Renesas RA Family

Quick Start Guide: Modbus Serial

Introduction

This document is a quick start guide for evaluating Modbus® communication with the RA microcomputer evaluation board.
Modbus protocol is a communication protocol developed by Modicon Inc. (Schneider Electric SA.) for programmable logic controllers (PLCs), and its specifications are open to the public.
For details, refer to the protocol specifications (PI-MBUS-300 Rev.J).

Target Device

RA8T2, RA8M2

Supported Evaluation Boards

MCK-RA8T2, EK-RA8M2, EK-RA8T2

Contents

1. Overview

This document is for the Modbus protocol stack that operates on the RA board, and describes the function overview, application programming interface (API), and application samples for developing and implementing applications using the protocol stack.

This package supports the RS-485 based Modbus RTU/ASCII protocol.

1.1 Abbreviations / Definitions

Table 1-1 Abbreviations/Definitions

Index	Abbreviations /Definitions	Description
1	USB	Universal Serial Bus
2	PC	Personal Computer
3	SW	Switch
4	EWARM	Embedded Workbench® for ARM
5	LED	Light Emitting Diode

1.2 Reference

For technical information about Modbus, please visit the Modbus organization site. Information about RA evaluation board is available through Renesas.

Table 1-2 Technical Inputs

Index	Technical Inputs
1	Modbus Application Protocol Specification Vxxx
2	MCK-RA8T2 Quick Start Guide / r12qs0088ejxxxx
3	MCK-RA8T2 User's Manual / r12uz0172ejxxxx
4	Evaluation Kit for RA8M2 Microcontroller Group EK-RA8M2 Quick Start Guide / r20qs0069egxxxx
5	Evaluation Kit for RA8M2 Microcontroller Group EK-RA8M2 v1 User’s Manual / 20ut5451egxxxx
6	Evaluation Kit for RA8T2 Microcontroller Group EK-RA8T2 v1 Quick Start Guide / r20qs0097egxxxx
7	Evaluation Kit for RA8T2 Microcontroller Group EK-RA8T2 v1 User’s Manual / r20ut5714egxxxx

2. Features

The Modbus protocol stack for RA allows for quick and easy development of the Modbus RTU/ASCII applications. The following nine codes can be implemented in this stack.

1. (0x01) - Read coils
2. (0x02) - Read discrete input
3. (0x03) - Read holding registers
4. (0x04) - Read input registers
5. (0x05) - Write single coil
6. (0x06) - Write single register
7. (0x0F) - Write multiple coils
8. (0x10) - Write multiple registers
9. (0x17) - Read/Write multiple registers

For more information about Modbus, refer to the following site:
http://www.modbus.org

Note: The version number may differ depending on the update. Refer to the latest manual.

3. Sample Program Package Configuration

This sample program package consists of three components:

- Modbus sample project using Modbus protocol stack
- Modbus sample application using Modbus protocol stack
- Modbus sample demo application

3.1 Modbus Sample Project

- Modbus_Serial / project / MCK-RA8T2
- Modbus_Serial / project / EK-RA8M2
- Modbus_Serial / project / EK-RA8T2
  - These folders contain a folder called “RA_Modbus” which includes a Modbus sample project for MCK-RA8T2 / EK-RA8M2 / EK-RA8T2 boards that uses the Modbus protocol stack.
  - The sample project is created for GCC compilers. If you want to use another compiler, refer to section "**5.2 Creation of new Modbus Project**" and create another project.

3.2 Modbus Sample Application

- Modbus_Serial / src / modbus_func.c, modbus_user.c, new_thread0_entry.c
  - Users can register their own implementations of Modbus function codes in the Modbus protocol stack.
  - The code in this directory provides examples of the Modbus protocol stack initialization process and the Modbus function codes processing using the Modbus protocol stack API.

3.3 Modbus Demo Application

- Modbus_tool / ModbusDemoApplication.exe
  - This executable file is Modbus demo application used for Modbus communication. It can be used to demonstrate the operation of Modbus sample application.

4. Project Setup

4.1 Operating Environment Requirements

The sample program package described in this manual runs in the following environment.

Table 4-1 Operating environment

Item	Description
Board	RA evaluation board
Integrated development environment	IAR Systems - IAR Embedded Workbench® for Arm Version 9.70.2 or later Renesas Electronics - e² Studio 2025-12 or later - Renesas RA Smart Configurator 2025-12 or later
Toolchain	IAR Embedded Workbench for Arm - IAR C/C++ Compiler for Arm 9.70.2 or later e² Studio - GCC Arm Embedded (13.2.1.arm-13-7) or later - LLVM for Arm (21.1.1) or later - Arm Compiler 6.24 or later
MCU software package	FSP (Flexible Software Package) v6.4.0 or later
Emulator	J-LINK® OB
Communications protocol	Modbus RTU or Modbus ASCII
Client tool	ModbusDemoApplication.exe: Modbus Demo Application

4.2 Setting the Board

Connect the PC to a supported evaluation board.
Power is supplied by connecting a USB cable to the board.
For Modbus serial communication, use RS485 connector and connect to PC with RS-485 cable.
The MCK-RA8T2 board connection setup diagram, EK-RA8M2 board connection setup diagram, EK-RA8T2 board connection setup diagram and Switch SW4 setting table are described below.

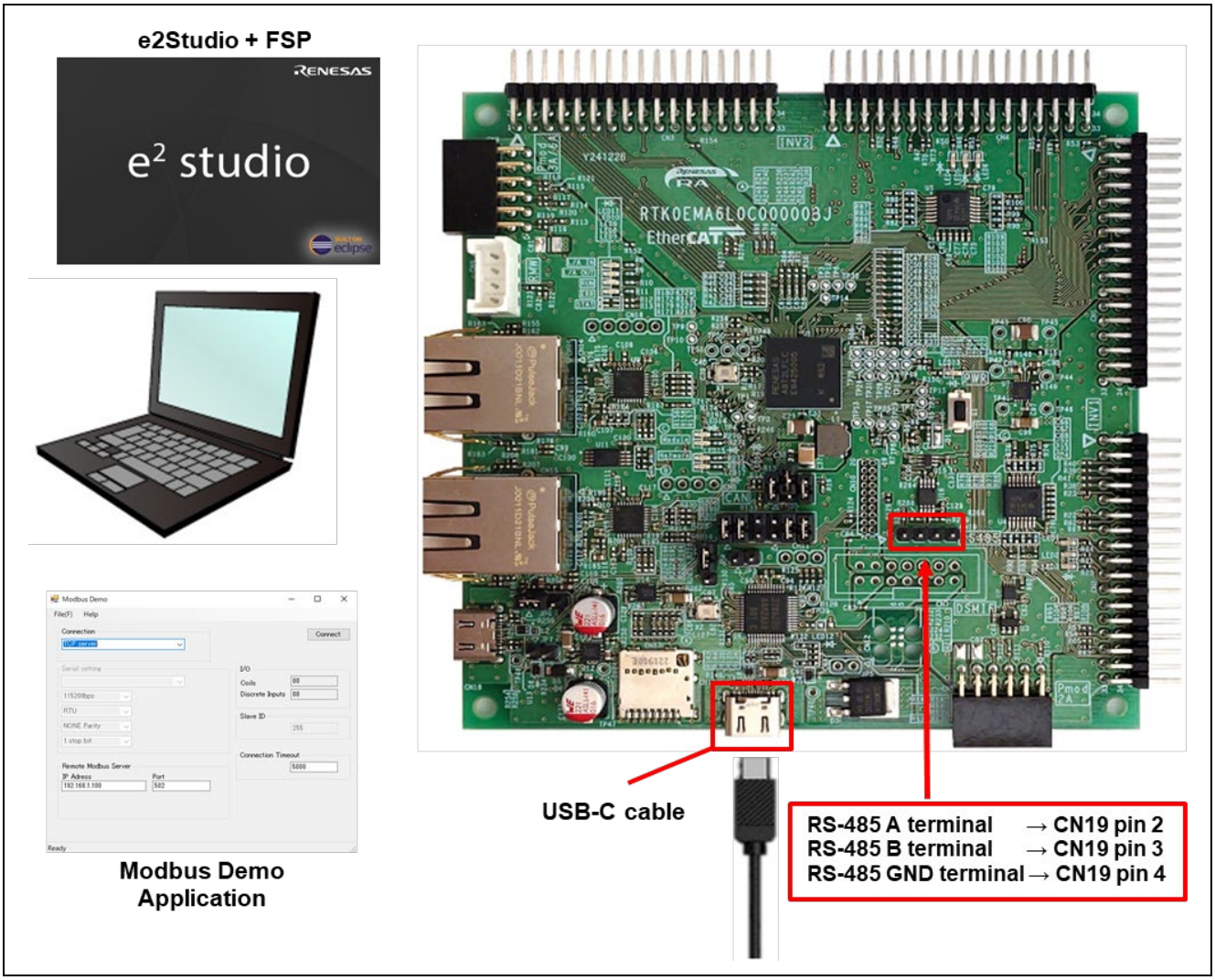


Figure 4.1: MCK-RA8T2 board connection setup

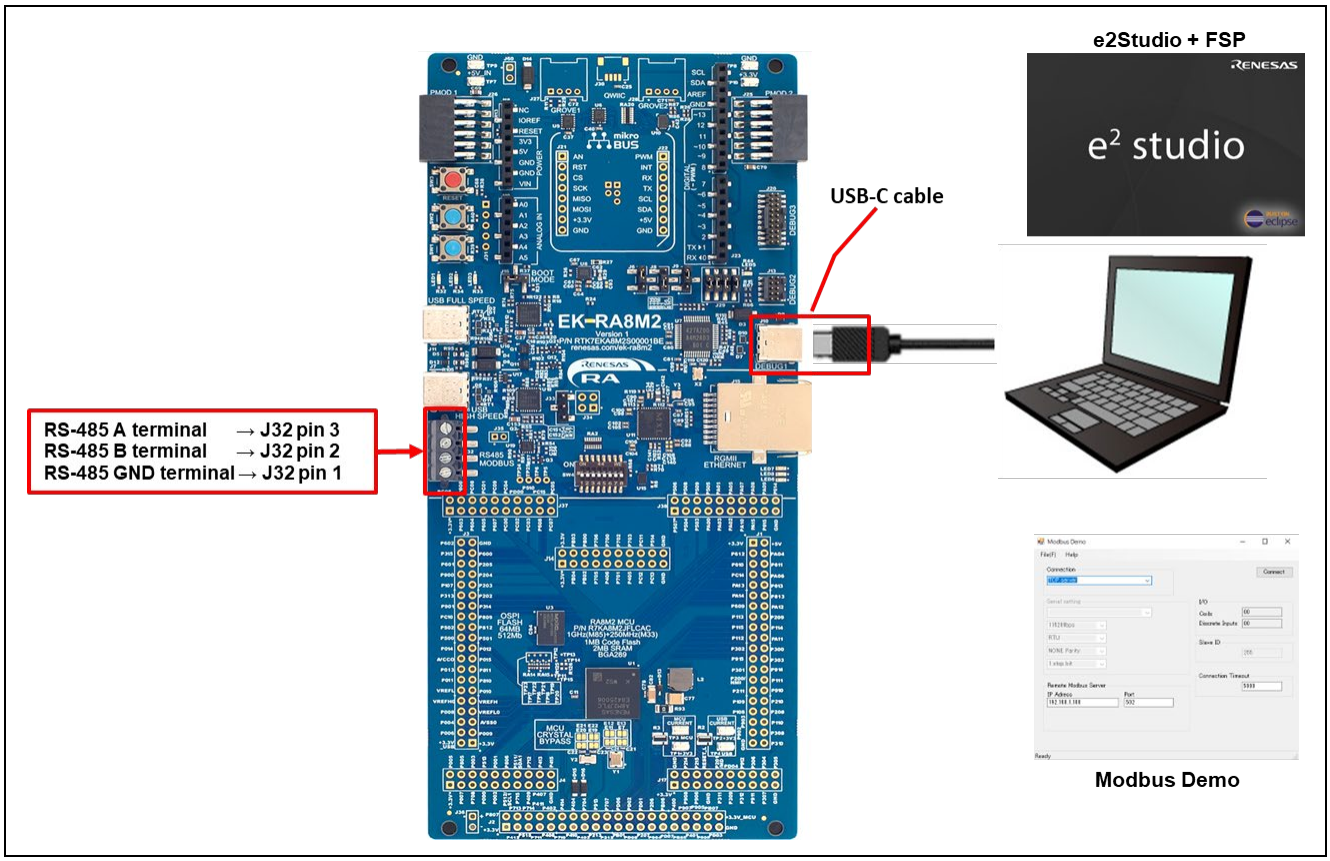


Figure 4.2: EK-RA8M2 board connection setup

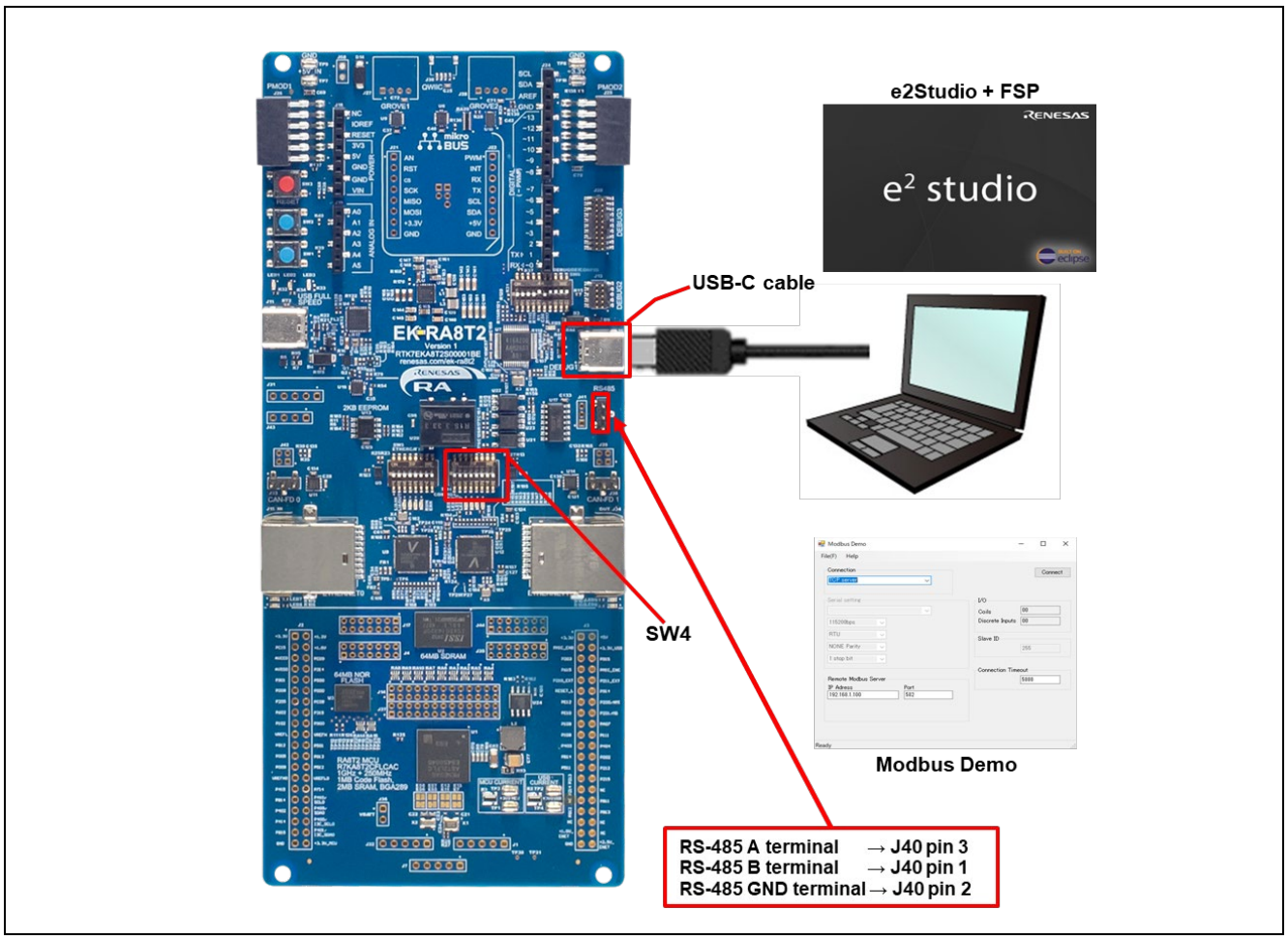


Figure 4.3: EK-RA8T2 board connection setup

Table 4.1 Switch SW4 setting

SW4	Setting	Description
SW4-8	OFF	**RS485 enabled**

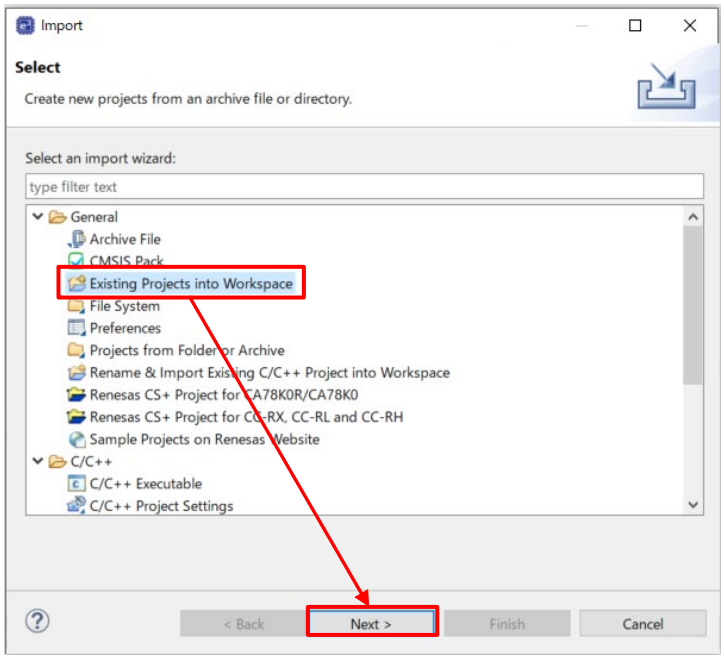
5. Setting Up the Modbus Sample Project

This section describes the procedure for importing/creating a Modbus sample project. Before this, read section "**4.1 Operating Environment Requirements**" first and complete the installation of the tools.

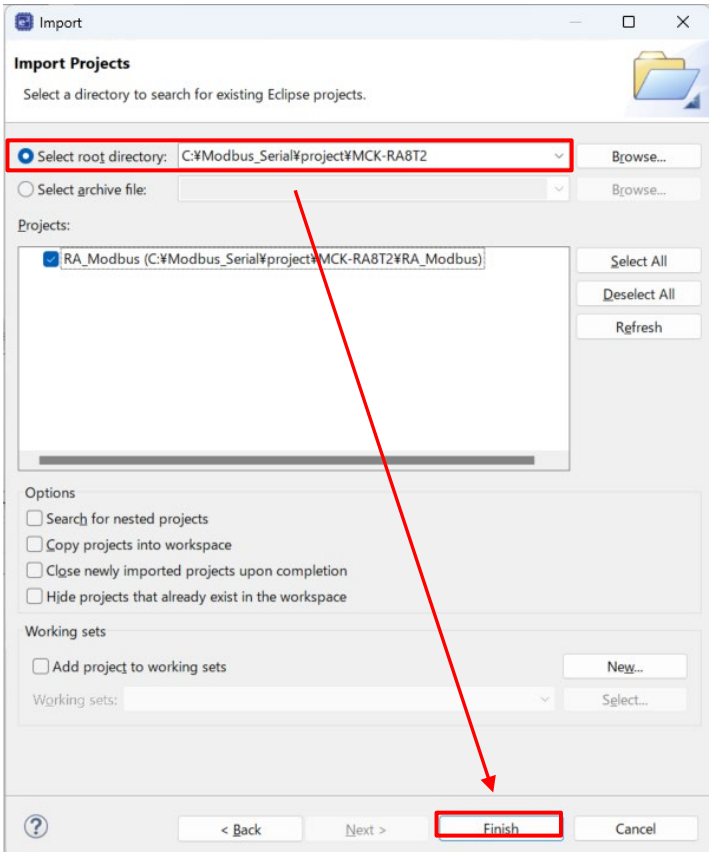
5.1 Import Modbus Sample Project

This section describes the procedure for importing the Modbus sample project and changing the evaluation board.

1. Import the sample project. Start e2 studio, select [File] → [Import], and when the Import window appears, select [General] → [Existing Projects into Workspace].



Check "Select root directory" and select the Modbus sample project folder (“Modbus_Serial/project/[evaluation board]”).



2. Execute the Modbus Sample Project.
   Refer to "**6. Execution of Modbus Sample Project**" and follow the steps.

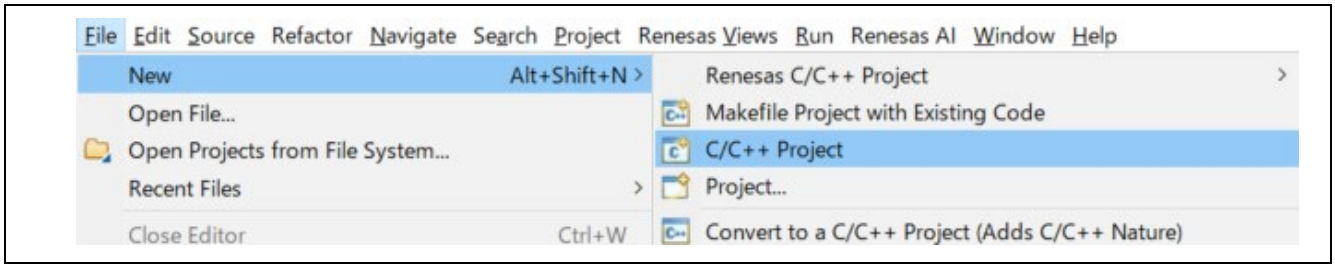
5.2 Creation of New Modbus Project

This section describes the procedure for creating a new Modbus sample project.

