

RA4W1 グループ

Bluetooth Mesh サンプルアプリケーション

要旨

本書は Bluetooth® Mesh スタックを使用したサンプルアプリケーションについて説明します。Bluetooth® Mesh スタックは、Bluetooth Mesh Networking 仕様に準拠した Mesh ネットワークを構成して、多対多デバイス間で無線通信を実行するためのソフトウェアライブラリです。

以後、本書では Bluetooth® Mesh を Mesh と称します。

本サンプルアプリケーションを使用した Mesh デモの実行方法については「RA4W1 グループ Bluetooth Mesh スタートアップガイド」(R01AN5847)を参照してください。

対象デバイス

RA4W1 グループ

関連ドキュメント

- Bluetooth Core 仕様 (<https://www.bluetooth.com>)
- Mesh Profile 仕様 (<https://www.bluetooth.com>) - 「Mesh Profile 1.0.1」を検索
- Mesh Model 仕様 (<https://www.bluetooth.com>) - 「Mesh Model 1.0.1」を検索
- Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)
- e² studio ユーザーズマニュアル 入門ガイド (R20UT4374)
- RA4W1 グループ BLE サンプルアプリケーション (R01AN5402)
- RA4W1 グループ Bluetooth Mesh スタートアップガイド (R01AN5847)
- RA4W1 グループ Bluetooth Mesh 開発ガイド (R01AN5849)

目次

1. 概要	3
1.1 デモプロジェクト	3
1.2 Mesh スタック機能	5
1.3 ソフトウェア構成	6
1.4 ファイル構成	7
1.5 API 仕様	7
1.6 動作環境	8
2. デモプロジェクトの使用方法	9
2.1 デモプロジェクトのインポート	9
2.2 ビルドとデバッグ	11
3. 新規プロジェクトの作成と設定方法	12
3.1 新規プロジェクトの作成	12
3.2 ヒープとスタックの設定	15
3.3 クロックの設定	16
3.4 Mesh スタックの追加と設定	17
3.4.1 ベアメタル環境での Mesh スタック追加	17
3.4.2 FreeRTOS 環境での Mesh スタック追加	20
3.4.3 Mesh スタック設定	25
3.4.4 その他の Mesh スタック設定	27
3.5 関連モジュールの追加と設定	32
3.5.1 r_gpt (g_timer0)	33
3.5.2 r_flash_lp	34
3.5.3 r_icu (g_external_irq0)	35
3.5.4 r_gpt (g_timer1)	36
3.5.5 r_icu (g_ble_sw_irq)	37
3.5.6 r_sci_uart	38
3.5.7 r_lpm	40
3.6 コードの生成	41
3.7 ビルドとデバッグ	41
4. Mesh アプリケーションの実装方法	42
5. Appendix	43
5.1 プログラムサイズ	43
改訂記録	45

1. 概要

1.1 デモプロジェクト

本書に同梱されているサンプルアプリケーションのプロジェクトを表 1-1 に示します。

表 1-1 プロジェクト

プロジェクト名	説明
ekra4w1_mesh_client_baremetal	EK-RA4W1 用の FreeRTOS を使用しない Client Models プロジェクト
ekra4w1_mesh_client_freertos	EK-RA4W1 用の FreeRTOS を使用する Client Models プロジェクト
ekra4w1_mesh_server_baremetal	EK-RA4W1 用の FreeRTOS を使用しない Server Models プロジェクト
ekra4w1_mesh_server_freertos	EK-RA4W1 用の FreeRTOS を使用する Server Models プロジェクト
ekra4w1_mesh_cli_client_baremetal	EK-RA4W1 用の FreeRTOS を使用しない Command Line Interface(CLI)プロジェクト
ekra4w1_mesh_cli_server_baremetal	EK-RA4W1 用の FreeRTOS を使用しない Command Line Interface(CLI)プロジェクト

これらのプロジェクトは EK-RA4W1 上で動作します。

Server Models プロジェクトはサーバーモデルを実行します。クライアントモデルのリモートデバイス(スマートフォンなど)からのメッセージを受信し、ボード上の LED を点滅し、受信した文字列を表示します。

Client Models プロジェクトはクライアントモデルを実行します。USB ケーブルで EK-RA4W1 ボードと接続した PC 上のターミナルエミュレータ(Tera Term など)からアクセスできる CLI の機能を備えています。ボード上のスイッチを押すか CLI に入力することによってメッセージを送信し、サーバーモデルのリモートデバイス上の LED を点滅させ、文字列を表示させることができます。

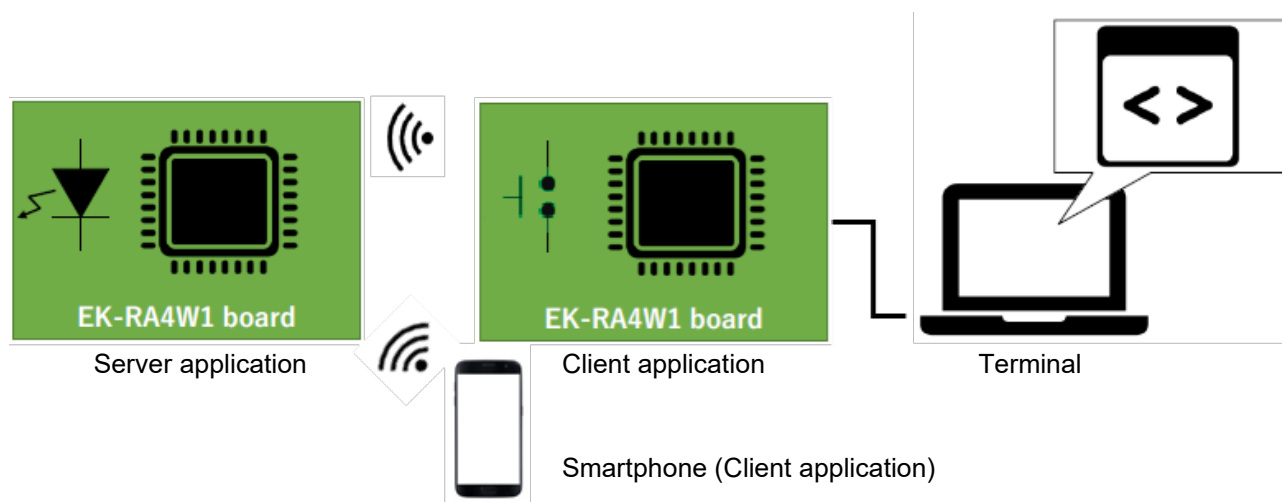


図 1-1 プロジェクト概要(サーバーとクライアント)

CLI プロジェクトは Mesh 仕様に定義されたすべてのモデルを実行できます。CLI を使用してメッセージを送受信することによって Mesh に関する様々な手続きを実行することができます。

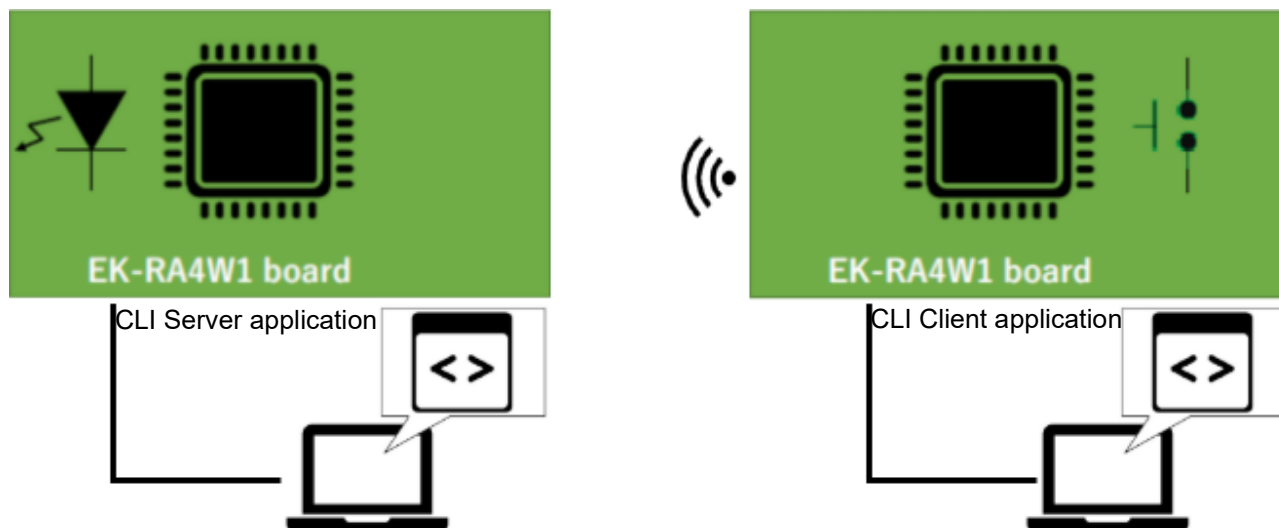


図 1-2 プロジェクト概要(CLI)

1.2 Mesh スタック機能

Mesh スタックは Bluetooth Mesh Profile 1.0.1 仕様および Bluetooth Mesh Model 1.0.1 仕様に準拠した多対多の無線通信機能をアプリケーションに提供します。本スタックがサポートする機能を以下に示します。

Bluetooth Core Mesh Profile 機能:

- Provisioning (Provisioning Server および Provisioning Client)
- Access
- Upper Transport
 - Friendship (Friend feature および Low Power feature)
- Lower Transport
- Network
 - Relay
 - Proxy (Proxy Server および Proxy Client)
- Bearer
 - ADV Bearer
 - GATT Bearer
- Foundation Model
 - Configuration Model (Configuration Server および Configuration Client)
 - Health Model (Health Server および Health Client)

Bluetooth Mesh Model 機能:

- Generic Models
 - OnOff, Power OnOff, Power OnOff Setup
 - Level, Power Level, Power Level Setup
 - Default Transition Time
 - Battery
 - Location, Location Setup
 - Manufacturer Property, Admin Property, User Property, Client Property
- Sensor Model
 - Sensor, Sensor Setup
- Time Model
- Scene Model
 - Scene, Scene Setup
- Scheduler Model
 - Scheduler, Scheduler Setup
- Light Models
 - Light Lightness, Light Lightness Setup
 - Light CTL, Light CTL Setup
 - Light HSL, Light HSL Setup
 - Light xyL, Light xyL Setup
 - Light Control

1.3 ソフトウェア構成

図 1-3 に Mesh スタックを使用するソフトウェアの構成を示します。

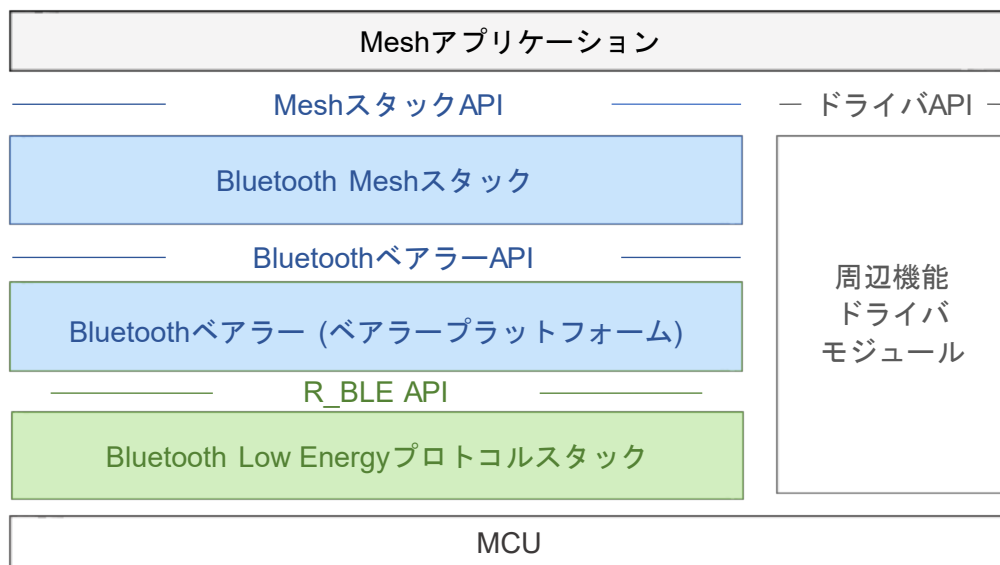


図 1-3 ソフトウェア構成

Mesh スタックを使用するソフトウェアは下記で構成されます。

- **Mesh アプリケーション**
Mesh アプリケーションは Bluetooth Mesh スタックが提供する機能を実行するプログラムです。
- **Bluetooth Mesh スタック**
Bluetooth Mesh スタックは Bluetooth Mesh Networking 仕様に準拠した多対多の無線通信機能をアプリケーションに提供するソフトウェアです。
- **Bluetooth ベアラ(ベアラプラットフォーム)**
Bluetooth ベアラは Bluetooth Low Energy プロトコルスタックのラッパー関数を Bluetooth Mesh スタックとアプリケーションに提供する抽象化レイヤーです。
- **Bluetooth Low Energy プロトコルスタック**
Bluetooth Low Energy プロトコルスタック(以後、"Bluetooth LE スタック"と表記)は Bluetooth Low Energy 仕様に準拠した無線通信機能を上位レイヤーに提供するソフトウェアです。

Mesh アプリケーションのサンプルプログラムは、本書に同梱のデモプロジェクトに含まれます。

Bluetooth Mesh スタックと Bluetooth ベアラは、[FSP](#) として提供されます。

Bluetooth LE スタックは、[FSP](#) として提供されます。

1.4 ファイル構成

デモプロジェクトのファイル構成を以下に示します。

ekra4w1_mesh_xxx_yyy	プロジェクトフォルダ
+---ra\fsp\inc\api\ r_ble_api.h rm_ble_mesh_xxx_api.h rm_mesh_bearer_platform_api.h 	Bluetooth LE スタック API ヘッダファイル Mesh スタック API ヘッダファイル Bluetooth ベアラーAPI ヘッダファイル
+---ra\fsp\inc\instances\ rm_ble_mesh_xxx.h rm_mesh_bearer_platform.h rm_mesh_xxx.h 	Mesh スタックヘッダファイル Bluetooth ベアラーヘッダファイル Mesh スタックヘッダファイル(Model 機能)
+---ra\fsp\lib\r_ble\ +---ra\fsp\lib\rm_ble_mesh\ 	Bluetooth LE スタックライブラリ Mesh スタックライブラリ
+---ra_cfg\fsp_cfg\ r_ble_cfg.h rm_ble_mesh_cfg.h 	Bluetooth LE スタック設定 Mesh スタック設定
+---src\ 	Mesh アプリケーション

Mesh スタックが提供する機能を利用するには、Mesh スタックをプロジェクトに組み込む必要があります。Mesh スタックの組み込み方法は、本書の 3 章を参照してください。

1.5 API 仕様

Mesh スタックの機能を実行するには、Mesh スタックの API を利用します。Mesh スタック API の仕様は、「Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)」を参照してください。

