Renesas RA ファミリ

クイックスタートガイド：Modbus Serial

要旨

本書は、RA マイコン評価ボードを使用して Modbus® 通信を評価するためのクイックスタートガイドです。
Modbus プロトコルは、Modicon Inc. (Schneider Electric SA.) がプログラマブル ロジックコントローラー (PLC) 向けに開発した通信プロトコルであり、その仕様は公開されています。
詳細についてはプロトコル仕様書（PI-MBUS-300 Rev.J）を参照してください。

対象デバイス

RA8T2, RA8M2

サポート対象評価ボード

MCK-RA8T2, EK-RA8M2, EK-RA8T2

目次

1. 概要

本書は、RA ボード上で動作する Modbus プロトコルスタックを対象としており、プロトコルスタックを使用してアプリケーションを開発および実装するための機能の概要、アプリケーションプログラミングインターフェイス(API)、およびアプリケーションサンプルについて説明しています。

このパッケージは、RS-485 ベースの Modbus RTU/ASCII プロトコルをサポートしています。

1.1 略語/定義

表 1-1 略語/定義

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 参考資料

Modbus に関する技術情報は Modbus 組織サイトで確認してください。RA 評価ボードに関する情報はルネサスサイトから入手できます。

表 1-2 技術資料

Index	Technical Inputs
1	Modbus Application Protocol Specification Vxxx
2	MCK-RA8T2 クイックスタートガイド / r12qs0088jjxxxx
3	MCK-RA8T2 ユーザーズマニュアル / r12uz0172jjxxxx
4	RA8M2 MCU グループ用評価キット EK-RA8M2 クイックスタートガイド / r20qs0069jgxxxx
5	RA8M2 MCU グループ用評価キット EK-RA8M2 v1 ユーザーズマニュアル / r20ut5451jgxxxx
6	Evaluation Kit for RA8T2 Microcontroller Group EK-RA8T2 v1 Quick Start Guide / r20qs0097egxxxx
7	RA8T2 MCU グループ用評価キット EK-RA8T2 v1 ユーザーズマニュアル / r20ut5714jgxxxx

2. 特徴

RA 用の Modbus プロトコルスタックにより、Modbus RTU/ASCII アプリケーションを迅速かつ簡単に開発が可能になります。 このスタックには、次の 9 つのコードを実装できます。

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

Modbus について詳細は以下のサイトをご覧ください。
http://www.modbus.org

* アップデートによりバージョン番号が異なる場合があります。 最新のマニュアルをご参照ください。

3. サンプルプログラムパッケージ構成

このサンプルプログラムパッケージは 3 つのブロックで構成されています

- Modbus プロトコルスタックを使用したサンプルプロジェクト
- Modbus プロトコルスタックを使用したサンプルアプリケーション
- Modbus デモアプリケーション

3.1 Modbus サンプルプロジェクト

- Modbus_Serial / project / MCK-RA8T2
- Modbus_Serial / project / EK-RA8M2
- Modbus_Serial / project / EK-RA8T2
  - これらのフォルダには Modbus プロトコルスタックを使用した MCK-RA8T2 / EK-RA8M2 / EK-RA8T2 用サンプルプロジェクトの「RA_Modbus」フォルダが格納されています。
  - サンプルプロジェクトは GCC 用に作成されています。他のコンパイラを使用する場合は「**5.2 新規プロジェクトの作成**」を参照し、プロジェクトを作成してください。

3.2 Modbus サンプルアプリケーション

- Modbus_Serial / src / modbus_func.c, modbus_user.c, new_thread0_entry.c
  - ユーザーは、Modbus ファンクションコードの独自の実装を Modbus プロトコルスタックに登録できます。
  - このディレクトリ内のコードは、Modbus プロトコルスタックの初期化処理と Modbus プロトコルスタック API を使用した Modbus ファンクションコード処理の例です。

3.3 Modbus デモアプリケーション

- Modbus_tool / ModbusDemoApplication.exe
  - この実行ファイルは、Modbus 通信用のデモアプリケーションです。この実行ファイルを使用して、Modbus サンプルアプリケーションのデモ動作を実行できます。

4. プロジェクト設定

4.1 動作環境

本書のサンプルプログラムパッケージは以下の環境で動作します。

表 4-1 動作環境

項目	概要
Board	RA evaluation board
Integrated development environment	IAR Systems - IAR Embedded Workbench® for Arm Version 9.70.2以降 Renesas Electronics - e² Studio 2025-12 以降 - Renesas RA Smart Configurator 2025-12 以降
Toolchain	IAR Embedded Workbench for Arm - IAR C/C++ Compiler for Arm 9.70.2 以降 e² Studio - GCC Arm Embedded (13.2.1.arm-13-7)以降 - LLVM for Arm (21.1.1)以降 - Arm Compiler 6.24 以降
MCU software package	FSP (Flexible Software Package) v6.4.0 以降
Emulator	J-LINK® OB
Communications protocol	Modbus RTU or Modbus ASCII
Client tool	ModbusDemoApplication.exe: Modbus デモアプリケーション

4.2　ボード設定

PC をサポート対象評価ボードに接続します。
電源は USB ケーブルをボードに接続することで供給されます。
Modbus シリアル通信の場合は RS485 コネクタを使用し、RS-485 ケーブルで PC に接続します。
MCK-RA8T2 ボード接続設定、EK-RA8M2 ボード接続設定、EK-RA8T2 ボード接続設定の図とスイッチ SW4 設定表を以下に記述します。

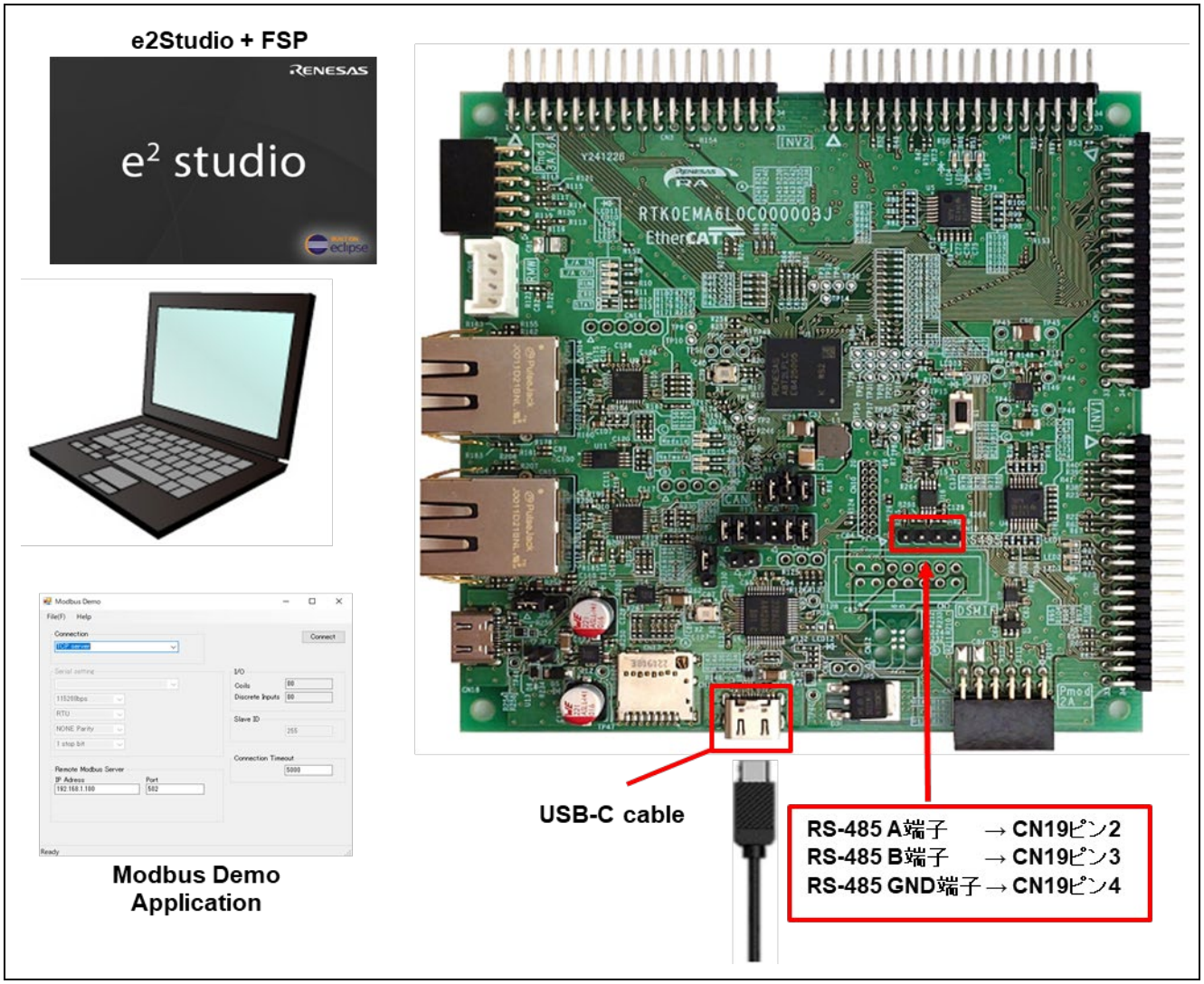


図 4.1: MCK-RA8T2 ボード接続設定

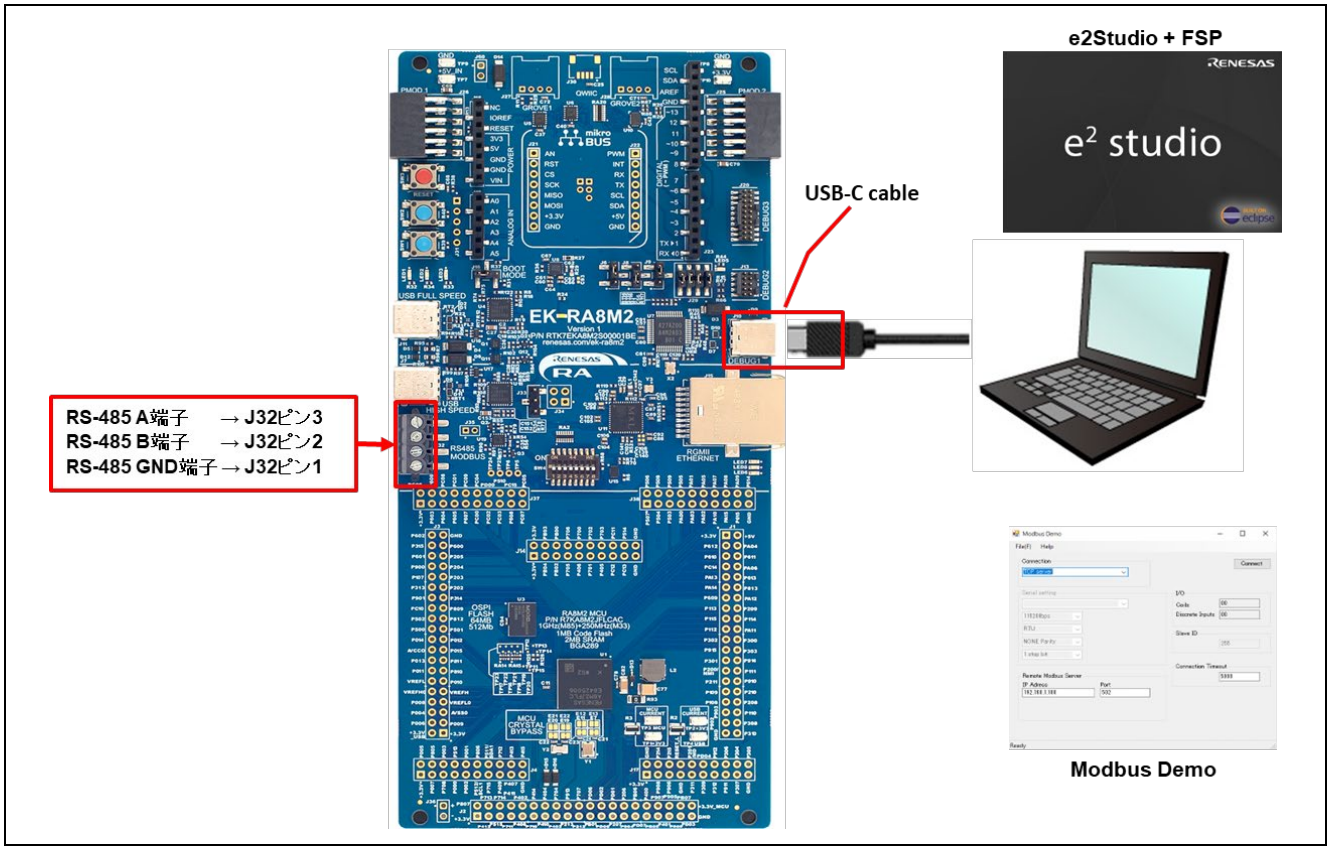


図 4.2: EK-RA8M2 ボード接続設定

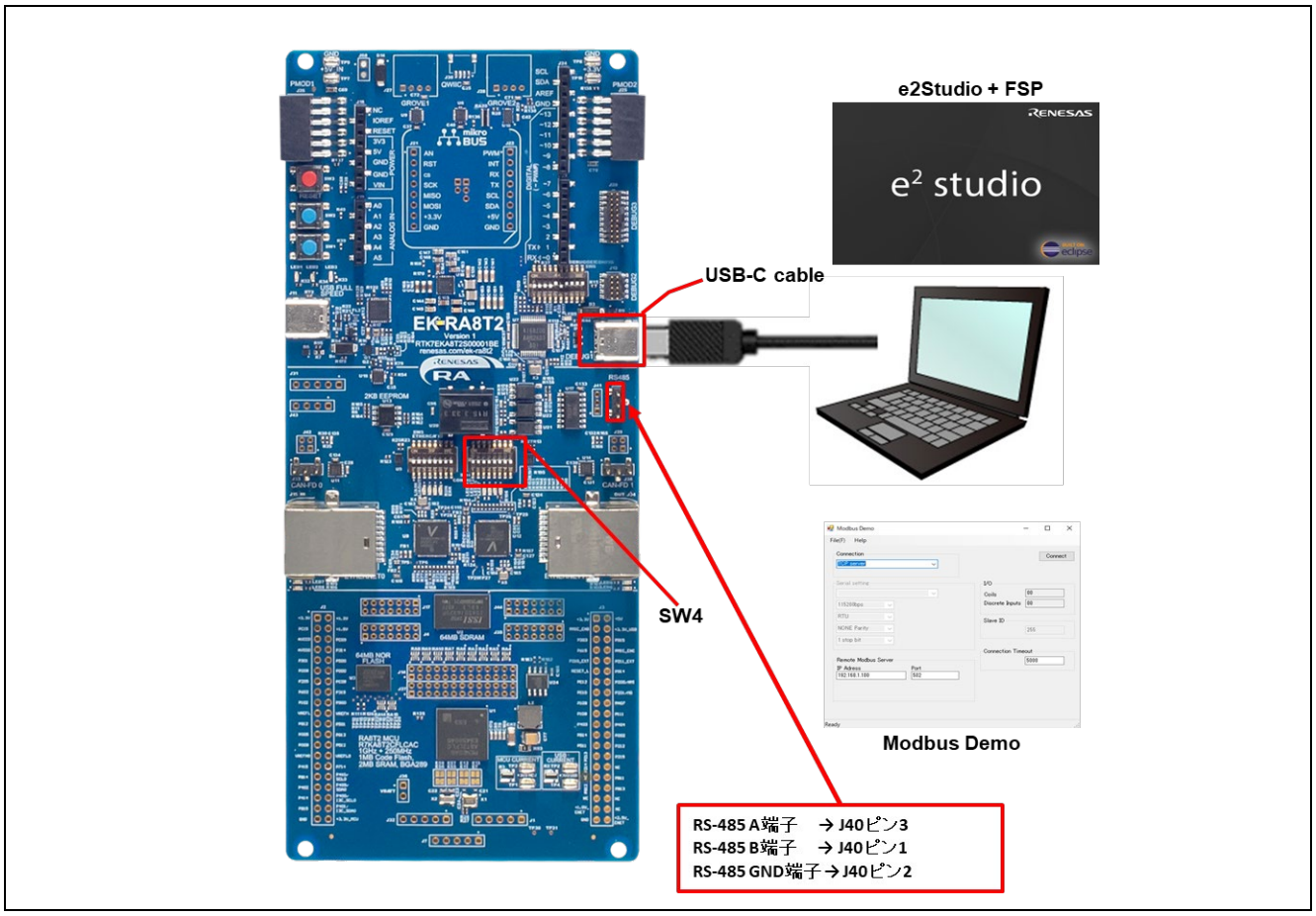


図 4.3: EK-RA8T2 ボード接続設定

表 4.1 スイッチ SW4 設定

SW4	Setting	Description
SW4-8	OFF	**RS485 が有効**

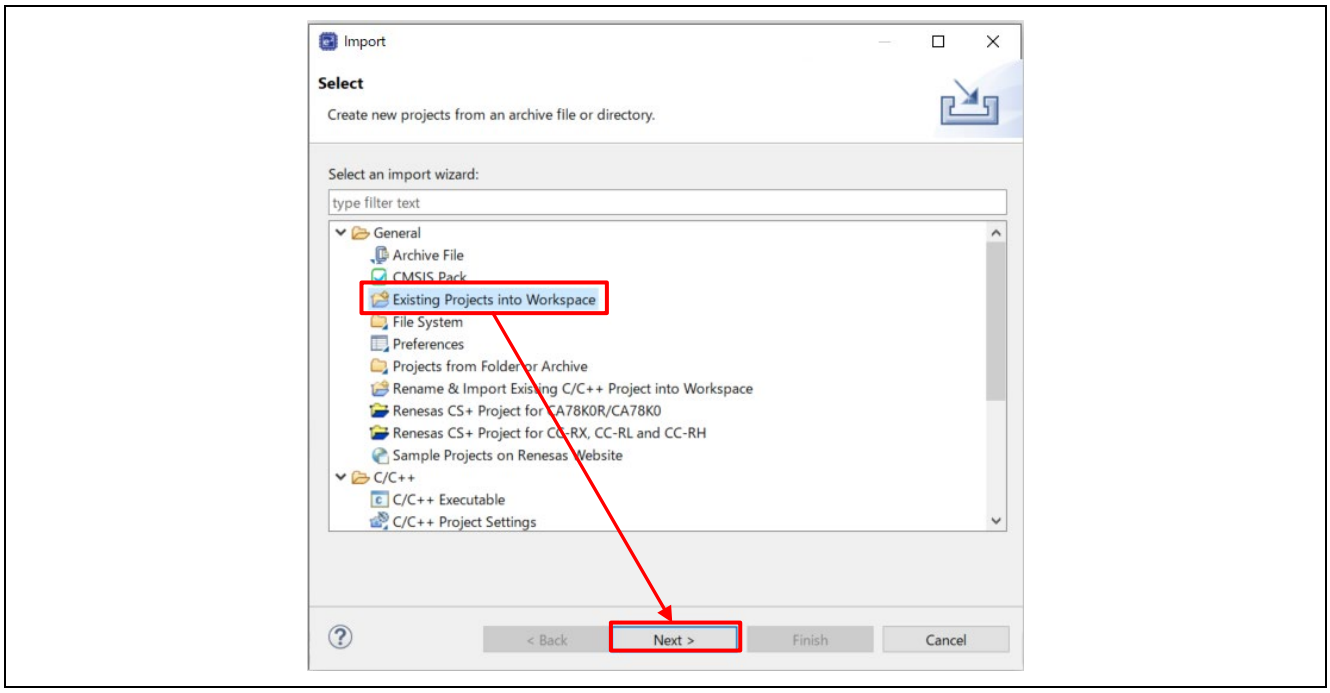
5. Modbus サンプルプロジェクトの構築

この章では、Modbus サンプルプロジェクトを構築する手順について説明します。
事前に「4.1 動作環境」を参照し、ツールのインストールを完了してください。

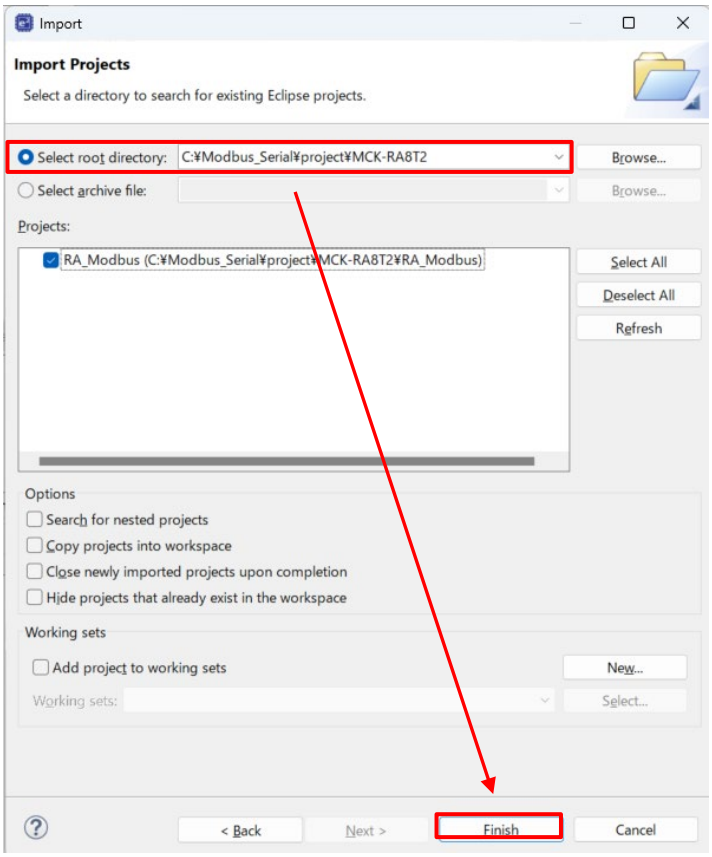
5.1 Modbus サンプルプロジェクトのインポート

この章では、Modbus サンプルプロジェクトのインポートおよび評価ボード変更の手順について説明します。

1. サンプルプロジェクトをインポートします。 e2 studio を起動し、[File] → [Import] を選択し、Import ウィンドウが表示されたら[General]→ [Existing Projects into Workspace] を選択します。



"Select root directory"にチェックを入れ、Modbus サンプルプロジェクトのフォルダ(”Modbus_Serial\project\[評価ボード名]”)を選択します。



