RENESAS

Application Note

RX Family
How to implement OTA by using Microsoft Azure Services

R01AN6928EJ0100
Rev.1.00
Jul.31.23

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): e^2 studio 2023-04

Compiler:	Renesas C/C++ Compiler for RX Family CC-RX V3.05.00
	GCC for Renesas 8.3.0.202204-GNURXGCC
Driver package (RDP):	RX Driver Package V1.39
Azure RTOS	: 6.2.1_rel-rx-1.0.1
Flash programming tool:	Renesas Flash Programmer V3.11.02
MOT file conversion tool:	Renesas Secure Flash Programmer (RX MCUs mot file converter 2.0.2) (Installation procedure described separately.)
Key generation tool:	Win32/Win64 OpenSSL v3.1.1 Light (Installation procedure described separately.)

Contents

1. Memory Allocation for ADU

The description in this document assumes that the memory is allocated as shown in the figures below.

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

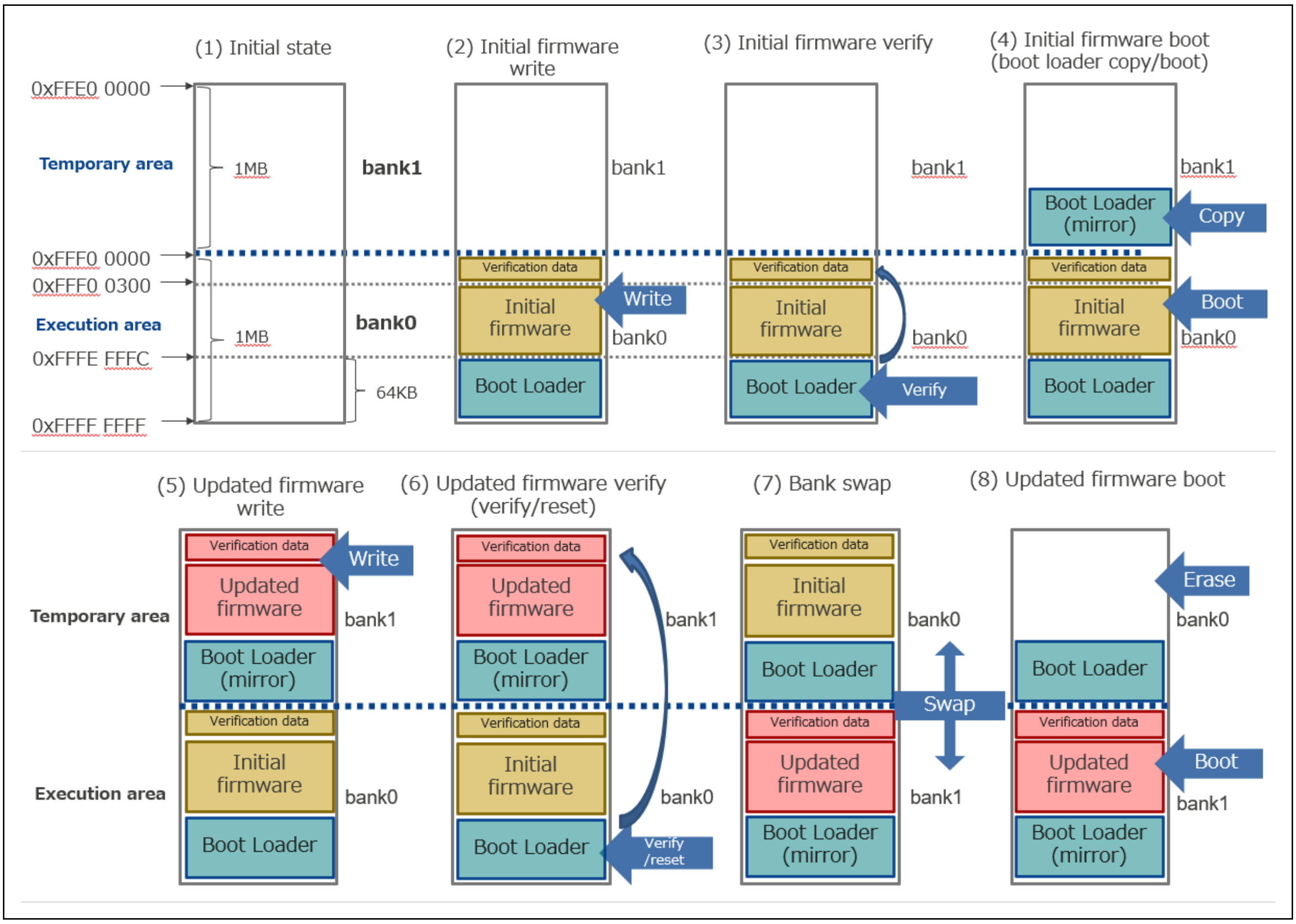


Figure 1.1 Memory Allocation for ADU (RX65N)

The various types of data are written to the following addresses in memory.

0xFFF00000 to 0xFFF002FF: Verification data

0xFFF00300 to 0xFFFEFFFB: Firmware

0xFFFEFFFC to 0xFFFFFFFF: Secure bootloader (Boot Loader)

The secure bootloader uses the verification data written to the address range 0xFFF00000 to 0xFFF002FF to verify that the previously programmed initial firmware and the updated firmware have not been tampered with ((3) and (6) in the above figure).

After the firmware update, the bank swapping functionality is used to exchange the memory areas containing the initial firmware and updated firmware, and then the old firmware is erased ((7) and (8) in the above figure). Utilizing the bank swapping functionality makes it possible for the addresses referenced by the application to remain unchanged after the firmware update.

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.

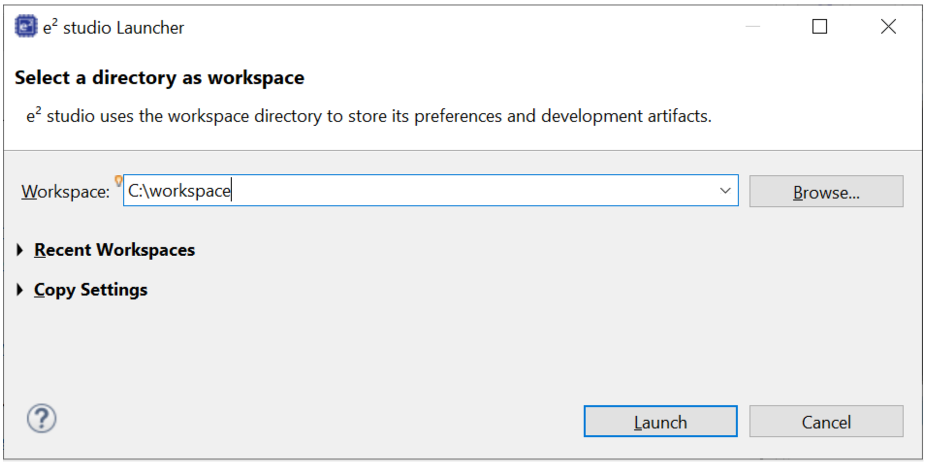
- Azure Device Update (ADU) sample project
- Secure bootloader sample project

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 e^2 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



e² studio Launcher

Select a directory as workspace

e² studio uses the workspace directory to store its preferences and development artifacts.

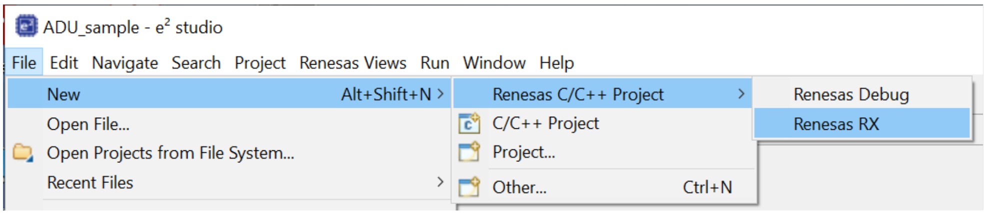
Workspace: C:\workspace

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.


ADU_sample - e² studio

File Edit Navigate Search Project Renesas Views Run Window Help

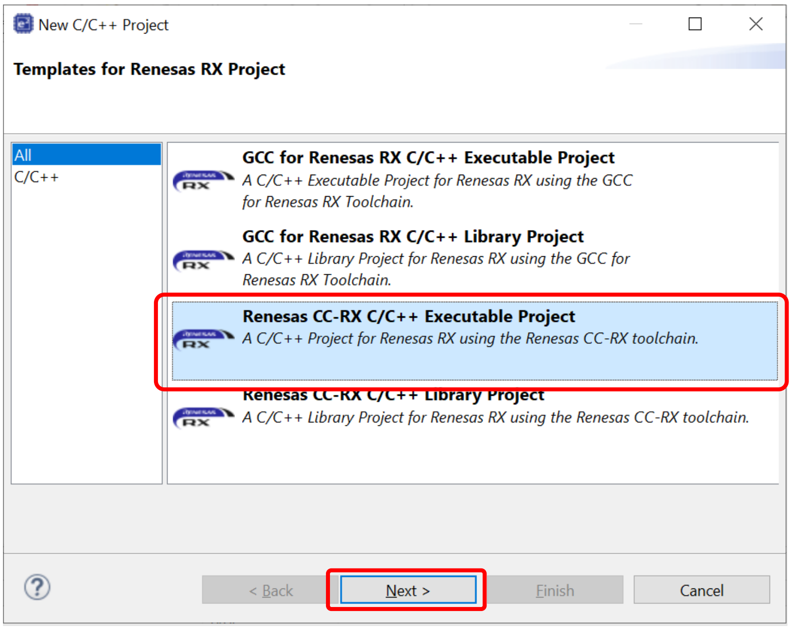
New Alt+Shift+N >
Open File...
Open Projects from File System...
Recent Files >

Renesas C/C++ Project >
C/C++ Project
Project...
Other... Ctrl+N

Renesas Debug
Renesas RX

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**.


New C/C++ Project

Templates for Renesas RX Project

All
C/C++

GCC for Renesas RX C/C++ Executable Project
A C/C++ Executable Project for Renesas RX using the GCC for Renesas RX Toolchain.

GCC for Renesas RX C/C++ Library Project
A C/C++ Library Project for Renesas RX using the GCC for Renesas RX Toolchain.

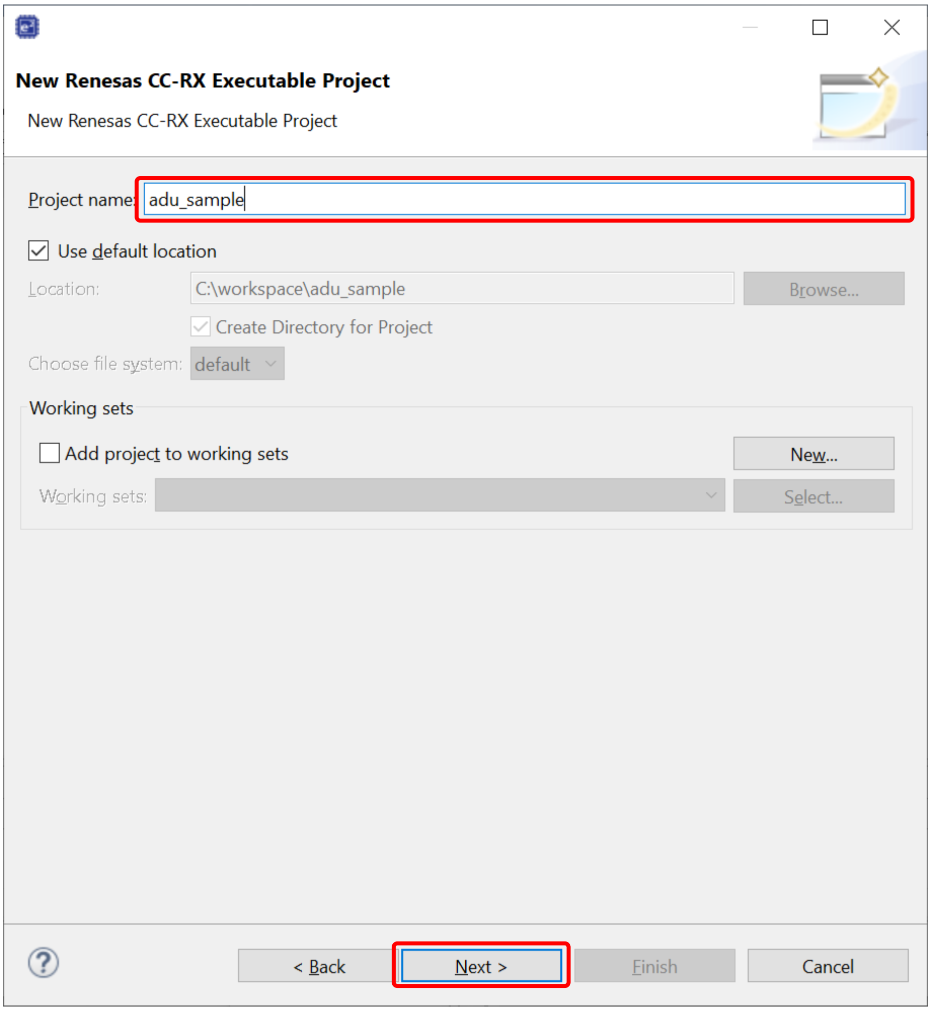
Renesas CC-RX C/C++ Executable Project
A C/C++ Project for Renesas RX using the Renesas CC-RX toolchain.

Renesas CC-RX C/C++ Library Project
A C/C++ Library Project for Renesas RX using the Renesas CC-RX toolchain.

< Back Next > Finish Cancel

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.



New Renesas CC-RX Executable Project

New Renesas CC-RX Executable Project

Project name: adu_sample

☑ Use default location

Location: C:\workspace\adu_sample

☑ Create Directory for Project

Choose file system: default

Working sets

☐ Add project to working sets

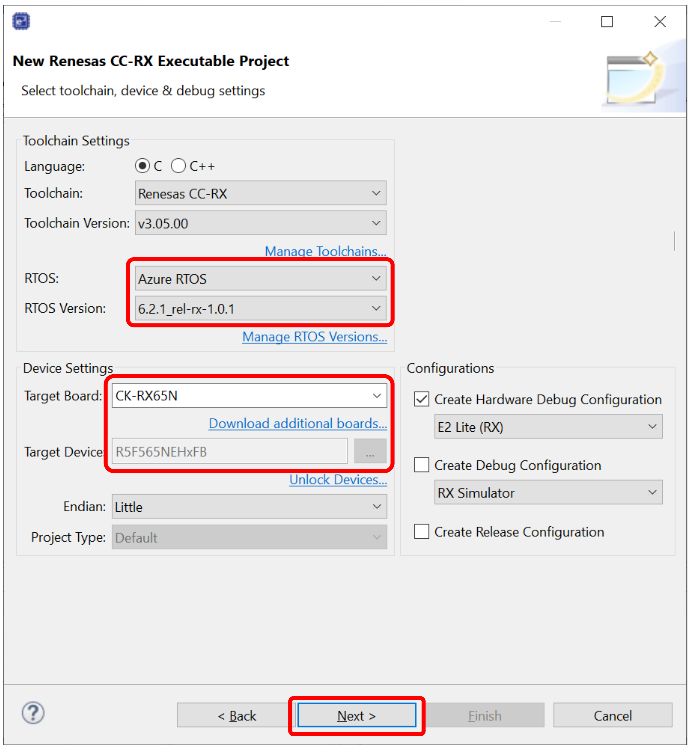
Working sets:

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 the appropriate version. If you are using e^2 studio for the first time or if the version you wish to use is not displayed as an option, click **Manage RTOS Versions...** to open the **RTOS Module Download** dialog box. Check the box next to the version you wish to use and click the **Download** button to download it.

For **Target Board:** select **CK-RX65N**. (**Target Device:** is selected automatically.) When all the settings have been configured, click the **Next >** button.



New Renesas CC-RX Executable Project

Select toolchain, device & debug settings

Toolchain Settings

Language: ● C ○ C++

Toolchain: Renesas CC-RX

Toolchain Version: v3.05.00

Manage Toolchains...

RTOS: Azure RTOS

RTOS Version: 6.2.1_rel-rx-1.0.1

Manage RTOS Versions...

Device Settings

Target Board: CK-RX65N

Download additional boards...

Target Device: R5F565NEHxFB

Unlock Devices...

