Renesas RX Family

Azure RTOS TraceX for Azure RTOS ThreadX Debugging

Introduction

Azure RTOS ThreadX is an RTOS from Microsoft Corporation and is based on a high-performance embedded kernel.

Azure RTOS TraceX is a Windows-based analysis tool for use with real-time operating systems. It provides embedded developers with a graphical view of real-time system events and enables them to visualize and better understand the behavior of their real-time systems.

This application note describes procedures for checking Azure RTOS ThreadX thread and object states (referred to as resources) during the development of applications in the e^2 studio. The procedure for installing and starting Azure RTOS ThreadX is also explained.

Target Device

RX65N Group (R5F565NEHDFB)

Operating Environment

Target Board	CK-RX65N
Integrated Development Environment (IDE)	e^2 studio version 2023-01
Compiler	CC-RX V3.04.00
Trace Tool	Microsoft Azure RTOS TraceX v6.1.12.0
OS	Microsoft Azure RTOS ThreadX v6.2.0

Note: Please download the installer for the IDE from the page at the URL linked below in advance.
e² studio -information for RX Family | Renesas

This application note has been adapted for RX-support from the following application note.

RA Family Azure RTOS TraceX for Azure RTOS ThreadX Debugging

Refer also to the following documents.

- RX Smart Configurator User's Guide: e^2 studio site:
  RX Smart Configurator User's Guide: e² studio
- Azure RTOS ThreadX documents on the Microsoft site:
  Azure RTOS ThreadX documentation | Microsoft Learn

  Azure RTOS TraceX documentation | Microsoft Learn

Contents

1. Installing the e2 studio 3
2. Installing Azure RTOS TraceX 3
3. Setting the CK-RX65N Board and Connection to the Host PC 4
4. Creating a ThreadX Project in the e2 studio 5
4.1 Procedure for Creating a Project 5
4.2 Downloading a Component 9
4.3 Configuring Components 11
4.4 Generating Source Files 14
4.5 Editing Source Files 15
4.6 Building the Project 17
4.7 Executing the Project 18
5. Sample Project 19
6. Setting the Trace Buffer for Use by TraceX 21
7. Using the [RTOS Resources] View 22
7.1 Displaying the [RTOS Resources] View 22
7.2 Context Menu 23
7.3 Stack Setting 24
7.4 Tabbed Pages 25
8. Starting Debugging of a Project with Azure RTOS TraceX 27
8.1 Acquiring the Address of the TraceX Data Buffer 27
8.2 Exporting a Data File for TraceX 27
9. Launching Azure RTOS TraceX 29
9.1 Display of Trace Information 29
9.1.1 Sequential View Mode 29
9.1.2 Time View Mode 30
9.2 Analysis Facilities of TraceX 30

1. Installing the e^2 studio

Install the e^2 studio with reference to the videos related to procedures for installing the e^2 studio at links from the following URL.

RX Family Software & Tool Course | Renesas

2. Installing Azure RTOS TraceX

Download Azure RTOS TraceX from the following Microsoft Store Apps site and install it.

Azure RTOS TraceX - Microsoft Store Apps

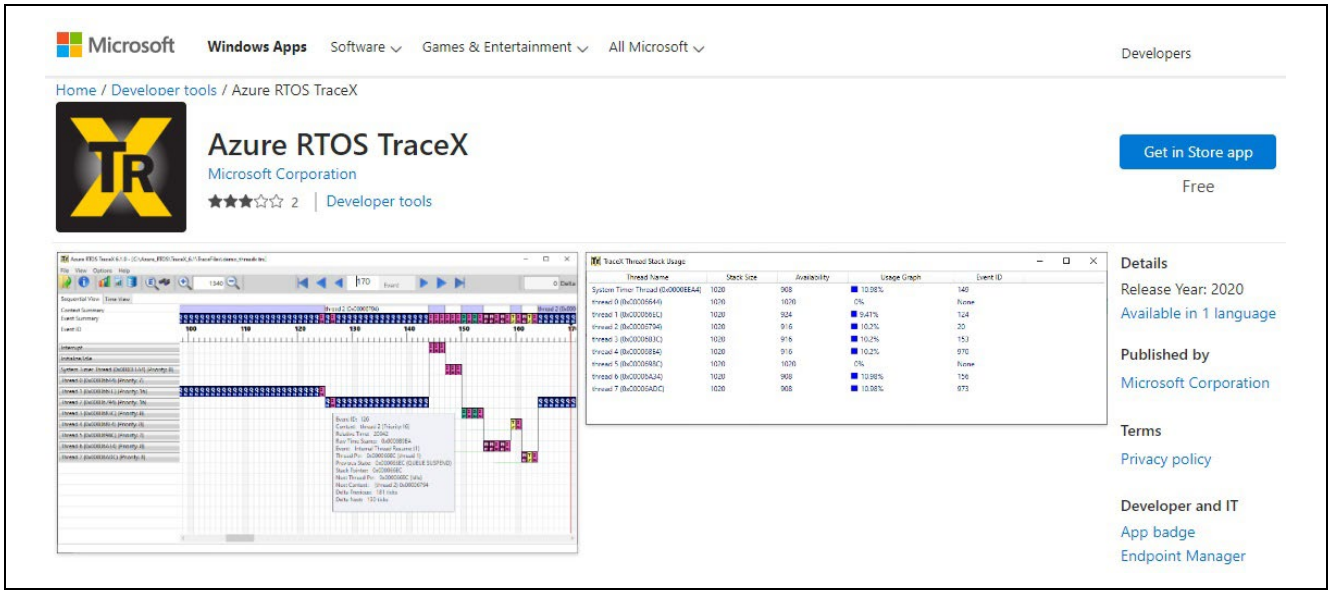


Figure 2-1 Microsoft Store Apps

3. Setting the CK-RX65N Board and Connection to the Host PC

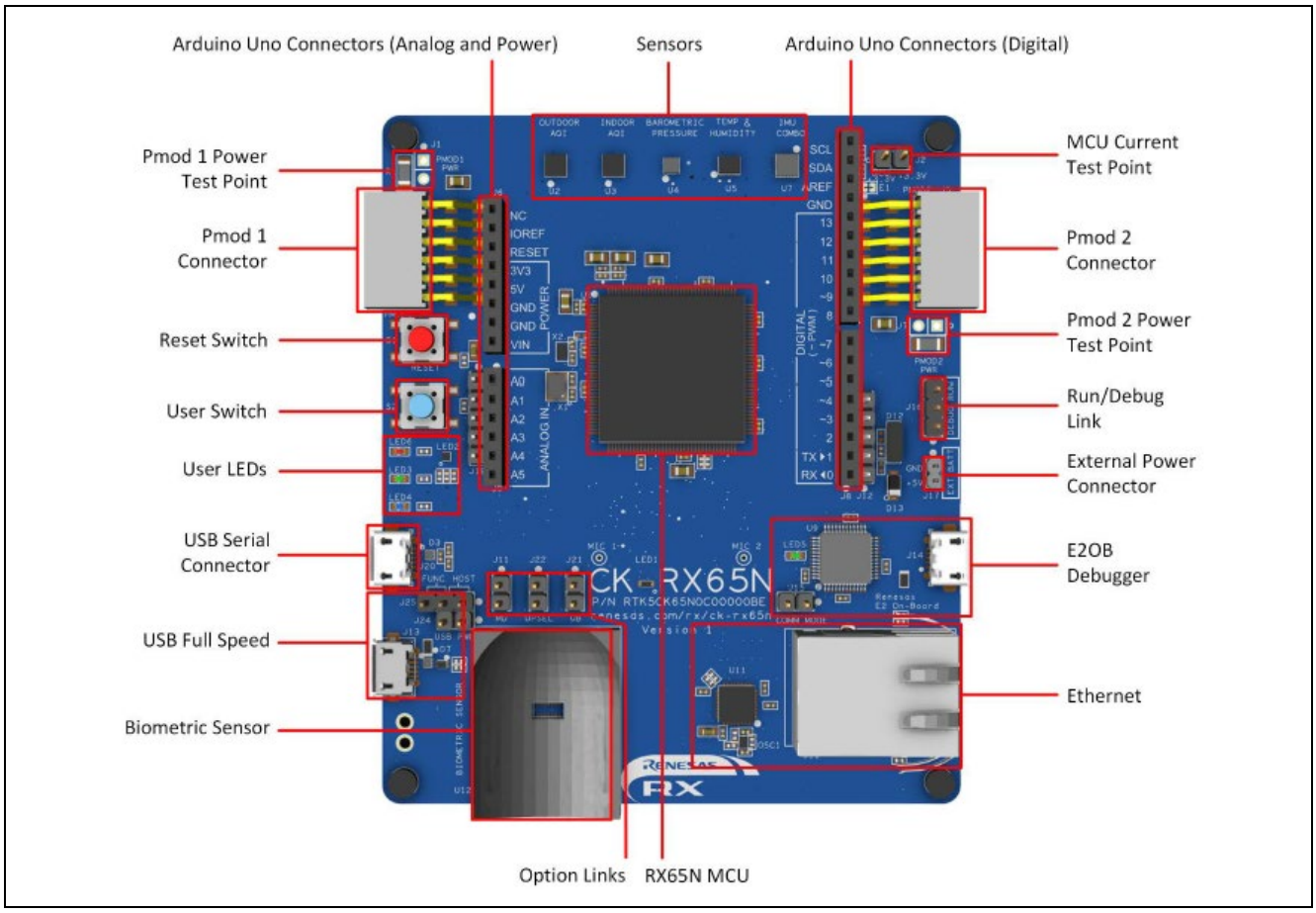


Figure 3-1 CK-RX65N Board

Make a jumper setting on the board and connect the board to the host PC with a USB cable as shown below.

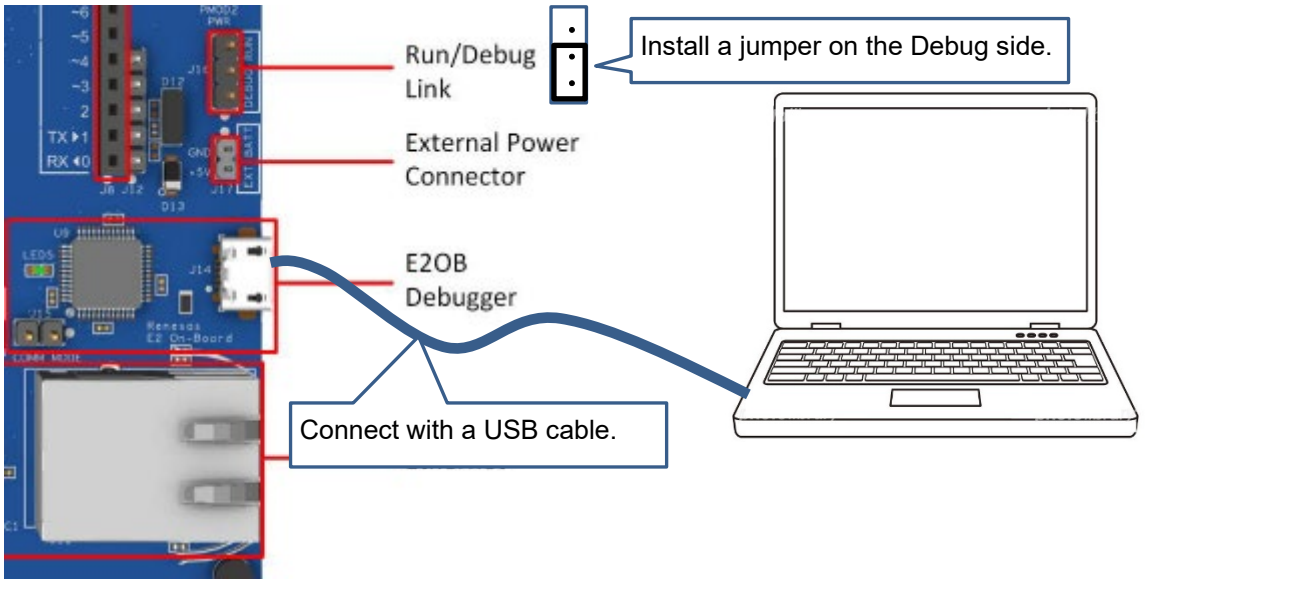


Figure 3-2 Jumper Setting and Connection between the CK-RX65N Board and the Host PC

4. Creating a ThreadX Project in the e² studio

Create a project by following the procedure below.

4.1 Procedure for Creating a Project

(1) Select the [File] menu → [New] → [Renesas C/C++ Project] → [Renesas RX].

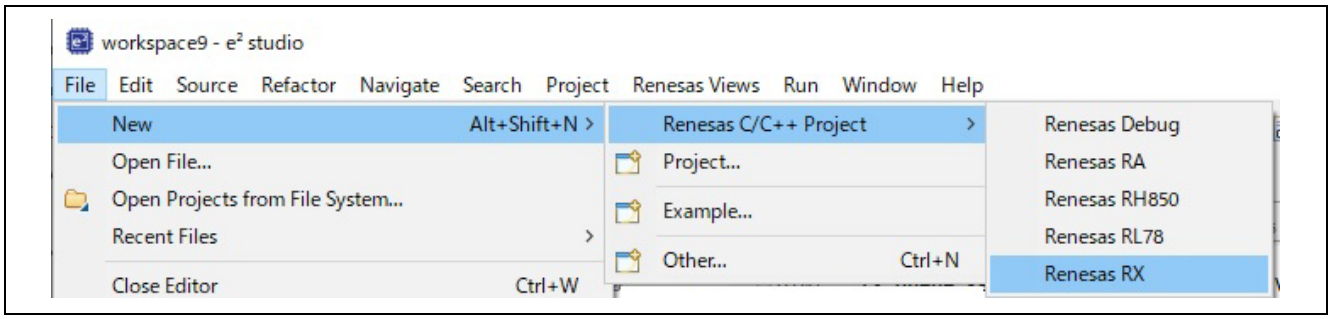


Figure 4-1 Selecting a Project Type and a Device Family

(2) Select the "Renesas CC-RX C/C++ Executable Project" template and click on [Next] to continue.

