

# 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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 (<u>https://www.bluetooth.com</u>)
- Mesh Profile 仕様 (<u>https://www.bluetooth.com</u>) 「Mesh Profile 1.0.1」を検索
- Mesh Model 仕様 (<u>https://www.bluetooth.com</u>) 「Mesh Model 1.0.1」を検索
- Renesas Flexible Software Package (FSP) User's Manual (R11UM0155)
- e<sup>2</sup> 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	2 FreeRTOS 環境での Mesh スタック追加	20
3.4.3	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	2 r_flash_lp	34
3.5.3	8 r_icu (g_external_irq0)	35
3.5.4	r_gpt (g_timer1)	36
3.5.5	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。





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。







## 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 ベアラー(ベアラープラットフォーム)
   Bluetooth ベアラーは 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 ベアラーは、<u>FSP</u>として提供されます。 Bluetooth LE スタックは、<u>FSP</u>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	Bluetooth LE スタック API ヘッダファイル
rm ble mesh xxx api.h	Mesh スタック API ヘッダファイル
rm_mesh_bearer_platform_api.h	Bluetooth ベアラーAPI ヘッダファイル
+ra\fsp\inc\instances\	
rm ble mesh xxx.h	Mesh スタックヘッダファイル
rm_mesh_bearer_platform.h	Bluetooth ベアラーヘッダファイル
rm_mesh_xxx.h	Mesh スタックヘッダファイル(Model 機能)
, +ra\fsp\lib\r_ble\	Bluetooth LE スタックライブラリ
+ra\fsp\lib\rm_ble_mesh\ 	Mesh スタックライブラリ
+ra_cfg\fsp_cfg\	
r_ble_cfg.h	Bluetooth LE スタック設定
rm_ble_mesh_cfg.h	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 <sup>®</sup> 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]
オンチップデバッギングエミュレータ	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	e <sup>2</sup> studio	2022-10	ルネサス製デバイス用の統合開発環境
	GCC ARM	V10	C/C++コンパイラ
	Embedded		(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. e<sup>2</sup> studio を起動しワークスペースディレクトリを指定します。

💽 e <sup>2</sup> studio Launcher	×
Select a directory as workspace	
e <sup>2</sup> studio uses the workspace directory to store its preferences and developm	ent artifacts.
Workspace: C:¥MyWork¥ra4w1_202110	<u>B</u> rowse
Use this as the default and do not ask again	
• <u>R</u> ecent Workspaces	
Launch	Cancel
Eddirit	cancer
図 2-1 ワークスペース選択	

2. File→Importを選択します。

📴 ra4w1_202110 - e² studio								
File	Edit	Navigate	Search	Project	Renesas Views	Ru		
۵,	New Open Open Recen	File Projects fro t Files	m File Sy	rstem	Alt+Shift+N	>		
	Close Close	Editor All Editors			Ctrl+W Ctrl+Shift+W			
9	Save Save Save Revert	As All			Ctrl+S Ctrl+Shift+S			
8	Move. Renan Refres Conve	 ne h rt Line Deli	miters To		F2 F5	>		
2	Print				Ctrl+P			
è	Impor	t	1	mport				
4	Export	t						
	Prope	rties			Alt+Enter			
	Switch Restar Exit	n Workspace t	9			>		

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。

Import		×
Select Create new projects from an archive file or directory.	[	2
Select an import wizard:		
type filter text		
<ul> <li>Seneral</li> <li>Archive File</li> <li>CMSIS Pack</li> <li>Existing Projects into Workspace</li> <li>File System</li> <li>Preferences</li> <li>Projects from Folder or Archive</li> <li>Rename &amp; Import Existing C/C++ Project into Workspace</li> </ul>		< v
? < <u>B</u> ack <u>Next &gt;</u> <u>F</u> inish	Canc	el

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。

Import Projects						^	
					Ĺ	7	
Select a directory to search for existing Eclipse projects.							
	[						
Select root directory:	C:¥MyWork¥ra4	w1_202110¥r01an	5848jj0100-ra4w1	~	B <u>r</u> ows	e	
O Select <u>a</u> rchive file:	C:¥MyWork¥ra4	w1_202110¥new2.	zip	~	B <u>r</u> owse		
<u>P</u> rojects:							
✓ ekra4w1_mesh_se	rver_baremetal (	C:¥MyWork¥ra4w1	_202110¥r01an584	8jj	<u>S</u> elect	All	
				[	<u>D</u> eselec	t All	
					R <u>e</u> fre	sh	
<				>			
Options							
Copy projects into we	ojects orkspace						
Close newly imported	d projects upon (	completion					
Hide projects that all	ready exist in the	workspace					
Working sets							
Add projec <u>t</u> to work	ing sets				Ne <u>w</u>		
Working sets:				1	S <u>e</u> lect		
<u> </u>							
?	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish		Cance	:I	
	_	•					