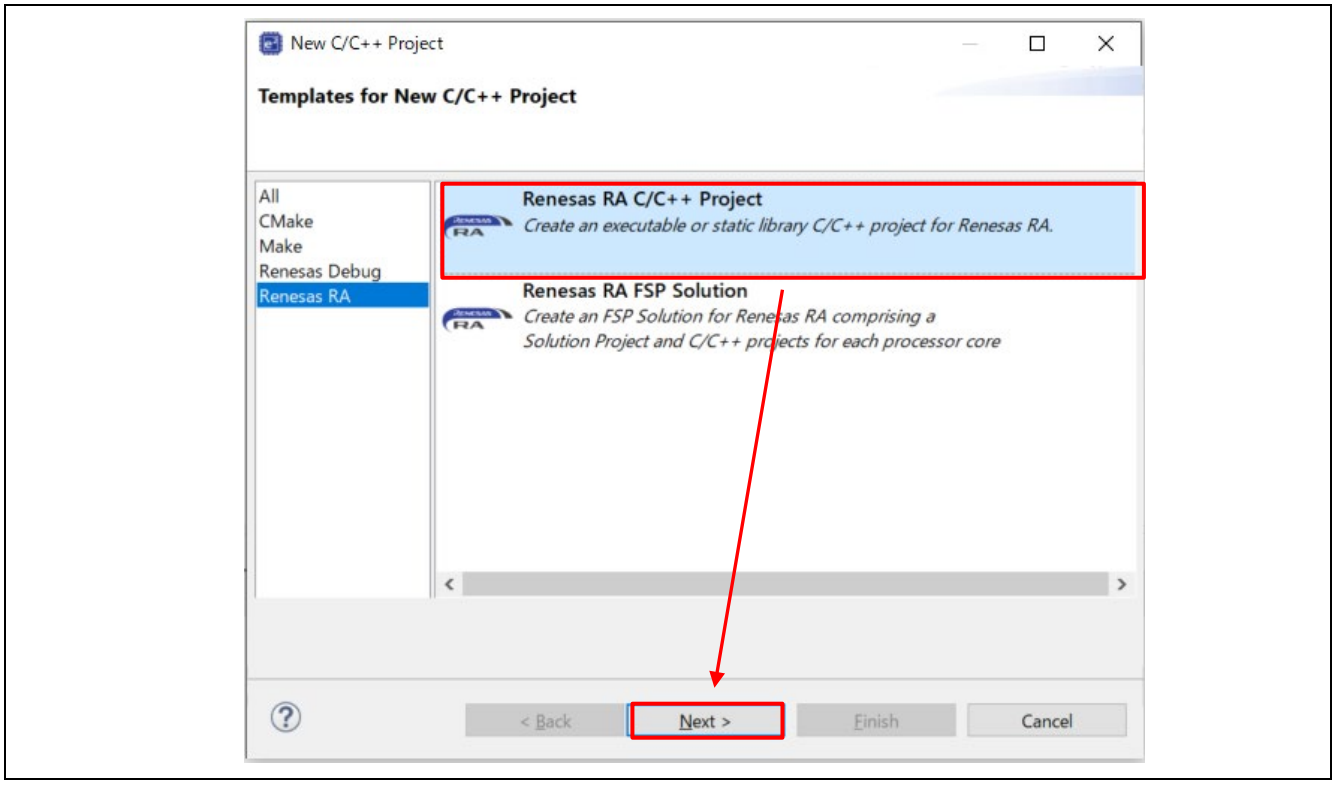
5.2.1 Common Procedures for all Evaluation Boards

This section describes the procedures common to all evaluation boards.

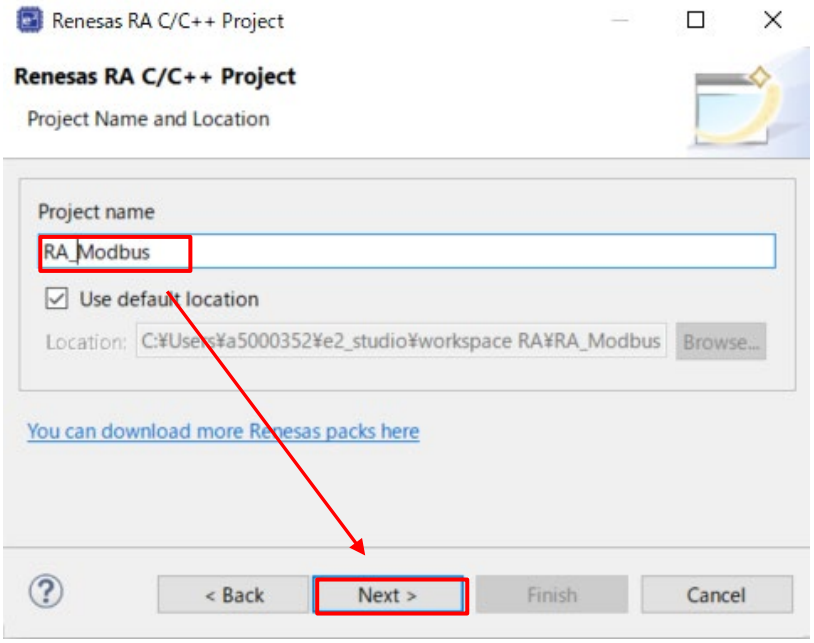
1. Set up a new project. Start e2 studio and select [File] → [New] → [C/C++ Project].



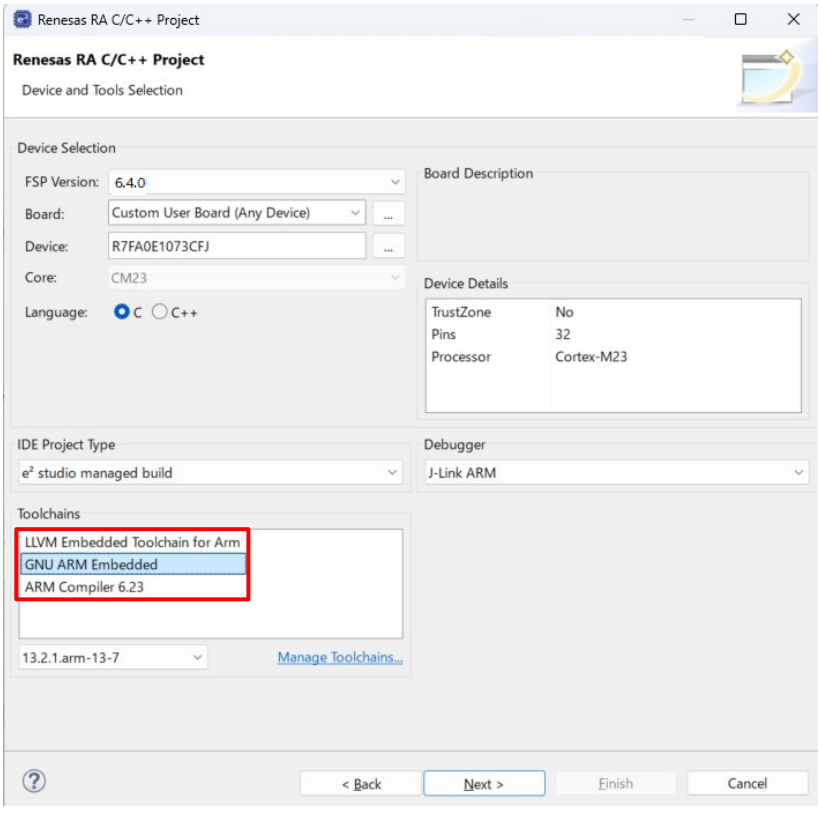
2. Select ”Renesas RA C/C++ Project”



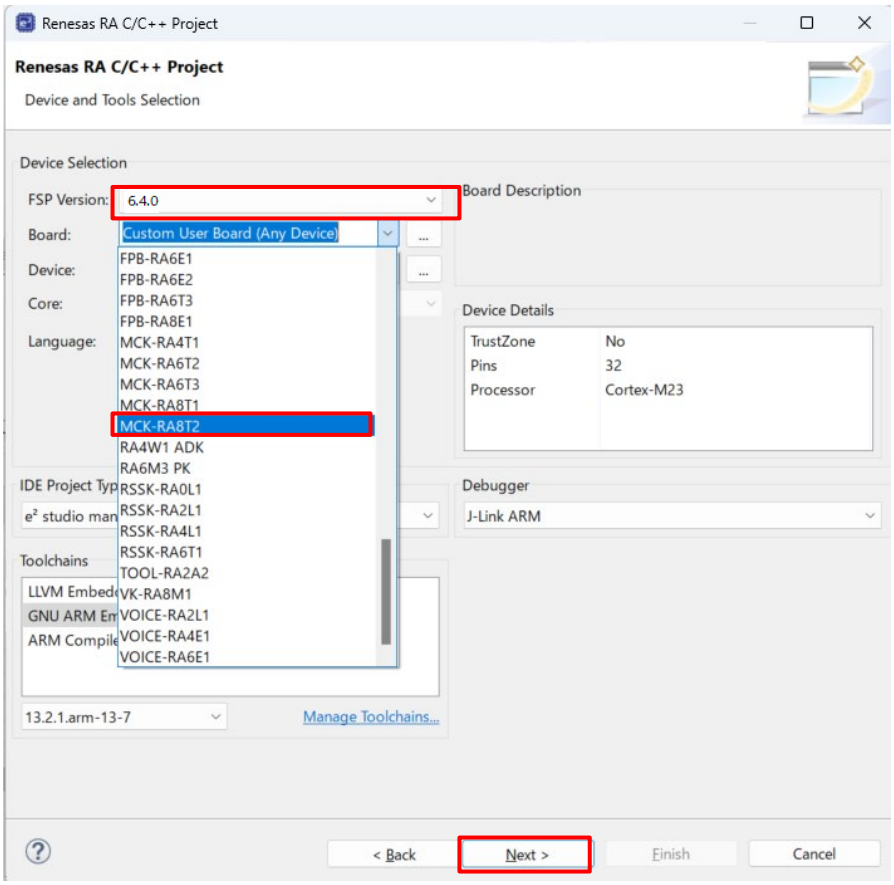
3. Enter any project name.



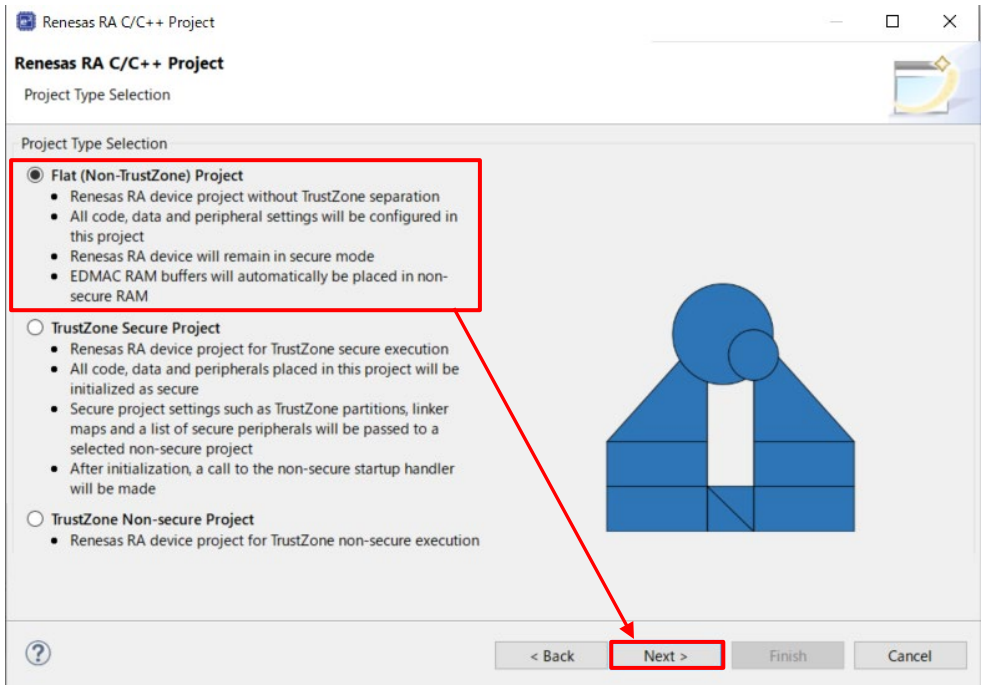
4. Select "Toolchains”



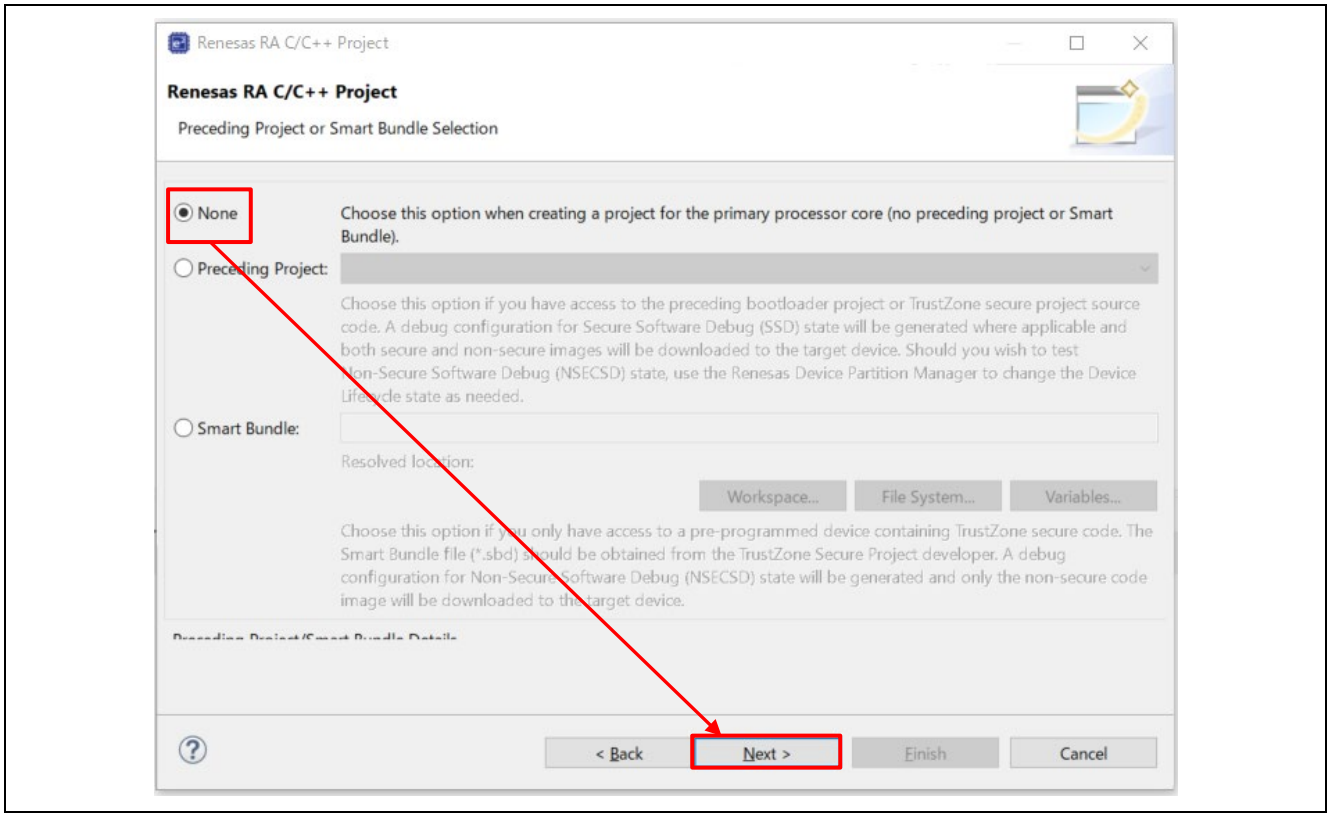
5. Select the FSP Version and Board



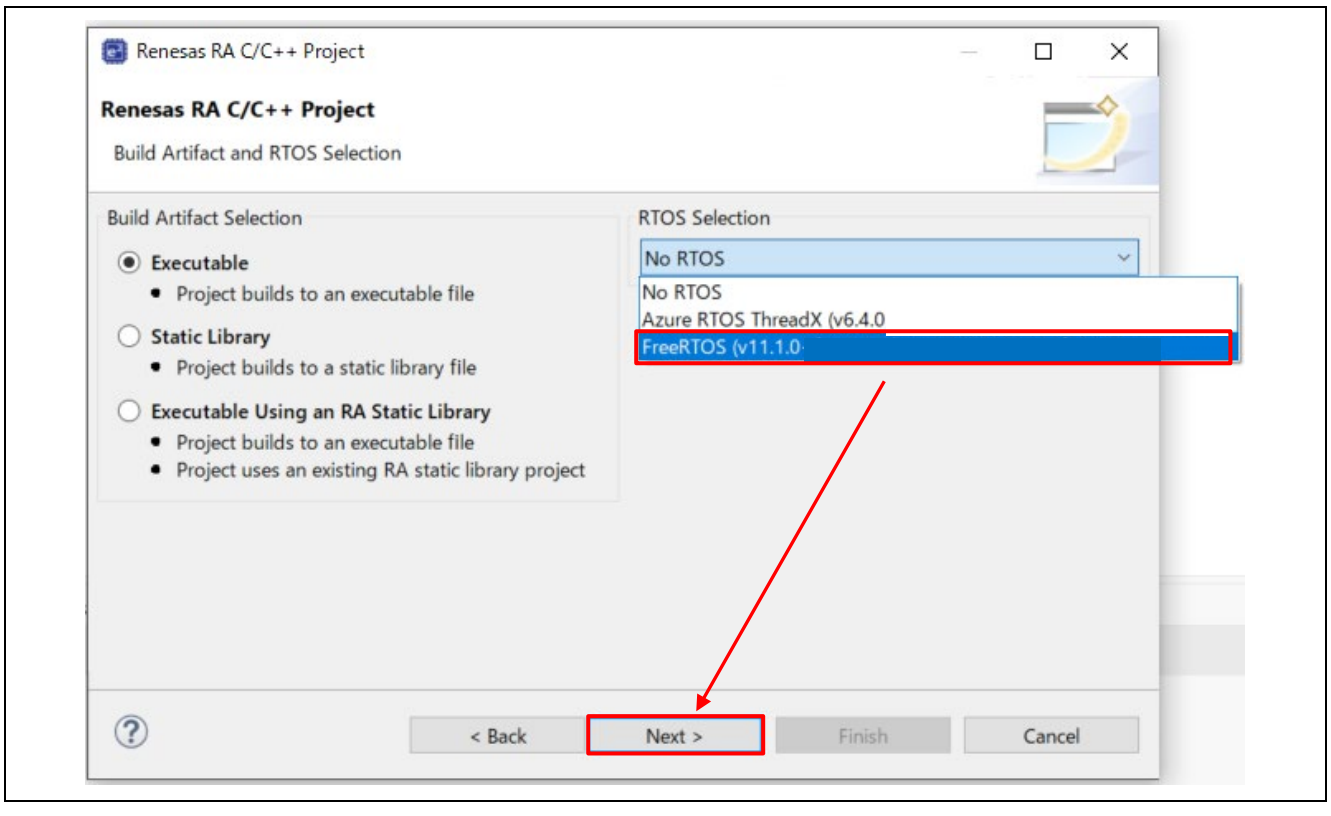
6. Select the project type for “Flat (Non-TrustZone) Project”



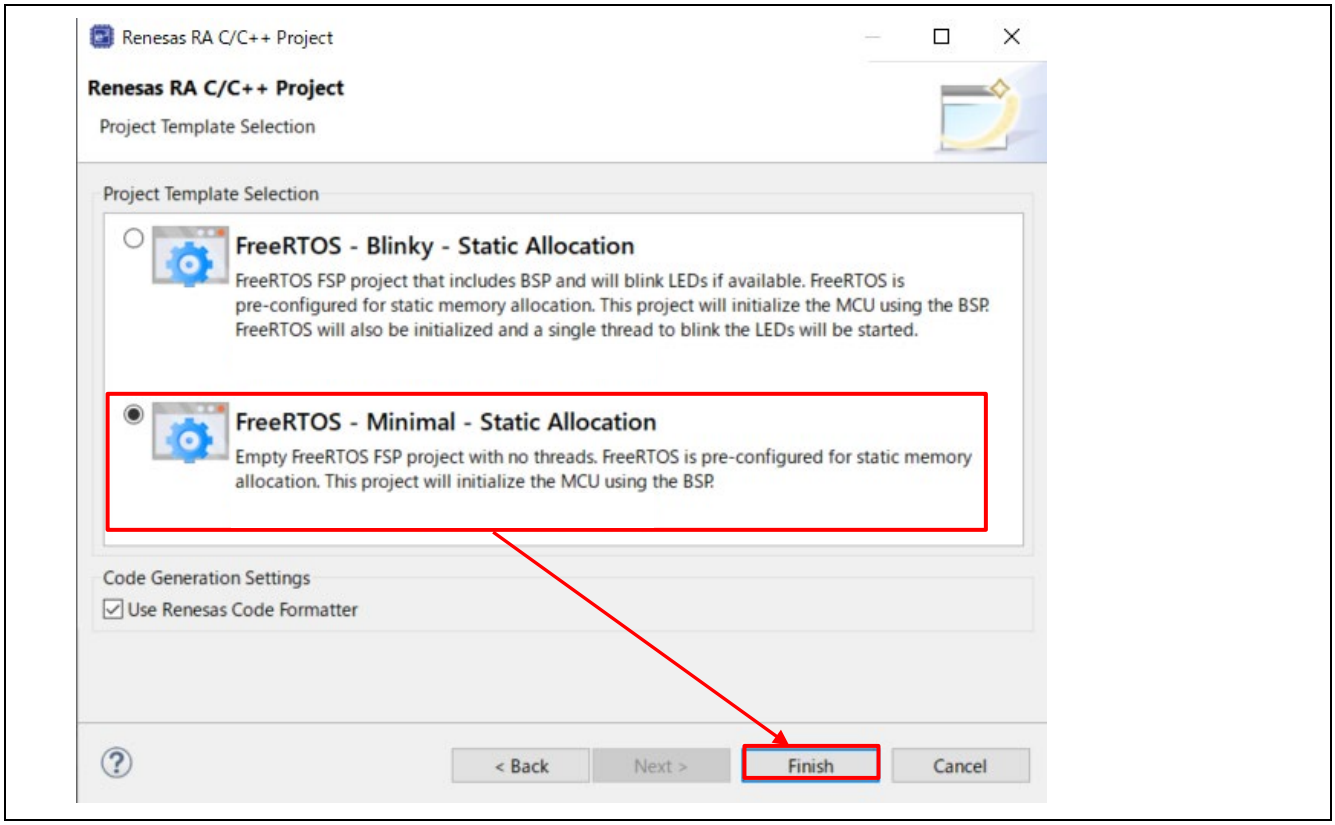
7. Select “None”



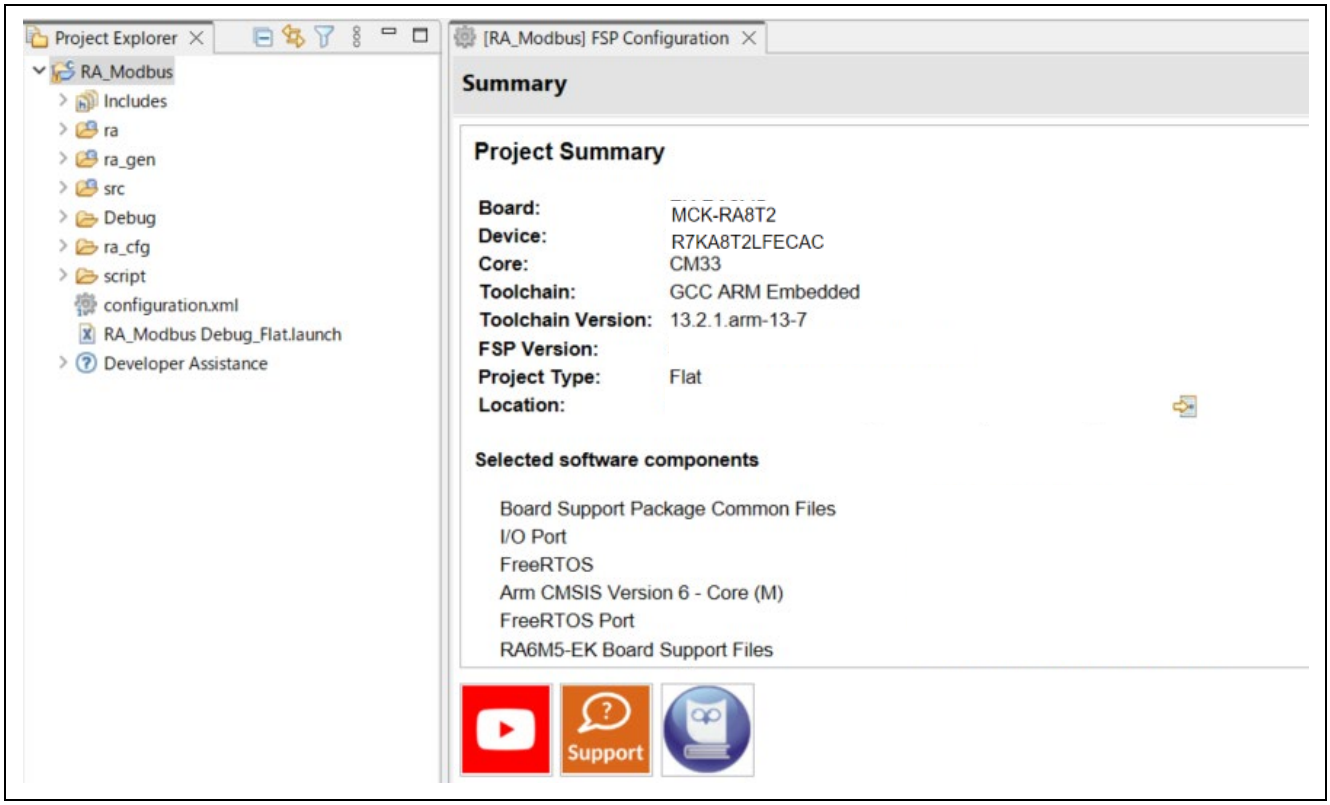
8. Select “FreeRTOS (xxx)”.