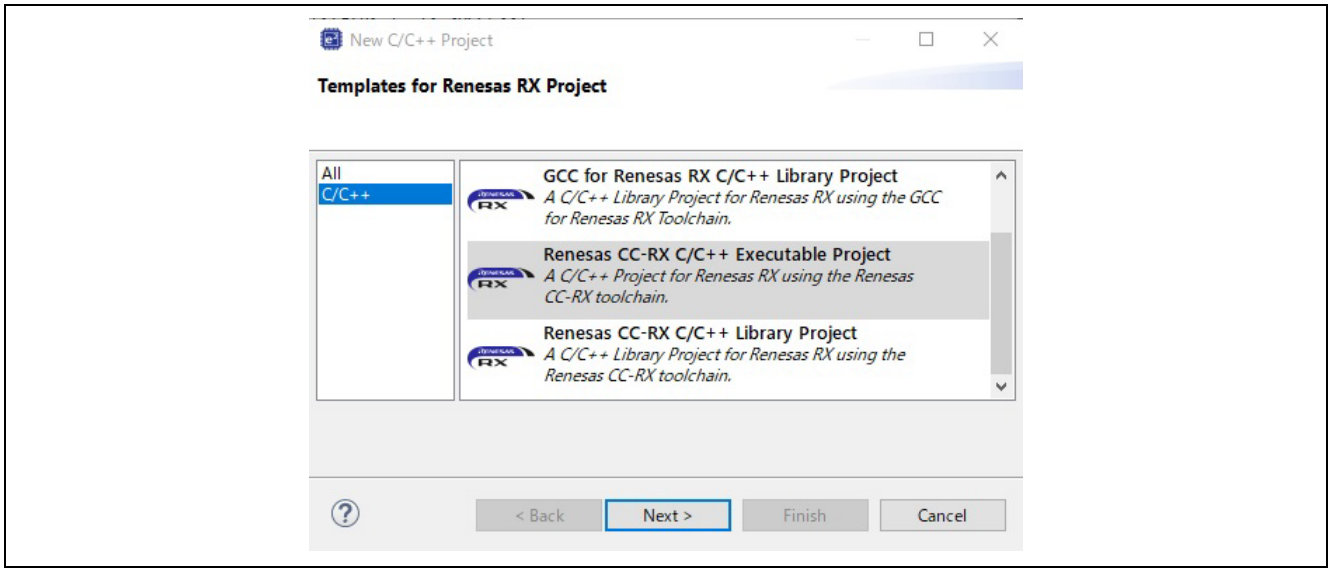


Figure 4-2 Selecting an Executable Project

(3) In the next dialog box, enter a project name and click on [Next].

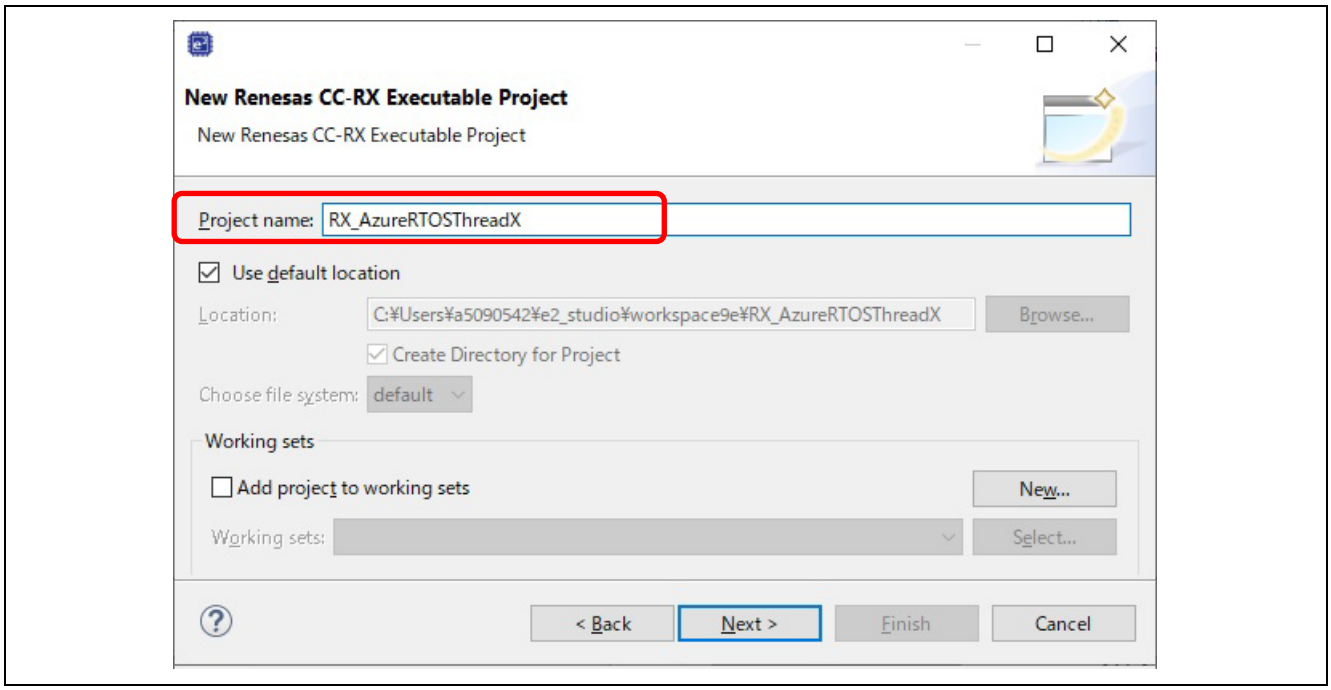


Figure 4-3 Entering a Project Name

(4) In the device selection dialog box, enter information on the device and tool.

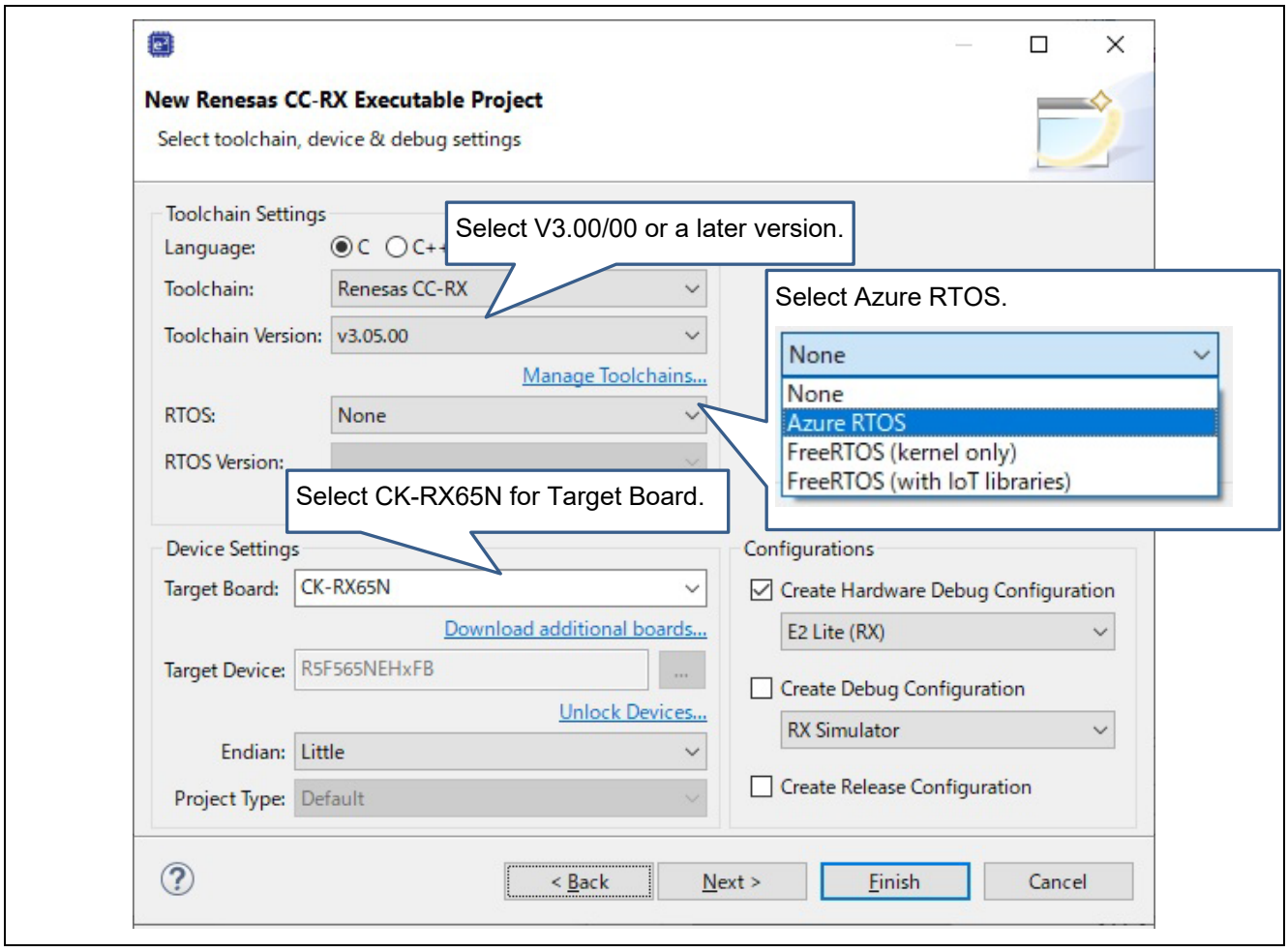


Figure 4-4 Selecting the Board and OS

(5) Use the default settings of the Smart Configurator.

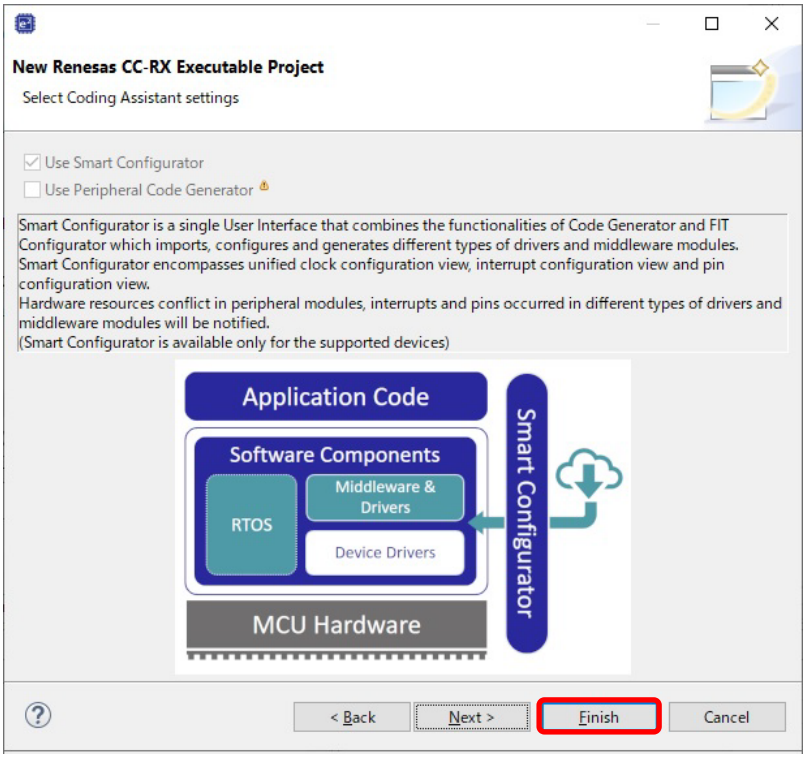


Figure 4-5 Smart Configurator Settings

(6) Select "ThreadX sample project".

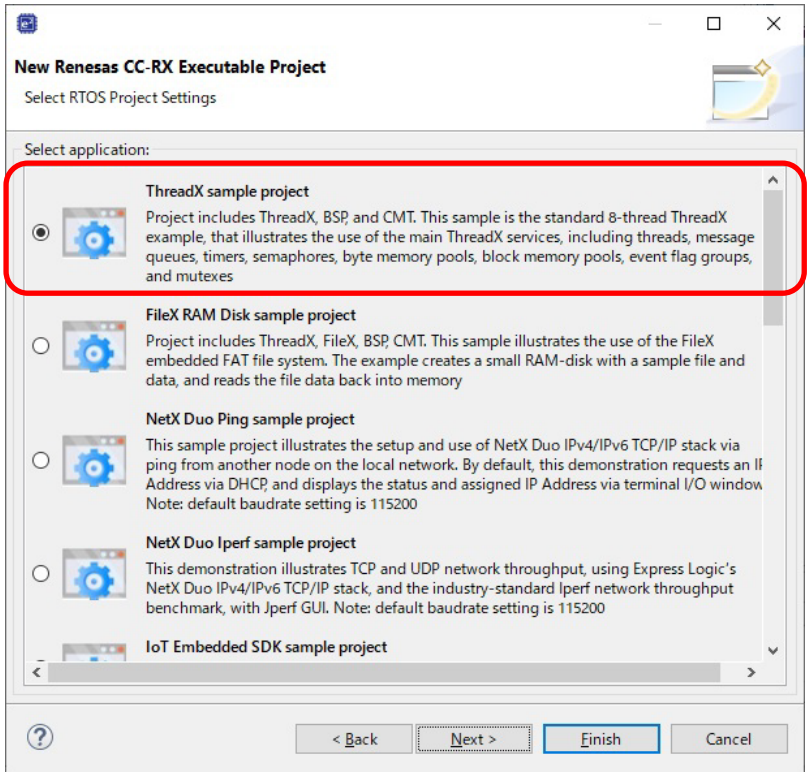


Figure 4-6 Selecting "ThreadX sample project"

(7) Click on [Finish] to complete project creation.

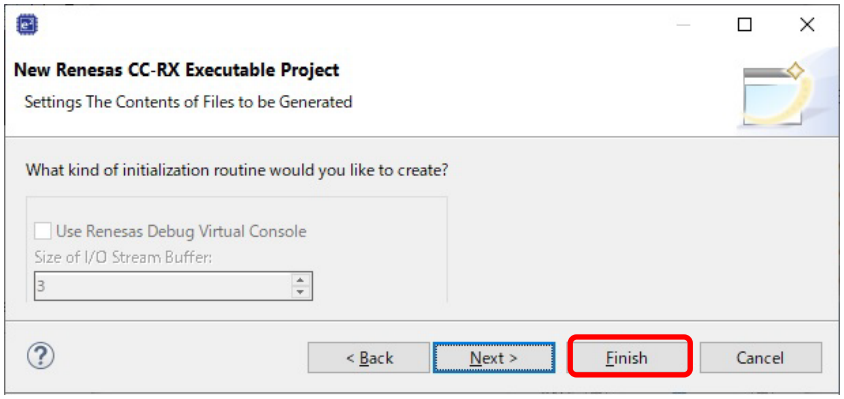


Figure 4-7 Completing Project Creation

(8) Select the current text editor.

