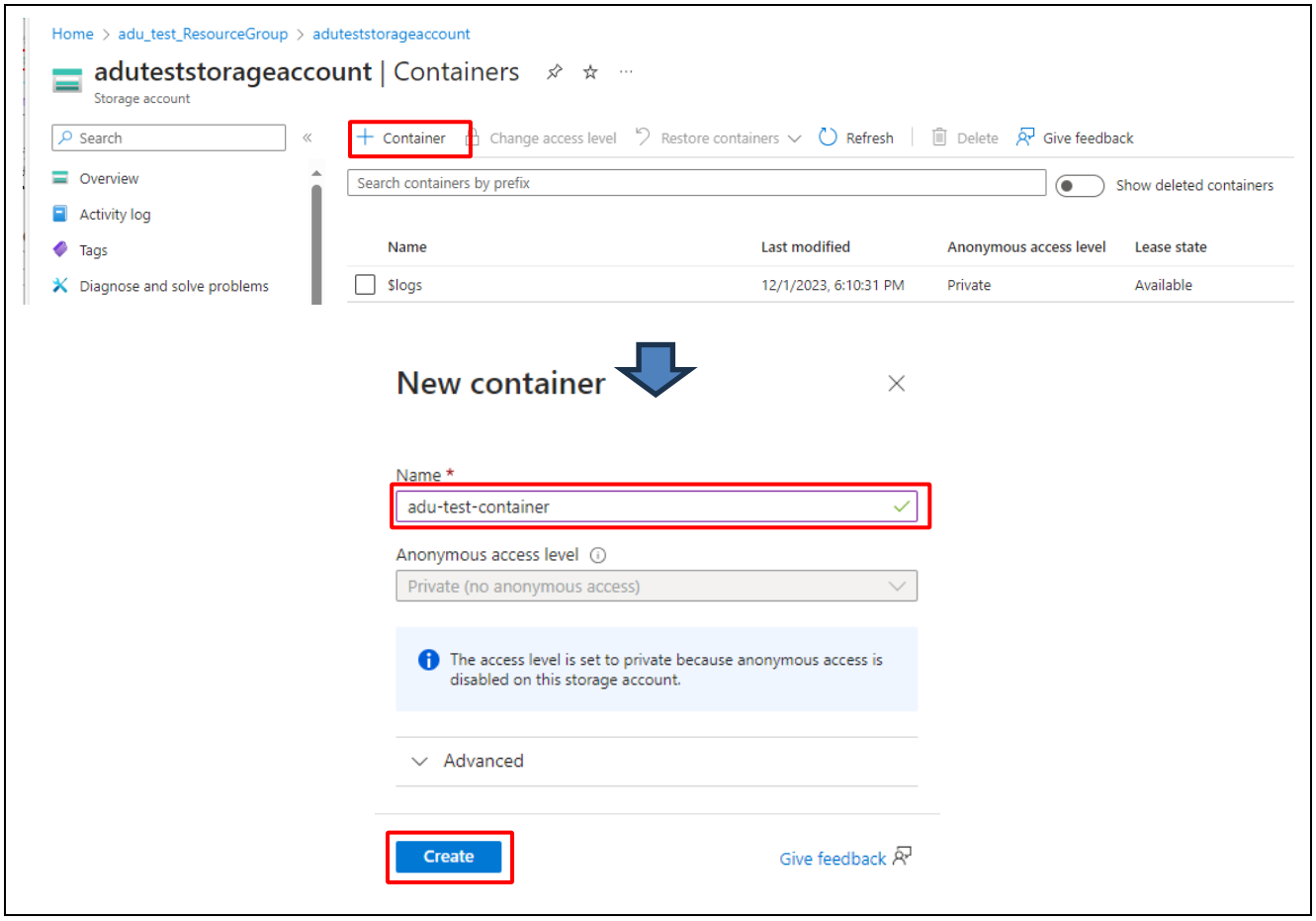

Figure 3.14 Creating a New Container

3.4 Preparing the Updated Firmware

3.4.1 Building the Updated Firmware

Create update firmware by referring to Chapter 2, Creating Sample Projects, and then create an RSU file.

3.4.2 Creating a Manifest File

A manifest file is a JSON file that defines information about the updated program required by Device Update for IoT Hub. The manifest file and binary file are used as a pair when uploading updated firmware to a IoT Hub. Follow the steps below to create a manifest file.

1. PowerShell v7.0 is used to create manifest files. Download and run the installer that matches your OS.

Note: Normally, download v7 (version released most recently). The following figure is an example when the version is v7.3.4.

Figure 3.15 PowerShell Installer Download

2. Launch PowerShell and change the current directory to the directory containing the scripts for creating ADU project manifest files.
   \adu_sample\tools\AzureDeviceUpdateScripts

3. Run the following command in PowerShell.
   Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process

4. Copy the RSU file created as described in 2.15, Creating the Updated Firmware, to the folder referenced in step 2. Assign the following name to the copied RSU file:
   firmware_1.1.0.rsu
   Replace **1.1.0** in the file name with the version number specified for the updated firmware. Also, do not change the file name again after the manifest file has been created.

5. Run the script shown below. Some items require input when the script is run, so enter the character strings shown in blue text below. The names of the scripts differ according to the names of the target boards they are intended to be used with. When reading the explanation, replace the relevant portion of the file name of the script as appropriate. The **LeafPath** item refers to the path setting of the child device connected to the target board, so press the Enter key for this item without entering anything.
   .\CreateCKRX65NUpdate.ps1

   cmdlet CreateCKRX65NUpdate.ps1 at command pipeline position 1
   Supply values for the following parameters:
   (Type !? for Help.)
   Version: 1.1.0
   HostPath: ./firmware_1.1.0.rsu
   LeafPath:
   Preparing update RENESAS.CK-RX65N.1.1.0 ...
   Preparing parent update RENESAS.CK-RX65N.1.1.0 ...
   Generating an import manifest RENESAS.CK-RX65N.1.1.0...
   Saving parent manifest file and payload(s) to .\RENESAS.CK-RX65N.1.1.0...

```
cmdlet CreateCKRX65NUpdate.ps1 at command pipeline position 1
Supply values for the following parameters:
(Type !? for Help.)
Version: 1.1.0
HostPath: ./firmware_1.1.0.rsu
LeafPath:
Preparing update RENESAS.CK-RX65N.1.1.0 ...
Preparing parent update RENESAS.CK-RX65N.1.1.0 ...
Generating an import manifest RENESAS.CK-RX65N.1.1.0...
    Saving parent manifest file and payload(s) to .¥RENESAS.CK-RX65N.1.1.0...
```

Figure 3.16 Script Run Window

6. When the script completes successfully, a folder named **RENESAS.CK-RX65N.1.1.0** is created in the script folder with the RSU file and manifest file listed below saved to it. Upload these two files to the storage container:
   — RENESAS.CK-RX65N.1.1.0.importmanifest.json
   — firmware_1.1.0.rsu

3.5 Uploading the Firmware Update to the Storage Container

Upload the previously generated updated firmware to the storage container. On the **Home** page of the Azure portal, perform the following steps.

On the **Home** page, click **Storage accounts** → <name of storage account to use> → **Containers** (under **Data storage**) → <name of container to use>. The **Containers** page appears. On the **Containers** page, click **Upload** to display the page for uploading updates (**Upload blob**).

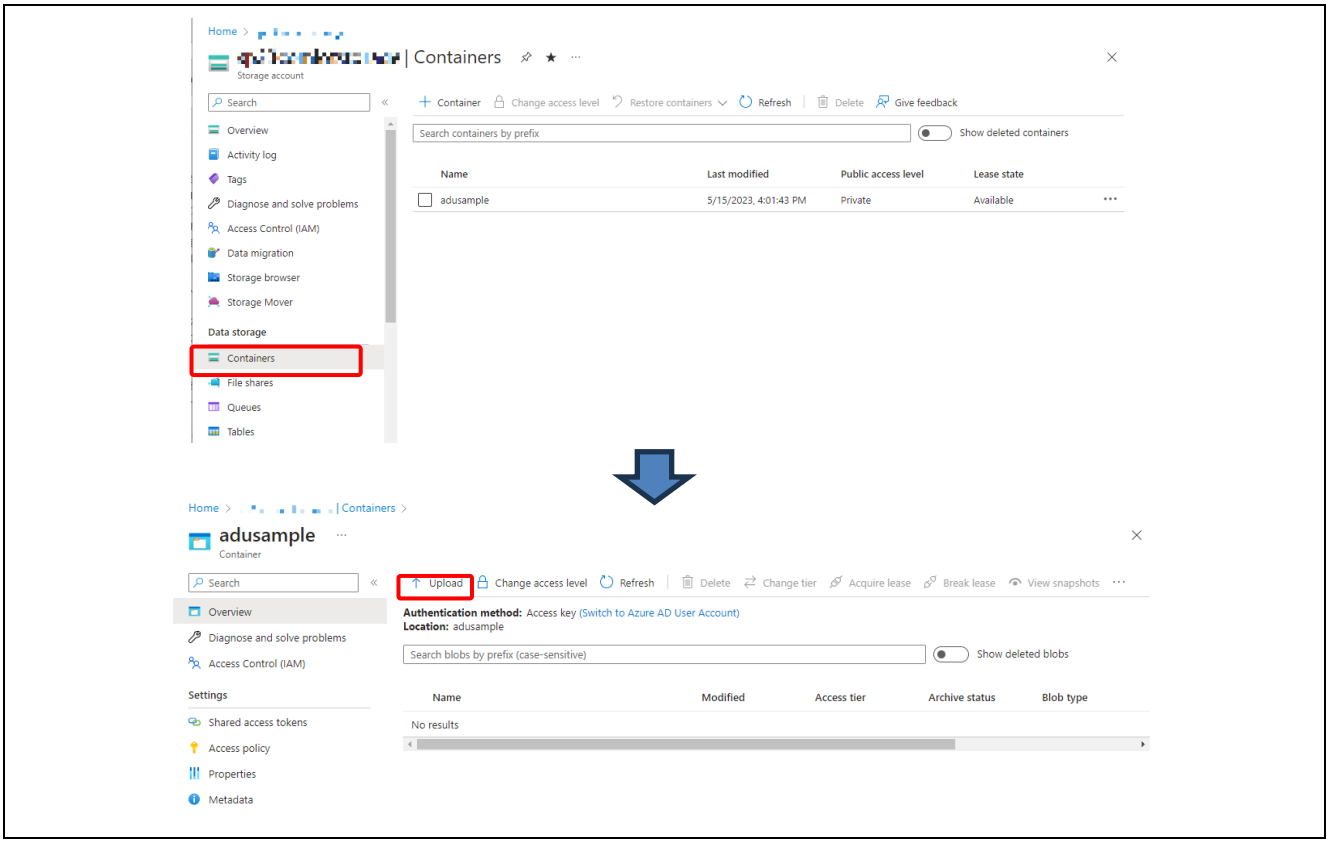

Figure 3.17 Containers Page

Drag and drop to the **Upload blob** page the **RENESAS.CK-RX65N.1.1.0** folder containing the firmware update binary file and manifest file prepared as described in section 3.4. When the files are registered, click the **Upload** button. When the upload completes, click the **×** button to close the **Upload blob** page. The added folder (or files) appear in the container contents list.