1.6 動作環境

デモプロジェクトをビルドおよびデバッグしたハードウェア動作確認環境を表 1-2 に示します。

表 1-2 ハードウェア環境

ハードウェア	説明
ホスト PC	USB インタフェースを搭載した Windows® 10 PC
MCU ボード	使用される MCU は BLE 機能をサポートする必要があります。 EK-RA4W1 [RTK7EKA4W1S00000BJ]
オンチップデバッグエミュレータ	EK-RA4W1 はオンボードデバッガ(J-Link OB)を搭載しているため、エミュレータを用意する必要はありません。
USB ケーブル	MCU ボードとの接続に使用します。

デモプロジェクトをビルドおよびデバッグしたソフトウェア動作確認環境を表 1-3 に示します。

表 1-3 ソフトウェア環境

ソフトウェア		バージョン	説明
GCC 環境	e² studio	2022-10	ルネサス製デバイス用の統合開発環境
	GCC ARM Embedded	V10	C/C++コンパイラ (e² studio のインストーラからダウンロード選択できます)
	Renesas Flexible Software Package (FSP)	V4.2.0	RA マイコンでアプリケーション開発を行うためのソフトウェアパッケージ
	SEGGER J-Flash	V6.86	マイコンのフラッシュメモリ書き込みツール
整数型			ANSI C99 を使用しています。これらの型は stdint.h で定義されています。
エンディアン			リトルエンディアン

2. デモプロジェクトの使用方法

本書に同梱のデモプロジェクトの使用方法について示します。

2.1 デモプロジェクトのインポート

以下の手順を実行します。

1. e² studio を起動しワークスペースディレクトリを指定します。

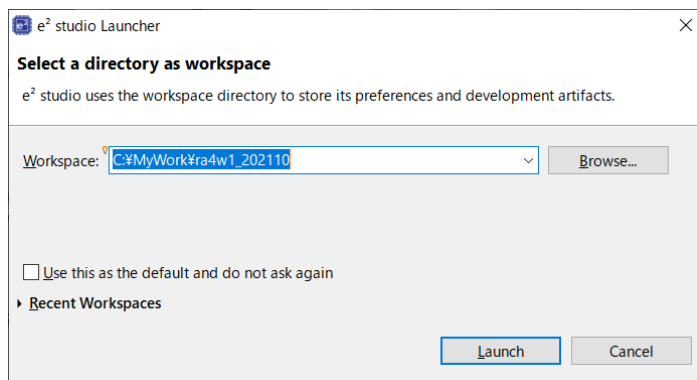


図 2-1 ワークスペース選択

2. File→Import を選択します。

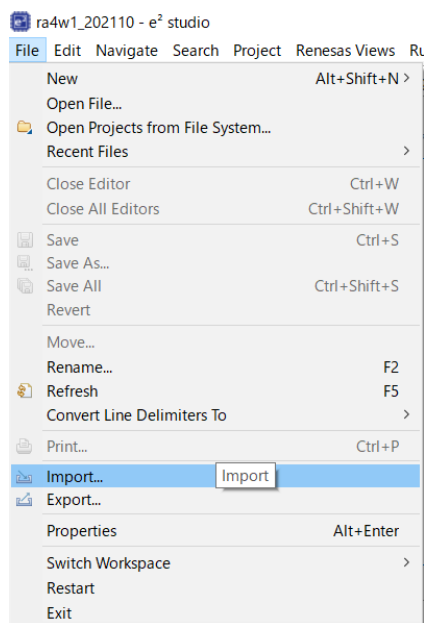


図 2-2 ファイルメニュー

3. 「Existing Projects into Workspace」を選択して「Next」をクリックします。

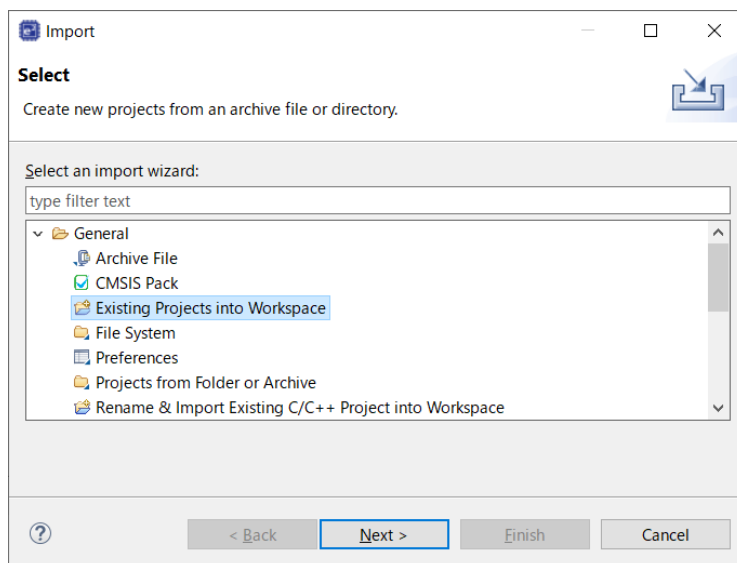


図 2-3 インポートウィザードの選択

4. 「Select root directory」を選択して「Browse...」をクリックし、デモプロジェクトのフォルダを選択します。「Finish」ボタンをクリックするとデモプロジェクトがインポートされます。

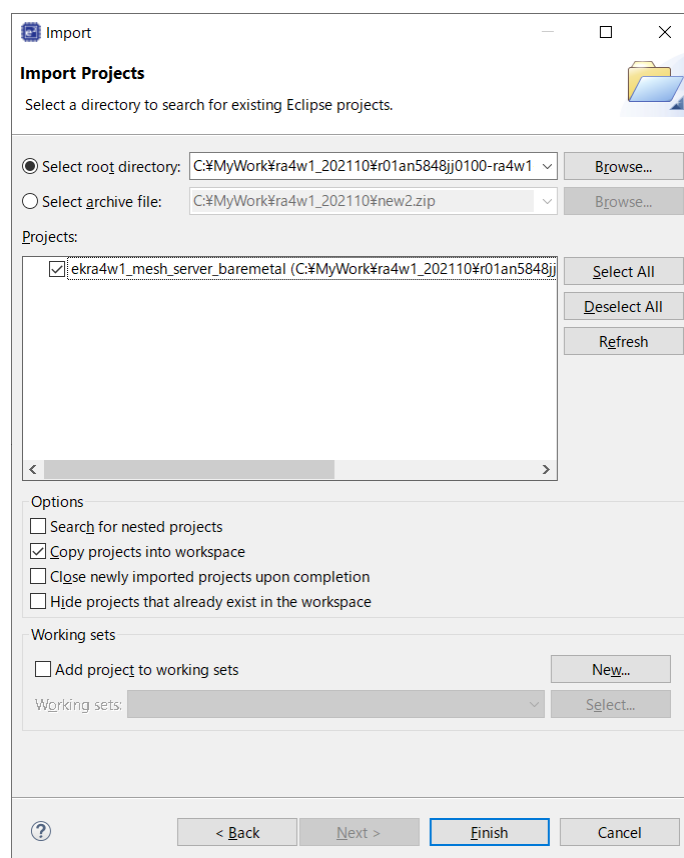


図 2-4 プロジェクトのインポート

2.2 ビルドとデバッグ

以下の手順を実行します。

e² studio でのデバッグについては「e² studio ユーザーズマニュアル 入門ガイド」(R20UT4374)の 5 章「デバッグ」を参照してください。

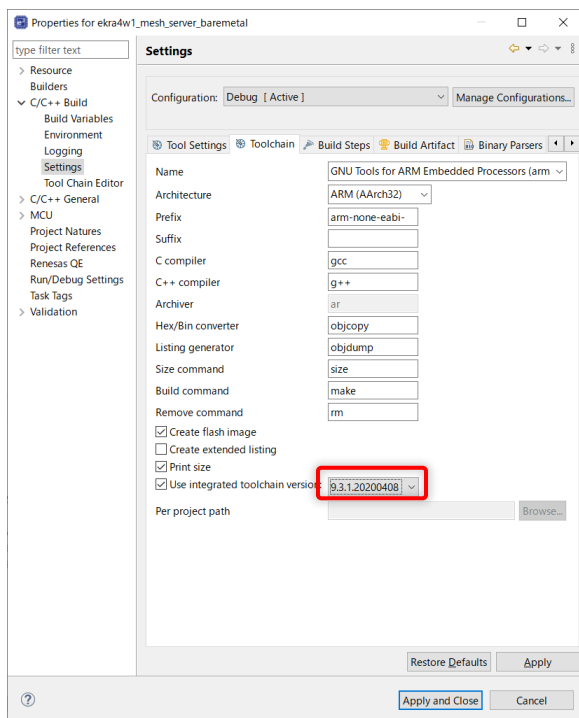
1. [Project]メニューの[Build Project]を選択するか、ビルドアイコン  をクリックしてプロジェクトをビルドします。[Console]タブで、ビルドログに続いて"Build Finished"と表示されれば、プロジェクトのビルドは成功です。

現在のプロジェクトは Launch Configuration  ekra4w1_mesh_server_barer で確認と変更ができます。
ビルド後、ファームウェア(.srec ファイル)がプロジェクトディレクトリの Debug\内に生成されます。

2. EK-RA4W1 を PC に接続します。
3. デバッグアイコン  をクリックしてデバッグモードでプロジェクトを起動します。
プロジェクトの起動後、ファームウェアが EK-RA4W1 にダウンロードされます。
4. デバッグパースペクティブの再開アイコン  をクリックしてプロジェクトを実行します。
5. プロジェクトのデバッグ後は、デバッグパースペクティブの終了アイコン  をクリックします。
プロジェクトのファームウェアは終了と電源オフ後も RA4W1 のフラッシュメモリに保持されます。

デモを実行するため 2 台以上の EK-RA4W1 を使用することを推奨します（少なくとも 1 台が Client、その他が Server）。

注: 「No toolchain set or toolchain not integrated.」とエラーが発生し、ビルドに失敗する場合は、[Project]→[C/C++ Project Settings]を開き、[C/C++ Build]→[Settings]→[Toolchain]タブに移動して、ツールチェーン(9.3.1.20200408 以降)を選択してください。



3. 新規プロジェクトの作成と設定方法

e² studio 上で FSP Configuration を使用して、新規プロジェクトに Mesh スタックを追加する方法について説明します。

3.1 新規プロジェクトの作成

1. e² studio を起動して [File]→[New]→[Renesas C/C++ Project]→[Renesas RA] を選択します。New C/C++ Project ダイアログで Renesas RA C/C++ Project を選択して Next ボタンをクリックします。

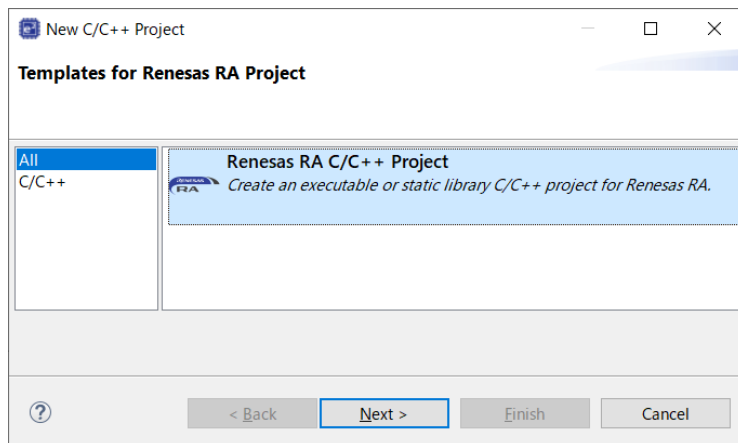


図 3-1 Templates for New C/C++ Project

2. プロジェクト名を入力して Next ボタンをクリックします。本書では sample_appl というプロジェクト名を付けました。

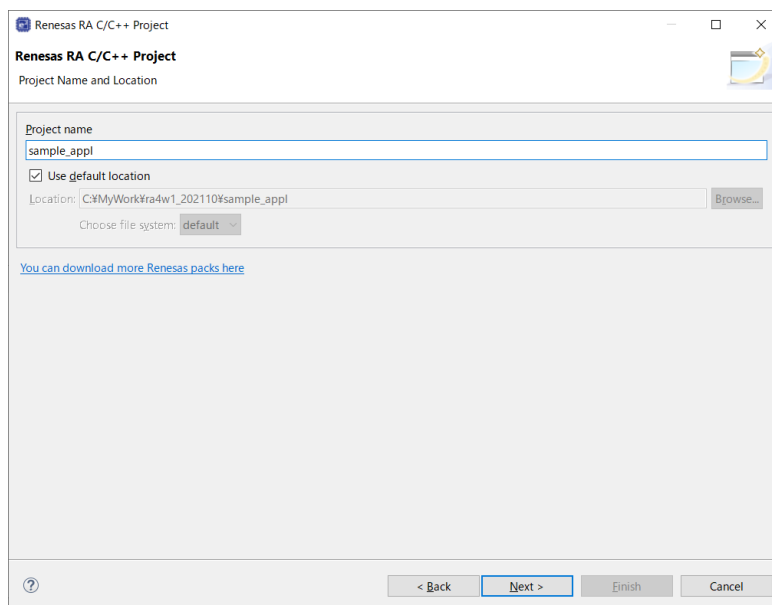


図 3-2 New Renesas Executable Project

3. Board から Custom User Board (Any Device)、Device から R7FA4W1AD2CNG を選択します。

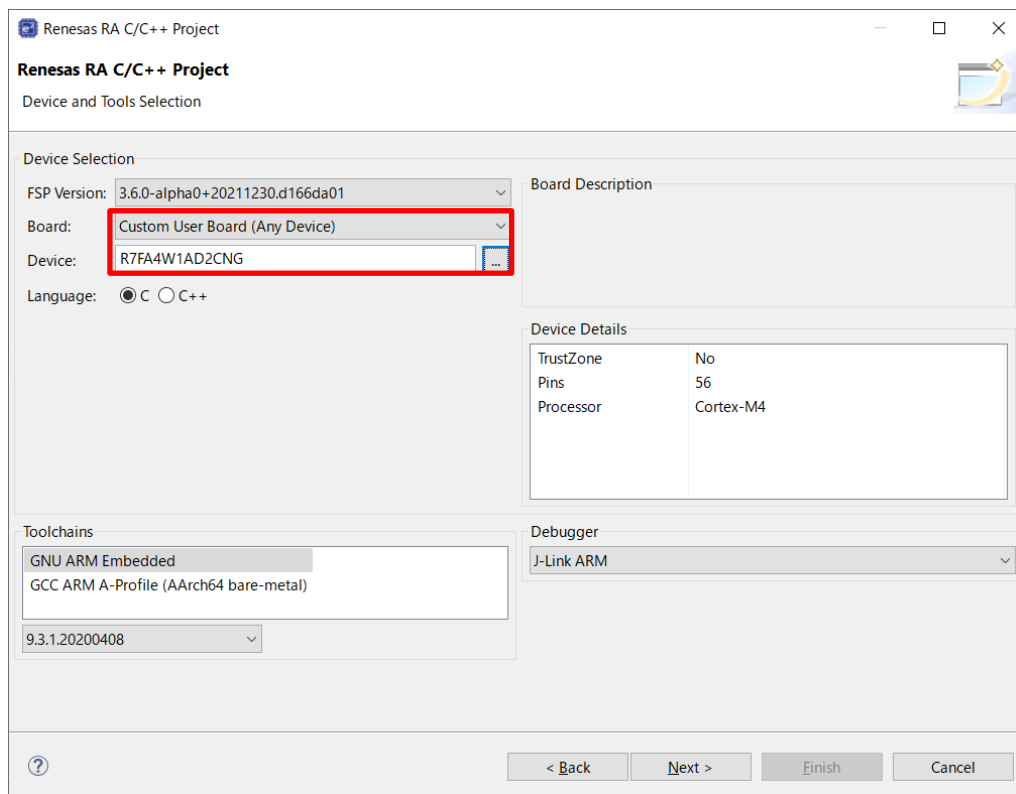


図 3-3 プロジェクト設定(Board と Device)