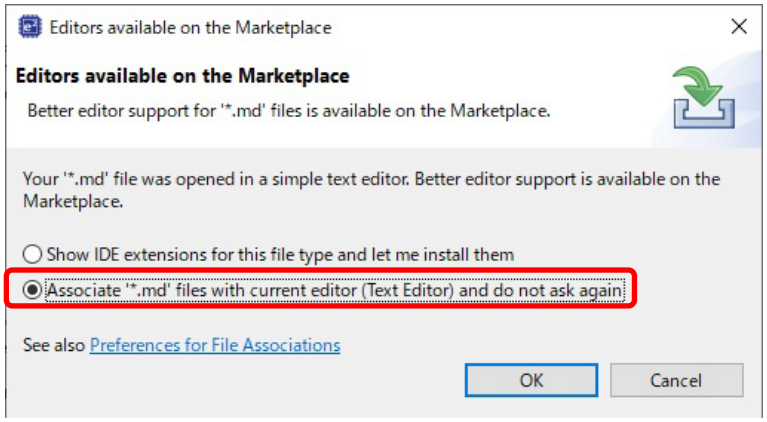


Figure 4-8 Selecting a Text Editor

4.2 Downloading a Component

Click on "RX_AzureRTOSThreadX.cfg" to open the [Software component configuration] panel.

If the icon for "r_cmt_rx" is gray, the component is disabled. Download it as follows.

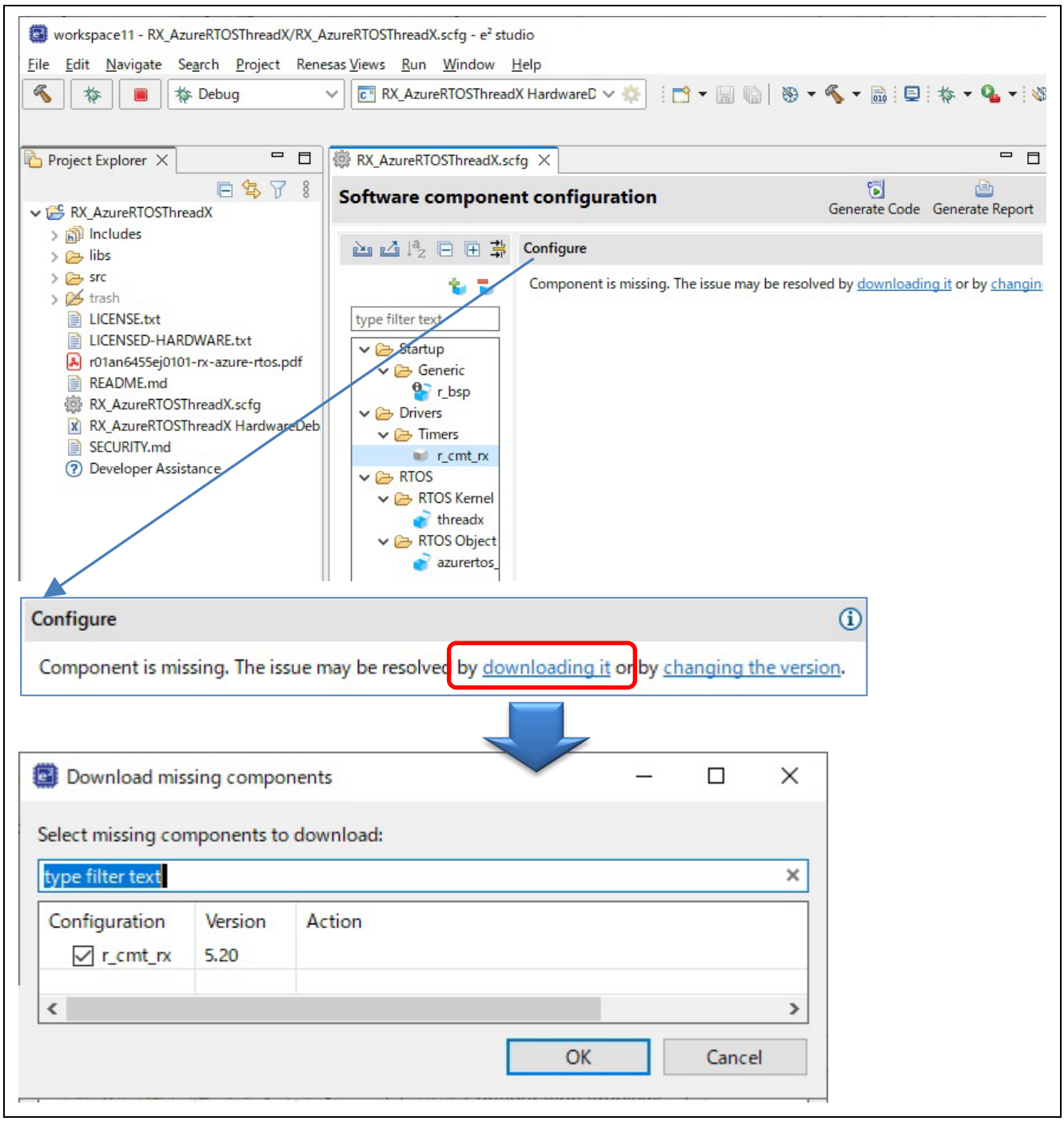


Figure 4-9 Downloading a Component

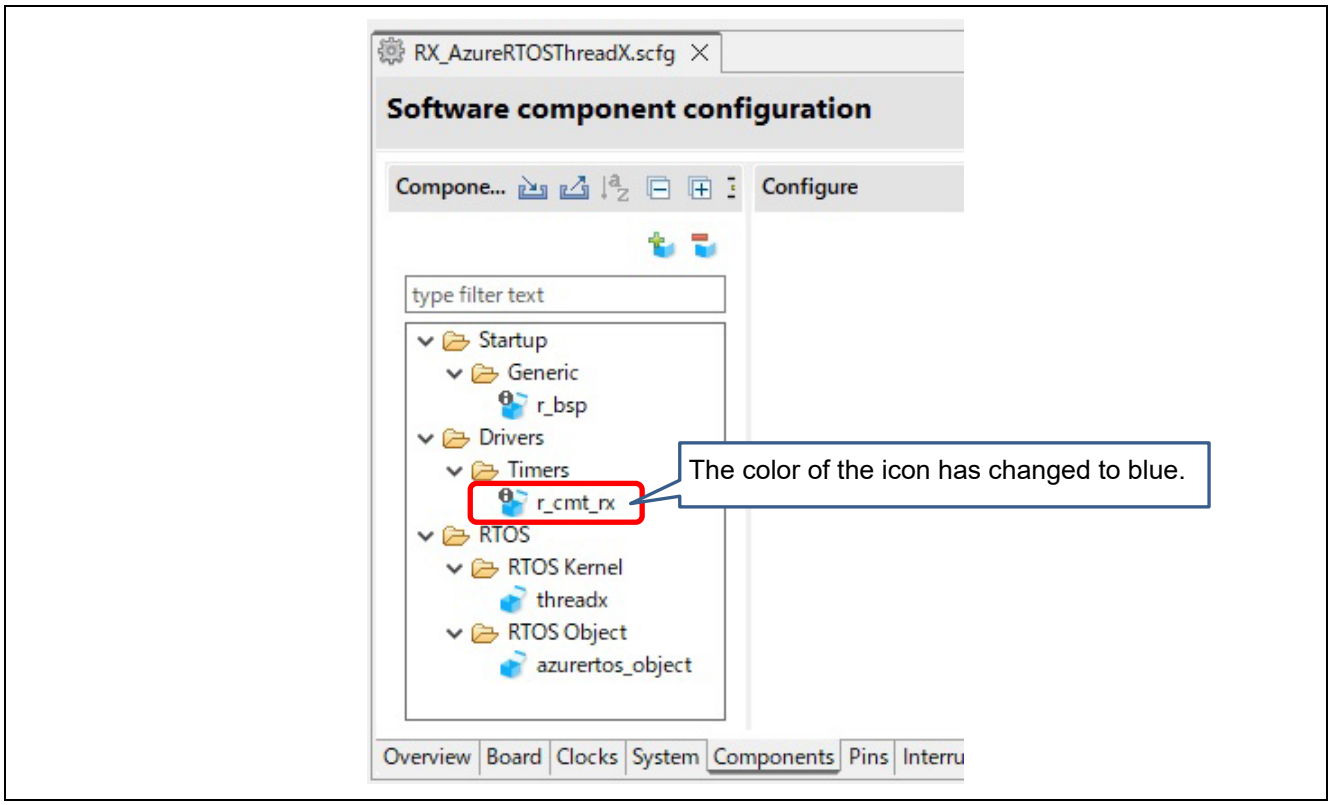


Figure 4-10 Component after Updating

Check the versions of the components on the [Overview] tabbed page.

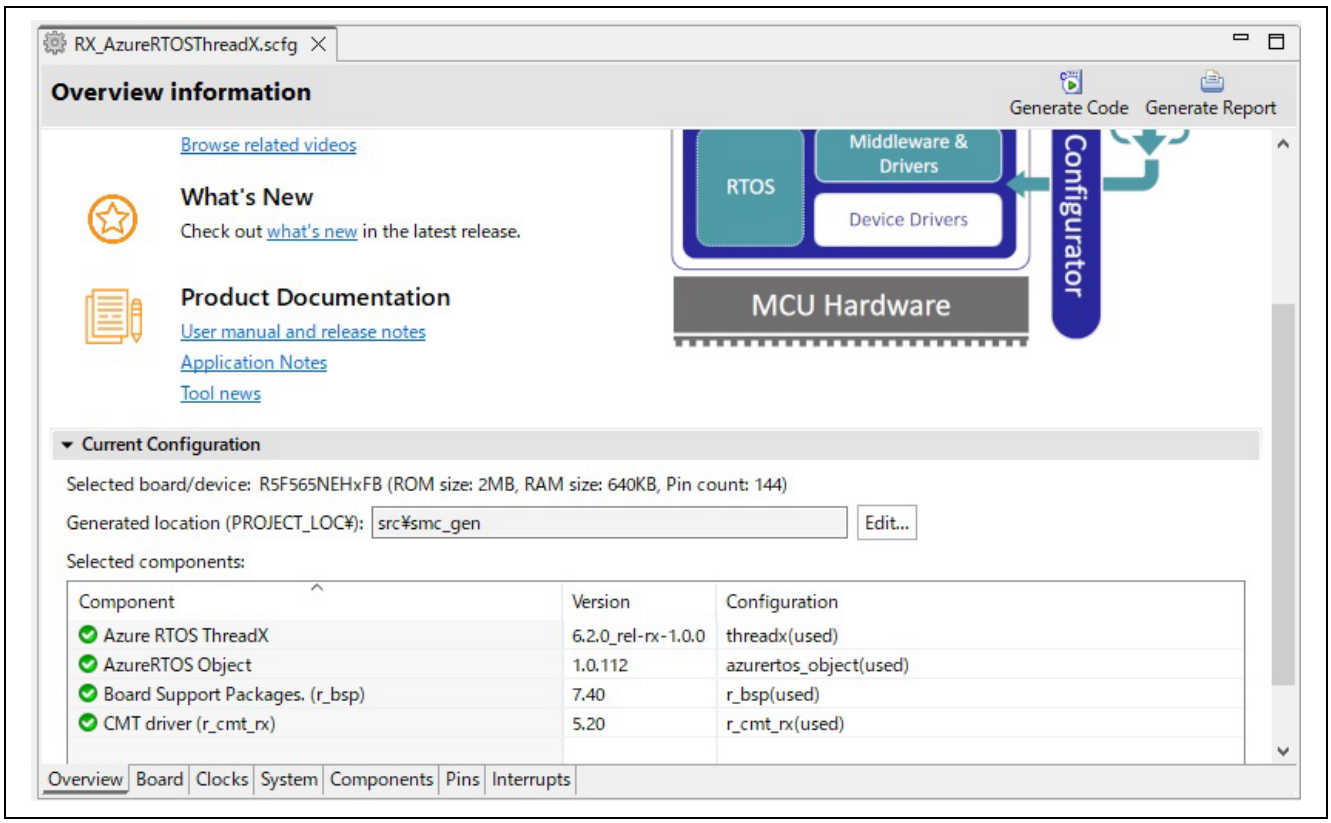


Figure 4-11 Checking the Versions of the Components Used

4.3 Configuring Components

(1) Enable the trace events shown in the figure below under [Property] in the [Configure] panel for the ThreadX component.