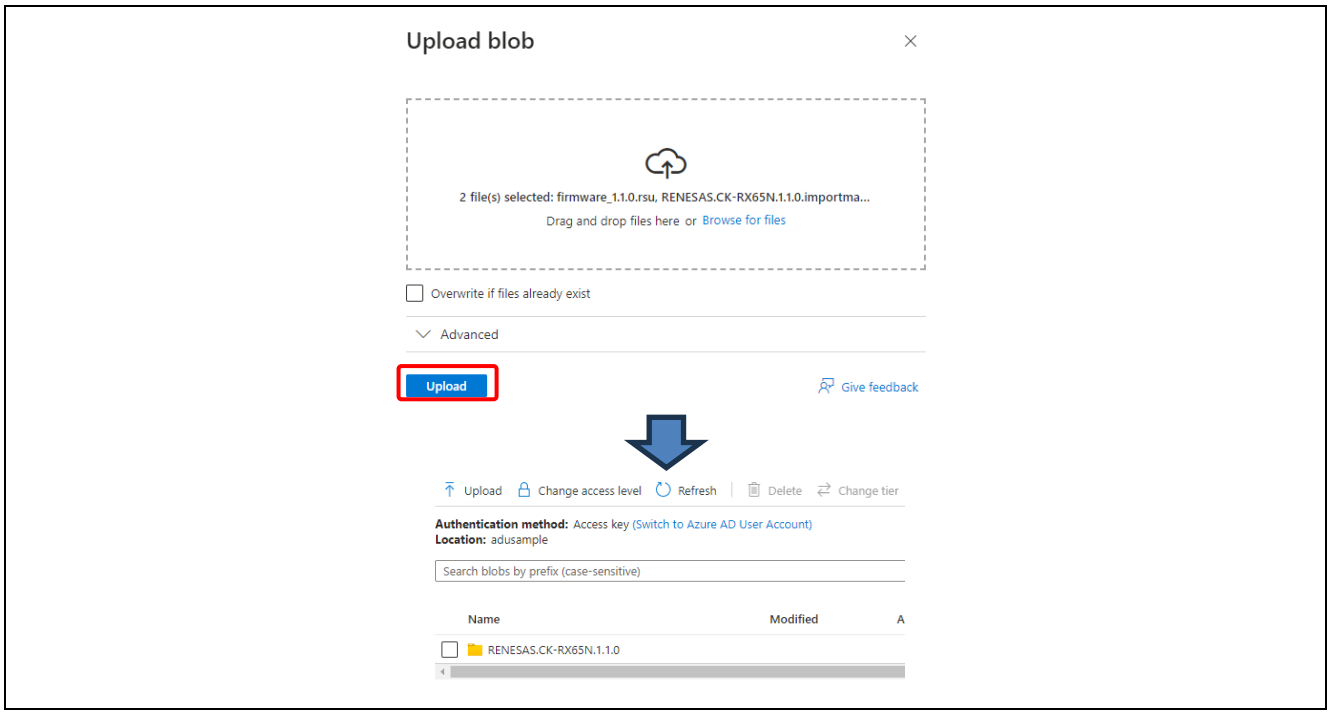

Figure 3.18 Uploading Update Files

3.6 Registering the Firmware Update

Register the firmware update uploaded to the storage container on the **IoT Hub** page. For the IoT Hub you are using, click **Updates** under **Device management** to display a list of updates. On this page, click **Updates** tab → **Import a new update**.

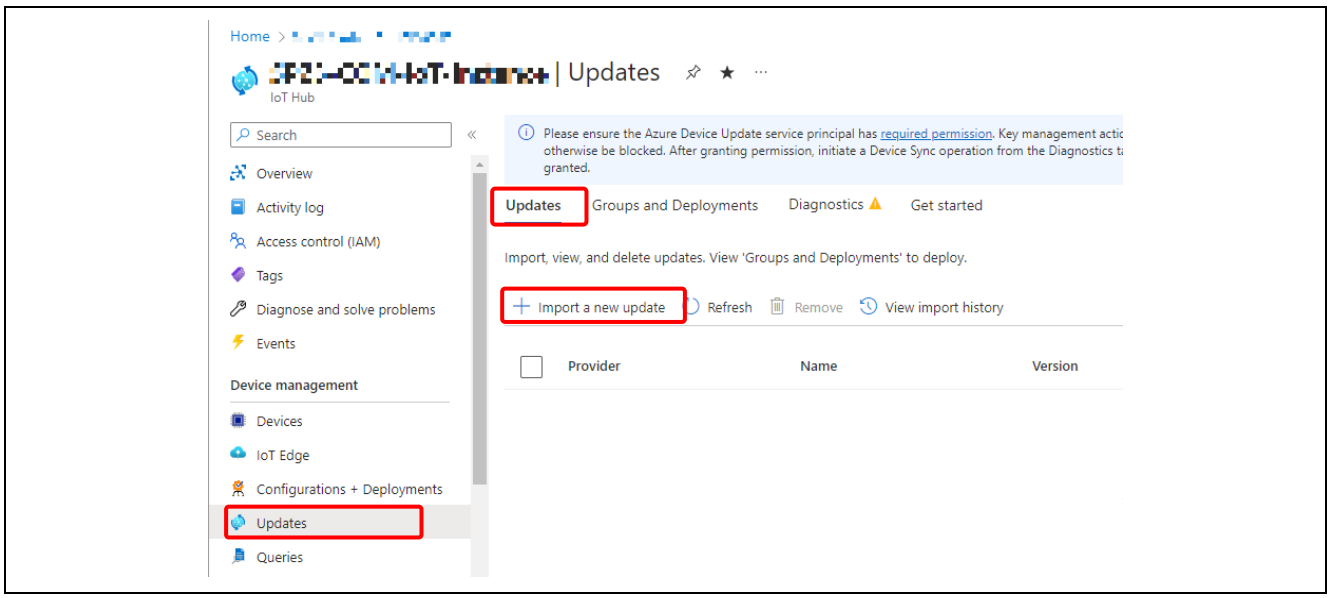

Figure 3.19 Updates List Page

The **Import update** page is displayed. Enter a description of the firmware update in the **Descriptive label** text field and click **Select from storage container**.

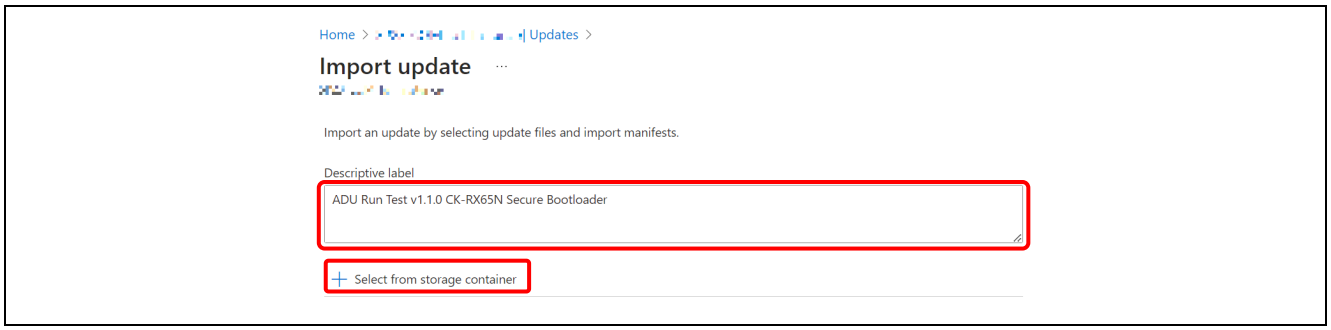

Figure 3.20 Entering a Description of the Update

Select **Storage account name** > **Containers name** in which the update firmware is to be stored.
Check the boxes next to the names of the firmware binary file and manifest file previously registered on the **Containers** page, and click the **Select** button.

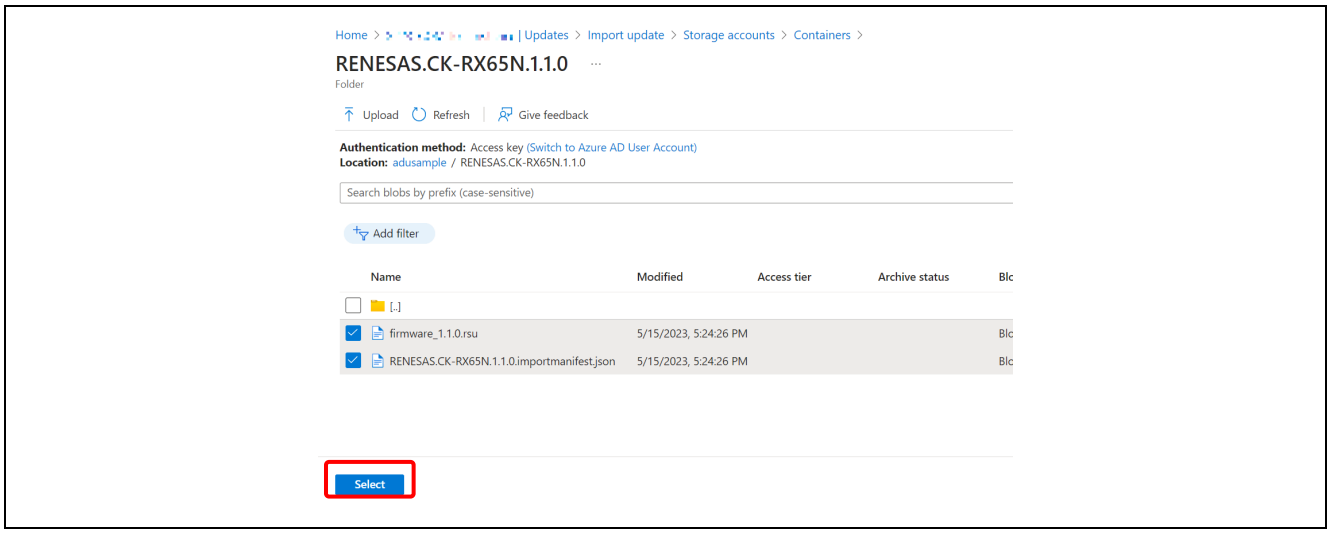

Figure 3.21 Selecting Update Files

The files you checked are already registered on the **Import update** page, so click the **Import update** button.