Endian: Little

Project Type: Default

Configurations

☑ Create Hardware Debug Configuration

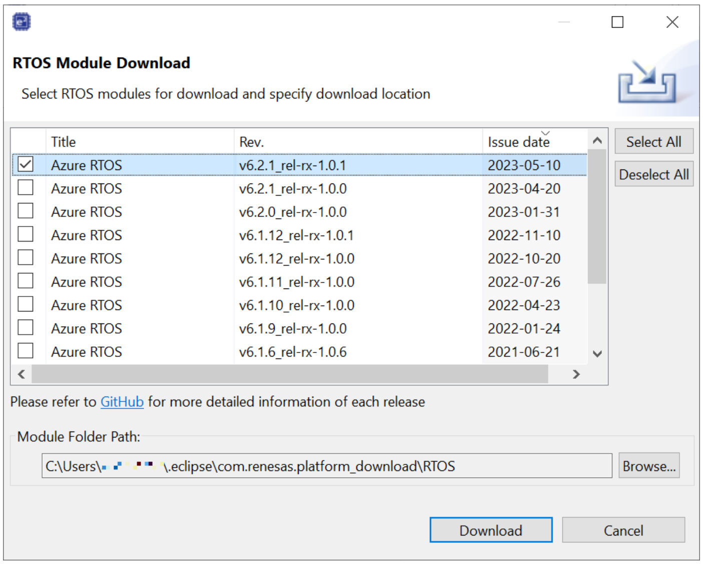
E2 Lite (RX)

□ Create Debug Configuration

RX Simulator

□ Create Release Configuration

Figure 2.5 Select toolchain, device & debug settings Window



RTOS Module Download

Select RTOS modules for download and specify download location

	Title	Rev.	Issue date
☑	Azure RTOS	v6.2.1_rel-rx-1.0.1	2023-05-10
□	Azure RTOS	v6.2.1_rel-rx-1.0.0	2023-04-20
□	Azure RTOS	v6.2.0_rel-rx-1.0.0	2023-01-31
□	Azure RTOS	v6.1.12_rel-rx-1.0.1	2022-11-10
□	Azure RTOS	v6.1.12_rel-rx-1.0.0	2022-10-20
□	Azure RTOS	v6.1.11_rel-rx-1.0.0	2022-07-26
□	Azure RTOS	v6.1.10_rel-rx-1.0.0	2022-04-23
□	Azure RTOS	v6.1.9_rel-rx-1.0.0	2022-01-24
□	Azure RTOS	v6.1.6_rel-rx-1.0.6	2021-06-21

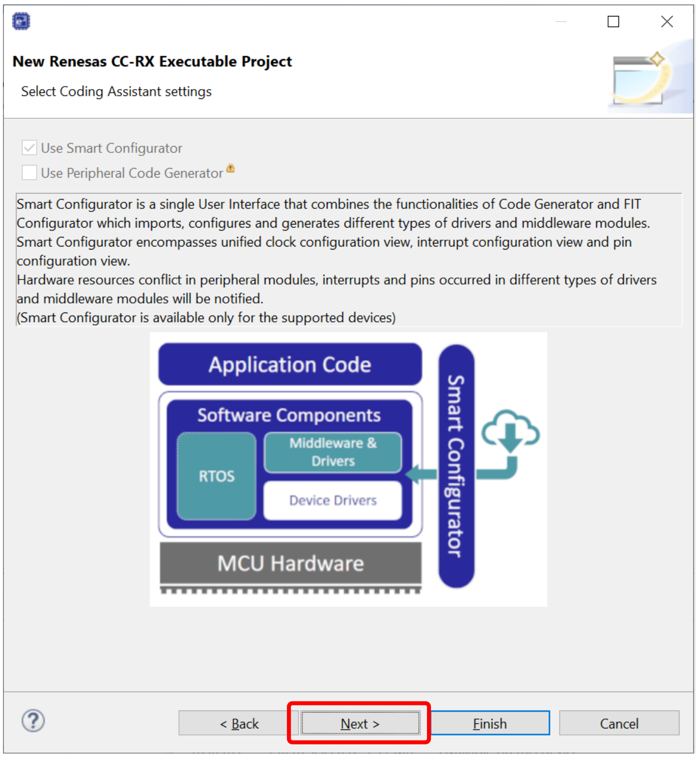
Please refer to GitHub for more detailed information of each release

Module Folder Path:

C:\Users\[redacted]\.eclipse\com.renesas.platform_download\RTOS

Figure 2.6 RTOS Module Download Window

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



New Renesas CC-RX Executable Project

Select Coding Assistant settings

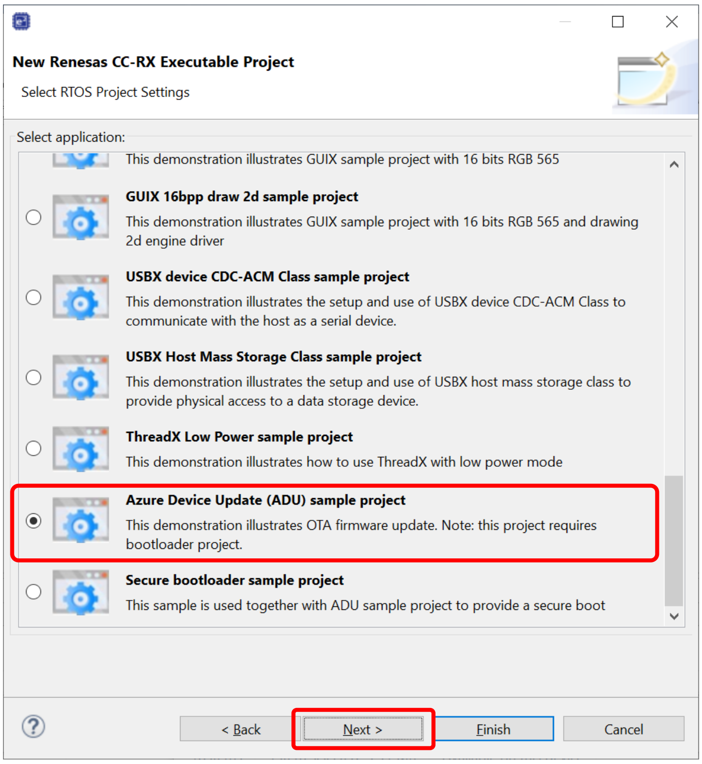
☑ Use Smart Configurator

☐ Use Peripheral Code Generator

Smart Configurator is a single User Interface that combines the functionalities of Code Generator and FIT Configurator which imports, configures and generates different types of drivers and middleware modules. Smart Configurator encompasses unified clock configuration view, interrupt configuration view and pin configuration view.
Hardware resources conflict in peripheral modules, interrupts and pins occurred in different types of drivers and middleware modules will be notified.
(Smart Configurator is available only for the supported devices)

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.



New Renesas CC-RX Executable Project

Select RTOS Project Settings

Select application:

This demonstration illustrates GUIX sample project with 16 bits RGB 565

○ **GUIX 16bpp draw 2d sample project**
This demonstration illustrates GUIX sample project with 16 bits RGB 565 and drawing 2d engine driver

○ **USBX device CDC-ACM Class sample project**
This demonstration illustrates the setup and use of USBX device CDC-ACM Class to communicate with the host as a serial device.

○ **USBX Host Mass Storage Class sample project**
This demonstration illustrates the setup and use of USBX host mass storage class to provide physical access to a data storage device.

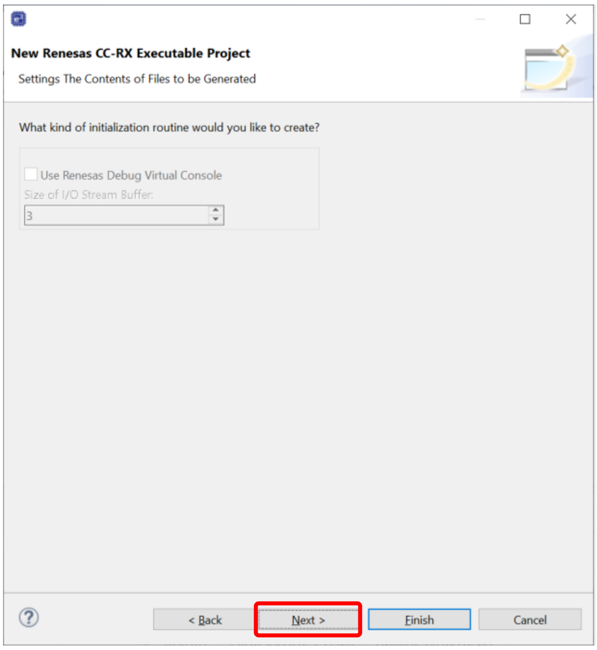
○ **ThreadX Low Power sample project**
This demonstration illustrates how to use ThreadX with low power mode

● **Azure Device Update (ADU) sample project**
This demonstration illustrates OTA firmware update. Note: this project requires bootloader project.

○ **Secure bootloader sample project**
This sample is used together with ADU sample project to provide a secure boot

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.



New Renesas CC-RX Executable Project

Settings The Contents of Files to be Generated

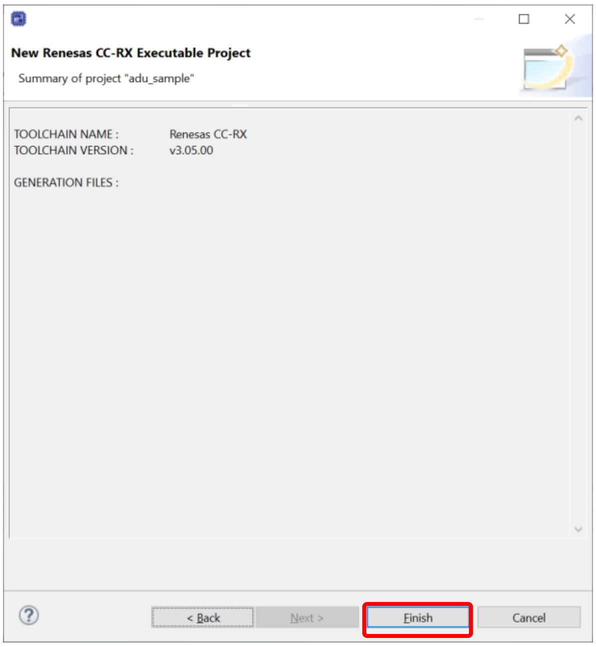
What kind of initialization routine would you like to create?

□ Use Renesas Debug Virtual Console

Size of I/O Stream Buffer: 3

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.



New Renesas CC-RX Executable Project

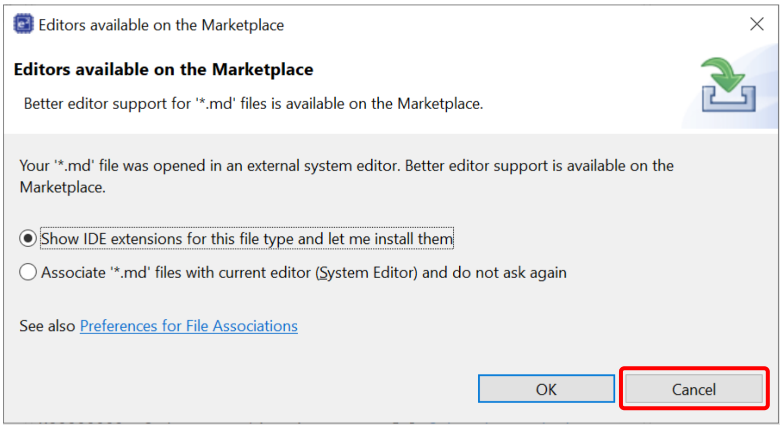
Summary of project "adu_sample"

TOOLCHAIN NAME : Renesas CC-RX
TOOLCHAIN VERSION : v3.05.00

GENERATION FILES :

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.



Editors available on the Marketplace

Editors available on the Marketplace

Better editor support for '*.md' files is available on the Marketplace.

Your '*.md' file was opened in an external system editor. Better editor support is available on the Marketplace.

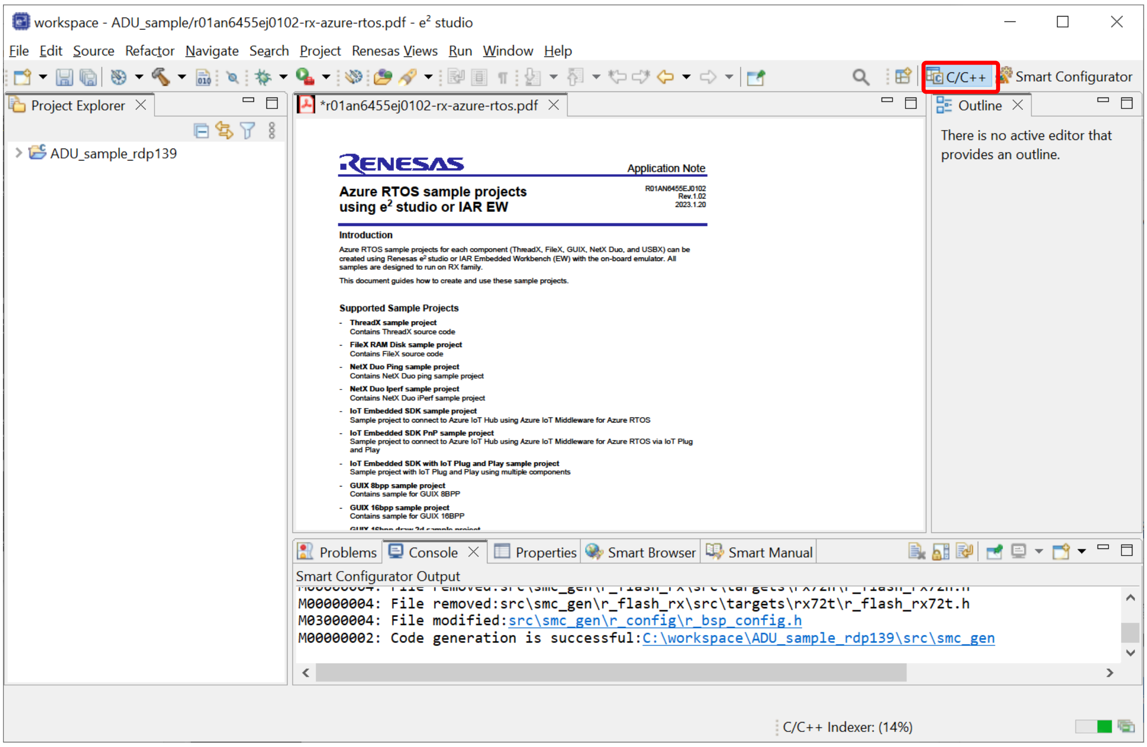
● Show IDE extensions for this file type and let me install them

○ Associate '*.md' files with current editor (System Editor) and do not ask again

See also Preferences for File Associations

Figure 2.11 Editors available in the Marketplace Window

The project is created in e² 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.



workspace - ADU_sample/r01an6455ej0102-rx-azure-rtos.pdf - e² studio

There is no active editor that provides an outline.

RENESAS Application Note

Azure RTOS sample projects using e² studio or IAR EW

R01AN6455EJ0102
Rev.1.02
2023.1.20

Introduction

Azure RTOS sample projects for each component (ThreadX, FileX, GUIX, NetX Duo, and USBX) can be created using Renesas e² studio or IAR Embedded Workbench (EW) with the on-board emulator. All samples are designed to run on RX family.

This document guides how to create and use these sample projects.

Supported Sample Projects

- **ThreadX sample project**
  Contains ThreadX source code
- **FileX RAM Disk sample project**
  Contains FileX source code
- **NetX Duo Ping sample project**
  Contains NetX Duo ping sample project
- **NetX Duo Iperf sample project**
  Contains NetX Duo iPerf sample project
- **IoT Embedded SDK sample project**
  Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS
- **IoT Embedded SDK PnP sample project**
  Sample project to connect to Azure IoT Hub using Azure IoT Middleware for Azure RTOS via IoT Plug and Play
- **IoT Embedded SDK with IoT Plug and Play sample project**
  Sample project with IoT Plug and Play using multiple components
- **GUIX 8bpp sample project**
  Contains sample for GUIX 8BPP
- **GUIX 16bpp sample project**
  Contains sample for GUIX 16BPP

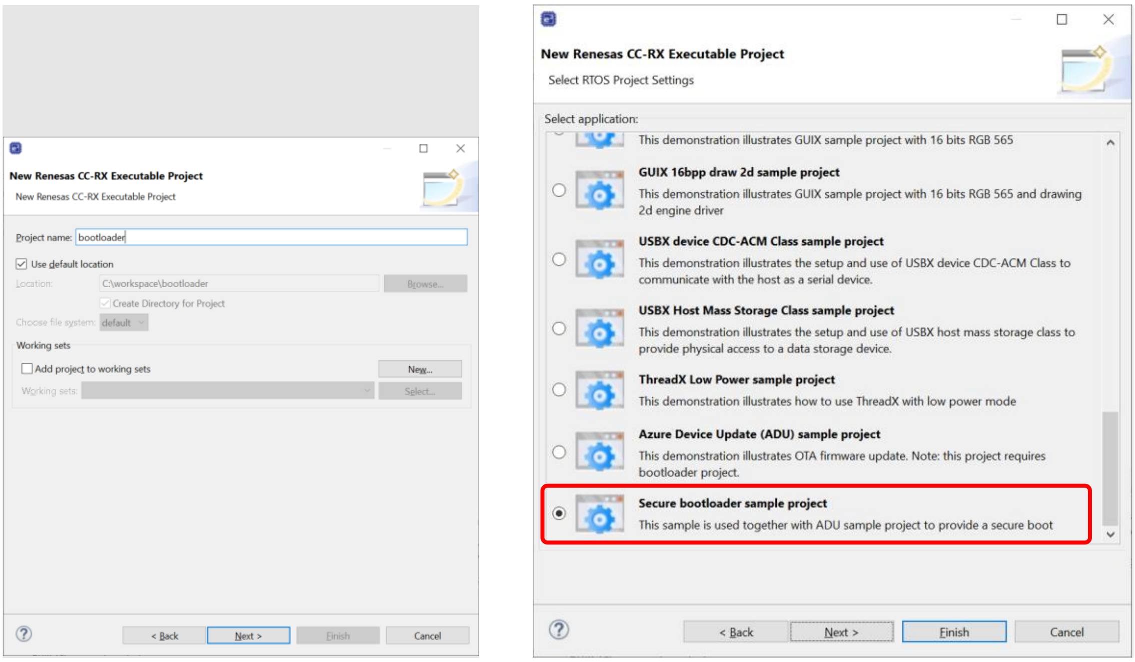
Smart Configurator Output

```
M00000004: File removed:src\smc_gen\r_flash_rx\src\targets\rx72t\r_flash_rx72t.h
M03000004: File modified:src\smc_gen\r_config\r_bsp_config.h
M00000002: Code generation is successful:C:\workspace\ADU_sample_rdp139\src\smc_gen
```

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.



New Renesas CC-RX Executable Project

New Renesas CC-RX Executable Project

Project name: bootloader

☑ Use default location

Location: C:\workspace\bootloader

☑ Create Directory for Project

Choose file system: default

Working sets

□ Add project to working sets

Working sets:

New Renesas CC-RX Executable Project

Select RTOS Project Settings

Select application:

This demonstration illustrates GUIX sample project with 16 bits RGB 565

○ **GUIX 16bpp draw 2d sample project**
This demonstration illustrates GUIX sample project with 16 bits RGB 565 and drawing 2d engine driver

○ **USBX device CDC-ACM Class sample project**
This demonstration illustrates the setup and use of USBX device CDC-ACM Class to communicate with the host as a serial device.

○ **USBX Host Mass Storage Class sample project**
This demonstration illustrates the setup and use of USBX host mass storage class to provide physical access to a data storage device.

○ **ThreadX Low Power sample project**
This demonstration illustrates how to use ThreadX with low power mode

○ **Azure Device Update (ADU) sample project**
This demonstration illustrates OTA firmware update. Note: this project requires bootloader project.