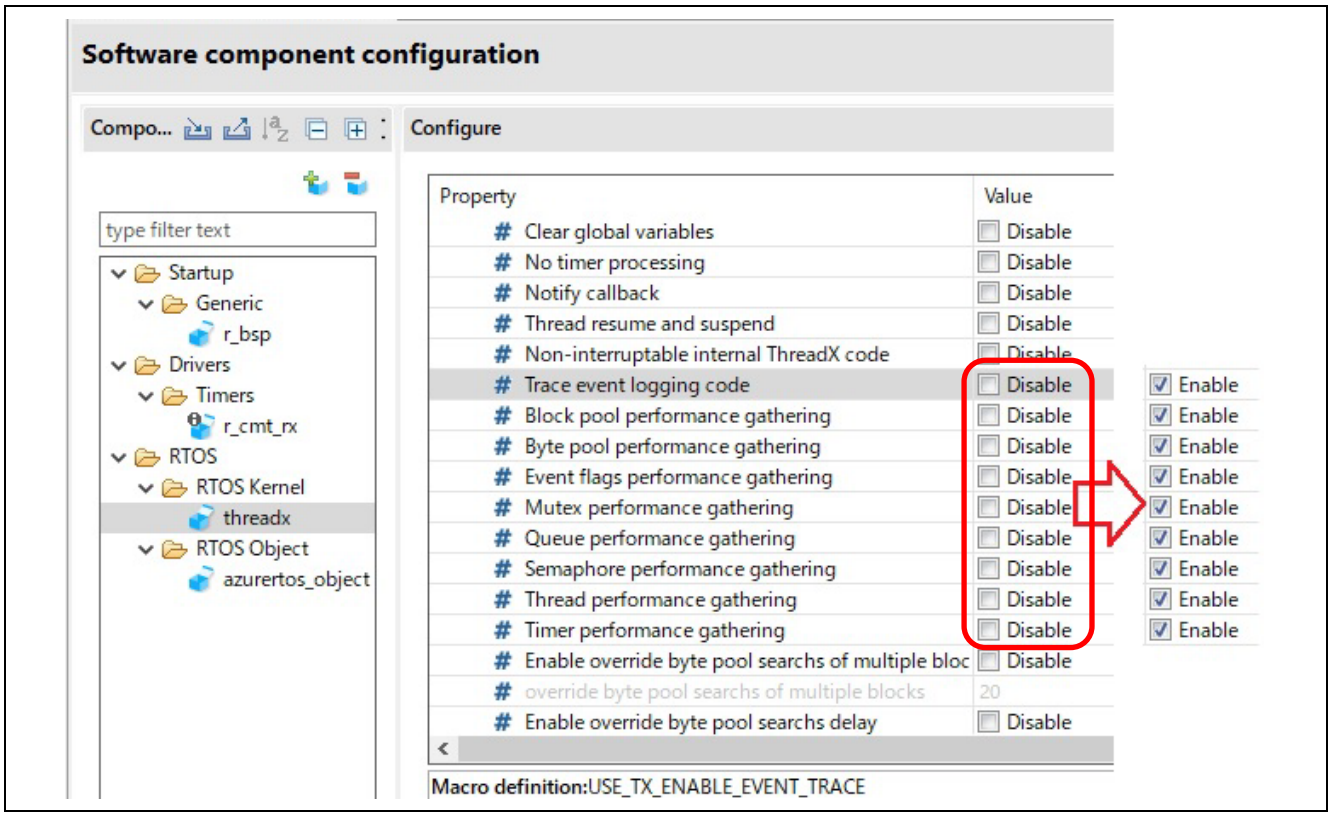


Figure 4-12 Setting ThreadX Properties

(2) Open the [Software Component Selection] dialog box and add "Ports".

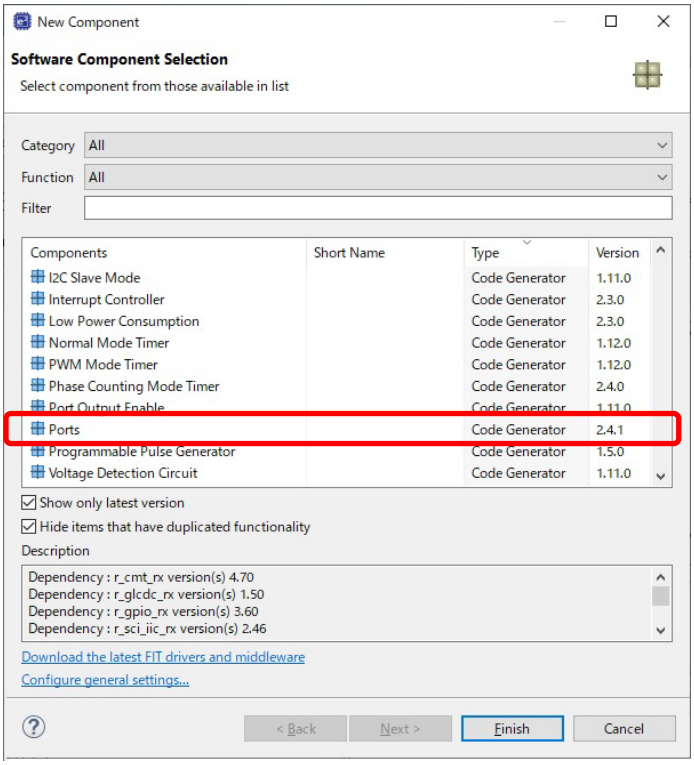
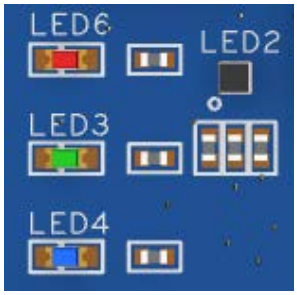


Figure 4-13 Adding "Ports"

(3) Configuring a port component

To control LED6 on the board, configure the corresponding port pin with the Smart Configurator.



Indication on the Board	Color of LED	Connected Ports
LED2	Color LED	P17, PA5, PA7
LED6	Red LED	P25
LED3	Green LED	P22
LED4	Blue LED	PA3

Figure 4-14 LED to be Controlled on the Board

LED6 is connected to P25.

The following diagram shows the connection of LED6 on the board. As the LED turns on when 0 is output from the port pin, the initial output value should be set to 1.

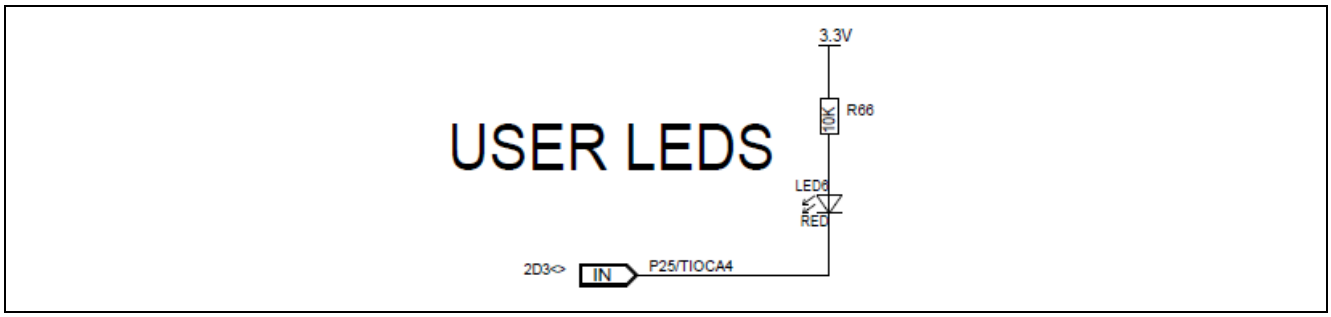


Figure 4-15 Schematic of LED6 Connection

(4) Configuring PORT2

Select the checkbox for PORT2.

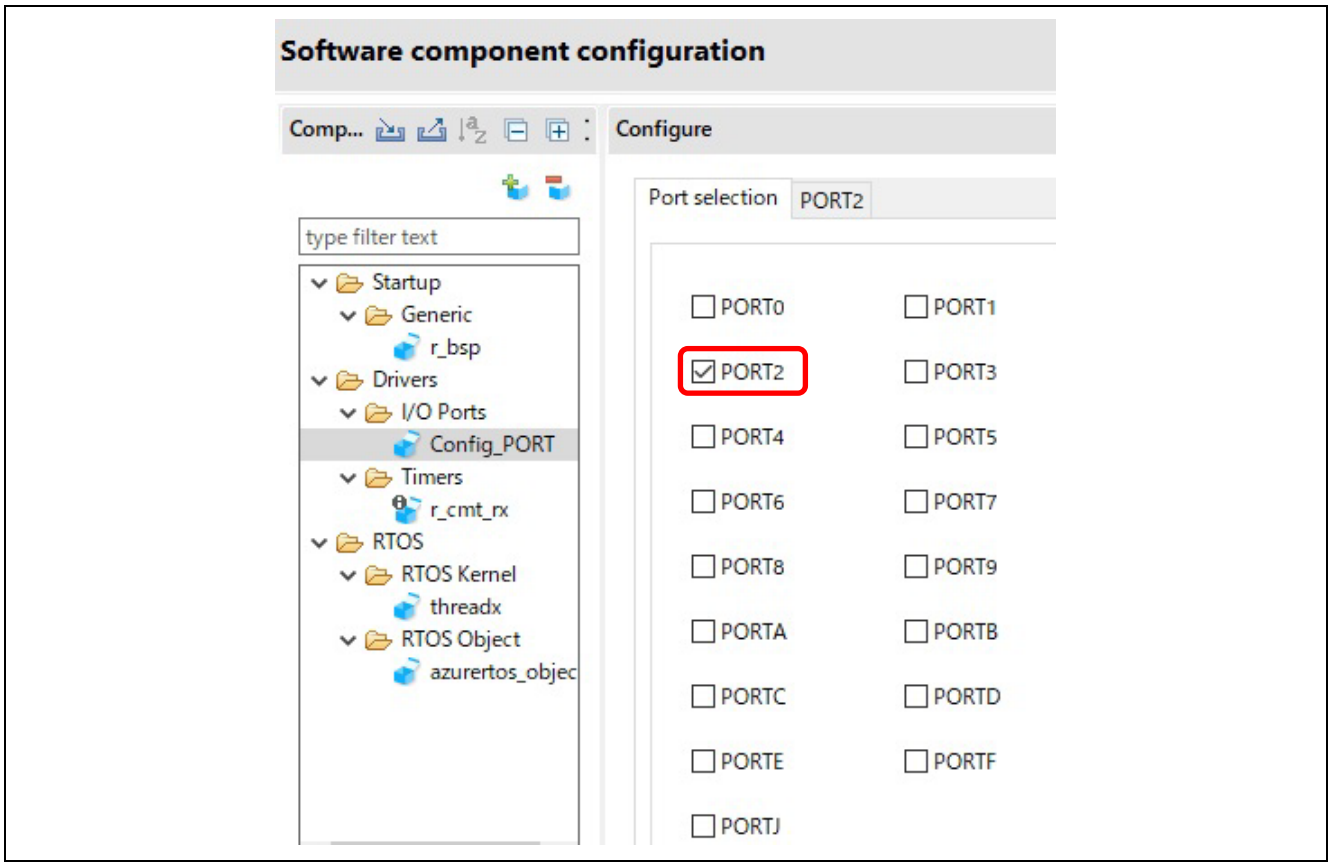


Figure 4-16 Configuring PORT2

(5) Setting P25 as an output port pin

As the LED turns on when 0 is output from the port pin, set the initial value to "Output 1".

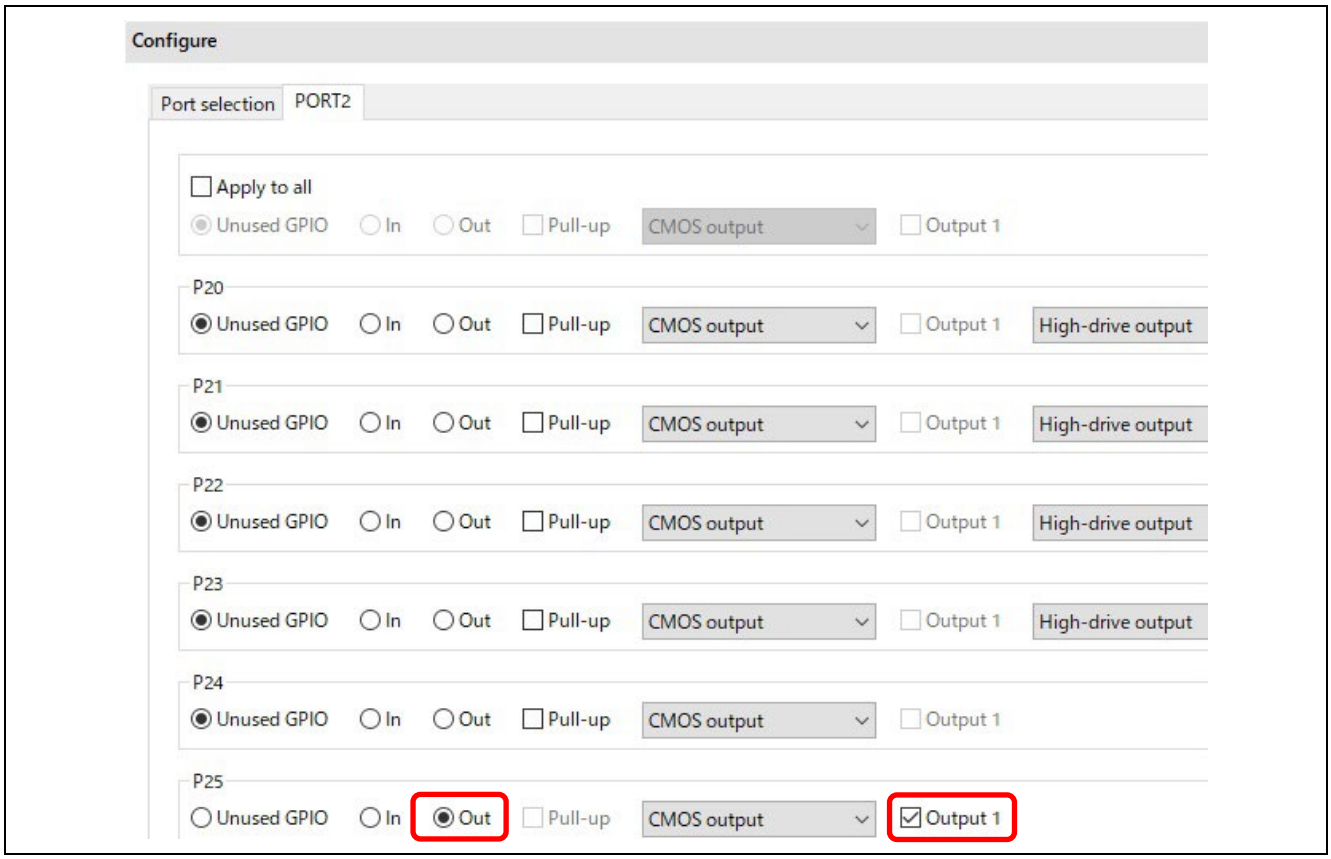


Figure 4-17 Configuring P25

4.4 Generating Source Files

Click on the [Generate Code] button to generate source files.