4. ベアメタル環境で MESH アプリケーションを作成する場合、No RTOS を選択します。FreeRTOS 環境で作成する場合は FreeRTOS を選択します。

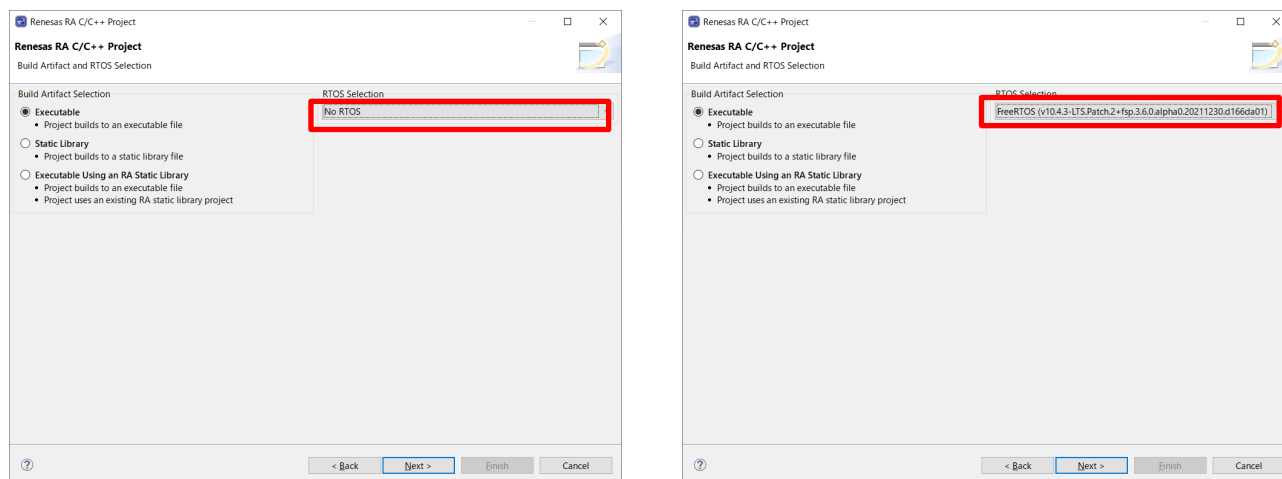


図 3-4 プロジェクト設定

5. Next ボタンをクリックします。

6. ベアメタル環境で MESH アプリケーションを作成する場合、BareMetal – Minimal を選択します。
FreeRTOS 環境で作成する場合、FreeRTOS – Minimal – Static Allocation を選択します。

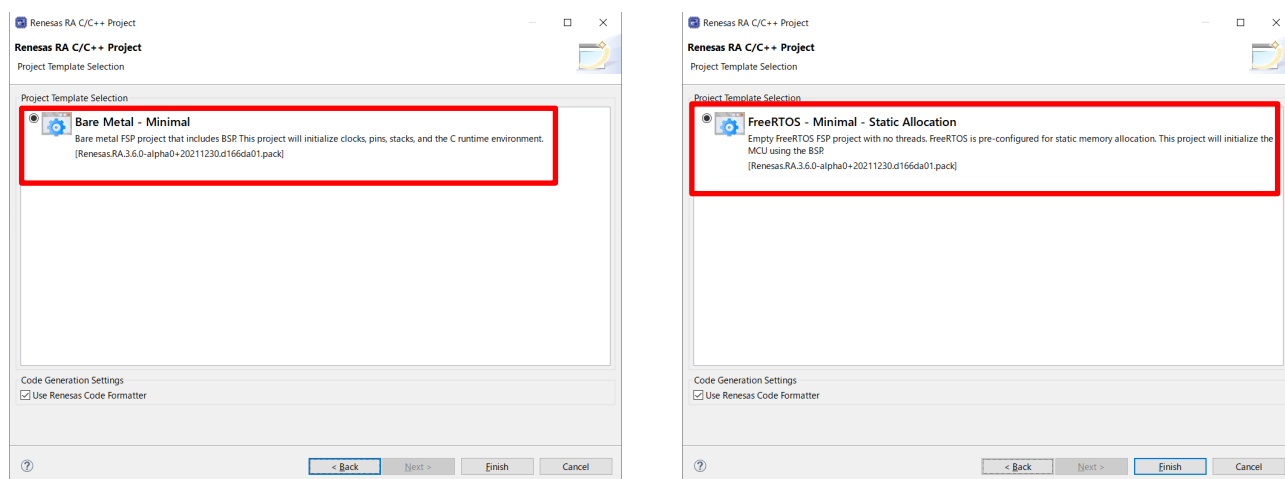


図 3-5 プロジェクト設定(テンプレート選択)

7. Finish ボタンをクリックしてしばらくするとプロジェクトが作成されます。

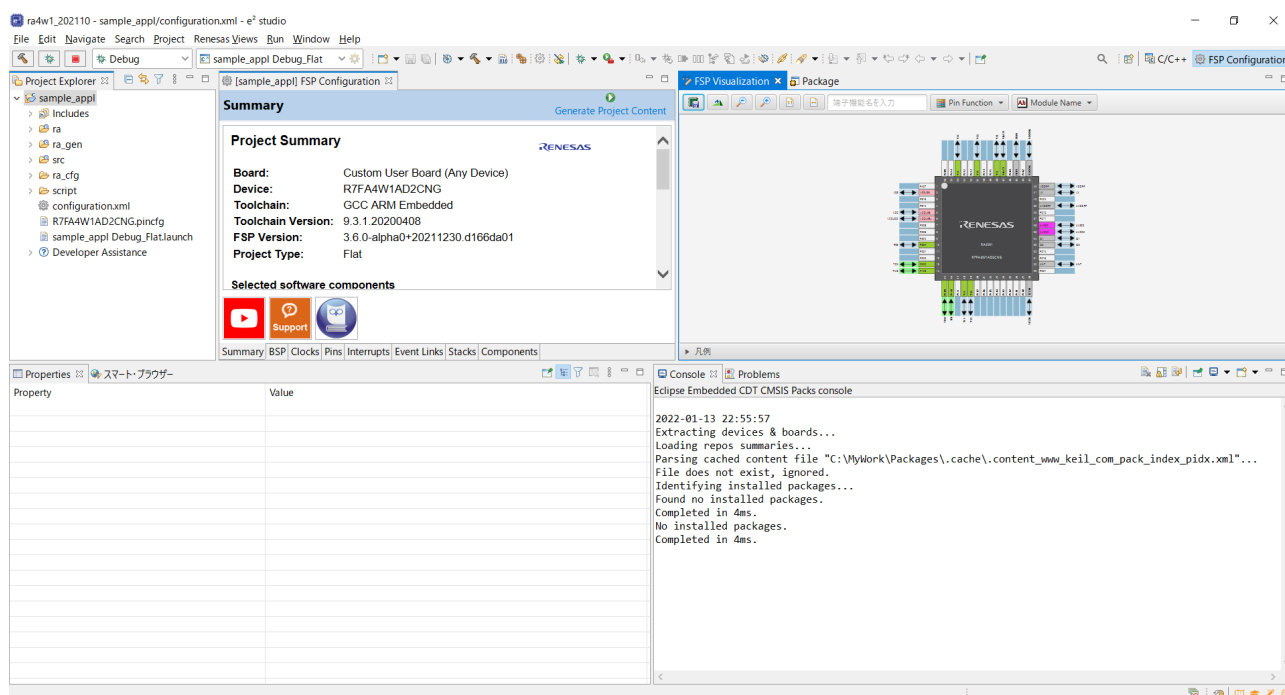


図 3-6 プロジェクト概観

3.2 ヒープとスタックの設定

Mesh スタックの動作に必要なメモリを確保するため、FSP Configuration の[BSP]タブの[Properties]で以下のようにヒープとスタックを設定します。

- [RA Common]→[Main stack size (bytes)] : 0x1400
- [RA Common]→[Heap size (bytes)] : 0x1000

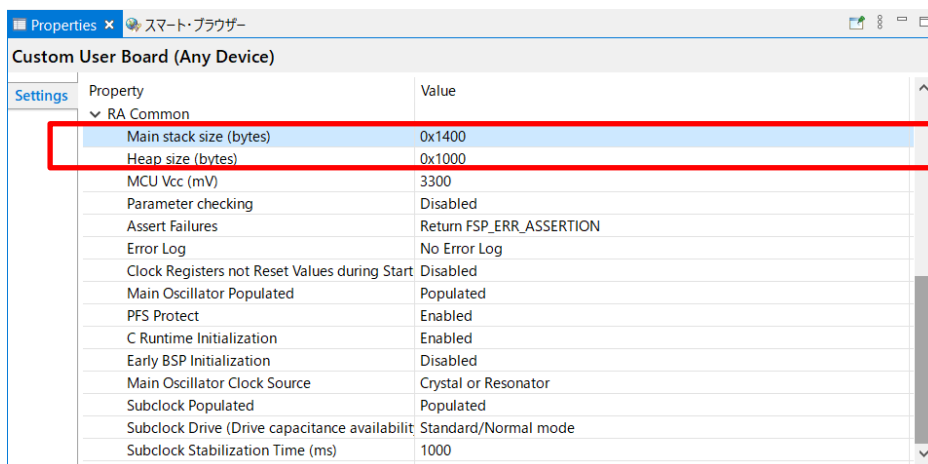


図 3-7 BSP 設定

BSP の表 3-1 に示す設定マクロは、上記設定によって変更されます。

注: Mesh スタックを使用する場合、本変更を必ず行ってください。

表 3-1 BSP の設定とマクロ

設定とマクロ	デフォルト値	Mesh 向け設定値
RA Common > Main stack size (bytes) (BSP_CFG_STACK_MAIN_BYTES)	0x400	0x1400
RA Common > Heap size (bytes) (BSP_CFG_HEAP_BYTES)	0	0x1000

3.3 クロックの設定

FSP Configuration の[Clocks]タブで、クロックの選択と周波数を設定します。Mesh スタックを使用する場合、下記の通り設定してください。

- システムクロック (ICLK): 8MHz 以上
- 周辺モジュールクロック A(PCLKA): 8MHz 以上

Bluetooth LE スタックは、ICLK と PCLKA の周波数が 32MHz の場合に最適化されています。このため ICLK と PCLKA の周波数が 32MHz となるように設定することを推奨します。

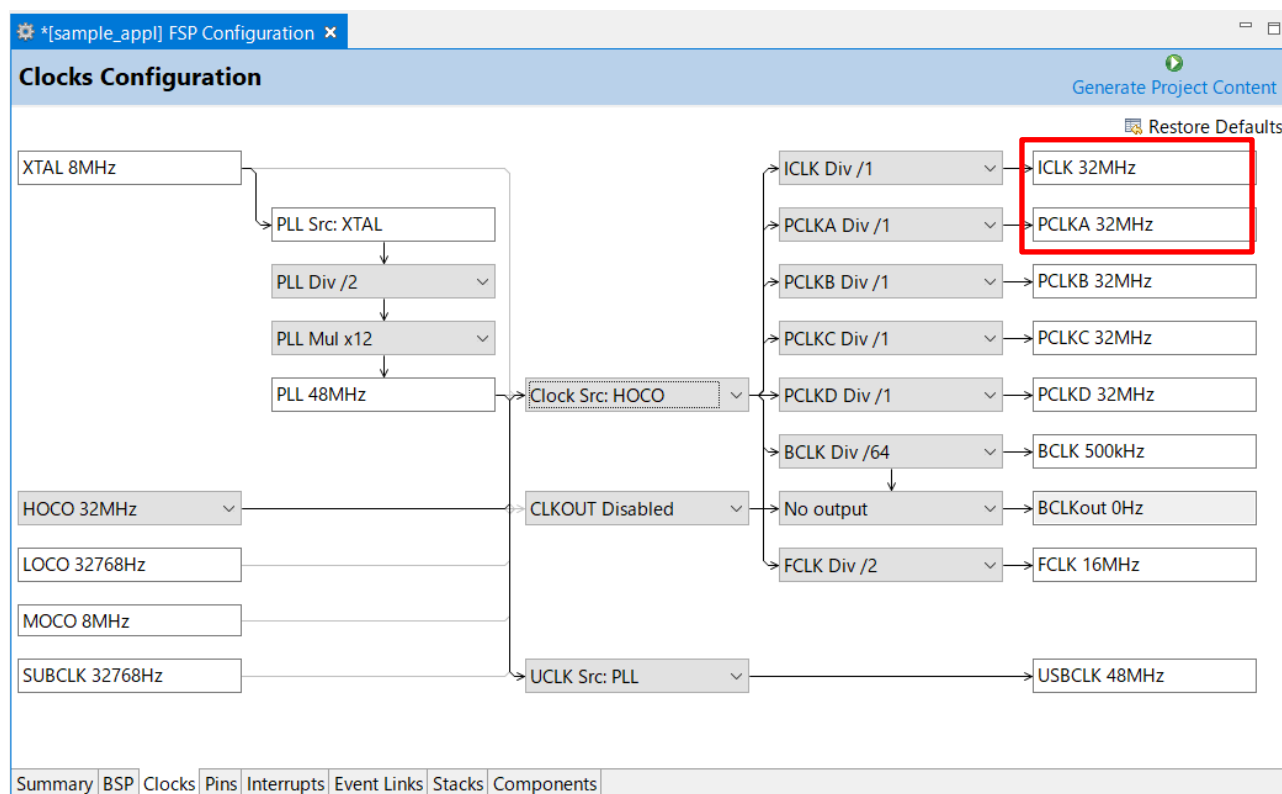


図 3-8 クロックの設定

3.4 Mesh スタックの追加と設定

Mesh アプリケーションに Mesh スタックを追加・設定する方法を説明します。プロジェクトの configuration.xml をクリックして FSP Configuration の[Stacks]タブで Mesh スタックを追加・設定してください。Mesh スタックを追加する手順はベアメタル環境と FreeRTOS 環境で異なります。3.4.1 項でベアメタル環境の手順を説明します。3.4.2 項で FreeRTOS 環境の手順を説明します。Mesh スタックの設定はベアメタル環境と FreeRTOS 環境で共通です。3.4.3 項で設定の詳細を説明します。