#### 2.2 ビルドとデバッグ

以下の手順を実行します。

e<sup>2</sup> studio でのデバッグについては「e<sup>2</sup> 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 Se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 <sup>I e kra4w1_mesh_server_barer > *</sup>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。
- 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 <sup>▶▶</sup>をクリックしてプロジェクトを実行します。
- 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 e 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

e<sup>2</sup> 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. e<sup>2</sup> 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。

New C/C++	Project – 🗆 🗙
Templates for	Renesas RA Project
All C/C++	Renesas RA C/C++ Project
	TRA Cleate an executable of static library C/C++ project for Reflesas RA.
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish Cancel

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。

Renesas RA C/C++ Project				_		×
Renesas RA C/C++ Project					Ē	-
Project Name and Location					6	
Project name						
sample_appl						
Use default location						
Location: C:¥MyWork¥ra4w1_202110¥sample_appl					Brov	vse
Choose file system: default 🗸						
0	< <u>B</u> ack	<u>N</u> ext >	Einish		Cance	el

**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。

Renesas RA C/C++ Project					×
Renesas RA C/C++ Project Device and Tools Selection					2
Device Selection FSP Version: 3.6.0-alpha0+20211230.d166da01 Board: Custom User Board (Any Device) Device: R7FA4W1AD2CNG Language: ● C ○ C++	Board Description				
	Device Details TrustZone Pins Processor	No 56 Cortex-M4			
Toolchains         GNU ARM Embedded         GCC ARM A-Profile (AArch64 bare-metal)         9.3.1.20200408	Debugger J-Link ARM				~
0	< <u>B</u> ack <u>N</u> e	ext > <u>F</u> inish		Cancel	

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。

Renesas RA C/C++ Project	- D X	Renesas RA C/C++ Project	- D X
Renesas RA C/C++ Project Build Artifact and RTOS Selection		Renesas RA C/C++ Project Build Artifact and RTOS Selection	2
Build Artifact Selection	RTOS Selection	Build Artifact Selection	PTOS Selection
Executable     Project builds to an executable file	No RTOS	<ul> <li>Executable</li> <li>Project builds to an executable file</li> </ul>	FreeRTOS (v10.4.3-LTS.Patch.2+fsp.3.6.0.alpha0.20211230.d166da01)
Static Library     Project builds to a static library file		Static Library     Project builds to a static library file	
Executable Using an RA Static Library     Project builds to an executable file     Project uses an existing RA static library project		Executable Using an RA Static Library     Project builds to an executable file     Project uses an existing RA static library project	
0	< <u>Back</u> Next > Einish Cancel	0	< Back Next > Finish Cancel

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。

Renesas RA C/C++ Project	□ ×	👩 Renesas RA C/C++ Project — 🗆 🗙
Renesas RA C/C++ Project	-	Renesas RA C/C++ Project
Project Template Selection		Project Template Selection
Project Template Selection		Project Template Selection
Bare Metal - Minimal Bare Metal - Minimal Remetal SP8 project and includes BSP This project will initialize clocks, pins, stacks, and the C runtime environment. [Remesas:RA3.6.0-alpha0+20211230.d166da01 pack]		FreeRTOS - Minimal - Static Allocation Encypt energics with no threads. FreeRTOS is pre-configured for static memory allocation. This project will initialize the MCU using the BSR (Renesas:RA3.60-alpha0+20211230.d166da01.pack)
Code Generation Settings Collues Renesss Code Formatter		Code Generation Settings ⊡luse Renesas Code Formatter
(2) Contract Next > Einish	Cancel	(2)     (

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

Propert	ies × 🏽 スマート・ブラウザー		28 -	' 🗆
Custom	User Board (Any Device)			
Settings	Property ✔ RA Common	Value		^
	Main stack size (bytes) Heap size (bytes)	0x1400 0x1000		
	MCU Vcc (mV) Parameter checking	3300 Disabled		-
	Assert Failures Error Log	Return FSP_ERR_ASSERTION No Error Log		
	Clock Registers not Reset Values during Start Main Oscillator Populated	Disabled Populated		
	PFS Protect C Runtime Initialization	Enabled Enabled		
	Early BSP Initialization Main Oscillator Clock Source	Disabled Crystal or Resonator		
	Subclock Populated Subclock Drive (Drive capacitance availabilit	Populated Standard/Normal mode		
	Subclock Stabilization Time (ms)	1000		$\sim$