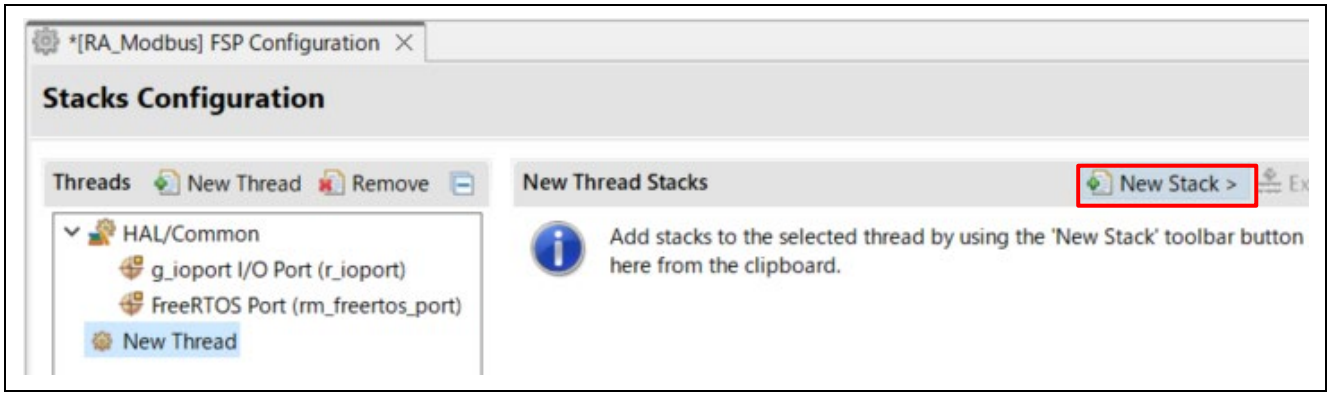
9. Select “FreeRTOS - Minimal - Static Allocation” under Project Template Selection.



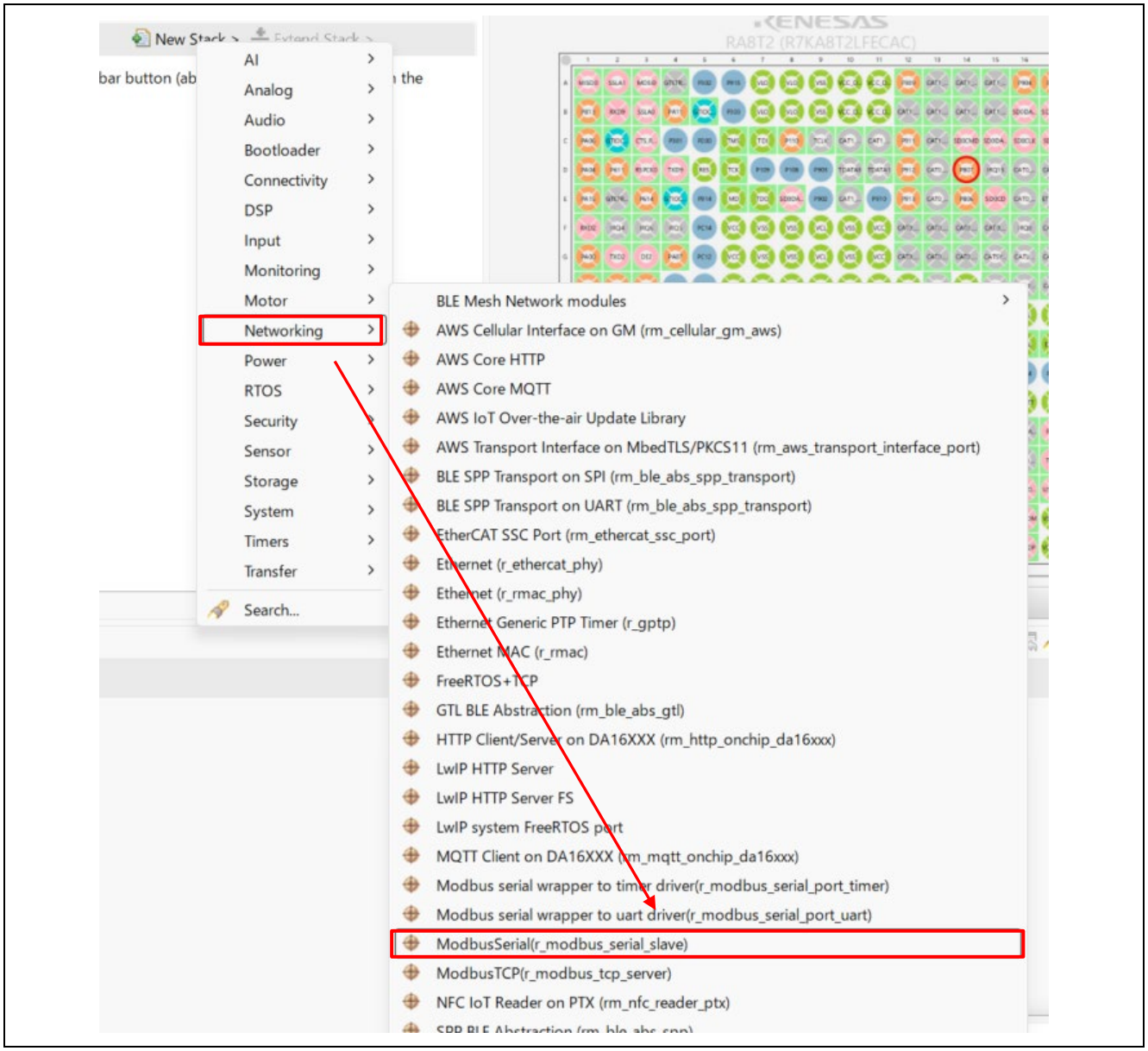
Once the project is created, it appears in the “Project Explorer” and the “FSP Configuration” tab is displayed.



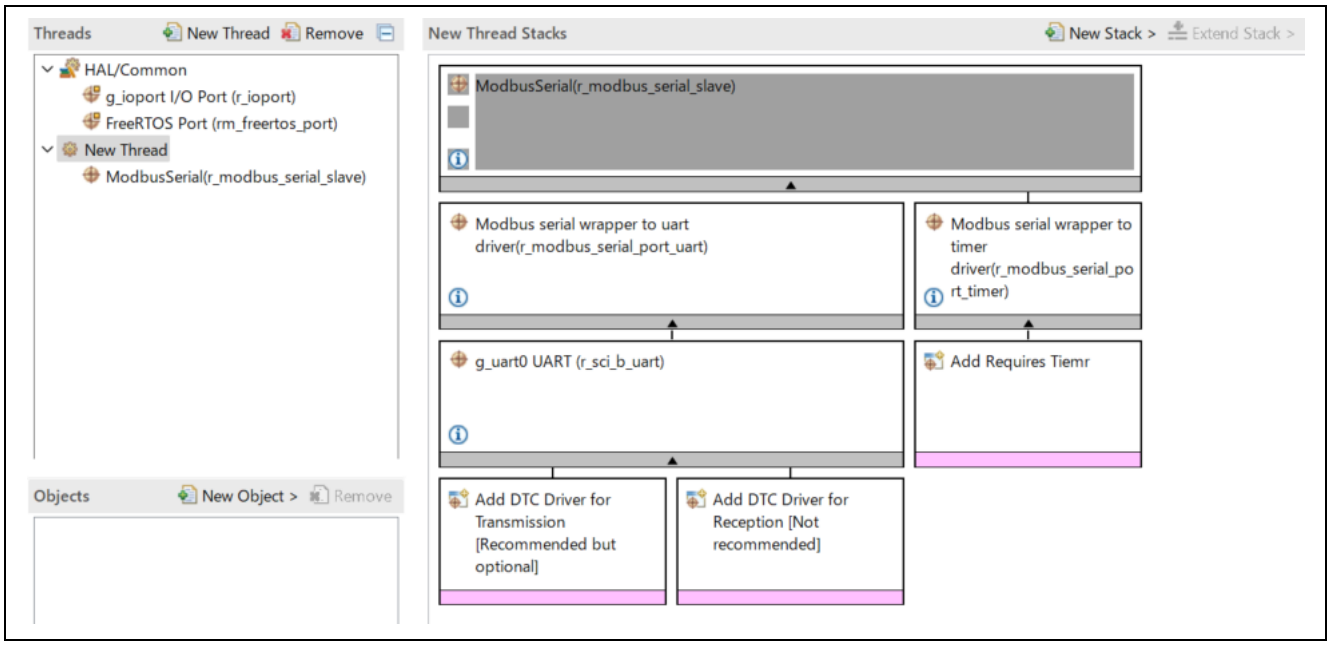
10. Create New Stack
    Select "New Stacks" to be configured.



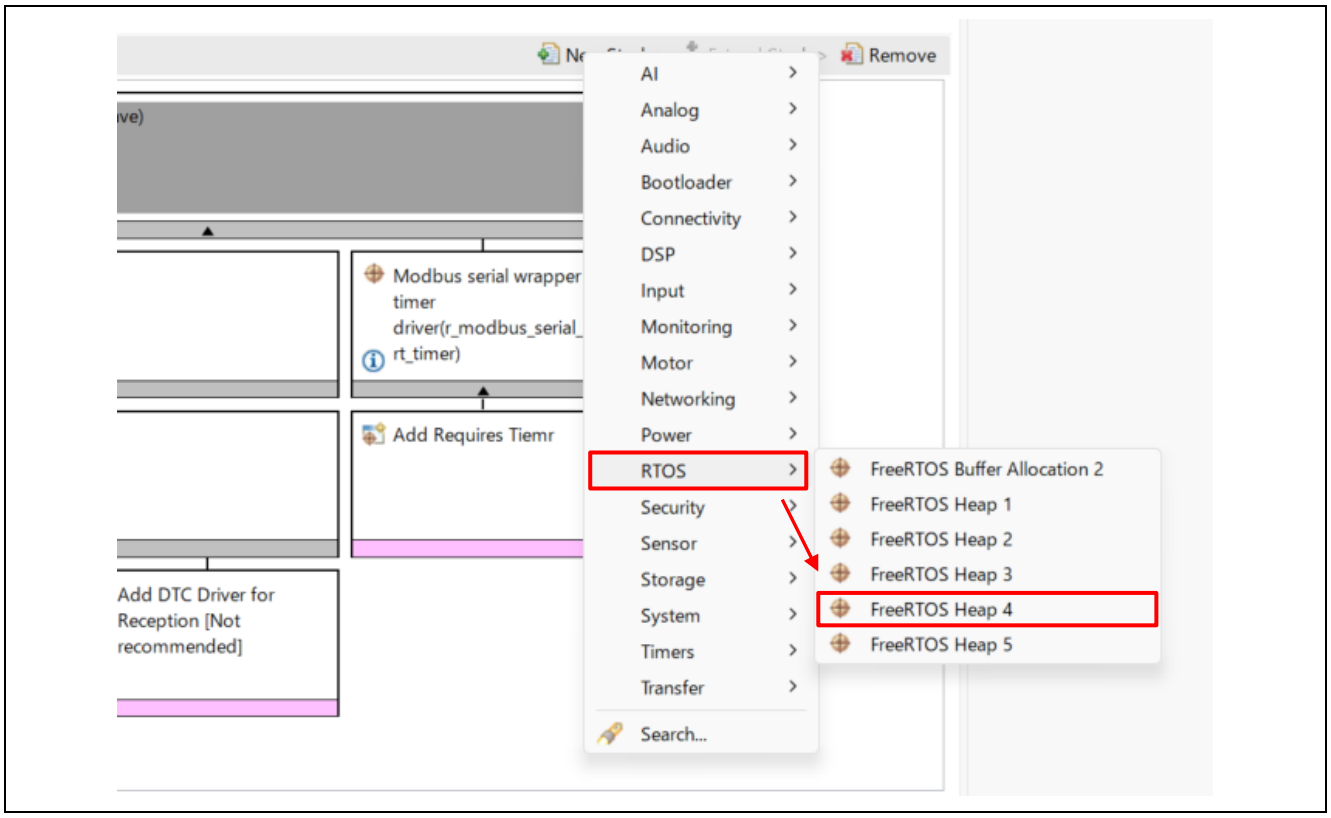
Select “Networking” → ”ModbusSerial(r_modbus_serial_slave)”.



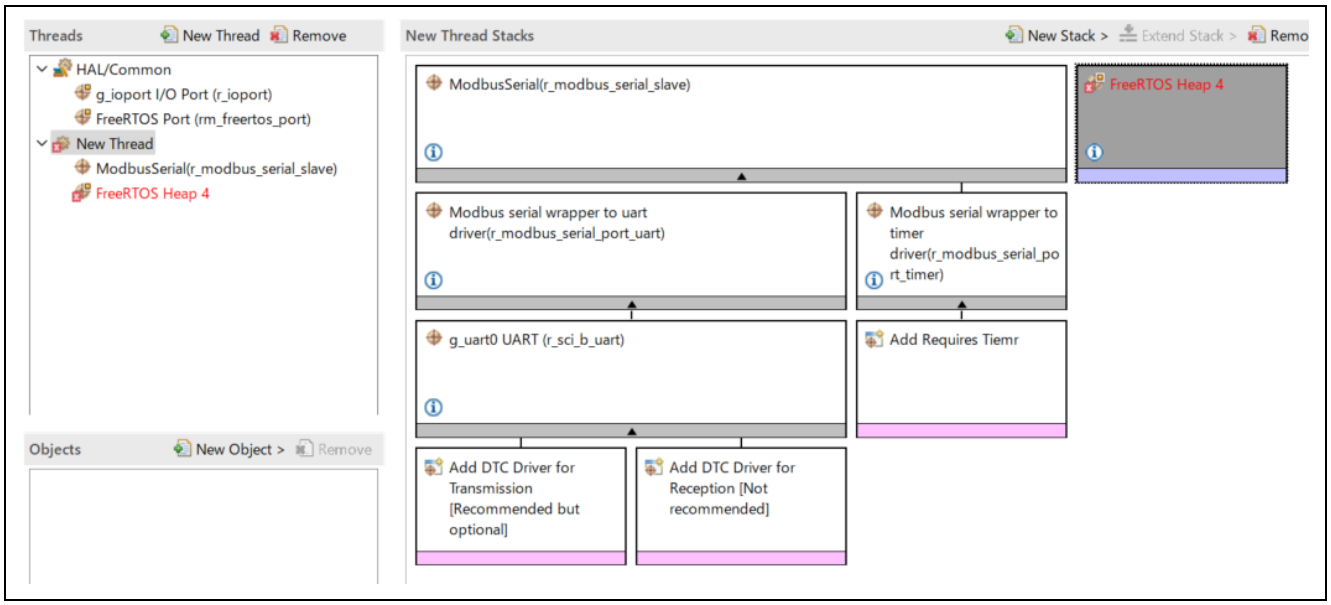
The stack is configured as follows:



11. Add Heap
    Select “New Stack” → “RTOS” → “FreeRTOS Heap 4”.

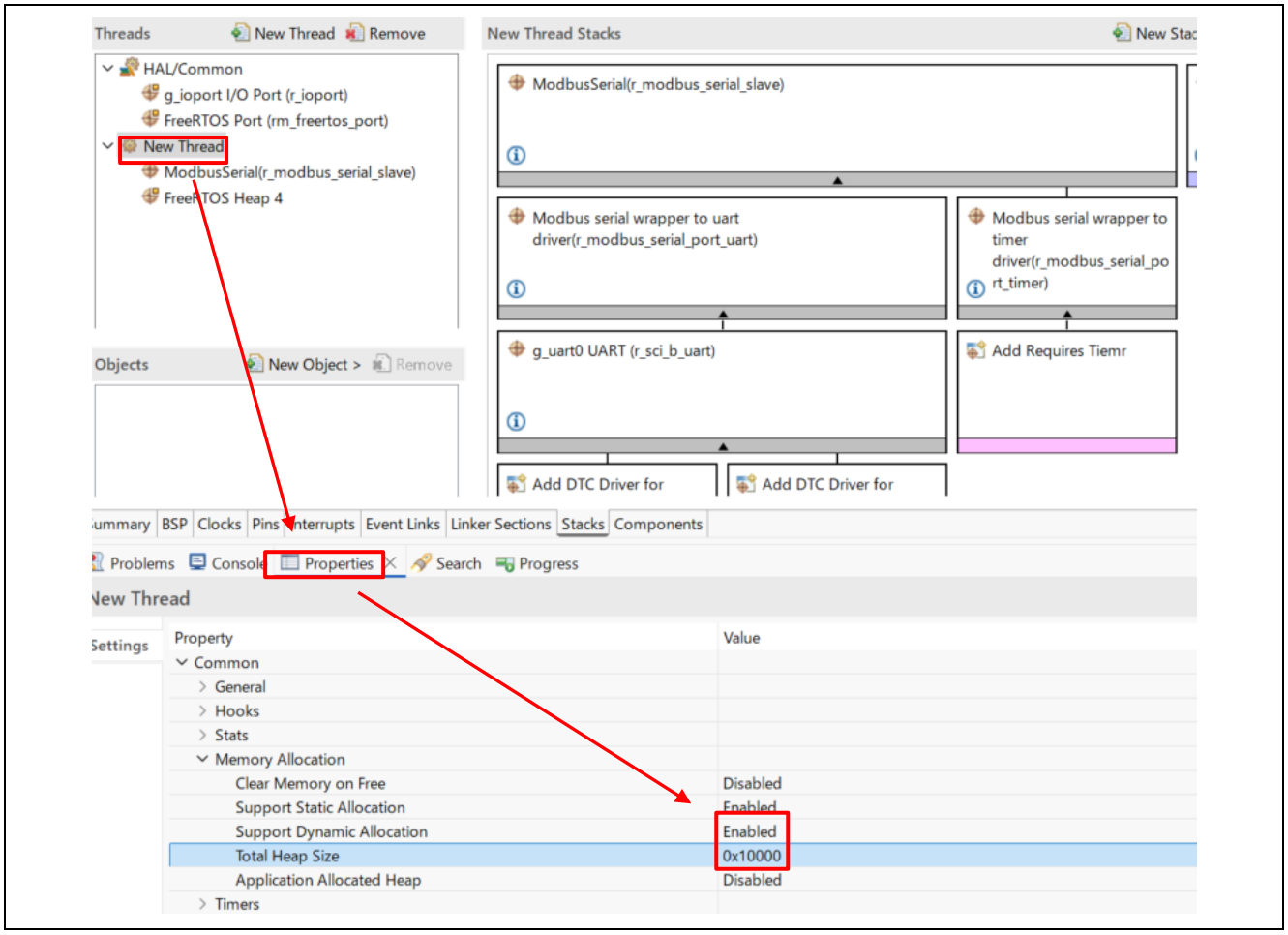


The stack will be configured as follows:



12. Set Support Dynamic Allocation and Total Heap Size
    Open “Properties” of “New Thread” in “Stacks” and change “Support Dynamic Allocation” and “Total Heap Size” in “Memory Allocation” to the following values.