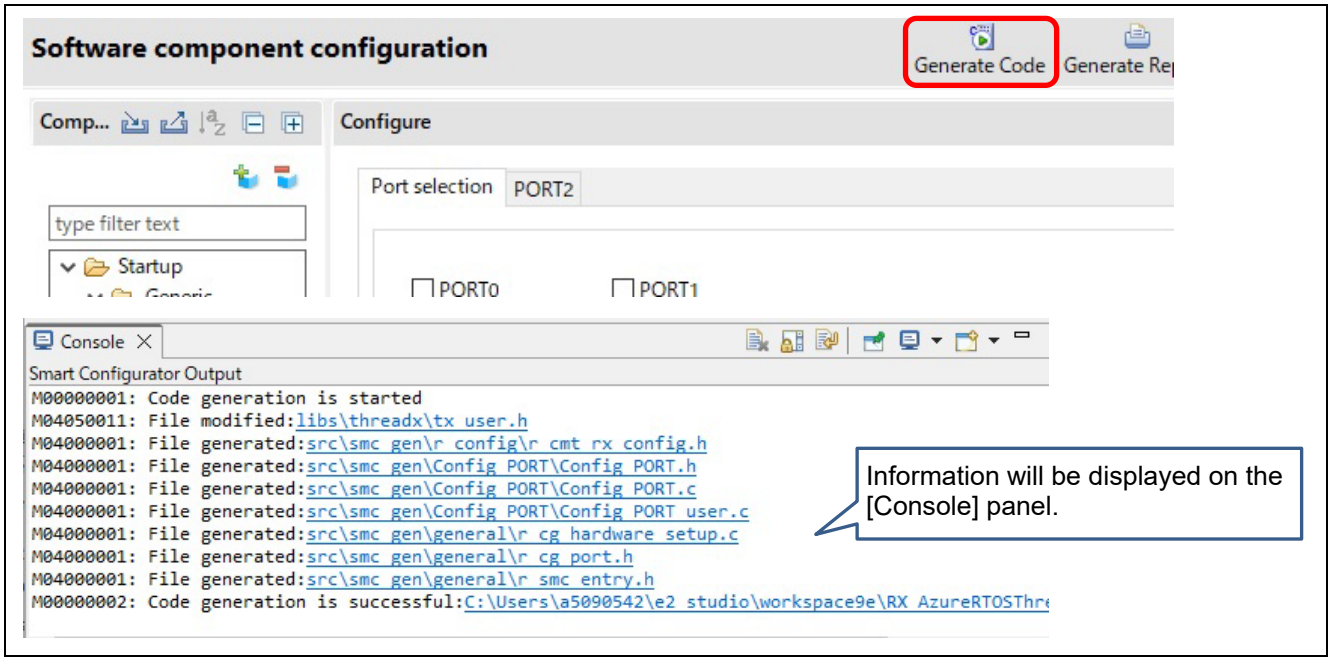


Figure 4-18 Reflecting the Configuration on Source Files

4.5 Editing Source Files

Open "hardware_setup.c" in the project tree.

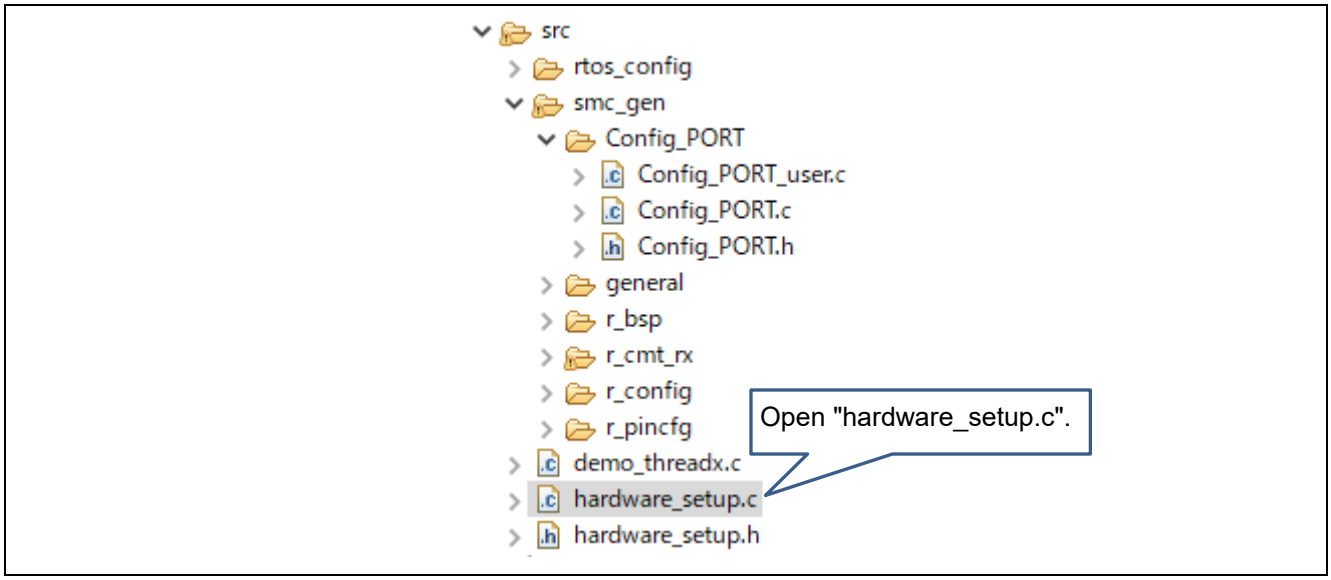


Figure 4-19 Opening a Source File in the Project Tree

Add the lines shown in red boxes below to "hardware_setup.c".

```
#include "r_smc_entry.h"
#include <platform.h>
#include <r_cmt_rx_if.h>
#include "tx_api.h"
#include "hardware_setup.h"

volatile uint8_t gTimer;

void _tx_timer_interrupt(void);

/* CMT Timer callback used as the system tick. */
void timer_callback(void * pdata)
{
    _tx_timer_interrupt();
    gTimer ^= 1;
    PORT2.PODR.BYTE = gTimer * 0x20;
}

void platform_setup(void)
{
    uint32_t chan;
    /* Create periodic timer for the system tick. */
    R_CMT_CreatePeriodic(TX_TIMER_TICKS_PER_SECOND, timer_callback, &chan);
}
```

Repeat switching of the gTimer value between 0 and 1 in the timer interrupt handler processing. This turns the LED connected to P25 on and off.

Edit "tx_api.h".

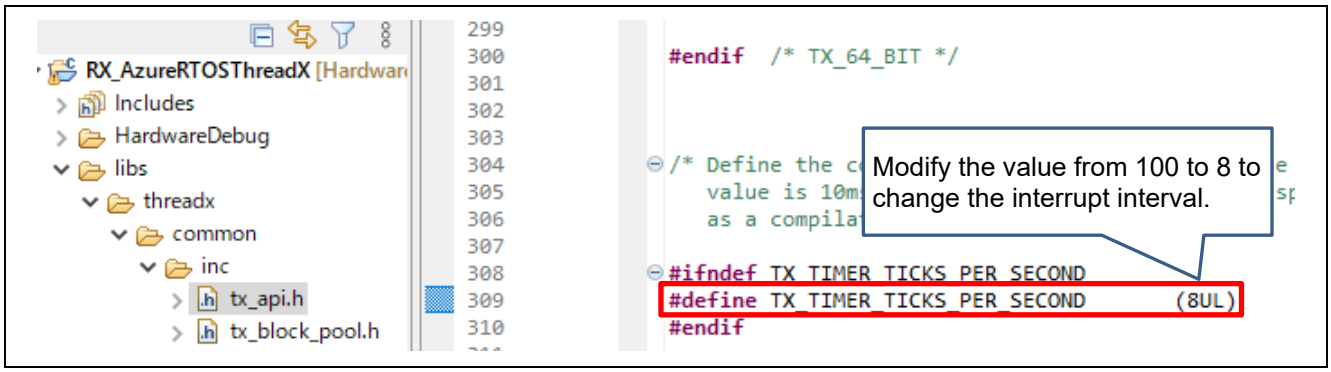


Figure 4-20 Target File and Content of Editing

4.6 Building the Project

Build the project.

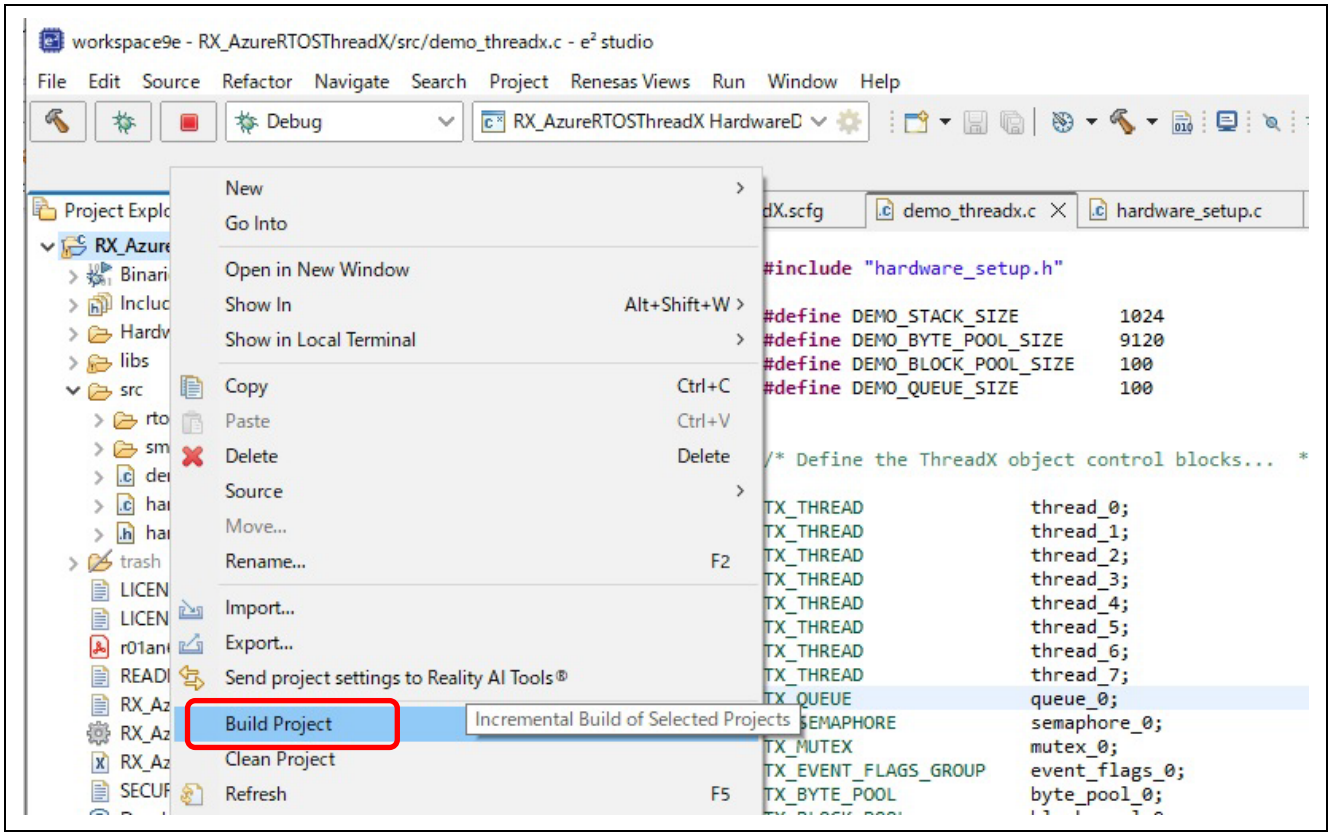


Figure 4-21 Building the Project

On completion of the build process, check that no errors have occurred.

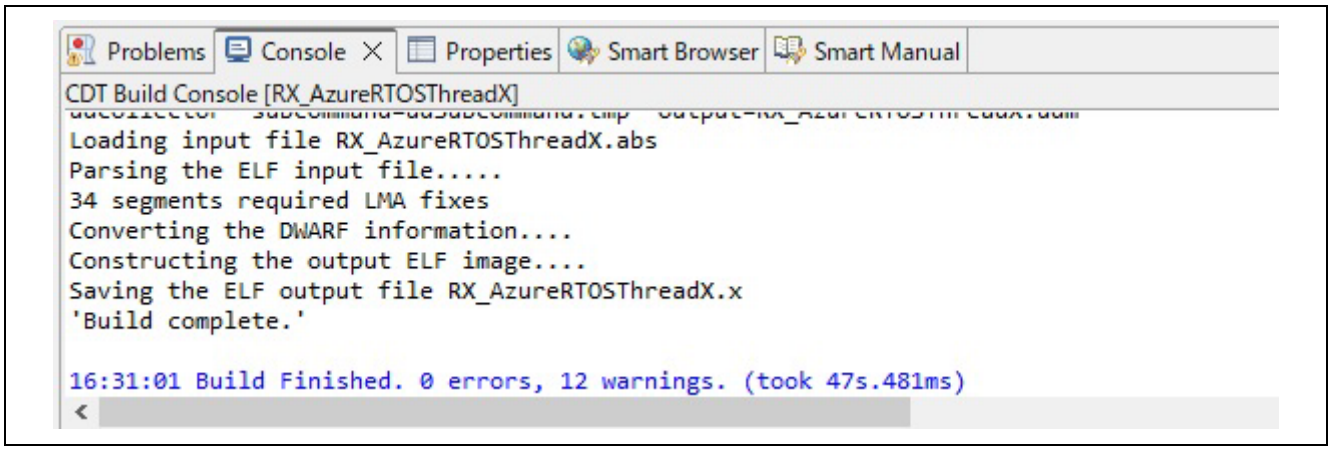


Figure 4-22 Results of Building

4.7 Executing the Project

Select a debug configuration and click on [Debug] to download and execute the program.