● **Secure bootloader sample project**
This sample is used together with ADU sample project to provide a secure boot

Figure 2.13 Settings for Creating Bootloader Project

2.3 Changing Project Settings

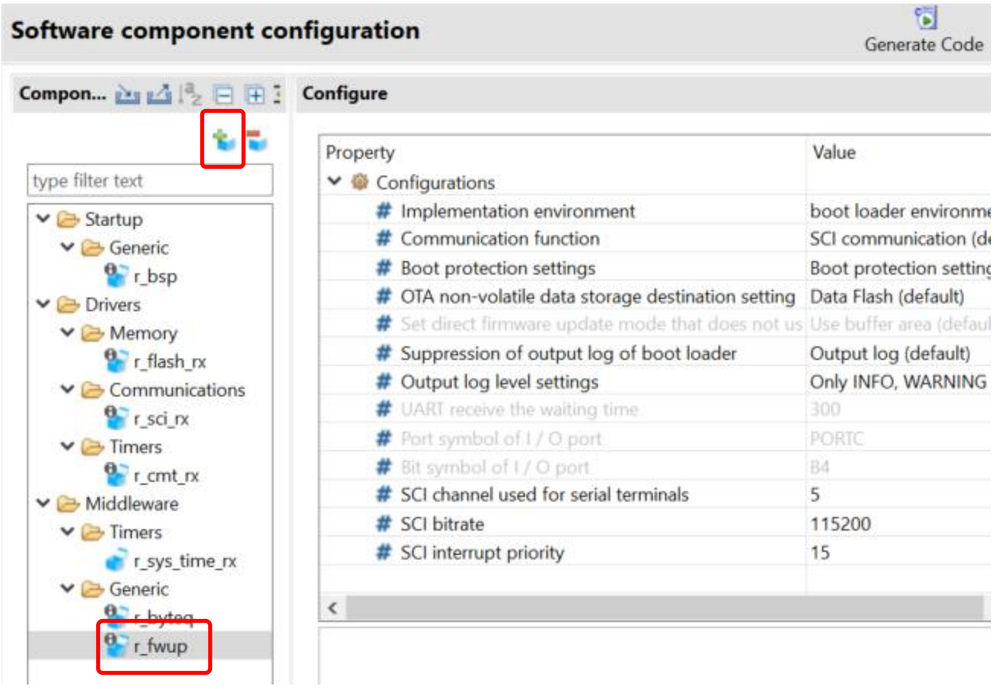
It is necessary to change the settings of the newly created projects in order to implement ADU. Note that the steps described in this section must be performed on both the ADU sample project and the bootloader sample project. The description of the steps below mainly uses the bootloader sample project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

2.3.1 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.

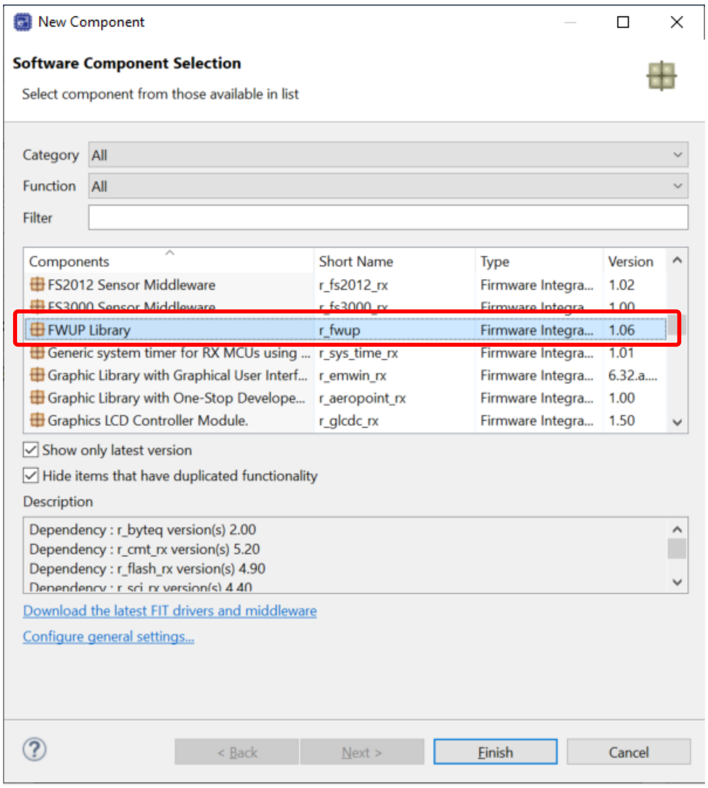


Property	Value
Configurations	
# Implementation environment	boot loader environme
# Communication function	SCI communication (d
# Boot protection settings	Boot protection settin
# OTA non-volatile data storage destination setting	Data Flash (default)
# Set direct firmware update mode that does not us	Use buffer area (defaul
# Suppression of output log of boot loader	Output log (default)
# Output log level settings	Only INFO, WARNING
# UART receive the waiting time	300
# Port symbol of I / O port	PORTC
# Bit symbol of I / O port	B4
# SCI channel used for serial terminals	5
# SCI bitrate	115200
# SCI interrupt priority	15

Figure 2.14 Software component configuration Window

Here, icons displayed with a gray overlay indicate components that have not been downloaded to e² studio. Follow the steps below to download these components.

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.)



New Component

Software Component Selection

Select component from those available in list

Category: All

Function: All

Filter:

Components	Short Name	Type	Version
FS2012 Sensor Middleware	r_fs2012_rx	Firmware Integra...	1.02
FS3000 Sensor Middleware	r_fs3000_rx	Firmware Integra...	1.00
FWUP Library	r_fwup	Firmware Integra...	1.06
Generic system timer for RX MCUs using ...	r_sys_time_rx	Firmware Integra...	1.01
Graphic Library with Graphical User Interf...	r_emwin_rx	Firmware Integra...	6.32.a....
Graphic Library with One-Stop Develope...	r_aeropoint_rx	Firmware Integra...	1.00
Graphics LCD Controller Module.	r_glcdc_rx	Firmware Integra...	1.50

☑ Show only latest version

☑ Hide items that have duplicated functionality

Description

Dependency : r_byteq version(s) 2.00
Dependency : r_cmt_rx version(s) 5.20
Dependency : r_flash_rx version(s) 4.90
Dependency : r_sci_rx version(s) 4.40

Download the latest FIT drivers and middleware

Configure general settings...

Figure 2.15 Software Component Selection Window

2. Configure initial settings for the firmware update module. Click **r_fwup** in the component tree, and check the settings listed below. Note that the settings differ for the bootloader sample and the ADU sample.
   - bootloader (bootloader)
     - SCI channel used for serial terminals: 5*1
   - ADU (adu_sample)
     - Implementation environment: Azure ADU
     - SCI channel used for serial terminals: 5*1

Note: 1. The sample projects have a mode that allows confirmation of their operating status using a terminal emulator program. The serial port (SCI) is used for output to the terminal emulator program. In the example above the setting is configured for SCI5 on the CK-RX65N. You should configure the setting for the channel that matches the specifications of the target board you are using.
On v1.04 and previous of the firmware update module it is not possible to select **Azure ADU** for **Implementation environment**. If **Azure ADU** is not listed, make sure to select v1.06 or later of the firmware update module.

3. 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.

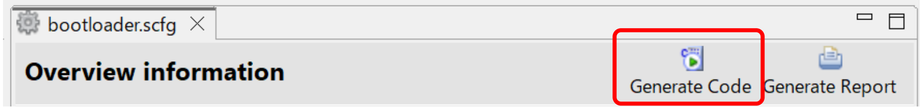


Figure 2.16 Generate Code Button

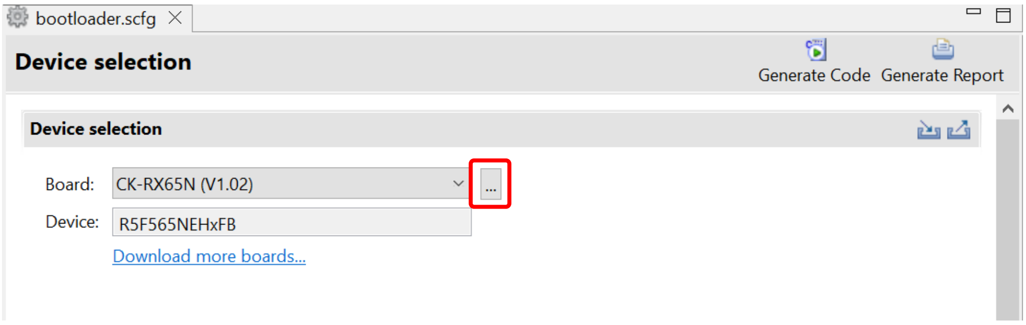
If there are other components with a gray overlay on their icons, repeat the above steps for each of them.

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

2.3.2 Changing the Device to Dual Mode

In order to implement ADU, the device must be configured for dual mode, in which the code flash of the device is treated as two banks. Perform the steps below to change this setting.

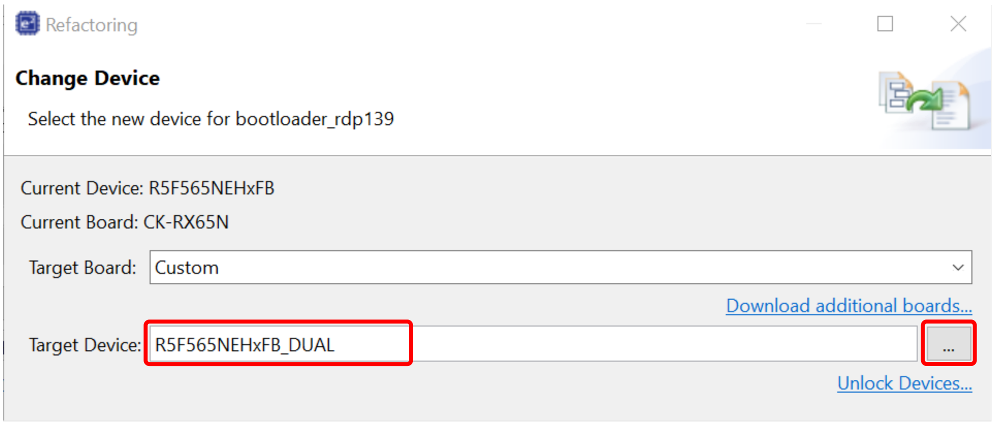
1. Change the device.
   In Project Explorer, expand the **bootloader** project tree and double-click **bootloader.scfg** to open the Smart Configurator perspective for the **bootloader** project. In the Smart Configurator window, select the **Board** tab to open the **Device selection** window. Click the **...** button next to **Board:** to open the **Change Device** window.



Board: CK-RX65N (V1.02)
Device: R5F565NEHxFB

Figure 2.17 Device selection Window

In the **Change Device** window, change the value of the **Target Device:** item to **R5F565NEHxFB_DUAL**. You can click the **...** button to the right of the text entry field to choose from a list of candidates. Leave the **Target Board:** setting of **Custom** unchanged.
After changing the device, click the **Next >** button. On the information window that appears, click the **Next >** button again without making changes.

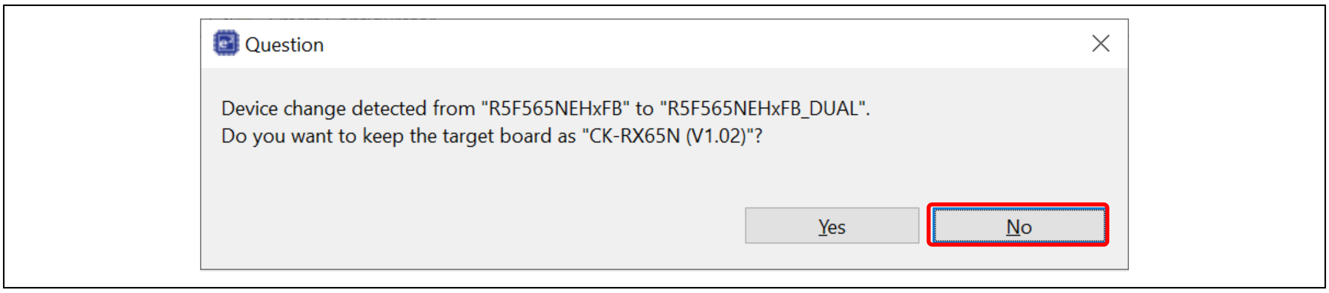


Change Device
Select the new device for bootloader_rdp139

Current Device: R5F565NEHxFB
Current Board: CK-RX65N
Target Board: Custom
Target Device: R5F565NEHxFB_DUAL

Figure 2.18 Change Device Window

Finally, a window appears asking you to confirm the change. Click the **Finish** button. This completes the process of changing the device. If a window like that shown below asking you to confirm a change of target board appears, click the **No** button.



Device change detected from "R5F565NEHxFB" to "R5F565NEHxFB_DUAL".
Do you want to keep the target board as "CK-RX65N (V1.02)"?

Figure 2.19 Target Board Change Confirmation Window

After making the change, check the **Device selection** window to confirm that **Board:** is set to **Custom User Board** and **Device:** is set to **R5F565NEHxFB_DUAL**.
After changing settings in Smart Configurator, click the **Generate Code** button to generate code reflecting the changed settings.

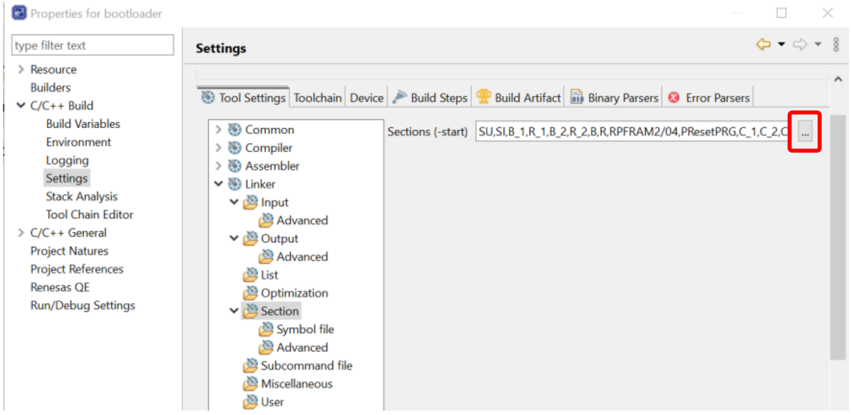
2. Configure section settings.
After changing the device to dual mode, refer to the next section and make the necessary setting changes to allocate the memory for dual mode.*1

Note: 1. After the device is changed, the section settings are cleared to their default values.

2.3.3 Section Information Settings

The section information is initialized to the default values after the device is changed to dual mode, so it is necessary to reconfigure the settings. Follow the steps below to configure the section information settings.

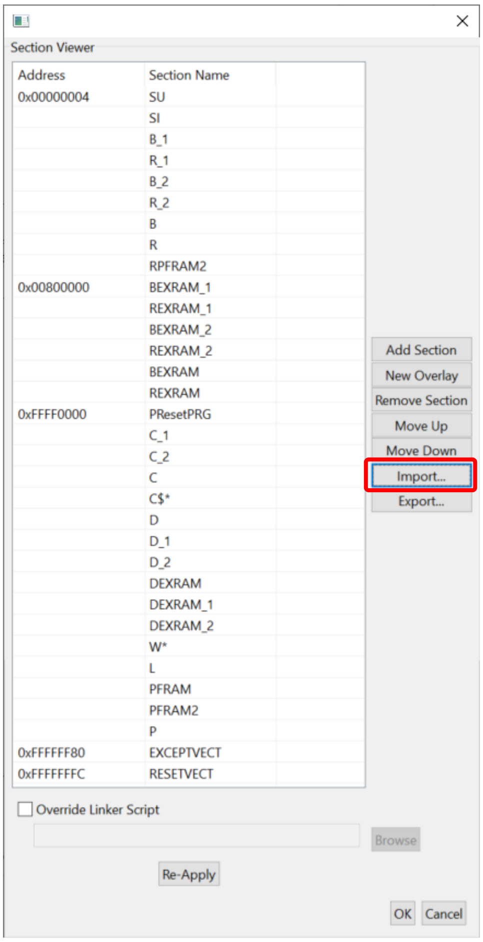
1. CC-RX:
   In Project Explorer, right-click the **bootloader** project and select **Properties** → **C/C++ Build** → **Settings** → **Tool Settings** tab → **Linker** → **Section**, and click the **…** button.



Sections (-start) SU,SI,B_1,R_1,B_2,R_2,B,R,RPFRAM2/04,PResetPRG,C_1,C_2,C

Figure 2.20 Launching Section Viewer

Section data for the sample is contained in the **src** folder. Click the **Import…** button and import the section data file named **linker_section_sample.esi**. Note that importing the file overwrites the section information.

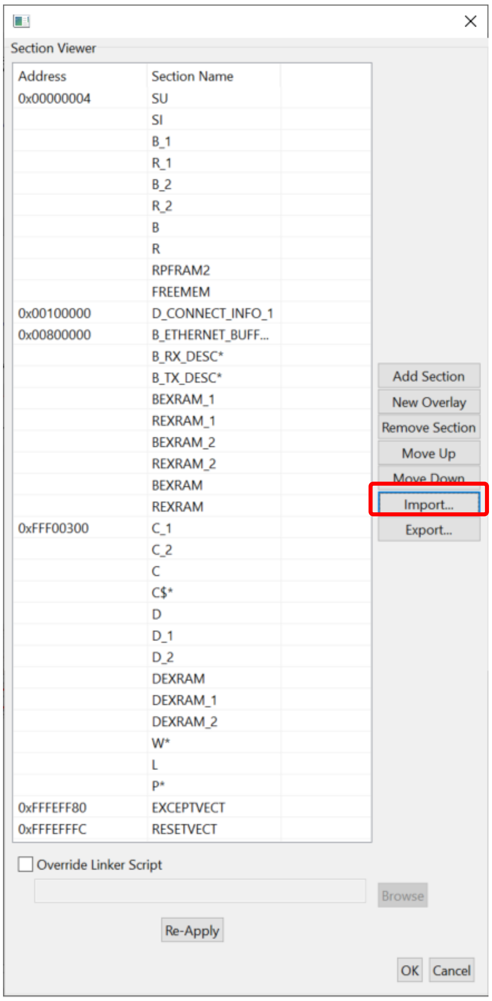


Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
	RPFRAM2
0x00800000	BEXRAM_1
	REXRAM_1
	BEXRAM_2
	REXRAM_2
	BEXRAM
	REXRAM
0xFFFF0000	PResetPRG
	C_1
	C_2
	C
	C$*
	D
	D_1
	D_2
	DEXRAM
	DEXRAM_1
	DEXRAM_2
	W*
	L
	PFRAM
	PFRAM2
	P
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

☐ Override Linker Script

Figure 2.21 Bootloader Sample (CC-RX) Section Settings

The section data of the bootloader sample and ADU sample are different, so when configuring the section data for the ADU sample, make sure to import the file in the **src** folder of the ADU sample project named **linker_section_sample.esi**.