2. Modbus サンプルプロジェクトを実行します。
「**6. Modbus サンプルプロジェクトの実行**」を参照し、手順を実行してください。

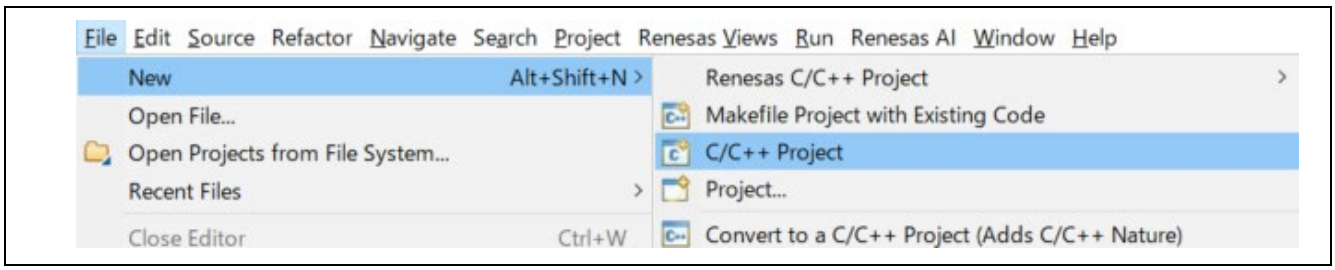
5.2　新規プロジェクトの作成

この章では、Modbus サンプルプロジェクトを新規作成する手順について説明します。

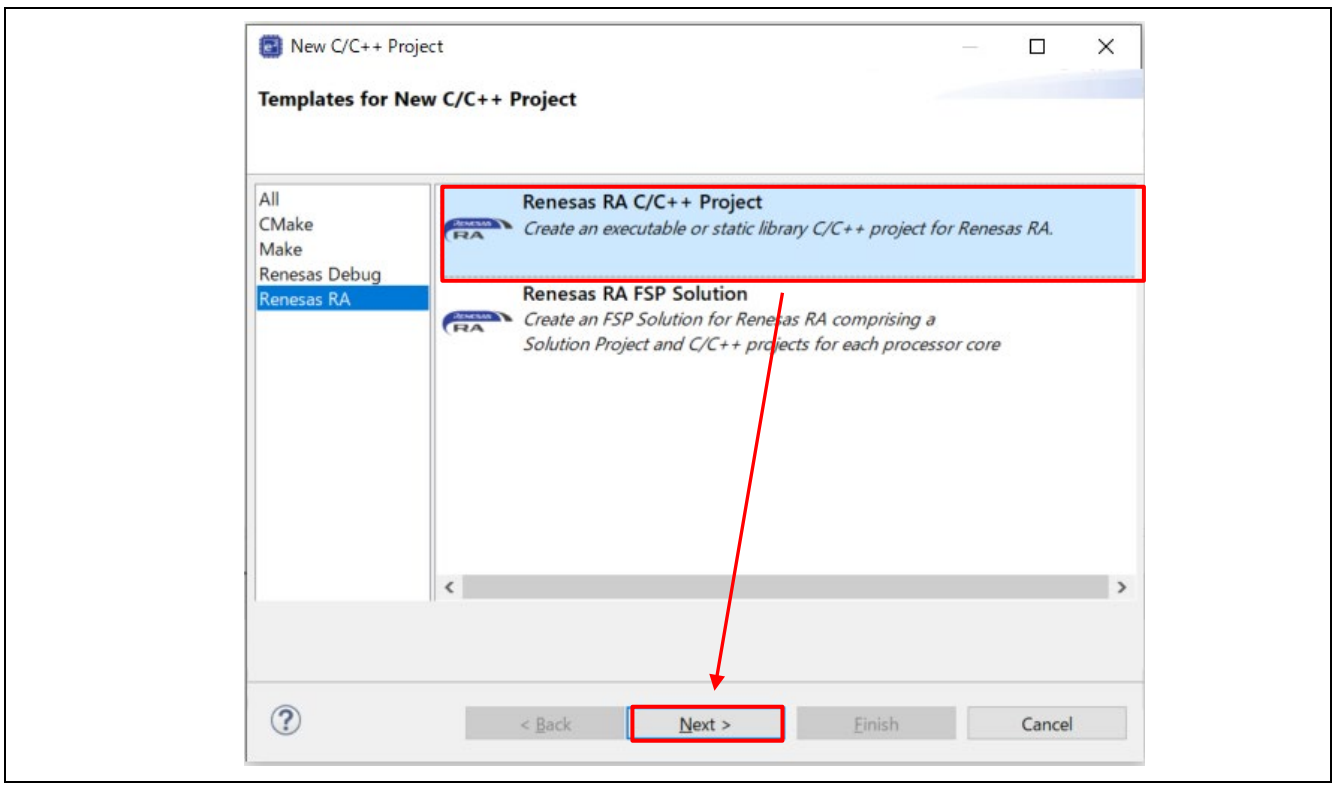
5.2.1　全評価ボード共通作成手順

全評価ボード共通の作成手順について説明します。

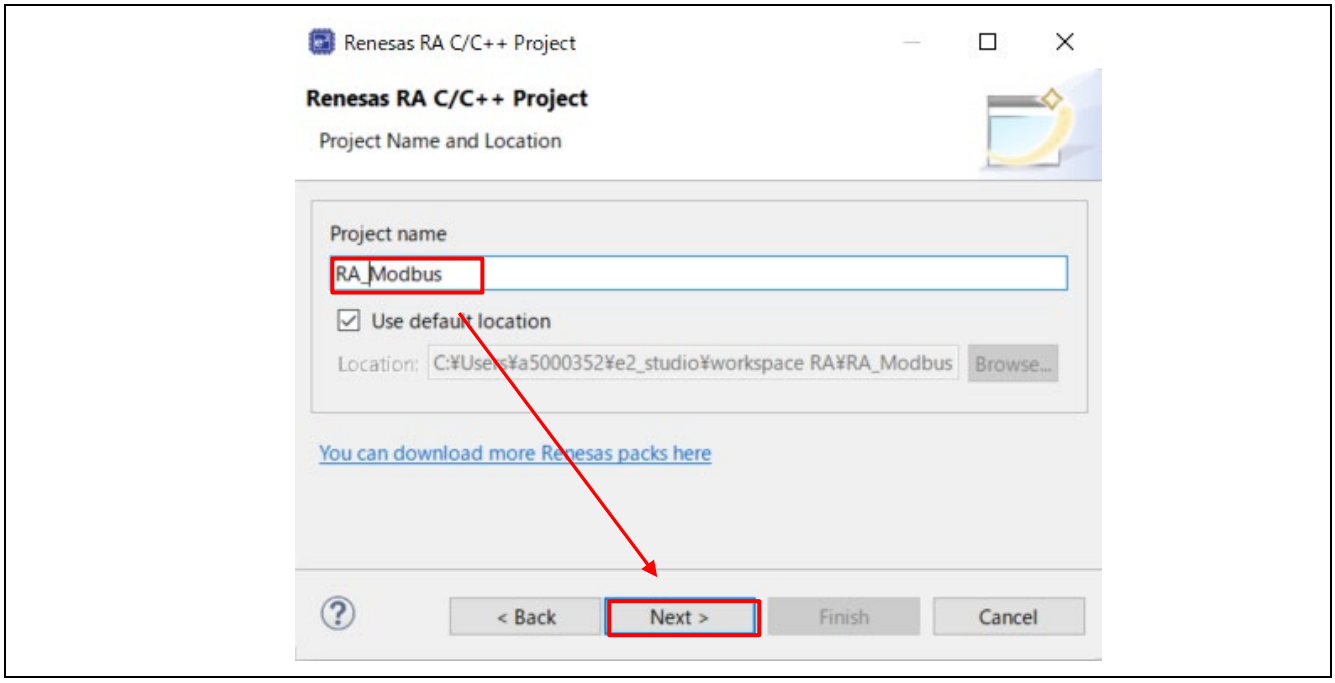
1. 新規プロジェクトを設定します。 e² studio を起動し、[File] → [New] → [C/C++ Project] →の順で選択します。



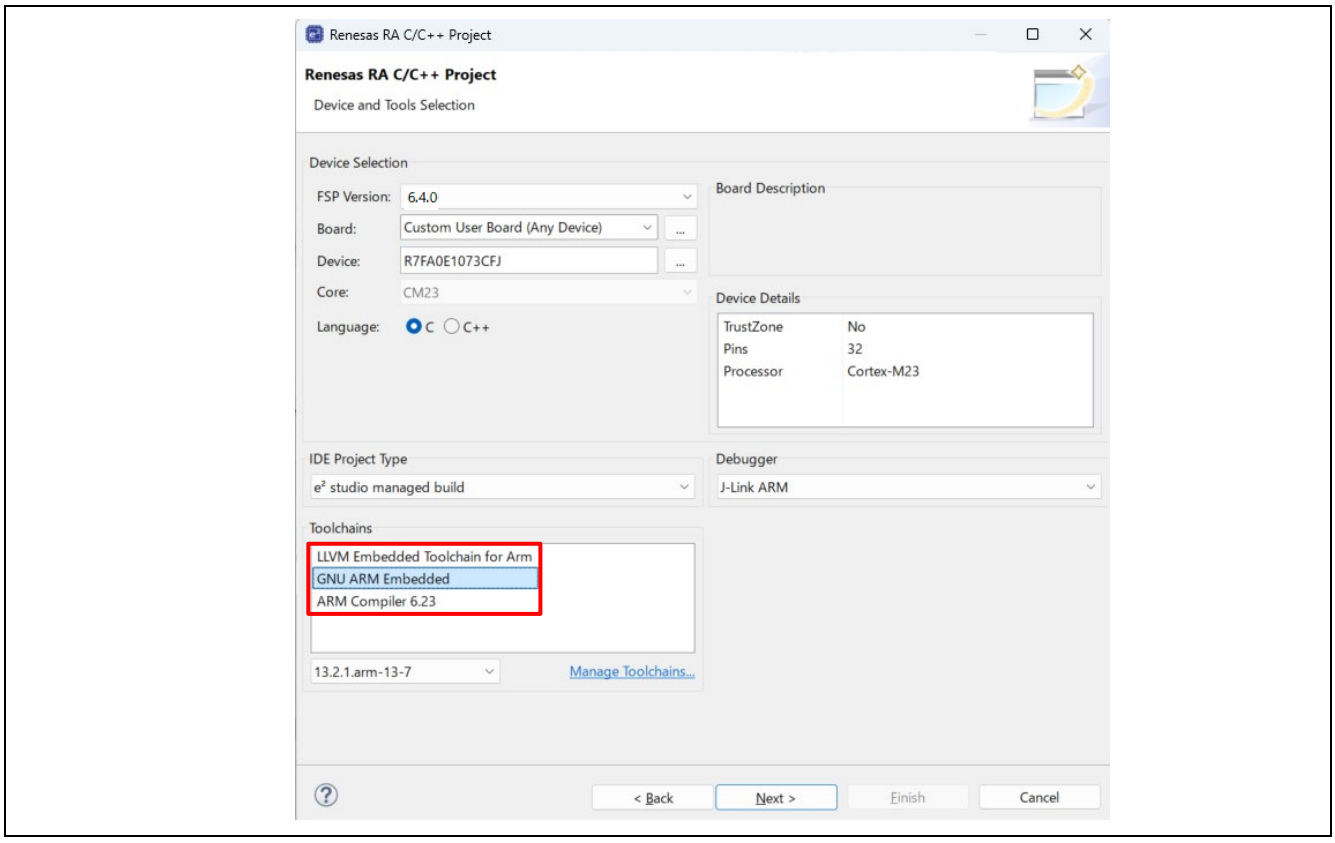
2. 「Renesas RA C/C++ Project」を選択します。



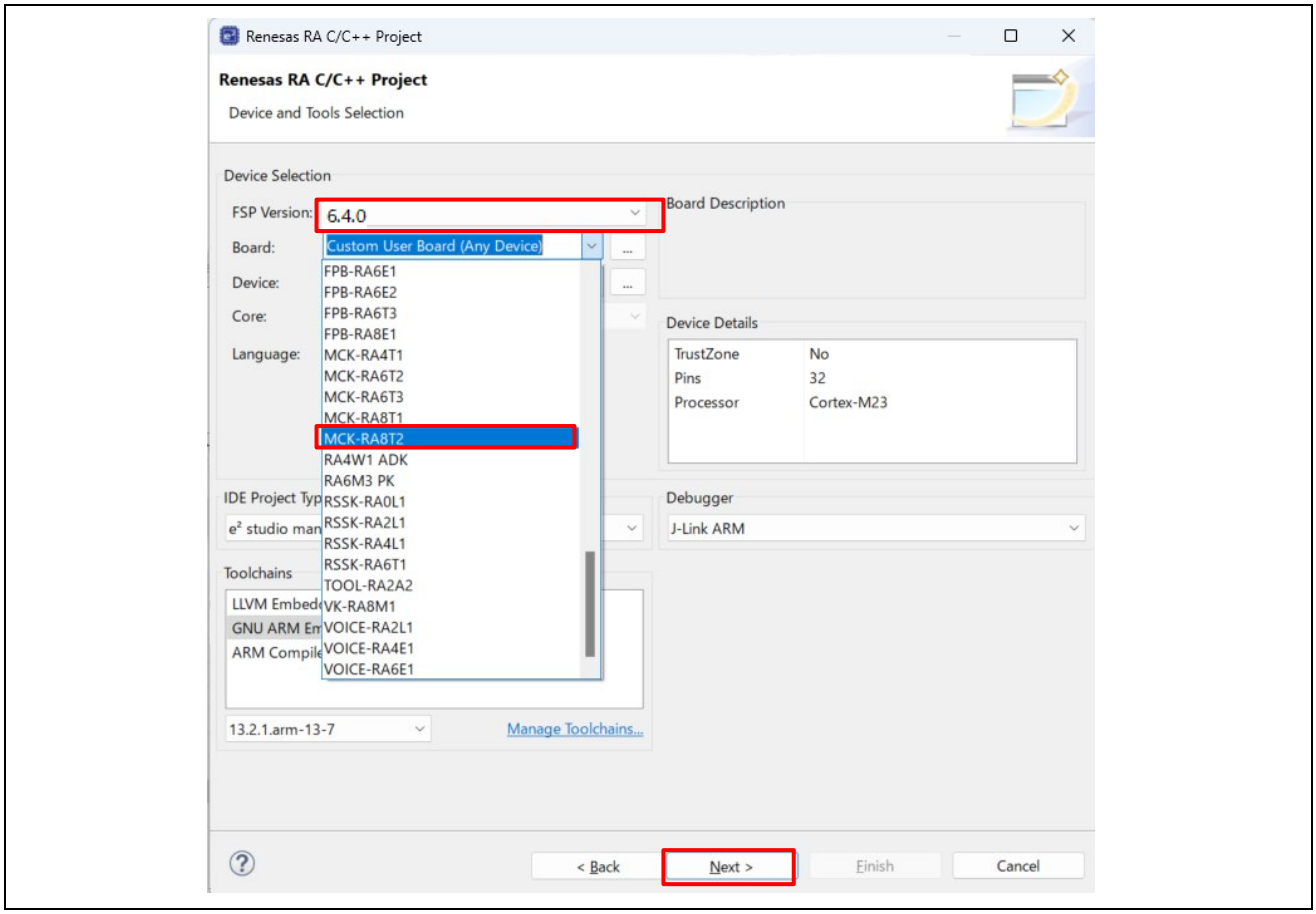
3. 任意のプロジェクト名を入力します。



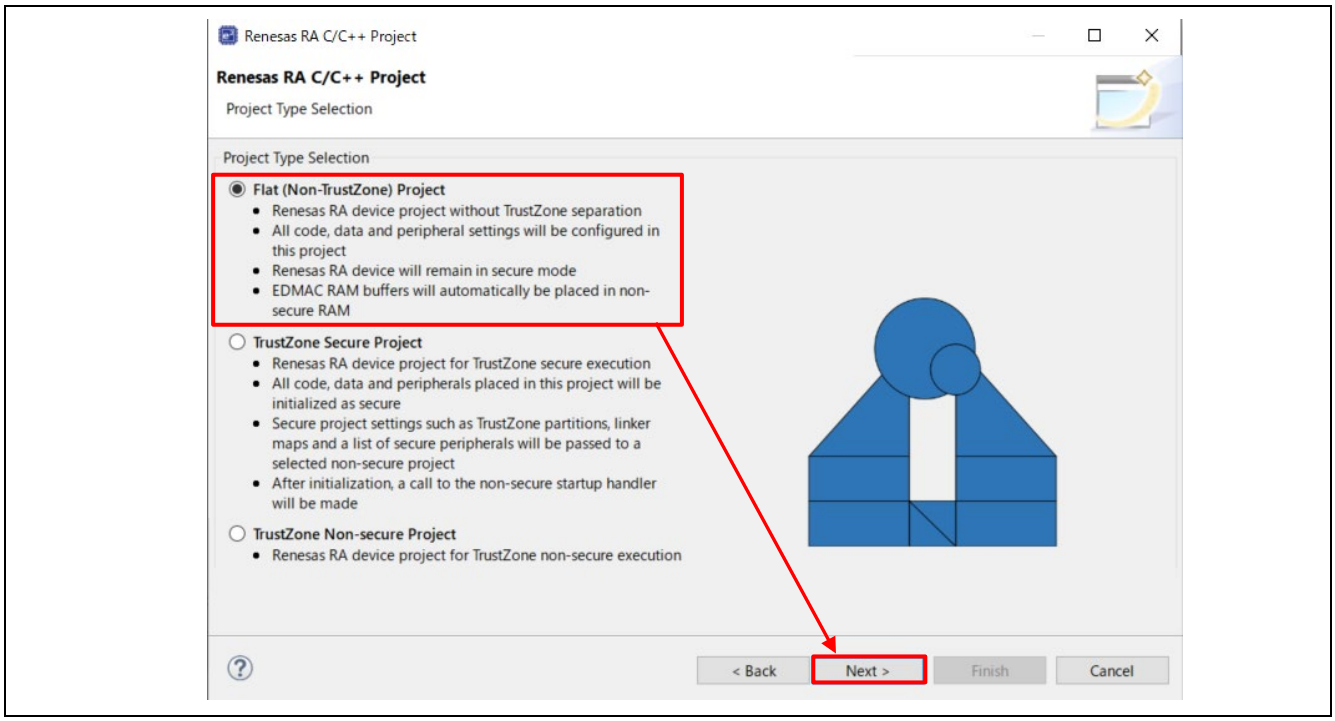
4. Toolchains を選択します。



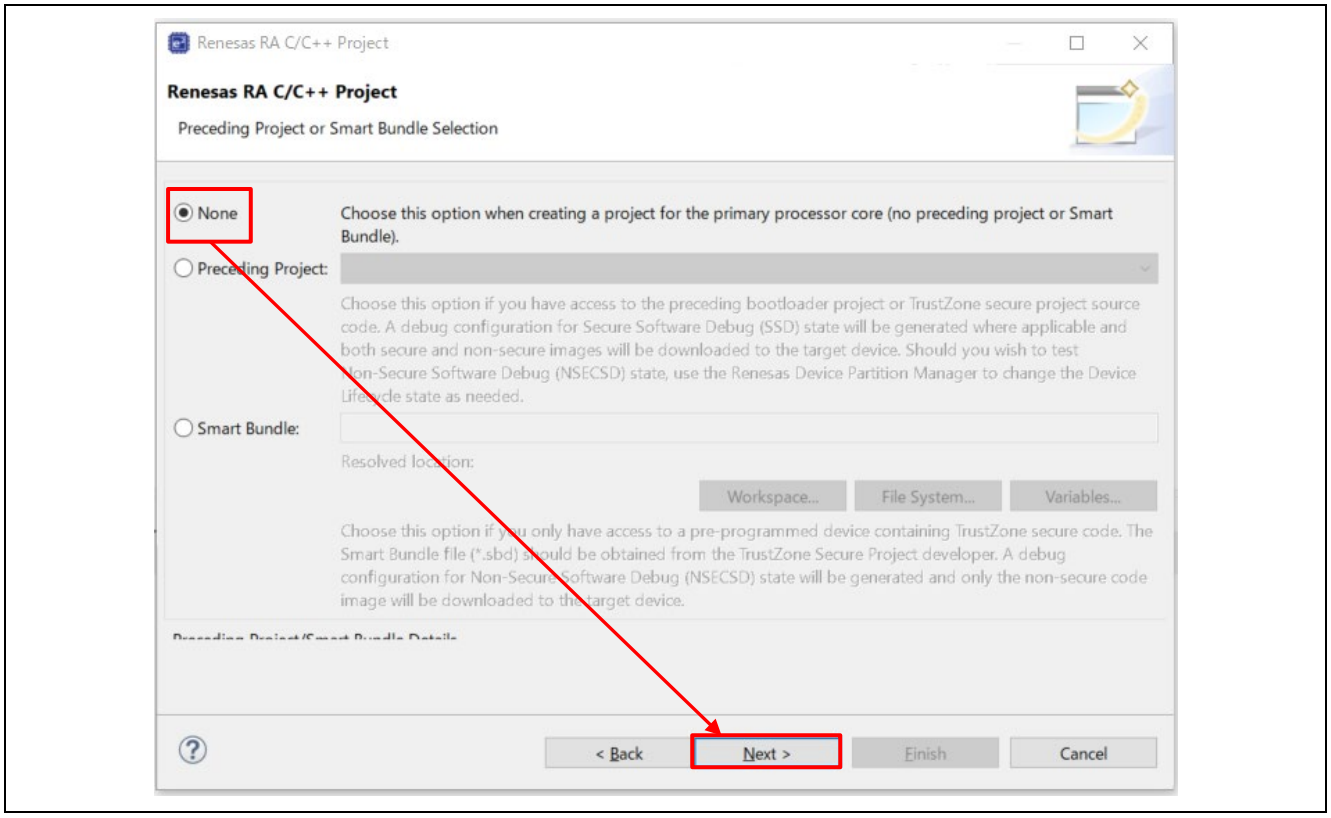
5. FSP Version とボードを選択します。



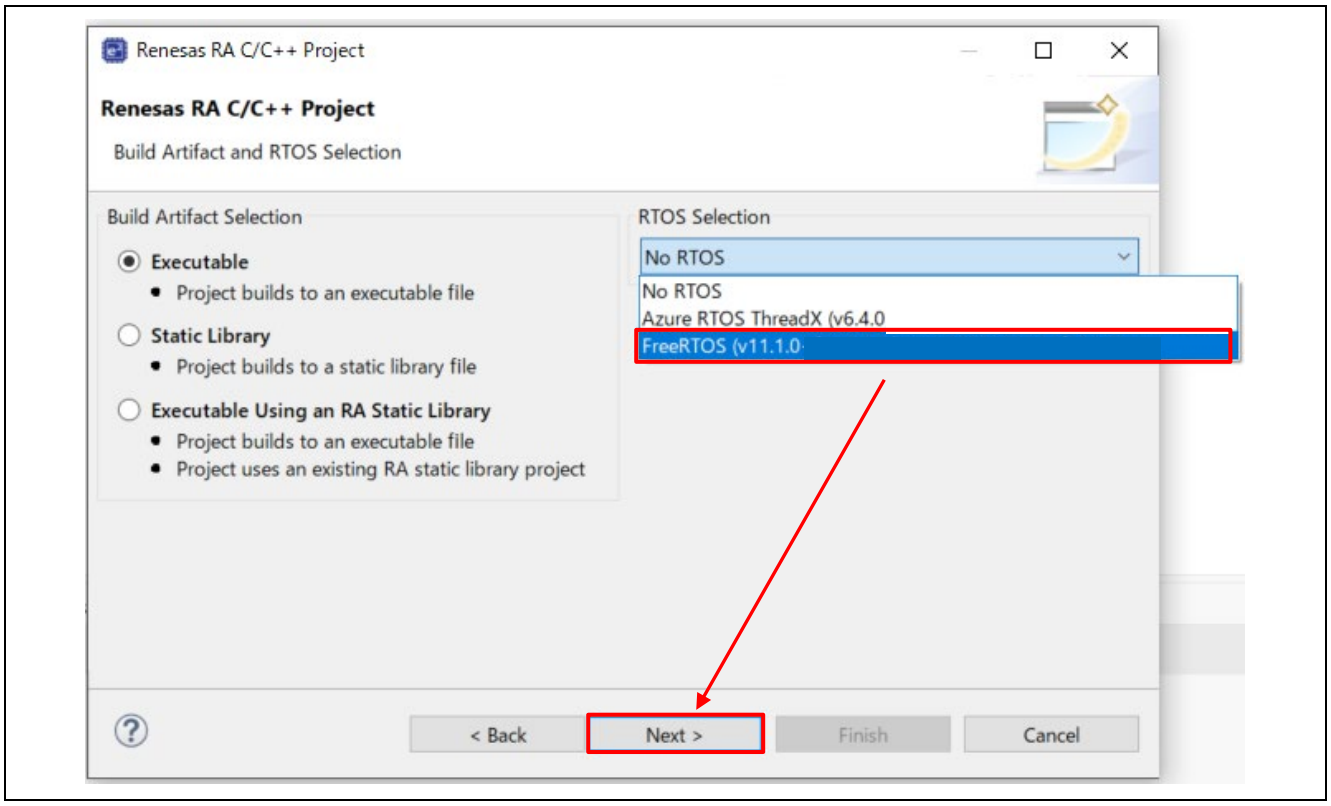
6. Project Type「Flat (Non-TrustZone) Project」を選択します。