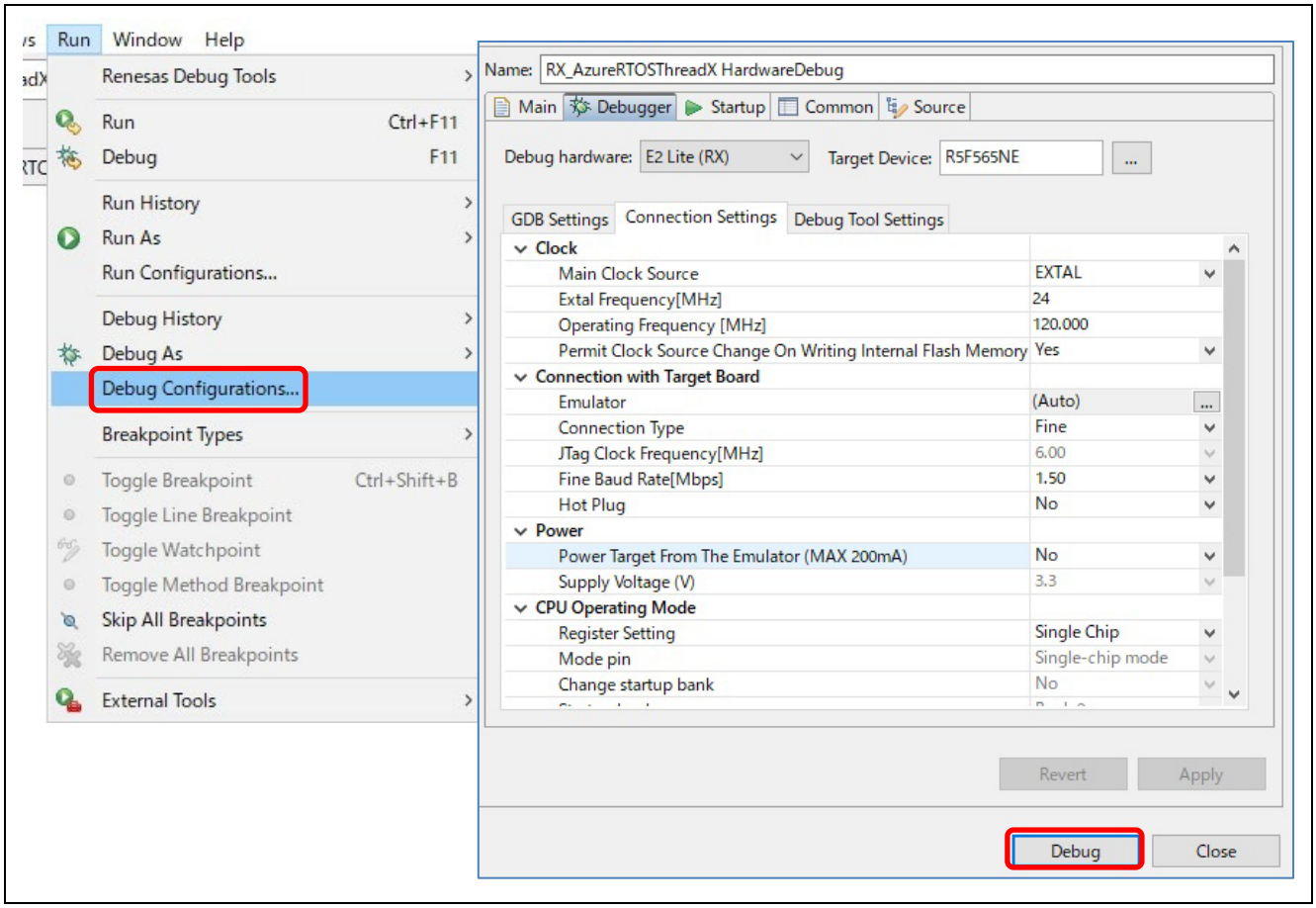


Figure 4-23 Selecting a Debug Configuration and Executing the Program

Figure 4-24 Appearance of the Board during Execution

5. Sample Project

The sample file "demo_threadx.c" contains a standard ThreadX project, which has eight threads.

This sample project file illustrates how to use major services of ThreadX such as threads, a message queue, a timer, a semaphore, a byte memory pool, a block memory pool, an event flag group, and a mutex.

Select [Window] → [Show View] → [Outline] in the e^2 studio.

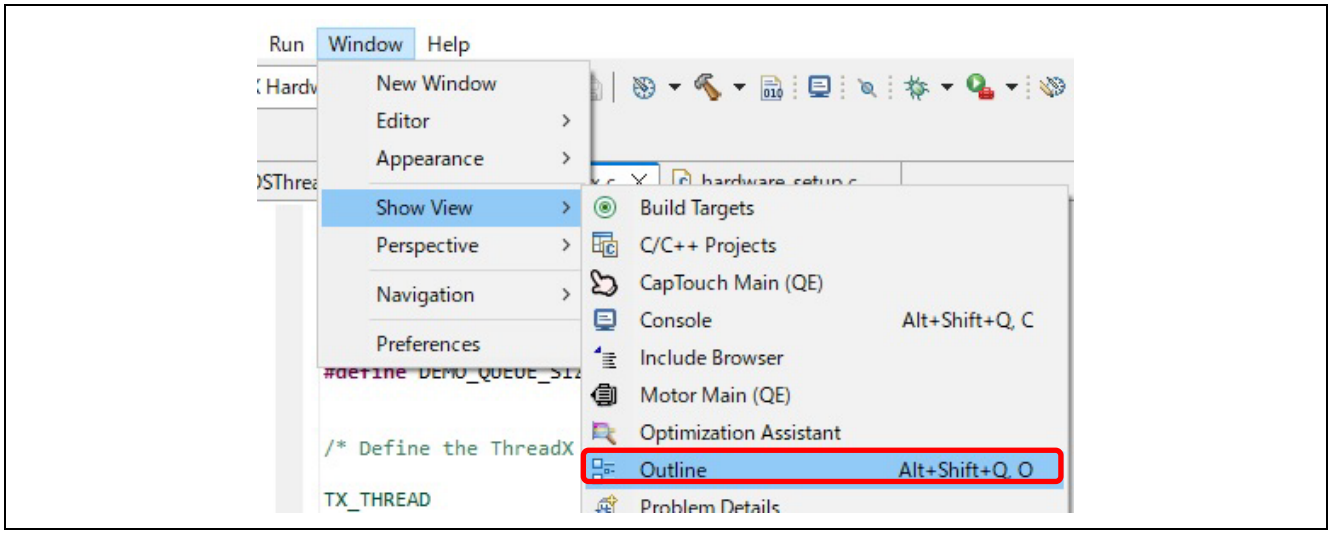


Figure 5-1 Selecting [Outline]

Select a thread in the outline view for "demo_threadx.c" and the display will jump to the function of the selected thread.

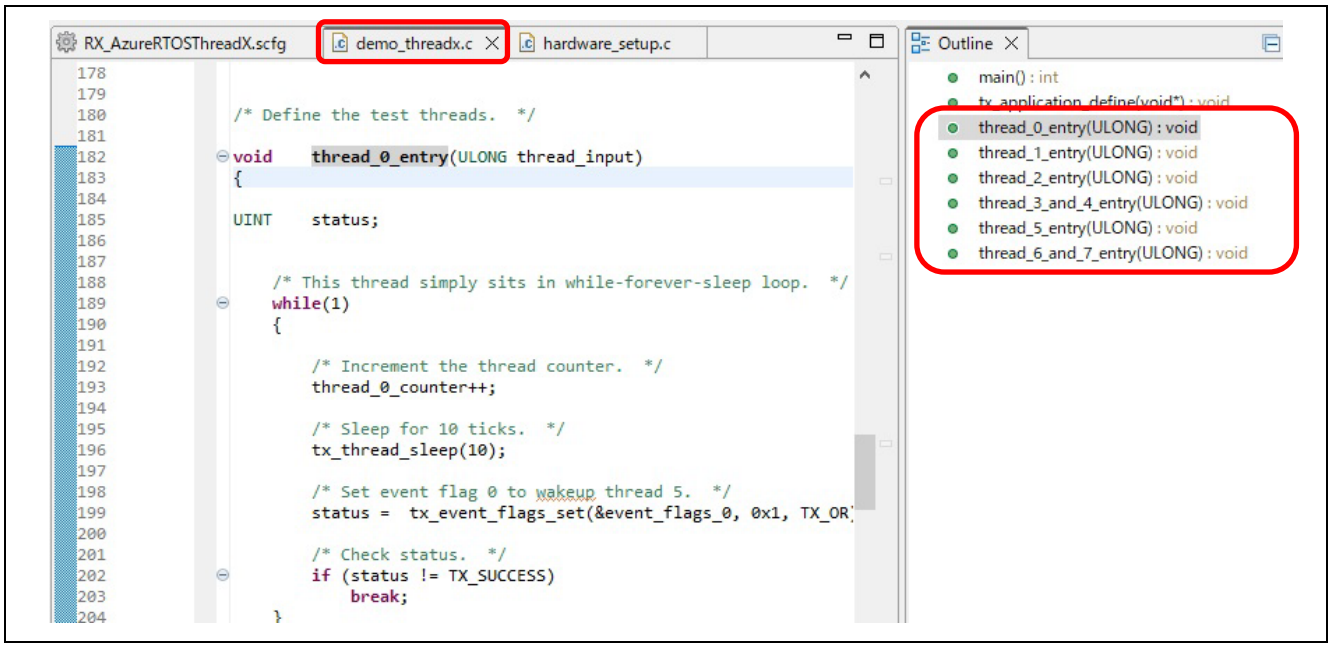


Figure 5-2 A Thread Displayed in the Outline View

The comment at the beginning of each function is a summary of the processing by the thread.
The summaries in the list below have been slightly modified for clarity from the summary comments in the code.

Table 5-1 Summary of the Threads in the Sample Project

Thread Number	Processing by the Thread
0	This thread simply sits in a while-forever-sleep loop.
1	This thread simply sends messages to a queue shared with thread 2.
2	This thread retrieves messages placed on the queue by thread 1.
3	This function is executed as both thread 3 and thread 4. As can be seen from the loop in the code, the threads run as this function alternate between ownership of semaphore_0.
4	
5	This thread simply waits for an event in an endless loop.
6	This function is executed as both thread 6 and thread 7. As can be seen from the loop in the code, the threads run as this function alternate between ownership of mutex_0.
7	

6. Setting the Trace Buffer for Use by TraceX

Azure RTOS TraceX displays the system event information collected by ThreadX that is being executed on the target embedded system.

Enabling analysis by TraceX requires that the program specify a trace buffer for use in analysis.

Use the following API function.

```
tx_trace_enable (VOID *trace_buffer_start, ULONG trace_buffer_size, ULONG registry_entries);
```

This API function enables event tracing in ThreadX. The application is required to specify the size of the trace buffer and the maximum number of ThreadX objects.

Open "demo_threadx.c" and add the following global variable.

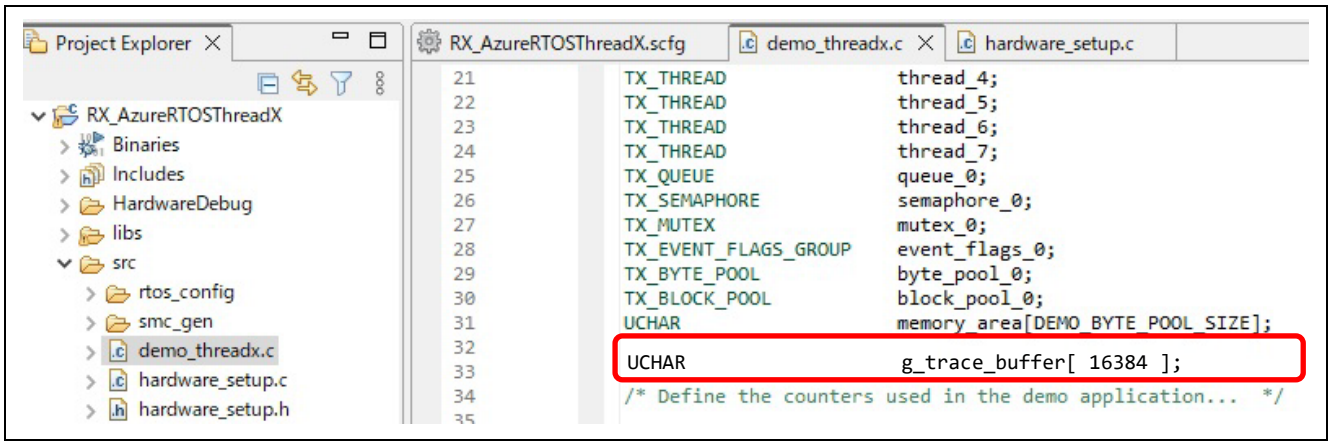


Figure 6-1 Adding a Global Variable

In addition, add the following API function to the tx_application_define function in "demo_threadx.c".

```
void    tx_application_define(void *first_unused_memory)
{
      CHAR *pointer = TX_NULL;

    _tx_trace_enable( &g_trace_buffer, 16384, 30 );
```

Figure 6-2 Adding an API Function

Notes: 1. The buffer size is set to 16384 in this example, but it can be set to a greater value such as 32768 if the RAM of the MCU has enough unused space.
2. Build the project again after the global variable and API function have been added to "demo_threadx.c".

For details, refer to the following page on the Microsoft educational site:

Chapter 5 - Generating trace buffers | Microsoft Learn

7. Using the [RTOS Resources] View

The e^2 studio has an RTOS resource view function that displays the state of Azure RTOS ThreadX resources. The following is a description of the procedure for using the [RTOS Resources] view.

7.1 Displaying the [RTOS Resources] View

The [RTOS Resources] view is only available while the debugger is running. Start the debugger and then select [Renesas Views] → [Partner OS] → [RTOS Resources]. After the [Select OS] dialog box is displayed, select "ThreadX" as shown in Figure 7-2. The [RTOS Resources] view will appear as shown in the figure.