3.4.1 ベアメタル環境での Mesh スタック追加

1. New Stack をクリックして Networking→BLE Mesh Bearer Platform (rm_ble_mesh_bearer_platform) を HAL/Common に追加します。このドライバはいくつかの周辺ドライバを含みます。これら周辺の設定は 3.5 節で説明します。

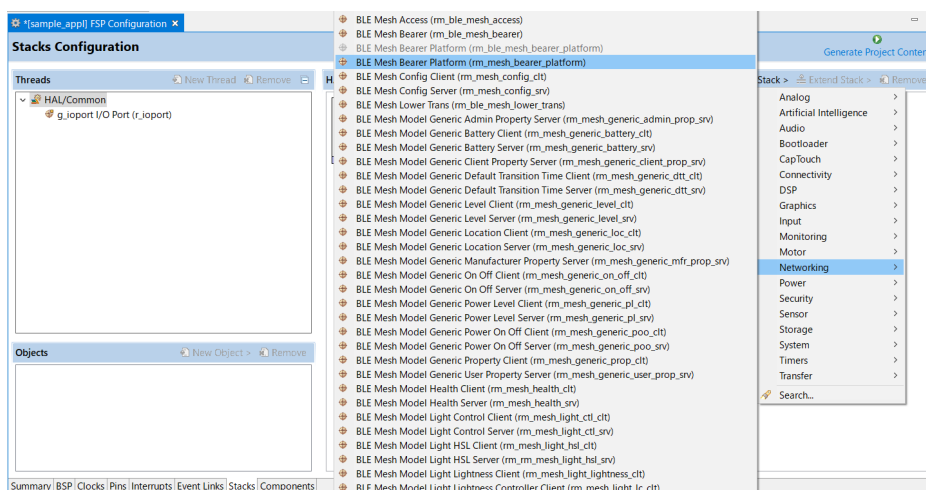


図 3-9 Bluetooth ベアラーの追加

2. Add BLE Mesh OS Module ボックスをクリックして New→BLE Mesh OS on Baremetal (rm_mesh_os_baremetal)を選択します。

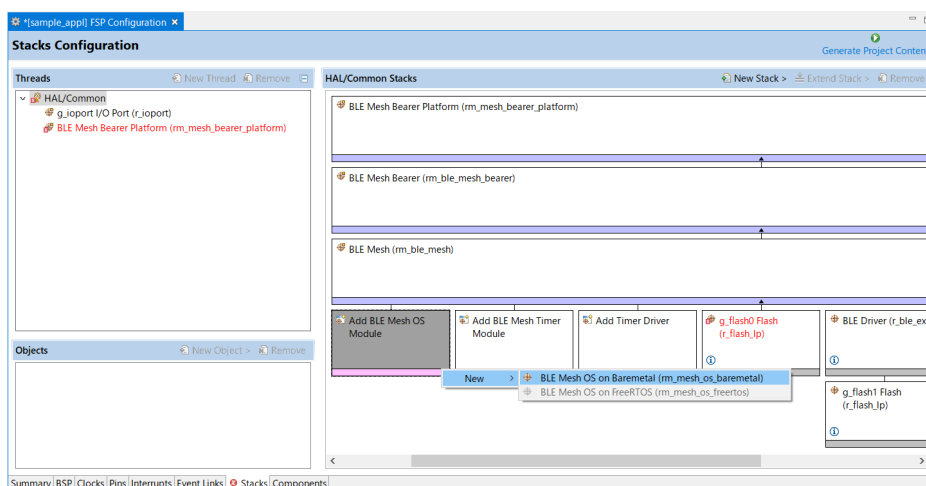


図 3-10 OS の追加

3. Add BLE Mesh Timer Module ボックスをクリックして New→BLE Mesh Timer on Baremetal (rm_mesh_timer_baremetal)を選択します。

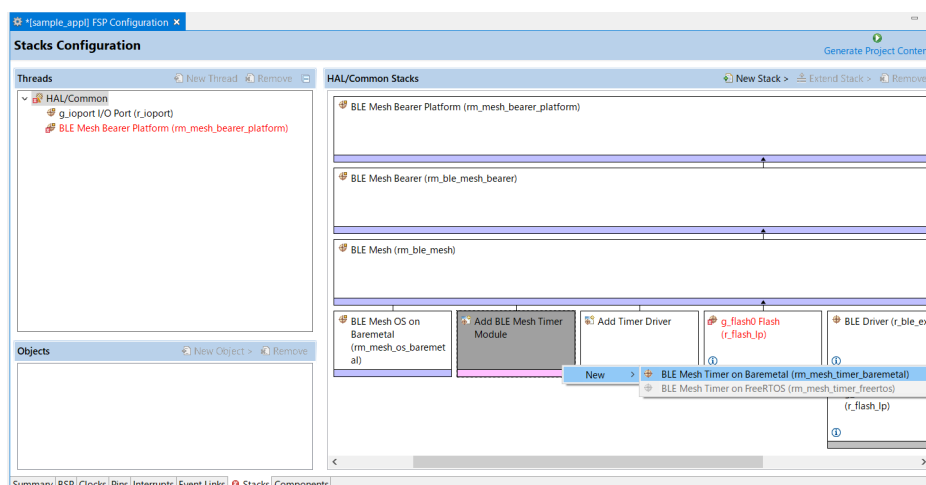


図 3-11 Timer の追加

4. Add BLE Network Driver ボックスをクリックして New→BLE Driver (r_ble_extended)を追加します。

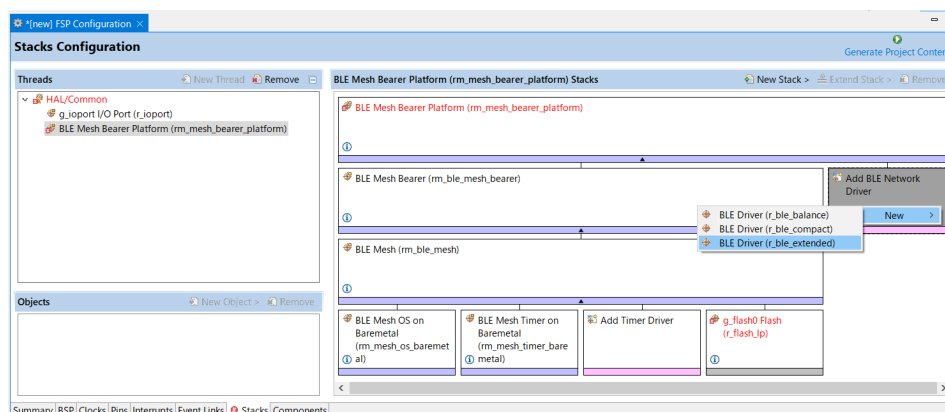


図 3-12 Bluetooth LE スタックの追加

5. New Stack をクリックして作成する Mesh アプリケーションに必要なモデルを HAL/Common に追加します。例えば Generic On OFF サーバーモデルを追加する場合は、Networking→BLE Mesh Model Generic On Off Server を選択します。

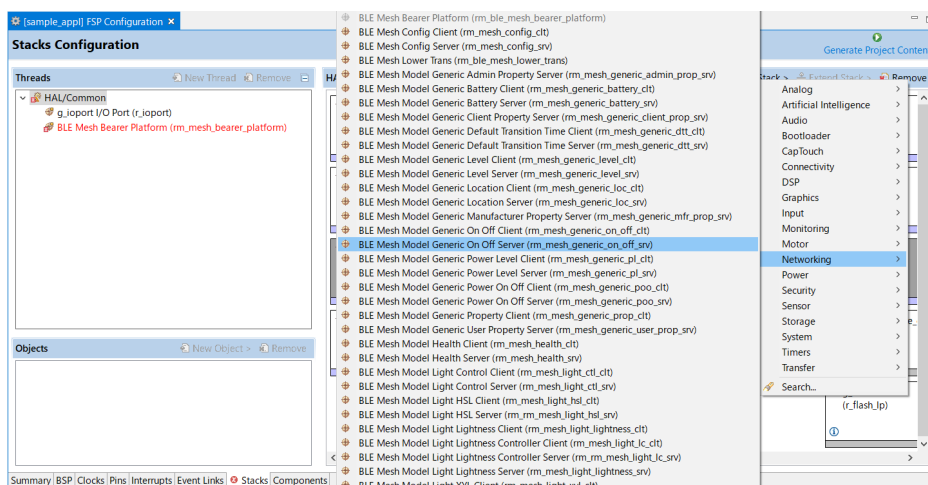
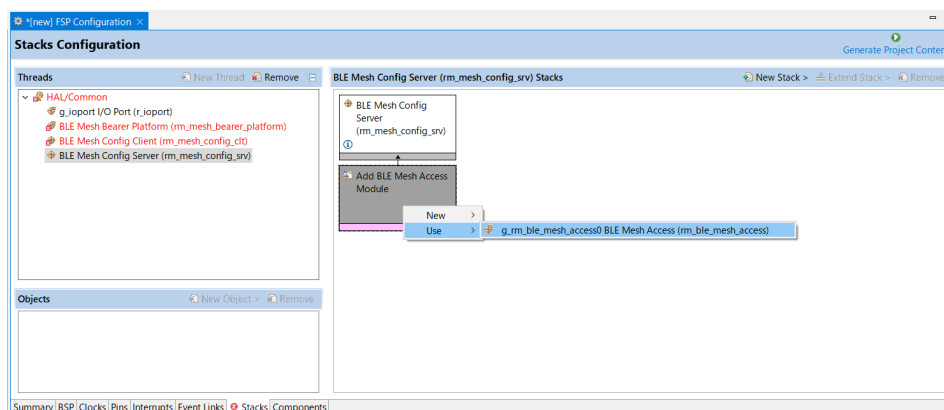


図 3-13 Mesh モデルの追加

注: 同じエレメントに 2 つ目以降のモデルを追加する場合は Add BLE Mesh Access Module ボックスをクリックして Use→g_rm_ble_access0 BLE Mesh Access (rm_ble_mesh_access)を選択してください。



3.4.2 FreeRTOS 環境での Mesh スタック追加

1. Threads エリアで New Thread をクリックして New Thread を追加します。例では BLE_CORE_TASK という名前を付けています。

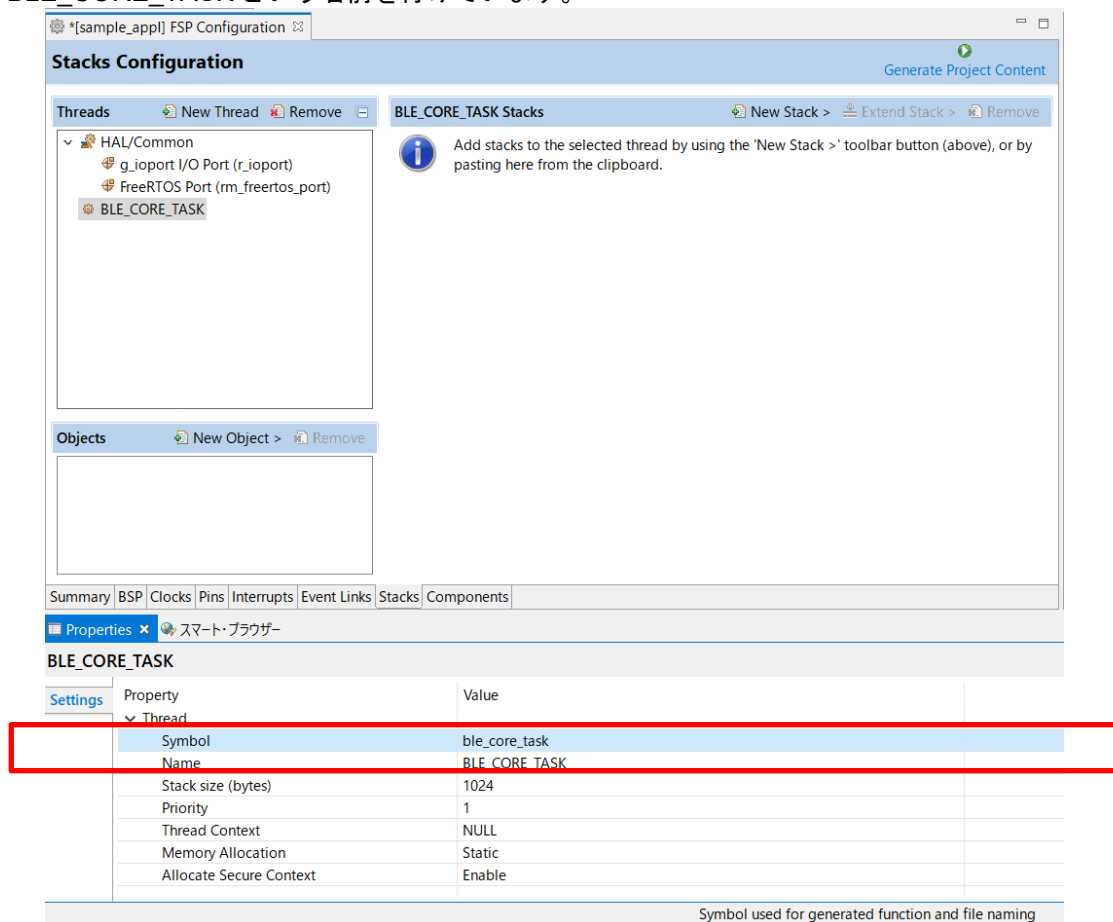


図 3-14 BLE_CORE_TASK の追加

2. Stack size を 0x3000 バイトに変更します。Priority を 2 に変更します。

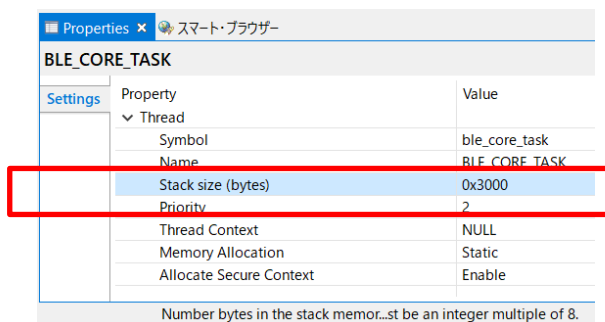


図 3-15 BLE_CORE_TASK の Stack size と Priority

3. BLE_CORE_TASK の Properties タブで以下のように FreeRTOS 設定を変更します。

表 3-2 FreeRTOS の設定とマクロ

設定とマクロ	変更値	初期値
Common > General > Minimal Stack Size (configMINIMAL_STACK_SIZE)	1024	128
Common > General > Use Mutexes (configUSE_MUTEXES)	Enabled	Disabled
Common > General > Use Recursive Mutexes (configUSE_RECURSIVE_MUTEXES)	Enabled	Disabled
Common > General > Enable Backward Compatibility (configENABLE_BACKWARD_COMPATIBILITY)	Enabled	Disabled
Common > Memory Allocation > Support Dynamic Allocation (configSUPPORT_DYNAMIC_ALLOCATION)	Enabled	Disabled
Common > Memory Allocation > Total Heap Size (configTOTAL_HEAP_SIZE)	11264	0
Common > Timers > Timer Queue Length (configTIMER_QUEUE_LENGTH)	32	10
Common > Optional Functions > <i>xTimerPendFunctionCall()</i> Function (INCLUDE_xTimerPendFunctionCall)	Enabled	Disabled