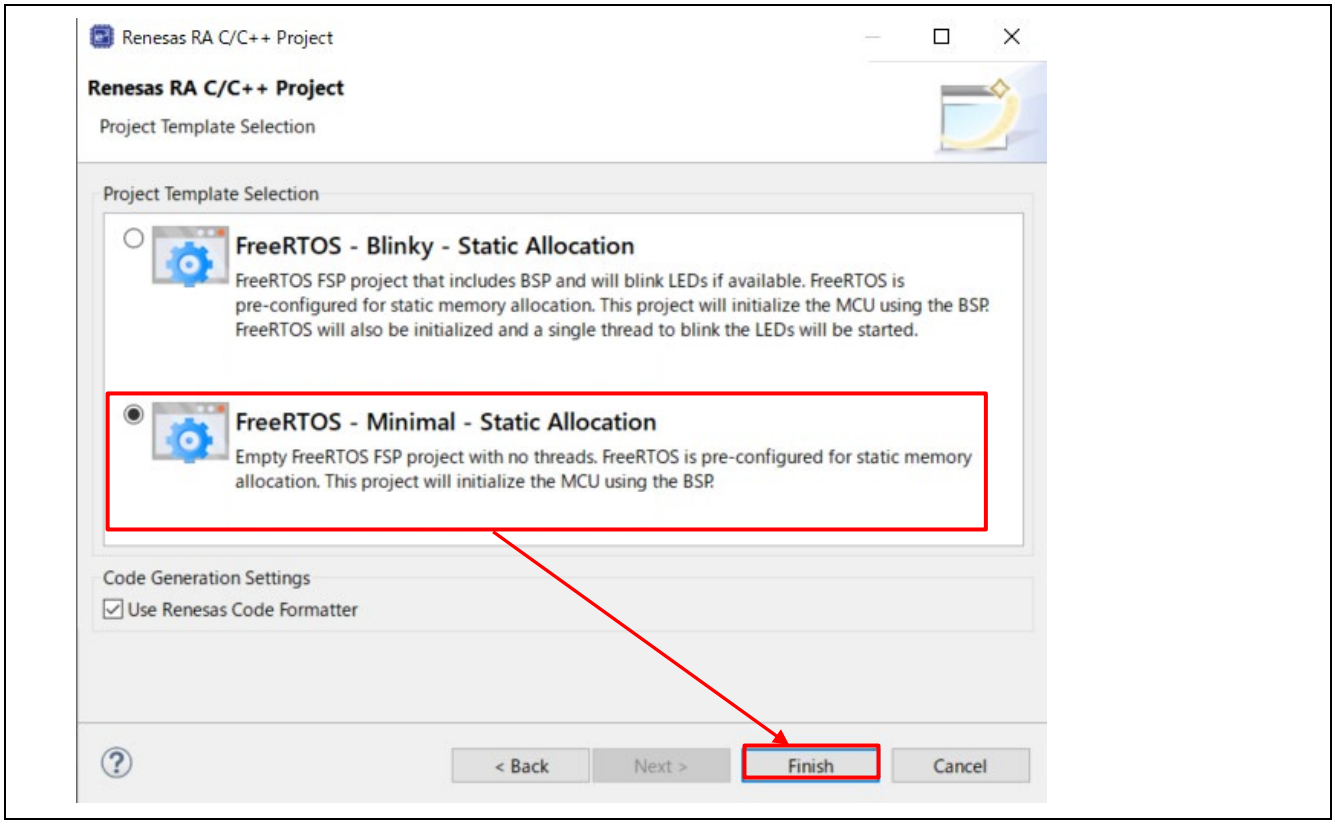
7. 「None」を選択します。



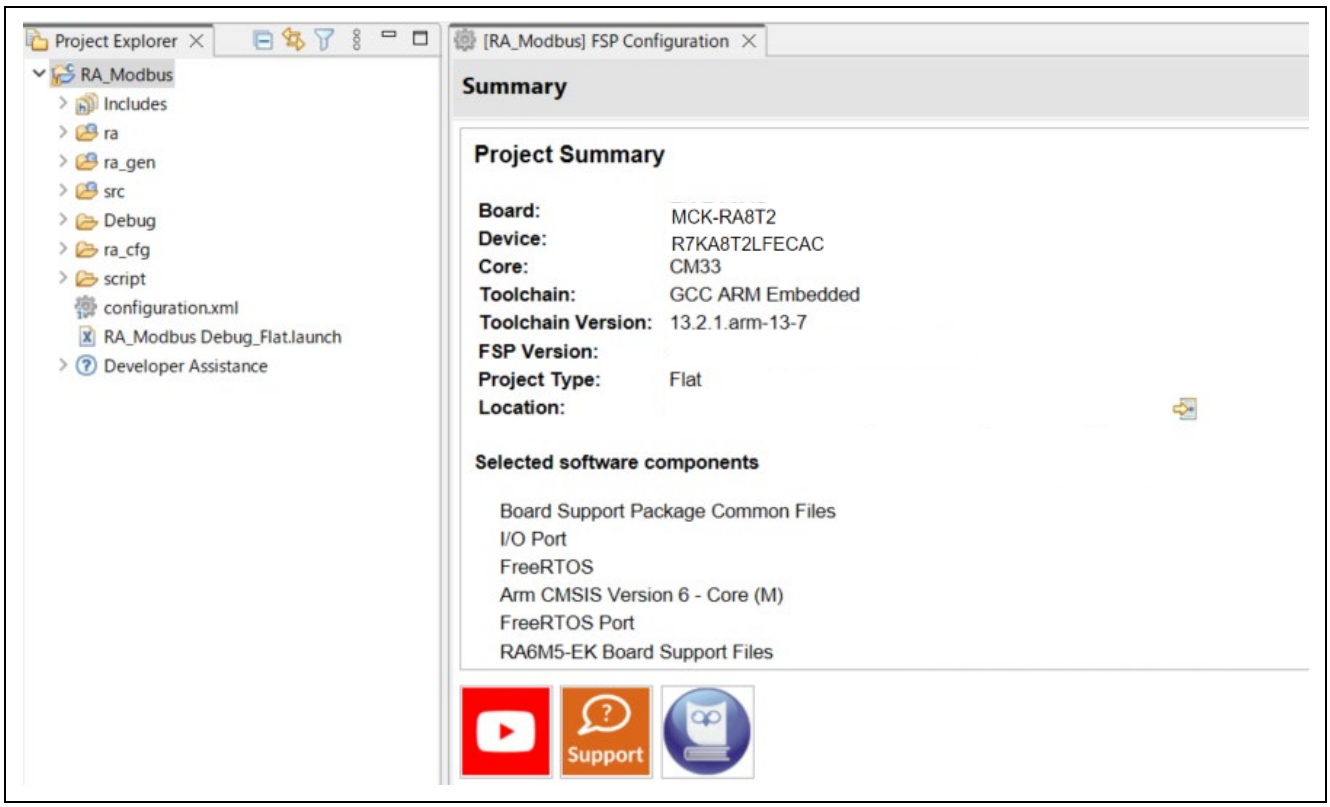
8. 「FreeRTOS (xxx)」を選択します。



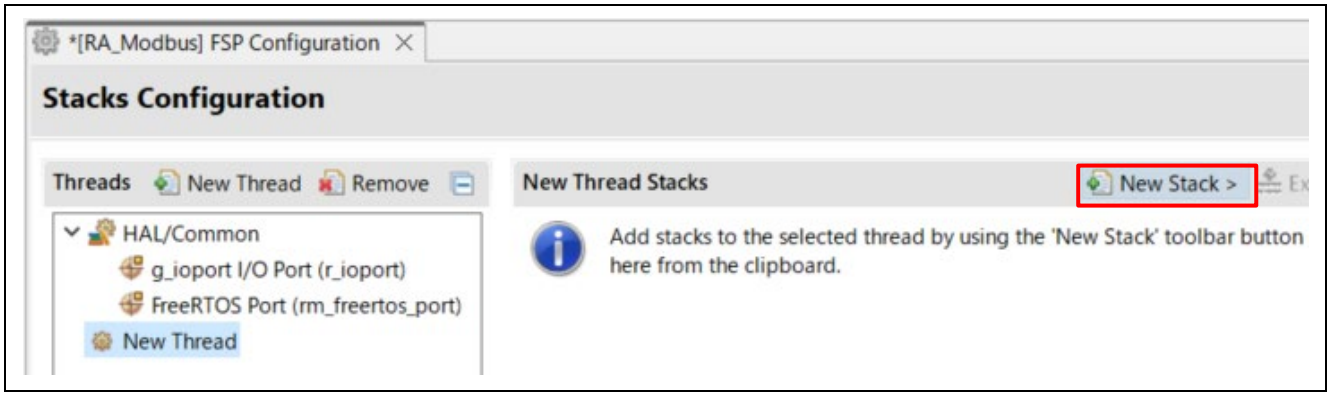
9. Project Template Selection の「FreeRTOS - Minimal - Static Allocation」を選択します。



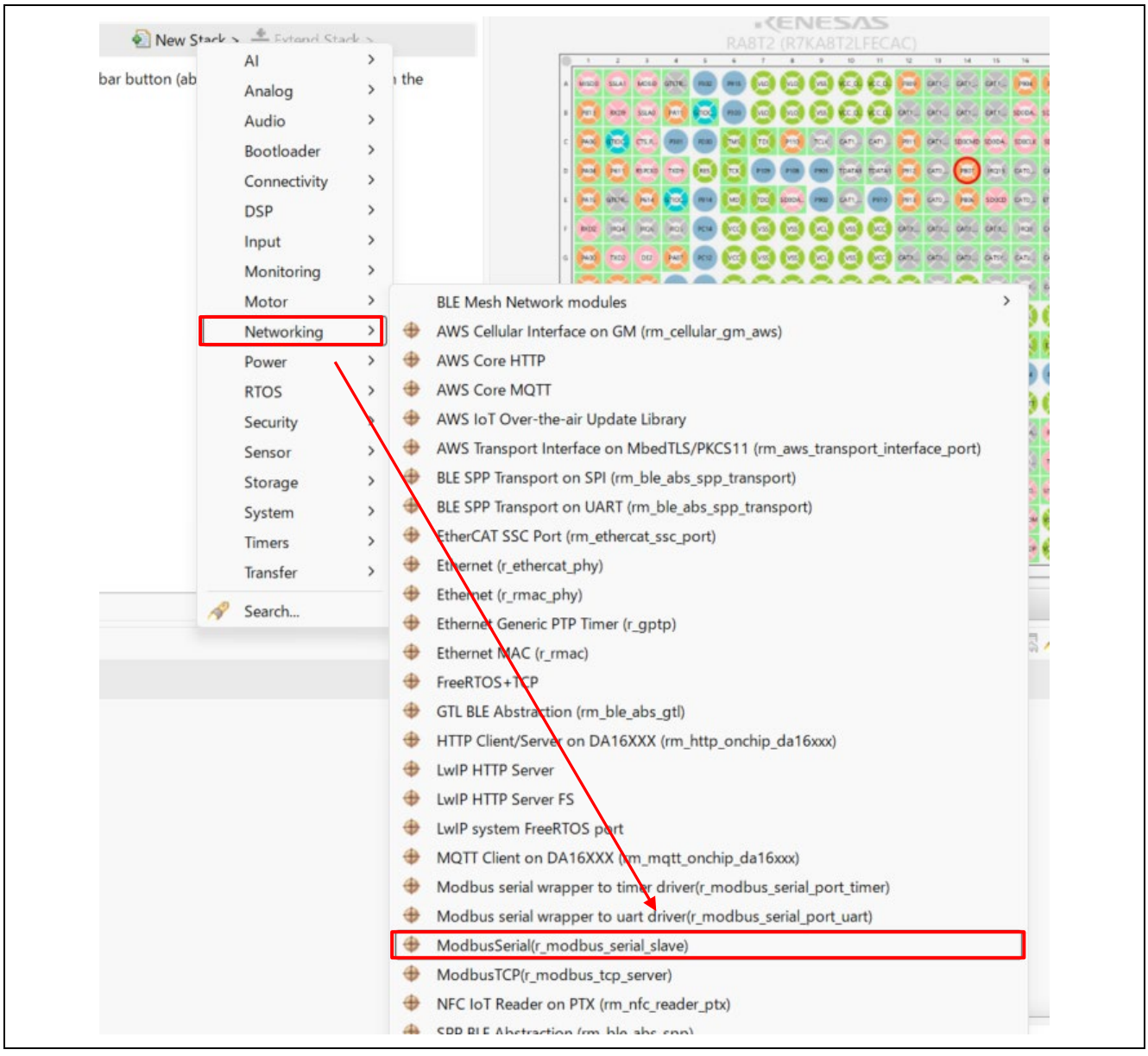
しばらくすると、作成したプロジェクトが「Project Explorer」に追加され、「FSP Configuration」タブが表示されます。



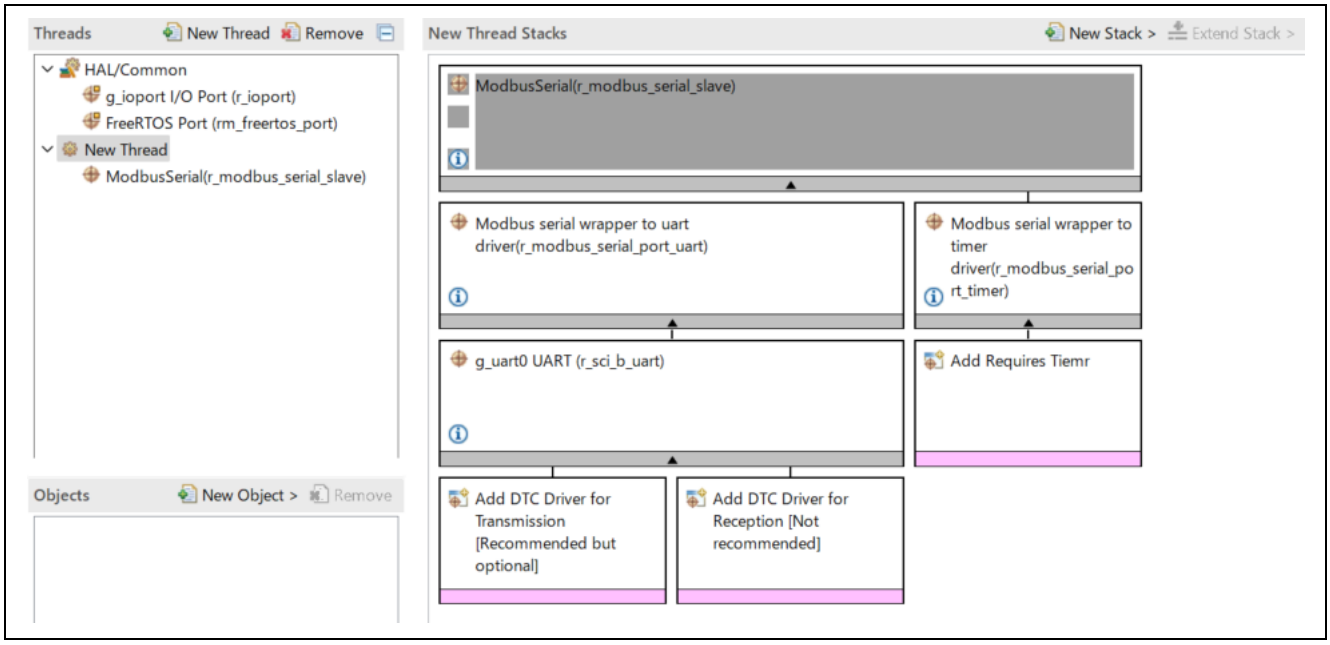
10. New Stacks の作成
「New Stacks」を選択します。



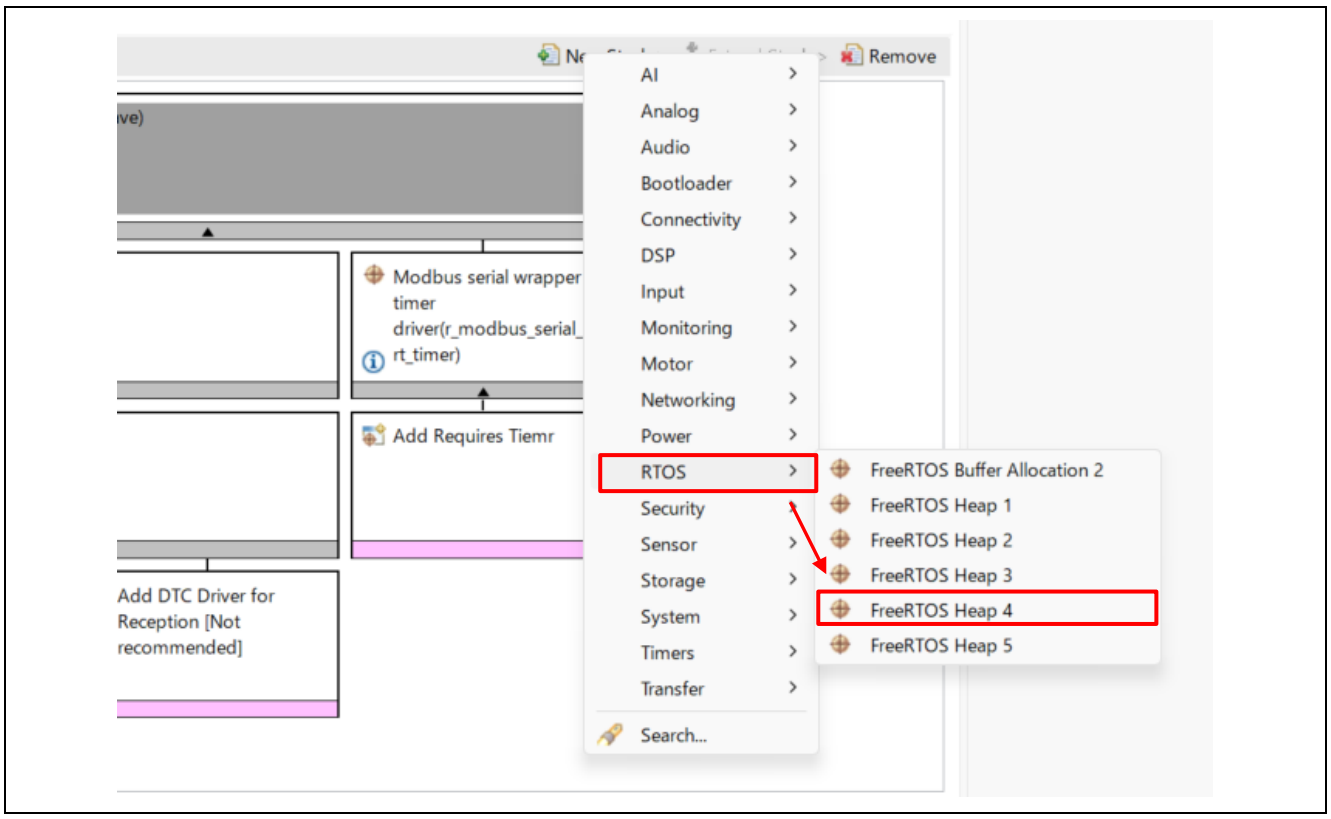
「Networking」→「ModbusSerial(r_modbus_serial_slave)」を選択します。



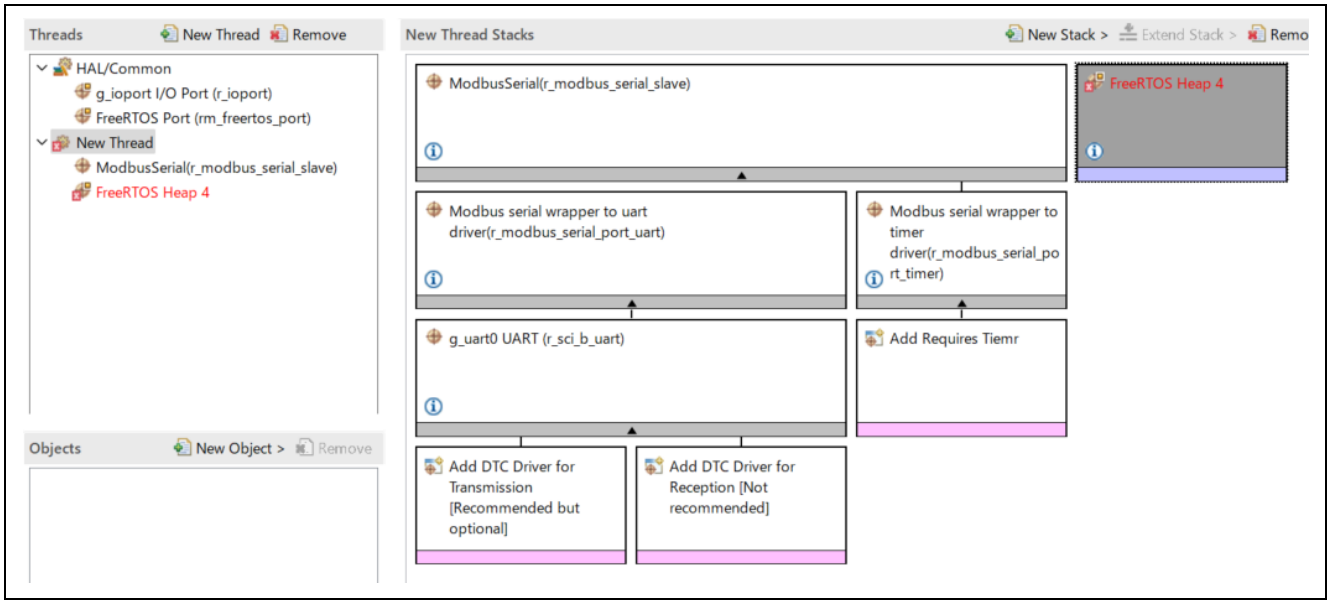
以下のようにスタックが構成されます。



11. Heap の追加
「New Stack」→「RTOS」→「FreeRTOS Heap 4」を選択します。

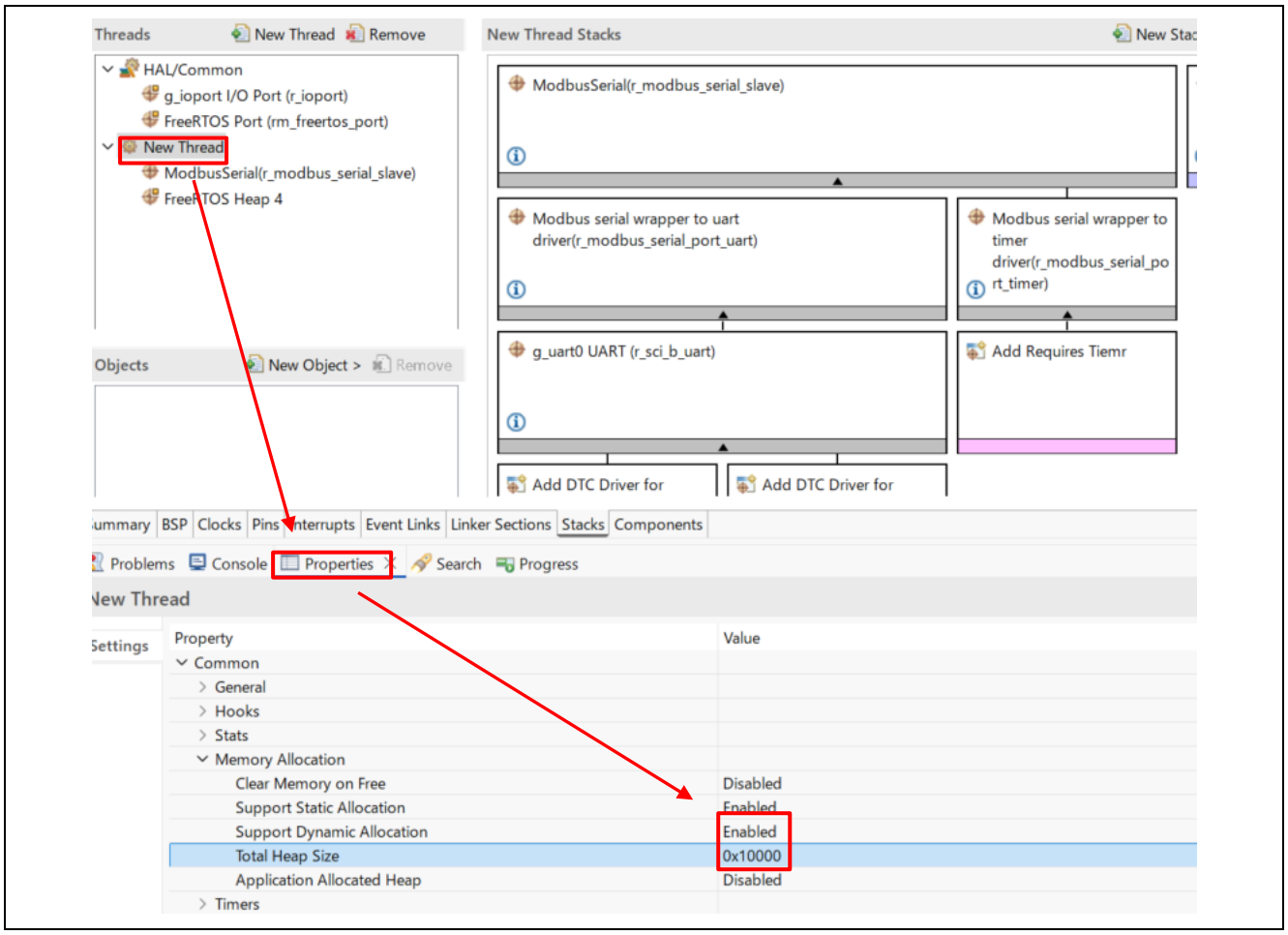


以下のようにスタックが追加されます。



12. Support Dynamic Allocation と Total Heap Size の設定
「New Thread」の「Properties」を展開し、「Memory Allocation」の「Support Dynamic Allocation」と「Total Heap Size」を以下の値に変更します。