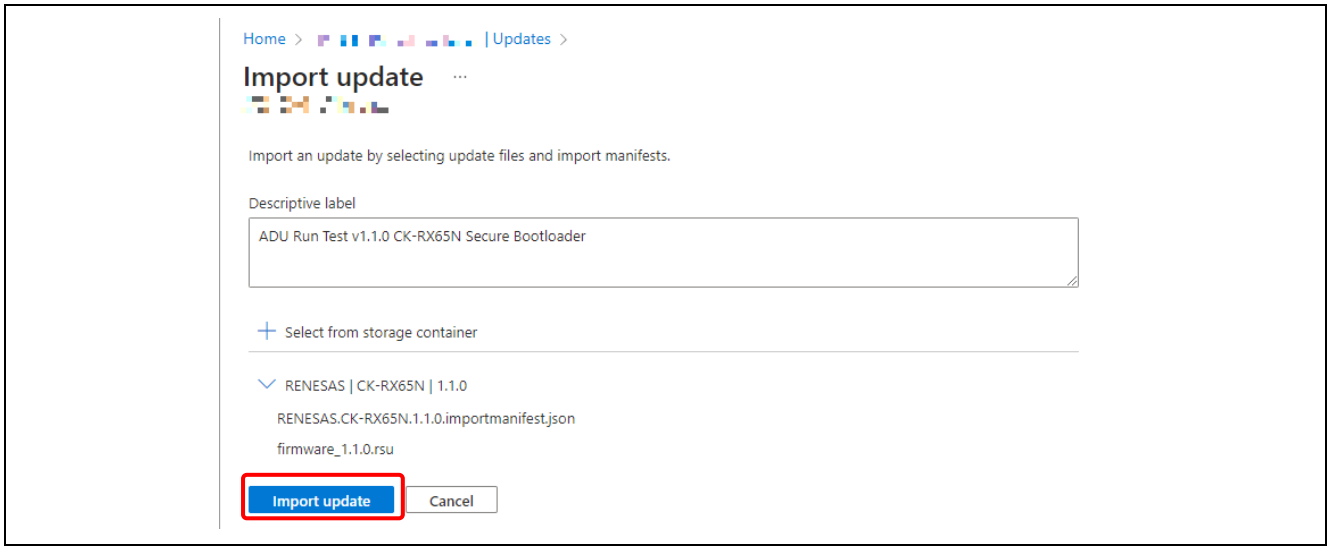

Figure 3.22 Import update Page

When import starts, a pop-up window shown in the following figure may appear at the top right of the page.

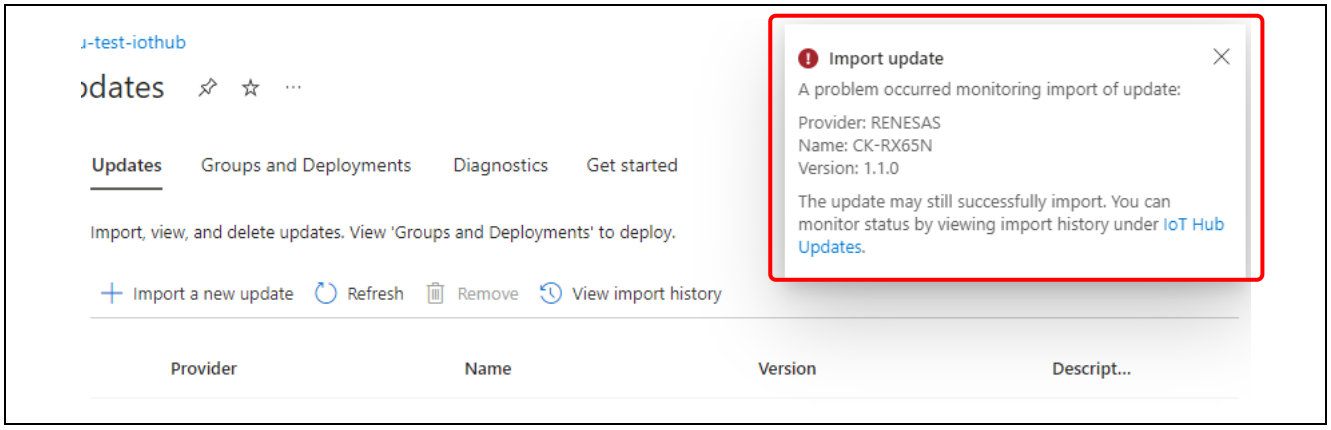

Figure 3.23 Warning Pop-Up Window for Import

Even when this pop-up window appears, the import processing is actually running. Therefore, check the import status by clicking **View import history**.

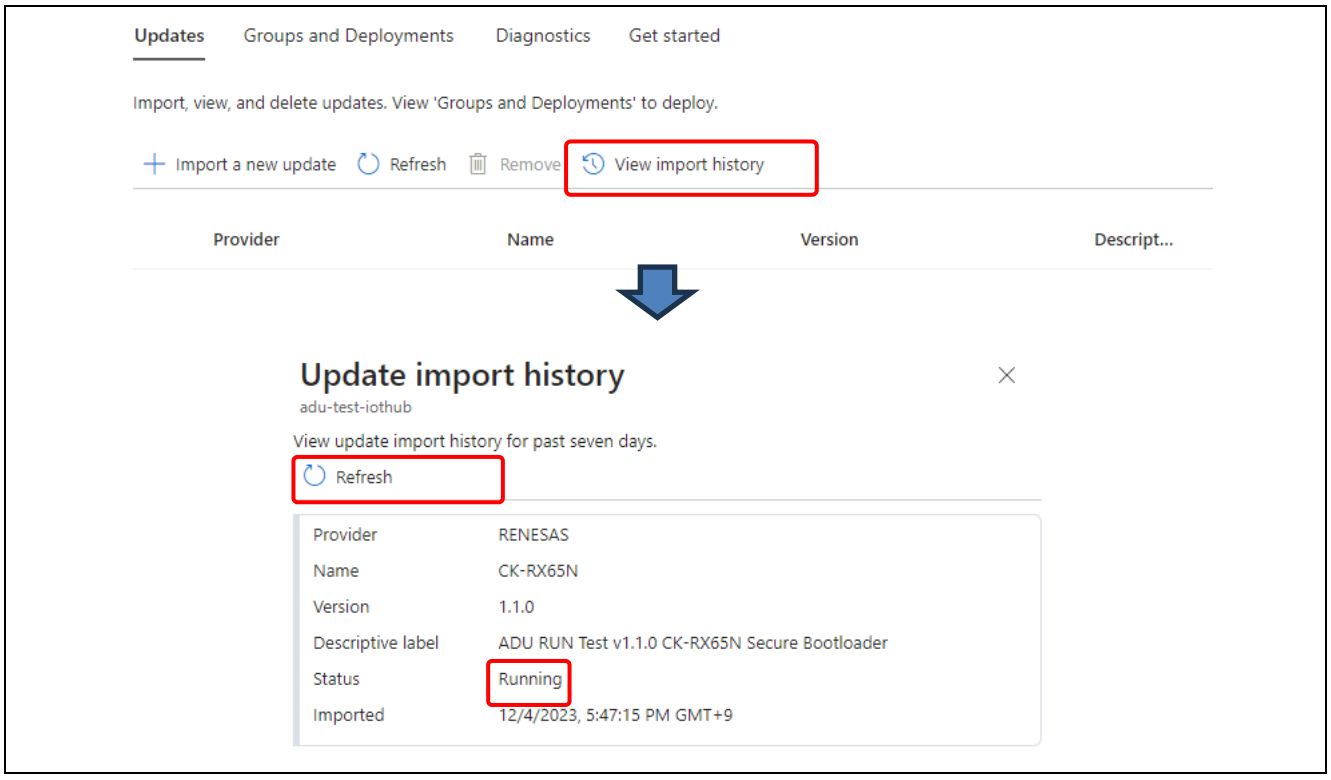

Figure 3.24 Viewing the Import History

The import processing takes some time.
If the status is **Running**, the import processing is in progress. Therefore, wait for a while.
Note that the import history is not automatically refreshed. To see the latest information, manually click the **Refresh** button.
When the status becomes **Succeeded**, the import processing is complete. If the import processing fails, correct the problem, and then reperform the processing.

After the import processing ends normally, when you click the **Refresh** button, the imported firmware is added to the update program list.