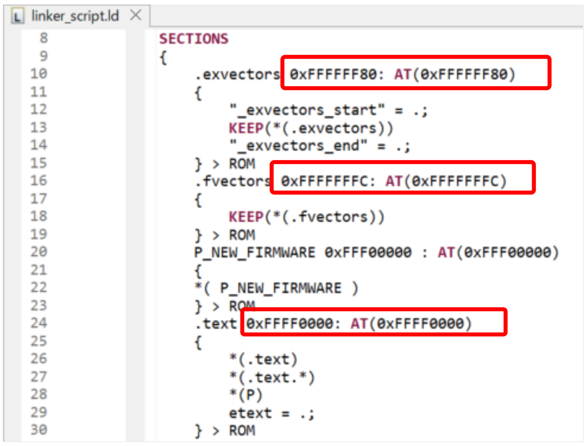


Address	Section Name
0x00000004	SU
	SI
	B_1
	R_1
	B_2
	R_2
	B
	R
	RPFRAM2
	FREEMEM
0x00100000	D_CONNECT_INFO_1
0x00800000	B_ETHERNET_BUFF...
	B_RX_DESC*
	B_TX_DESC*
	BEXRAM_1
	REXRAM_1
	BEXRAM_2
	REXRAM_2
	BEXRAM
	REXRAM
0xFFF00300	C_1
	C_2
	C
	C$*
	D
	D_1
	D_2
	DEXRAM
	DEXRAM_1
	DEXRAM_2
	W*
	L
	P*
0xFFFFFF80	EXCEPTVECT
0xFFFFFFFC	RESETVECT

□ Override Linker Script

Figure 2.22 ADU Sample (CC-RX) Section Settings

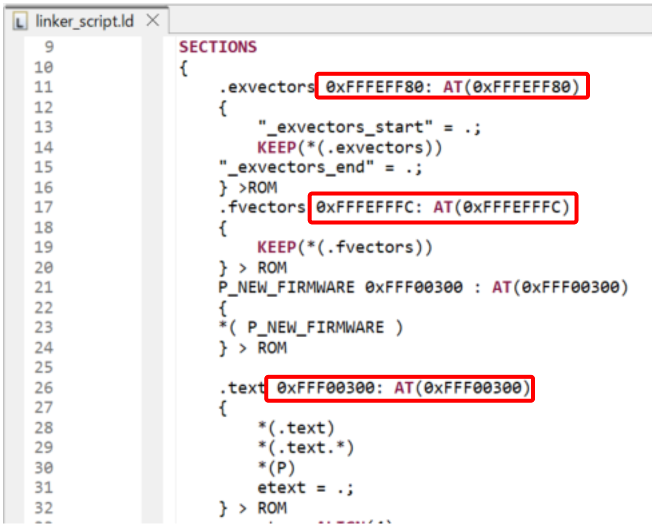
2. GCC
   Please rename \src\linker_script_sample.ld of the adu_sample project to \src\linker_script.ld and overwrite it. Open **\src\linker_script.ld** in the **bootloader** project, click the **linker_script.ld** tab, and confirm that **.text** is set to **0xFFFF0000**, **.exvectors** to **FFFFFF80**, and **.fvectors** to **FFFFFFFC** as shown in the figure below. Also confirm that the **AT()** values in brackets match the above.



```
SECTIONS
{
    .exvectors 0xFFFFFF80: AT(0xFFFFFF80)
    {
        "_exvectors_start" = .;
        KEEP(*(.exvectors))
        "_exvectors_end" = .;
    } > ROM
    .fvectors 0xFFFFFFFC: AT(0xFFFFFFFC)
    {
        KEEP(*(.fvectors))
    } > ROM
    P_NEW_FIRMWARE 0xFFF00000 : AT(0xFFF00000)
    {
    *( P_NEW_FIRMWARE )
    } > ROM
    .text 0xFFFF0000: AT(0xFFFF0000)
    {
        *(.text)
        *(.text.*)
        *(P)
        etext = .;
    } > ROM
```

Figure 2.23 Bootloader Sample (GCC) Section Settings

The section data of the bootloader sample and ADU sample are different.
Please rename \src\linker_script_sample.ld of the adu_sample project to \src\linker_script.ld and overwrite it. Open **\src\linker_script.ld** in the **adu_sample** project, click the **linker_script.ld** tab, and confirm that **.text** is set to **0xFFF00300**, **.exvectors** to **FFFEFF80**, and **.fvectors** to **FFFEFFFC** as shown in the figure below. Also confirm that the **AT()** values in brackets match the above.



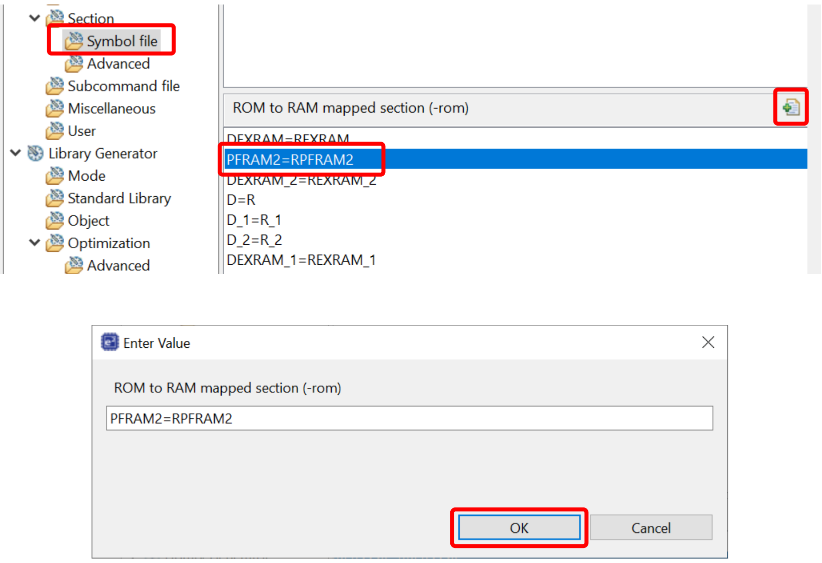
```
SECTIONS
{
    .exvectors 0xFFFEFF80: AT(0xFFFEFF80)
    {
        "_exvectors_start" = .;
        KEEP(*(.exvectors))
    "_exvectors_end" = .;
    } >ROM
    .fvectors 0xFFFEFFFC: AT(0xFFFEFFFC)
    {
        KEEP(*(.fvectors))
    } > ROM
    P_NEW_FIRMWARE 0xFFF00300 : AT(0xFFF00300)
    {
    *( P_NEW_FIRMWARE )
    } > ROM

    .text 0xFFF00300: AT(0xFFF00300)
    {
        *(.text)
        *(.text.*)
        *(P)
        etext = .;
    } > ROM
```

Figure 2.24 ADU Sample (GCC) Section Settings

2.3.4 Adding a Section Mapped from ROM to RAM

Add a definition for a section mapped from ROM to RAM. In Project Explorer, right-click the **bootloader** project and select **Properties** → **C/C++ Build** → **Settings** → **Tool Settings** tab → **Linker** → **Section** → **Symbol file**, and click the **+** button to the right of **ROM to RAM mapped section (-rom)**. Enter the value **FRAM2=RPFRAM2** and click the **OK** button.

This setting applies to the CC-RX only. It is not required on the GCC.



ROM to RAM mapped section (-rom)

```
DEXRAM=REXRAM
PFRAM2=RPFRAM2
DEXRAM_2=REXRAM_2
D=R
D_1=R_1
D_2=R_2
DEXRAM_1=REXRAM_1
```

Enter Value

ROM to RAM mapped section (-rom)

PFRAM2=RPFRAM2

Figure 2.25 Adding a Section Mapped from ROM to RAM

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 completes, open **System Properties** → **Environment Variables** in Windows and add the OpenSSL install folder to the **Path** environment variable.

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

2.4.2 Generating a Key Pair for ECC in OpenSSL

The MOT file conversion tool, which you will need to use later, requires you to specify an ECC public key and private key. 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
Signature ok
subject=/C=JP/ST=Tokyo/L=Kodaira/O=Renesas Electronics/OU=Software Development
Division/CN= Renesas Tarou/emailAddress= Tarou.Renesas@sample.com
Getting CA Private Key
```

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
```

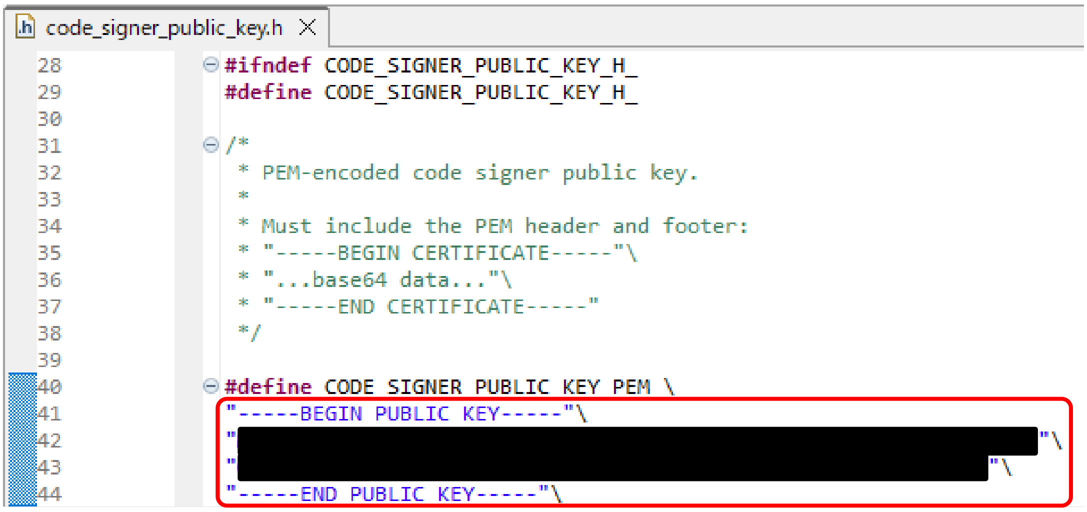
F. Extract a Public Key for Elliptic Curve Cryptography (Parameter: secp256r1)

```
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 (\).



```
#ifndef CODE_SIGNER_PUBLIC_KEY_H_
#define CODE_SIGNER_PUBLIC_KEY_H_

/*
 * PEM-encoded code signer public key.
 *
 * Must include the PEM header and footer:
 * "-----BEGIN CERTIFICATE-----"\
 * "...base64 data..."\
 * "-----END CERTIFICATE-----"
 */

#define CODE_SIGNER_PUBLIC_KEY_PEM \
"-----BEGIN PUBLIC KEY-----"\
"[redacted]"\
"-----END PUBLIC KEY-----"\
```

Figure 2.26 Public Key Information Setting

2.5 Building the bootloader Project

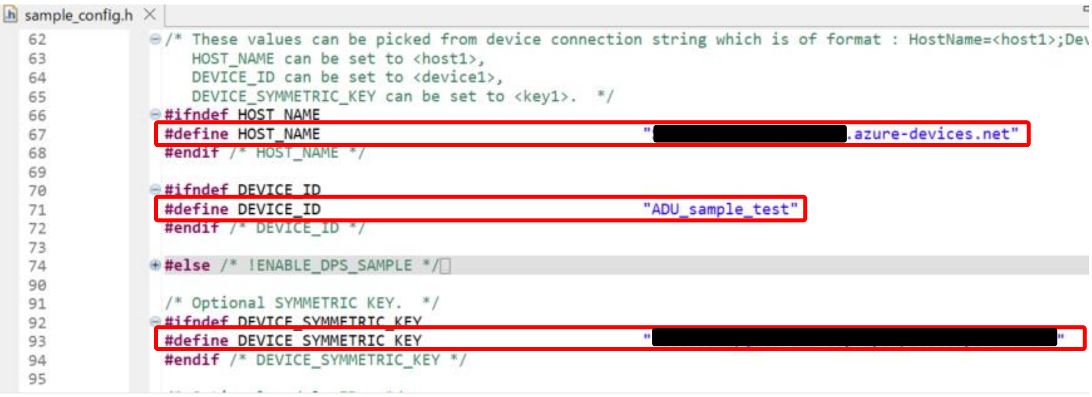
Build the **bootloader** project and create a **bootloader.mot** file.

The MOT file is created in the following folder.

\bootloader\HardwareDebug/

2.6 Connection Information Macro Settings

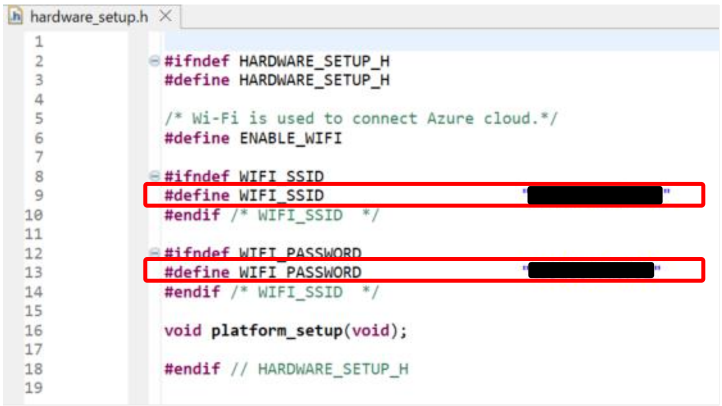
Open **\src\sample_config.h** from the **adu_sample** project and specify setting values for **HOST_NAME**, **DEVICE_ID**, and **DEVICE_SYMMETRIC_KEY**. For the setting values, refer to the parameters configured on the Azure portal as described in 3.1, IoT Hub and Device Registration.



```
/* These values can be picked from device connection string which is of format : HostName=<host1>;De\
   HOST_NAME can be set to <host1>,
   DEVICE_ID can be set to <device1>,
   DEVICE_SYMMETRIC_KEY can be set to <key1>.  */
#ifndef HOST_NAME
#define HOST_NAME                                   "[redacted].azure-devices.net"
#endif /* HOST_NAME */

#ifndef DEVICE_ID
#define DEVICE_ID                                   "ADU_sample_test"
#endif /* DEVICE_ID */

#else /* !ENABLE_DPS_SAMPLE */

/* Optional SYMMETRIC KEY.  */
#ifndef DEVICE_SYMMETRIC_KEY
#define DEVICE_SYMMETRIC_KEY                        "[redacted]"
#endif /* DEVICE_SYMMETRIC_KEY */
```

Figure 2.27 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.

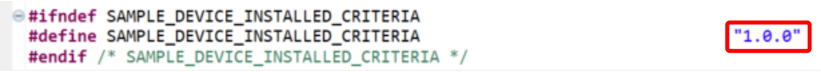


```

#ifndef HARDWARE_SETUP_H
#define HARDWARE_SETUP_H

/* Wi-Fi is used to connect Azure cloud.*/
#define ENABLE_WIFI

#ifndef WIFI_SSID
#define WIFI_SSID                      "[redacted]"
#endif /* WIFI_SSID  */

#ifndef WIFI_PASSWORD
#define WIFI_PASSWORD                  "[redacted]"
#endif /* WIFI_SSID  */

void platform_setup(void);

#endif // HARDWARE_SETUP_H

```

Figure 2.28 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**.



```
#ifndef SAMPLE_DEVICE_INSTALLED_CRITERIA
#define SAMPLE_DEVICE_INSTALLED_CRITERIA                                     "1.0.0"
#endif /* SAMPLE_DEVICE_INSTALLED_CRITERIA */
```

Figure 2.29 Initial Firmware Version

2.8 Building the adu_sample Project

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

The MOT file is created in the following folder.

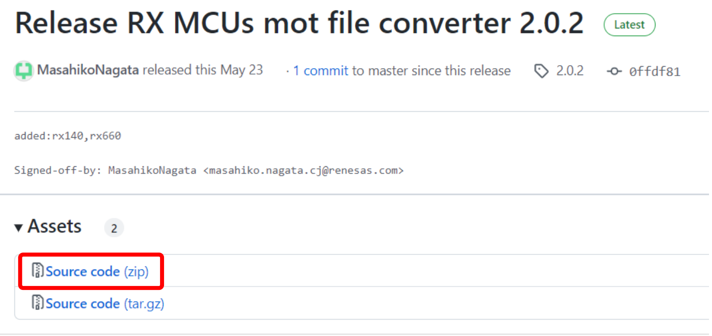
\adu_sample\HardwareDebug\

2.9 Creating the Initial Firmware

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

Create the initial firmware by combining the bootloader and firmware.

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.



Release RX MCUs mot file converter 2.0.2 Latest

MasahikoNagata released this May 23 · 1 commit to master since this release 2.0.2 0ffdf81

```
added:rx140,rx660

Signed-off-by: MasahikoNagata <masahiko.nagata.cj@renesas.com>
```

▾ Assets 2

Source code (zip)

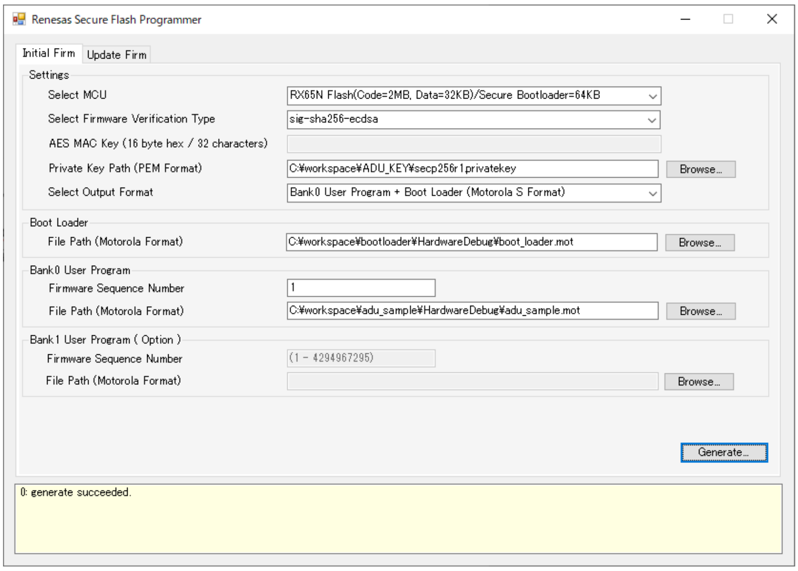
Source code (tar.gz)

Figure 2.30 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.

Click the **Initial Firm** tab and enter the following settings.

- 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



Renesas Secure Flash Programmer

Initial Firm | Update Firm

Settings

Select MCU: RX65N Flash(Code=2MB, Data=32KB)/Secure Bootloader=64KB

Select Firmware Verification Type: sig-sha256-ecdsa

AES MAC Key (16 byte hex / 32 characters):

Private Key Path (PEM Format): C:¥workspace¥ADU_KEY¥secp256r1privatekey

Select Output Format: Bank0 User Program + Boot Loader (Motorola S Format)

Boot Loader

File Path (Motorola Format): C:¥workspace¥bootloader¥HardwareDebug¥boot_loader.mot

Bank0 User Program

Firmware Sequence Number: 1

File Path (Motorola Format): C:¥workspace¥adu_sample¥HardwareDebug¥adu_sample.mot

Bank1 User Program (Option)

Firmware Sequence Number: (1 - 4294967295)

File Path (Motorola Format):