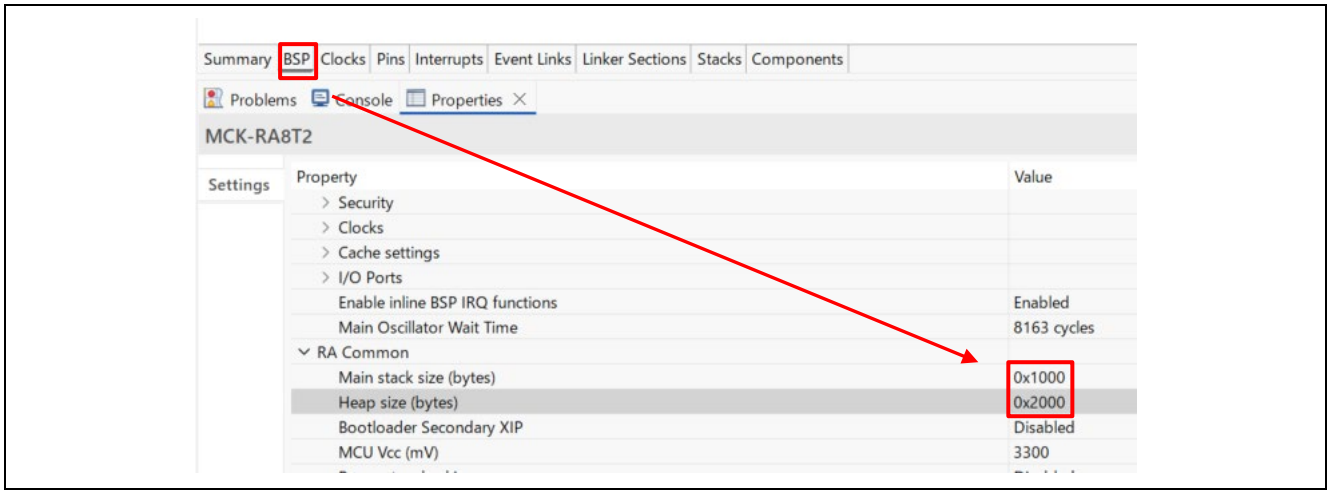
    Support Dynamic Allocation：**Enabled**
    Total Heap Size：**0x10000**



“FreeRTOS Heap4” stack configuration errors are resolved.

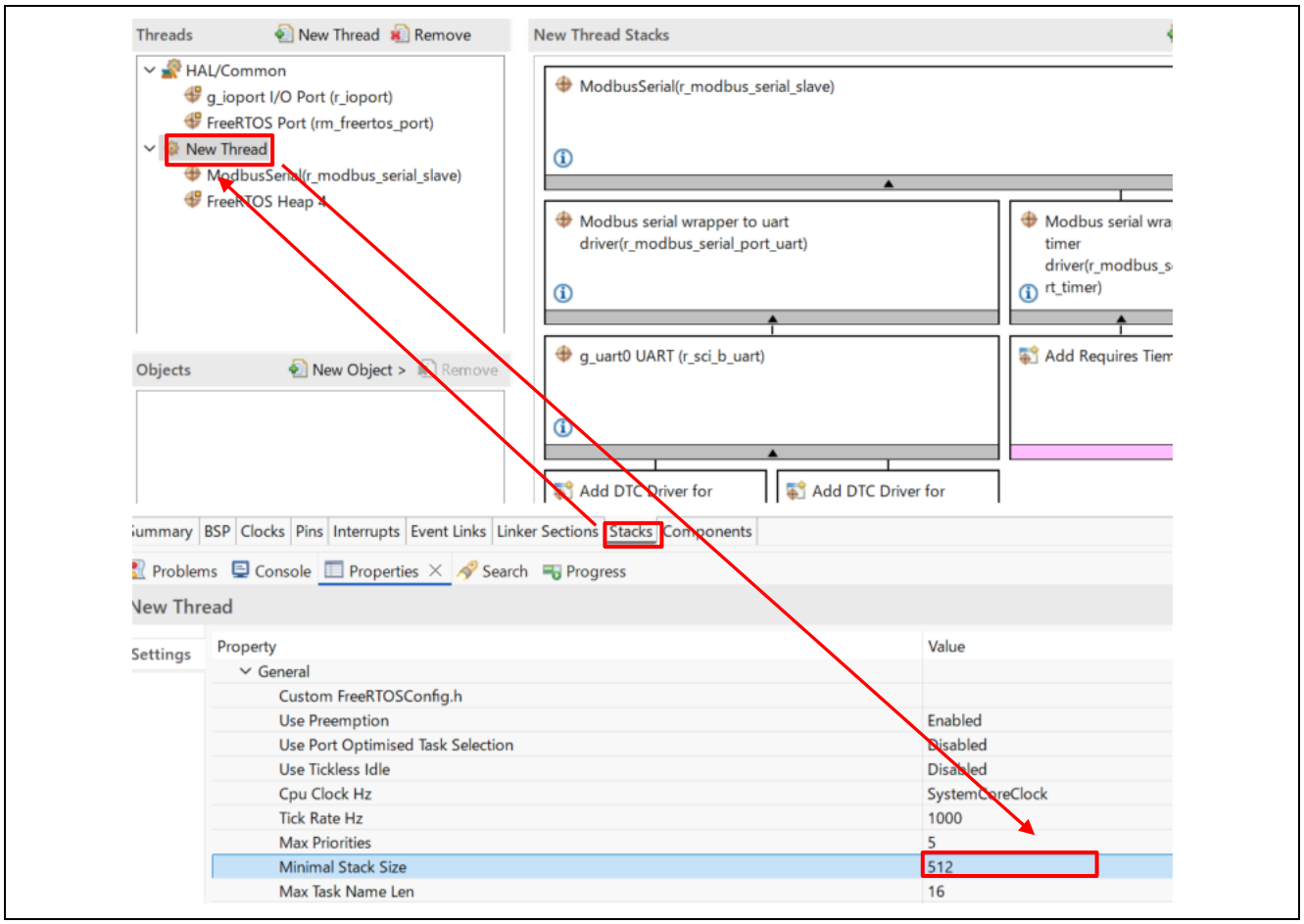
13. Set Stack size
Open “Properties” of “BSP” and change “Main stack size” and “Heap size” in “RA Common” to the following values.

Main stack size：**0x1000**, Heap size：**0x2000**



Open “Properties” of “New Thread”, then click on “Stacks” and change “Minimal Stack size” in “General” to the following values.

Minimal Stack size：**512**



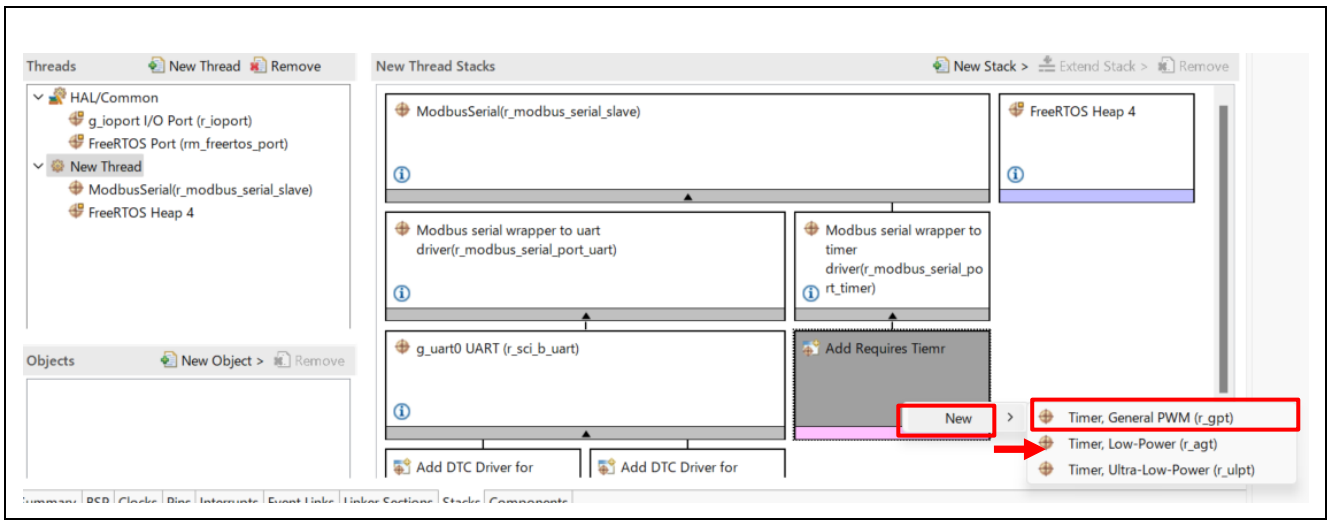
14. Perform the steps for each evaluation board.
    Refer to the following for the evaluation board you are using and follow the steps:

    - MCK-RA8T2, EK-RA8M2：**5.2.2 MCK-RA8T2 / EK-RA8M2 Creating Procedures**
    - EK-RA8T2：**5.2.3 EK-RA8T2 Creating Procedures**

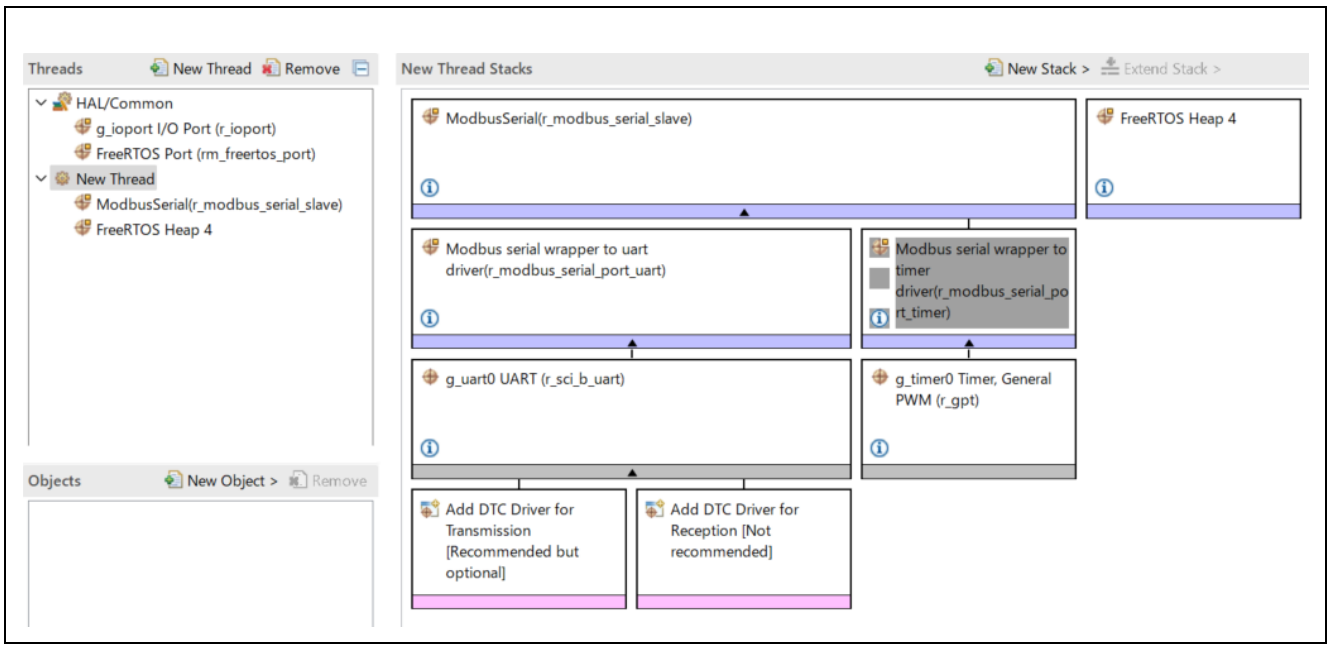
5.2.2 MCK-RA8T2 / EK-RA8M2 Creating Procedures

This section describes the procedures for creating MCK-RA8T2 / EK-RA8M2.

1. Add Timer Driver
   Click “New” → “Timer, General PWM (r_gpt)” to “Add Requires Timer”.



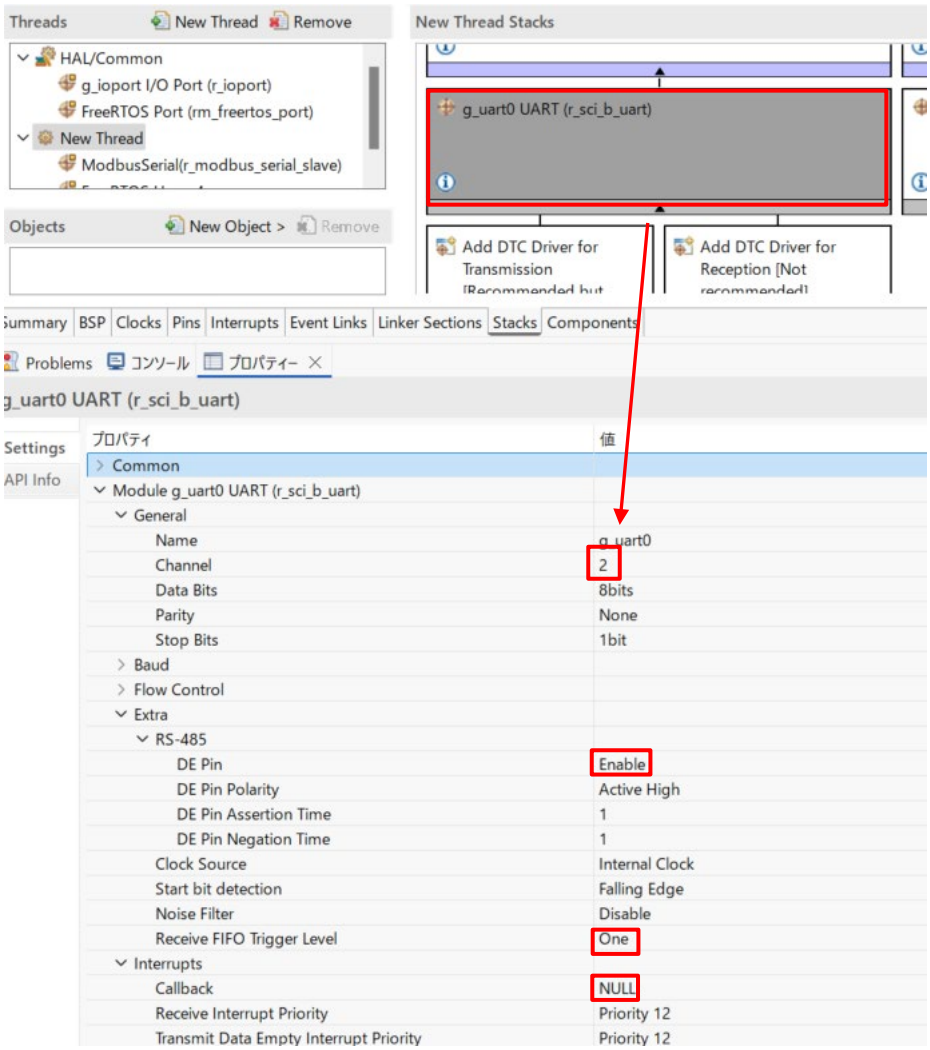
The stack is configured as follows:



2. Set UART
Open “Properties” of “g_uart0 UART (r_sci_b_uart)” in “Stacks” and change “Channel” (values vary depending on the evaluation board), “DE Pin” and “Receive FIFO Trigger Level” to the following values. To change the serial transmission mode, baud rate, parity bit, or stop bit, see “**5.3 Settings the Serial Transmission Modes / Baud Rate / Parity Bit / Stop Bits**”.

Channel：MCK-RA8T2 : **2,** EK-RA8M2 : **5**
DE Pin：**Enable**
Receive FIFO Trigger Level：**One**
Interrupts / Callback：**NULL** *1

***1:**
Since the UART callback function is set in the Modbus protocol stack initialization process, the Interrupts/Callback property setting of "g_uart0 UART(r_sci_b_uart)" should be set to the default "NULL".

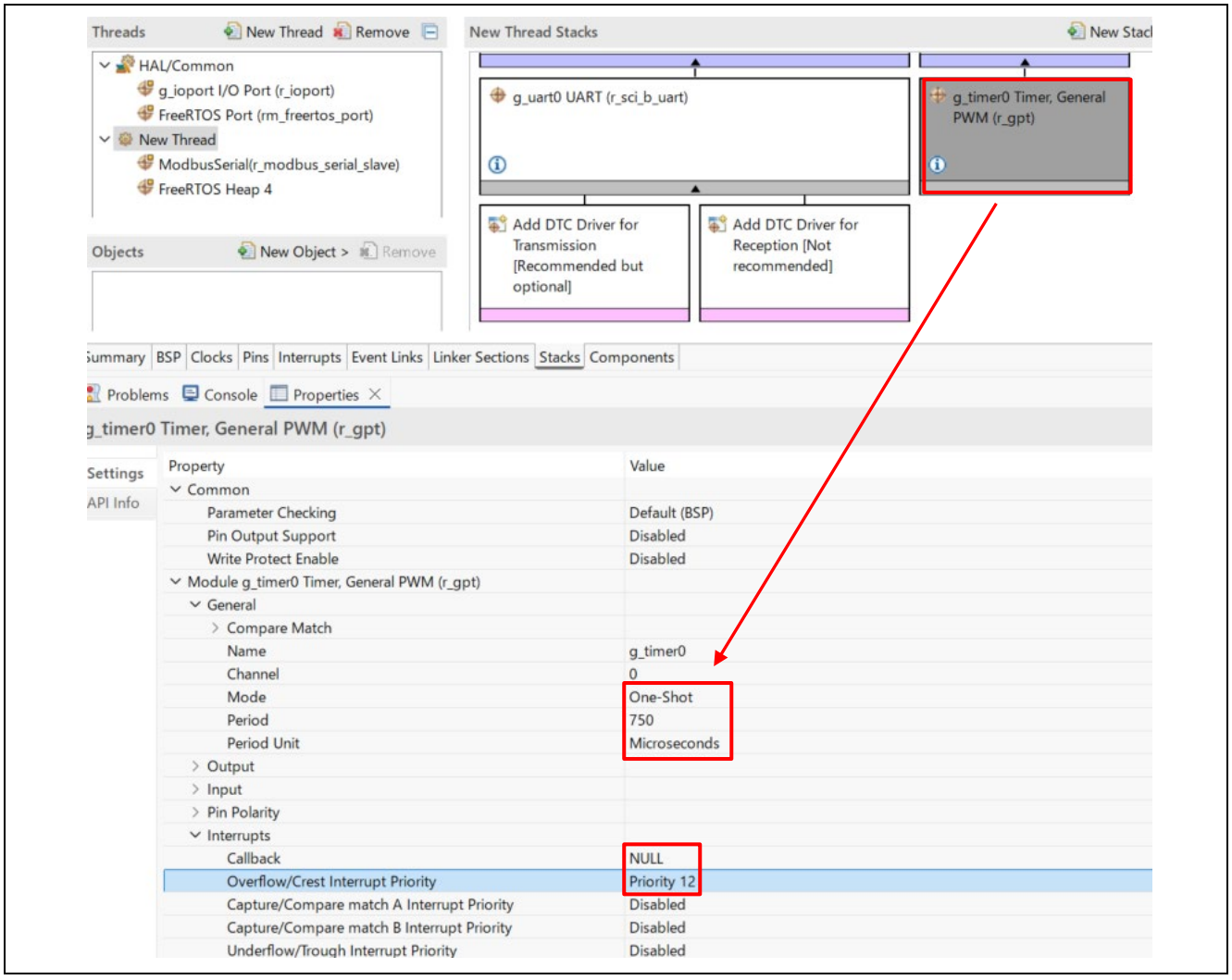


3. Set Timer

Open “Properties” of “g_timer0 Timer, General PWM (r_gpt)” in “Stacks” and change “Mode”, “Period”, “Period Unit” and “Overflow/Crest Interrupt Priority” to the following values.

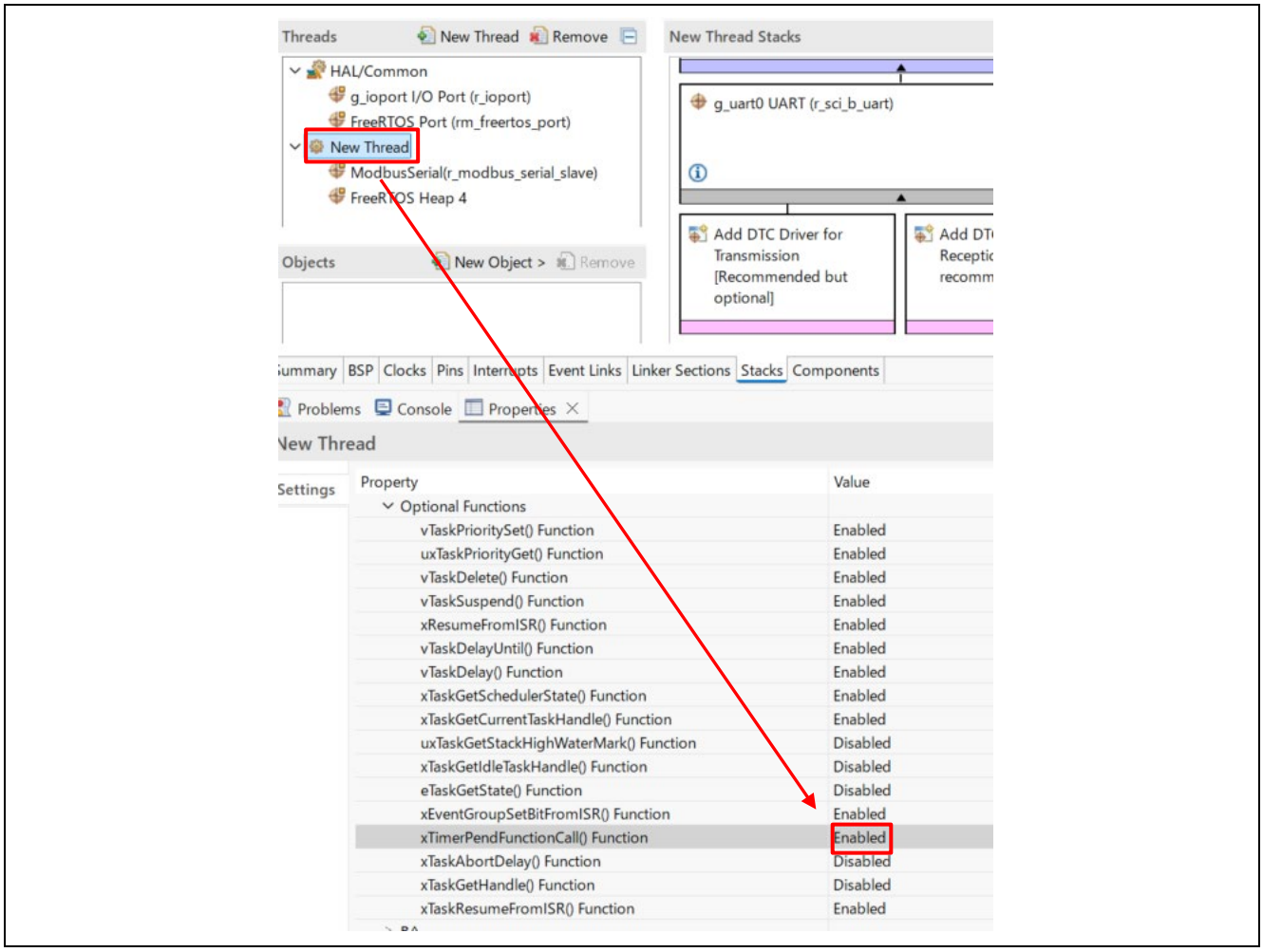
Mode：**One-Shot**
Period：**750**
Period Unit：**Microseconds**
Interrupts / Callback：**NULL** *2
Overflow/Crest Interrupt Priority：**Priority 12**

*2**:**
Since the Timer callback function is set in the Modbus protocol stack initialization process, the Interrupts/Callback property setting of "g_timer0 Timer, General PWM (r_gpt)" should be set to the default "NULL".