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)	0x400	0x1400
(BSP_CFG_STACK_MAIN_BYTES)		
RA Common > Heap size (bytes)	0	0x1000
(BSP_CFG_HEAP_BYTES)		



## 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。

*[sample_appl] FSP Configuration ×		
Clocks Configuration		Generate Project Content
XTAL 8MHz > PLL Src: XTAL V PLL Div /2 V	<ul> <li>→ ICLK Div /1</li> <li>→ PCLKA Div /1</li> <li>→ PCLKB Div /1</li> </ul>	Restore Defaults     ICLK 32MHz     PCLKA 32MHz     PCLKB 32MHz
PLL Mul x12 PLL 48MHz HOCO 32MHz CLKOUT Disabled CLKOUT Disabled	<ul> <li>→ PCLKC Div /1</li> <li>→ PCLKD Div /1</li> <li>→ BCLK Div /64</li> <li>→ No output</li> </ul>	$\sim$ → PCLKC 32MHz $\sim$ → PCLKD 32MHz $\sim$ → BCLK 500kHz $\sim$ → BCLKout 0Hz
LOCO 32768Hz MOCO 8MHz	→ FCLK Div /2	✓ → FCLK 16MHz
SUBCLK 32768Hz		USBCLK 48MHz
Summary BSP Clocks Pins Interrupts Event Links Stacks Components		

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

 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。

Stacks Configuration <ul> <li>BLE Mesh Bearer (Im, be, mesh, bearer)</li> <li>BLE Mesh Bearer Plator(Im, Im, mesh, bearer, platform)</li> <li>BLE Mesh Bearer Platorm (Im, mesh, config.dt)</li> <li>BLE Mesh Model Generic Adminet Poperty Server (Im, mesh, generic, battery_clt)</li> <li>BLE Mesh Model Generic Adminet Poperty Server (Im, mesh, generic, battery_clt)</li> </ul>	Generate Pro itack > ≗ Extend Stack > Analog Artificial Intelligence Audio Pontienates	Diject Content
Threads <ul> <li>Image: States</li> <li></li></ul>	Generate Pro itack > ≗ Extend Stack > Analog Artificial Intelligence Audio Bentienden	Remove
Threads         Investignment         Investignment<	itack > ≗ Extend Stack > Analog Artificial Intelligence Audio	Remove
Threads         •	itack > ≚ Extend Stack > Analog Artificial Intelligence Audio	K Remove
<ul> <li>✓ MAU/Common</li> <li>Ø g.joport (/O Port (r_joport)</li> <li>Ø BLE Mesh Nodel Generic Admin Property Server (Line Munesh.generic_admin_prop_srv)</li> <li>Ø BLE Mesh Nodel Generic Admin Property Server (Line Munesh.generic_battery_cit)</li> <li>BLE Mesh Nodel Generic Battery Clint (m_mesh.generic_battery_cit)</li> <li>BLE Mesh Nodel Generic Battery Clint (m_mesh.generic_battery_cit)</li> </ul>	Analog Artificial Intelligence Audio	>
Image: Second	Artificial Intelligence Audio	>
BLE Mesh Model Generic Battery Client (m_mesh_generic_battery_clt)     BLE Mesh Model Generic Battery Client (m_mesh_generic_battery_clt)	Audio	
BEE Wish Model Generic Battery Crient (Int_Inest_generic_battery_crie)	Deptionales	>
Ithe Mach Model (separate Vattery Separate Internet Generate battery and	boolloadel	>
BLE Mean Model Generic Datery Server (Infinite generic Datery Server)     Beneric Datery Server (Infinite generic Charles) Server (Infini	CapTouch	>
BLE Mesh Model Generic Cheric Poperty Server (in mesh generic direct property of the server)	Connectivity	>
A RIE Mach Model Generic Default Transition Time Server (m med-generic def on)	DSP	>
DE Mart Model Generic Levial Client (minimum activity)     A BLE Mart Model Generic Levial Client (minimum activity)	Craphics	
BLE Mash Model Generic Level Server (m. mesh generic level sn)	land	
BEL Mash Model Generic Location (Linet (m. mesh deperic) (d. )	input	
BIE Mesh Model Generic Location Server (m_mash dependence)	Monitoring	2
Resh Madel Generic Manifacturer Property Server (m. mesh generic mfr. prop. sv)	Motor	>
Resh Model Generic On Off Client (mmssh generic on off cli)	Networking	>
BLE Mesh Model Generic On Off Server (m mesh generic on off six)	Power	>
BLE Mesh Model Generic Power Level Client (rm mesh generic pl.ch)	Security	>
BLE Mesh Model Generic Power Level Server (mmmesh generic nl sv)	Sensor	>
BLE Mesh Model Generic Power On Off Client (rm mesh generic poo clt)	Storage	>
BLE Mesh Model Generic Power On Off Server (m mesh generic pool sn)	System	>
Objects  New Object > K) Remove BLE Mesh Model Generic Property Client (m mesh generic prop. clt)	Timers	>
BLE Mesh Model Generic User Property Server (rm mesh generic user prop srv)	Transfer	>
BLE Mesh Model Health Client (rm. mesh. health.clt)	0 0 1	
BLE Mesh Model Health Server (rm mesh health srv)	Search	
BLE Mesh Model Light Control Client (rm mesh light ctl ctt)		
BLE Mesh Model Light Control Server (rm_mesh_light_ctl_srv)		
BLE Mesh Model Light HSL Client (rm_mesh_light_hsl_clt)		
BLE Mesh Model Light HSL Server (rm_rm_nesh_light_hsl_srv)		
BLE Mesh Model Light Lightness Client (rm_mesh_light_lightness_clt)		
Summary BSP Clocks Pins Interrupts Event Links Stacks Components 🛛 🕹 BLE Mesh Model Light Lightness Controller Client (m_mesh_light_lc_clt)		