4. New Stack をクリックして Networking→BLE Mesh Bearer Platform (rm_ble_mesh_bearer_platform)を BLE_CORE_TASK に追加します。このドライバはいくつかの周辺ドライバを含みます。これら周辺の設定は 3.5 節で説明します。

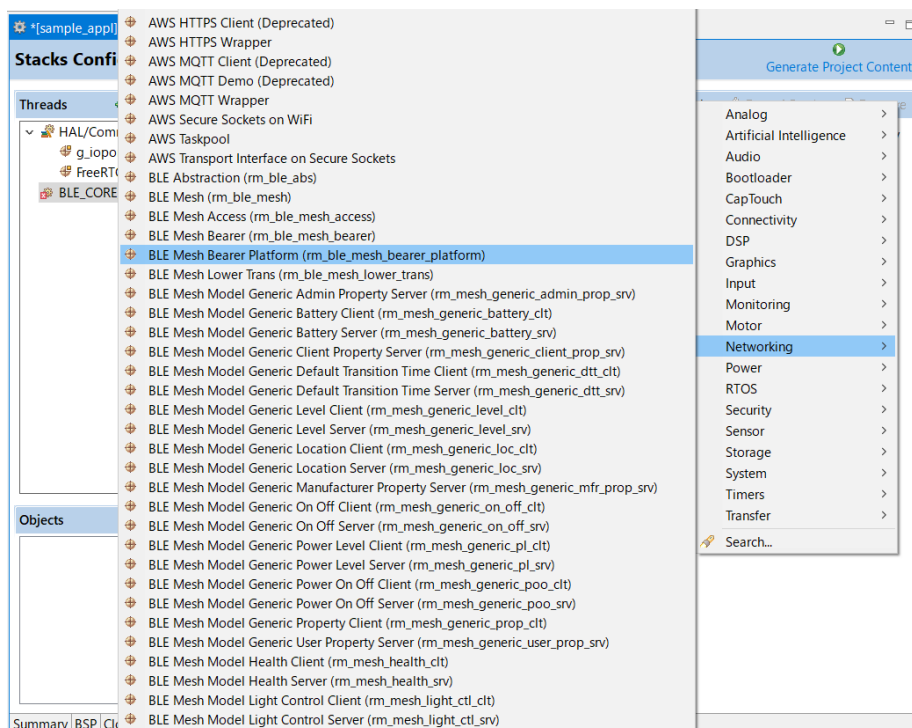


図 3-16 Bluetooth ベアラの追加

5. Add BLE Mesh OS Module ボックスをクリックして New→BLE Mesh OS on FreeRTOS (rm_mesh_os_freertos)を選択します。

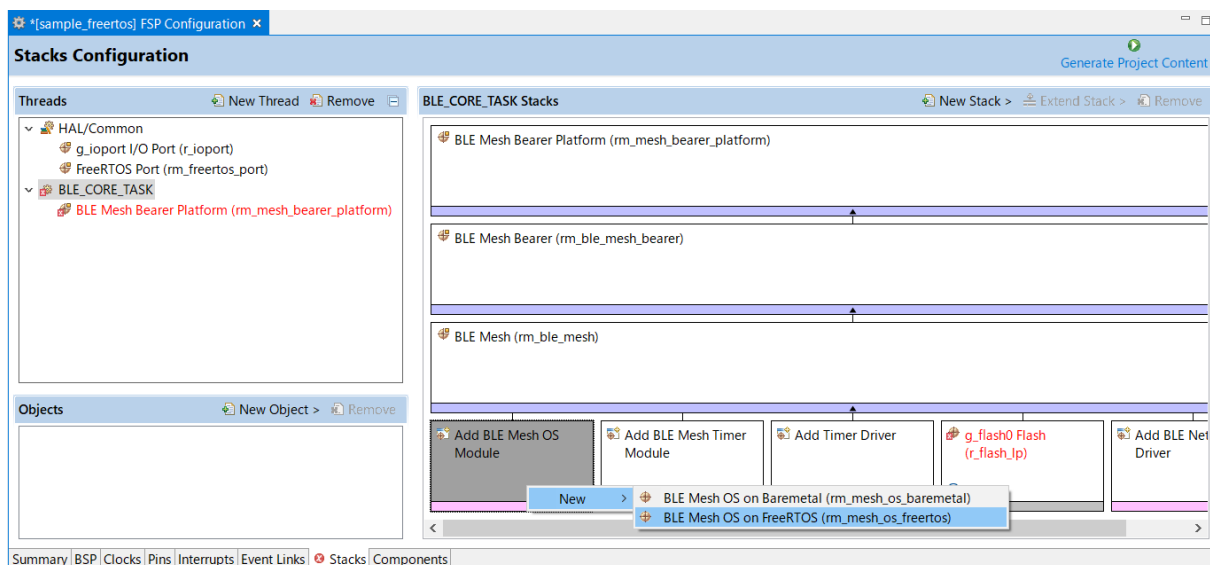


図 3-17 OS の追加

6. Add BLE Mesh Timer Module ボックスをクリックして New→BLE Mesh Timer on FreeRTOS (rm_mesh_timer_freertos)を選択します。

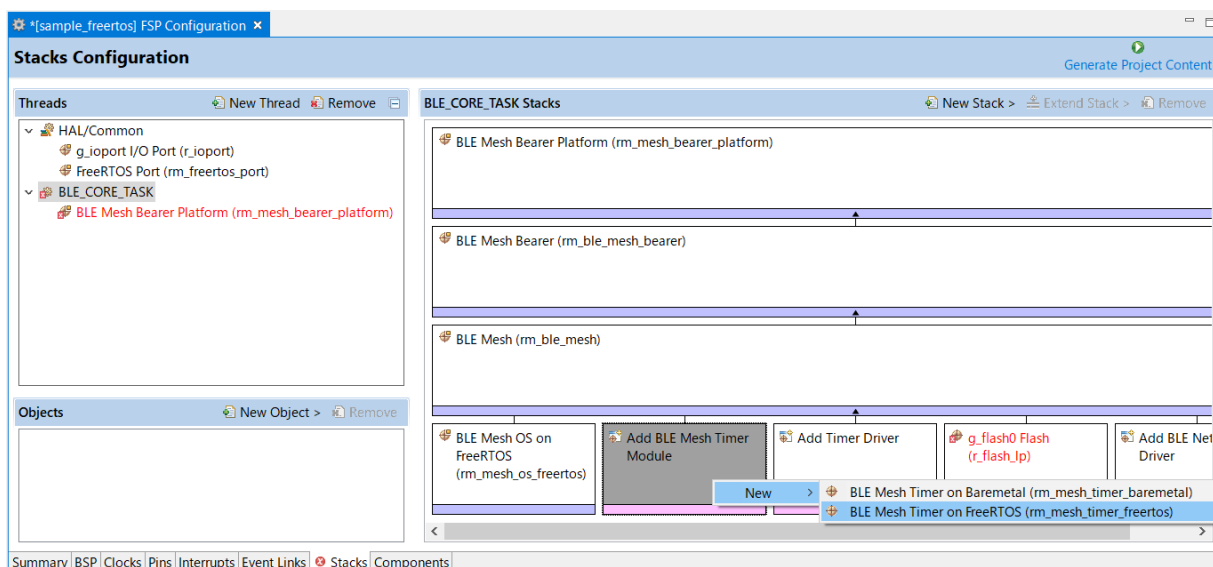


図 3-18 Timer の追加

7. Add BLE Network Driver ボックスをクリックして New→BLE Driver (r_ble_extended_freertos)を追加します。

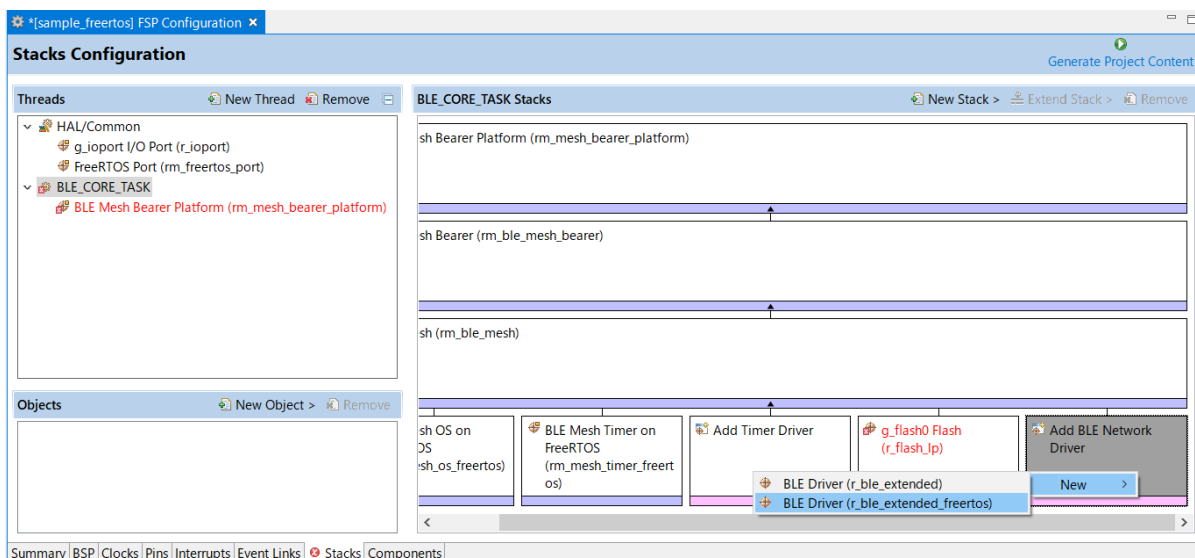


図 3-19 Bluetooth LE スタックの追加

8. New Stack をクリックして作成する Mesh アプリケーションに必要なモデルを BLE_CORE_TASK に追加します。例えば Generic On Off サーバーモデルを追加する場合は、Networking→BLE Mesh Model Generic On Off Server を選択します。

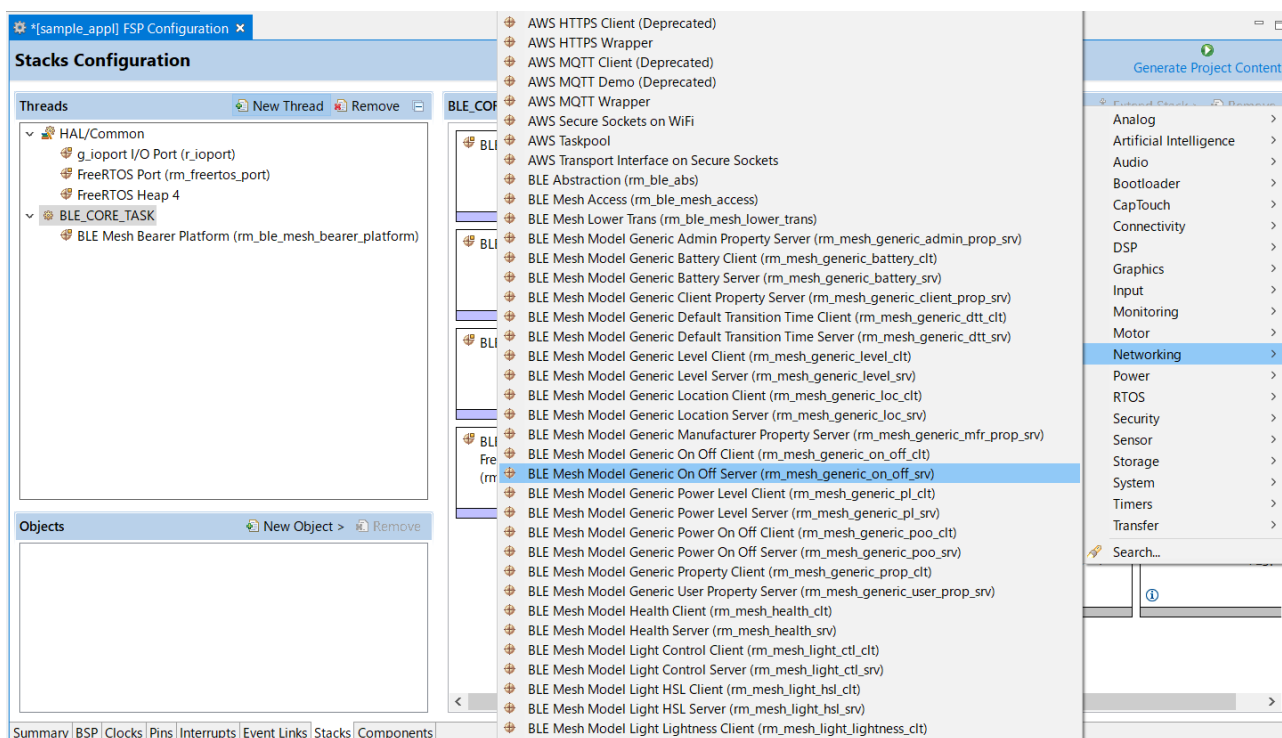


図 3-20 Mesh モデルの追加

注: 同じエレメントに 2 つ目以降のモデルを追加する場合は Add BLE Mesh Access Module ボックスをクリックして Use→g_rm_ble_access0 BLE Mesh Access (rm_ble_mesh_access)を選択してください。