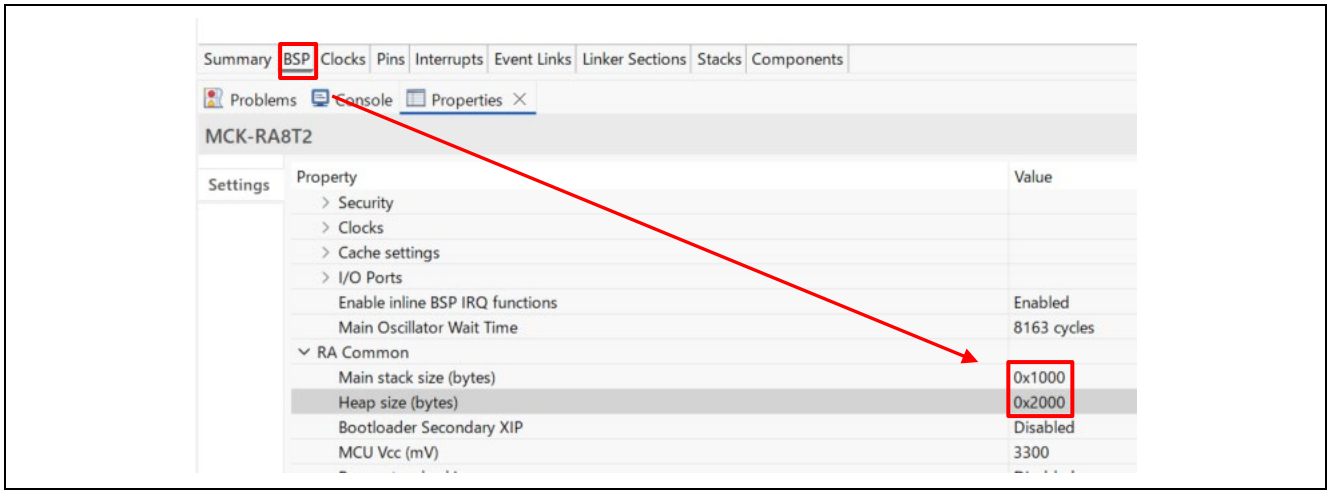
Support Dynamic Allocation：**Enabled**
Total Heap Size：**0x10000**



「FreeRTOS Heap4」のスタック構成のエラーが解消されます。

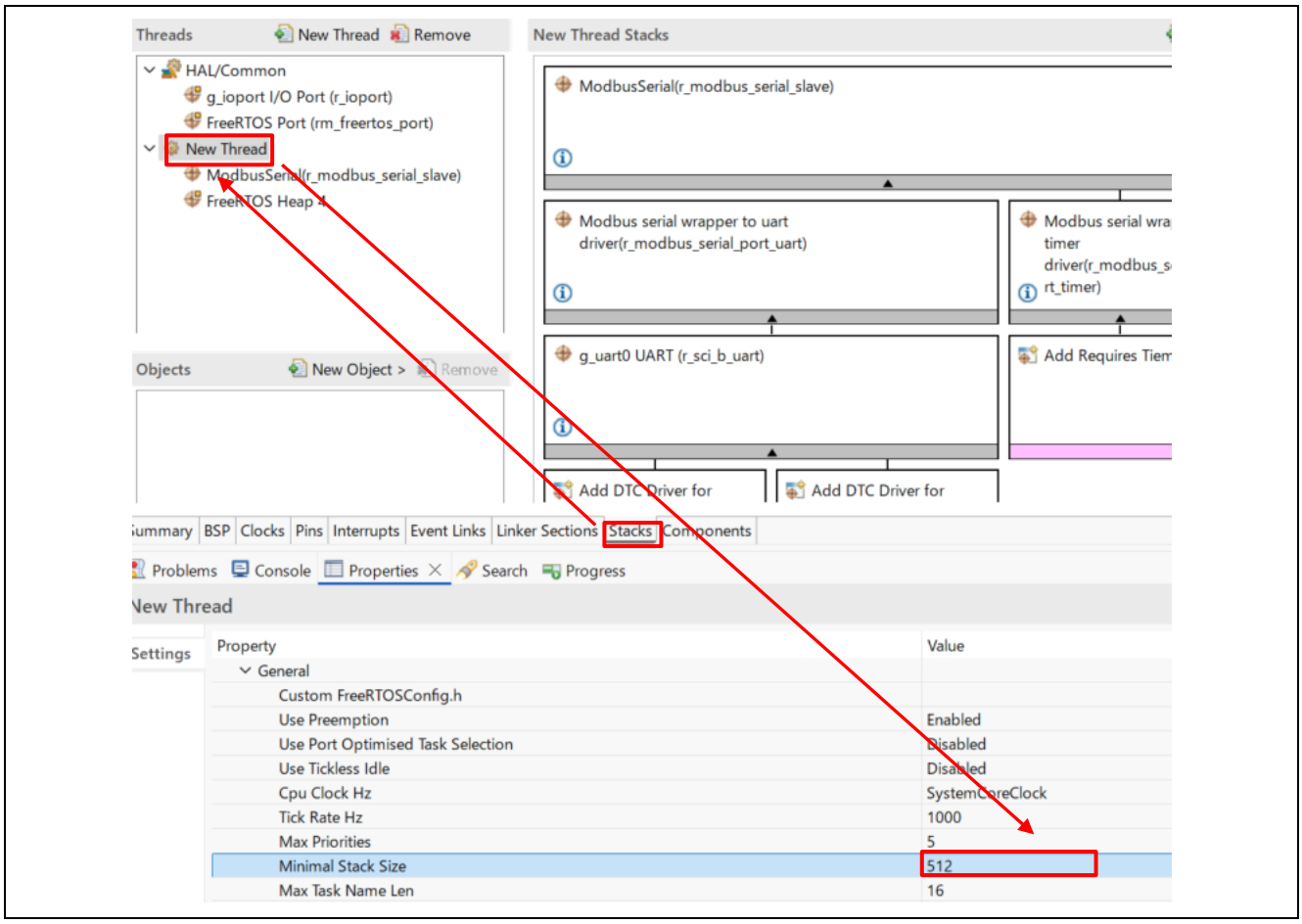
13. Stack size の設定
「BSP」→「RA Common」の「Main stack size」と,「Heap size」を以下の値に変更します。

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



「Stacks」→「New Thread」→「General」の「Minimal Stack size」を以下の値に変更します。

Minimal Stack size：**512**



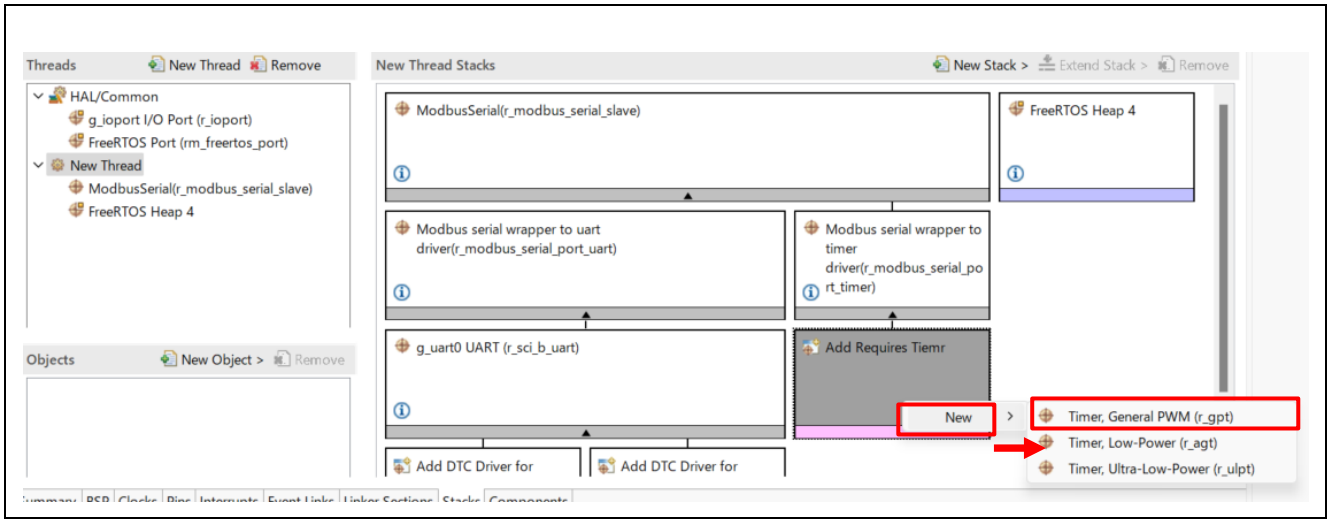
14. 各評価ボード用の手順を実行します。
使用する評価ボードに合わせて以下を参照し、手順を実行してください。

- MCK-RA8T2, EK-RA8M2：**5.2.2 MCK-RA8T2 / EK-RA8M2 用作成手順**
- EK-RA8T2：**5.2.3 EK-RA8T2 用作成手順**

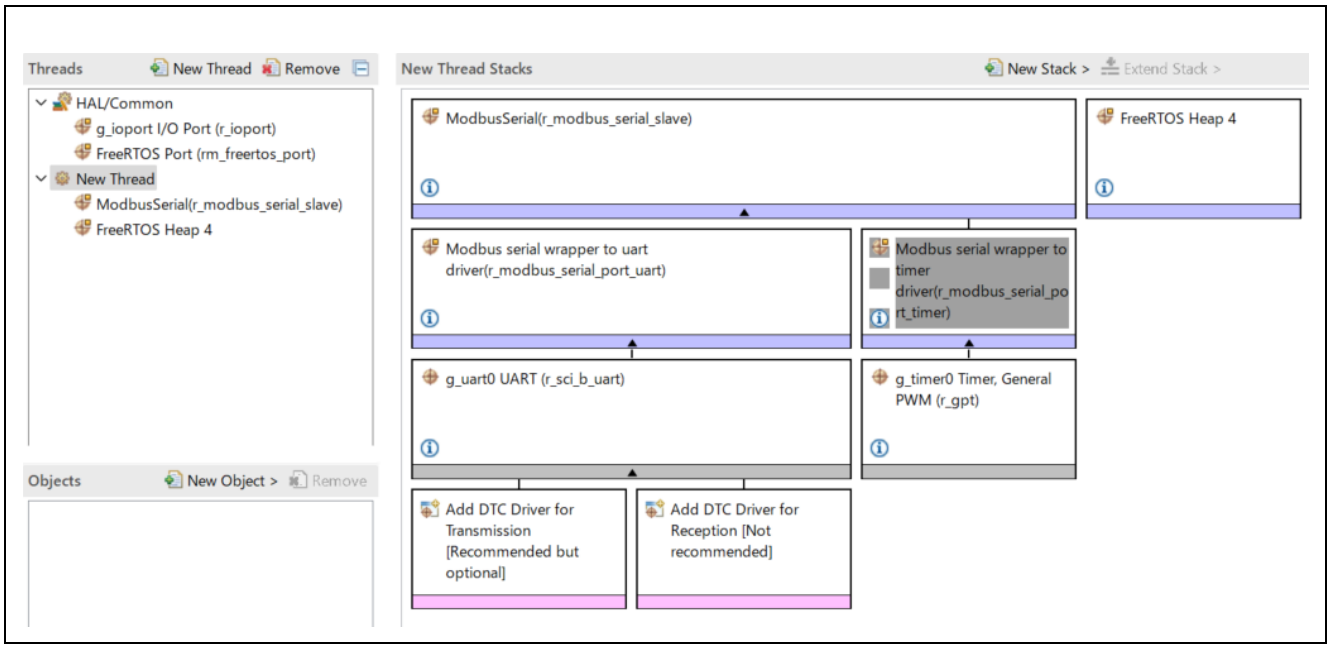
5.2.2　MCK-RA8T2 / EK-RA8M2 用作成手順

MCK-RA8T2 / EK-RA8M2 用の作成手順について説明します。

1.　Timer Driver の追加
「Add Requires Timer」に、「New」→「Timer, General PWM (r_gpt)」を追加します。



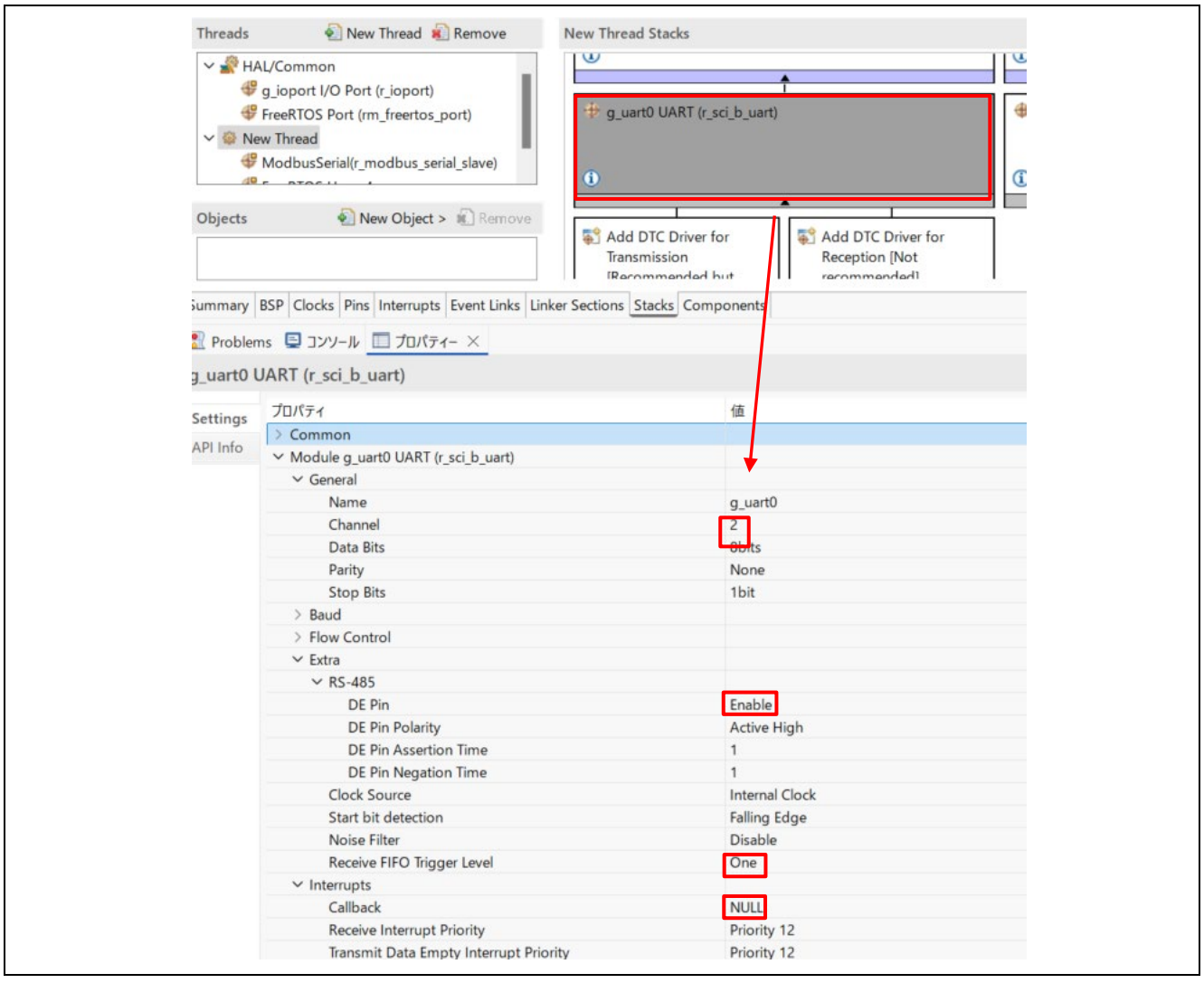
以下のようにスタックが構成されます。



2. UART の設定
「g_uart0 UART (r_sci_b_uart)」の「Properties」を展開し、「Channel」(評価ボードにより値が異なります)、「DE Pin」、「Receive FIFO Trigger Level」を以下の値に変更します。
シリアル伝送モード / ボーレート / パリティビット / ストップビットを変更する場合は「**5.3 シリアル伝送モード / ボーレート / パリティビット / ストップビットの設定**」を参照してください。

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

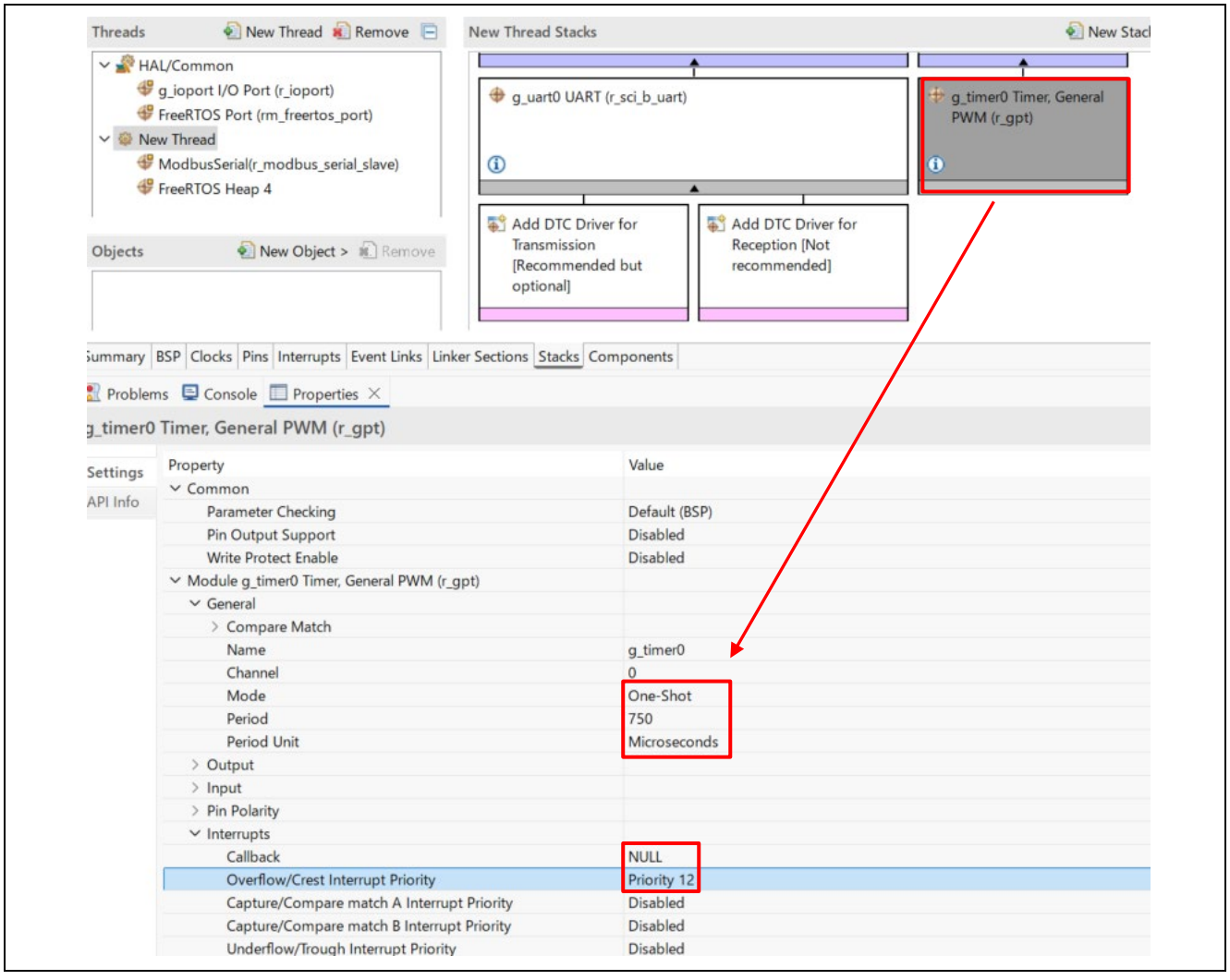
※UART 用コールバック関数は Modbus プロトコルスタックの初期化処理で設定されるため、「g_uart0 UART(r_sci_b_uart)」の Interrupts / Callback プロパティ設定はデフォルトの「NULL」を設定してください。



3. Timer の設定
「g_timer0 Timer, General PWM (r_gpt)」の「Properties」を展開し、「Mode」、「Period」、「Period Unit」、「Overflow/Crest Interrupt Priority」を以下の値に変更します。