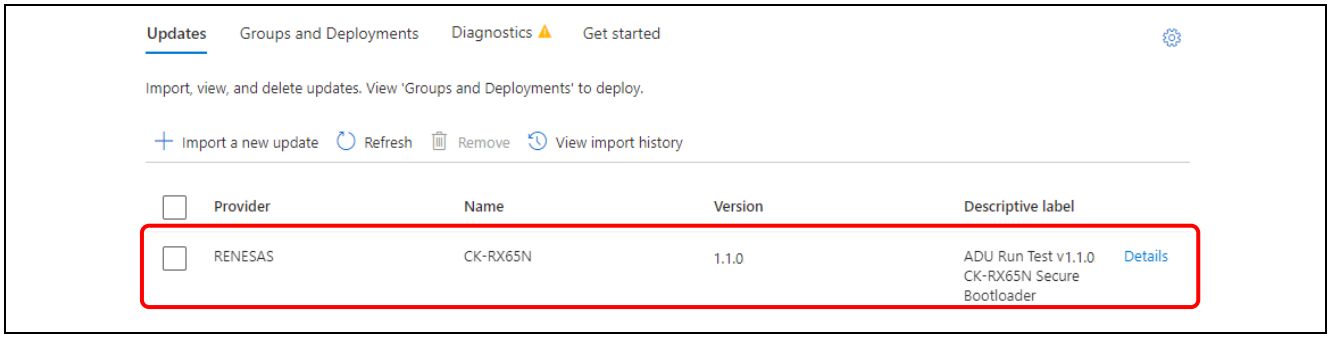

Figure 3.25 List of Updates

3.7 Creating an ADU Group

Add an ADU group to the update. Configure settings to add an ADU group in order to link the IoT Hub device and the ADU group. On the **IoT Hub** page, click **Devices** under **Device management**. From the list of devices, select the device created as described in 3.1 to open the **Device settings** page. On the **Device settings** page, click **Tags (edit)** to display the **Edit tags** page.

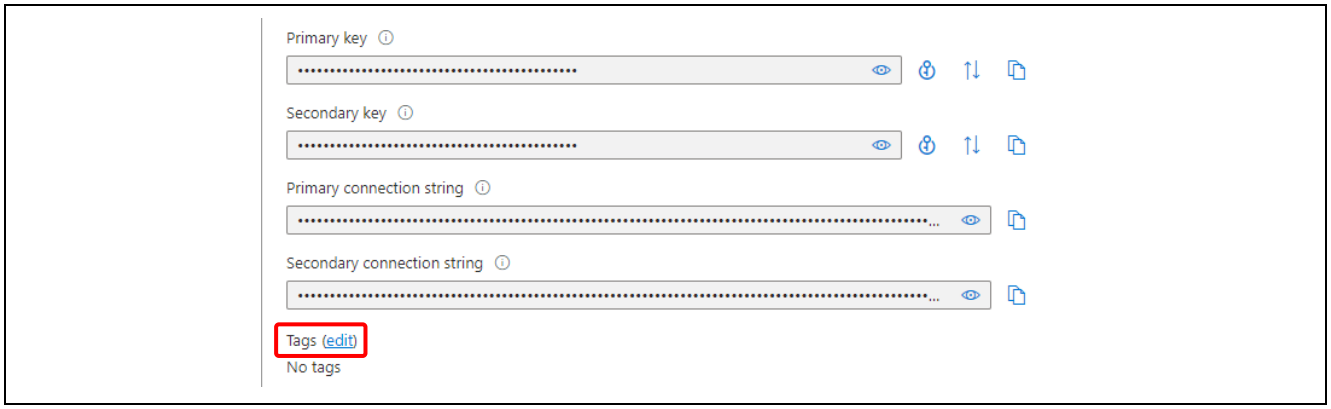

Figure 3.26 Device settings Page

On the **Edit tags** page, enter values for **Name** and **Value**. For **Name** enter **ADUGroup**, and for **Value** enter a character string of your choice. After entering the above, click the **Save** button to close the **Edit tags** page. Confirm that the **Name** and **Value** tags you specified have been registered on the **Device settings** page, then click the **Save** button at the top left of the page.

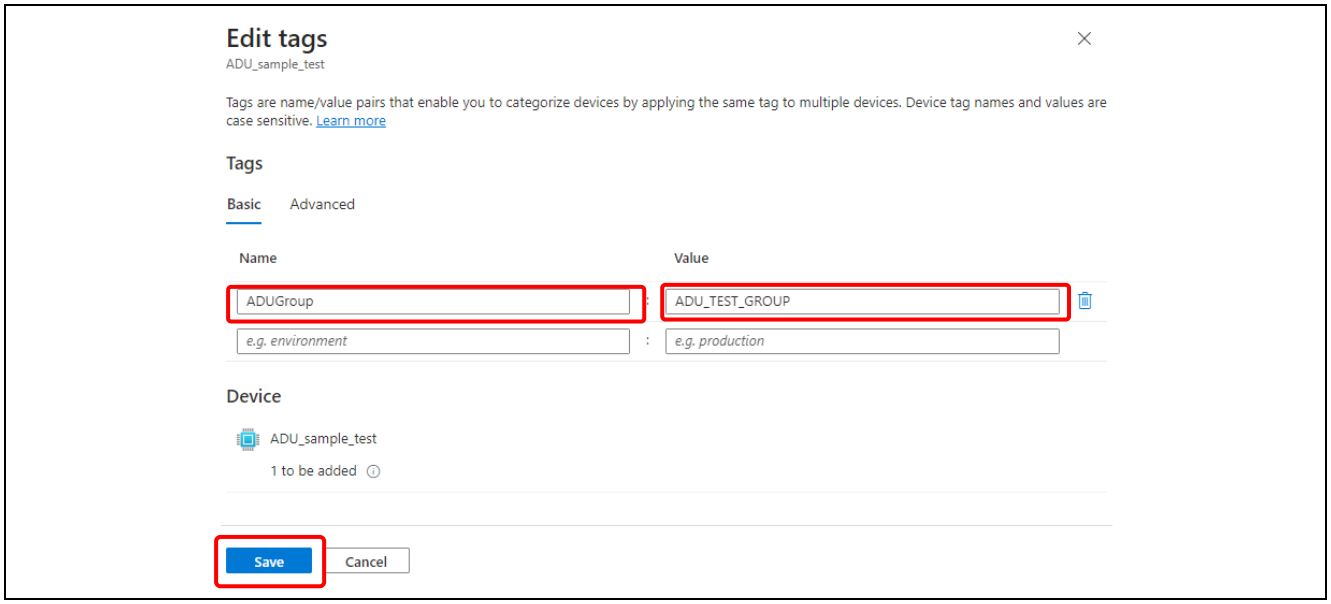

Figure 3.27 Edit tags Page

On the **IoT Hub** page, click **Updates** under **Device management** → **Groups and Deployments**. The group for which you specified the **Value** tag should have been added to the list of groups that is displayed.*[1]

Note: 1. A group may fail to be added to the list of ADU groups in the case of a device where no communication has ever taken place with the target board. In this case, connect once from the target board to the IoT Hub device to be used with ADU.

3.8 Updating the Firmware

Download the firmware update registered to the IoT Hub to the target board.

3.8.1 Execution on the Target Board

To begin, run the initial firmware previously programmed to the target board. For the CK-RX65N, operation starts when you connect the USB of J14 to a USB port on the PC to supply power by connecting a jumper line from JP16 to the RUN side.

Also, by connecting a PC to the J20 USB connector it is possible to monitor the operating state using a terminal emulator program such as Tera Term. When you establish a USB connection between J20 and the PC, a port is registered in Windows Device Manager. You can then connect to the target board by specifying the newly registered COM port number in the terminal emulator program. Configure the serial port communication settings as follows.

- Data rate: 115,200 bps
- Data bits: 8
- Stop bits: 1
- Parity: None

When you run the project, the sample program sends status information as serial output to the terminal emulator program, as shown in the example below.*[1] Check the text displayed by the terminal emulator program to confirm that a connection has successfully been established to the IoT Hub and device. Also confirm the current firmware version indicated following **Installed Criteria:**.

Note: 1. The information output to the terminal emulator program may differ depending on which project you built.

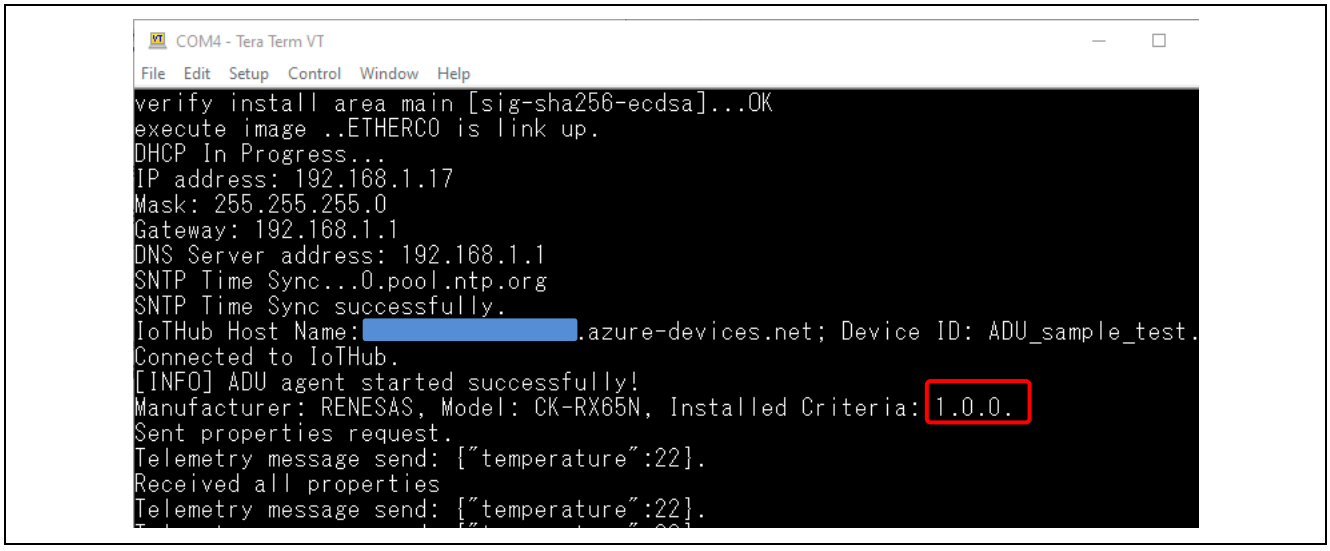

Figure 3.28 Output in Terminal Emulator Window (Example)

3.8.2 Deploying the Firmware Update

Deploy the firmware update from the IoT Hub to the environment, and update the target firmware. For the IoT Hub you are using, click **Updates** under **Device management** to display a list of updates, and then click **Groups and Deployments**. A list of devices in the device group is displayed.