```
0: generate succeeded.
```

Figure 2.31 Initial Firmware Creation Window

Finally, click the **Generate...** button and save the file **userprog.mot** to a folder of your choice. The process is complete when the message “generate succeeded.” appears at the bottom of the window.

The structure of the generated **userprog.mot** file is shown below. The verification data, initial firmware, and bootloader are contained in a single MOT file. The verification data includes information such as keys used to verify the firmware.

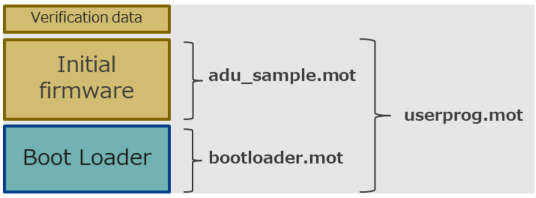


Figure 2.32 Structure of userprog.mot File

2.10 Installing the Flash Programming Tool

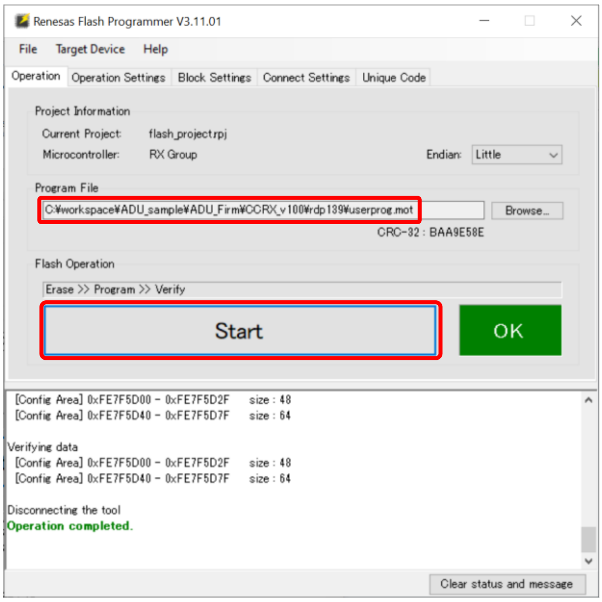
Access the Renesas Flash Programmer download site, download the Renesas Flash Programmer V3.11.02 Windows installer, and run it to install the tool.

2.11 Launching Initial Firmware

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

The initial firmware is programmed to the RX65N in its initial state, which is linear mode. 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.



Project Information
Current Project: flash_project.rpj
Microcontroller: RX Group
Endian: Little

Program File
C:¥workspace¥ADU_sample¥ADU_Firm¥CCRX_v100¥rdp139¥userprog.mot
CRC-32 : BAA9E58E

Flash Operation
Erase >> Program >> Verify

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

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

Disconnecting the tool
Operation completed.
```

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.

Renesas Flash Programmer recognizes linear mode and dual mode as separate MCUs. Since A) is recognized as linear mode and B) is recognized as dual mode, when A) or B) is executed consecutively, “**Error (E3000107): This device does not match the connection parameters.**” will occur.

After programming of the initial firmware finishes, the program runs on the RX65N. The execution results can be confirmed by using terminal software. 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. It also calculates a hash value for the firmware overall and confirms that it matches the decrypted hash value. If the values match, it launches the firmware.

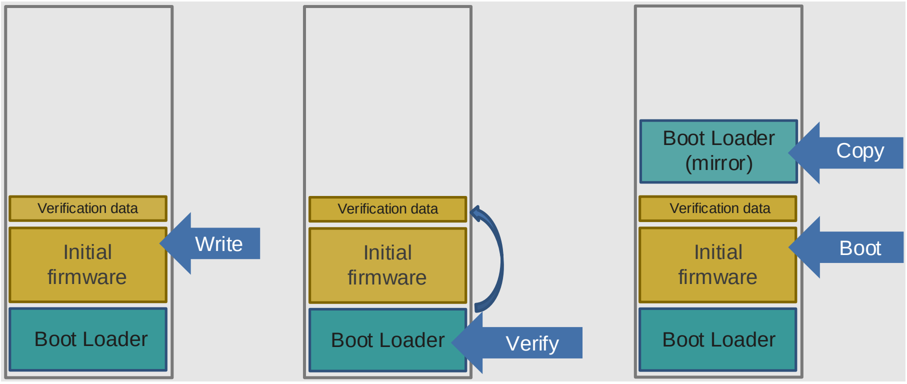


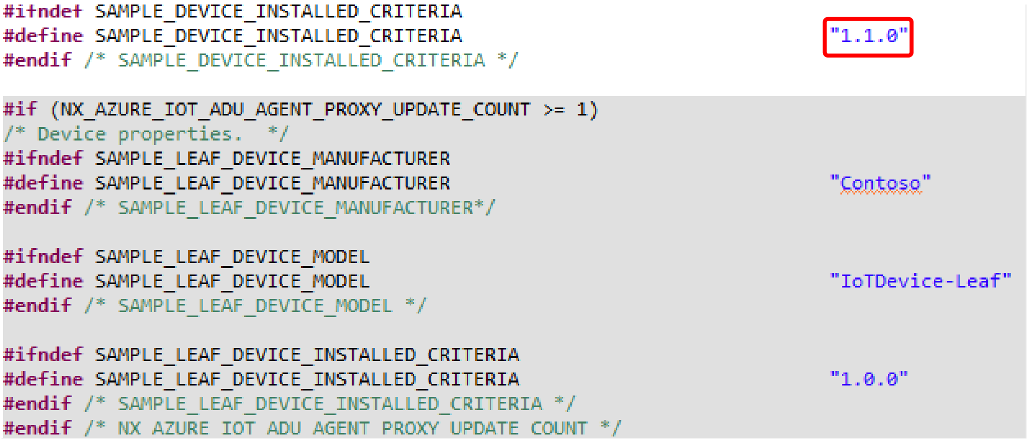
Figure 2.34 Bootloader Operation

2.12 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 2.35 Updated Firmware Setting

2.13 Building the Updated Firmware

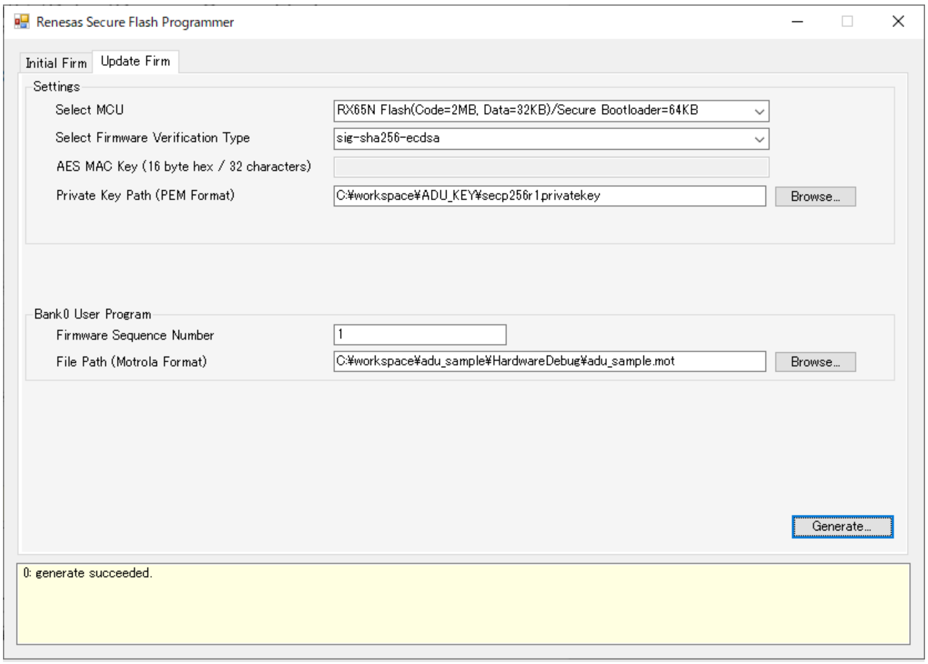
Build the **adu_sample** project and create an **adu_sample.mot** file.

2.14 Creating the Updated Firmware

Convert the updated firmware to RSU format.*[1]

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. Then click the **Update Firm** tab and enter the following settings.

- 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 (Motrola format): \adu_sample\HardwareDebug\adu_sample.mot



Renesas Secure Flash Programmer

Settings

Select MCU: RX65N Flash(Code=2MB, Data=32KB)/Secure Bootloader=64KB

Select Firmware Verification Type: sig-sha256-ecdsa

AES MAC Key (16 byte hex / 32 characters):

Private Key Path (PEM Format): C:¥workspace¥ADU_KEY¥secp256r1.privatekey

Bank0 User Program

Firmware Sequence Number: 1

File Path (Motrola Format): C:¥workspace¥adu_sample¥HardwareDebug¥adu_sample.mot

0: generate succeeded.

Figure 2.36 Updated Firmware Creation Window

Finally, click the **Generate...** button to save **userprog.rsu**.

The process is complete when the message “generate succeeded.” appears at the bottom of the window.

The structure of the generated **userprog.rsu** file is shown below. The binary file contains both the verification data and the updated firmware.

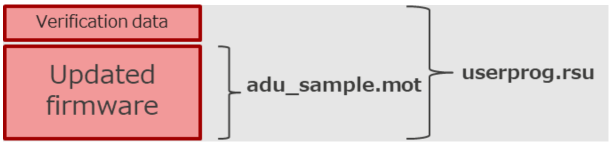


Figure 2.37 Structure of userprog.rsu File

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), an binary file format exclusive to Renesas that allows storage of verification data(Other data such as hash values) alongside the actual firmware data, is used for ADU releases. Refer to section 7.1, Download Data Format, in the application note Renesas MCU Firmware Update Design Policy for an overview of 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.

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. 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.

The following three parameters of the newly created IoT Hub are declared in the ADU sample source code.

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

3.2 Creating a Device Update Account and Instance

Follow the guide at the link below to create a Device Update account and instance on the Azure portal.

Create Device Update for IoT Hub resources

The newly created Device Update for IoT Hub account is assigned to the same resource group as the IoT Hub.*1

Also create a storage container for uploading updated firmware.

Note: 1. Be aware that a paid account is required in order to use Device Update.

3.3 Preparing the Updated Firmware

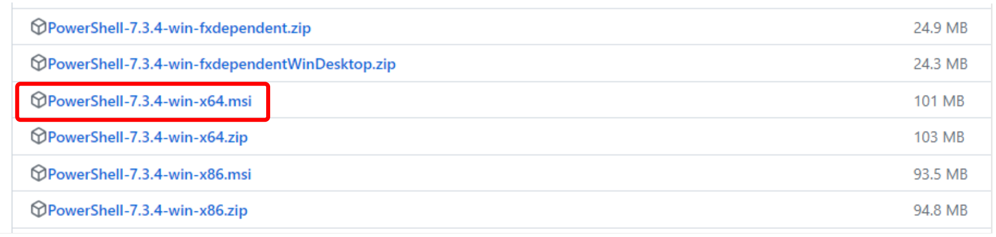
3.3.1 Building the Updated Firmware

Follow the procedure described in section 2 to create the updated firmware and generate a binary file. A binary file for use by the secure bootloader must be in RSU file format. Follow the procedure to create an RSU file.

3.3.2 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.



PowerShell-7.3.4-win-fxdependent.zip	24.9 MB
PowerShell-7.3.4-win-fxdependentWinDesktop.zip	24.3 MB
PowerShell-7.3.4-win-x64.msi	101 MB
PowerShell-7.3.4-win-x64.zip	103 MB
PowerShell-7.3.4-win-x86.msi	93.5 MB
PowerShell-7.3.4-win-x86.zip	94.8 MB

Figure 3.1 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

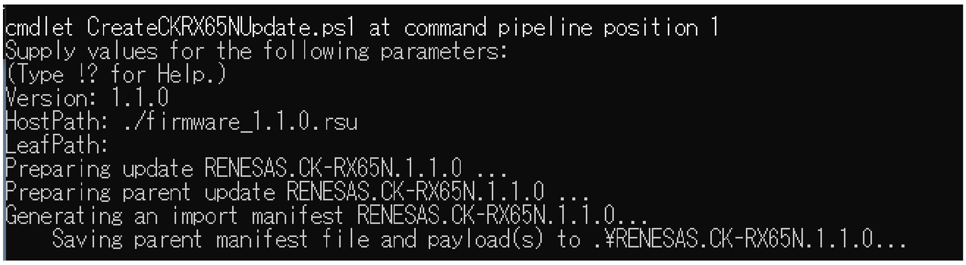
3. Run the following command in PowerShell.
   Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process