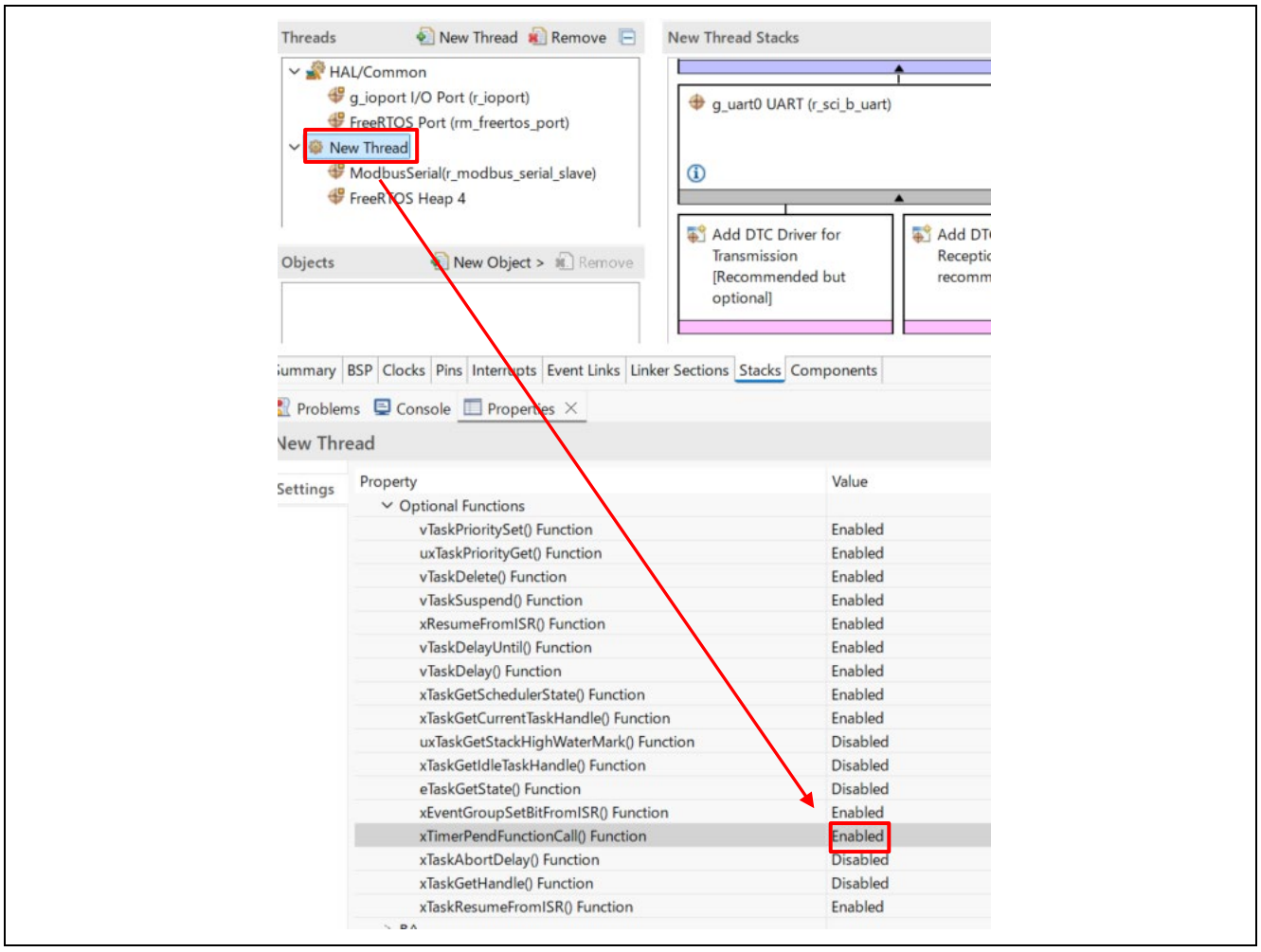
Mode：**One-Shot**
Period：**750**
Period Unit：**Microseconds**
Interrupts / Callback：**NULL** ※
Overflow/Crest Interrupt Priority：**Priority 12**

※Timer 用コールバック関数は Modbus プロトコルスタックの初期化処理で設定されるため、「g_timer0 Timer, General PWM (r_gpt)」の Interrupts / Callback プロパティ設定はデフォルトの「NULL」を設定してください。

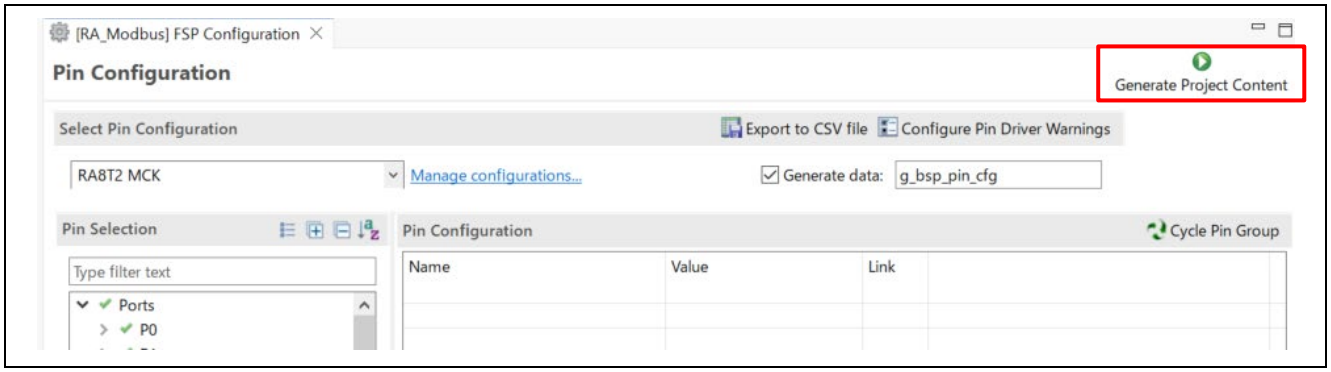


4. xTimerPendFunctionCall() Function の設定
「New Thread」の「Properties」を展開し、「xTimerPendFunctionCall() Function」を以下の値に変更します。

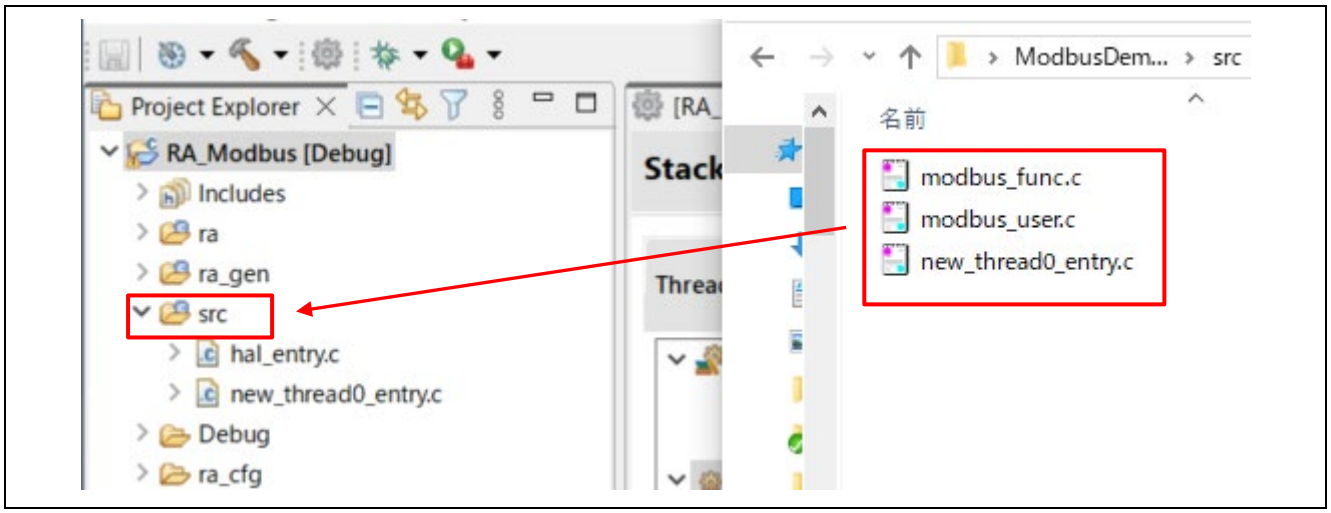
xTimerPendFunctionCall() Function：**Enabled**

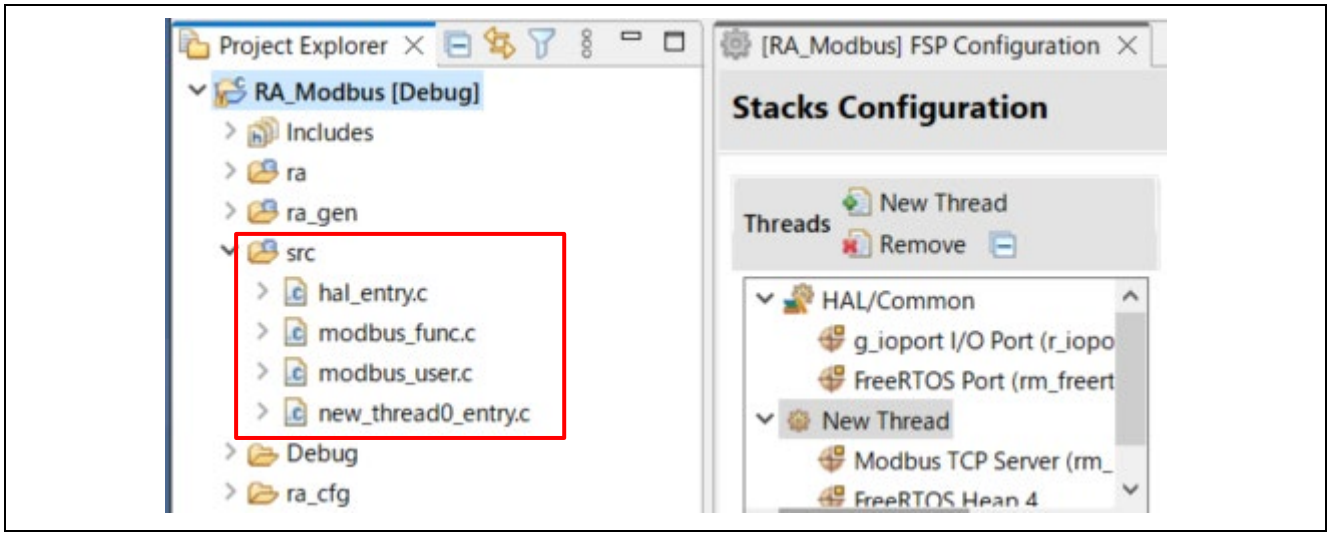


5. コード生成
   プロジェクトコンテンツの生成でコードを生成します。



6. Modbus サンプルアプリケーションの追加
   サンプルプログラムパッケージの src フォルダ以下の modbus_func.c、modbus_user.c、new_thread0_entry.c をプロジェクトの src フォルダに上書きコピーします。

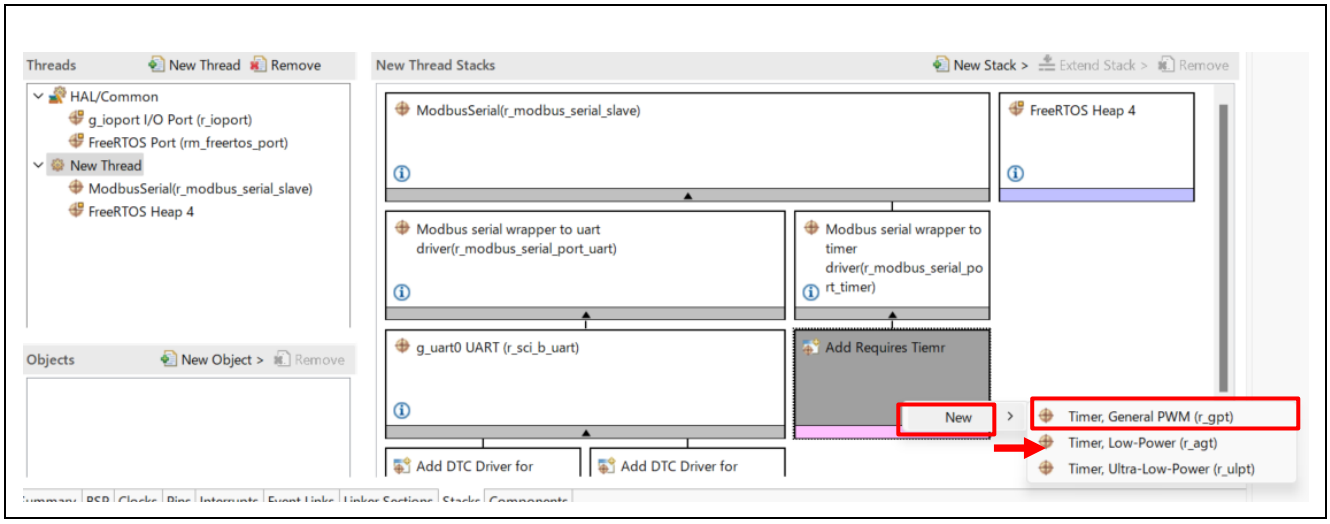




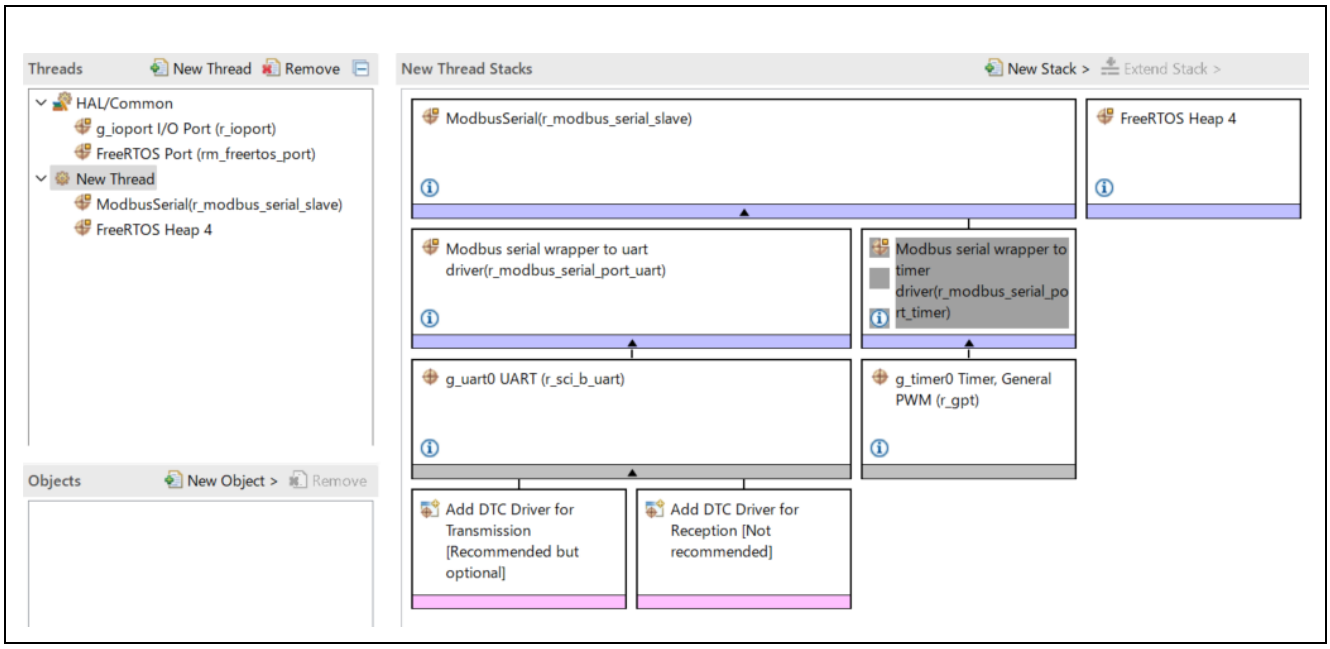
5.2.3　EK-RA8T2 用作成手順

EK-RA8T2 用の作成手順について説明します。

1. Timer Driver の追加
   「Add Requires Timer」に、「New」→「Timer, General PWM (r_gpt)」を追加します。



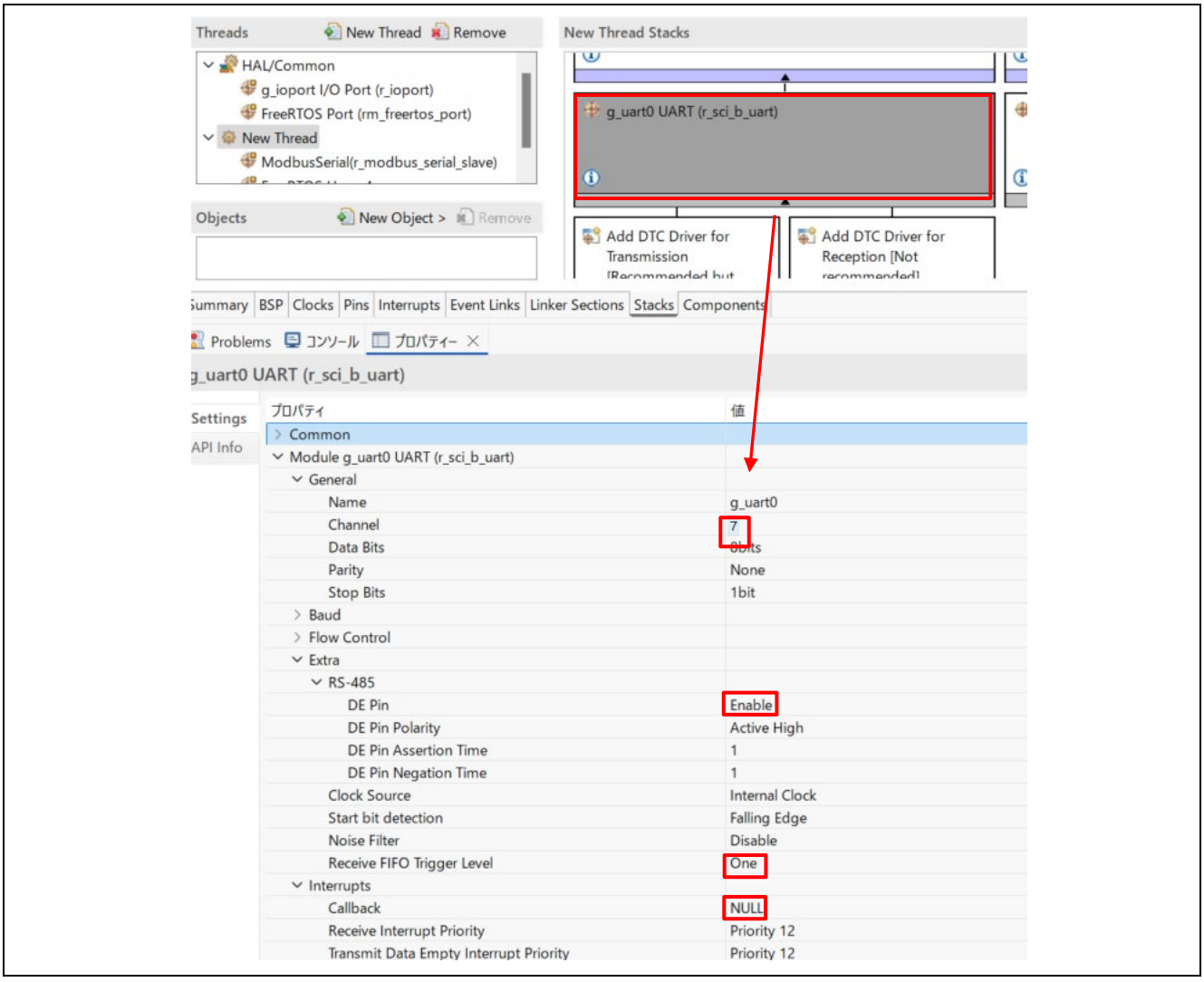
以下のようにスタックが構成されます。



2. UART の設定
「g_uart0 UART (r_sci_b_uart)」の「Properties」を展開し、「Channel」(評価ボードにより値が異なります)、「DE Pin」、「Receive FIFO Trigger Level」を以下の値に変更します。
シリアル伝送モード / ボーレート / パリティビット / ストップビットを変更する場合は「**5.3 シリアル伝送モード / ボーレート / パリティビット / ストップビットの設定**」を参照してください。

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

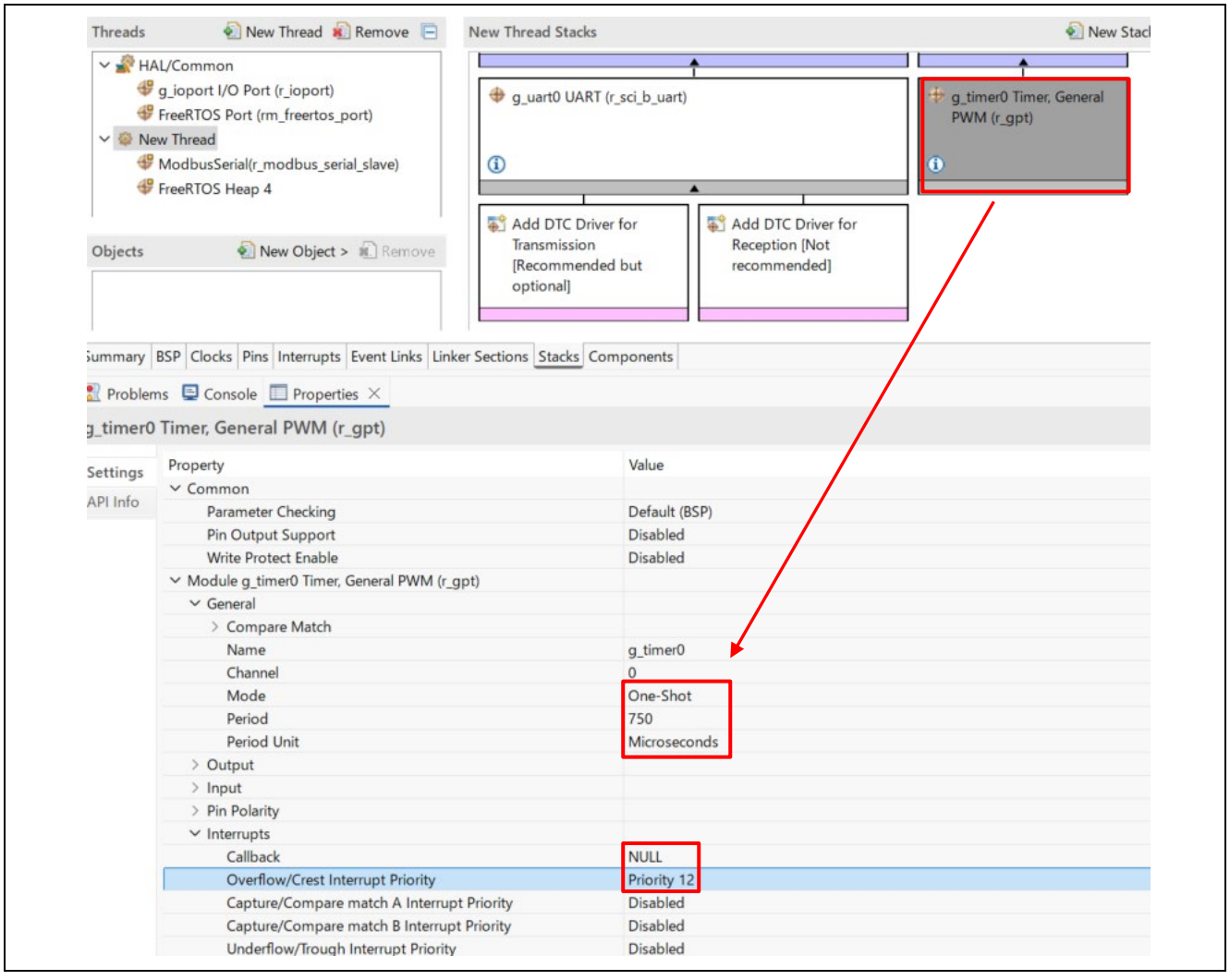
※UART 用コールバック関数は Modbus プロトコルスタックの初期化処理で設定されるため、「g_uart0 UART(r_sci_b_uart)」の Interrupts / Callback プロパティ設定はデフォルトの「NULL」を設定してください。



3. Timer の設定
「g_timer0 Timer, General PWM (r_gpt)」の「Properties」を展開し、「Mode」、「Period」、「Period Unit」、「Overflow/Crest Interrupt Priority」を以下の値に変更します。