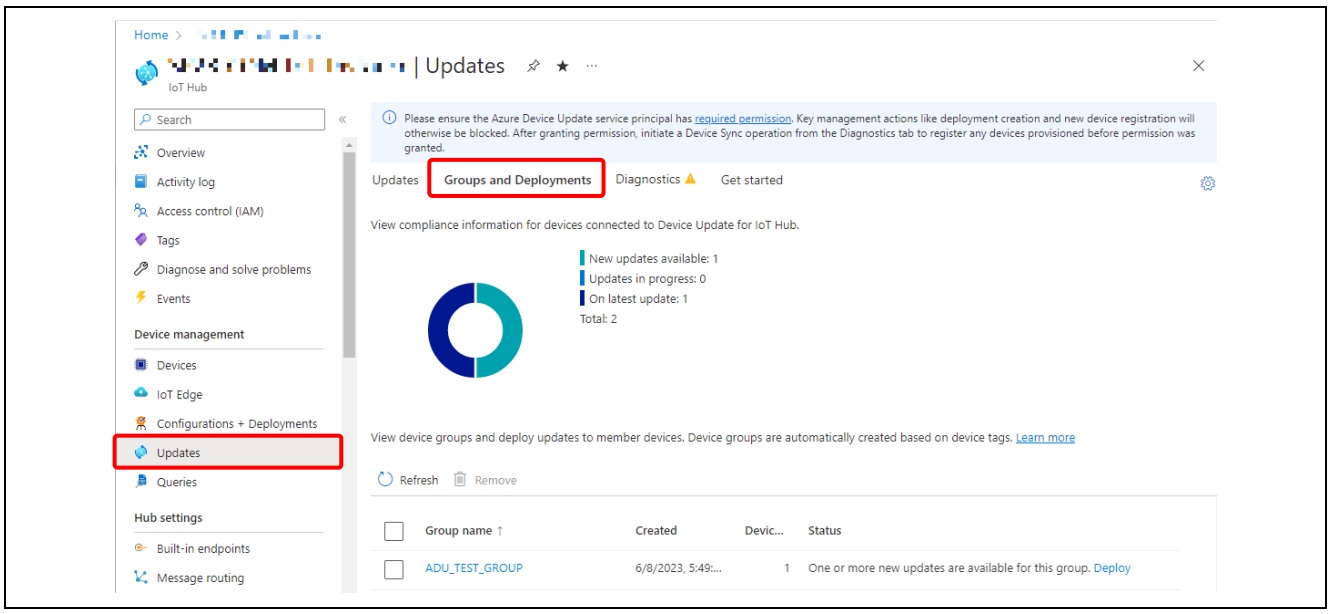

Figure 3.29 Groups and Deployments Display

Groups are displayed in the list of device groups using ADU group names registered as described in 3.6. Click the group name to be used as the firmware update. The **Group details** page appears. On the **Group details** page, click the **Group basics** tab and then click **Deploy** in the center of the page. Alternatively, you can click **Deploy** next to the desired group name on the **Groups and Deployments** page.

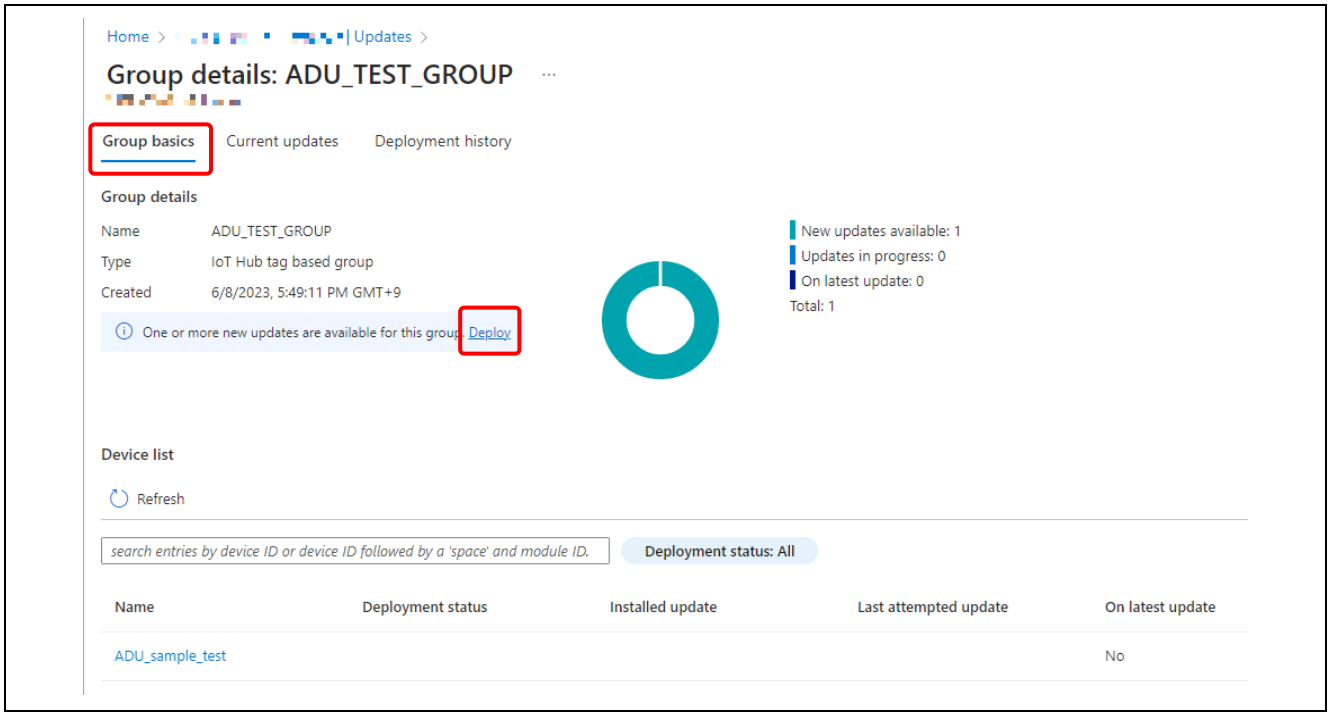

Figure 3.30 Group details Page

On the **New updates** page that appears, check that the version number is correct, then click the **Deploy** button. The **Create deployment** page is displayed.

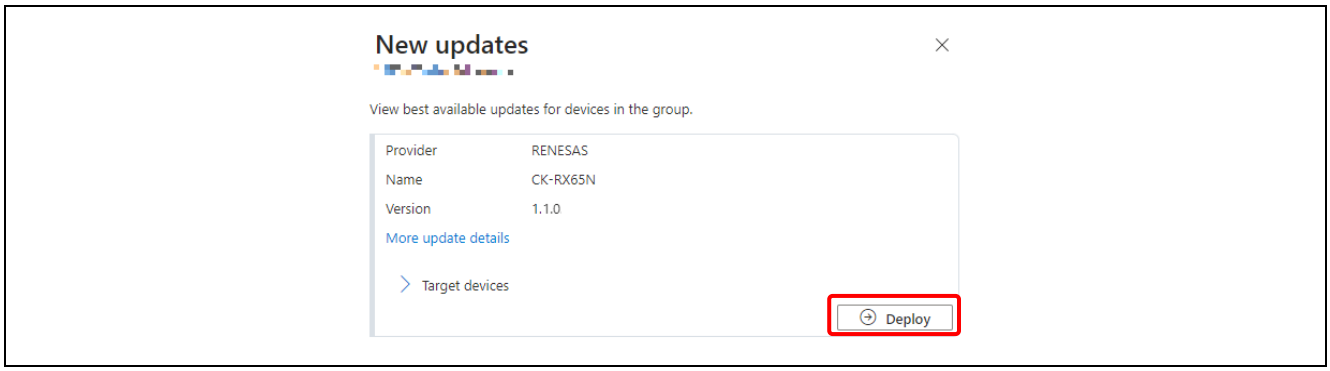

Figure 3.31 New updates Page

On the **Create deployment** page, click the **Create** button. Deployment starts.

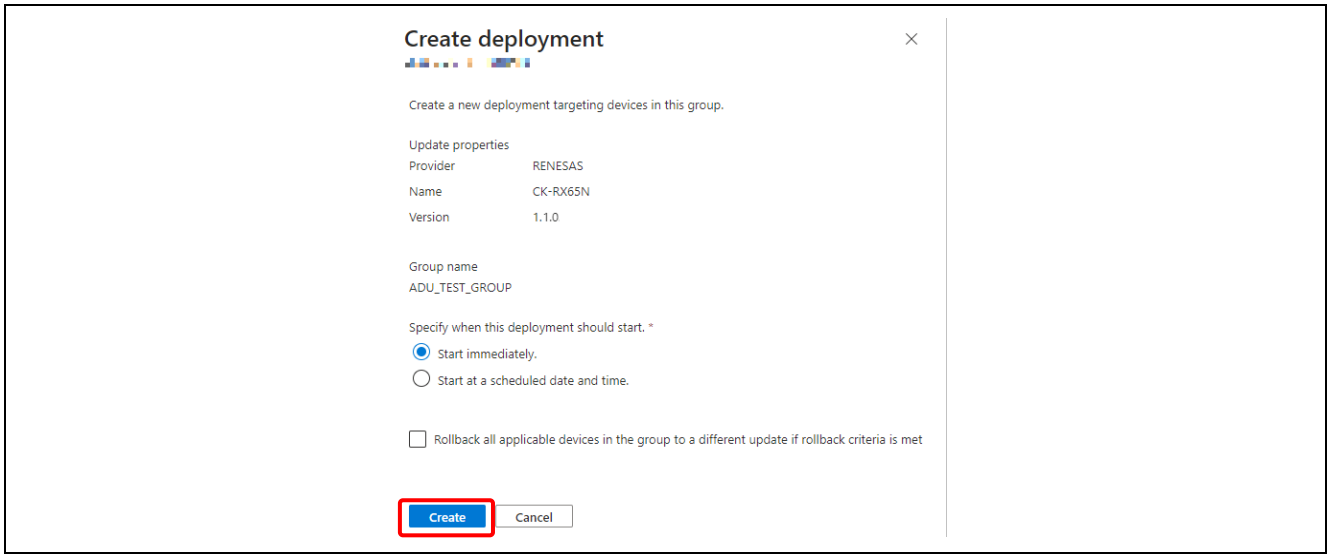

Figure 3.32 Create deployment Page

Once deployment starts, the download status is output to the terminal emulator program connected to the CK-RX65N. The downloaded firmware update is written to the flash memory, and version checking and verification data are used to verify the firmware. Next, bank switching takes place on the device, and the firmware update is applied after a software reset.