9. HAL/Common に RTOS→FreeRTOS Heap4 を追加します。

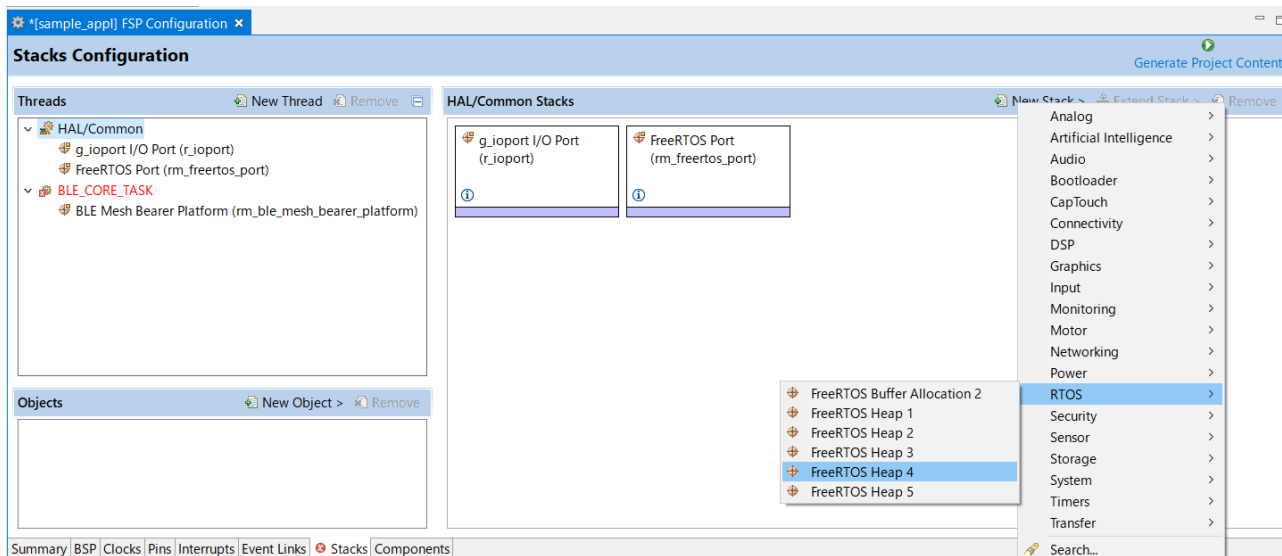


図 3-21 Heap4 の追加

3.4.3 Mesh スタック設定

本節では Mesh スタックに必要な設定を記載します。

[Stacks] タブで [BLE Mesh Bearer Platform (rm_mesh_bearer_platform)] を選択して表 3-3 のようにプロパティを変更してください。各プロパティ値の詳細については「Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)」を参照してください。

表 3-3 Mesh スタックの設定とマクロ

設定とマクロ	デフォルト値	Mesh 向け設定値
Maximum number of connections (MESH_BEARER_PLATFORM_CFG_RF_CONNECTION_MAXIMUM)	7	1
Maximum advertising data length (MESH_BEARER_PLATFORM_CFG_RF_ADVERTISING_DATA_MAXIMUM)	1650	31
Maximum advertising set number (MESH_BEARER_PLATFORM_CFG_RF_ADVERTISING_SET_MAXIMUM)	4	1
Maximum periodic sync set number (MESH_BEARER_PLATFORM_CFG_RF_SYNC_SET_MAXIMUM)	2	1

[Maximum number of connections] を"1"に、[Maximum advertising data length] を"31"に、[Maximum advertising set number] を"1"に、[Maximum periodic sync set number] を"1"に設定します。

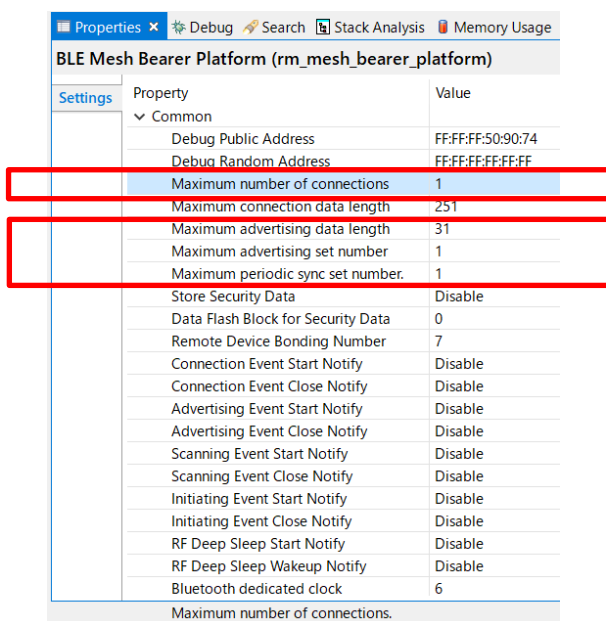


図 3-22 Mesh スタック設定(1)

注: BLE Driver ボックスのプロパティにも同じ値を設定してください。

[Stacks] タブで [BLE Mesh (rm_ble_mesh)] を選択して表 3-4 のようにプロパティを変更してください。
各プロパティ値の詳細については「Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)」を参照してください。

表 3-4 Mesh スタックの設定

設定	デフォルト値	Mesh 向け設定値
Storage→Block Number Mesh 情報を格納するデータフラッシュのブロック数 最小値: 1 最大値: 8	1	6
Memory Pool→Memory Pool Size	0x4000	0x3000

[Block number] を "6" に、[Memory pool size] を "0x3000" に設定します。

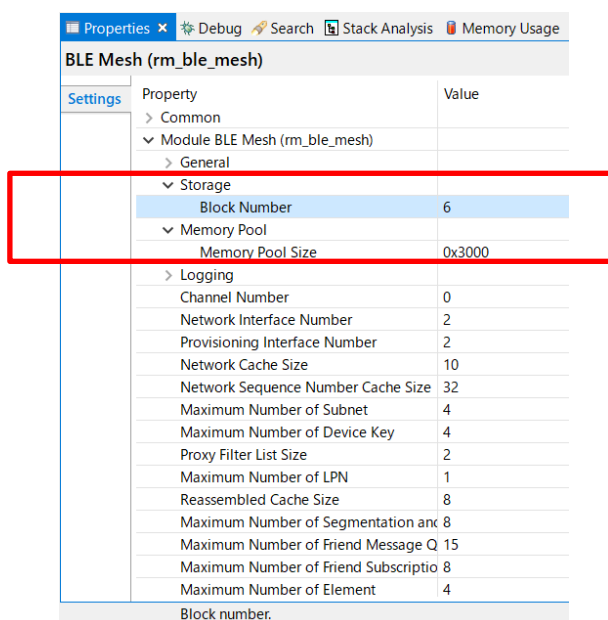


図 3-23 Mesh スタック設定(2)

3.4.4 その他の Mesh スタック設定

Mesh スタックには、Mesh ネットワーク規模やノードの要件に応じて設定可能なパラメータがあります。これらのパラメータは FSP Configuration で設定でき、コードの生成時に common_data.c に反映されます。

表 3-5 BLE Mesh (rm_ble_mesh)の設定と変数名

設定と変数名	詳細
Bearer→Network Interface Number (network_interfaces_num) *デフォルト値: 2	Mesh ネットワークに使用するベアラ数 最小値: 1 最大値: (1 + MESH_BEARER_PLATFORM_CFG_RF_CONNECTION_MAXIMUM) 最初のベアラは ADV Bearer、残るベアラは同時に接続を確立できる GATT Bearer となる。 本設定を 1 とした場合、ADV Bearer のみ使用できる。
Bearer→Provisioning Interface Number (provisioning_interfaces_num) *デフォルト値: 2	プロビジョニングに使用するベアラ数 最小値: 1 最大値: 2 本設定を 1 とした場合、PB-ADV Bearer のみ使用できる。本設定を 2 とした場合、PB-ADV Bearer と 1 つの PB-GATT Bearer を使用できる。
Provisioning→Unprovisioned Device Beacon Timeout in Milliseconds (unprov_device_beacon_timeout) *デフォルト値: 200	Unprovisioned Device Beacon の送信間隔[msec] 最小値: 20 PB-ADV のみ使用する場合、Unprovisioned Device Beacon を本設定間隔で送信する。PB-GATT のみ使用する場合、Connectable Advertising PDU を本設定間隔で送信する。PB-ADV と PB-GATT の両方を使用する場合、Unprovisioned Device Beacon と Connectable Advertising PDU を本設定間隔で交互に送信する。
Network→Network Cache Size (network_cache_size) *デフォルト値: 10	Network Message Cache が保持できる最大ノード数 最小値: 2 最大ノード数のキャッシュ情報が保持された状態でさらに新しいノードからのメッセージを受信した場合、最も古く登録されたノードのキャッシュ情報は消去される。
Network→Network Sequence Number Cache Size (network_sequence_num_cache_size) *デフォルト値: 32	Network Message Cache が保持できる各ノードの SEQ 番号数 最小値: 32
Network→Maximum Number of Subnet (maximum_subnets) *デフォルト値: 4	Network Key、NID などのサブネット情報の最大数 最小値: 1
Network→Maximum Number of Device Key (maximum_device_keys) *デフォルト値: 4	Device Key の最大数 最小値: 1 Configuration Client Model を使用しない場合、本設定は 1 でよい。
Network→Proxy Filter List Size (proxy_filter_list_size) *デフォルト値: 2	各 Proxy Filter List に登録可能なアドレスの最大数 最小値: 1

Network→Network Sequence Number Block Size (net_sequence_number_block_size) *デフォルト値: 2048	Data Flash に書き込む SEQ 番号の間隔 最小値: 1 本設定の間隔で Data Flash に SEQ 番号を退避する。 MCU がリセットされた場合、退避した次の間隔から再開する。 例) 本設定が 2048 ならば、SEQ 番号が 2048、4096 と 2048 の倍数に到達する毎に Data Flash へ書き込まれる。もし SEQ 番号が 3000 の時点で MCU がリセットされた場合、SEQ 番号は 4096 から再開する。 本設定間隔が短いほど、Data Flash への書き込み頻度が高くなる。本設定間隔が長くなるほど、MCU リセット後にスキップする SEQ 番号の幅が大きくなる。
Network→Network Transmit Count for Network Packets (net_tx_count) *デフォルト値: 1	Network Transmit Count ステートのデフォルト値 最小値: 0 最大値: 7
Network→Network Interval Steps for Network Packets (net_tx_interval_steps) *デフォルト値: 4	Network Transmit Interval Steps ステートのデフォルト値 最小値: 0 最大値: 31
Network→Network Transmit Count for Relayed Packets (net_relay_tx_count) *デフォルト値: 0	Relay Retransmit Count ステートのデフォルト値 最小値: 0 最大値: 7
Network→Network Interval Steps for Relayed Packets (net_relay_tx_interval_steps) *デフォルト値: 9	Relay Retransmit Interval Steps ステートのデフォルト値 最小値: 0 最大値: 31
Network→Proxy ADV Network ID Timeout for Each Subnet in Milliseconds (proxy_subnet_netid_adv_timeout) *デフォルト値: 100	Proxy Advertisement with Network ID の送信間隔[msec] 最小値: 20
Network→Proxy ADV Node Identity Timeout for Each Subnet in Milliseconds (proxy_subnet_nodeid_adv_timeout) *デフォルト値: 300	Proxy Advertisement with Node Identity の送信間隔[msec] 最小値: 20
Network→Proxy ADV Node Identity Overall Time Period in Milliseconds (proxy_nodeid_adv_timeout) *デフォルト値: 60	Proxy Advertisement with Node Identity の送信期間[sec] 最小値: 1
Network→Maximum Number of Queued Messages for Transmission (net_tx_queue_size) *デフォルト値: 64	Network PDU の送信キューサイズ 最小値: 2
Transport→Maximum Number of LPN (maximum_lpn) *デフォルト値: 1	Friend Node としてフレンドシップを確立する対向 Low Power Node の最大数 最小値: 1
Transport→Replay Protection Cache Size (replay_cache_size) *デフォルト値: 10	Replay Protection Cache サイズ 最小値: 2
Transport→Reassembled Cache Size (reassembled_cache_size) *デフォルト値: 8	Segmentation and Reassembly(SAR)処理の受信メッセージキャッシュサイズ 最小値: 2
Transport→Friend Poll Retry Count (frnd_poll_retry_count) *デフォルト値: 10	Low Power Node が Friend Update メッセージを受信できなかった場合の Friend Poll メッセージ再送回数 最小値: 1

Transport→Maximum Number of Segmentation and Reassembly Context (maximum_ltrn_sar_context) *デフォルト値: 8	Segmented メッセージの送受信に使用する Segmentation and Reassembly(SAR)処理のコンテキスト数 最小値: 2
Transport→Lower Transport Segment Transmission Timeout in Milliseconds (ltrn_rtx_timeout) *デフォルト値: 300	Segmented メッセージの再送間隔[msec] 最小値: 200
Transport→Lower Transport Segment Re-Transmission Count (ltrn_rtx_count) *デフォルト値: 2	Segmented メッセージの再送回数 最小値: 2
Transport→Lower Transport Acknowledgement Timeout in Milliseconds (ltrn_ack_timeout) *デフォルト値: 200	Segmented Acknowledgement メッセージの送信間隔 [msec] 最小値: 200
Transport→Lower Transport Incomplete Timeout in Milliseconds (ltrn_incomplete_timeout) *デフォルト値: 20	Segmented メッセージの受信キャンセルタイムアウト 時間 [sec] 最小値: 10
Transport→Friendship Receive Window (frnd_receive_window) *デフォルト値: 100	Low Power Node の受信ウィンドウサイズ [msec] 最小値: 100 最大値: 255
Transport→Maximum Number of Friend Message Queue (maximum_friend_message_queue) *デフォルト値: 15	各 Low Power Node に対するメッセージキュー数 最小値: 2
Transport→Maximum Number of Friend Subscription List (maximum_friend_subscription_list) *デフォルト値: 8	各 Low Power Node に対するフレンドサブスクリプションリストの最大数 最小値: 1
Transport→Friend Clear Confirmation Timeout in Milliseconds (lpn_clear_retry_timeout_initial) *デフォルト値: 1000	Low Power Node が Friend Clear Confirmation メッセージを受信できなかった場合の Friend Clear メッセージの再送間隔[msec] 最小値: 1000
Transport→Friend Clear Retry Count (lpn_clear_retry_count) *デフォルト値: 5	Low Power Node が Friend Clear Confirmation メッセージを受信できなかった場合の Friend Clear メッセージ再送回数 最小値: 1
Transport→Friendship Retry Timeout in Milliseconds (trn_frndreq_retry_timeout) *デフォルト値: 1200	Low Power Node による Friend Request メッセージの送信期間 [msec] 最小値: 1100
Access→Maximum Number of Element (maximum_access_element_num) *デフォルト値: 4	エレメントの最大数 最小値: 1
Access→Maximum Number of Model (maximum_access_model_num) *デフォルト値: 60	モデルの最大数 最小値: 1
Access→Maximum Number of Application (maximum_application) *デフォルト値: 8	Application Key の最大数 最小値: 1
Access→Maximum Number of Virtual Address (maximum_virtual_address) *デフォルト値: 8	バーチャルアドレスの最大数 最小値: 1
Access→Maximum Number of Non-Virtual Address (maximum_non_virtual_address) *デフォルト値: 8	非バーチャルアドレス(ユニキャストアドレス、グループアドレス)の最大数 最小値: 1