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。

✿ *[sample_appl] FSP Configuration ×		- 8
Stacks Configuration		Generate Project Content
Threads 🗟 New Thread 🗟 Remove 🕞	HAL/Common Stacks	🛎 Extend Stack > 🖻 Remove
<ul> <li></li></ul>	e BLE Mesh Bearer Platform (m_mesh_bearer_platform)	
	U BLE Mesh Bearer (m_ble_mesh_bearer)	
	(	
	Stadd BLE Mesh OS         Stadd BLE Mesh Timer           Module         Stadd Resh Timer	BLE Driver (r_ble_ext
Objects 😢 New Object > 🖻 Remove		0
	New         Image: BLE Mesh OS on Baremetal (rm_mesh_os_freenteal)           Image: BLE Mesh OS on FreeRIOS (rm_mesh_os_freenteal)	<ul> <li>g_flash1 Flash (r_flash_lp)</li> </ul>
		0
Summary BSP Clocks Pins Interrupts Event Links @ Stacks Component	nts	

#### 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。

🌣 *[sample_appl] FSP Configurati	ion ×		- 0
Stacks Configuration			Generate Project Content
Threads	🗟 New Thread 🔊 Remove 😑	HAL/Common Stacks	k > 🚔 Extend Stack > 🖻 Remove
<ul> <li>✓ All AL/Common</li> <li>④ g_ioport I/O Port (r_iop</li> <li>④ BLE Mesh Bearer Platfo</li> </ul>	oort) rm (rm_mesh_bearer_platform)	et BLE Mesh Bearer Platform (m_mesh_bearer_platform)	
		f # BLE Mesh Bearer (rm_ble_mesh_bearer)	
		€ BLE Mesh (m_b)e_mesh)	
Objects	🗟 New Object > 🔊 Remove	# BLE Mesh OS on Baremetal (m. mesh os, baremet         Add BLE Mesh Timer Module         \$\$ Add Timer Driver (r,flash.p)	h BLE Driver (r_ble_ext
		New      BLE Mesh Timer on Bareme     BLE Mesh Timer on PreeRTO	tal (m_mesh_timer_baremetal) DS (rm_mesh_timer_freertos)
			(r_flash_lp)
Summary BSP Clocks Pins Intern	upts Event Links @ Stacks Componer	<	>