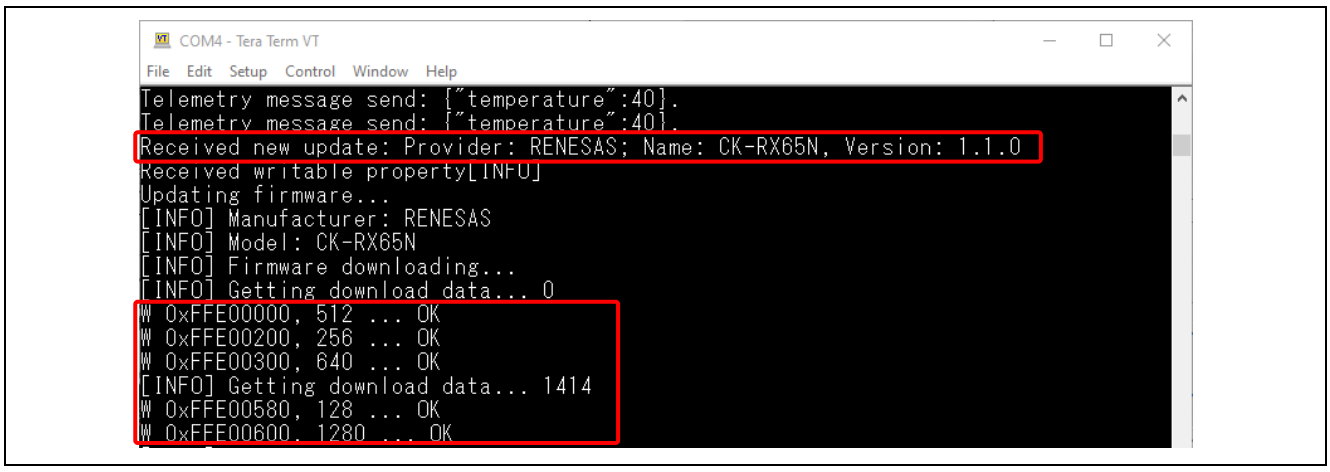

Figure 3.33 Downloading the Firmware

After the firmware is downloaded, the update is complete when the firmware version indicated following **Installed Criteria:** matches that of the firmware update.

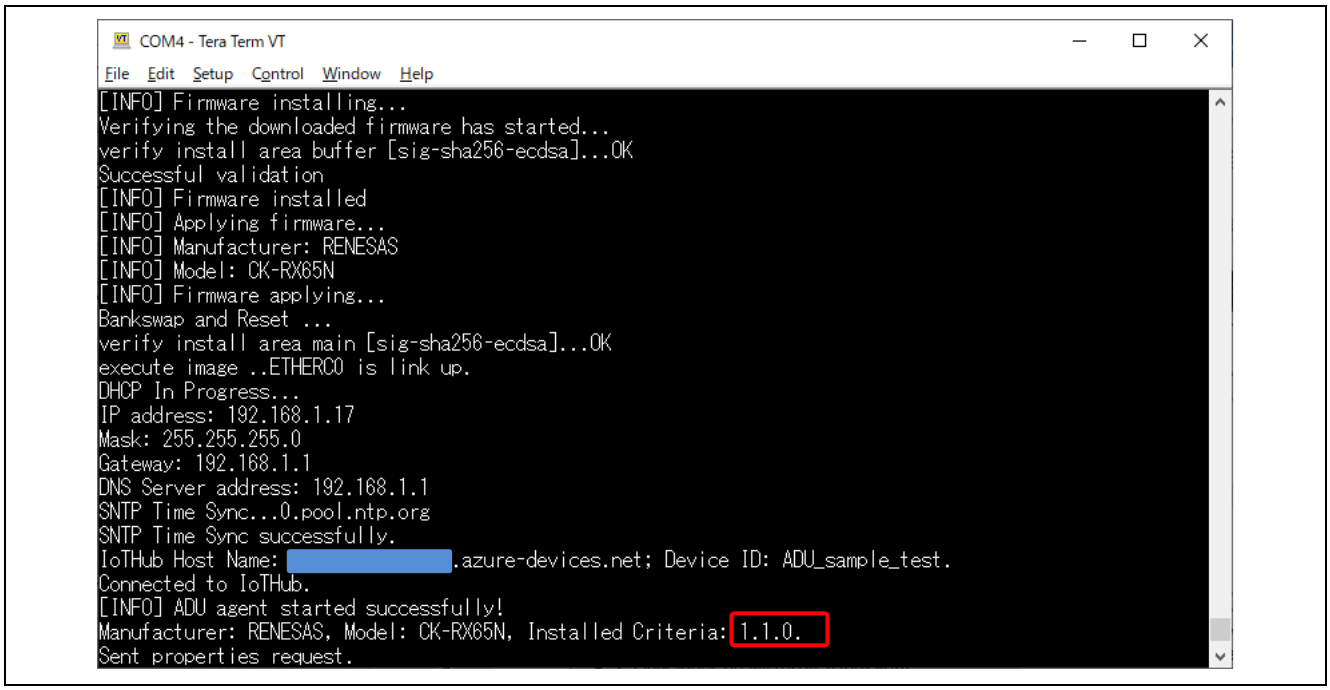

Figure 3.34 Firmware Update Complete

After the firmware update finishes, the **Deployment status** shown on the **Group details** page is **Succeeded**.

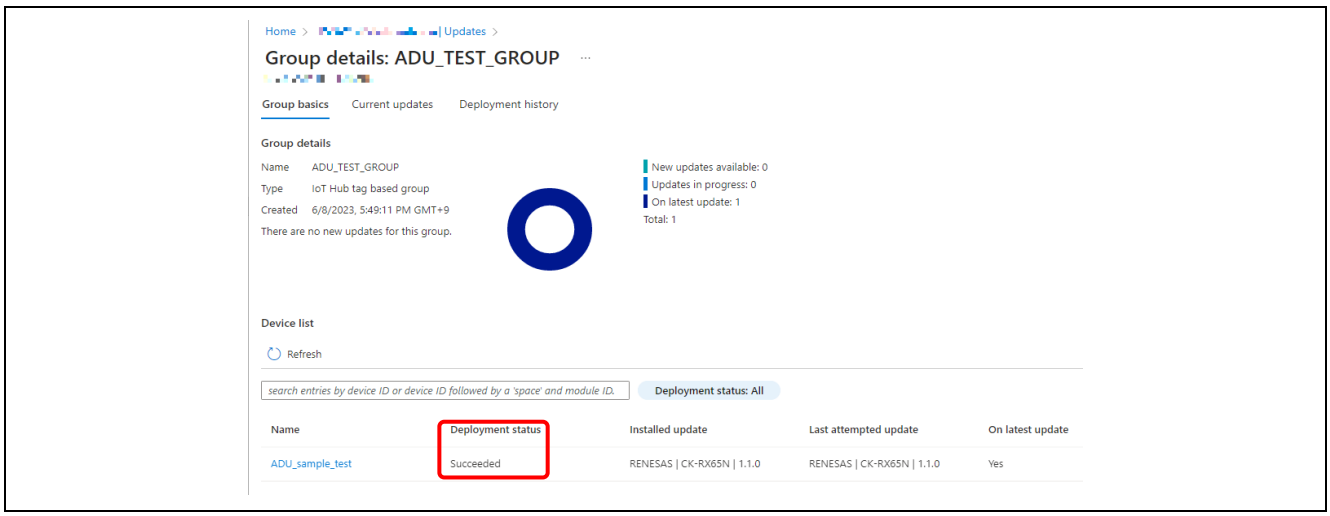

Figure 3.35 Group details Page

You can deploy a version of the firmware that is not the latest version by clicking the applicable group name in the device group list, clicking the **Current updates** tab on the **Group details** page, and then clicking **View available updates** under **Deployment details** to display the **Available updates** page. You can then select the desired version from a list and deploy it.

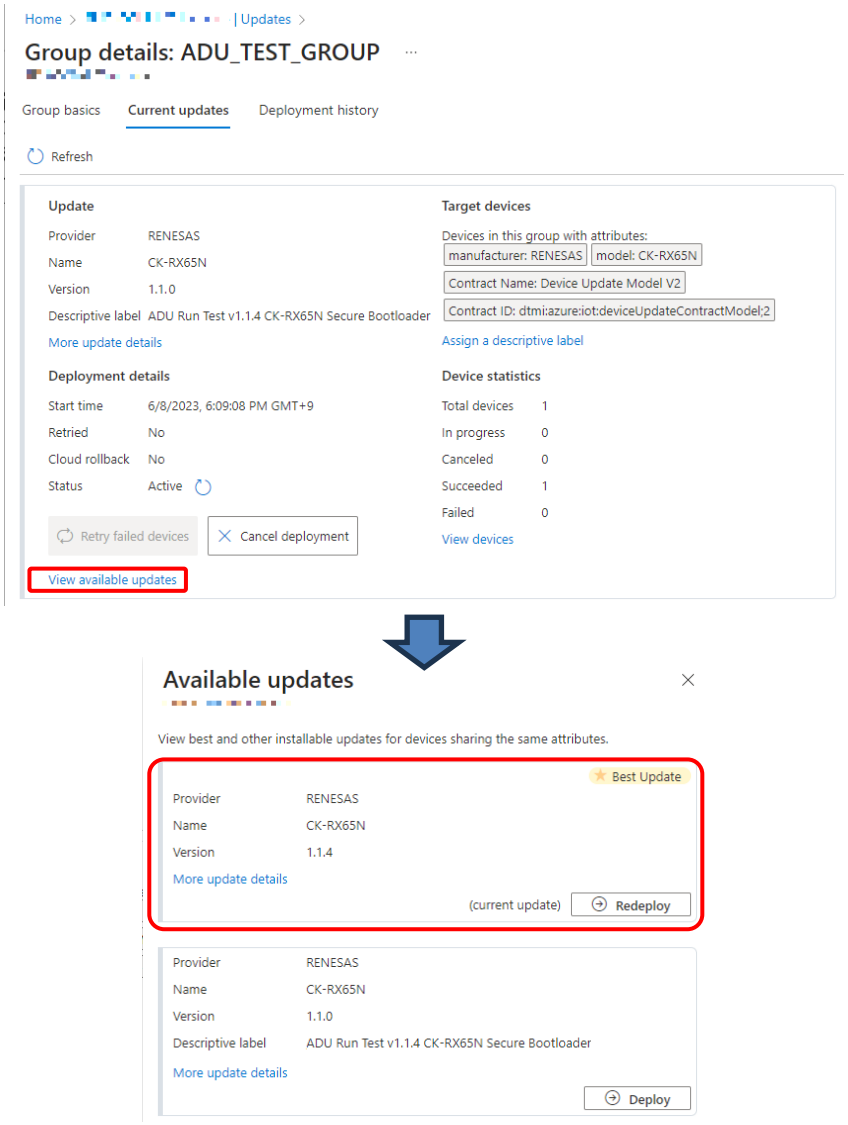

Figure 3.36 Selecting Among Available Updates

4. Appendix

4.1 Control of Commands in Azure ADU and the Sample Project

This sample project performs several kinds of control for updating the firmware according to the commands received from Azure. The application that controls these commands is called an “ADU agent”. This control is the main processing that this sample project performs.