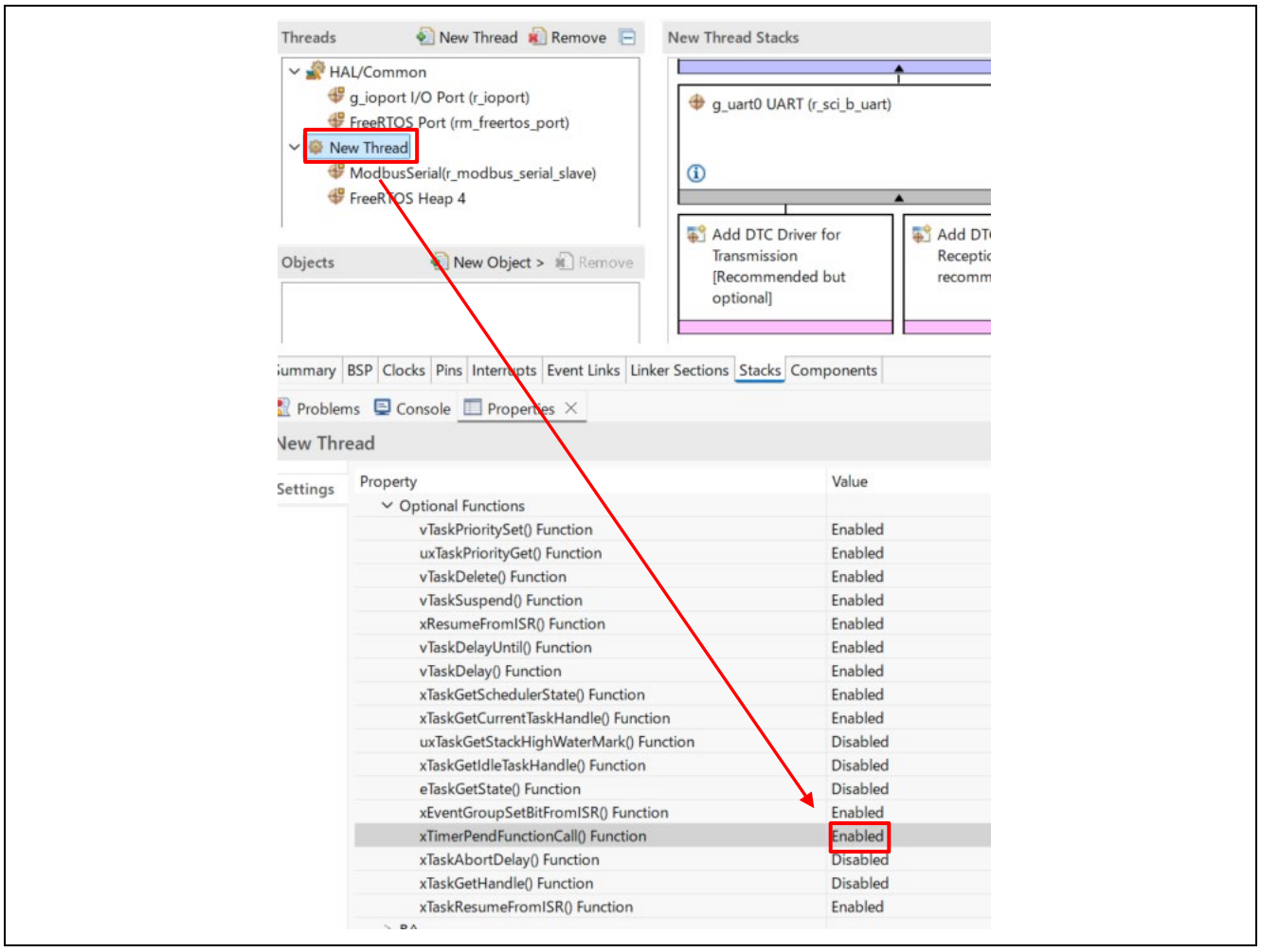
Mode：**One-Shot**
Period：**750**
Period Unit：**Microseconds**
Interrupts / Callback：**NULL** ※
Overflow/Crest Interrupt Priority：**Priority 12**

※Timer 用コールバック関数は Modbus プロトコルスタックの初期化処理で設定されるため、「g_timer0 Timer, General PWM (r_gpt)」の Interrupts / Callback プロパティ設定はデフォルトの「NULL」を設定してください。

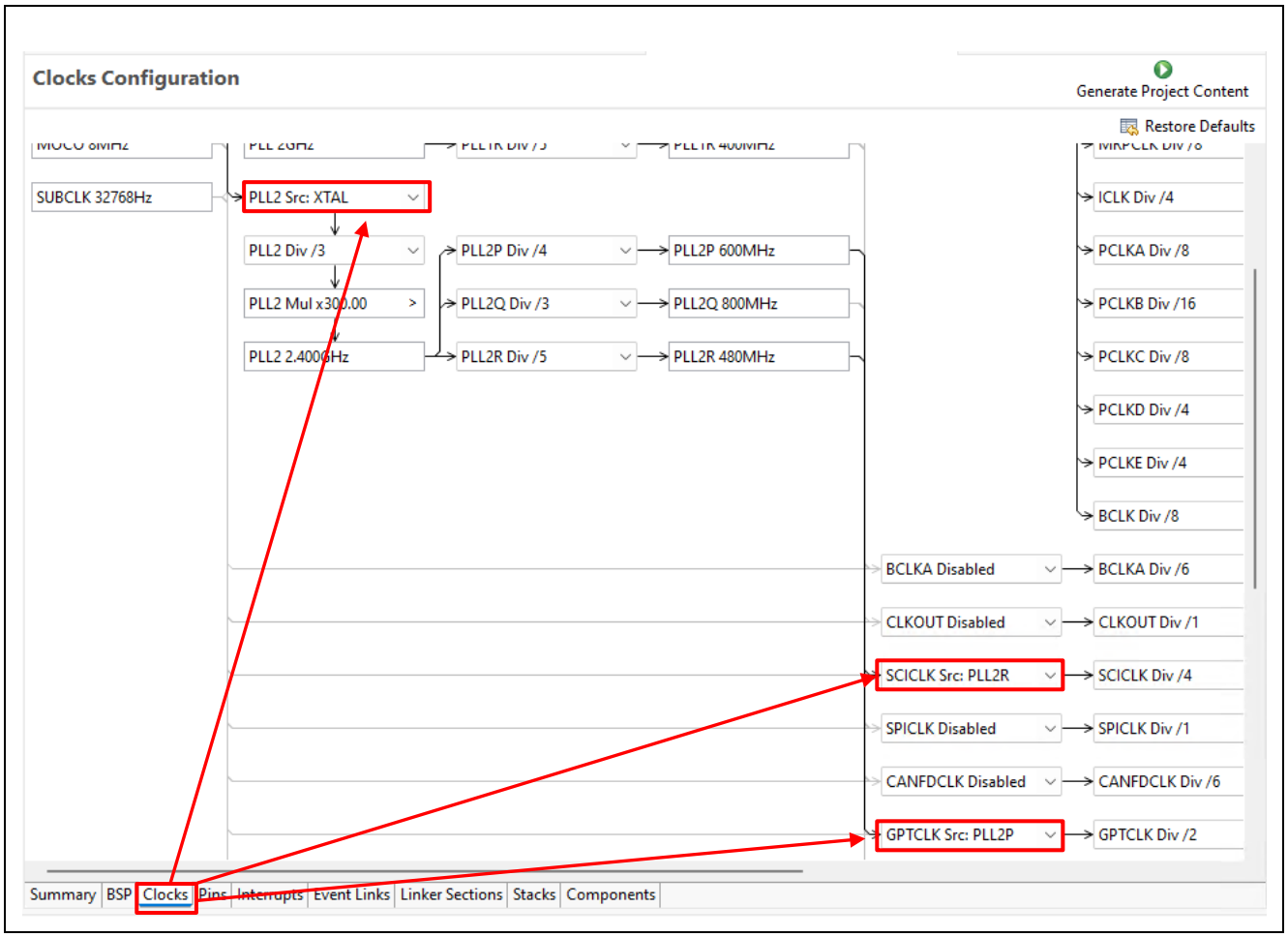


4. xTimerPendFunctionCall() Function の設定
「New Thread」の「Properties」を展開し、「xTimerPendFunctionCall() Function」を以下の値に変更します。

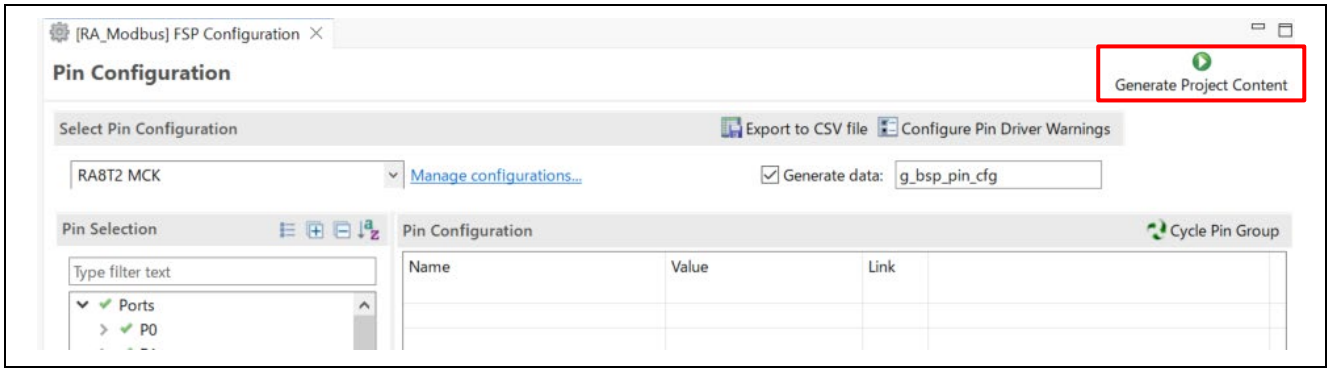
xTimerPendFunctionCall() Function：**Enabled**



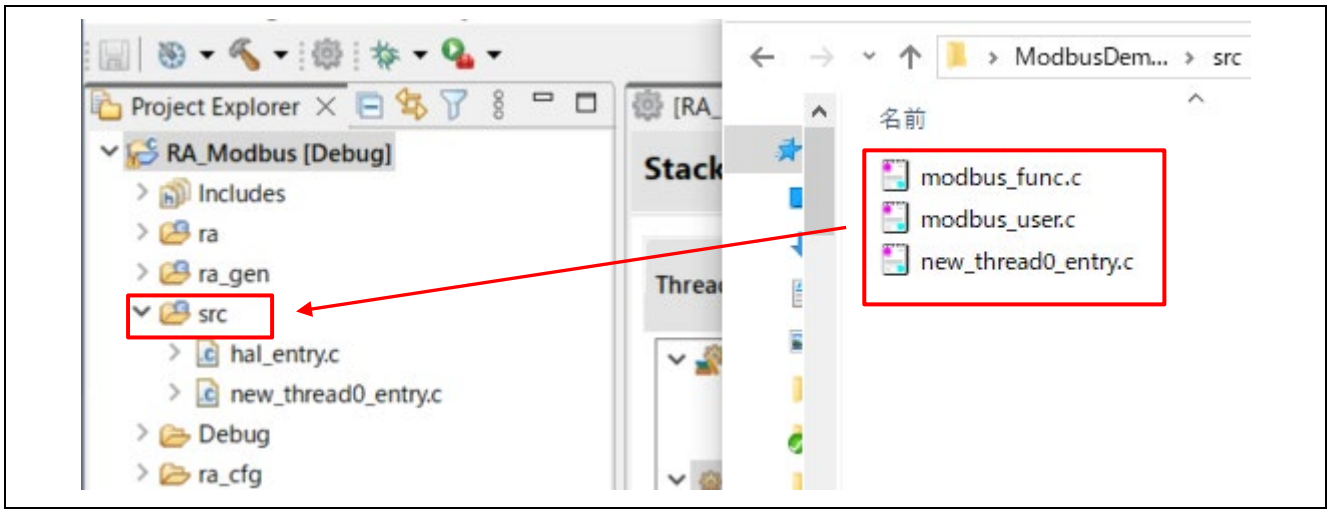
5. クロックの設定
「Clocks」の「PLL2 Src」、「SCICLK Src」、「GPTCLK Src」を以下の値に変更します。
PLL2 Src：**XTAL**
SCICLK Src：**PLL2R**
GPTCLK Src：**PLL2P**

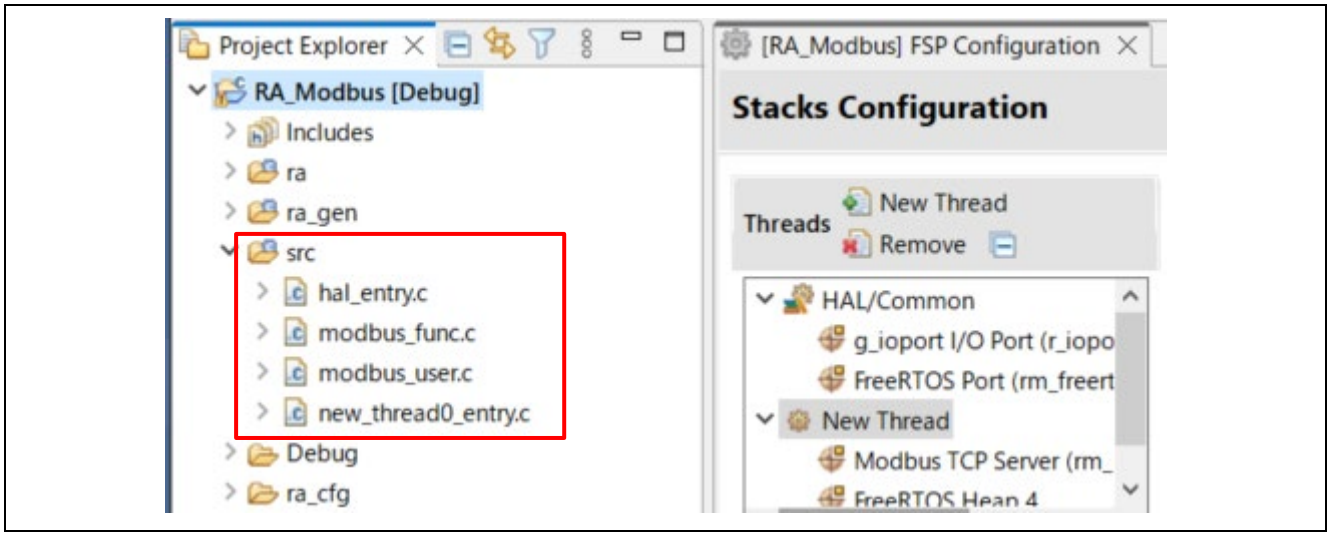


6. コード生成
   プロジェクトコンテンツの生成でコードを生成します。



7. Modbus サンプルアプリケーションの追加
   サンプルプログラムパッケージの src フォルダ以下の modbus_func.c、modbus_user.c、new_thread0_entry.c をプロジェクトの src フォルダに上書きコピーします。





5.3　シリアル伝送モード / ボーレート / パリティビット / ストップビットの設定

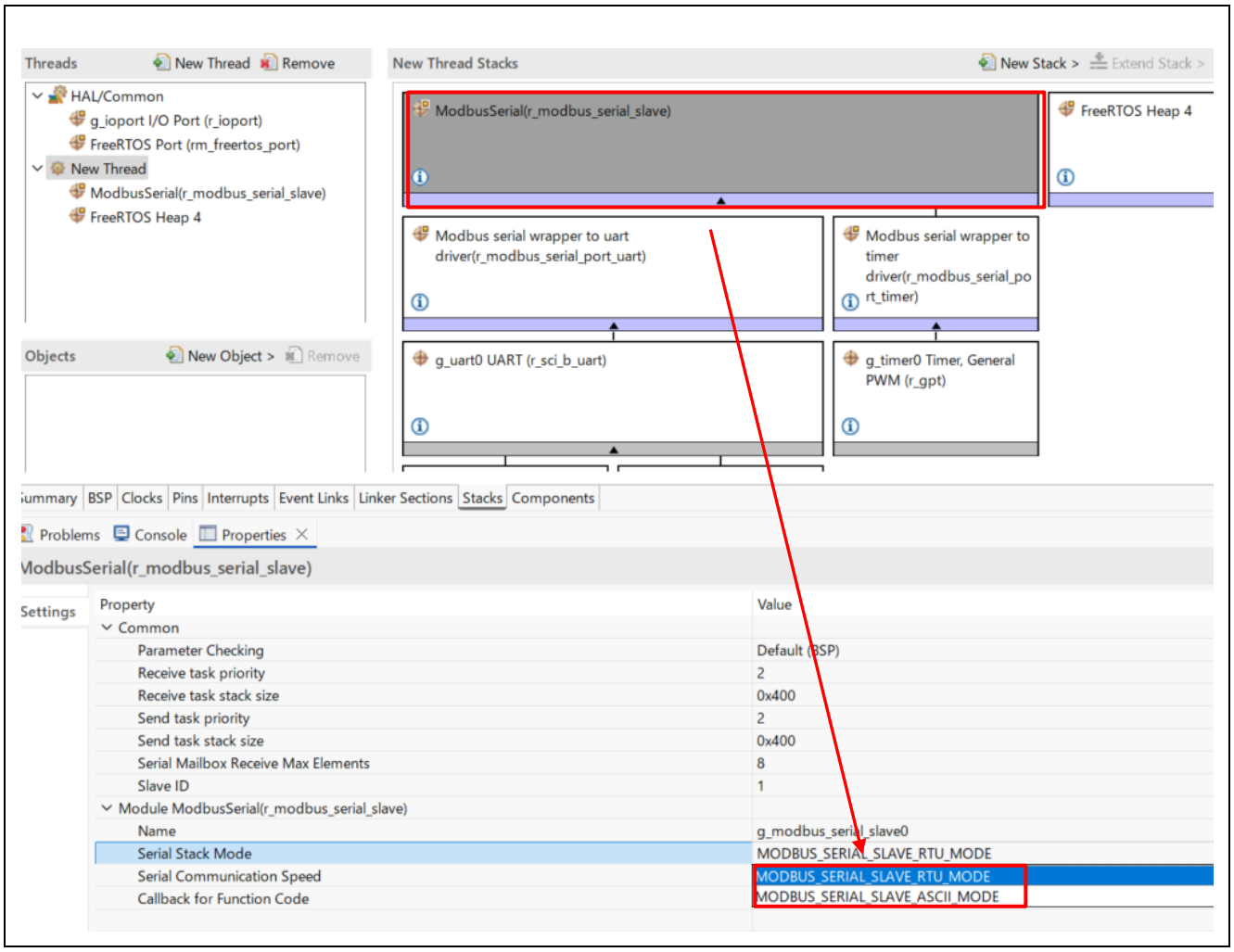
この章では、シリアル伝送モード / ボーレート / パリティビット / ストップビットの設定について説明します。

5.3.1　シリアル伝送モードの設定

シリアル送信モードの設定について説明します。

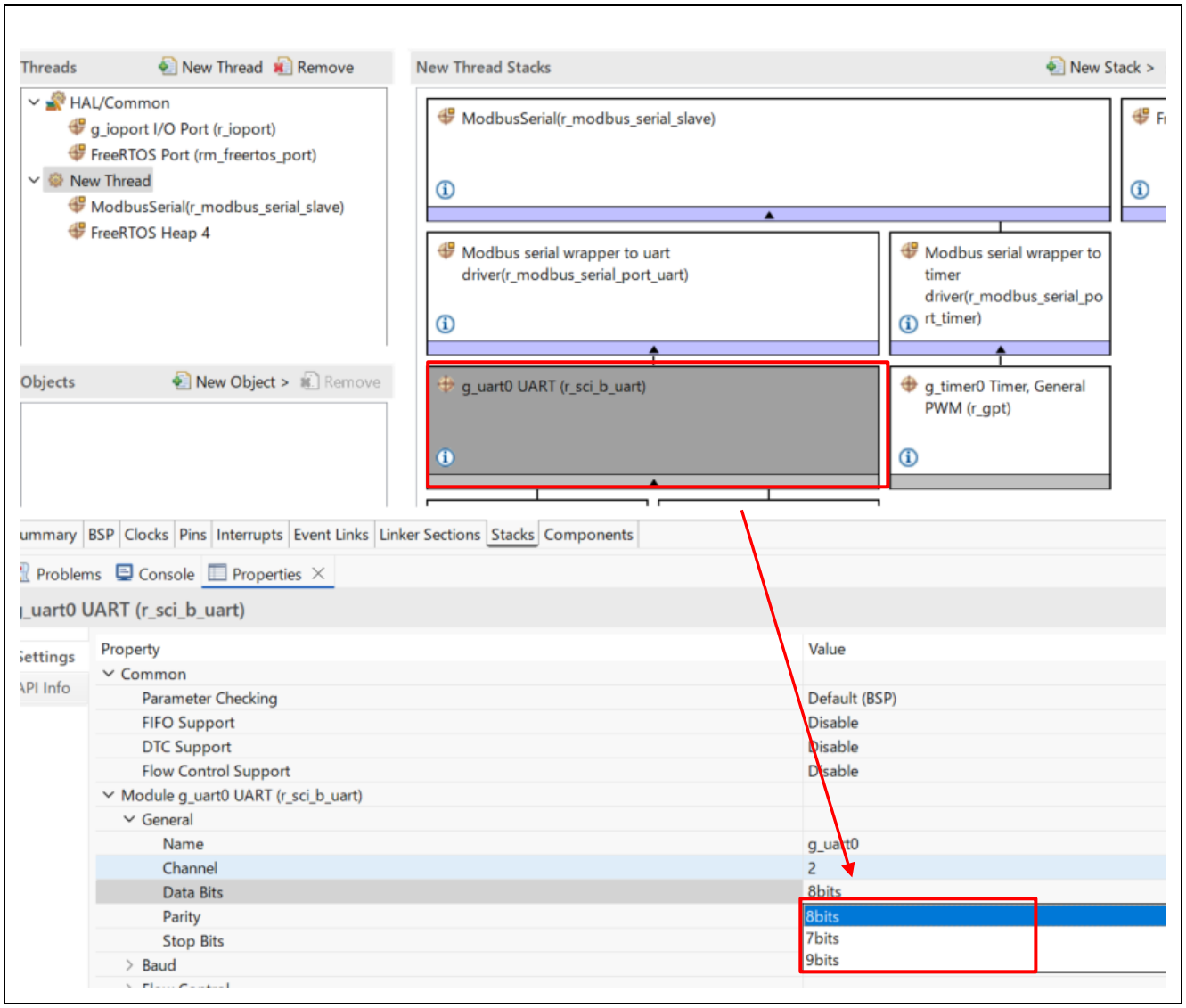
1. ModbusSerial の設定
   「ModbusSerial(r_modbus_serial_slave)」の「Properties」を展開し、「Serial Stack Mode」を以下の値に変更します。

- RTU モードで通信する場合
  Serial Stack Mode： **MODBUS_SERIAL_SLAVE_RTU_MODE**
- ASCII モードで通信する場合
  Serial Stack Mode： **MODBUS_SERIAL_SLAVE_ASCII_MODE**



2. UART の設定
「g_uart0 UART (r_sci_b_uart)」の「Properties」を展開し、「Data Bits」を以下の値に変更します。

- RTU モードで通信する場合
  Data Bits： **8 bits**
- ASCII モードで通信する場合
  Data Bits： **7 bits**