Access→Maximum Number of Transition Timers (max_num_transition_timers) *デフォルト値: 5	モデルの State Transition タイマの数 最小値: 1
Access→Maximum Number of Periodic Step Timers (max_num_periodic_step_timers) *デフォルト値: 5	モデルの Periodic Publication タイマの数 最小値: 1
Foundation→Config Server Secure Network Beacon Interval (config_server_snb_timeout) *デフォルト値: 10	Secure Network Beacon の送信間隔 [sec] 最小値: 10 最大値: 600
Foundation→Maximum Number of Health Server Instance (maximum_health_server_num) *デフォルト値: 2	Health Server Model の最大数 最小値: 1
Model→Maximum Number of Light Lightness Controller Server Instance (maximum_light_lc_server_num) *デフォルト値: 1	Light Lightness Controller Server Model の最大数 最小値: 1
ID→Company ID (default_company_id) *デフォルト値: 0x0036	Bluetooth SIG に登録されたカンパニーID 最小値: 0x0000 最大値: 0xFFFF
ID→Product ID (default_product_id) *デフォルト値: 0x0001	ベンダー独自の製品 ID 最小値: 0x0000 最大値: 0xFFFF
ID→Vendor ID (default_vendor_id) *デフォルト値: 0x0100	ベンダー独自の製品バージョン ID 最小値: 0x0000 最大値: 0xFFFF
logging→Packet Bitfield (p_logging_cfg->packet_bitfield) *デフォルト値: 0	パケット関連のログ取得
logging→Module Info Bitfield (p_logging_cfg->module_info_bitfield) *デフォルト値: 0	モジュール関連のログ取得
logging→Generic Log Bitfield (p_logging_cfg->generic_log_bitfield) *デフォルト値: 0	汎用ログ取得
logging→function (p_logging_cfg->p_logging_func) *デフォルト値: logging_function	ログ取得時のコールバック

表 3-6 BLE Mesh Provision (rm_ble_mesh_provision)の設定と変数名

設定と変数名	詳細
Provision Capabilities→Number of Elements (num_elements) *デフォルト値: 1	詳細は Mesh Profile 仕様を参照してください。
Provision Capabilities→Supported Algorithms (supported_algorithms) *デフォルト値: 1	詳細は Mesh Profile 仕様を参照してください。
Provision Capabilities→Public Key Type (supported_pubkey) *デフォルト値: 1	詳細は Mesh Profile 仕様を参照してください。
Provision Capabilities→Static OOB Type (supported_soob) *デフォルト値: 1	詳細は Mesh Profile 仕様を参照してください。
Provision Capabilities→Output OOB Action (output_oob.action) *デフォルト値: 0x1F	詳細は Mesh Profile 仕様を参照してください。

Provision Capabilities→Output OOB Size (output_oob.size) *デフォルト値: 0x08	詳細は Mesh Profile 仕様を参照してください。
Provision Capabilities→Input OOB Action (input_oob.action) *デフォルト値: 0x0F	詳細は Mesh Profile 仕様を参照してください。
Provision Capabilities→Input OOB Size (input_oob.size) *デフォルト値: 0x04	詳細は Mesh Profile 仕様を参照してください。
Provision Callback (p_callback) *デフォルト値: NULL	Provisioning 処理のコールバック

表 3-7 BLE Mesh Network (rm_ble_mesh_network)の設定と変数名

設定と変数名	詳細
Callback (p_callback) *デフォルト値: NULL	Network 処理のコールバック

表 3-8 BLE Mesh Upper Trans (rm_ble_mesh_upper_trans)の設定と変数名

設定と変数名	詳細
Callback (p_callback) *デフォルト値: NULL	Upper Transport 処理のコールバック

表 3-9 BLE Mesh Access (rm_ble_mesh_access)の設定と変数名

設定と変数名	詳細
Location Descriptor (p_element_descriptor->loc) *デフォルト値: 0	エレメントが設置されている場所
Element Number (element_number) *デフォルト値: 0	エレメントを識別するための番号

表 3-10 BLE Mesh Bearer Platform (rm_mesh_bearer_platform)の設定と変数名

設定と変数名	詳細
Device Address Type (device_address_type) *デフォルト値: 1	BD アドレスのタイプ 0: パブリックアドレス 1: ランダムアドレス
GATT Server Callback Number (gatt_server_callback_num) *デフォルト値: 15	登録するコールバックの数 最小値: 1 最大値: 15
GATT Client Callback Number (gatt_client_callback_num) *デフォルト値: 15	登録するコールバックの数 最小値: 1 最大値: 15
Vender Specific Callback (vender_specific_callback) *デフォルト値: NULL	Bearer Platform の open API による BD アドレスの設定完了のコールバック

3.5 関連モジュールの追加と設定

Mesh スタックは Mesh 通信を行うために以下の周辺機能を使用します。

表 3-11 関連する周辺機能

項目	用途
Bluetooth Low Energy Driver (r_ble_extended or r_ble_extended_freertos)	Bluetooth Low Energy 通信
General PWM Timer Driver (r_gpt)	Bluetooth Mesh スタック用タイマ(g_timer0) Bluetooth LE スタック用タイマ(g_timer1)
Low-Power Flash Driver (r_flash_lp)	ボンディング情報の保存など
Interrupt Controller Unit Driver (r_icu)	BLE(H/W)からの割り込み(g_external_irq0) スイッチ(H/W)からの割り込み(g_ble_sw_irq)
Serial Communication Interface Driver (r_sci_uart)	シリアル通信
Low Power Modes Driver (r_lpm)	MCU の低電力モード

本節では、前節で追加した Mesh スタックに関連する周辺機器(タイマー、割り込み)を構成する方法について説明します。本節で説明する手順は、ベアメタルおよび FreeRTOS 環境に共通です。

3.5.1 r_gpt (g_timer0)

1. Add Timer Driver ボックスをクリックして New→Timer, General PWM (r_gpt)を選択します。

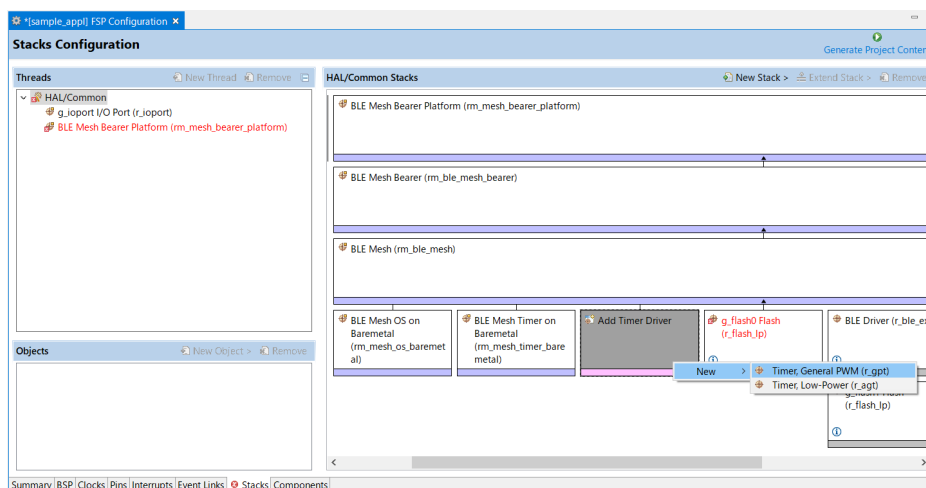


図 3-24 Bluetooth Mesh スタック用 GPT の追加

1. Properties タブで g_timer0 Timer, General PWM (r_gpt)の Overflow/Crest Interrupt Priority を Priority 3 に設定します。

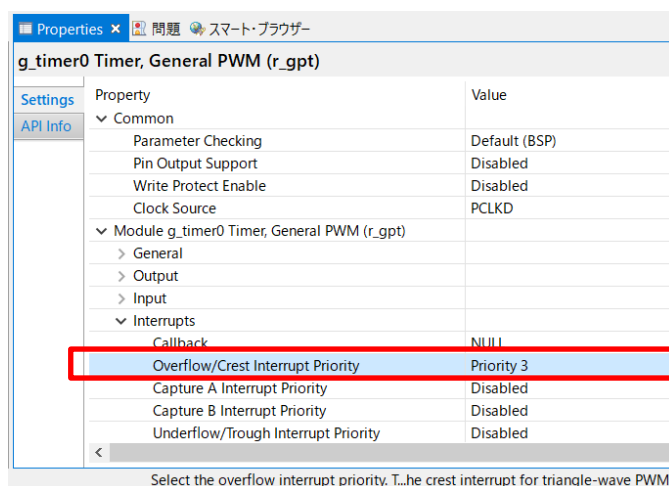


図 3-25 Bluetooth Mesh スタック用 GPT の設定

3.5.2 r_flash_lp

以下のように設定します。

1. Properties タブで g_flash0 Flash (r_flash_lp) の Data Flash Background Operation を Disabled に設定します。

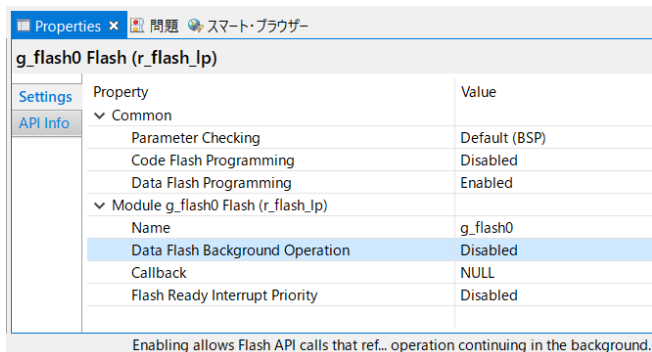


図 3-26 Flash の設定

3.5.3 r_icu (g_external_irq0)

1. g_external_irq0 External IRQ (r_icu)の Pin Interrupt Priority を以下のように設定します。

- ・ ペアメタル環境

Priority に Priority 0

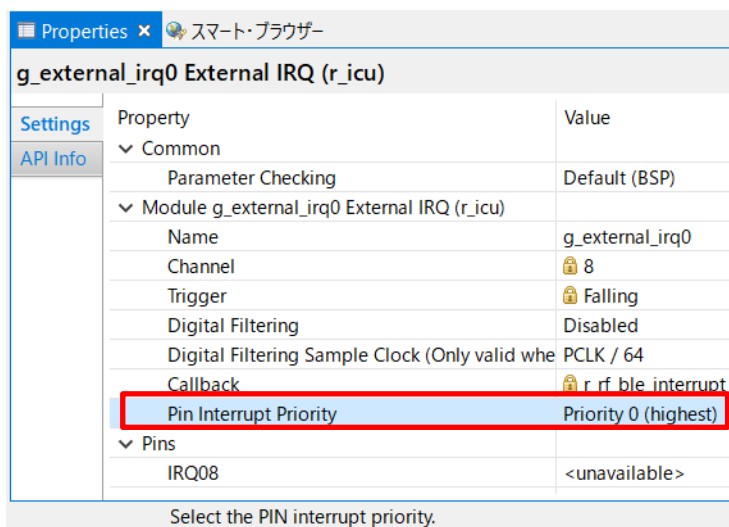


図 3-27 ICU の設定(ペアメタル環境)

- ・ FreeRTOS 環境

Priority に Priority 1 (FreeRTOS 環境でいちばん高い優先度)

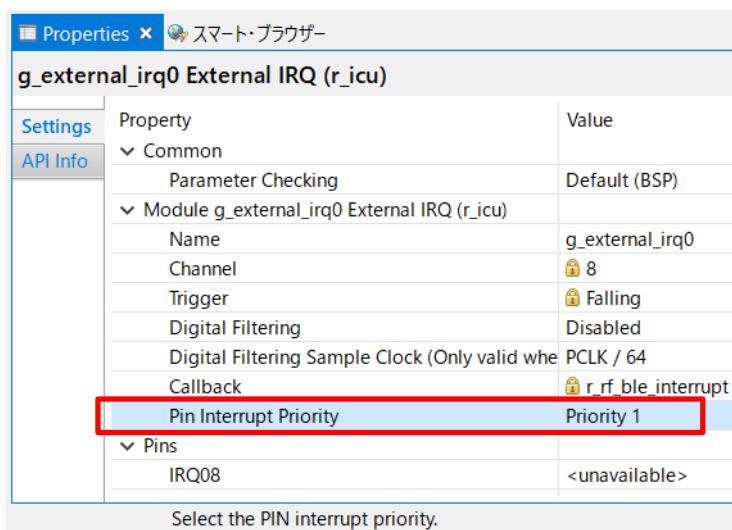


図 3-28 ICU の設定(FreeRTOS 環境)

3.5.4 r_gpt (g_timer1)

2. Add GPT Driver ボックスをクリックして New→Timer, General PWM (r_gpt)を選択します。

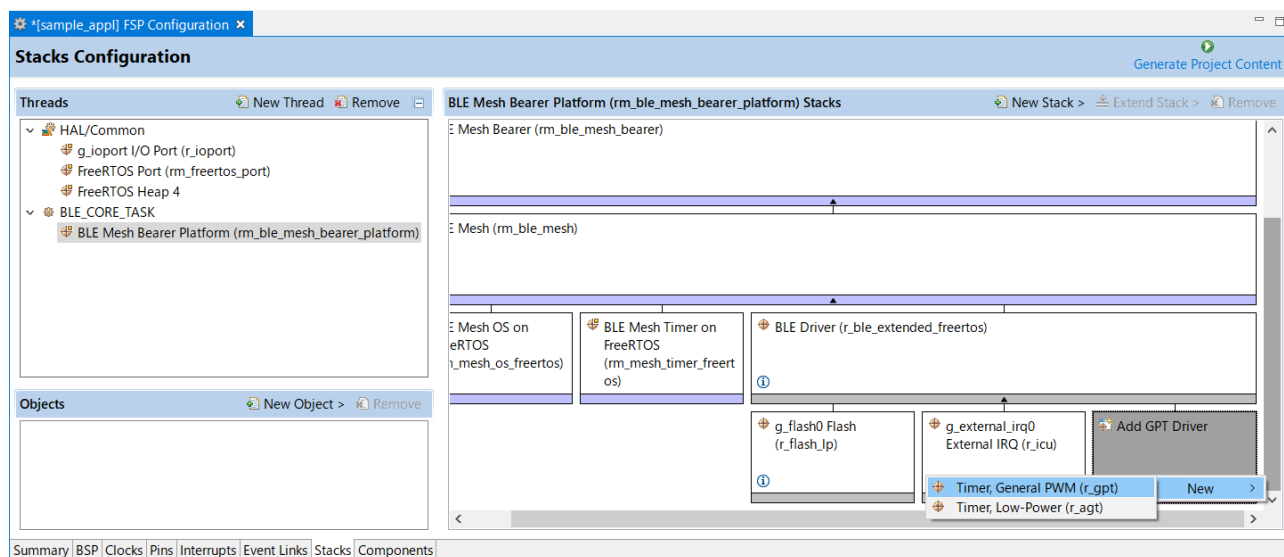


図 3-29 Bluetooth LE スタック用 GPT の追加

3. Properties タブで g_timer1 Timer, General PWM (r_gpt)の Overflow/Crest Interrupt Priority を Priority 2 に設定します。

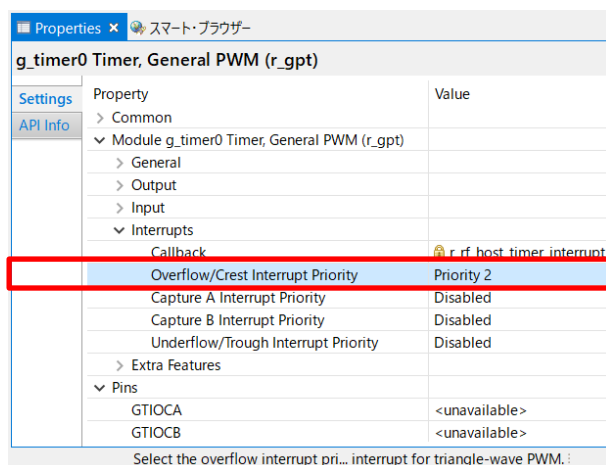


図 3-30 Bluetooth LE スタック用 GPT の設定

3.5.5 r_icu (g_ble_sw_irq)