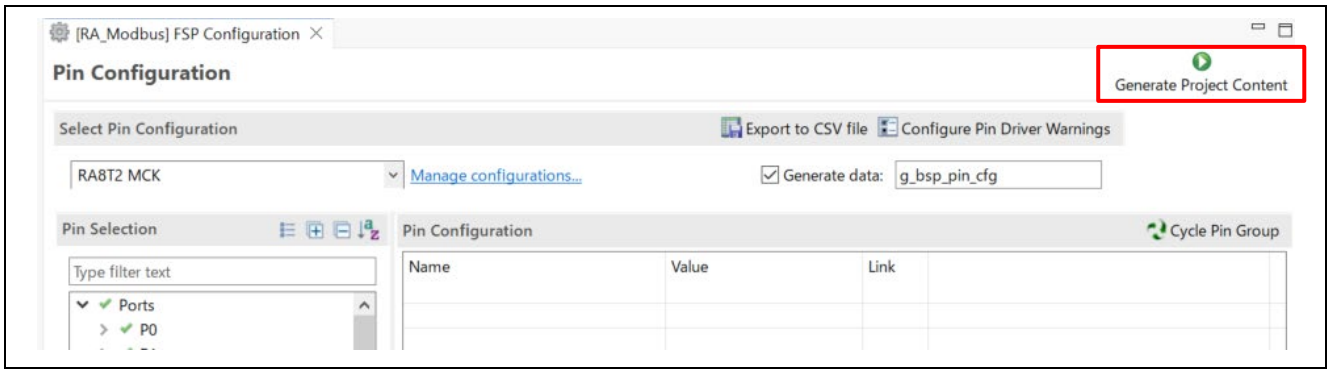


4. Set xTimerPendFunctionCall() Function
   Open “Properties” of “New Thread” in “Stacks” and change “xTimerPendFunctionCall() Function” to the following values.

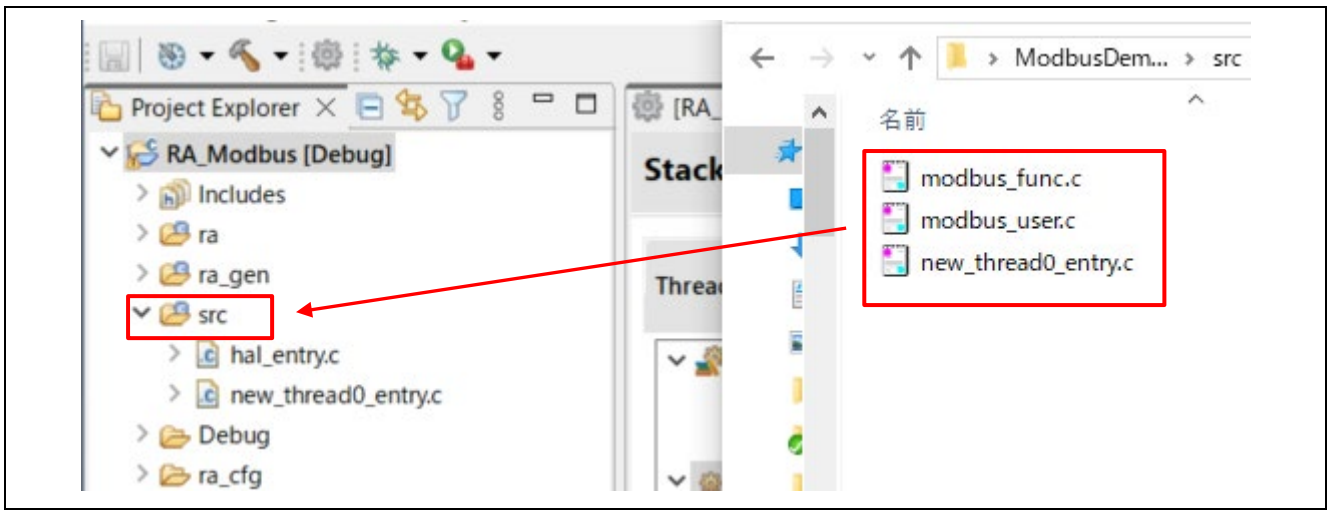
   xTimerPendFunctionCall() Function：**Enabled**

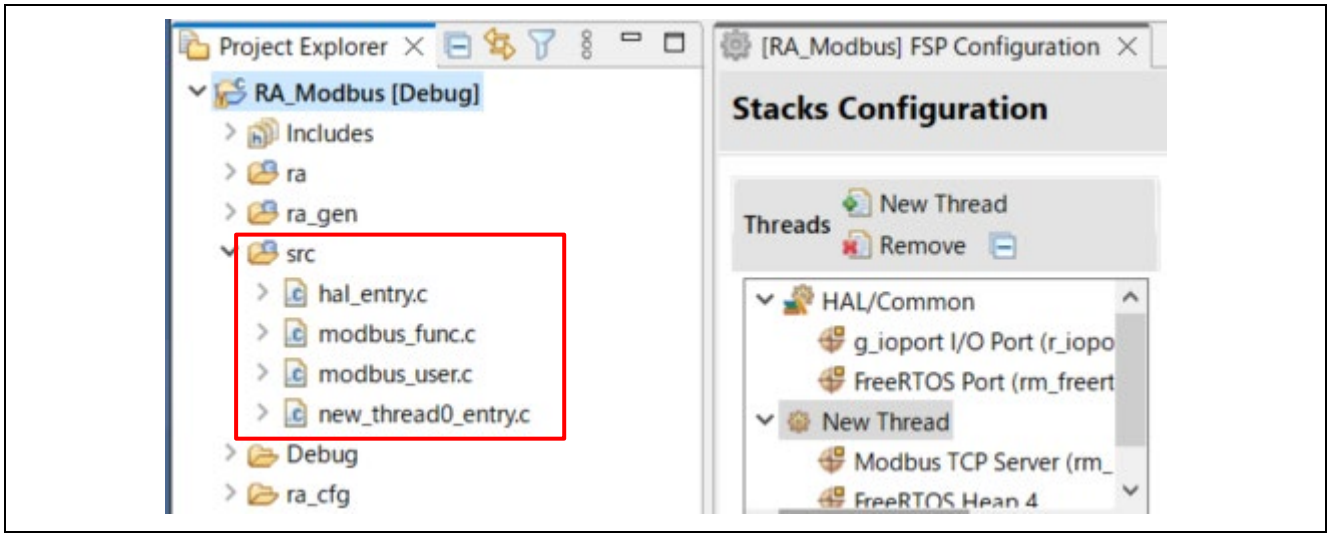


5. Generate the code
   Generate the code with "Generate Project Content".



6. Add Modbus Sample Application
   Copy modbus_func.c, modbus_user.c, and new_thread0_entry.c from src folder of the sample program package to the src folder of project and overwrite them.

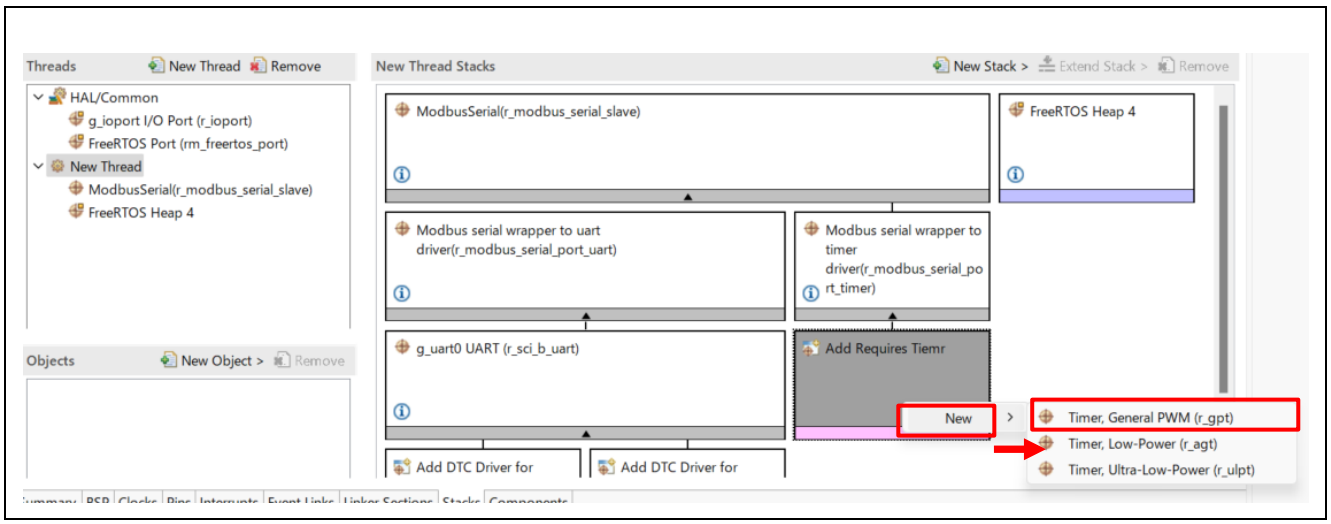




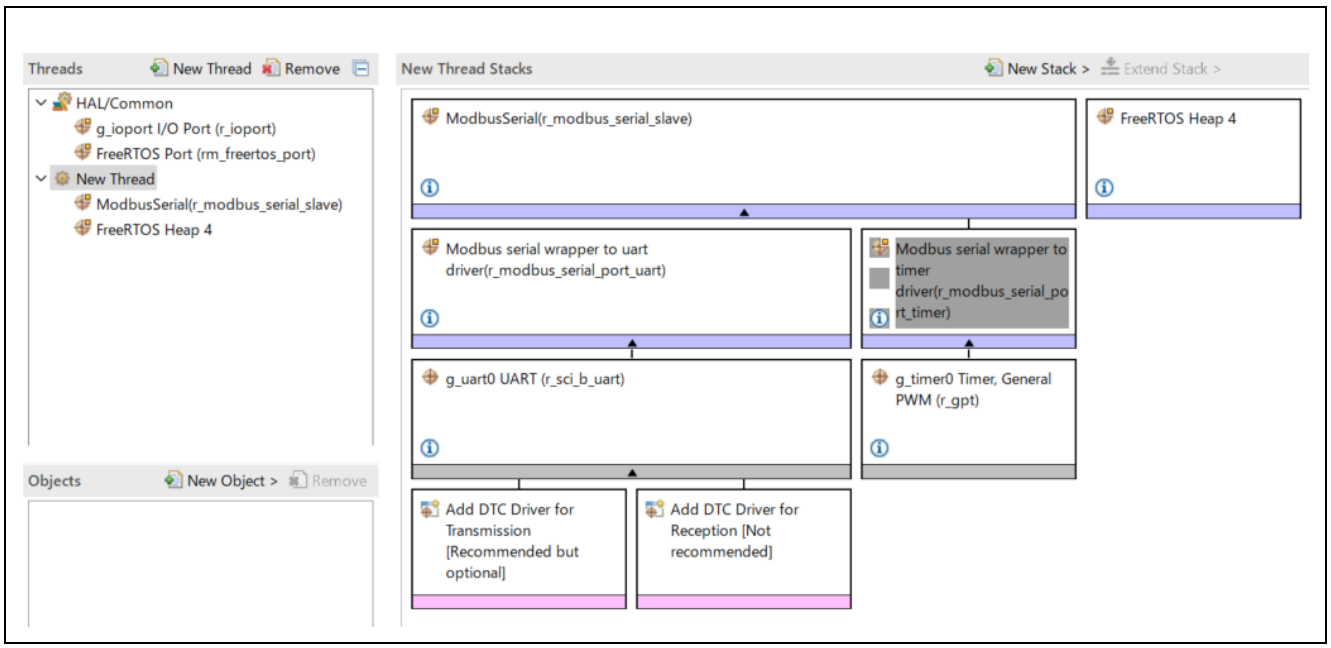
5.2.3 EK-RA8T2 Creating Procedures

This section describes the procedures to create EK-RA8T2.

1. Add Timer Driver
   Click “New” → “Timer, General PWM (r_gpt)” to “Add Requires Timer”.



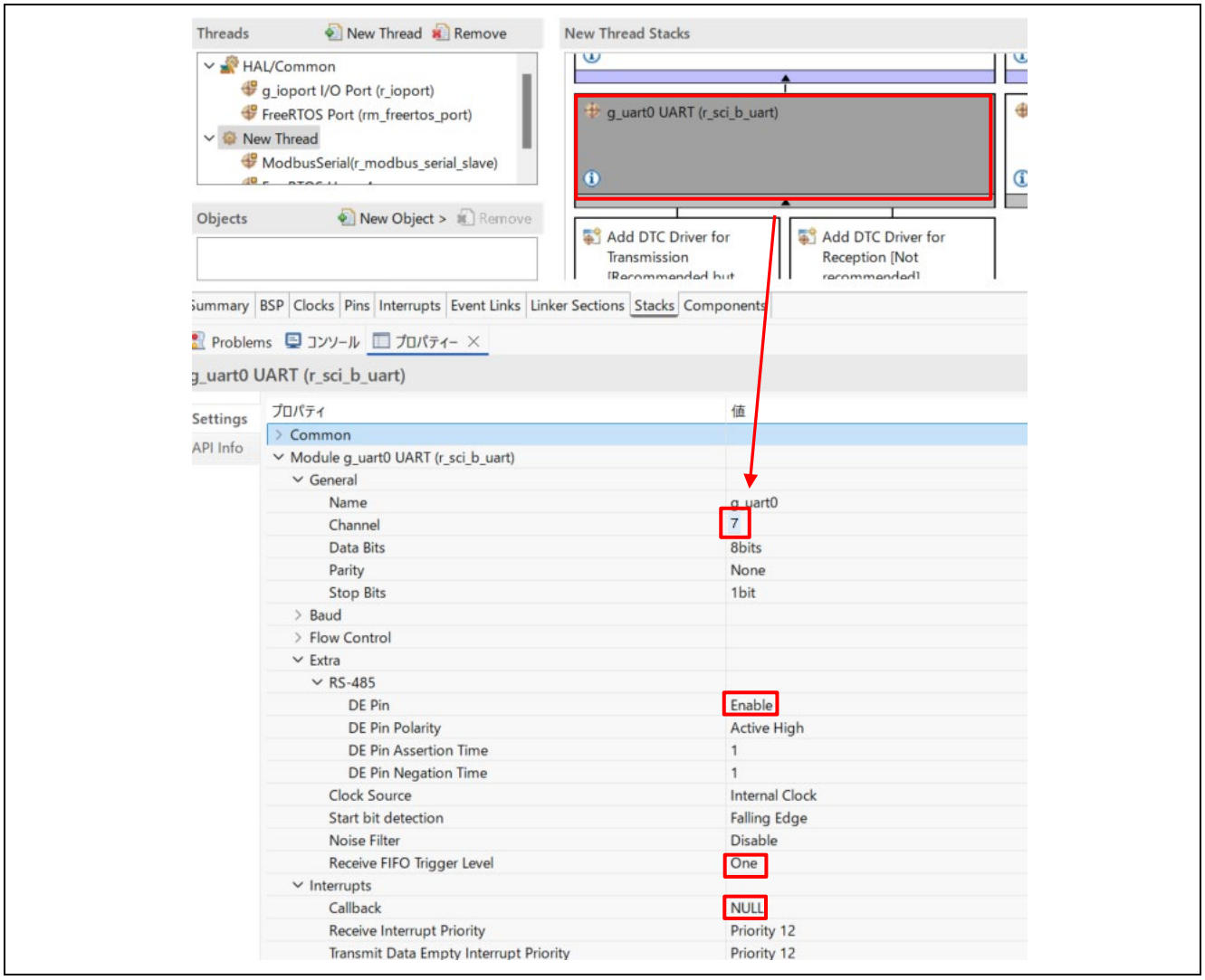
The stack is configured as follows:



2. Set UART
Open “Properties” of “g_uart0 UART (r_sci_b_uart)” in “Stacks” and change “Channel” (values vary depending on the evaluation board), “DE Pin” and “Receive FIFO Trigger Level” to the following values. To change the serial transmission mode, baud rate, parity bit, or stop bit, see “**5.3 Settings the Serial Transmission Modes / Baud Rate / Parity Bit / Stop Bits**”.

Channel：**7**
DE Pin：**Enable**
Receive FIFO Trigger Level：**One**
Interrupts / Callback：**NULL** *1

***1:**
Since the UART callback function is set in the Modbus protocol stack initialization process, the Interrupts/Callback property setting of "g_uart0 UART(r_sci_b_uart)" should be set to the default "NULL".

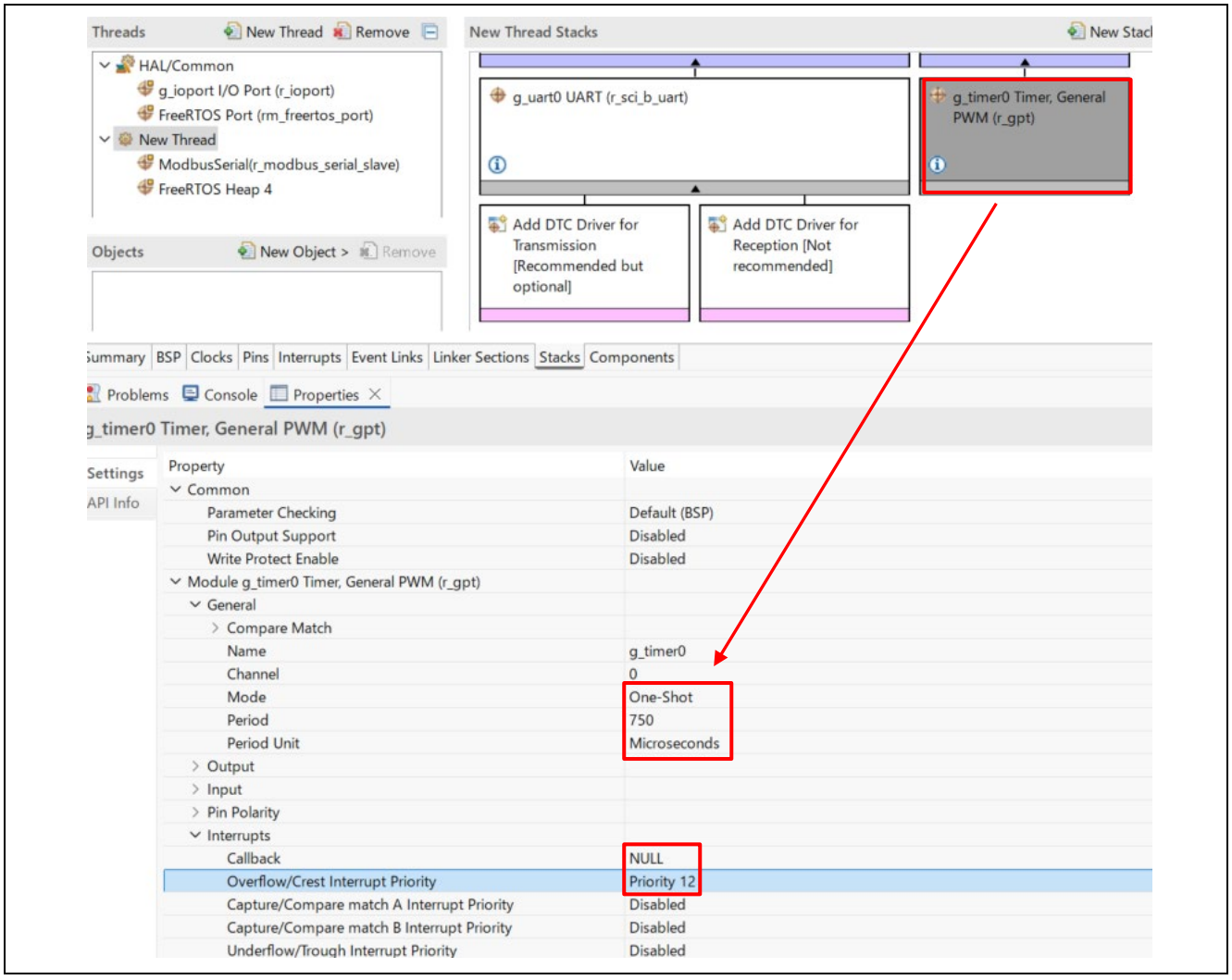


3. Set Timer

Open “Properties” of “g_timer0 Timer, General PWM (r_gpt)” in “Stacks” and change “Mode”, “Period”, “Period Unit” and “Overflow/Crest Interrupt Priority” to the following values.

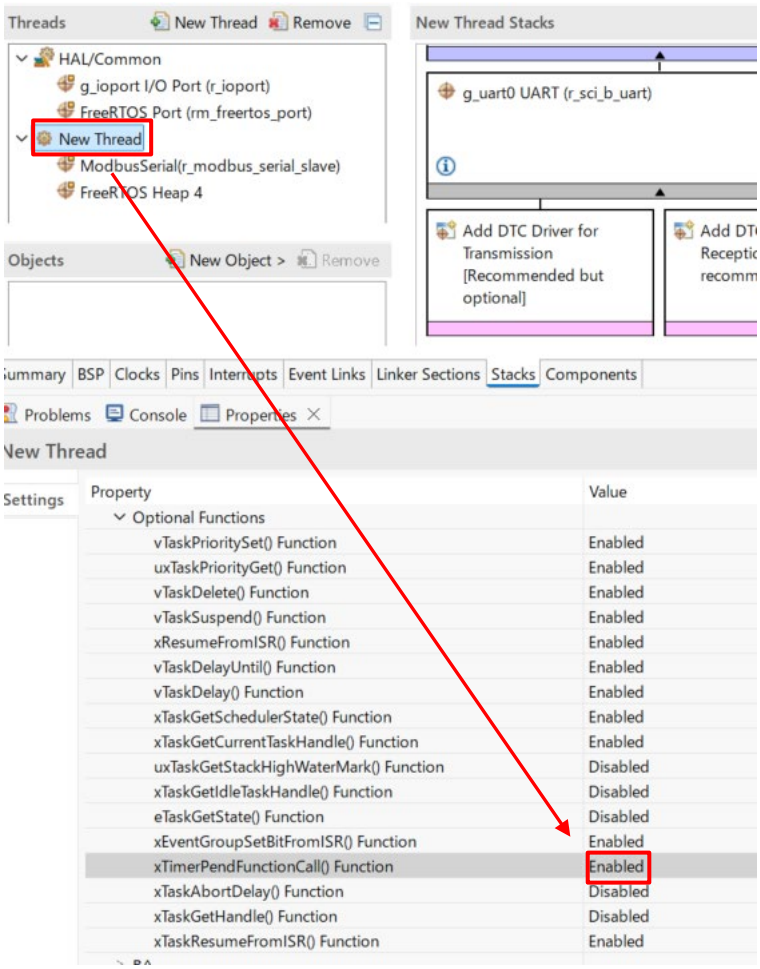
Mode：**One-Shot**
Period：**750**
Period Unit：**Microseconds**
Interrupts / Callback：**NULL** *2
Overflow/Crest Interrupt Priority：**Priority 12**

*2**:**
Since the Timer callback function is set in the Modbus protocol stack initialization process, the Interrupts/Callback property setting of "g_timer0 Timer, General PWM (r_gpt)" should be set to the default "NULL".