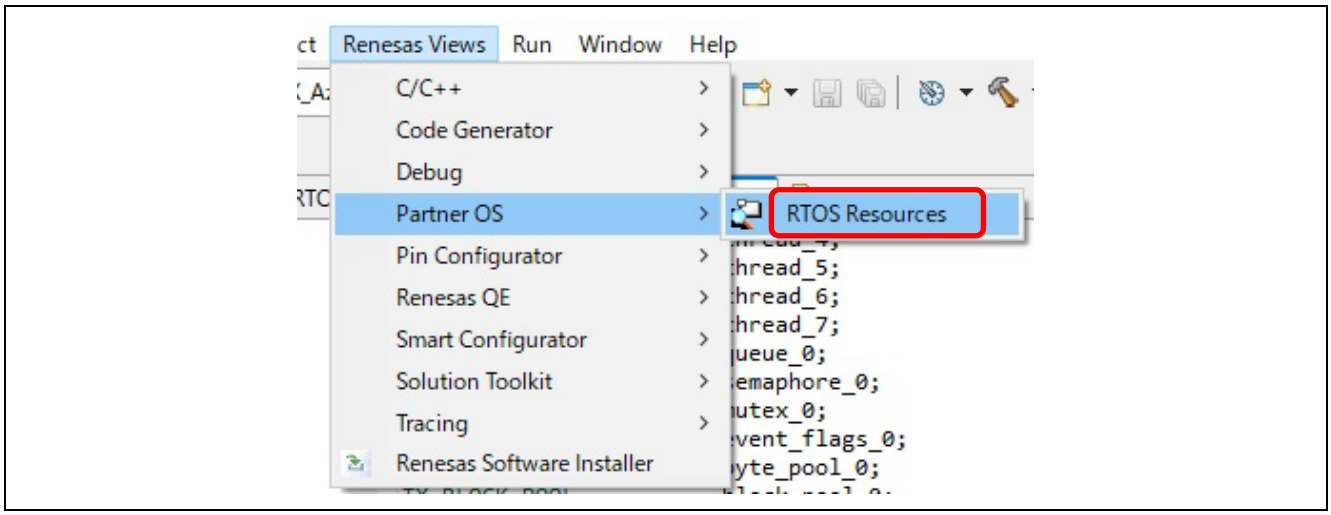


Figure 7-1 Selecting [RTOS Resources]

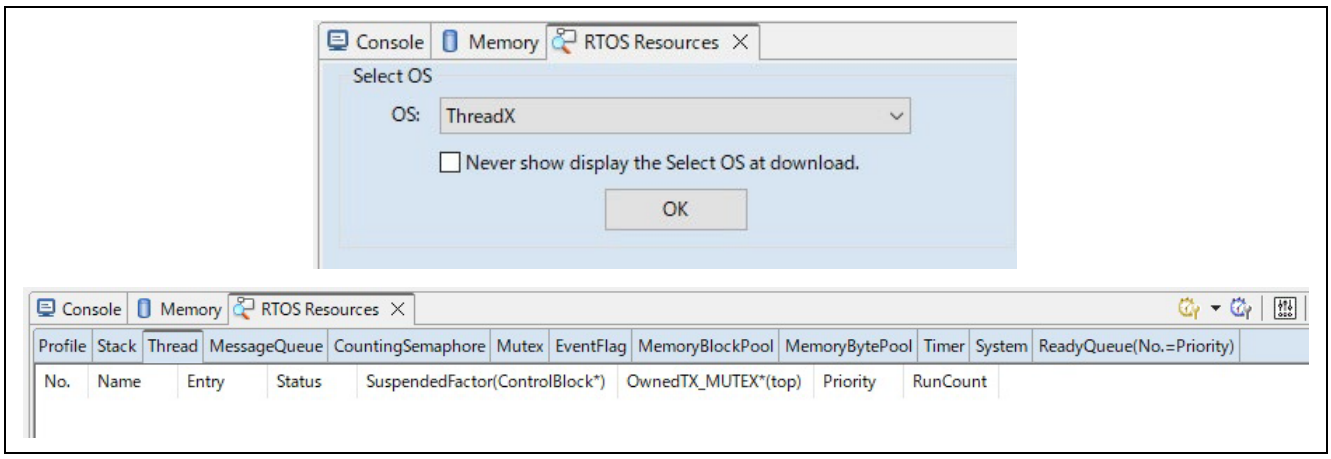


Figure 7-2 Selecting the OS and Displaying the [RTOS Resources] View

7.2 Context Menu

Display the context menu by right-clicking on the mouse with the cursor in the [RTOS Resources] view.

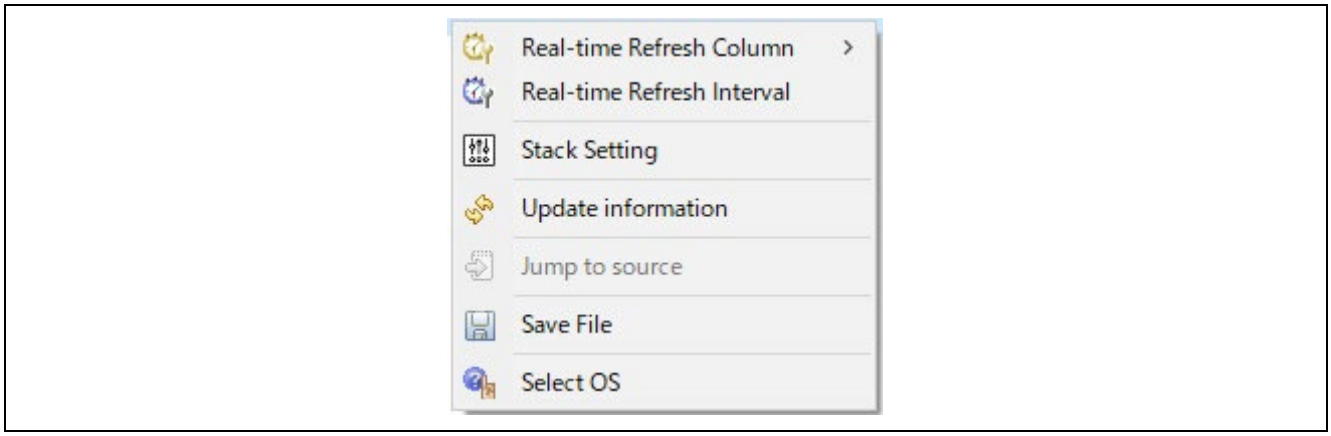


Figure 7-3 Context Menu

- **Real-time Refresh Column:**
  Enables or disables real-time updating of information displayed in the individual columns (tabbed pages).
  This is grayed out and not selectable while the program is running.
- **Real-time Refresh Interval:**
  Specifies the interval for real-time updating of the display. The specifiable values are in the range from 500 ms to 10000 ms.
  This is grayed out and not selectable while the program is running.
- **Stack Setting:**
  Enables or disables loading of the stack data and specifies the threshold for the stack warning function.
  This is grayed out and not selectable while the program is running.
- **Update information:**
  Updates the displayed information.
- **Jump to source:**
  Opens an editor view displaying the source code of the task/thread or handler. Double-clicking on a task/thread or a handler also opens an editor view.
  This is grayed out and not selectable while the program is running.
- **Save File:**
  Saves the data on the currently selected tabbed page in a text file (*.txt).
  This is grayed out and not selectable while the program is running.
- **Select OS:**
  Opens the [Select OS] dialog box.
  This is grayed out and not selectable while the program is running.

7.3 Stack Setting

This is for enabling the loading of stack data and setting the stack threshold.

a. Open the context menu and select [Stack Setting].
b. To load stack data to the [RTOS Resources] view, check the [Enable loading Stack data] checkbox in the [Stack Setting] dialog box. If this option is not enabled, stack data will not be loaded in the next debugging session.
c. A desired threshold value can be set in the [Stack Threshold (%)] textbox. Click on [OK] to save the setting. The value is set to 80% in this example.

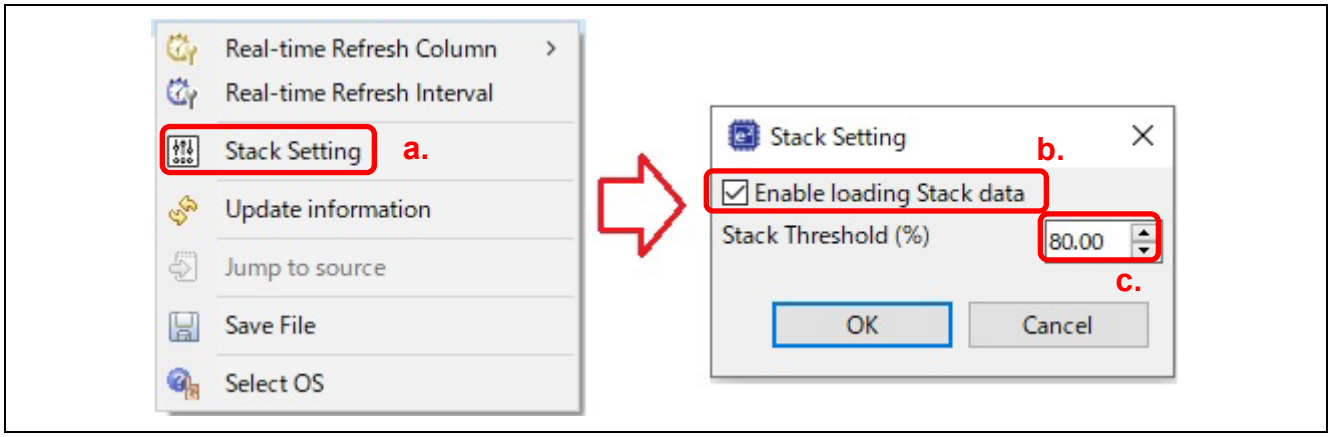


Figure 7-4 Settings for the Stack

7.4 Tabbed Pages

Table 7-1 lists the items displayed on the individual tabbed pages.

Table 7-1 Contents of Individual Tabbed Pages

Name of the Tabbed Page in the [RTOS Resources] View	Name of Displayed Information and Selection	Information to be Displayed
Profile	The facilities of the [Profile] tabbed page are not currently available.	—
Stack	Name	Names of the threads
	Entry	Function that started each thread
	StackPointer	Current stack pointer
	StackStart	Address of the beginning of the stack
	StackEnd	Address of the end of the stack
	StackSize(bytes)	Stack size
	MaxStackUsage(bytes)	Maximum amount of stack currently in use
Thread	Name	Names of the threads
	Entry	Function that started the given thread
	Status	State of each thread
	Suspended Factor (Control Block*)	Resource that is the source of suspension
	OwnedTX_MUTEX*(top)	Top acquired mutex
	Priority	Priority
	RunCount	Number of times the thread has been executed
MessageQueue	Name	Names of the message queues
	UsedCount	Number of message queues in use
	FreeCount	Number of available message queues
	TotalCount	Total number of message queues
	MessageSize	Message size
	SuspendedTX_THREAD*(top)	Top waiting thread in the queue
	SuspendedCount	Number of suspended threads
	StartAddress	Address where the message queue starts
	EndAddress	Address where the message queue ends
CountingSemaphore	Name	Names of the semaphores
	SemaphoreCount	Number of semaphores
	SuspendedTX_THREAD*(top)	Top waiting thread in the queue
	SuspendedCount	Number of suspended threads
Mutex	Name	Names of the mutexes
	OwnerTX_THREAD*	Acquiring thread
	OwnerCount	Number of owners
	SuspendedTX_THREAD*(top)	Top waiting thread in the queue
	SuspendedCount	Number of suspended threads
EventFlag	Name	Names of the event flags
	Flag	Current flag pattern
	SuspendedTX_THREAD*(top)	Top waiting thread in the queue
	SuspendedCount	Number of suspended threads

MemoryBlockPool	Name	Names of the memory blocks
	FreeCount	Number of available blocks
	TotalCount	Total number of blocks
	BlockSize(bytes)	Block size
	TotalSize(bytes)	Total size of memory block pools
	SuspendedTX_THREAD*(top)	Top waiting thread in the queue
	SuspendedCount	Number of suspended threads
	StartAddress	Top address of a memory block pool
MemoryBytePool	Name	Names of the memory pools
	Free(bytes)	Number of available bytes
	Total(bytes)	Total size of memory byte pools
	FragmentCount	Number of fragments
	SuspendedTX_THREAD*(top)	Top waiting thread in the queue
	SuspendedCount	Number of suspended threads
	StartAddress	Address where the memory byte pool starts
Timer	Name	Names of the timers
	Remaining Tick	Remaining time in ticks
	Re-initialization Tick	Cycle time in ticks
System	SystemClock	System clock
ReadyQueue(No. = Priority)	QueuedTX_THREAD*(top)	Top ready thread

8. Starting Debugging of a Project with Azure RTOS TraceX

Select the [Run] menu → [Debug] to launch the debugger.

Create a data file for Azure RTOS TraceX by following the procedure below.

8.1 Acquiring the Address of the TraceX Data Buffer

Use the [Add Watch Expression] menu to add “g_trace_buffer” in “demo_threadx.c” to the expressions to be watched and acquire the address of the data buffer for Azure RTOS TraceX. The address is "0x736c" in the following example.

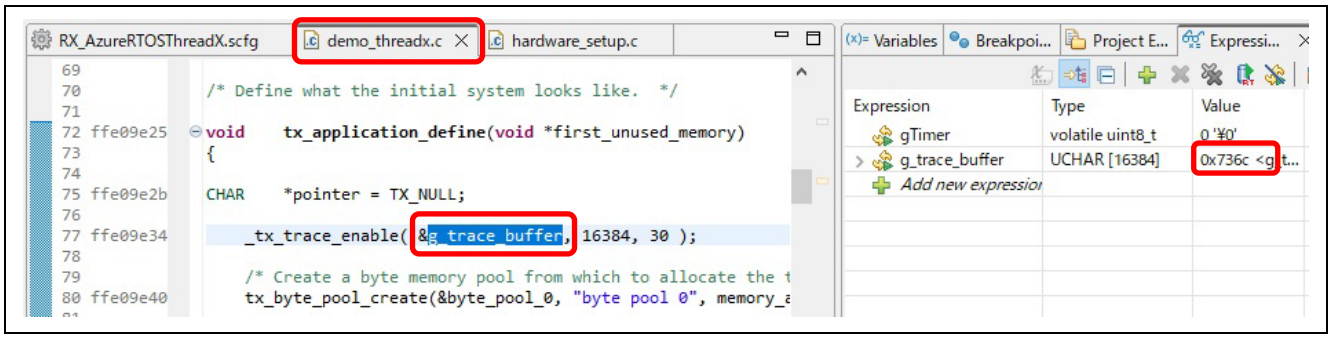


Figure 8-1 Acquiring the Address of the Data Buffer

8.2 Exporting a Data File for TraceX

Export a data file for TraceX by following the procedure below.

Execute the program with the button and then stop it with the button.