The ADU agent performs various kinds of processing according to the request commands received from Azure.
The following table lists the request commands supported by this sample project.

Command	Function in this sample project	Support status
INITIALIZE	Initializes the firmware update module and other items.	Supported
PREPROCESS	Performs no processing in this sample project.	Supported
WRITE	Writes the firmware data that was sent from Azure to flash memory.	Supported
INSTALL	Installs firmware. This command also verifies the firmware written to the flash memory. If verification fails, this command deletes the update firmware.	Supported
APPLY	Applies the firmware. In this sample project, this command switches the startup bank, and then performs a software reset.	Supported
CANCEL	Not supported by this sample project.	Not supported
UPDATE_CHECK	Not supported by this sample project.	Not supported

The request commands listed above are common to the sample projects that support firmware update modules v1 and v2.

4.2 Firmware Debugging Method

How to source debug initial and updated firmware using e² studio is described below.

4.2.1 Debugging the Initial Firmware

1. Change the debug settings for the **bootloader** project as follows in e² studio.
   From the menu bar select **Run** → **Debug Configurations...**, and then click **bootloader Hardware Debugging** in the pane at the left of the **Debug Configurations** window. Click the **Main** tab, click the **Browse…** button under **C/C++ Application:**, and select the **userprog.mot** file containing the initial firmware created as described in 2.9, Creating the Initial Firmware.

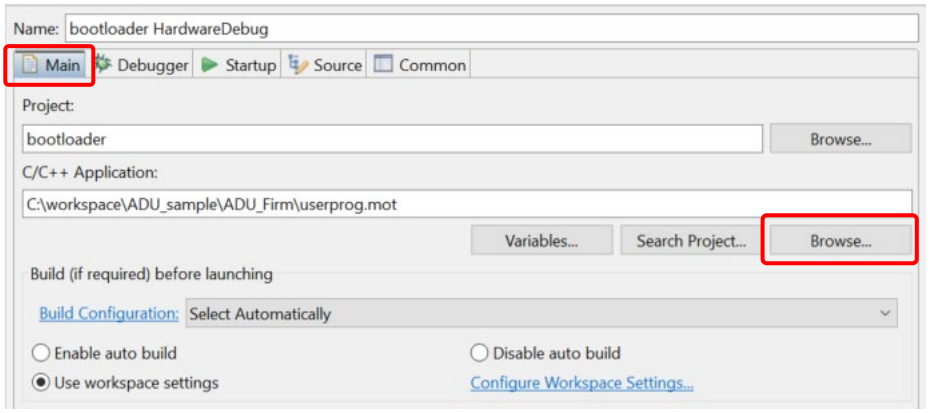

Figure 4.1 Specifying userprog.x for C/C++ Application

In the **Debugger** tab, open the **Connection Settings** tab. Then, set **Change startup bank** to **Yes** and **Startup bank** to **Bank 0**.

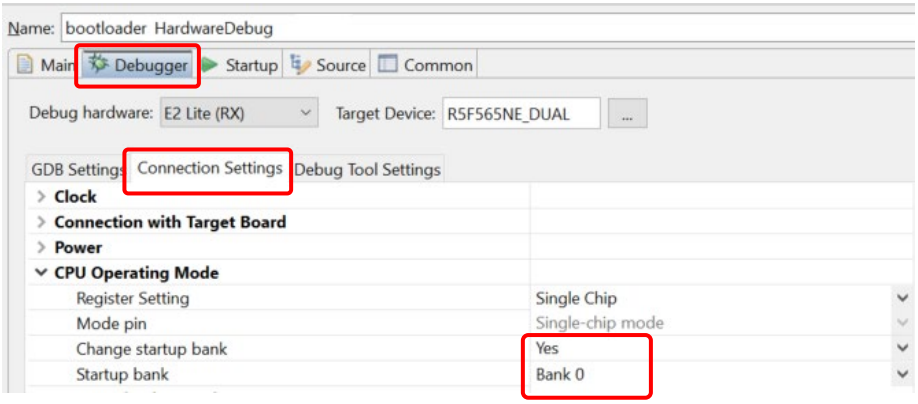

Figure 4.2 Changing the Startup Bank

Select **Debugger** tab → **Debug Tool Settings** and set **Debug the program re-writing the on-chip PROGRAM ROM** to **Yes**.

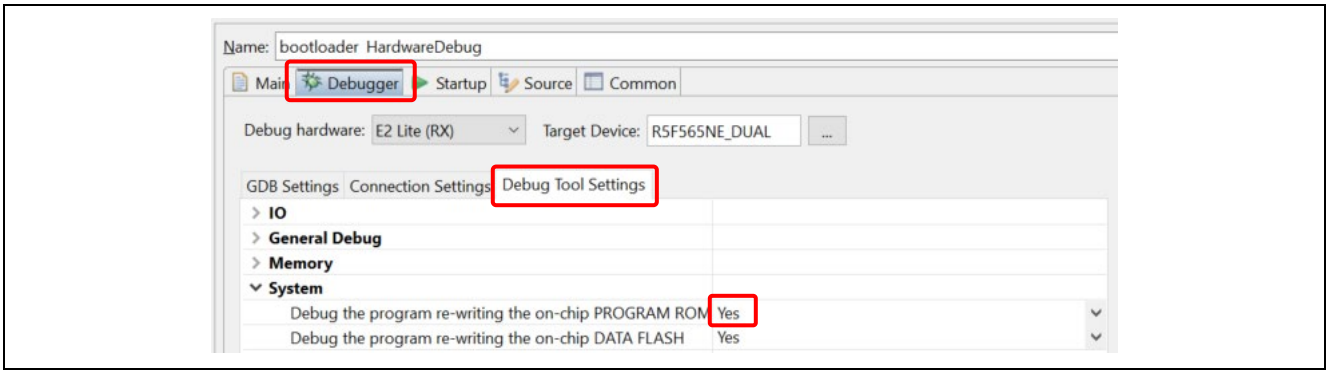

Figure 4.3 Debug Tool Settings Tab

2. In the debug settings for the same **bootloader** project, click the **Startup** tab and under **Load image and symbols** add items and settings as follows.
   - Change the **Load type** of **userprog.mot** to **Image only**.
   - Add **adu_sample.x** by clicking **Add…** → **File system...** and change the **Load type** to **Symbols only**.
   - Add **bootloader.x** by clicking **Add…** → **File system...** and change the **Load type** to **Symbols only**.

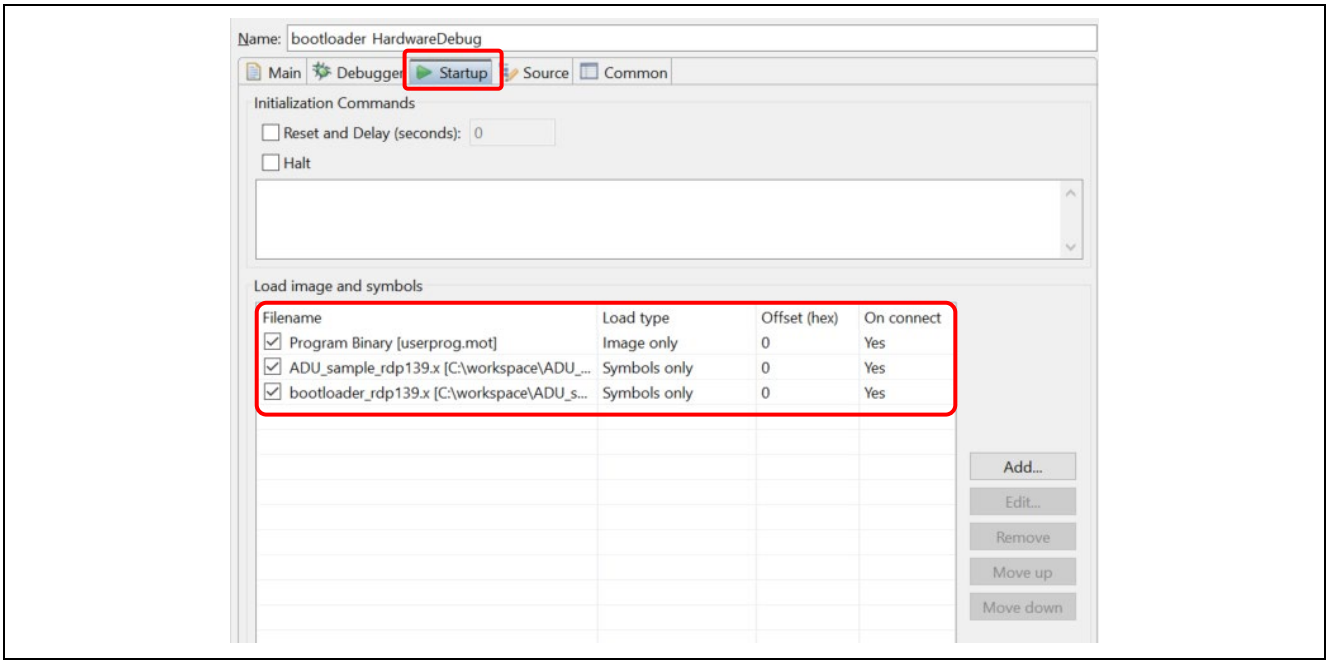

Figure 4.4 Load image and symbols Settings

3. Start debugging the **bootloader** project and make sure it breaks at the main function of **bootloader/src/bootloader.c**. The following figure is an example when the version of the firmware update module is v1.