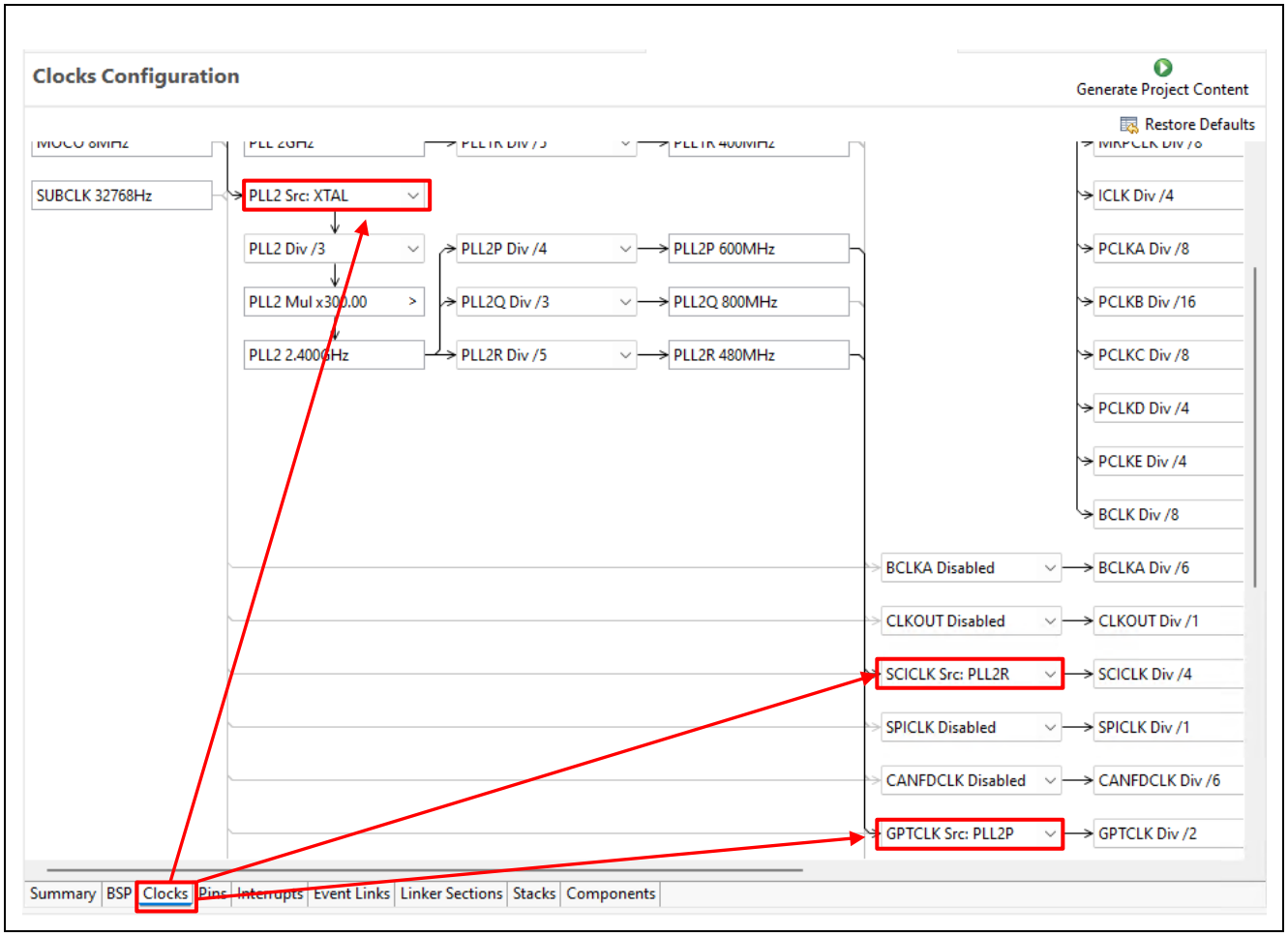


4. Set xTimerPendFunctionCall() Function
   Open “Properties” of “New Thread” in “Stacks” and change “xTimerPendFunctionCall() Function” to the following values.

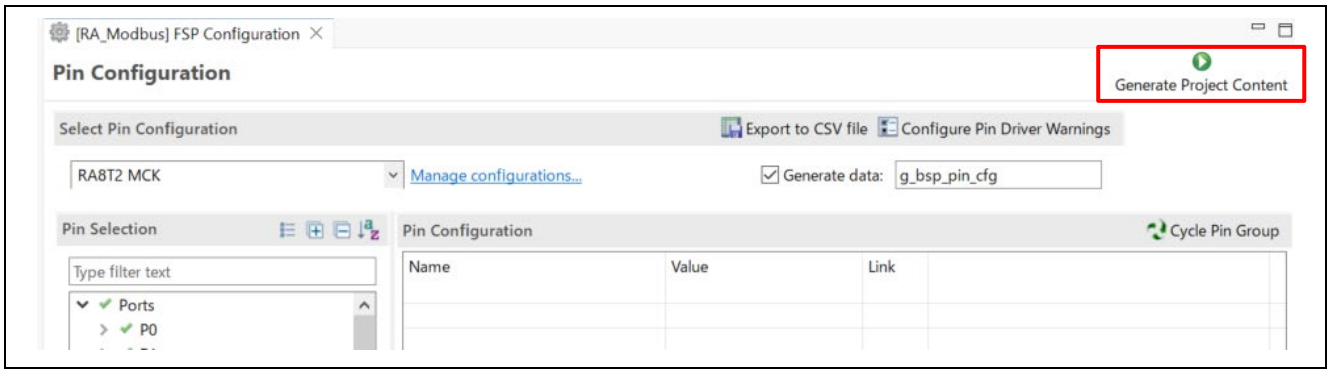
   xTimerPendFunctionCall() Function：**Enabled**



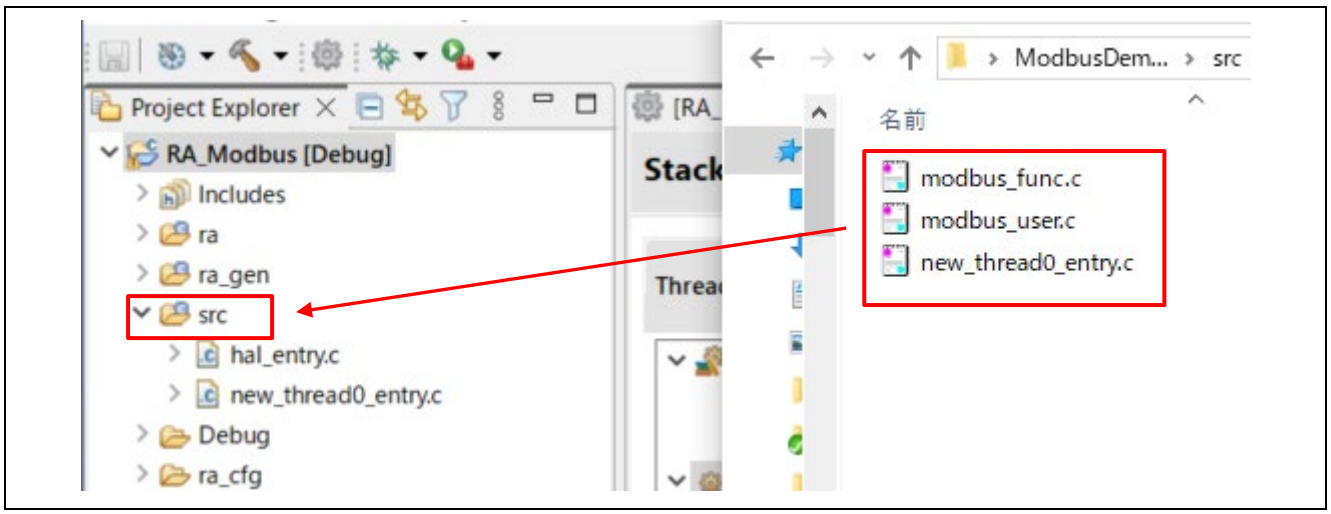
5. Set Clocks
   Change "PLL2 Src", "SCICLK Src", and "GPTCLK Src" in "Clocks" to the following values.
   PLL2 Src : **XTAL**
   SCICLK Src : **PLL2R**
   GPTCLK Src : **PLL2P**

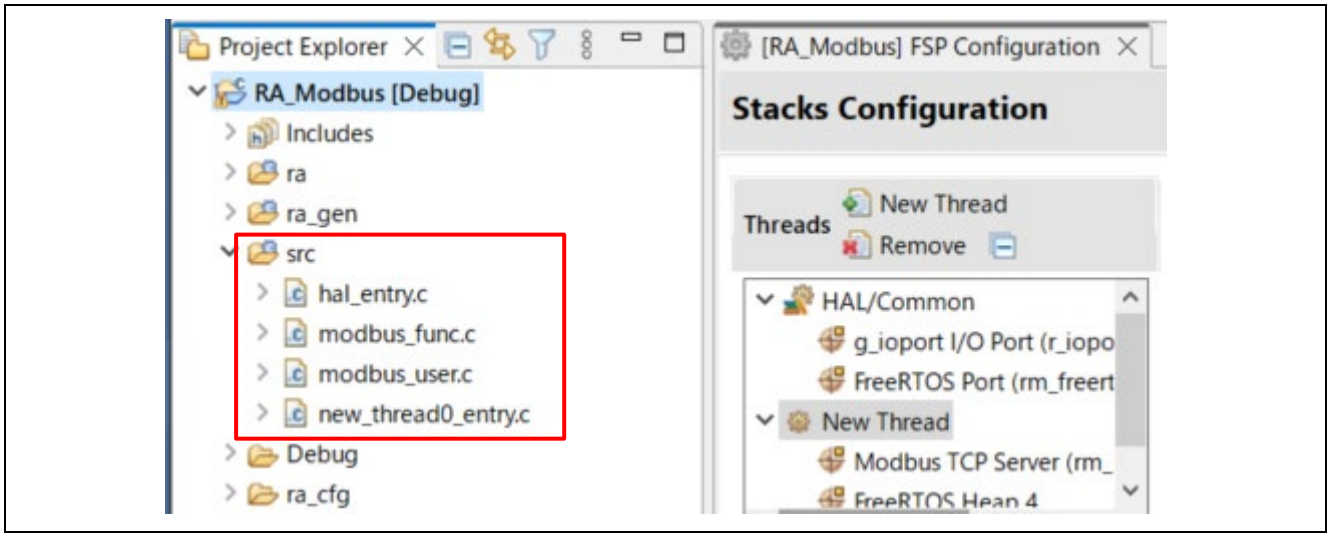


6. Generate the code
   Generate the code with "Generate Project Content".



7. Add Modbus Sample Application
   Copy modbus_func.c, modbus_user.c, and new_thread0_entry.c from src folder of the sample program package to the src folder of project and overwrite them.





5.3 Settings the Serial Transmission Modes / Baud Rate / Parity Bit / Stop Bits

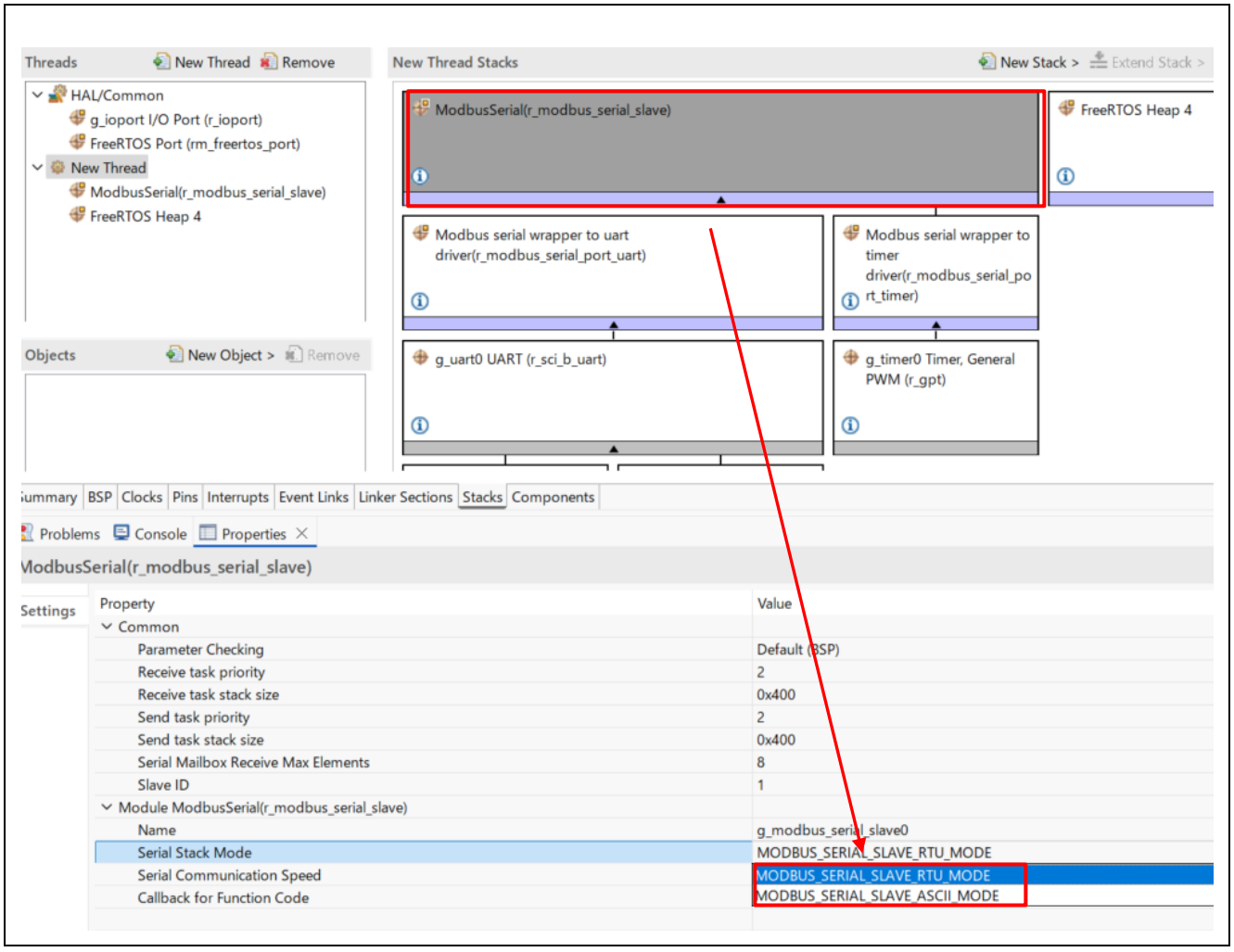
This section describes the settings of serial transmission modes, baud rate, parity bit and stop bits.

5.3.1 Settings the Serial Transmission Modes

This section describes the settings of serial transmission modes.

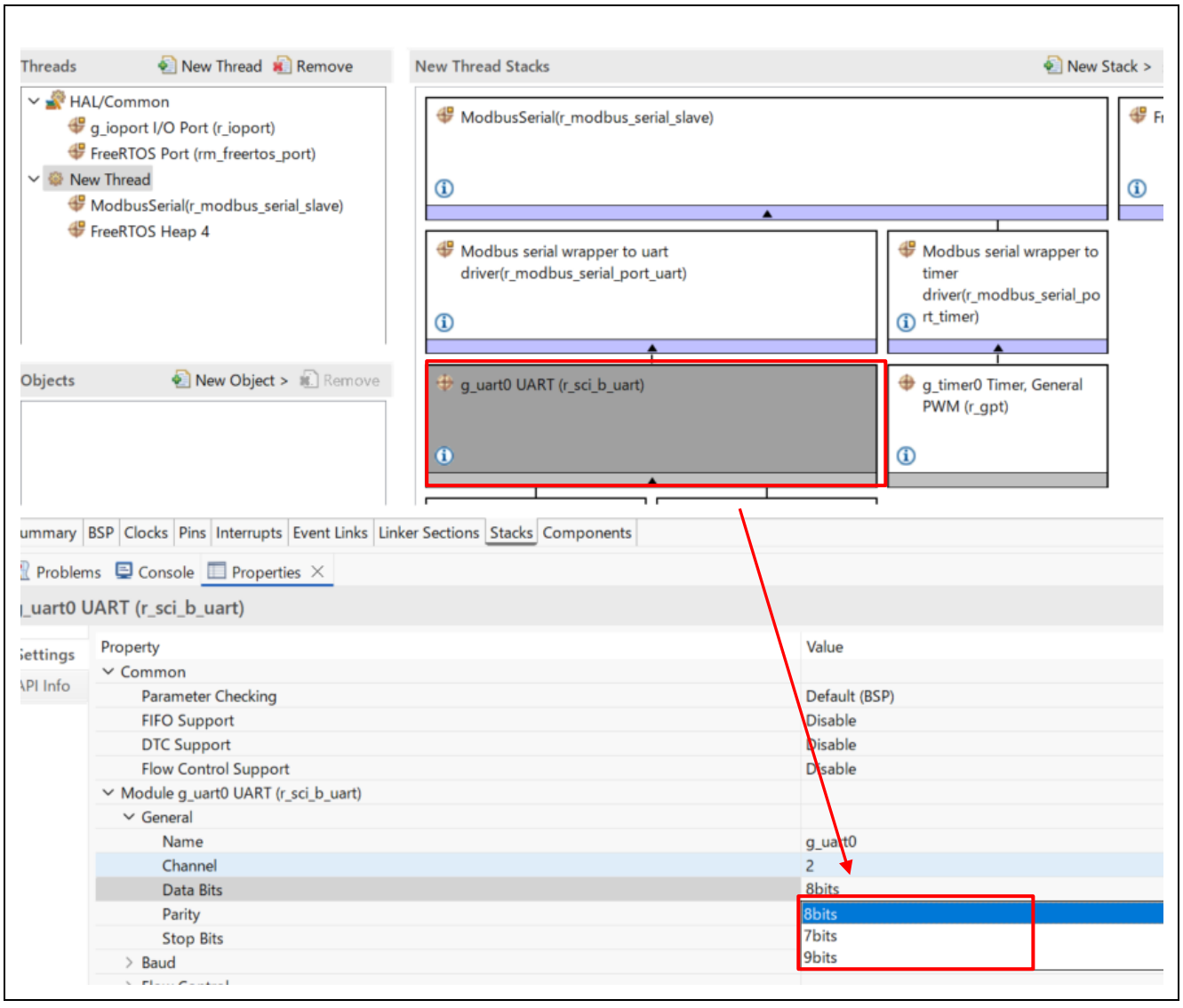
1. Set ModbusSerial
   Open “Properties” of “ModbusSerial(r_modbus_serial_slave)” in “Stacks” and change “Serial Stack Mode” to the following values.

- When communicating at RTU
  Serial Stack Mode： **MODBUS_SERIAL_SLAVE_RTU_MODE**
- When communicating at ASCII
  Serial Stack Mode： **MODBUS_SERIAL_SLAVE_ASCII_MODE**



2. Set UART
   Open “Properties” of “g_uart0 UART (r_sci_b_uart)” in “Stacks” and change “Data Bits” to the following values.

- When communicating at RTU
  Data Bits : **8 bits**
- When communicating at ASCII
  Data Bits : **7 bits**

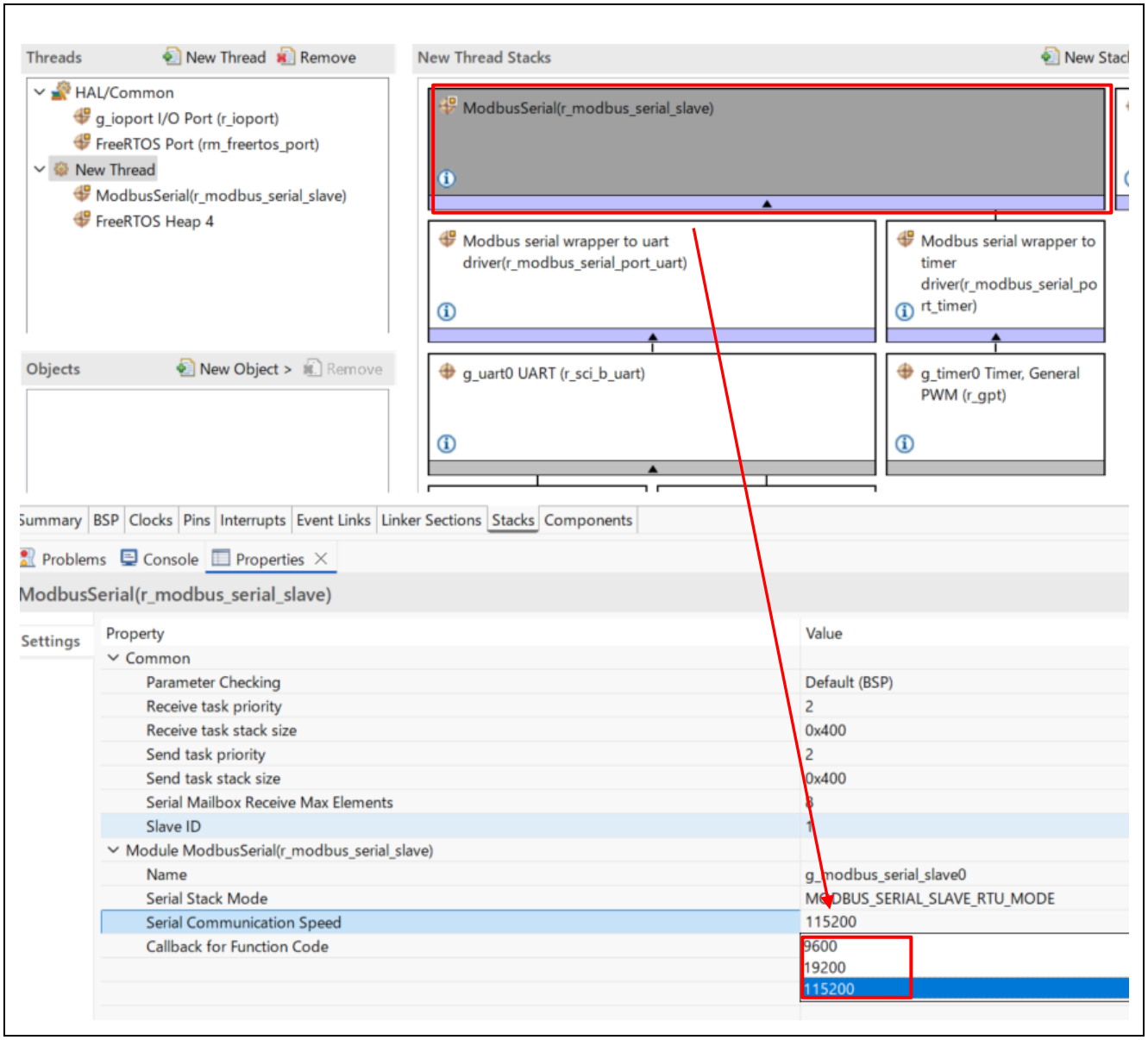


5.3.2 Settings the Baud Rate

This section describes the settings baud rate. Baud rate that can be set in the Modbus protocol stack is 9600 bps / 19200 bps / 115200 bps.