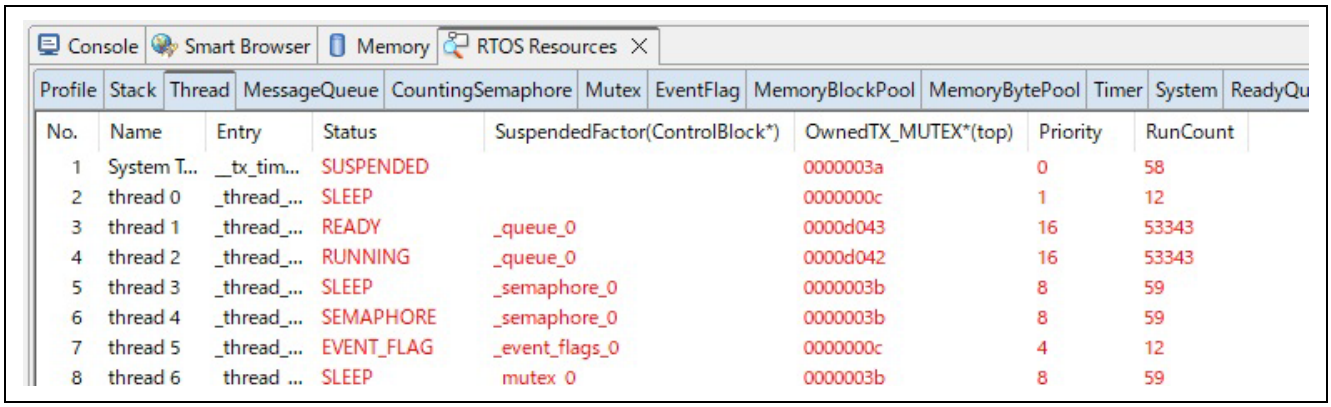


Figure 8-2 Information Displayed on the [Thread] Tabbed Page

a. Select [Window] → [Show View] → [Memory].
b. Open the [Monitor Memory] dialog box and enter the address of the data buffer for Azure RTOS TraceX that was acquired in section 8.1.

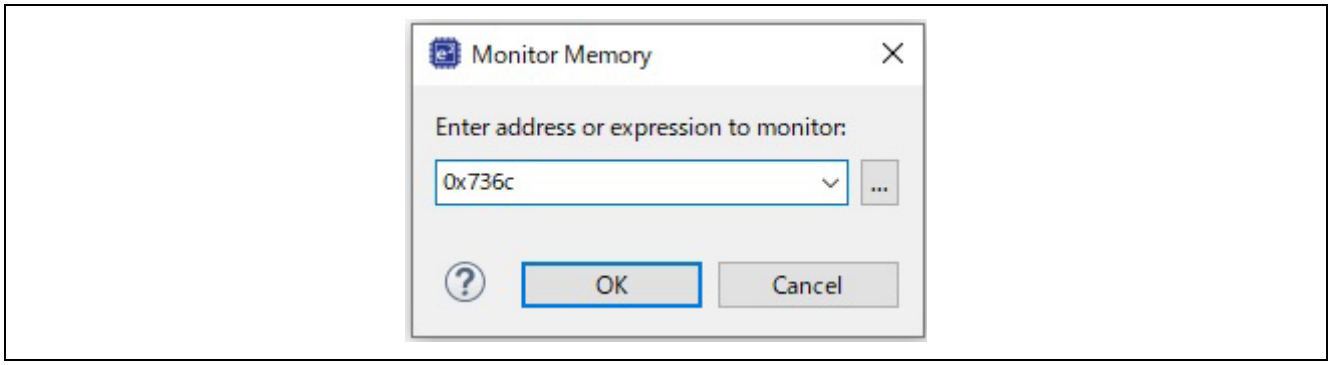


Figure 8-3 Entering the Address for Monitoring

c. Press the [Export] button to the right of the [Memory] tab.
d. The [Export Memory] dialog box will pop up.
d. For [Format], select "RAW Binary".
f. For [Start address], enter the address of the TraceX data buffer that was acquired in section 8.1.
g. For [Length], enter 16384* as the size of the TraceX data buffer, "g_trace_buffer".
h. For [File name], create a desired file name with the extension ".trx".
i. Press [OK]. The data file for TraceX is then exported with the file name specified in (h).

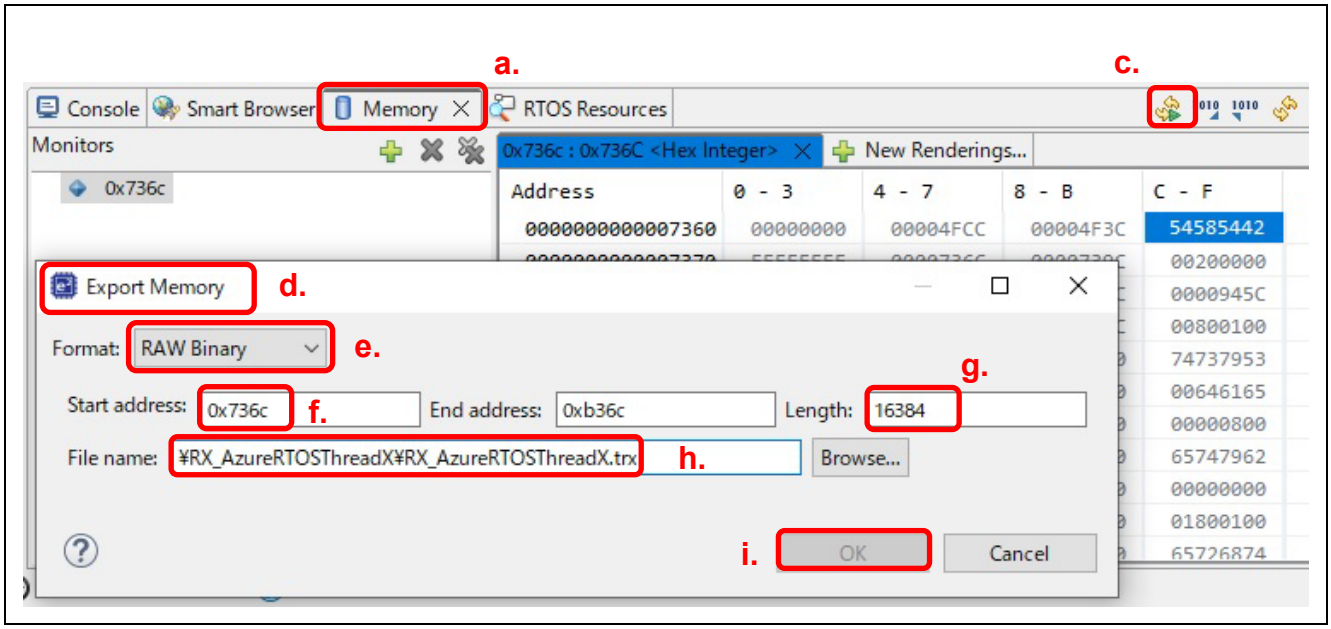


Figure 8-4 Exporting a Data File

Note: * Enter the size specified in section 6, Setting the Trace Buffer for Use by TraceX.

9. Launching Azure RTOS TraceX

Launch Azure RTOS TraceX that has been installed on the host PC.

Click on the [File] menu → [Open] for Azure RTOS TraceX.

Select the TraceX data file "****.trx" exported according to the steps in section 8.2.

Trace information will be shown in TraceX.

9.1 Display of Trace Information

Trace information can be displayed in either of the following two ways.

- Sequential view mode
- Time view mode

9.1.1 Sequential View Mode

The sequential view shows events immediately followed by others without regard to the elapsed times between events. This mode is useful to get an overview of the system operation.

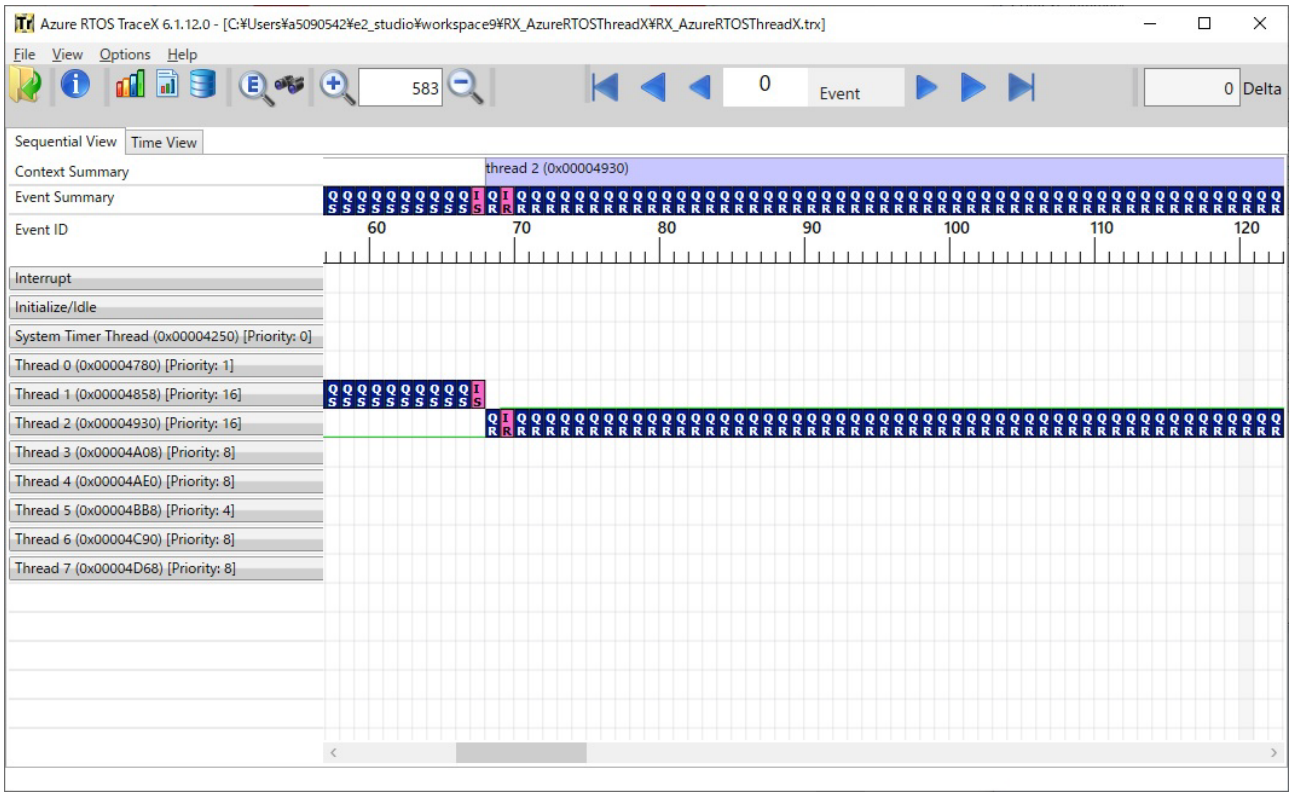


Figure 9-1 Display in the Sequential View

9.1.2 Time View Mode

This mode is useful to find the locations where the bulk of processing takes place in the system and can help the user to tune the system for better performance or responsiveness.

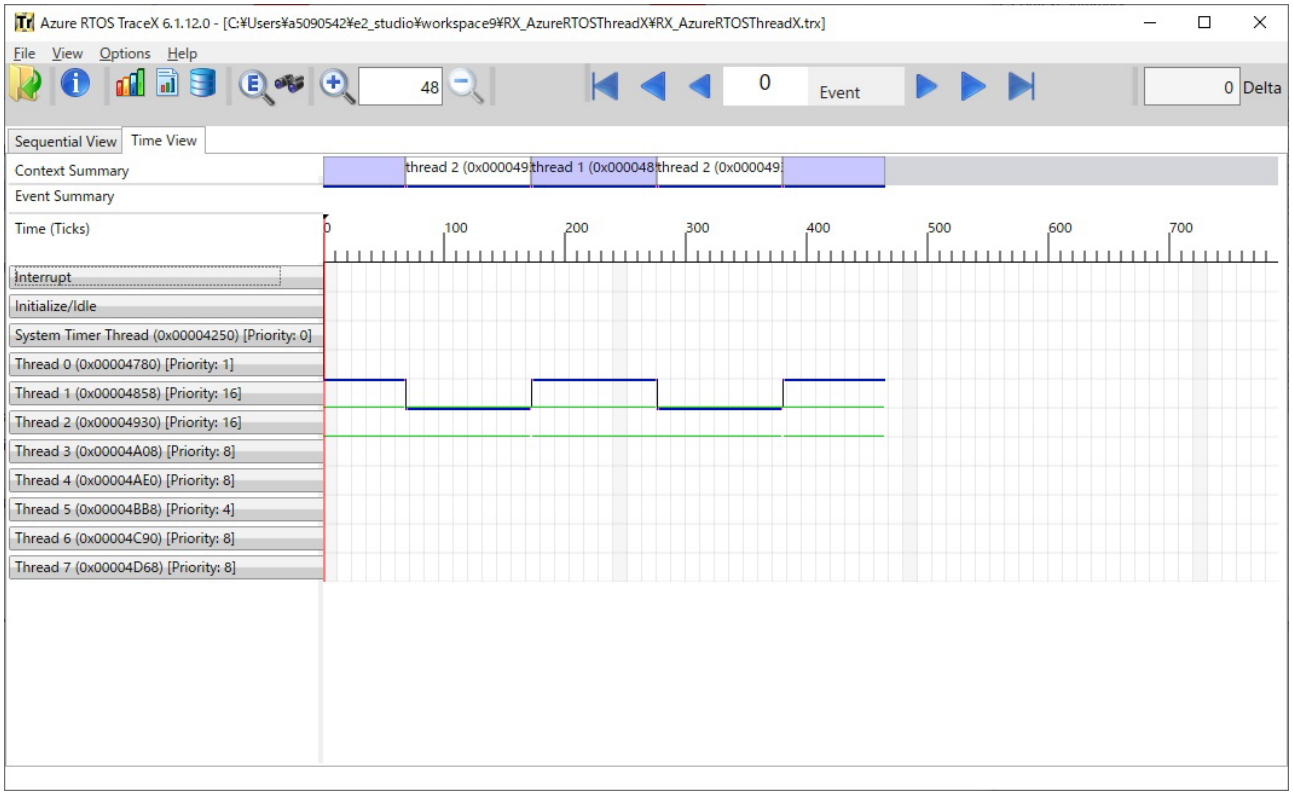


Figure 9-2 Display in the Time View

9.2 Analysis Facilities of TraceX

For details of the analysis facilities, see [Help] → [Manual].

Refer to the following descriptions in "Chapter 3 – Description of Azure RTOS TraceX" of the *TraceX User Guide* | Microsoft Docs.

- Sequential View Mode
- Time View Mode

Also refer to the following descriptions in "Chapter 4 - Azure RTOS TraceX performance analysis" of the *TraceX User Guide* | Microsoft Docs.

- Execution Profile
- Popular Services
- Thread Stack Usage
- Performance Statistics
- FileX Statistics
- NetX Statistics
- Trace File Information

Website and Support

Visit the following URLs to learn about key elements of the RX family, download components, and related documentation, and get support.

RX Family Product Information	www.renesas.com/rx
RX Family Product Support Forum	www.renesas.com/rx/forum
Renesas Support	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar. 01, 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/.