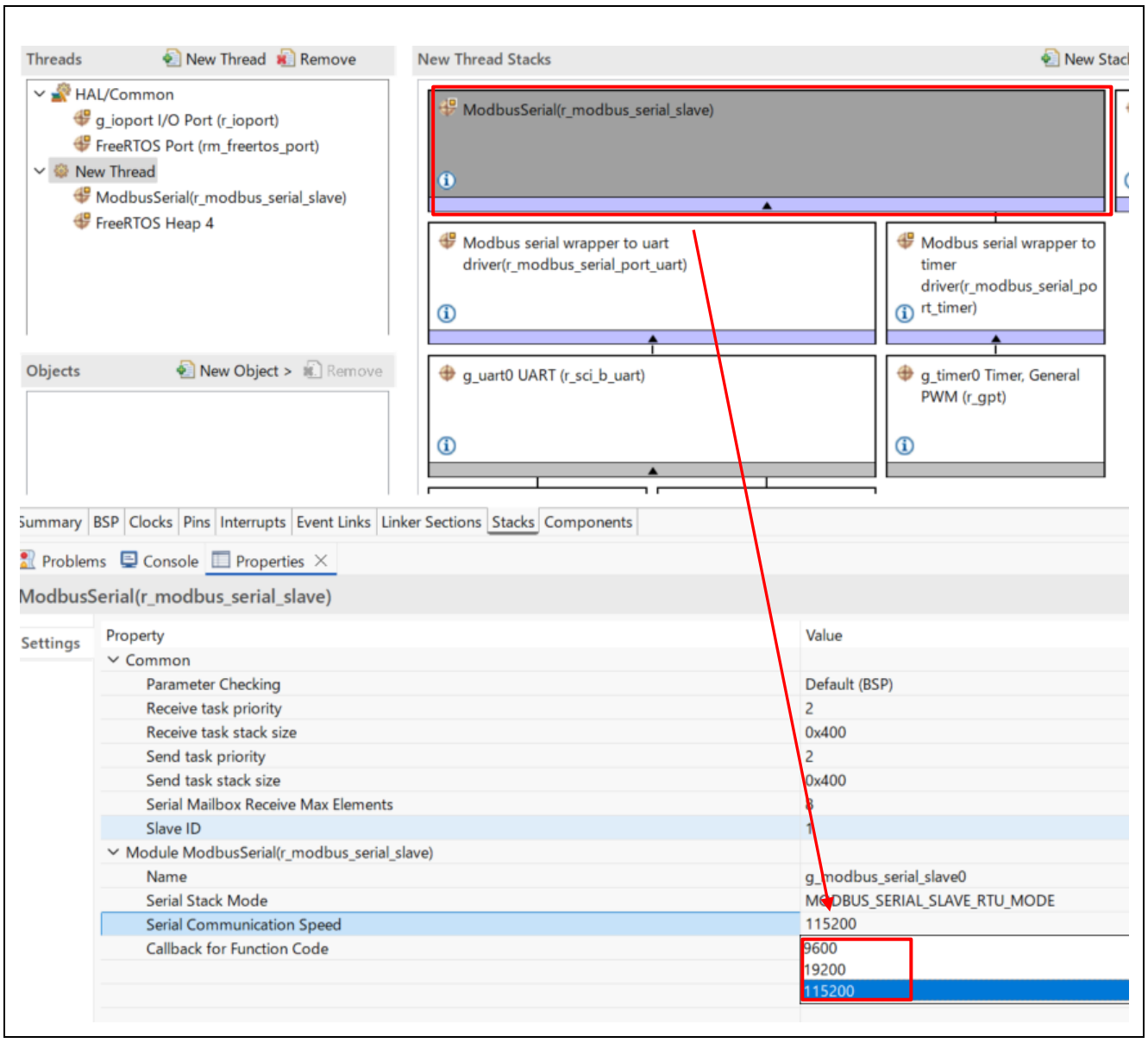


5.3.2　ボーレートの設定

ボーレートの設定について説明します。Modbus プロトコルスタックで設定できるボーレートは 9600bps / 19200bps / 115200bps です。

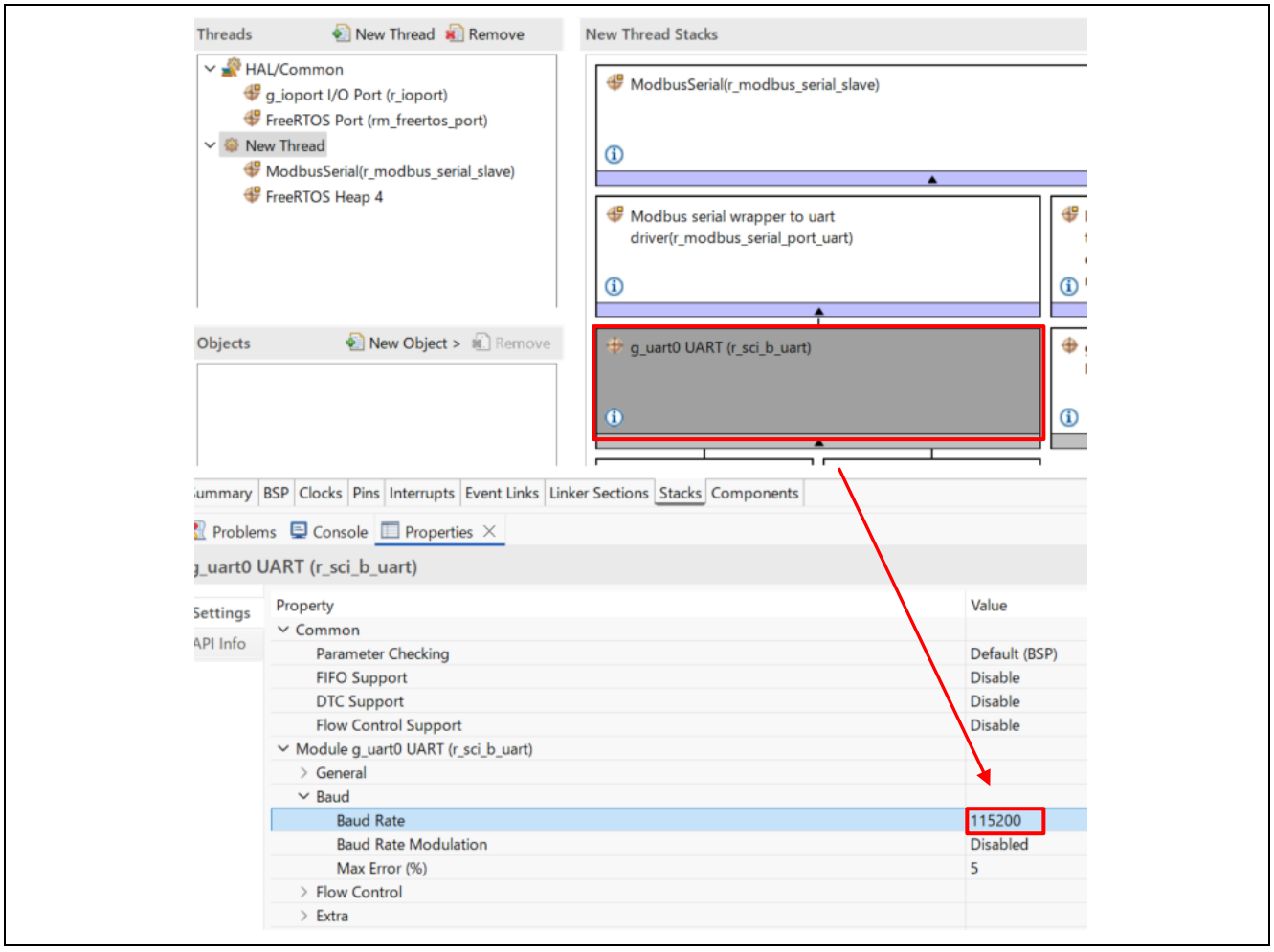
1. ModbusSerial の設定
   「ModbusSerial(r_modbus_serial_slave)」の「Properties」を展開し、「Serial Communication Speed」を以下の値に変更します。

- 9600 bps で通信する場合
  Serial Communication Speed： **9600**
- 19200 bps で通信する場合
  Serial Communication Speed：**19200**
- 115200 bps で通信する場合
  Serial Communication Speed： **115200**



2. UART の設定
「g_uart0 UART (r_sci_b_uart)」の「Properties」を展開し、「Baud Rate」を以下の値に変更します。

- 9600 bps で通信する場合
  Baud Rate： **9600**
- 19200 bps で通信する場合
  Baud Rate：**19200**
- 115200 bps で通信する場合
  Baud Rate： **115200**

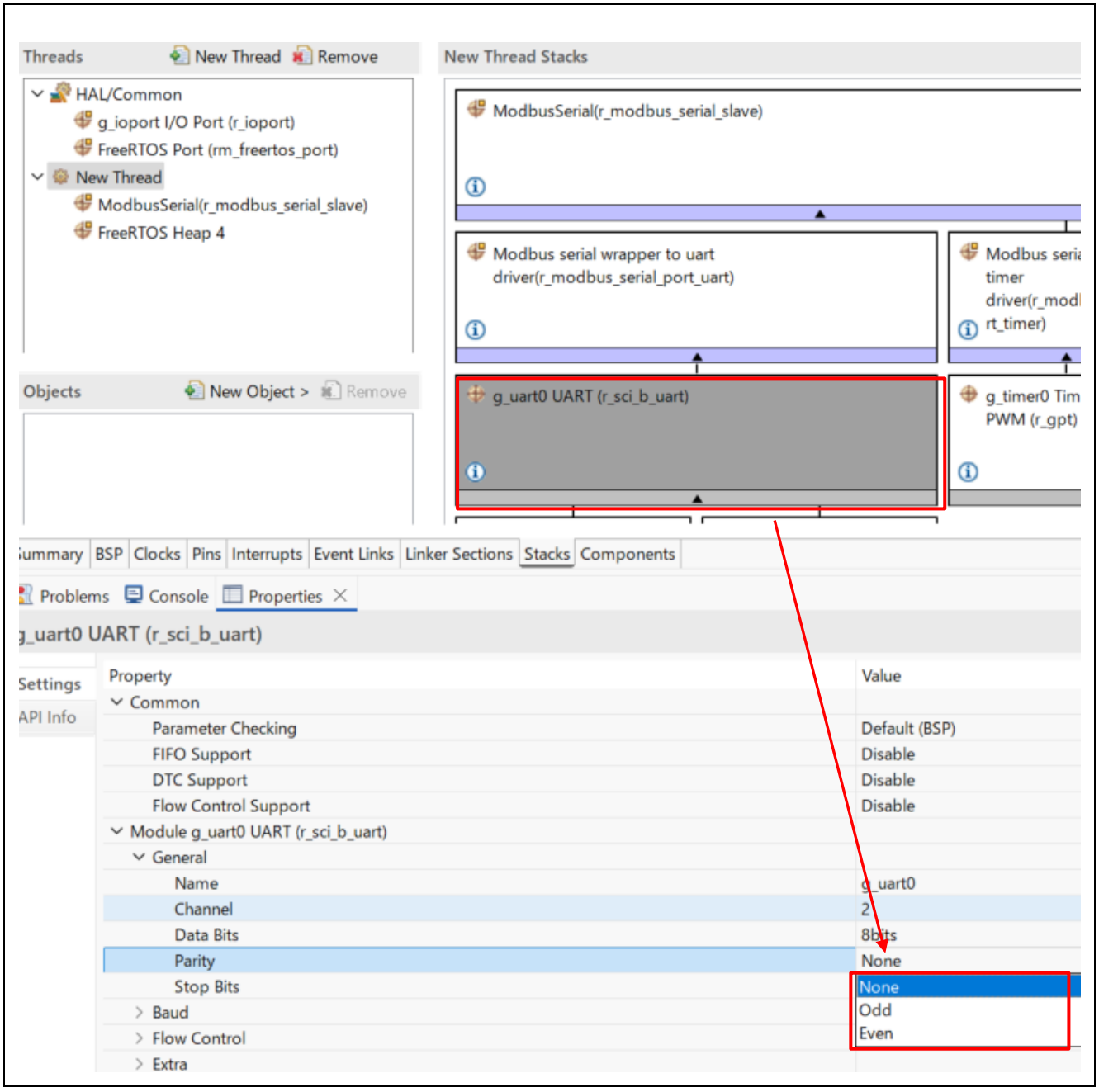


5.3.3　パリティビットの設定

パリティビットの設定について説明します。

1. UART の設定
   「g_uart0 UART (r_sci_b_uart)」の「Properties」を展開し、「Parity」を以下の値に変更します。

- パリティなしで通信する場合
  Parity： **None**
- 奇数パリティで通信する場合
  Parity： **Odd**
- 偶数パリティで通信する場合
  Parity： **Even**

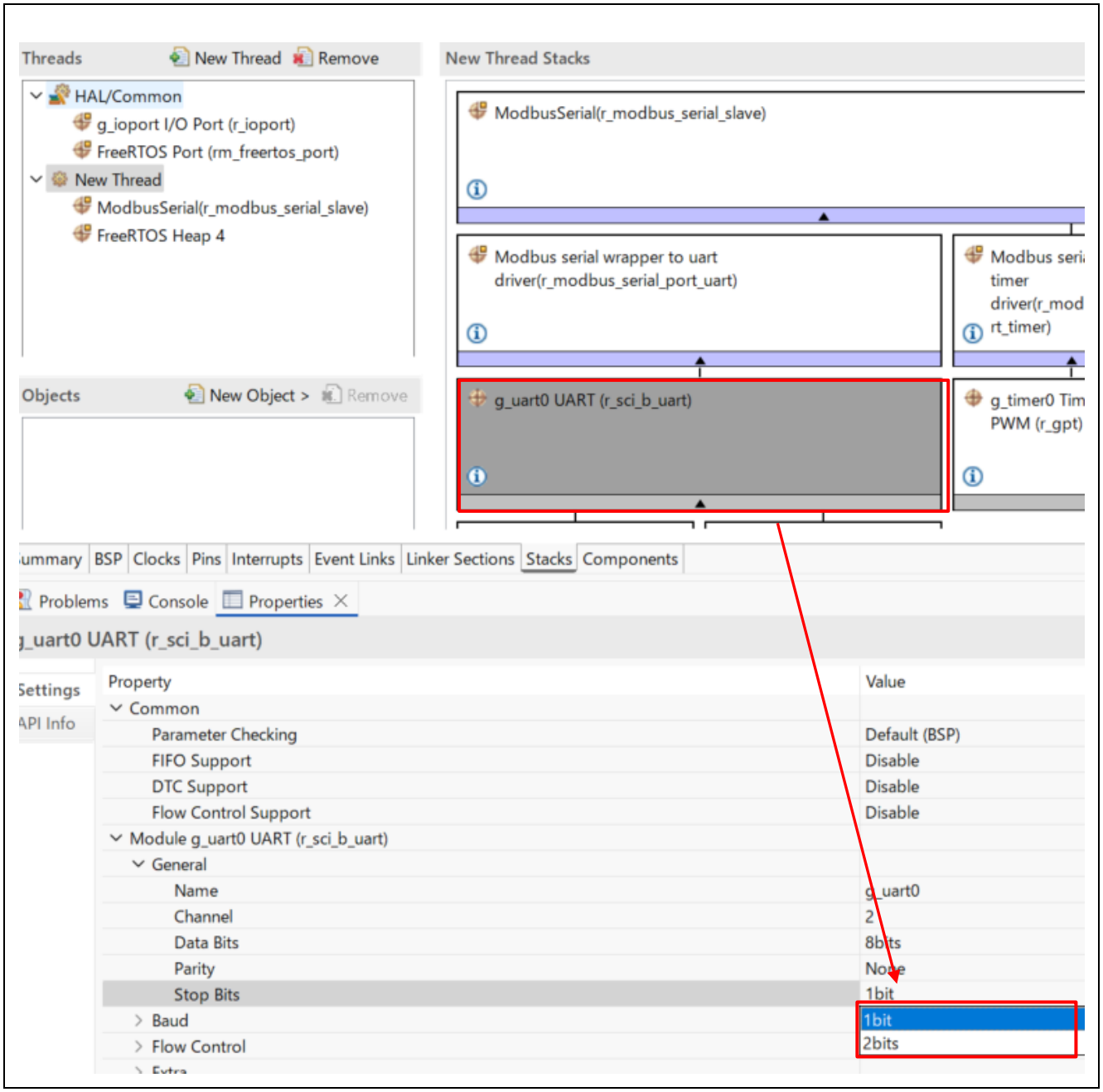


5.3.4　ストップビットの設定

ストップビットの設定について説明します。

1. UART の設定
   「g_uart0 UART (r_sci_b_uart)」の「Properties」を展開し、「Stop Bits」を以下の値に変更します。

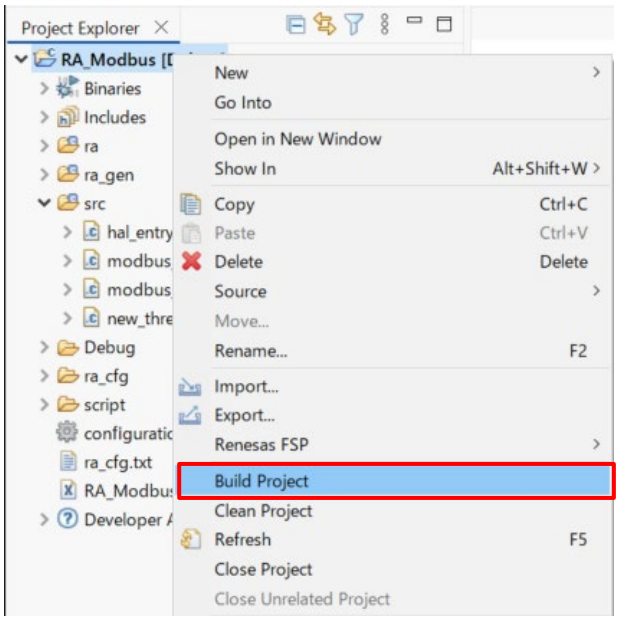
- 1 ストップビットで通信する場合
  Stop Bits： **1bit**
- 2 ストップビットで通信する場合
  Stop Bits： **2bits**



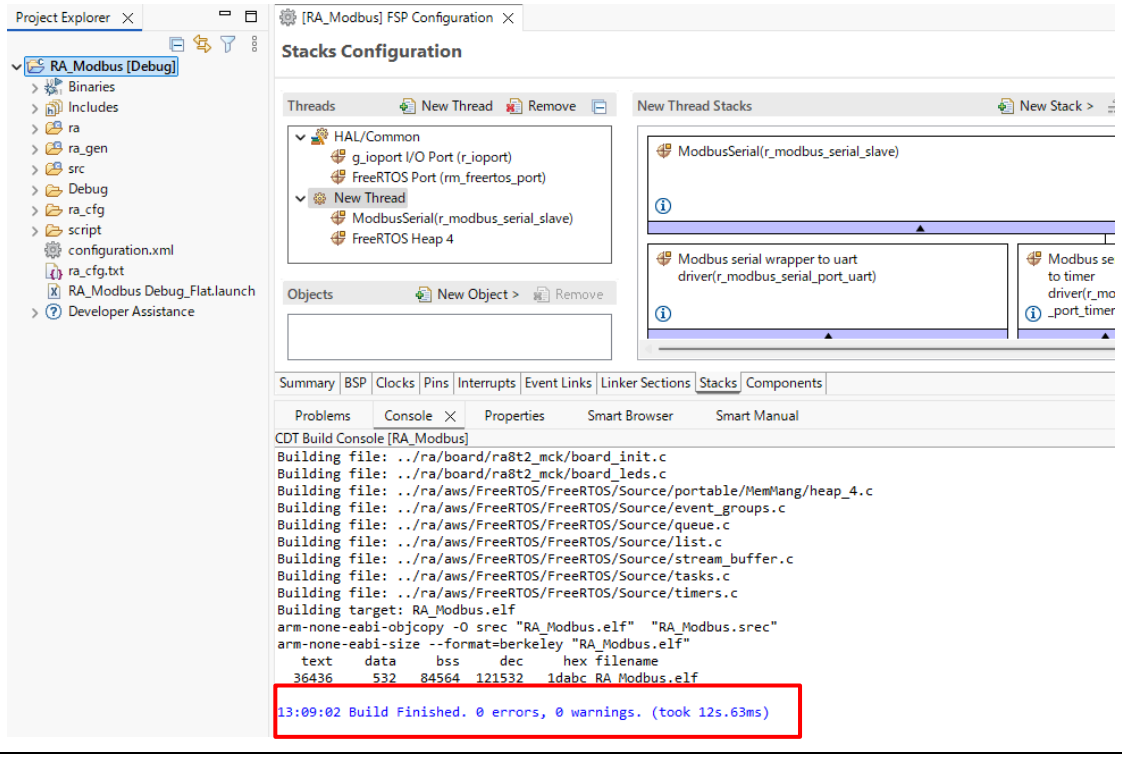
6. Modbus サンプルプロジェクトの実行

この章では、Modbus サンプルプロジェクトを実行する手順について説明します。
事前に「**4.2. ボード設定**」を参照し、ハードウェアの接続を完了してください。
また、「**5. Modbus サンプルプロジェクトの構築**」を参照し、Modbus サンプルプロジェクトを用意してください。

1. ビルド実行
   [Project Explorer]ビューで、ビルドするプロジェクトのノードを右クリックし、 [Build Project]を選択します。



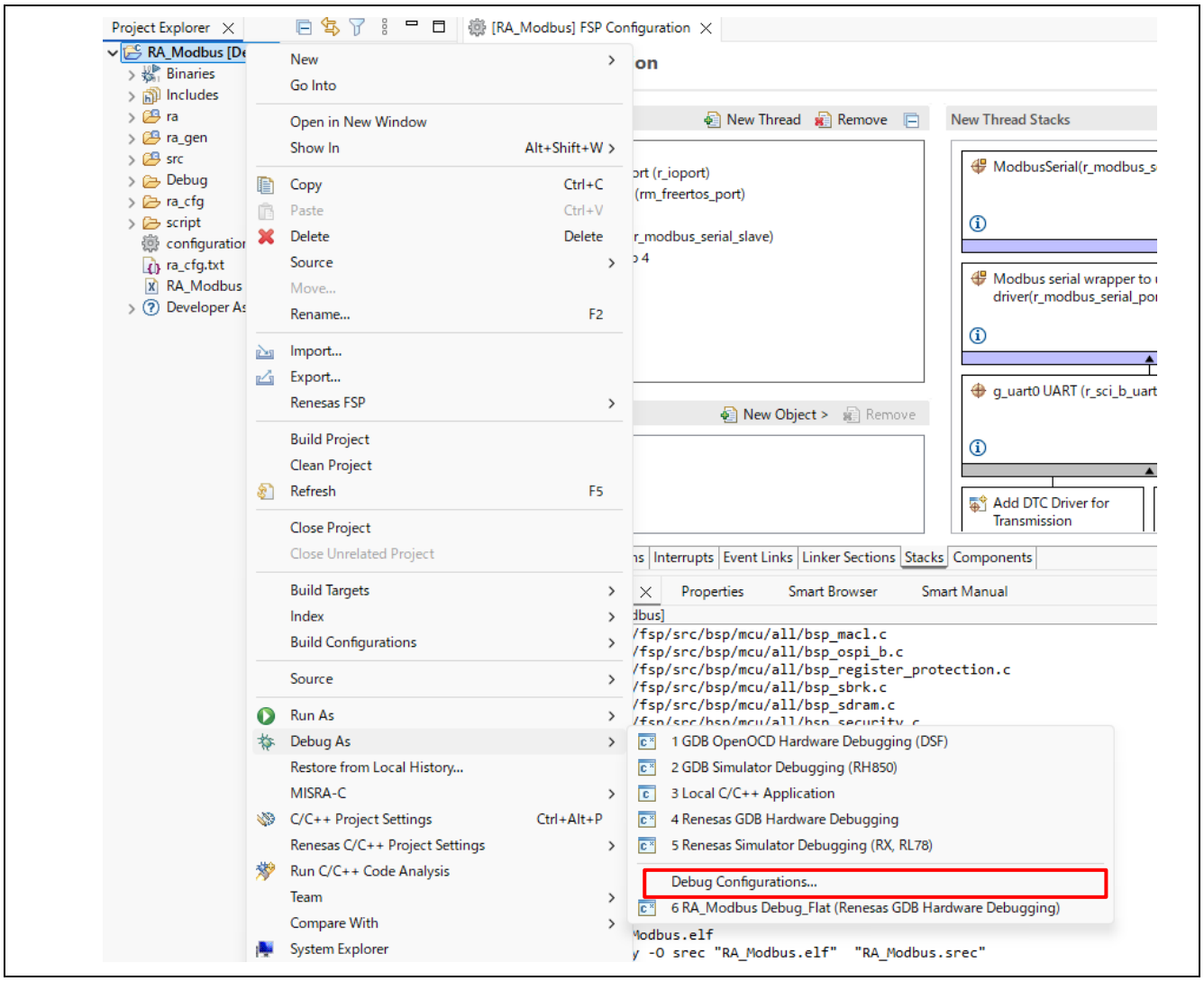
このとき、ビルドエラーがないことを確認してください。



2. アプリケーションのダウンロードとデバッガの実行

以下の手順でデバッグを開始します。

[Project Explorer]ビューで、デバッグするプロジェクトのノードを右クリックし、[Debug As]→[Debug Configurations]を選択します。