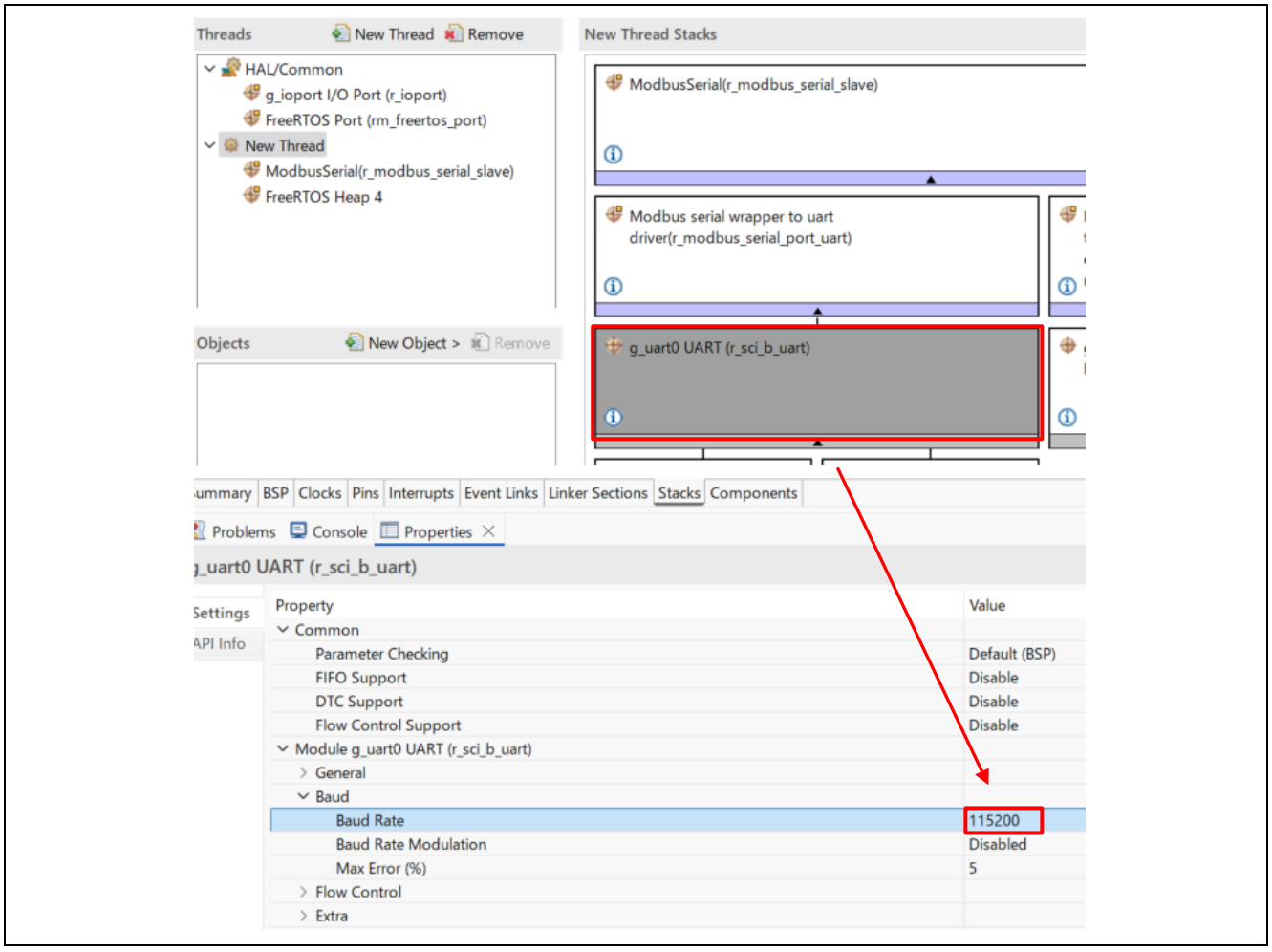
1. Set ModbusSerial
   Open “Properties” of “ModbusSerial(r_modbus_serial_slave)” in “Stacks” and change “Serial Communication Speed” to the following values.

- When communicating at 9600 bps
  Serial Communication Speed : **9600**
- When communicating at 19200 bps
  Serial Communication Speed : **19200**
- When communicating at 115200 bps
  Serial Communication Speed : **115200**



2. Set UART
   Open “Properties” of “g_uart0 UART (r_sci_b_uart)” in “Stacks” and change “Baud Rate” to the following values.

- When communicating at 9600 bps
  Baud Rate： **9600**
- When communicating at 19200 bps
  Baud Rate： **19200**
- When communicating at 115200 bps
  Baud Rate： **115200**

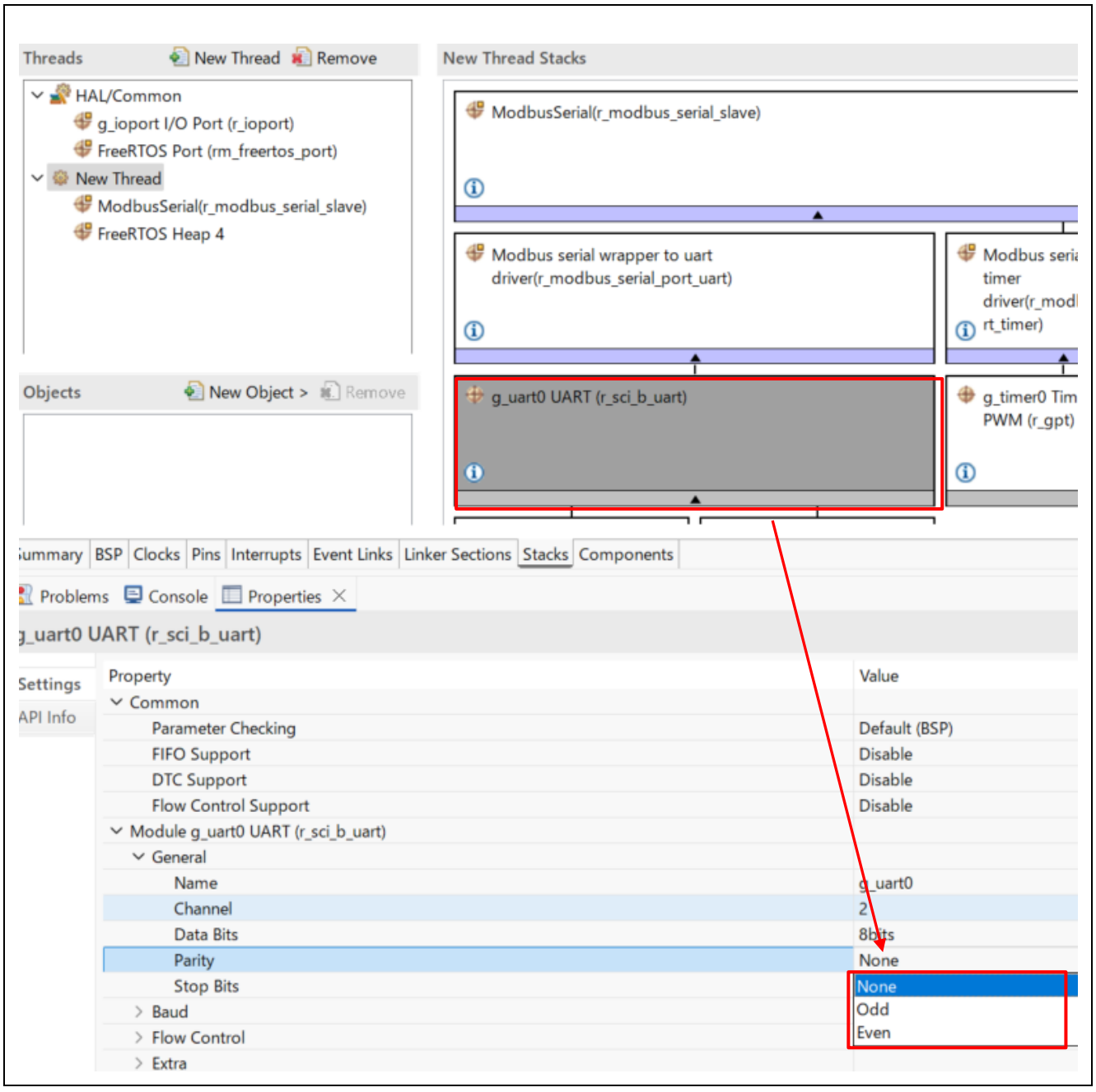


5.3.3 Settings the Parity Bit

This section describes the settings parity bit.

1. Set UART
   Open “Properties” of “g_uart0 UART (r_sci_b_uart)” in “Stacks” and change “Parity” to the following values.

- When communicating none parity
  Parity： **None**
- When communicating odd parity
  Parity： **Odd**
- When communicating even parity
  Parity： **Even**

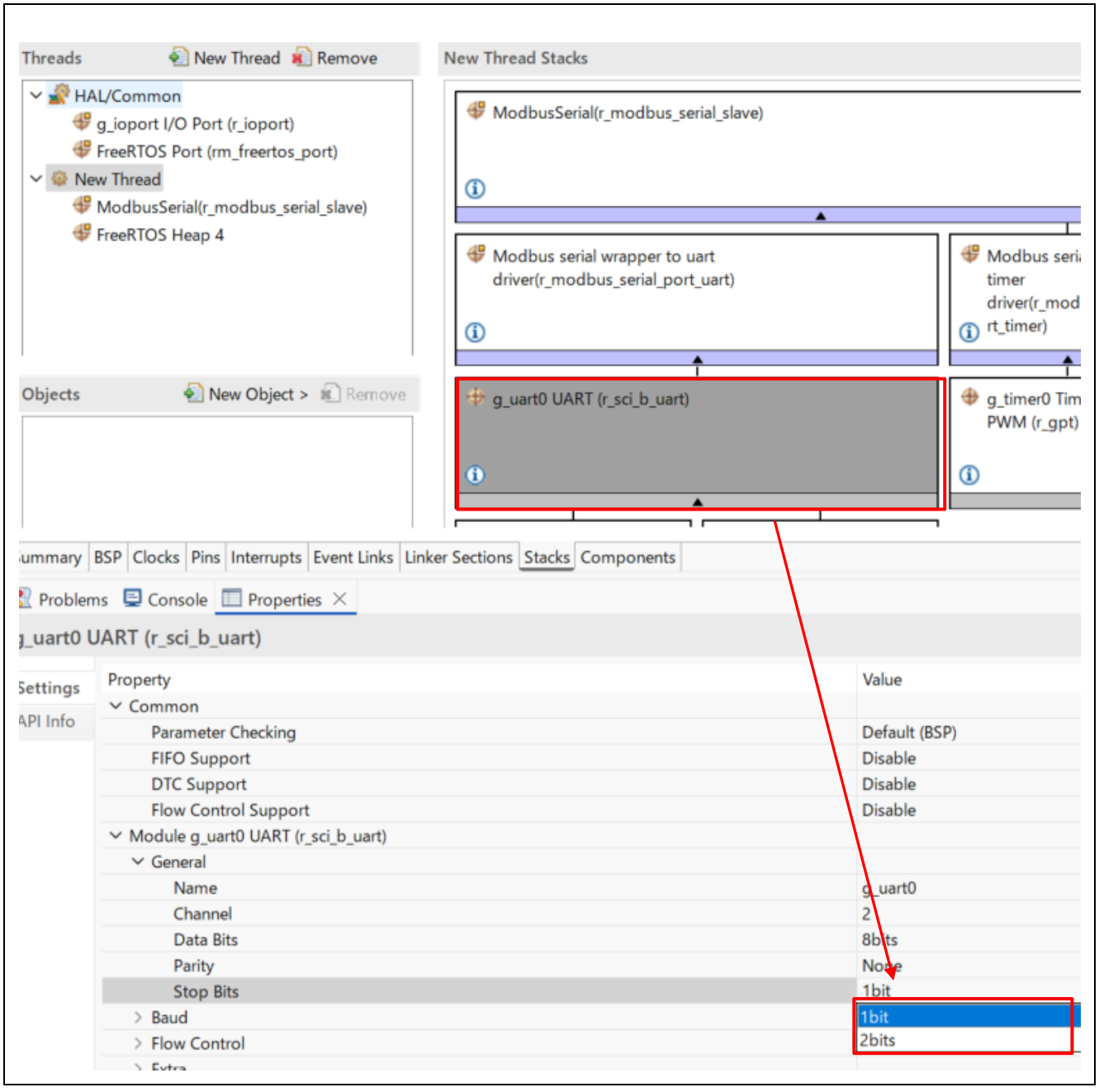


5.3.4 Settings the Stop Bits

This section describes the settings stop bits.

1. Set UART
   Open “Properties” of “g_uart0 UART (r_sci_b_uart)” in “Stacks” and change “Stop Bits” to the following values.

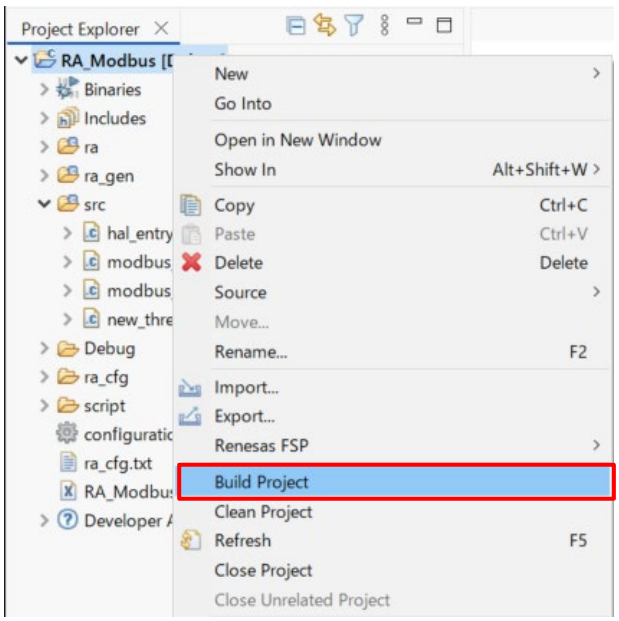
- When communicating 1 stop bit
  Stop Bits : **1 bit**
- When communicating 2 stop bits
  Stop Bits : **2 bits**



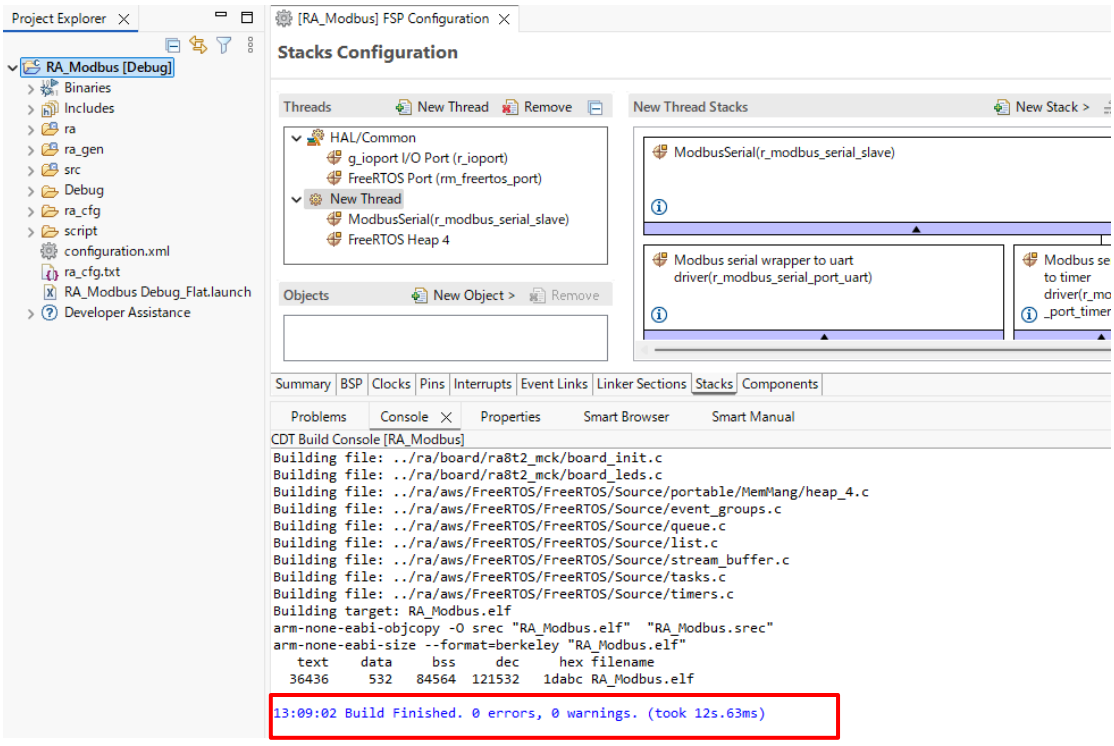
6. Execution of Modbus Sample Project

Before you begin, read "**4.2 Setting the Board**" to complete the hardware connections. Also, read "**5. Setting Up the Modbus Sample Project**" to complete the preparation of the Modbus sample project.

1. Build the project
   Right-click on the project from the “Project Explorer”, then select “Build Project”.

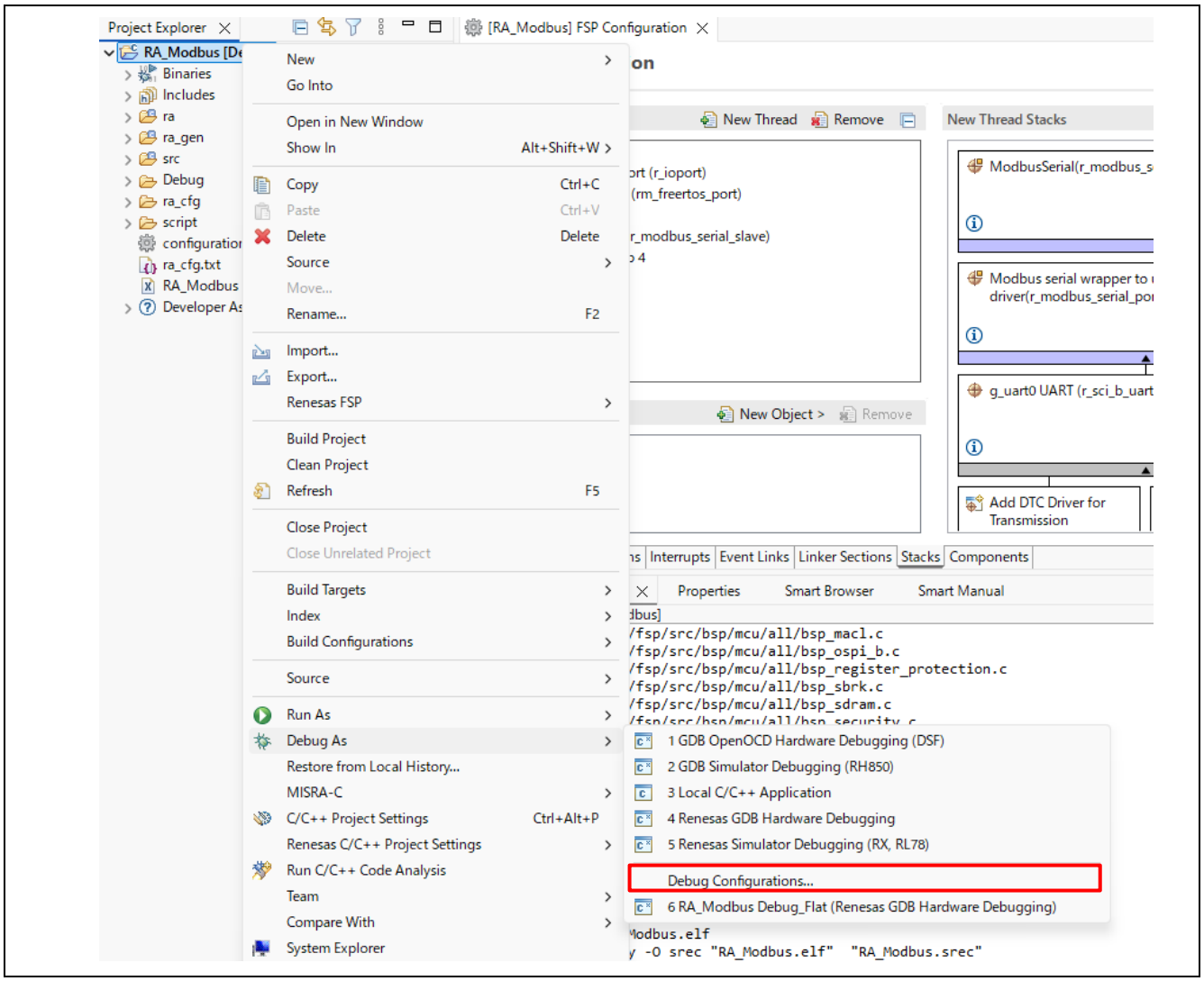


At this time, make sure there are no build errors.