4. Copy the RSU file created as described in 2.14, Creating the Updated Firmware, to the folder referenced in step 2. Change the name of the copied RSU file to the following.
   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...
```

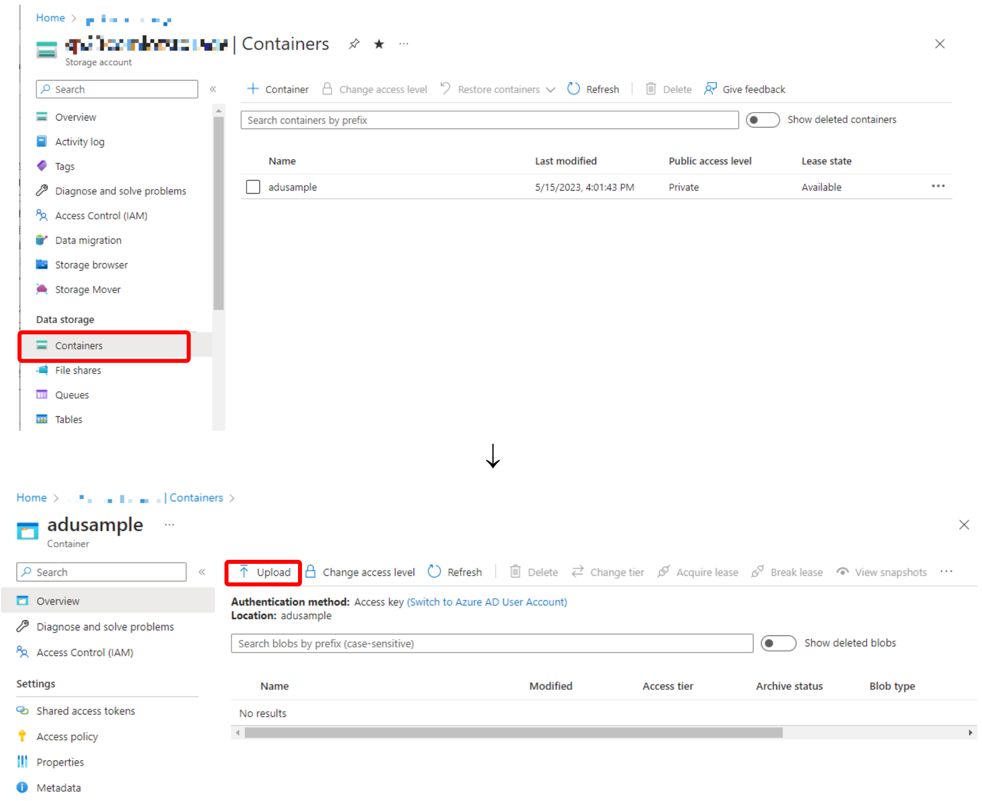
Figure 3.2 Script Run Window

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. Store these two files in the storage container.
   — RENESAS.CK-RX65N.1.1.0.importmanifest.json
   — firmware_1.1.0.rsu

3.4 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**).



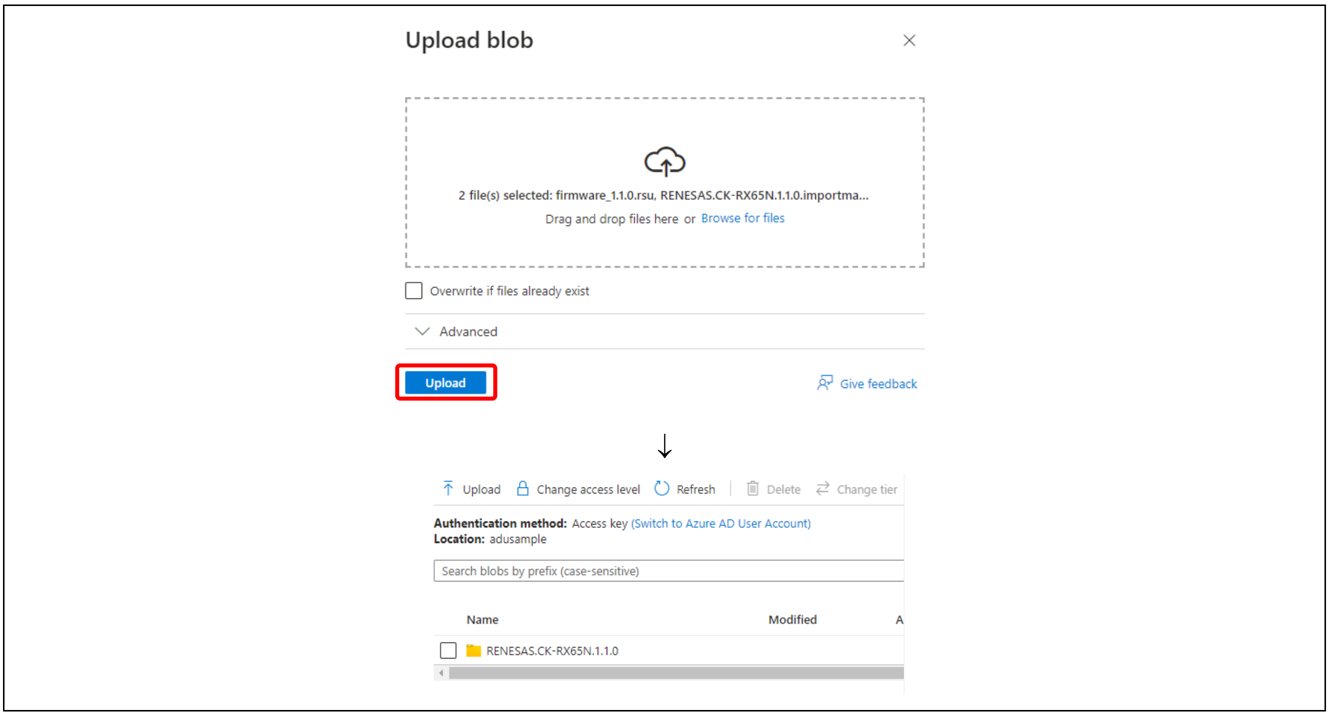
Name	Last modified	Public access level	Lease state
□ adusample	5/15/2023, 4:01:43 PM	Private	Available

Authentication method: Access key (Switch to Azure AD User Account)
Location: adusample

Name	Modified	Access tier	Archive status	Blob type
No results				

Figure 3.3 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 3.3. 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.



Upload blob

2 file(s) selected: firmware_1.1.0.rsu, RENESAS.CK-RX65N.1.1.0.importma...

Drag and drop files here or Browse for files

□ Overwrite if files already exist

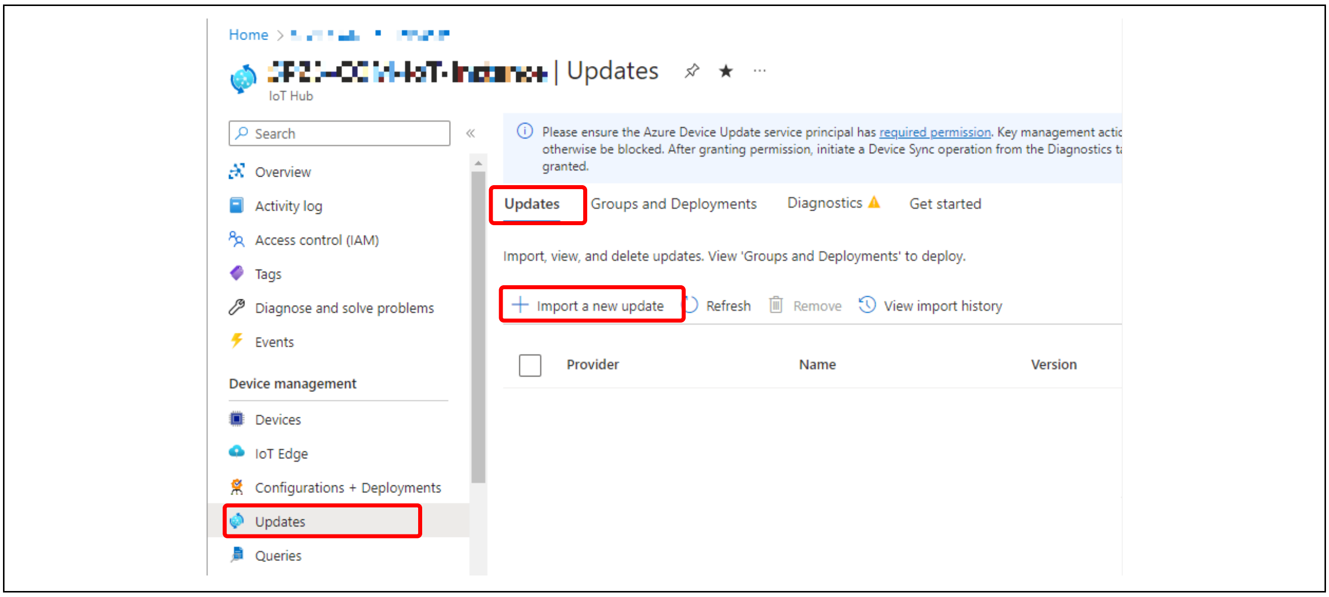
Advanced

Authentication method: Access key (Switch to Azure AD User Account)
Location: adusample

Figure 3.4 Uploading Update Files

3.5 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**.

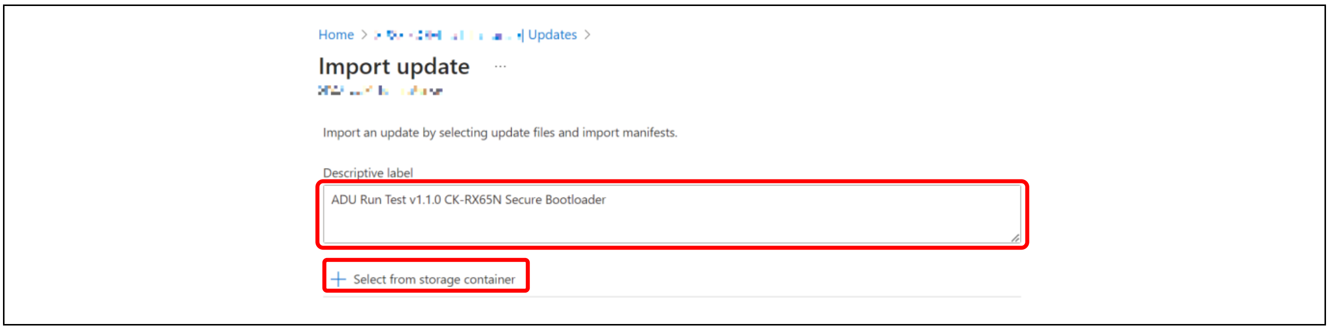


Please ensure the Azure Device Update service principal has required permission. Key management actic otherwise be blocked. After granting permission, initiate a Device Sync operation from the Diagnostics t granted.

Import, view, and delete updates. View 'Groups and Deployments' to deploy.

Figure 3.5 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**.

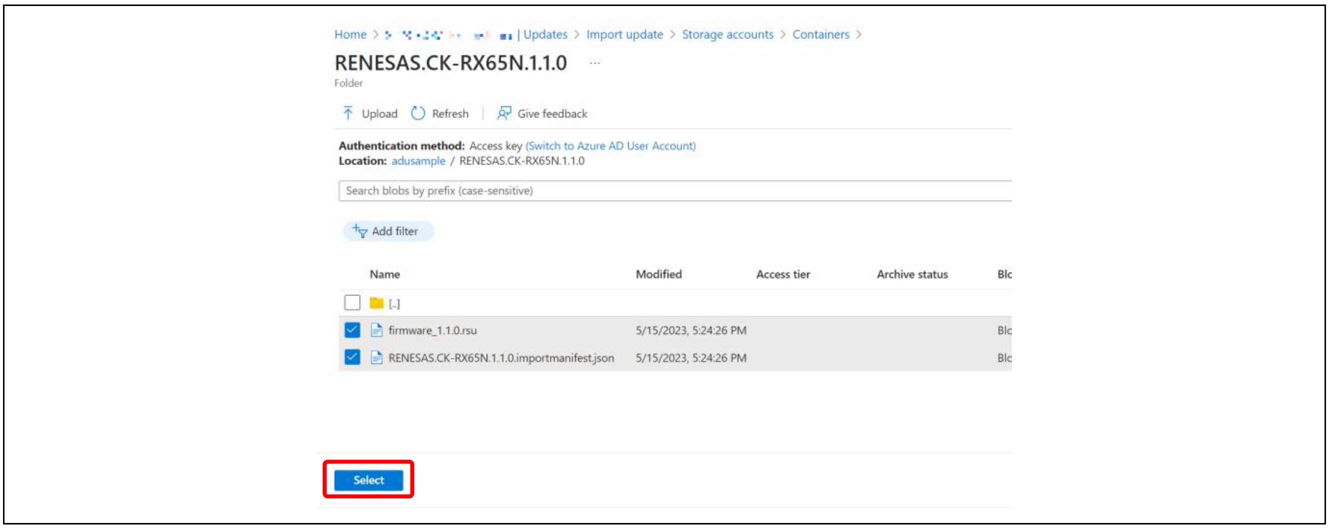


Import an update by selecting update files and import manifests.

Descriptive label: ADU Run Test v1.1.0 CK-RX65N Secure Bootloader

Figure 3.6 Entering a Description of the Update

Click <name of storage account> → <name of container> corresponding to the location to which you uploaded the firmware. 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.



Home > [redacted] | Updates > Import update > Storage accounts > Containers >

RENESAS.CK-RX65N.1.1.0 ···

Folder

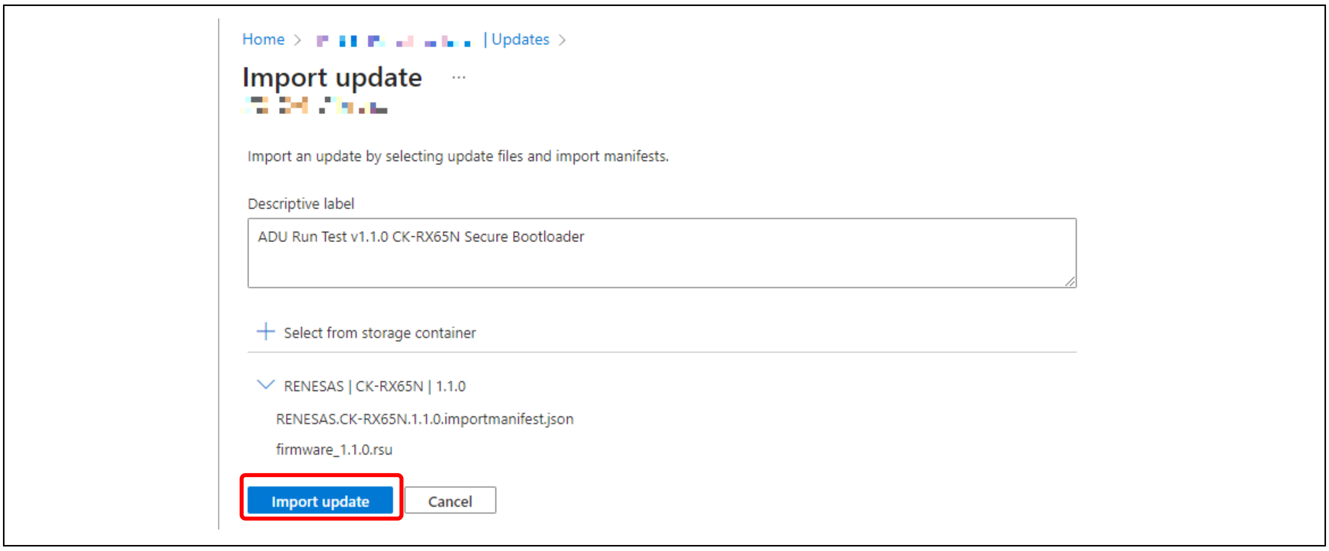
Authentication method: Access key (Switch to Azure AD User Account)
Location: adusample / RENESAS.CK-RX65N.1.1.0

Search blobs by prefix (case-sensitive)

	Name	Modified	Access tier	Archive status	Blc
□	[..]				
☑	firmware_1.1.0.rsu	5/15/2023, 5:24:26 PM			Blc
☑	RENESAS.CK-RX65N.1.1.0.importmanifest.json	5/15/2023, 5:24:26 PM			Blc

Figure 3.7 Selecting Update Files

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



Home > [redacted] | Updates >

Import update ···

[redacted]

Import an update by selecting update files and import manifests.

Descriptive label

ADU Run Test v1.1.0 CK-RX65N Secure Bootloader

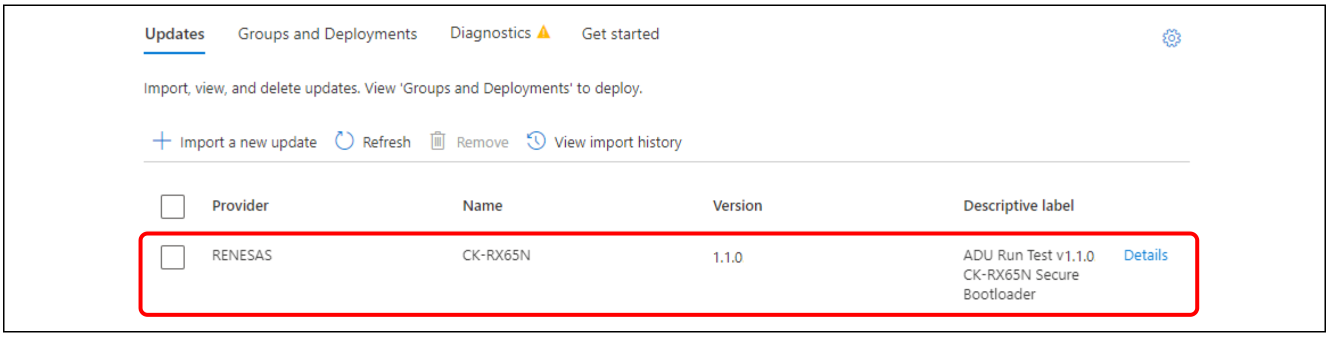
RENESAS | CK-RX65N | 1.1.0

RENESAS.CK-RX65N.1.1.0.importmanifest.json

firmware_1.1.0.rsu

Figure 3.8 Import update Page

If the import completes successfully, the imported firmware is added to the list of updates.



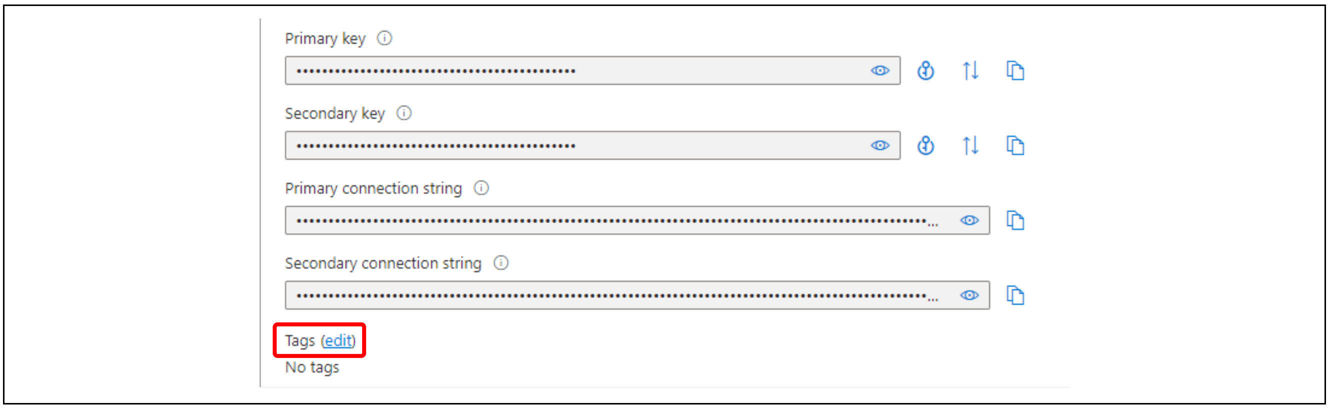
Import, view, and delete updates. View 'Groups and Deployments' to deploy.

□	Provider	Name	Version	Descriptive label	
□	RENESAS	CK-RX65N	1.1.0	ADU Run Test v1.1.0 CK-RX65N Secure Bootloader	Details

Figure 3.9 List of Updates

3.6 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.



Primary key ⓘ

Secondary key ⓘ

Primary connection string ⓘ

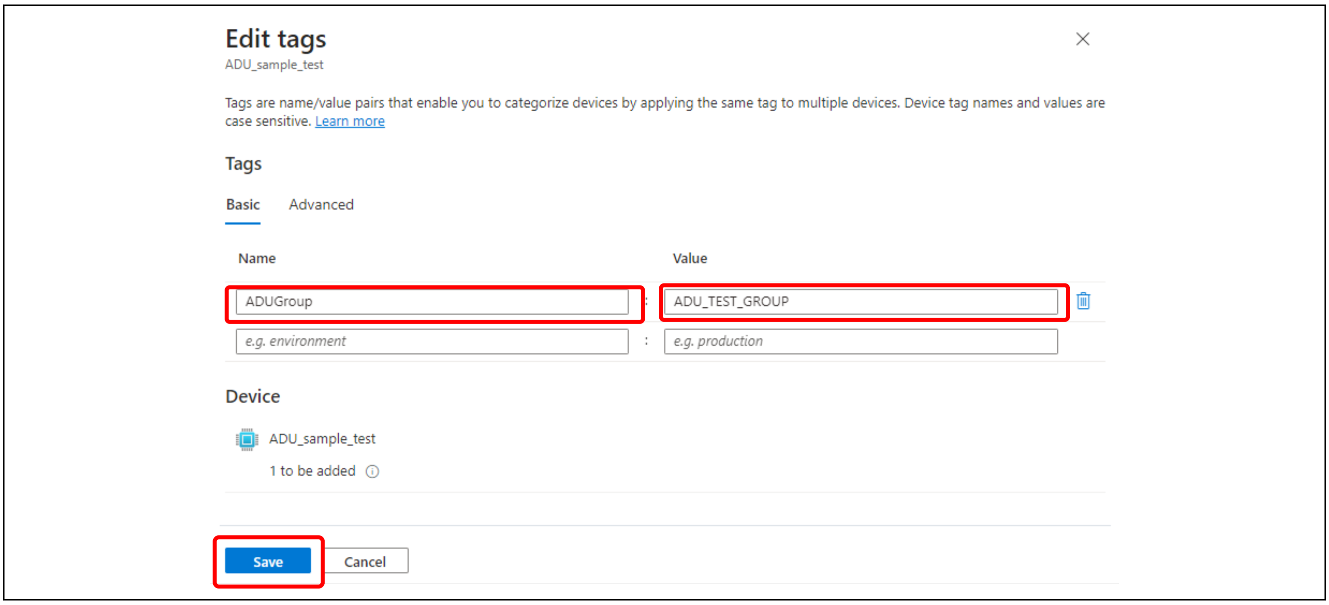
Secondary connection string ⓘ

Tags (edit)

No tags

Figure 3.10 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.



Edit tags

ADU_sample_test

Tags are name/value pairs that enable you to categorize devices by applying the same tag to multiple devices. Device tag names and values are case sensitive. Learn more

Tags

Basic Advanced

Name	Value
ADUGroup	ADU_TEST_GROUP
e.g. environment	e.g. production

Device

ADU_sample_test

1 to be added ⓘ

Save Cancel

Figure 3.11 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.

3.7 Updating the Firmware

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

3.7.1 Execution on the Target Board

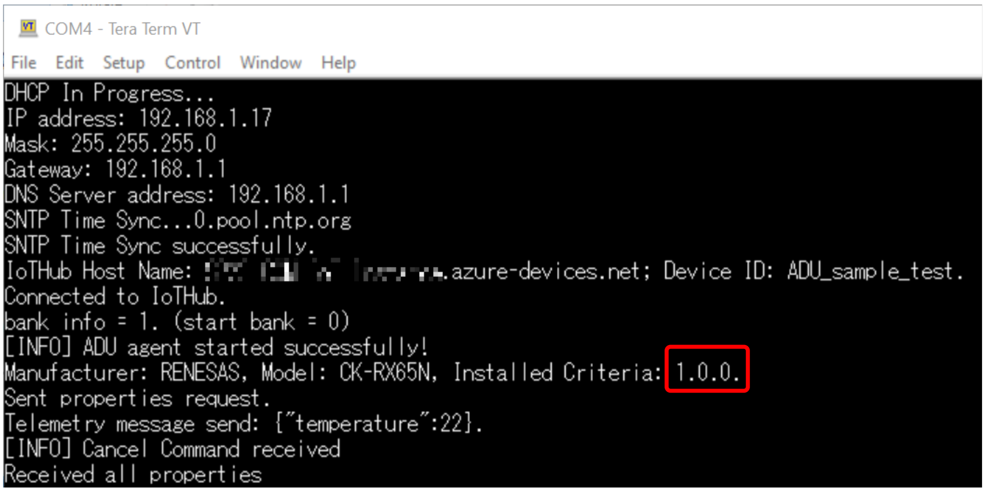
To begin, run the initial firmware previously programmed to the target board. On the CK-RX65N, configure the J16 jumper pins for “RUN” and make a connection to the J14 USB connector to start operation.

Also, by connecting a PC to the J12 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 J12 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.

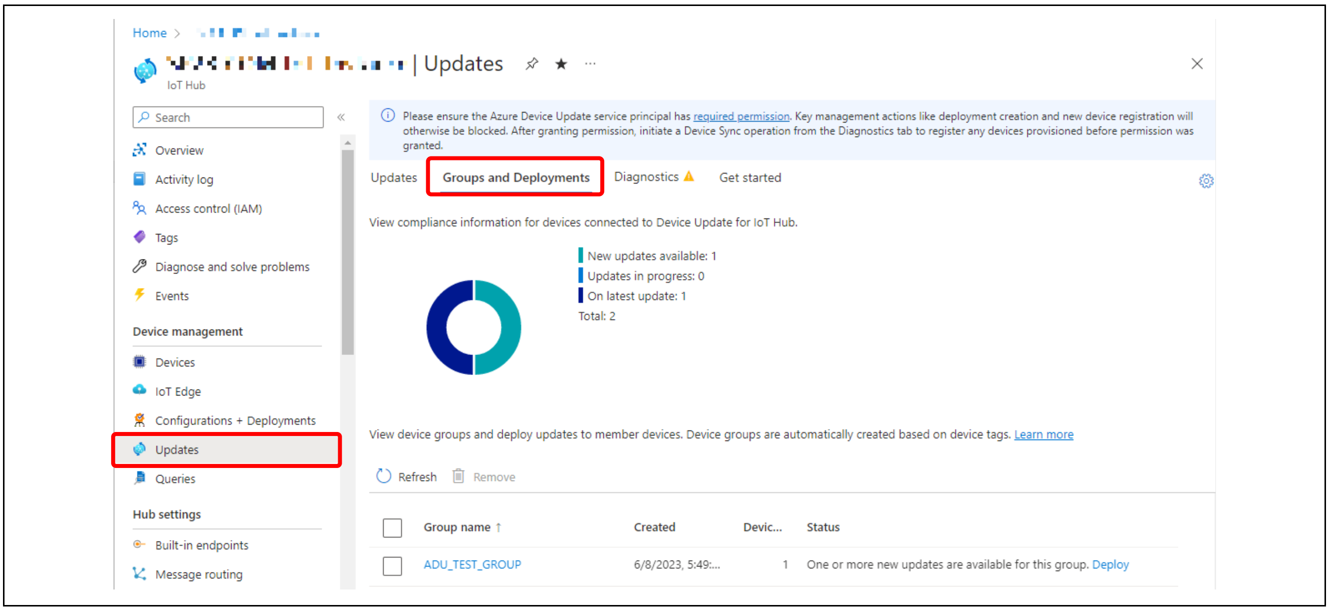