EK-RA4W1 に実装されたスイッチを使用する場合、以下のように設定します。

1. New Stack をクリックして Input→External IRQ (r_icu) を HAL/Common に追加します。

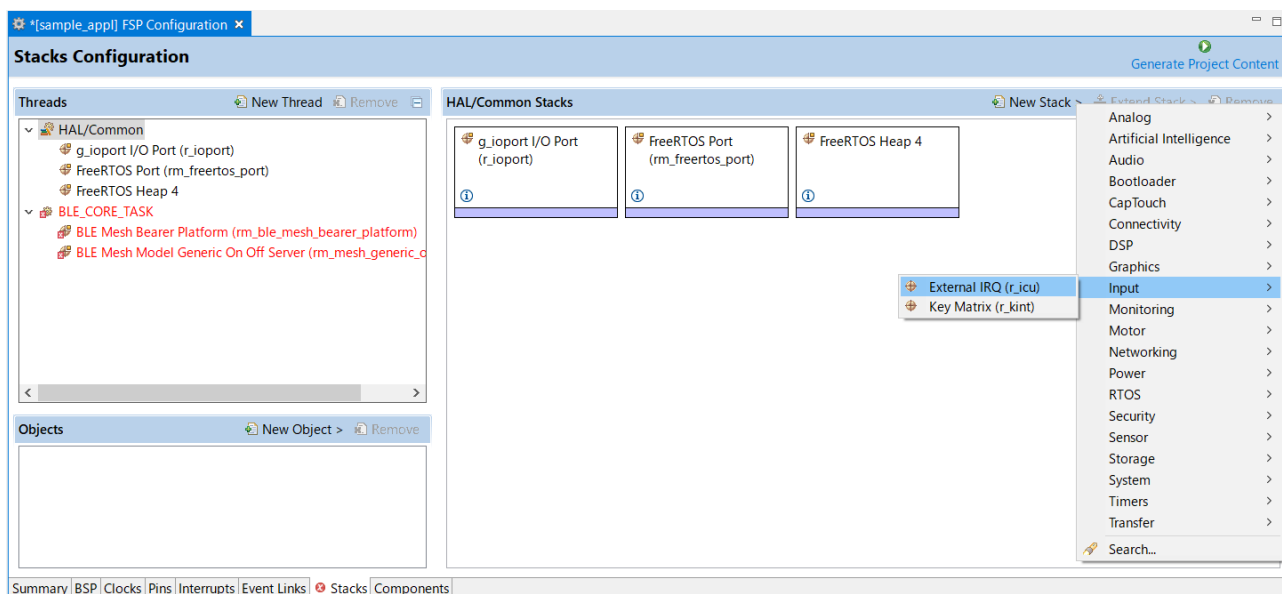


図 3-31 ICU Driver の追加

2. g_external_irq1 External IRQ (r_icu) を以下のように設定します。

- [Name] : g_ble_sw_irq
- [Channel] : 4
- [Trigger] : Falling
- [Callback] : Callback_ble_sw_irq
- [Pin Interrupt Priority] : Priority 2

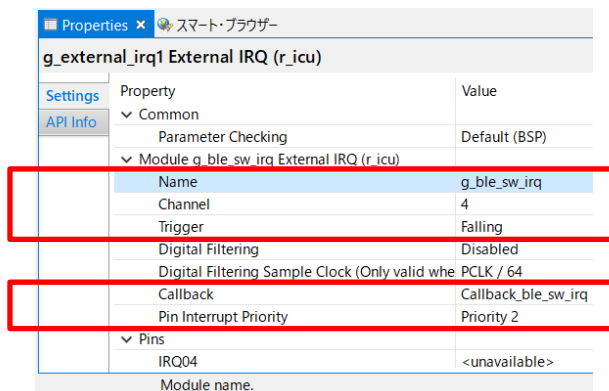


図 3-32 ICU Driver の設定

3.5.6 r_sci_uart

EK-RA4W1 でシリアル通信するためのコマンドラインインタフェース(CLI)を使用する場合、以下のよう
に設定します。

3.5.6.1 関連ソースファイル

CLI の関連ソースファイルはデモプロジェクトの./src/app_lib 配下に展開されています。デモプロジェク
トから app_lib フォルダを使用したいプロジェクトにコピーすることで CLI 機能を使用することができま
す。

3.5.6.2 SCI の設定

プロジェクトの FSP Configuration を開いて Stacks タブを表示します。HAL/Common に New
Stack→Connectivity→UART (r_sci_uart)を追加します。追加された r_sci_uart の設定を以下のように変更し
ます。

- [General]→[Channel] : 4
- [Interrupts]→[Callback] : user_uart_callback_ble_cli
- [Interrupts]→[xxx Interrupt Priority] : Priority 2
- [Pins]→[TXD] : P205
- [Pins]→[RXD] : P206

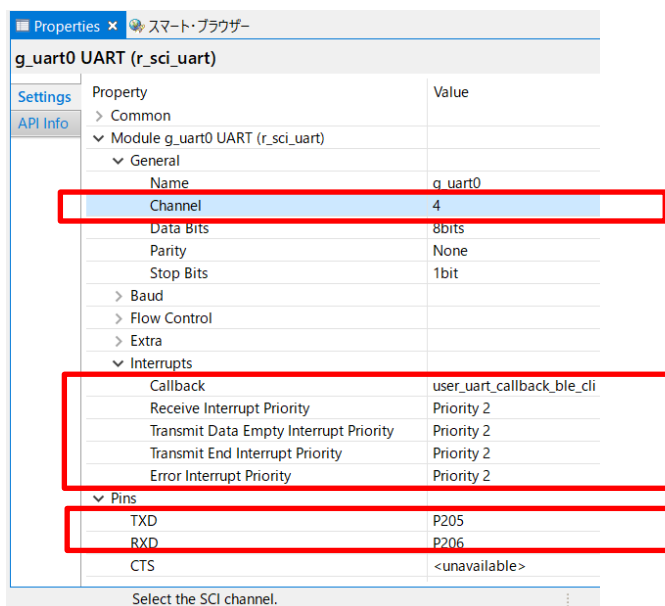


図 3-33 UART の設定

3.5.6.3 モジュール名の指定

r_sci_uart のモジュール名を g_uart0 から変更した場合は app_lib/r_ble_console.c の BLE_UART_INSTANCE マクロの値を編集します。

```

/*****
Macro definitions
*****/

#define BLE_TX_BUFSIZ      (180)
#define BLE_UART_INSTANCE (g_uart0)

```

Code 1. BLE_UART_INSTANCE マクロ

3.5.6.4 UART のシリアルデータ出力

UART のシリアルデータ出力は R_BLE_CLI_Printf() 関数によって行われます。R_BLE_CLI_Printf() 関数は printf() 関数と同じようにフォーマットされた文字列行を出力することができます。

表 3-12 R_BLE_CLI_Printf() の記述方法

Function Name	R_BLE_CLI_Printf	
Format	void R_BLE_CLI_Printf(const char *format, ...);	
Return	void	-
Arguments	const char *format	フォーマットを含む定数文字列行を指定します。
	...	フォーマットで指定された可変個の引数を指定します。

3.5.7 r_lpm

MCU の低電力モードを使用する場合、以下のように設定します。

1. New Stack をクリックして Power→Low Power Modes (r_lpm)を HAL/Common に追加します。

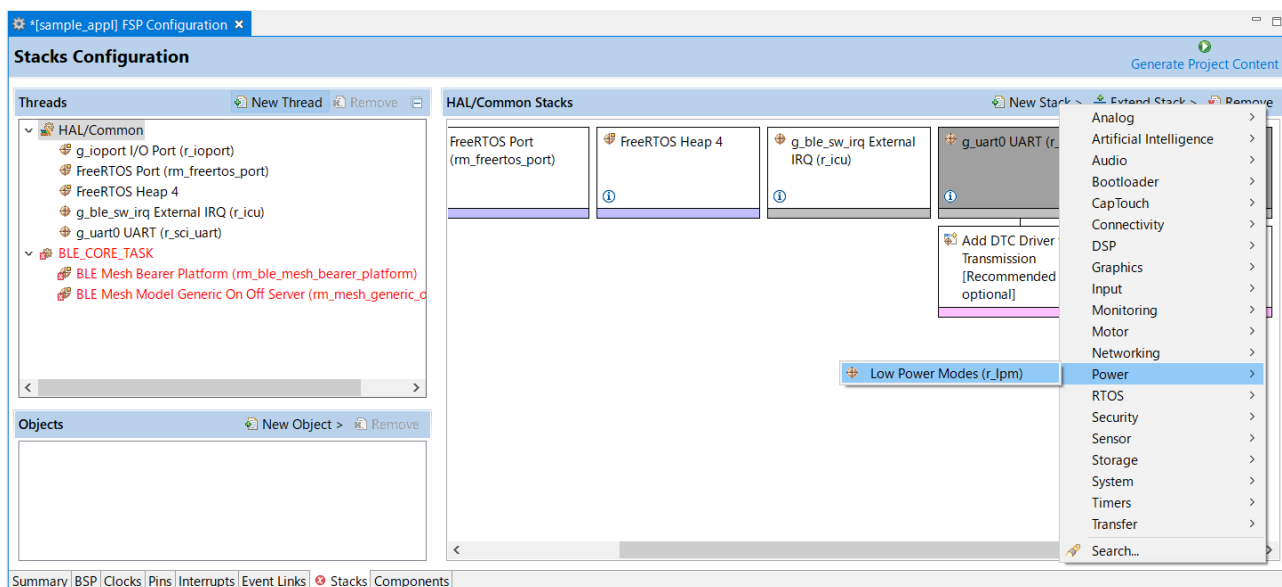


図 3-34 LPM の追加

3.6 コードの生成

FSP Configuration で[コードの生成]ボタン  をクリックします。
プロジェクトの ra、ra_gen、ra_cfg フォルダに各モジュールの API ヘッダ、ライブラリ、コード、データ、設定ファイルが生成されます。

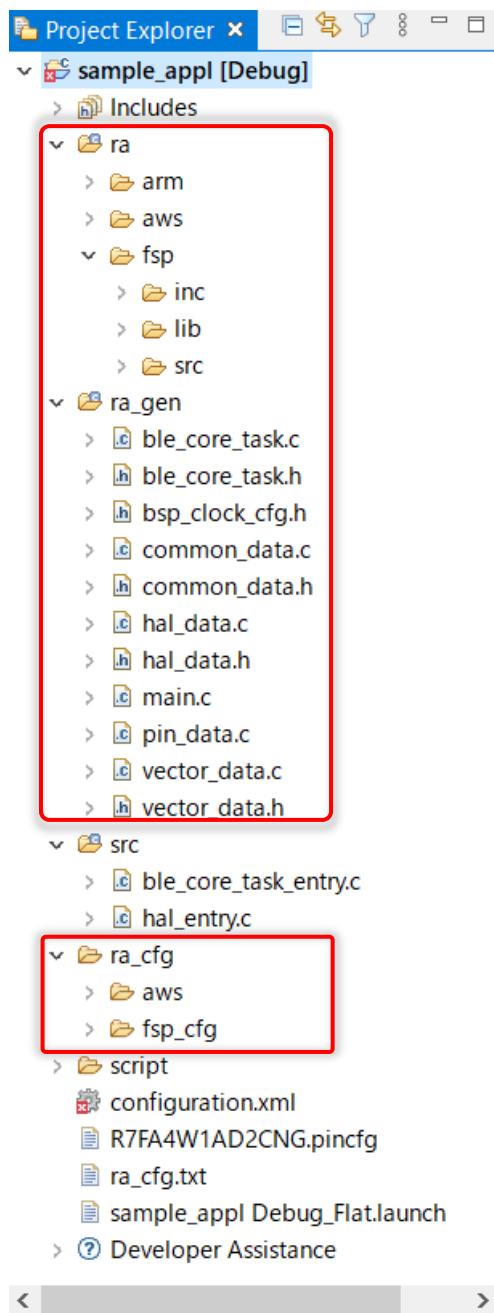


図 3-35 コードの生成結果

3.7 ビルドとデバッグ

2.2 節を参照してください。

4. Mesh アプリケーションの実装方法

Mesh スタックを使用する Mesh アプリケーションの実装方法は「RA4W1 グループ Bluetooth Mesh 開発ガイド」(R01AN5849)を参照してください。

5. Appendix

5.1 プログラムサイズ

表 5-1 にデモプロジェクトのプログラムサイズを示します。

表 5-1 プログラムサイズ

プロジェクト	ROM サイズ	RAM サイズ
ekra4w1_mesh_cli_client_baremetal	510KB	52KB
ekra4w1_mesh_cli_server_baremetal	477KB	53KB
ekra4w1_mesh_client_baremetal	334KB	48KB
ekra4w1_mesh_client_freertos	348KB	82KB
ekra4w1_mesh_server_baremetal	335KB	48KB
ekra4w1_mesh_server_freertos	349KB	82KB

商標権および著作権

Bluetooth® のワードマークおよびロゴは Bluetooth SIG, Inc が所有する登録商標であり、ルネサスエレクトロニクス株式会社はこれらのマークをライセンスに基づいて使用しています。その他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。

RA4W1 グループ Bluetooth Mesh スタックは次のオープンソースソフトウェアを使用します。

- [crackle](#); AES-CCM, AES-128bit 機能
BSD 2-Clause License

Copyright (c) 2013-2018, Mike Ryan
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

改訂記録

Rev.	発行日	改定内容	
1.00	2022.02.25	-	初版発行
1.01	2022.04.27	P.21	Total Heap Size を 10240 から 11264 に変更
		-	同梱のデモプロジェクトを更新
1.03	2022.08.29	P. 8	ソフトウェア要件を更新
		P.25	BLE Mesh Bearer Platform と BLE Driver ボックスのプロパティに同じ値を設定する注意書きを追加
		-	同梱のデモプロジェクトを更新
1.04	2022.10.26	P. 8	ソフトウェア環境を更新
		-	同梱のデモプロジェクトを更新
1.05	2022.12.16	P. 8	ソフトウェア環境を更新
		P. 27	FSP Configuration のパラメータを更新
		-	同梱のデモプロジェクトを更新

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くと、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、ノイズ輻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）、交通管制（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したします。
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。