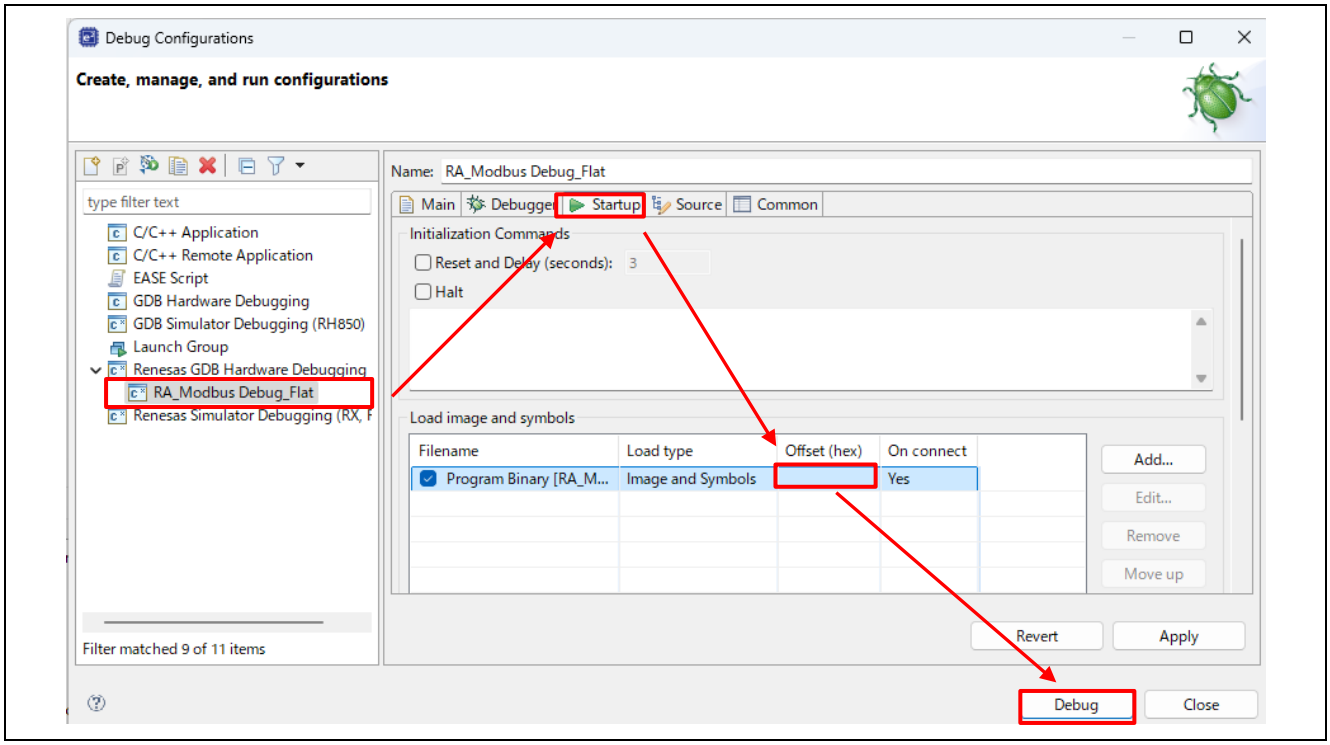
2. Download the application and run the debugger.
   To start debugging, follow the steps below:

   In “Project Explorer” view, right-click on the project to be debugged and select “Debug As” → “Debug Configurations”.



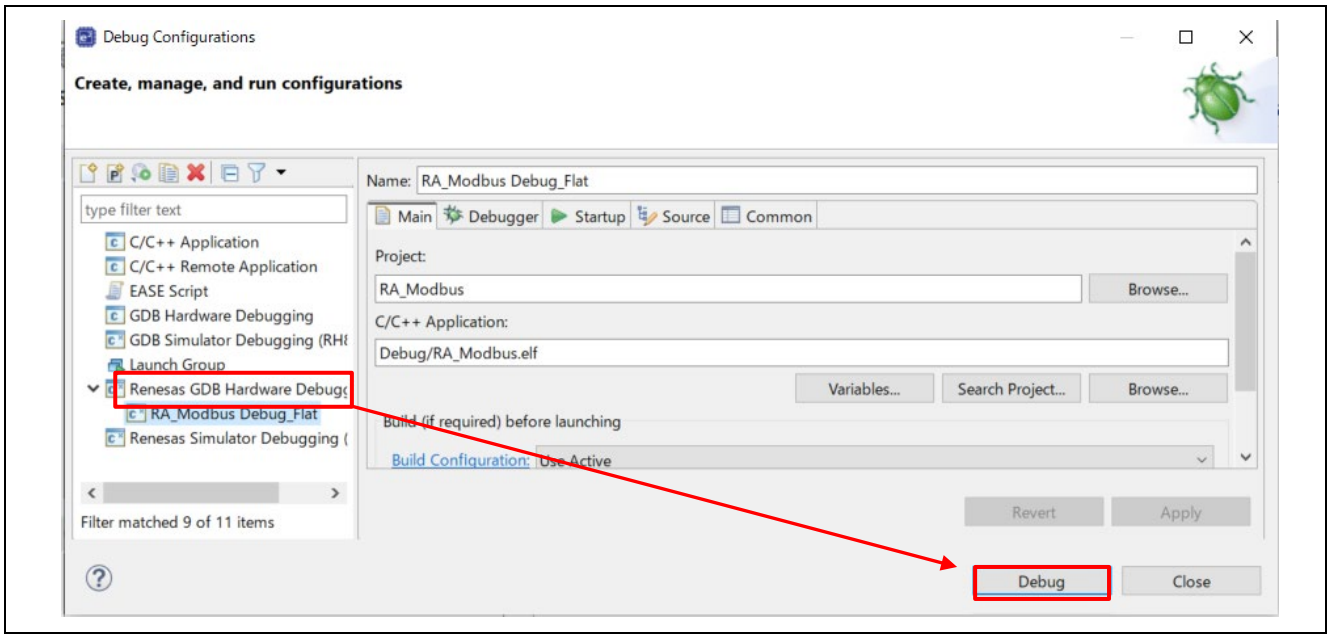
3. Program download.
When toolchain is the Arm Compiler:

Select “Renesas GDB Hardware Debugging” → “RA_Modbus Debug_Flat”, then select “Startup” tab.
Remove the “Offset (hex)” value "0" for “Load image and symbols” and click “Debug”.

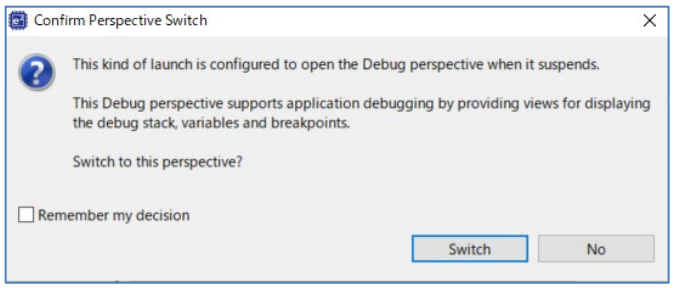


When toolchain is not the Arm Compiler:

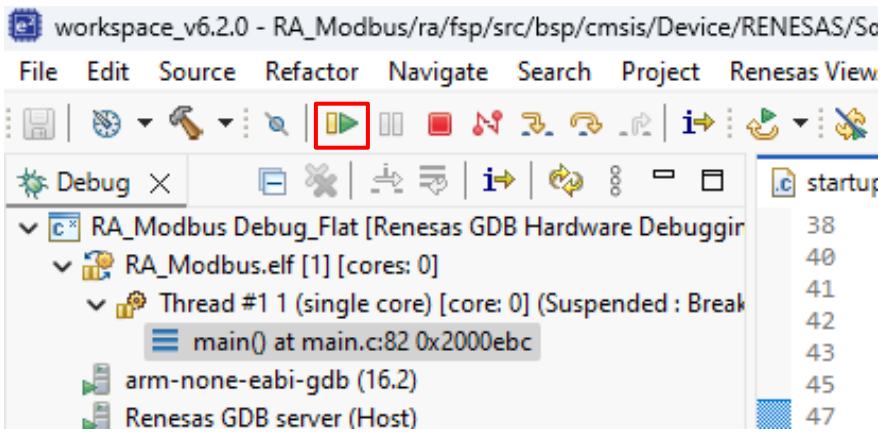
Select “Renesas GDB Hardware Debugging” → “RA_Modbus Debug_Flat”, then click “Debug”.



The following dialog appears, switch to the debug screen.



4. Program starts.
   Click the **Resume** button.
   When debugging starts, the program is suspended at main.c.
   Click the **Resume** button again to run the program.



7. Modbus Communication Using Modbus Demo Application

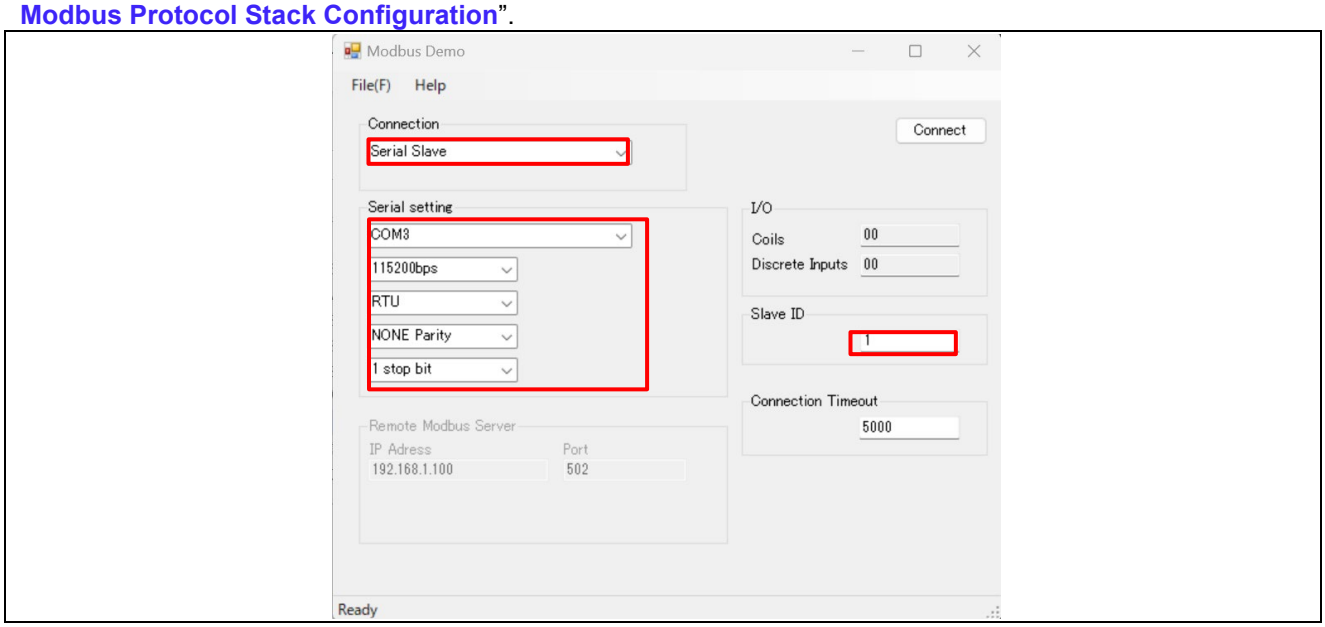
This section demonstrates the procedure for checking the demo operation of the Modbus sample application using the Modbus demo application. For information on configuring the Modbus protocol stack, see "**8.1. Appendix A: Modbus Protocol Stack Configuration**".

7.1 Setting up of Modbus Demo Application

Launch "ModbusDemoApplication.exe" included in this package and configure the following settings.

Connection : **Serial Slave**
Serial setting :
Serial Port No (for example, “COM3”)
Baud Rate (for example, ”115200bps”)
Communication mode (for example,” RTU”)
Parity (for example,” NONE Parity”)
Stop Bit (for example, ”1 stop bit”)
Slave ID : Slave ID (for example, “1”).

Note:
The "ModbusDemoApplication.exe " settings should match the Modbus sample project settings listed in “**5.3 Settings the Serial Transmission Modes / Baud Rate / Parity Bit / Stop Bits**" and "**8.1. Appendix A: Modbus Protocol Stack Configuration**”.



7.2 Modbus Demo Application Specification

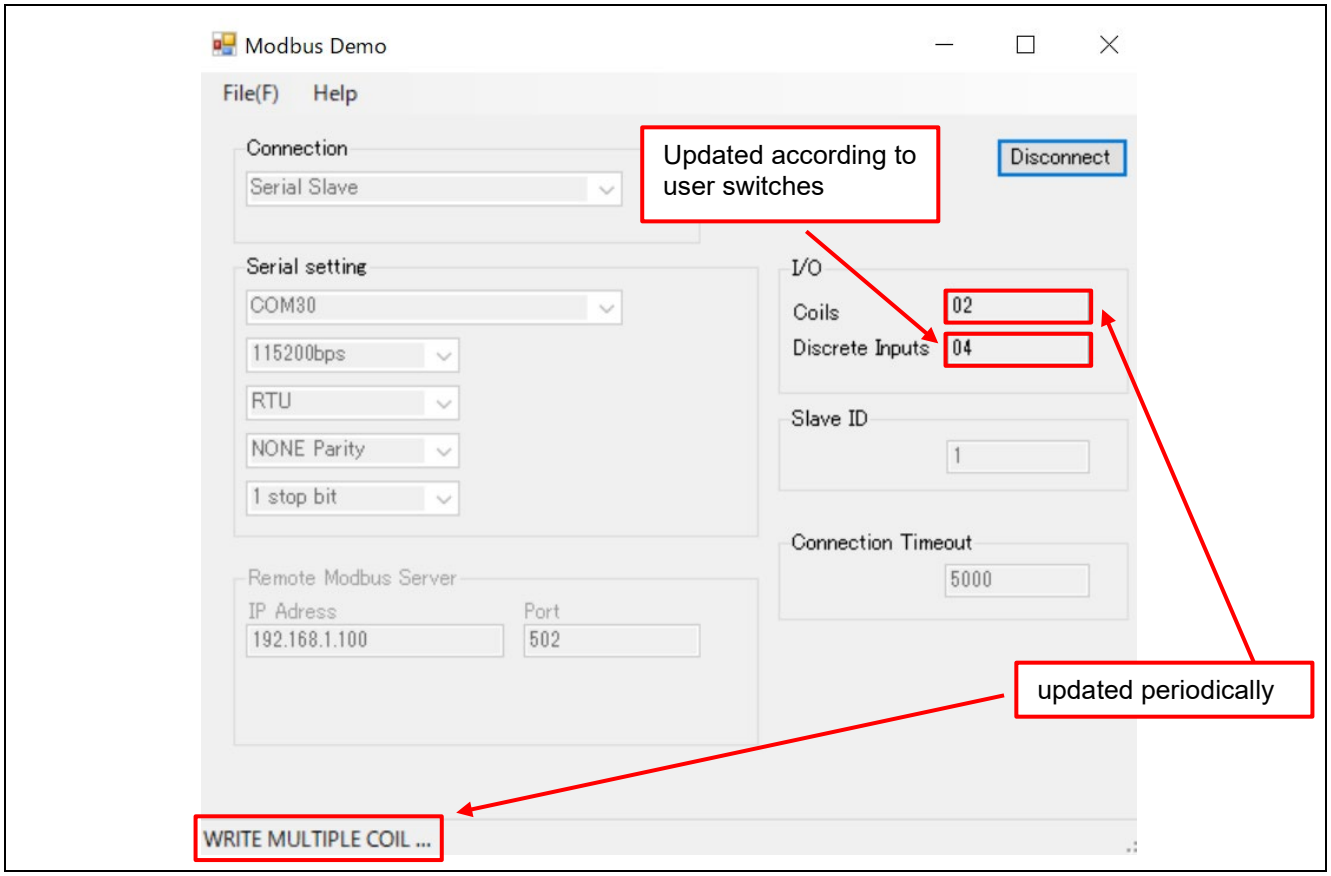
Users can see the simple demonstration using this Modbus protocol stack in this sample project.

By communicating with PC through the Modbus RTU / ASCII protocol, LED blinking speed is controlled dynamically.

For this control, "Read_Discrete_Inputs" and "Write_Single_Coil" function codes are used.

After clicking the "Connect" button, LEDs 1 to 3 will blink periodically.

"Discrete Inputs" are updated depending on whether the user switch listed in "**8.3.2 User-defined Functions**" is ON or OFF.



8. Appendix

8.1 Appendix A: Modbus Protocol Stack Configuration

The Modbus protocol stack configuration is described below.

Configuration	Options	Default	Description
Common			
Parameter Checking	・Default (BSP) ・Enable ・Disable	Default (BSP)	If selected code for parameter checking is included in the build.
Receive task priority	Legal values range from 0 through (Max Priorities - 1)	2	Receive task priority. Legal values range from 0 through (Max Priorities - 1).
Receive task stack size	Legal values are 0x400 or higher	0x400	Receive task stack size. Legal values are 0x400 or higher.
Send task priority	Legal values range from 0 through (Max Priorities - 1)	2	Send task priority. Legal values range from 0 through (Max Priorities - 1).
Send task stack size	Legal values are 0x400 or higher	0x400	Send task stack size. Legal values are 0x400 or higher.
Serial Mailbox Receive Max Elements	Legal values are 8 or higher	8	The length of the queue that passes data between tasks.
Slave ID	Legal values are from 1 to 247	1	The Modbus standard defines slave ID as 1-247.
Module ModbusSerial(r_modbus_serial_slave)			
Name	Name Must Be a Valid C Symbol	g_modbus_serial_slave0	Module name.
Serial Stack Mode	・MODBUS_SERIAL_SLAVE_RTU_MODE ・MODBUS_SERIAL_SLAVE_ASCII_MODE	MODBUS_SERIAL_SLAVE_RTU_MODE	Mode in which the MODBUS serial stack should work in as RTU / ASCII Slave. Select the desired stack mode.
Serial Communication Speed	・9600 ・19200 ・115200	115200	UART communication speed.
Callback for Function Code	Name must be a valid C symbol	function_code_callback	Please enter the user callback function name for "Function Code".

8.2 Appendix B. Application Programming Interface

Function

· **R_MODBUS_SERIAL_SLAVE_Open**

Description	MODBUS serial slave stack is initialized with this function.	
Format	modbus_err_t R_MODBUS_SERIAL_SLAVE_Open (modbus_serial_slave_handle_t * const p_handle, modbus_serial_slave_cfg_t const * const p_cfg)	
Parameter	modbus_serial_slave_handle_t * const p_handle	Control block. Allocate an instance specific control block to pass into the Modbus Serial API calls.
	modbus_serial_slave_cfg_t const * const p_cfg	Modbus Serial Configuration parameters.
Return value	Error code	
Error code	MODBUS_ERR_OK	MODBUS Serial Slave stack initialized successfully.
	MODBUS_ERR_ALREADY_OPEN	MODBUS Serial Slave stack has already been initialized.
Example	R_MODBUS_SERIAL_SLAVE_Open(&g_modbus_serial_slave0_ctrl, &g_modbus_serial_slave0_cfg);	

· **R_MODBUS_SERIAL_SLAVE_Close**

Description	MODBUS serial slave stack is terminated with this function.	
Format	modbus_err_t R_MODBUS_SERIAL_SLAVE_Close (modbus_serial_slave_handle_t * const p_handle)	
Parameter	modbus_serial_slave_handle_t * const p_handle	Control block. Allocate an instance specific control block to pass into the Modbus Serial API calls.
Return value	Error code	
Error code	MODBUS_ERR_OK	MODBUS Serial Slave stack terminated successfully.
	MODBUS_ERR_STACK_TERM	MODBUS Serial Slave stack terminated failure.
	MODBUS_ERR_NOT_OPEN	MODBUS Serial Slave stack is not initialized.
Example	R_MODBUS_SERIAL_SLAVE_Close(&g_modbus_serial_slave0_ctrl);	

8.3 Appendix C: User-defined function

This section describes Modbus sample application. Users can register their own implementation of Modbus function code with Modbus protocol stack.

8.3.1 Register Function Code

Definition file: src/modbus_func.c
Define the function to be registered in call back function of Modbus protocol stack.

8.3.2 User-defined Functions

User-defined functions are defined in the src/modbus_user.c
User-defined Read/Write functions are used to process each function.

Some functions access the following components of the evaluation board.

Evaluation Boards	User LED			User-Switch	
	①	②	③	①	②
MCK-RA8T2	LED1	LED2	LED3	-	-
EK-RA8M2	User LED1	User LED2	User LED3	User-Switch SW1	User-Switch SW2
EK-RA8T2	User LED1	User LED2	User LED3	User-Switch SW1	User-Switch SW2

Read/Write functions and tables are provided that correspond to each address of the coil/discrete input/holding register/input register.
The evaluation board parts and global variables accessed by the functions in each table are as follows.

【Read Coils】	
address	**Access**
0001	Above table User - LED①, g_coils_area
0002	Above table User - LED②, g_coils_area
0003	Above table User - LED③, g_coils_area
0004	g_coils_area
0005	g_coils_area
0006	g_coils_area
0007	g_coils_area
0008	g_coils_area

【Write_Single_Coils】	
address	**access**
0001	Above table User - LED①, g_coils_area *1
0002	Above table User - LED②, g_coils_area *1
0003	Above table User - LED③, g_coils_area *1
0004	g_coils_area
0005	g_coils_area
0006	g_coils_area
0007	g_coils_area
0008	g_coils_area

*1：Each LED on the evaluation board turns on/off depending on the value written by this function.

【Read_Discrete_Inputs】	
address	**access**
1001	g_discrete_input_area
1002	g_discrete_input_area
1003	g_discrete_input_area
1004	g_discrete_input_area
1005	g_discrete_input_area
1006	Above table User-Switch①, g_discrete_input_area *2
1007	Above table User-Switch②, g_discrete_input_area *2
1008	g_discrete_input_area
1009	g_discrete_input_area
10010	ILLEGAL DATA ADDRESS
10011	g_discrete_input_area
10012	g_discrete_input_area

*2：The value read by this function is displayed in “Discrete Inputs” of the Modbus demo application.

【Read_Discrete_Inputs】	
address	**access**
3001	g_input_reg_area
3002	g_input_reg_area
3003	g_input_reg_area
3004	ILLEGAL DATA ADDRESS
3005	ILLEGAL DATA ADDRESS
3006	ILLEGAL DATA ADDRESS
3007	ILLEGAL DATA ADDRESS
3008	g_input_reg_area

【READ_HOLDING_REGISTERS】	
address	**access**
4001	g_holding_reg_area
4002	g_holding_reg_area
4003	g_holding_reg_area
4004	ILLEGAL DATA ADDRESS
4005	ILLEGAL DATA ADDRESS
4006	ILLEGAL DATA ADDRESS
4007	g_holding_reg_area

【WRITE_SINGLE_REGISTER】	
address	**access**
4001	g_holding_reg_area
4002	g_holding_reg_area
4003	g_holding_reg_area
4004	ILLEGAL DATA ADDRESS
4005	ILLEGAL DATA ADDRESS
4006	ILLEGAL DATA ADDRESS
4007	g_holding_reg_area

9. Limitations

No Limitations

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.26.2025	-	First Edition issued
1.10	Mar.31.2026		Added EK-RA8T2
		Page 3	Updated Table 1.2
		Page 5	Updated 3.1 Modbus Sample Project
		Page 9	Added Figure 4.3
		Page 11	Added steps to execute the Modbus sample project
		Page 28 - 33	Added 5.2.3 EK-RA8T2 Creating Procedures
			Updated Operating Environment Requirements
		Page 6	Updated Table 4.1
		Page 44	Change the description of the settings
		Page 45	Added information about Discrete Inputs
		Page 48	Updated the User LED and User Switch tables

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/.