```
DHCP In Progress...
IP address: 192.168.1.17
Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS Server address: 192.168.1.1
SNTP Time Sync...0.pool.ntp.org
SNTP Time Sync successfully.
IoTHub Host Name: [redacted].azure-devices.net; Device ID: ADU_sample_test.
Connected to IoTHub.
bank info = 1. (start bank = 0)
[INFO] ADU agent started successfully!
Manufacturer: RENESAS, Model: CK-RX65N, Installed Criteria: 1.0.0.
Sent properties request.
Telemetry message send: {"temperature":22}.
[INFO] Cancel Command received
Received all properties
```

Figure 3.12 Output in Terminal Emulator Window (Example)

3.7.2 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.



Please ensure the Azure Device Update service principal has required permission. Key management actions like deployment creation and new device registration will otherwise be blocked. After granting permission, initiate a Device Sync operation from the Diagnostics tab to register any devices provisioned before permission was granted.

View compliance information for devices connected to Device Update for IoT Hub.

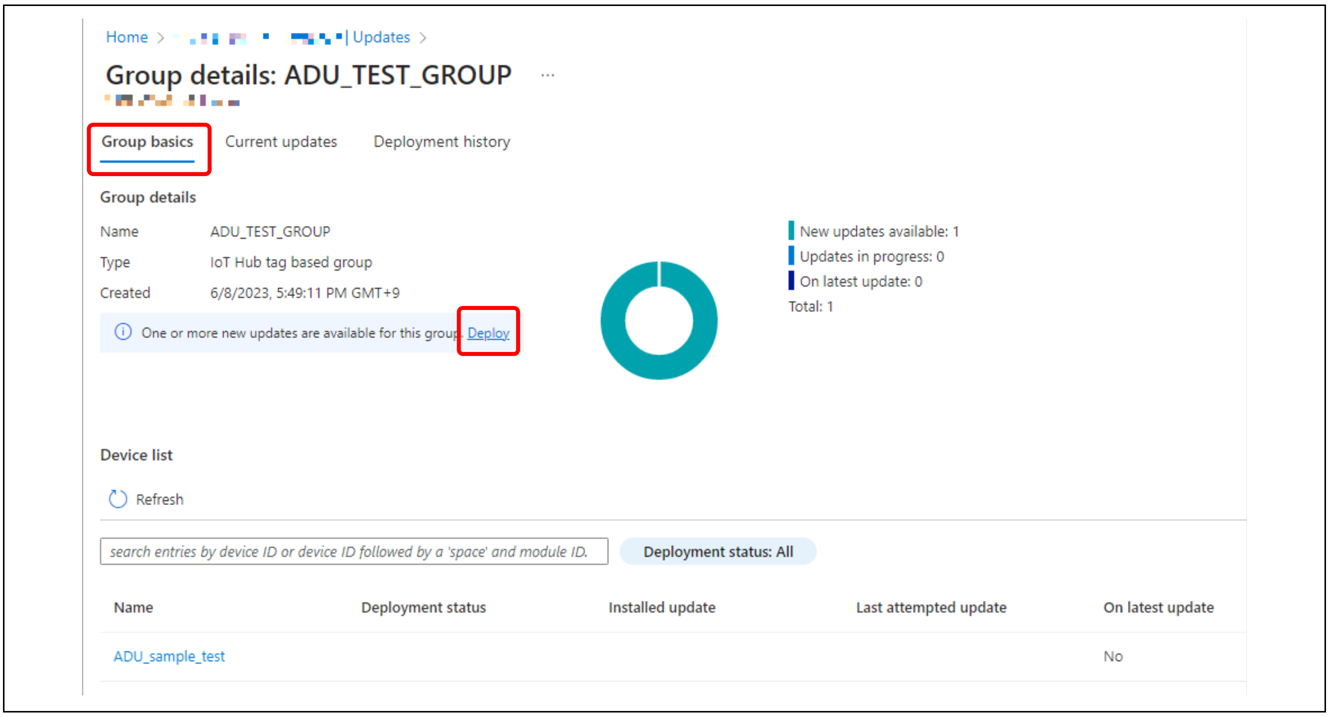
New updates available: 1
Updates in progress: 0
On latest update: 1
Total: 2

View device groups and deploy updates to member devices. Device groups are automatically created based on device tags. Learn more

□	Group name ↑	Created	Devic...	Status
□	ADU_TEST_GROUP	6/8/2023, 5:49:...	1	One or more new updates are available for this group. Deploy

Figure 3.13 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.



Group details

Name	ADU_TEST_GROUP
Type	IoT Hub tag based group
Created	6/8/2023, 5:49:11 PM GMT+9

One or more new updates are available for this group. Deploy

New updates available: 1
Updates in progress: 0
On latest update: 0
Total: 1

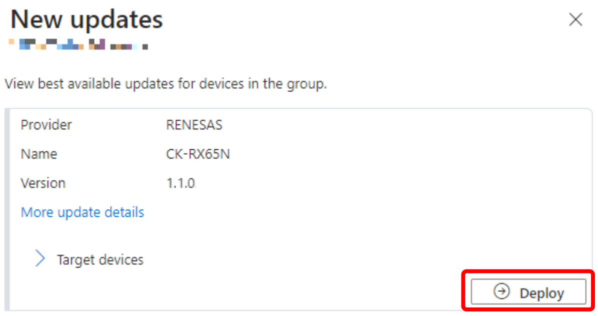
Device list

search entries by device ID or device ID followed by a 'space' and module ID.

Name	Deployment status	Installed update	Last attempted update	On latest update
ADU_sample_test				No

Figure 3.14 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.



New updates

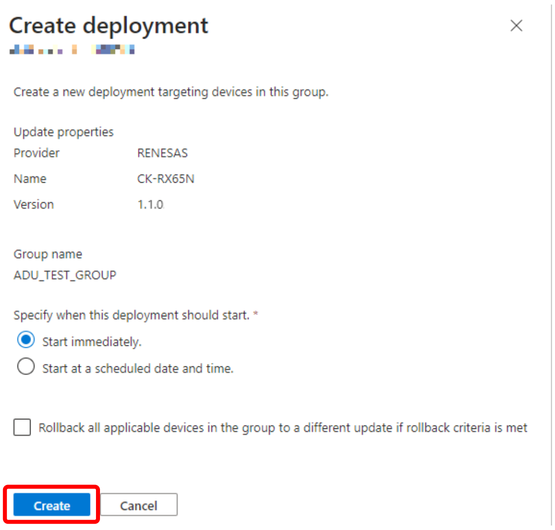
[redacted]

View best available updates for devices in the group.

Provider RENESAS
Name CK-RX65N
Version 1.1.0

Figure 3.15 New updates Page

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



Create deployment

[redacted]

Create a new deployment targeting devices in this group.

Update properties
Provider RENESAS
Name CK-RX65N
Version 1.1.0

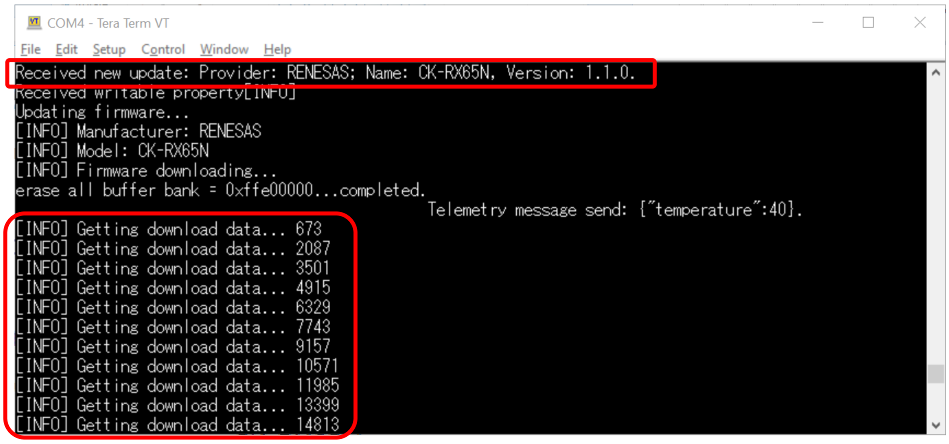
Group name
ADU_TEST_GROUP

Specify when this deployment should start. *
● Start immediately.
○ Start at a scheduled date and time.

□ Rollback all applicable devices in the group to a different update if rollback criteria is met

Figure 3.16 Create deployment Page

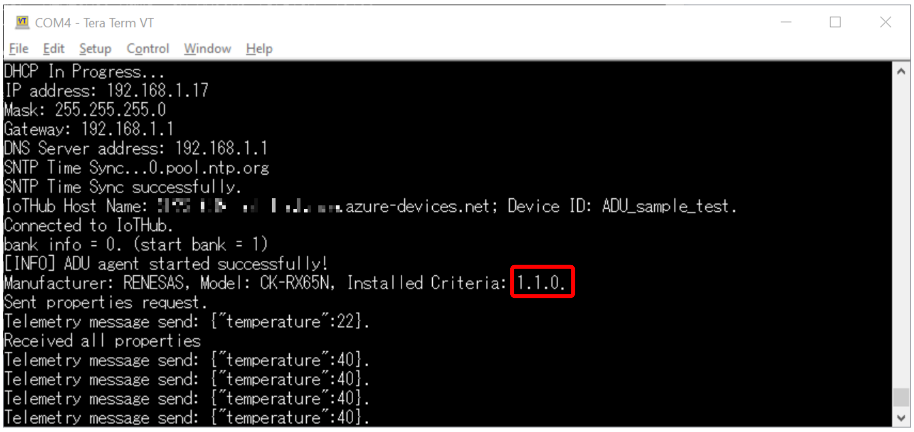
Once deployment starts, the download status is output to the terminal emulator program connected to the CK-RX65N. The downloaded firmware update is written to the flash memory, and version checking and verification data are used to verify the firmware. Next, bank switching takes place on the device,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



```
Received new update: Provider: RENESAS; Name: CK-RX65N, Version: 1.1.0.
Received writable property[INFO]
Updating firmware...
[INFO] Manufacturer: RENESAS
[INFO] Model: CK-RX65N
[INFO] Firmware downloading...
erase all buffer bank = 0xffe00000...completed.
                                                   Telemetry message send: {"temperature":40}.
[INFO] Getting download data... 673
[INFO] Getting download data... 2087
[INFO] Getting download data... 3501
[INFO] Getting download data... 4915
[INFO] Getting download data... 6329
[INFO] Getting download data... 7743
[INFO] Getting download data... 9157
[INFO] Getting download data... 10571
[INFO] Getting download data... 11985
[INFO] Getting download data... 13399
[INFO] Getting download data... 14813
```

Figure 3.17 Downloading the Firmware

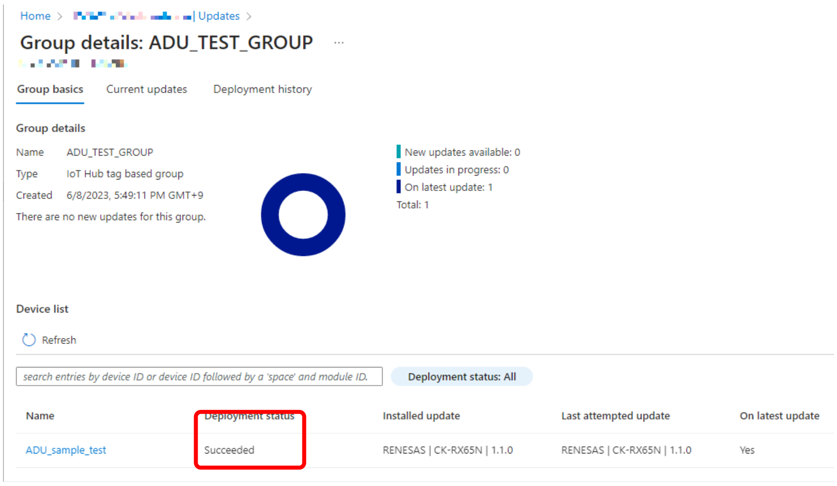
After the firmware is downloaded, the update is complete when the firmware version indicated following **Installed Criteria:** matches that of the firmware update.