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。

SP *(new) FSP Configuration $\times$		- 0
Stacks Configuration		O Generate Project Content
Threads 🔊 New Thread 🕯 Remove 🖻	BLE Mesh Bearer Platform (rm_mesh_bearer_platform) Stacks	🙆 New Stack > 😤 Extend Stack > 🔞 Remove
<ul> <li>✓ MAL/Common</li> <li>              ₫ g_ioport I/O Port (r_ioport) <u>g</u><sup>2</sup> BLE Mesh Bearer Platform (rm_mesh_bearer_platform)      </li> </ul>	BLE Mesh Bearer Platform (rm_mesh_bearer_platform)	
	BLE Mesh Bearer (rm_ble_mesh_bearer)	Add BLE Network Driver
	BLE Mesh (m,ble_mesh)	BLE Driver (r_ble_balance)     BLE Driver (r_ble_compact)     BLE Driver (r_ble_extended)
Objects ① New Object > 12) Pernove	Image: BLE Mesh OS on Baremetal (m_mesh_use_bare)         Image: BLE Mesh Timer on Baremetal (m_mesh_use_bare)         Image: BLE Mesh Timer on Baremetal (m_mesh_use_bare)           Image: BLE Mesh Timer on Baremetal (m_mesh_use_bare)         Image: BLE Mesh Timer on Baremetal (m_mesh_use_bare)         Image: BLE Mesh Timer on Baremetal (m_mesh_use_bare)	r r g_flash Flash (r_flash_lp) ⊕
Summary BSP Clocks Pins Interrupts Event Links 9 Stacks Component		>

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。

🔅 [sample_appl] FSP Co	onfiguration ×	0	BLE Mesh Bearer Platform (rm_ble_mesh_bearer_platform)		
		\$	BLE Mesh Config Client (rm_mesh_config_clt)		0
Stacks Configura	ation	<b>+</b>	BLE Mesh Config Server (rm_mesh_config_srv)		Generate Project Con
		+	BLE Mesh Lower Trans (rm_ble_mesh_lower_trans)		
Threads	🔕 New Thread 🔞 Remove 📄	H/ 🌐	BLE Mesh Model Generic Admin Property Server (rm_mesh_generic_admin_prop_srv)	tac	🛛 🗢 Extend Stack 5 🖉 Rema
x 🖓 HAL/Common		•	BLE Mesh Model Generic Battery Client (rm_mesh_generic_battery_clt)		Analog >
a ioport I/O	Port (r. ioport)		BLE Mesh Model Generic Battery Server (rm_mesh_generic_battery_srv)		Artificial Intelligence
@ g_lopointi/O	arer Distform (m mach hearer platform)		BLE Mesh Model Generic Client Property Server (rm_mesh_generic_client_prop_srv)		Audio >
By DEC Wesh De	arei Plationn (m_nesi_bearei_plationn)	<b>+</b>	BLE Mesh Model Generic Default Transition Time Client (rm_mesh_generic_dtt_clt)		Bootloader >
		<b>•</b>	BLE Mesh Model Generic Default Transition Time Server (rm_mesh_generic_dtt_srv)		CapTouch >
		L 🕀	BLE Mesh Model Generic Level Client (rm_mesh_generic_level_clt)		Connectivity >
		• 🕀	BLE Mesh Model Generic Level Server (rm_mesh_generic_level_srv)		DSD
		+	BLE Mesh Model Generic Location Client (rm_mesh_generic_loc_clt)		Combine
		+	BLE Mesh Model Generic Location Server (rm_mesh_generic_loc_srv)		Graphics
		+	BLE Mesh Model Generic Manufacturer Property Server (rm_mesh_generic_mfr_prop_srv)		Input >
		E 🕁	BLE Mesh Model Generic On Off Client (rm_mesh_generic_on_off_clt)		Monitoring >
		•	BLE Mesh Model Generic On Off Server (rm_mesh_generic_on_off_srv)		Motor >
		•	BLE Mesh Model Generic Power Level Client (rm_mesh_generic_pl_clt)		Networking >
		•	BLE Mesh Model Generic Power Level Server (rm_mesh_generic_pl_srv)		Power >
		•	BLE Mesh Model Generic Power On Off Client (rm_mesh_generic_poo_clt)		Security >
		E 🕈	BLE Mesh Model Generic Power On Off Server (rm_mesh_generic_poo_srv)		Sensor >
		E 🕀	BLE Mesh Model Generic Property Client (rm_mesh_generic_prop_clt)		Storage )
		•	BLE Mesh Model Generic User Property Server (rm_mesh_generic_user_prop_srv)		Sustam
Objects	New Object > Remove		BLE Mesh Model Health Client (rm_mesh_health_clt)		Timer
,		+	BLE Mesh Model Health Server (rm_mesh_health_srv)		Timers /
		E 🕈	BLE Mesh Model Light Control Client (rm_mesh_light_ctl_clt)		Iranster >
		<b>+</b>	BLE Mesh Model Light Control Server (rm_mesh_light_ctl_srv)	1	Search
		<b>+</b>	BLE Mesh Model Light HSL Client (rm_mesh_light_hsl_clt)		(r flash Ip)
		•	BLE Mesh Model Light HSL Server (rm_rm_mesh_light_hsl_srv)		((_((((((((((((((((((((((((((((((((((((
			BLE Mesh Model Light Lightness Client (rm_mesh_light_lightness_clt)		(II)
		+	BLE Mesh Model Light Lightness Controller Client (rm_mesh_light_lc_clt)		-
		< 🕈	BLE Mesh Model Light Lightness Controller Server (rm_rm_mesh_light_lc_srv)		>
			BLE Mesh Model Light Lightness Server (rm_mesh_light_lightness_srv)		
ummary BSP Clocks F	Pins Interrupts Event Links 🤨 Stacks Compone	nts 🕁	BLE Mesh Model Light XYL Client (rm mesh light xyl clt)		