[bootloader.c]

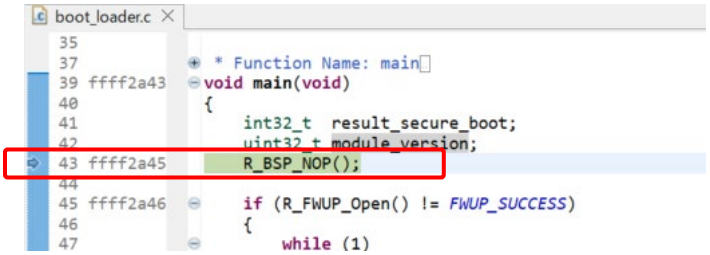

Figure 4.5 Breakpoint in bootloader.c

Also, check that the firmware starts normally by setting a breakpoint at the main function in **adu_sample/src/main.c**.

[main.c]

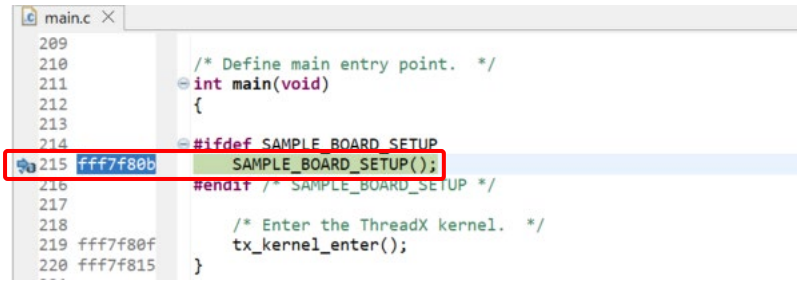

Figure 4.6 Breakpoint in main.c

If execution does not break at the main function, set a breakpoint at the location shown above.

When generating the initial firmware **userprog.mot**, a MOT file with blank area filled with 0xFF is generated. In other words, even if you are not using the data flash area, this area is filled with 0xFF. At this time, if you rewrite the data in the data flash area by the program, download initial firmware again, and execute debugging, it will be overwritten with 0xFF.

In this case, copy the rewritten data flash area before downloading initial firmware, and write it back after downloading. In the case of e^2 studio, it is possible to output the memory contents to a file and read it from the file by using the **dump/restore** command.

Example:
To output the data flash area at address 0x100000-0x107FFF to S-format file and read it, execute the following GDB command in the **Debugger Console** view.

- When outputting to **memdump.mot** file

```
dump srec memory memdump.mot 0x100000 0x107FFF
```

- When writing from **memdump.mot** file to memory

```
restore memdump.mot 0 0x100000 0x107FFF
```

Note that **memdump.mot** is generated in the project folder.

4.2.2 Debugging the Updated Firmware

To debug the firmware update, create separate projects for the initial firmware and firmware update. Running the bootloader executes the initial firmware.

1. After the initial firmware starts, set a breakpoint before the bootloader boots the update firmware. The position at which to set a breakpoint differs depending on whether the version of the firmware update module is v1 or v2. For details, refer to the following sections.
   (a) If the version of the firmware update module is v1
   The following process in the **bootloader/src/smc_gen/r_fwup/src/r_fwup_boot_loader.c** file is the process that boots the updated firmware.

[r_fwup_boot_loader.c]

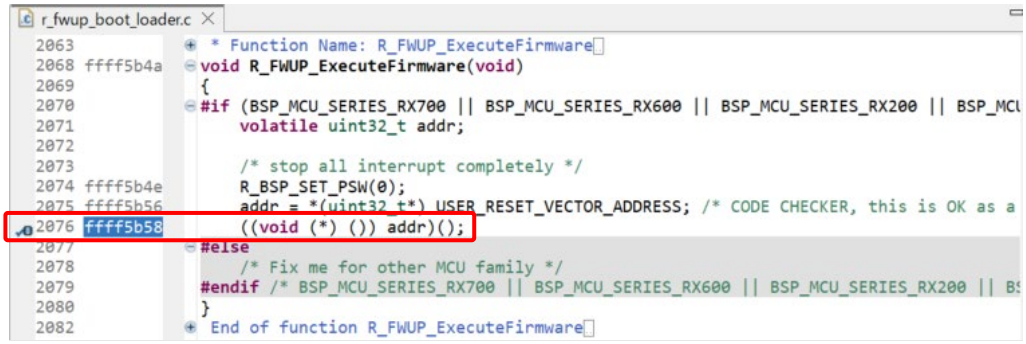

Figure 4.7 Position At Which to Set a Breakpoint (If the Version is v1)

(b) If the version of the firmware update module is v2
Firmware is started by the following processing in the “bootloader/src/smc_gen/r_fwup/src/r_fwup.c” file:

[r_fwup.c]

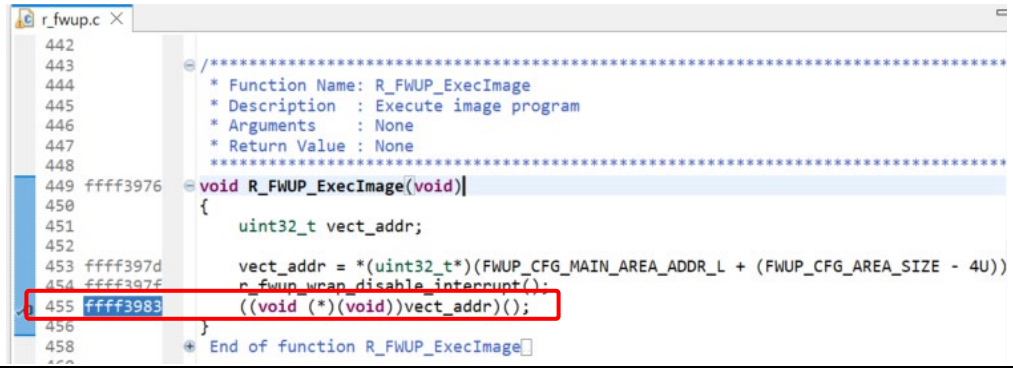

Figure 4.8 Position At Which to Set a Breakpoint (If the Version is v2)

2. After updating the firmware with ADU, a bank swap and software reset occur.
3. Just before the updated firmware starts, it breaks at the breakpoint specified in step 1 above.
4. After the break, execute the following GDB command to update the symbol information. Use the **Debugger Console** at the bottom of e^2 studio to execute GDB commands.
   Example: In the case where the .x file of update firmware is located in “C:\workspace\ADU_sample\HardwareDebug”*1
   The following is an example when the .x file of update firmware is “ADU_sample.x”.
   symbol-file C:/\workspace/\ADU_sample/\HardwareDebug/\ADU_sample.x -readnow*1

Note: 1. Add a forward slash (/) before each backslash (\) when specifying the path.

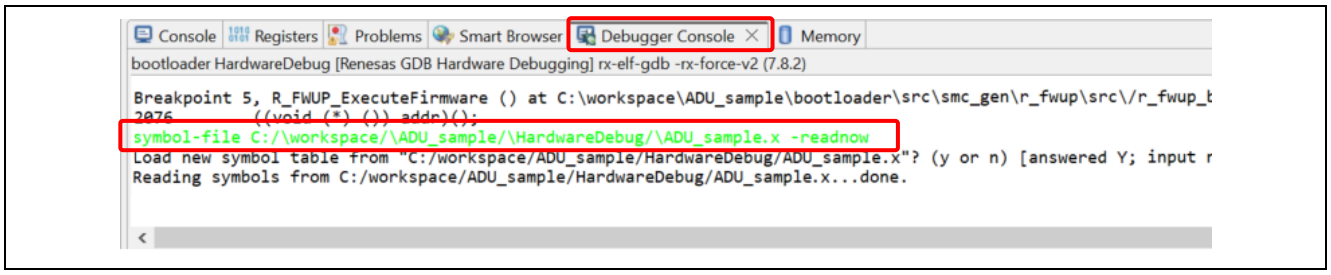

Figure 4.9 Command to Update Symbol Information

5. After you set a breakpoint in any source code of the updated firmware and restart debugging, confirm that it breaks at that breakpoint.

Online technical support and information is available at: https://en-support.renesas.com/dashboard

Technical contact details: https://www.renesas.com/support/contact.html

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	July 31, 2023	—	First edition issued
2.00	Dec. 21, 2023	1	Updated the development environment used.
		2	Added a table that lists combinations of a target board and a firmware update module (FWUP).
		5 to 8	Added a description of memory area allocation and operation (FWUP v1/v2). Added a description of area allocation in data flash memory.
		11, 12	Changed the Azure RTOS version and target board type to be specified. Added a description of dual mode.
		17	Added a description of BGO mode.
		17 to 20	Modified the description of adding missing components. Added a procedure for installing FWUP v2.
		21, 22	Added a description of code generation during the build processing. Deleted the procedures for changing the dual mode, section, and symbol settings, because it is no longer necessary to change these settings.
		26 to 29 35 to 41	Added a procedure for creating firmware that supports FWUP FIT v2.
		34	Modified the section that describes execution of the initial firmware.
		42 to 47	Added a supplementary note on creating an IoT hub. Added procedures for creating a Device Update account and a storage account.
		58	Added a description of a warning message (about a storage container) that appears when uploading firmware.
		66	Added a description of controlling Azure ADU commands.
		70	Added a description about FWUP FIT v2 to the update firmware debugging method.
2.01	July 1, 2024	-	Supported Azure RTOS v.6.4.0
		12	Added a description of the Bank Mode setting.

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/.

© 2024 Renesas Electronics Corporation. All rights reserved.