```
DHCP In Progress...
IP address: 192.168.1.17
Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS Server address: 192.168.1.1
SNTP Time Sync...0.pool.ntp.org
SNTP Time Sync successfully.
IoTHub Host Name: [redacted].azure-devices.net; Device ID: ADU_sample_test.
Connected to IoTHub.
bank info = 0. (start bank = 1)
[INFO] ADU agent started successfully!
Manufacturer: RENESAS, Model: CK-RX65N, Installed Criteria: 1.1.0.
Sent properties request.
Telemetry message send: {"temperature":22}.
Received all properties
Telemetry message send: {"temperature":40}.
Telemetry message send: {"temperature":40}.
Telemetry message send: {"temperature":40}.
Telemetry message send: {"temperature":40}.
```

Figure 3.18 Firmware Update Complete

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



Home > [redacted] | Updates >

Group details: ADU_TEST_GROUP ...

[redacted]

Group basics
Current updates
Deployment history

Group details

Name ADU_TEST_GROUP

Type IoT Hub tag based group

Created 6/8/2023, 5:49:11 PM GMT+9

There are no new updates for this group.

New updates available: 0

Updates in progress: 0

On latest update: 1

Total: 1

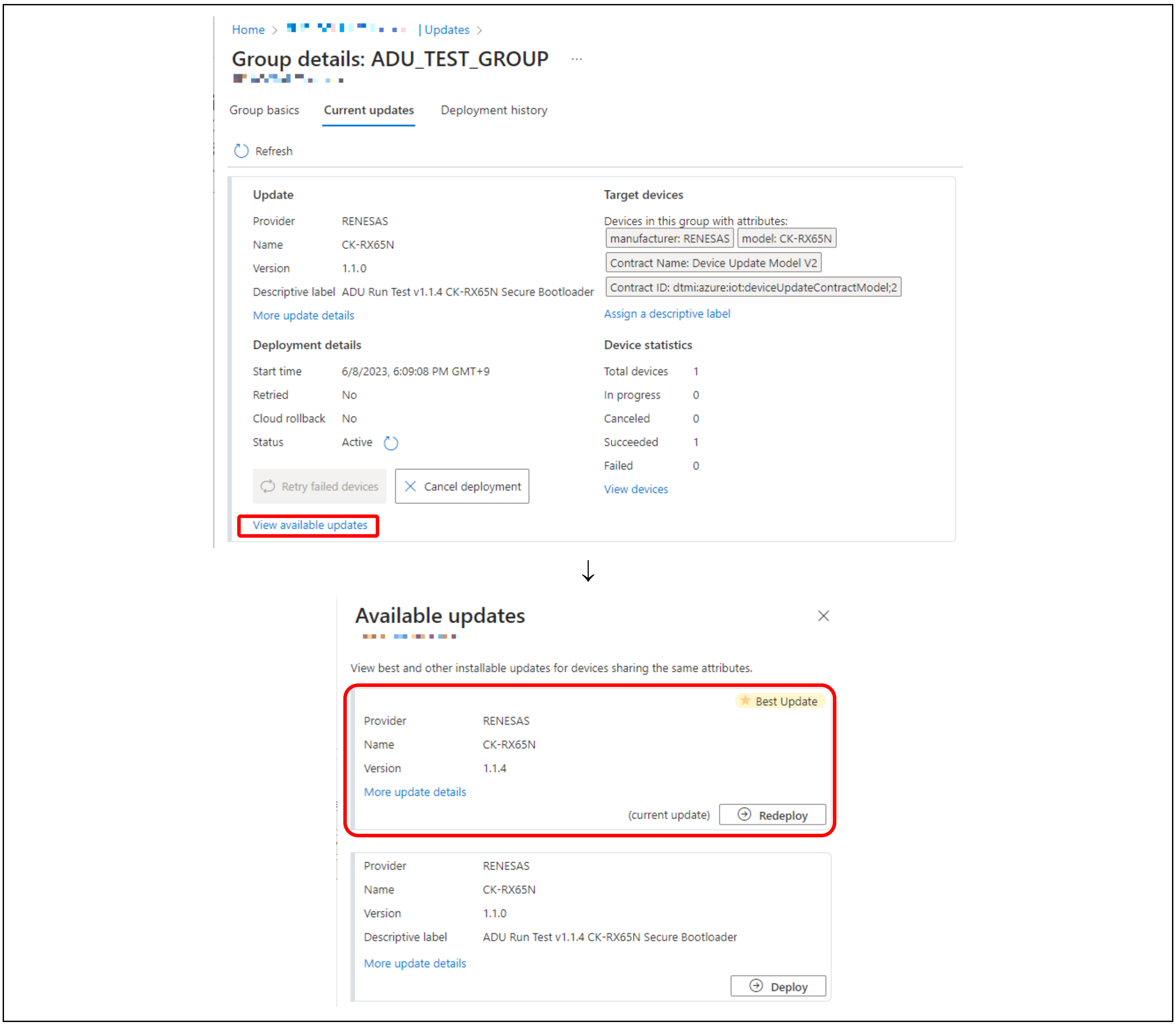
Device list

Refresh
search entries by device ID or device ID followed by a 'space' and module ID.
Deployment status: All

Name	Deployment status	Installed update	Last attempted update	On latest update
ADU_sample_test	Succeeded	RENESAS | CK-RX65N | 1.1.0	RENESAS | CK-RX65N | 1.1.0	Yes

Figure 3.19 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.



Home > [redacted] | Updates >

Group details: ADU_TEST_GROUP ...

[redacted]

Group basics | Current updates | Deployment history

Refresh

Update

Provider	RENESAS
Name	CK-RX65N
Version	1.1.0
Descriptive label	ADU Run Test v1.1.4 CK-RX65N Secure Bootloader

More update details

Target devices

Devices in this group with attributes:

manufacturer: RENESAS | model: CK-RX65N

Contract Name: Device Update Model V2

Contract ID: dtmi:azure:iot:deviceUpdateContractModel;2

Assign a descriptive label

Deployment details

Start time	6/8/2023, 6:09:08 PM GMT+9
Retried	No
Cloud rollback	No
Status	Active

Retry failed devices | Cancel deployment

View available updates

Device statistics

Total devices	1
In progress	0
Canceled	0
Succeeded	1
Failed	0

View devices

↓

Available updates

[redacted]

View best and other installable updates for devices sharing the same attributes.

Best Update

Provider	RENESAS
Name	CK-RX65N
Version	1.1.4

More update details

(current update) Redeploy

Provider	RENESAS
Name	CK-RX65N
Version	1.1.0
Descriptive label	ADU Run Test v1.1.4 CK-RX65N Secure Bootloader

More update details

Deploy

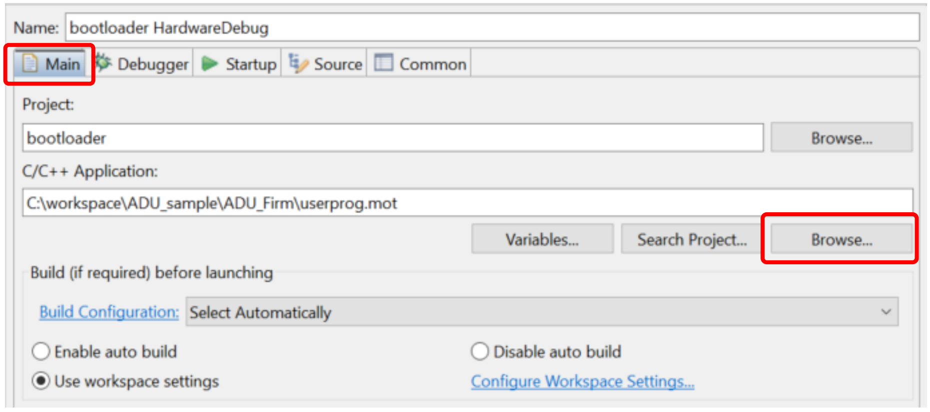
Figure 3.20 Selecting Among Available Updates

4. Appendix

How to source debug initial and updated firmware using e^2 studio is described below.

4.1 Debugging the Initial Firmware

1. Change the debug settings for the **bootloader** project as follows in e^2 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.



Name: bootloader HardwareDebug

Project: bootloader

C/C++ Application: C:\workspace\ADU_sample\ADU_Firm\userprog.mot

Build (if required) before launching

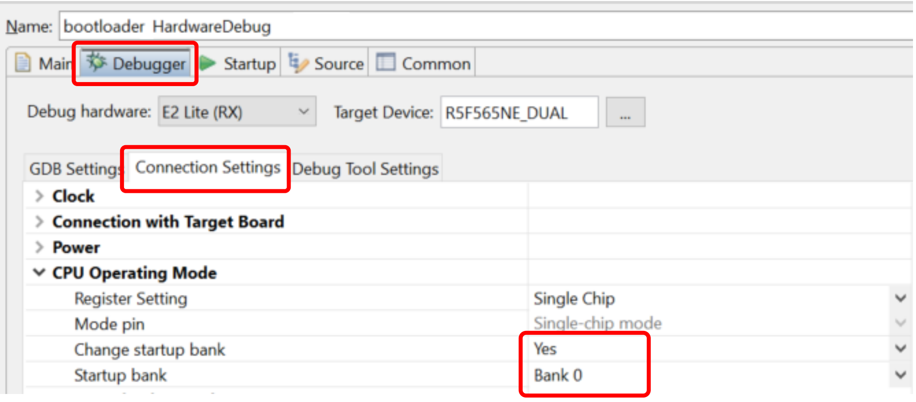
Build Configuration: Select Automatically

○ Enable auto build ○ Disable auto build

● Use workspace settings

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

Select **Debugger** tab → **Connection Settings** and set **Startup bank** to **Bank 0**.



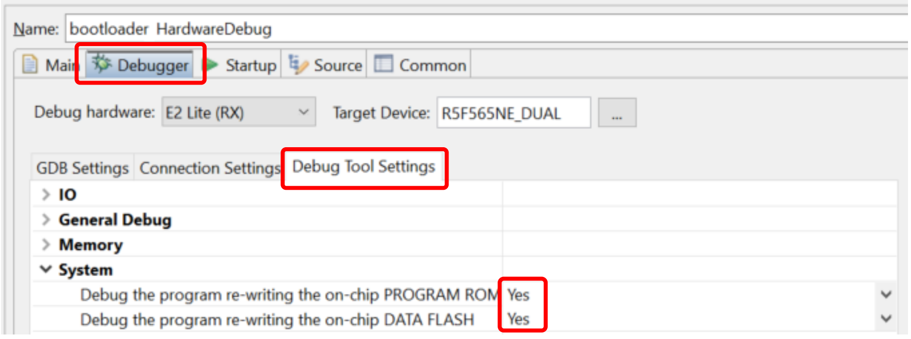
Name: bootloader HardwareDebug

Debug hardware: E2 Lite (RX) Target Device: R5F565NE_DUAL

> Clock	
> Connection with Target Board	
> Power	
∨ CPU Operating Mode	
Register Setting	Single Chip
Mode pin	Single-chip mode
Change startup bank	Yes
Startup bank	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**.



Name: bootloader HardwareDebug

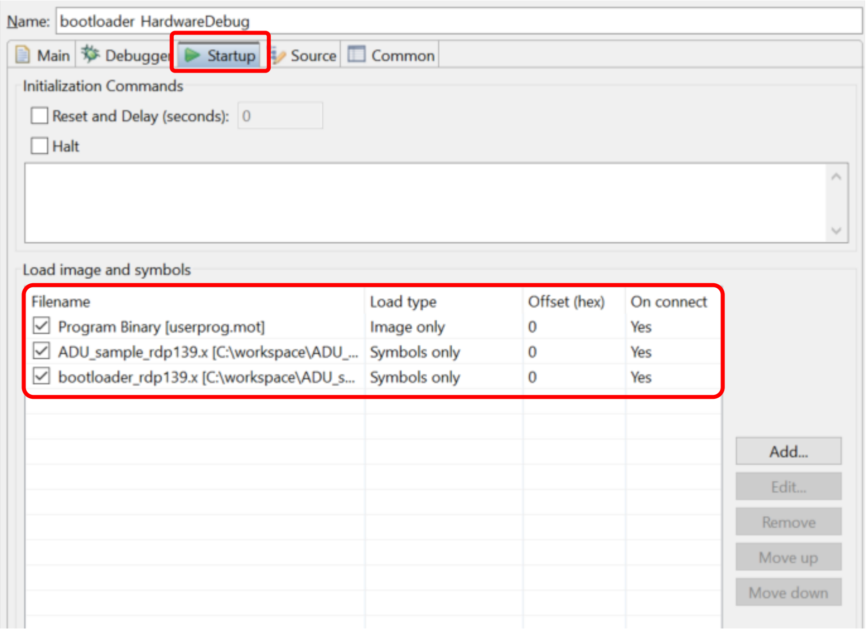
Debug hardware: E2 Lite (RX) Target Device: R5F565NE_DUAL

Debug the program re-writing the on-chip PROGRAM ROM Yes

Debug the program re-writing the on-chip DATA FLASH 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**.



Name: bootloader HardwareDebug

Initialization Commands

□ Reset and Delay (seconds): 0

□ Halt

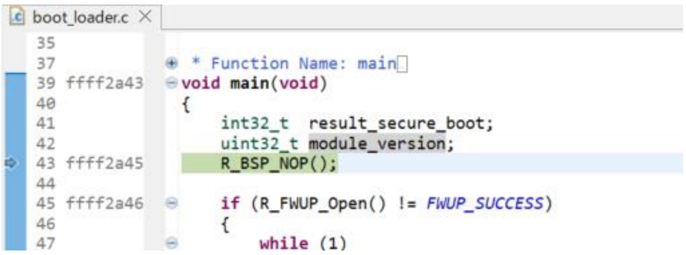
Load image and symbols

Filename	Load type	Offset (hex)	On connect
☑ Program Binary [userprog.mot]	Image only	0	Yes
☑ ADU_sample_rdp139.x [C:\workspace\ADU_...	Symbols only	0	Yes
☑ bootloader_rdp139.x [C:\workspace\ADU_s...	Symbols only	0	Yes

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

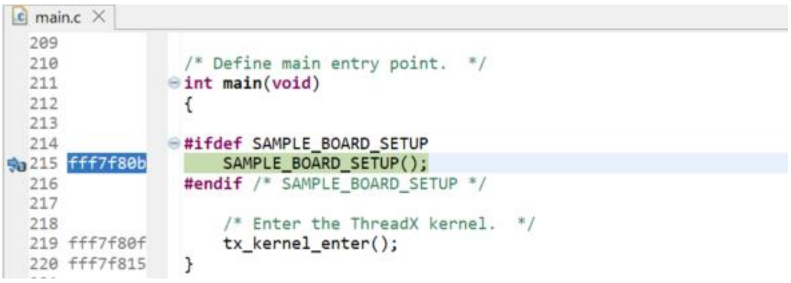
[bootloader.c]



```
 * Function Name: main
void main(void)
{
    int32_t  result_secure_boot;
    uint32_t module_version;
    R_BSP_NOP();

    if (R_FWUP_Open() != FWUP_SUCCESS)
    {
        while (1)
```

Figure 4.5 Breakpoint in bootloader.c

[main.c]



```
/* Define main entry point.  */
int main(void)
{

#ifdef SAMPLE_BOARD_SETUP
    SAMPLE_BOARD_SETUP();
#endif /* SAMPLE_BOARD_SETUP */

    /* Enter the ThreadX kernel.  */
    tx_kernel_enter();
}
```

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
```

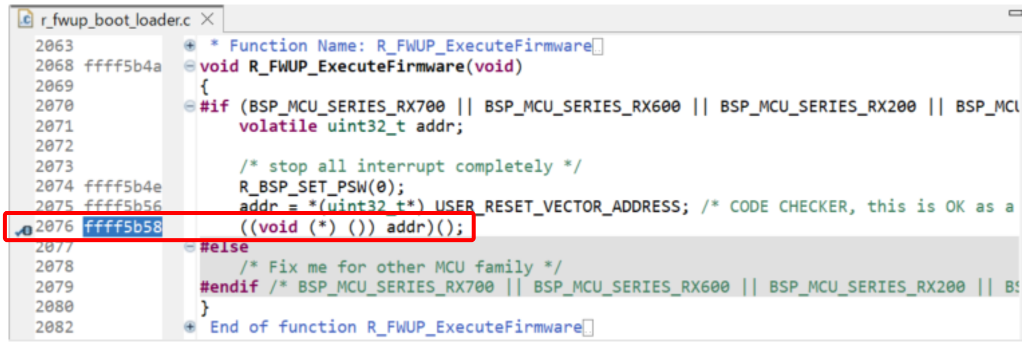
Note that **memdump.mot** is generated in the project folder.

4.2 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. Set breakpoints after the initial firmware boots and just before the bootloader boots the updated firmware. 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]

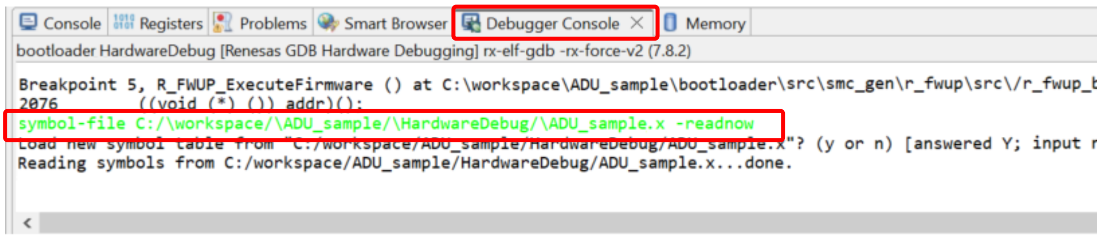


```
                * Function Name: R_FWUP_ExecuteFirmware
ffff5b4a       void R_FWUP_ExecuteFirmware(void)
               {
               #if (BSP_MCU_SERIES_RX700 || BSP_MCU_SERIES_RX600 || BSP_MCU_SERIES_RX200 || BSP_MCU
                   volatile uint32_t addr;

                   /* stop all interrupt completely */
ffff5b4e           R_BSP_SET_PSW(0);
ffff5b56           addr = *(uint32_t*) USER_RESET_VECTOR_ADDRESS; /* CODE CHECKER, this is OK as a
ffff5b58           ((void (*) ()) addr)();
               #else
                   /* Fix me for other MCU family */
               #endif /* BSP_MCU_SERIES_RX700 || BSP_MCU_SERIES_RX600 || BSP_MCU_SERIES_RX200 || BS
               }
                End of function R_FWUP_ExecuteFirmware
```

Figure 4.7 Location of Breakpoint

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: If the updated firmware **.x** file is in **C:\temp***[1]
   symbol-file C:/\temp/\adu_sample.x -readnow

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



bootloader HardwareDebug [Renesas GDB Hardware Debugging] rx-elf-gdb -rx-force-v2 (7.8.2)

```
Breakpoint 5, R_FWUP_ExecuteFirmware () at C:\workspace\ADU_sample\bootloader\src\smc_gen\r_fwup\src\/r_fwup_b
2076            ((void (*) ()) addr)();
symbol-file C:/\workspace/\ADU_sample/\HardwareDebug/\ADU_sample.x -readnow
Load new symbol table from "C:/workspace/ADU_sample/HardwareDebug/ADU_sample.x"? (y or n) [answered Y; input r
Reading symbols from C:/workspace/ADU_sample/HardwareDebug/ADU_sample.x...done.
```

Figure 4.8 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/us/en/support/contact.html

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	July 31, 2023	—	First edition issued

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/.