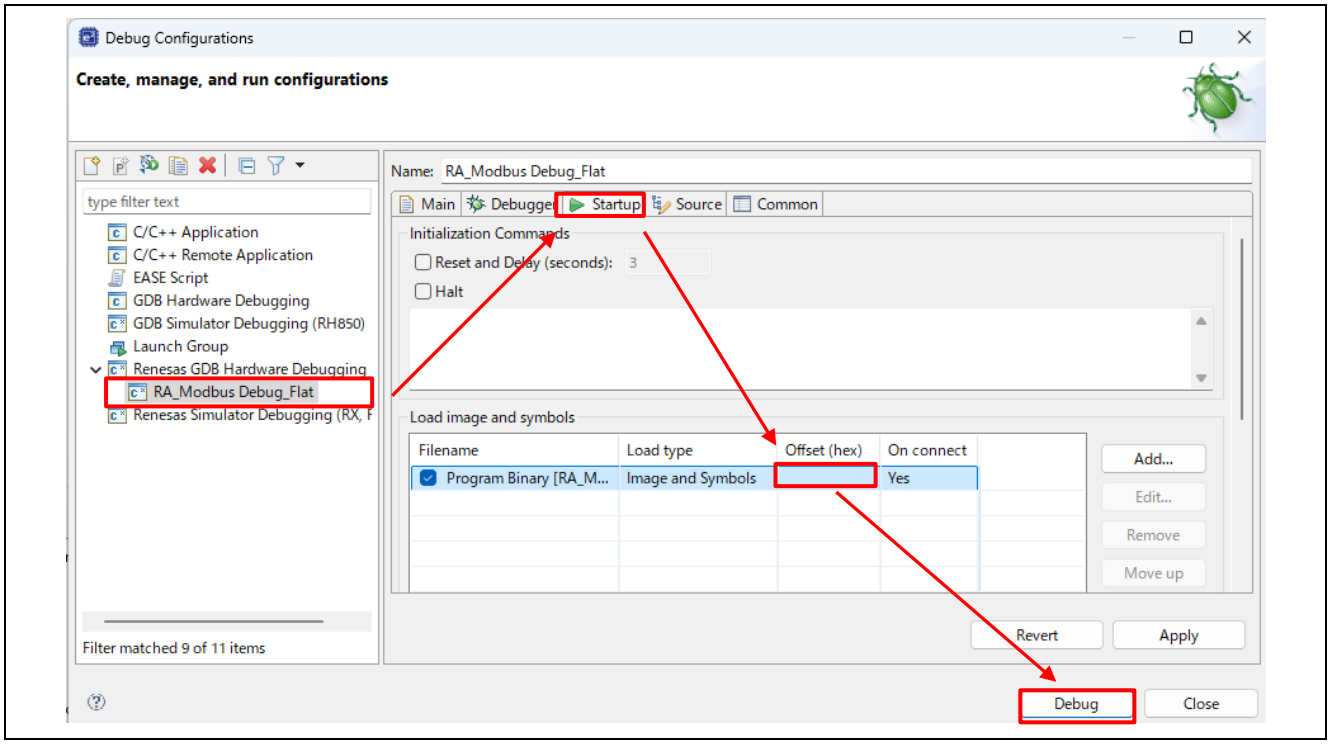


3. プログラムのダウンロード

Toolchain が Arm Compiler の場合：

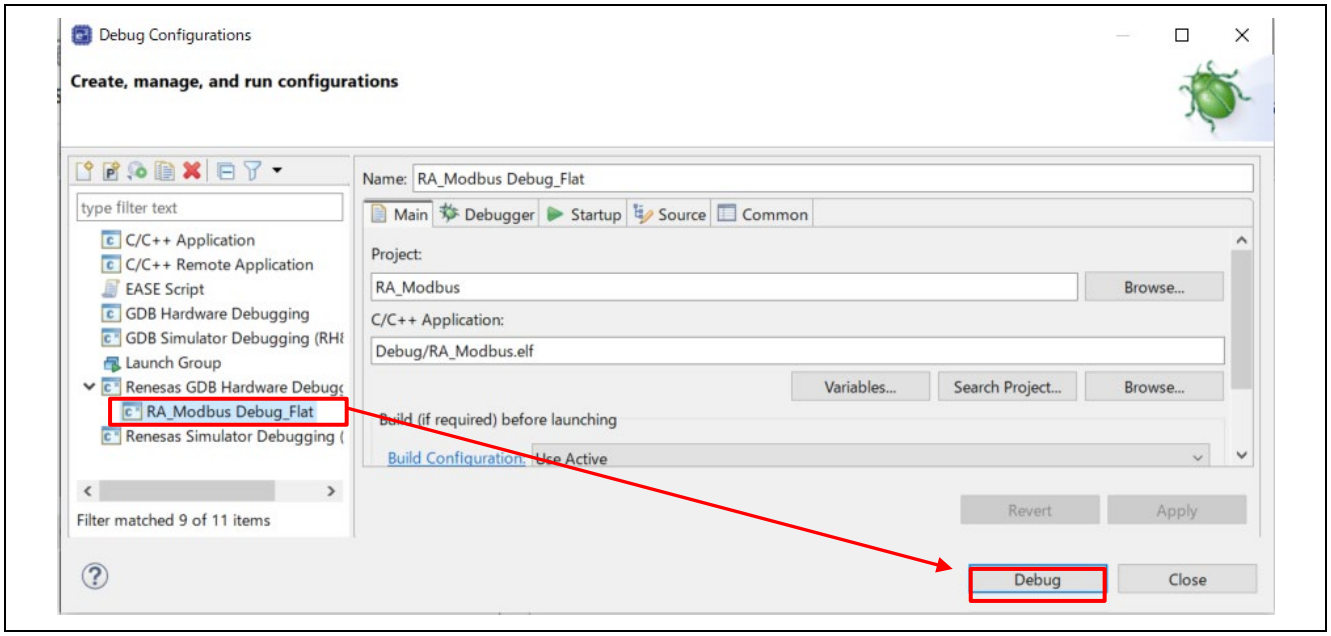
[Renesas GDB Hardware Debugging] → [RA_Modbus Debug_Flat]を選択し、[Startup]タブを選択します。

[Load image and symbols]の[Offset (hex)]値”0”を削除し、[Debug]を押します。

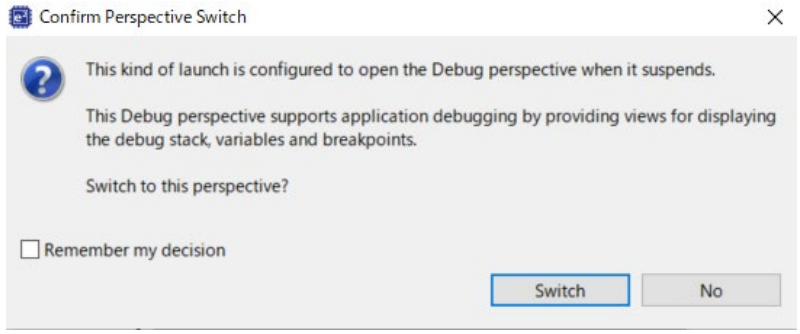


Toolchain が Arm Compiler 以外の場合：

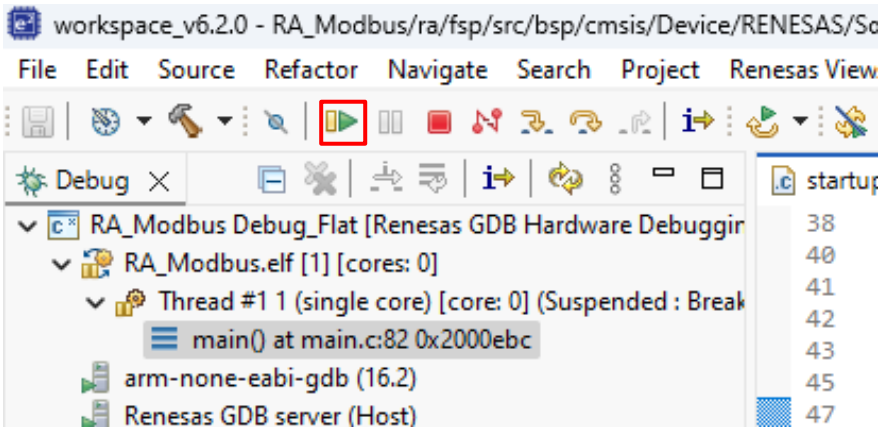
[Renesas GDB Hardware Debugging] → [RA_Modbus Debug_Flat]を選択し、[Debug]を押します。



以下のダイアログが表示されるので、デバッグ画面に切り替えます。



4. プログラムの開始
"Resume" ボタンを押します。
デバッグが開始されると、プログラムは「main.c;」で中断されます。
もう一度「Resume」ボタンを押してください。 プログラムが実行されます。



7. Modbus デモアプリケーションを用いた Modbus 通信デモ

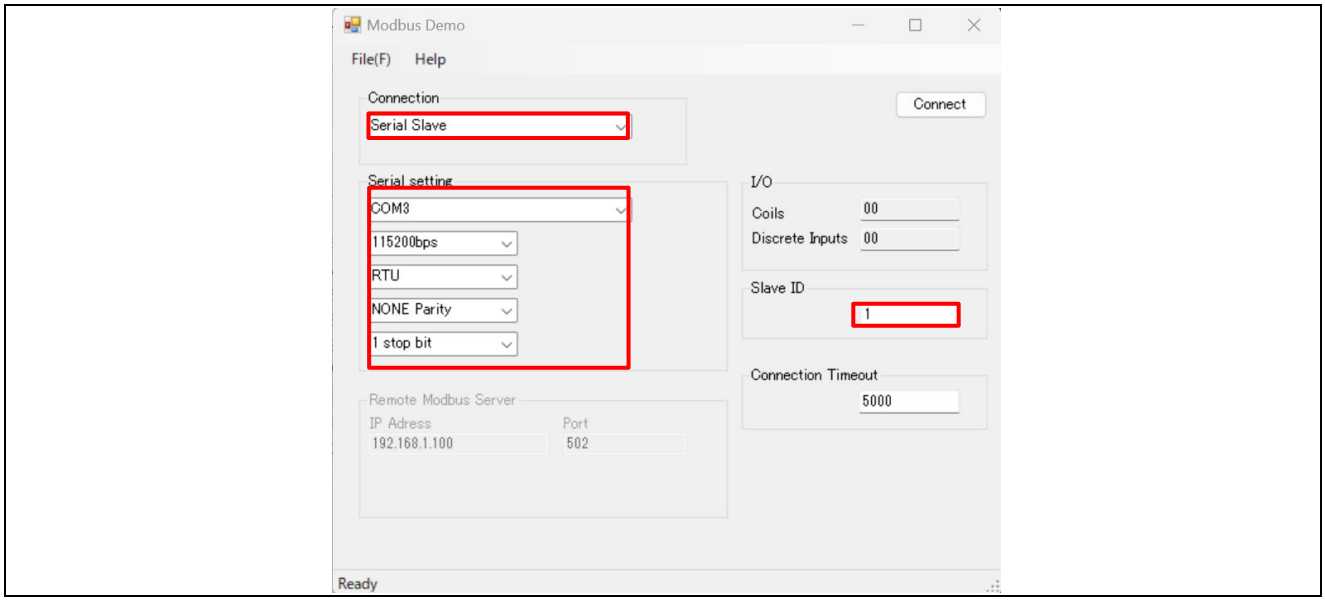
この章では、Modbus デモアプリケーションを使用して、Modbus サンプルアプリケーションのデモ動作を確認する手順を説明します。Modbus プロトコルスタックのコンフィグレーションについては「**8.1. Appendix A: Modbus Protocol Stack Configuration**」を参照してください。

7.1 Modbus デモアプリケーションの設定

このサンプルプログラムパッケージに含まれている「ModbusDemoApplication.exe」を起動し、以下の設定をします。

Connection：**Serial Slave**
Serial setting：
　シリアルポート番号(例「COM3」)
　ボーレート(例:「115200bps」)
　通信モード(例「RTU」)
　パリティ(例「NONE Parity」)
　ストップビット(例「1 stop bit」)
Slave ID：スレーブ ID(例「1」)

※「ModbusDemoApplication.exe」の設定は「**5.3 シリアル伝送モード / ボーレート / パリティビット / ストップビットの設定**」と「**8.1. Appendix A: Modbus Protocol Stack Configuration**」に記載している Modbus サンプルプロジェクトの設定に合わせてください。



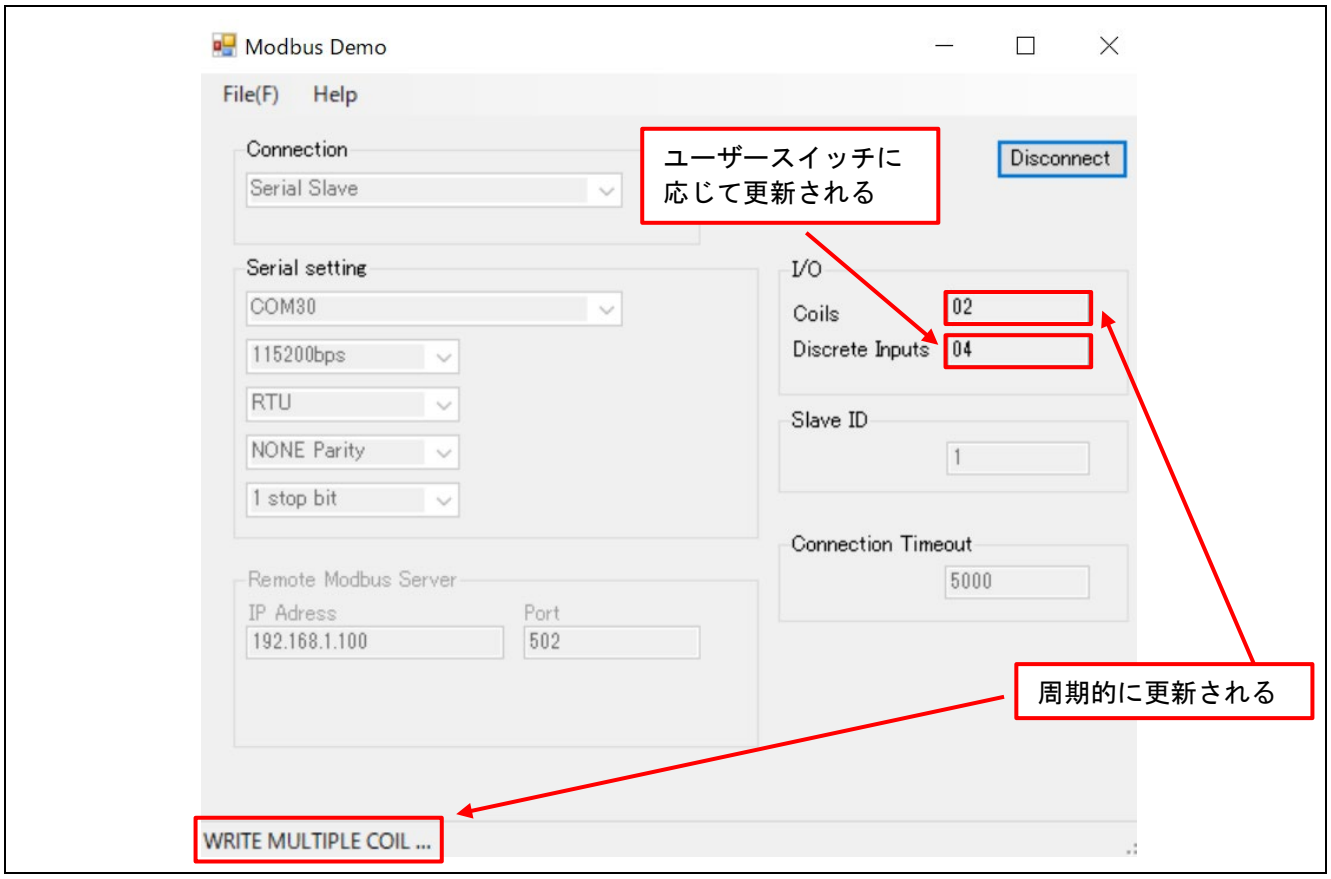
7.2　Modbus デモアプリケーションの仕様

このサンプル プロジェクトで Modbus プロトコル スタックを使用した簡単なデモンストレーションを確認することができます。

Modbus RTU / ASCII プロトコルを介して PC と通信することにより、LED の点滅速度を動的に制御します。

この制御では"Read_Discrete_Inputs" および "Write_Single_Coil" ファンクションコードが使用されます。「Connect」ボタンをクリック後、LED1～3 が周期的に点滅します。

「**8.3.2 User-defined Functions**」に記載されているユーザースイッチの ON/OFF に応じて「Discrete Inputs」が更新されます。



8. Appendix

8.1 Appendix A: Modbus Protocol Stack Configuration

Modbus プロトコルスタックの FSP コンフィグレーションを以下に記述します。

Configuration	Options	Default	Description
Common			
Parameter Checking	・Default (BSP) ・Enable ・Disable	Default (BSP)	Enabled を選択した場合、パラメーター チェックのコードがビルドに含まれます。
Receive task priority	有効な値の範囲は 0 から (Max Priorities - 1) です	2	Receive task 優先度。 有効な値の範囲は 0 から (Max Priorities - 1) です。
Receive task stack size	有効な値は 0x400 以上です	0x400	Receive task スタックサイズ。 有効な値は 0x400 以上です。
Send task priority	有効な値の範囲は 0 から (Max Priorities - 1) です	2	Send task 優先度。 有効な値の範囲は 0 から (Max Priorities - 1) です。
Send task stack size	有効な値は 0x400 以上です	0x400	Send task スタックサイズ。 有効な値は 0x400 以上です。
Serial Mailbox Receive Max Elements	有効な値は 8 以上です	8	タスク間でデータを渡すキューの長さ。
Slave ID	有効な値は 1 から 247 です	1	Modbus 標準ではスレーブ ID を 1 ～ 247 と定義しています。
Module ModbusSerial(r_modbus_serial_slave)			
Name	名前は有効な C Symbol である必要があります	g_modbus_serial_slave0	モジュール名
Serial Stack Mode	・MODBUS_SERIAL_SLAVE_RTU_MODE ・MODBUS_SERIAL_SLAVE_ASCII_MODE	MODBUS_SERIAL_SLAVE_RTU_MODE	MODBUS シリアルスタックが RTU / ASCII スレーブとして動作するモード。目的のスタックモードを選択します。
Serial Communication Speed	・9600 ・19200 ・115200	115200	UART 通信速度
Callback for Function Code	名前は有効な C Symbol である必要があります	function_code_callback	「Function Code」のコールバック関数名を入力してください。

8.2 Appendix B. Application Programming Interface

関数

· R_MODBUS_SERIAL_SLAVE_Open

概要	MODBUS シリアルスレーブスタックはこの関数で初期化されます。	
関数定義	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)	
パラメーター	modbus_serial_slave_handle_t * const p_handle	コントロールブロック。 Modbus シリアル API 呼び出しに渡すインスタンス固有の制御ブロックを割り当てます。
	modbus_serial_slave_cfg_t const * const p_cfg	Modbus シリアルコンフィグレーションパラメーター
戻り値	Error code	
エラーコード	MODBUS_ERR_OK	MODBUS シリアルスレーブ スタックの初期化に成功しました。
	MODBUS_ERR_ALREADY_OPEN	MODBUS シリアルスレーブ スタックは既に初期化されています。
実行例	R_MODBUS_SERIAL_SLAVE_Open(&g_modbus_serial_slave0_ctrl, &g_modbus_serial_slave0_cfg);	

· R_MODBUS_SERIAL_SLAVE_Close

概要	MODBUS シリアルスレーブスタックはこの関数で終了します。	
関数定義	modbus_err_t R_MODBUS_SERIAL_SLAVE_Close (modbus_serial_slave_handle_t * const p_handle)	
パラメーター	modbus_serial_slave_handle_t * const p_handle	コントロールブロック。 Modbus シリアル API 呼び出しに渡すインスタンス固有の制御ブロックを割り当てます。
戻り値	Error code	
エラーコード	MODBUS_ERR_OK	MODBUS シリアルスレーブスタックの終了に成功しました。
	MODBUS_ERR_STACK_TERM	MODBUS シリアルスレーブスタックの終了に失敗しました。
	MODBUS_ERR_NOT_OPEN	MODBUS シリアルスレーブスタックが初期化されていません。
実行例	R_MODBUS_SERIAL_SLAVE_Close(&g_modbus_serial_slave0_ctrl);	

8.3 Appendix C: User-defined function

このセクションでは、Modbus サンプルアプリケーションについて説明します。 ユーザーは、Modbus ファンクションコードの独自の実装を Modbus プロトコルスタックに登録できます。

8.3.1 Register Function Code

定義ファイル: src/modbus_func.c
Modbus プロトコルスタックのコールバック関数に登録する関数を定義します。

8.3.2 User-defined Functions

ユーザー定義関数は src/modbus_user.c で定義されます。
各関数の処理にはユーザー定義の Read / Write 関数を使用します。
一部の関数は以下の評価ボードの部品にアクセスします。

評価ボード	ユーザーLED			ユーザースイッチ	
	①	②	③	①	②
MCK-RA8T2	LED1	LED2	LED3	-	-
EK-RA8M2	ユーザー LED1	ユーザー LED2	ユーザー LED3	ユーザースイッチ SW1	ユーザースイッチ SW2
EK-RA8T2	ユーザー LED1	ユーザー LED2	ユーザー LED3	ユーザースイッチ SW1	ユーザースイッチ SW2

コイル/ディスクリート入力/ホールディングレジスタ/入力レジスタの各アドレスに対応する Read/Write 関数とそのテーブルを用意しています。各テーブルの関数がアクセスする評価ボード部品とグローバル変数は以下の通りです。

【Read Coils】	
address	**Access**
0001	上記表ユーザーLED①, g_coils_area
0002	上記表ユーザーLED②, g_coils_area
0003	上記表ユーザー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	上記表ユーザーLED①, g_coils_area **注 1**
0002	上記表ユーザーLED②, g_coils_area **注 1**
0003	上記表ユーザーLED③, g_coils_area **注 1**
0004	g_coils_area
0005	g_coils_area
0006	g_coils_area
0007	g_coils_area
0008	g_coils_area

注 1：
この関数で Write した値により評価ボードの各 LED が点灯/消灯する。

【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	上記表ユーザースイッチ①, g_discrete_input_area **注 2**
1007	上記表ユーザースイッチ②, g_discrete_input_area **注 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

注 2：
この関数で Read した値が
Modbus デモアプリケーションの
”Discrete Inputs”に表示される。

【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. 制限事項

制限事項なし

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2025.12.26	-	初版発行
1.10	2026.3.31		EK-RA8T2 の追加
		P.3	表 1.2 を更新
		P.5	3.1 Modbus サンプルプロジェクトを更新
		P.9	図 4.3、表 4.1 を追加
		P.11	Modbus サンプルプロジェクトを実行する手順を追加
		P.28～33	5.2.3 EK-RA8T2 用作成手順を追加
			動作環境の更新
		P.6	表 4.1 を更新
		P.44	設定の記述を変更
		P.45	Discrete Inputs に関する記載を追加
		P.48	ユーザーLED、ユーザースイッチの表を更新

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。