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。

# (new) FSP Configuration $\times$			- 0
Stacks Configuration			Generate Project Content
Threads	🕙 New Thread 🔞 Remove 🗉	BLE Mesh Config Server (rm_mesh_config_srv) Stacks	🔕 New Stack > 🔺 Extend Stack > 🔞 Remove
<ul> <li>         ▼ dispert I/O Port (r.)         Ø gjopent I/O Port (r.)      </li> <li>         Ø BLE Mesh Rever Pia      </li> <li>         Ø BLE Mesh Config Ciril      </li> <li>         Ø BLE Mesh Config Ser     </li> </ul>	ioport) tform (m. mesh_bearer_platform) ent (m. mesh_config. dt) veer (m. mesh_config.sv)	BLE Mesh Config Server (m, mesh, config.srv)      Add BLE Mesh Access Module     New     Server     Grm, ble, mesh, access0 BLE Mesh Ar	iceis (m.ble.mestr.access)
Objects	ඬ New Object > 🛍 Remove		
Summary BSP Clocks Pins Inte	errupts Event Links 9 Stacks Components	s	



## 3.4.2 FreeRTOS 環境での Mesh スタック追加

 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。

Threads	🔄 New Thread  😰 Remove 🛛 🖃	BLE_CORE_TASK Stacks	New Stack >	≗ Extend Stack >	Remove
<ul> <li>✓ ▲ H.</li> <li>⊕</li> <li>⊕</li> <li>BI</li> </ul>	AL/Common g gioport I/O Port (r_ioport) FreeRIOS Port (rm_freertos_port) LE_CORE_TASK	Add stacks to the selected three pasting here from the clipboar	ead by using the 'New Stack >	•* toolbar button (a	bove), or by
Objects	New Object > K. Remove				
Ummary Propert	<ul> <li>New Object &gt; メ」Remove</li> <li>BSP Clocks Pins Interrupts Event Links ties × 余スマート・ブラウザー</li> </ul>	Stacks Components			
ummary Propert	◆ New Object > ★ Remove BSP Clocks Pins Interrupts Event Links : ites × → スマート・ブラウザー RE_TASK	Stacks Components			
Ubjects ummary I Propert LE_COF ettings	<ul> <li>New Object &gt; ★: Remove</li> <li>BSP Clocks Pins Interrupts Event Links :</li> <li>BSP Clocks Pins Interrupts Event Links :</li> <li>Clocks Pins Interrupts Event Links :</li> <li>Property</li> <li>Thread</li> </ul>	Stacks Components Value			
Ummary Propert LE_COF ettings	<ul> <li>New Object &gt; ★: Remove</li> <li>BSP Clocks Pins Interrupts Event Links :</li> <li>BSP Clocks Pins Interrupts Event Links :</li> <li>Clocks Pins Interrupts Event Links :</li> <li>Thread</li> <li>Symbol</li> </ul>	Stacks Components Value ble_core_task			
ummary Propert LE_COF ettings	• New Object > ★: Remove BSP Clocks Pins Interrupts Event Links : ites ×	Stacks Components Value ble_core_task BLE_CORE_TASK			
ummary I Propert LE_COP ettings	<ul> <li>◆] New Object &gt; ★] Remove</li> <li>BSP Clocks Pins Interrupts Event Links tites ×</li></ul>	Stacks Components Value ble_core_task BLE CORE TASK 1024			
ummary I Propert LE_COF ettings	◆] New Object > x.] Remove BSP Clocks Pins Interrupts Event Links   BSP Clocks Pins Interrupts   BSP Clocks Pins Interrupts   BSP Clocks Pins   BSP Clocks	Stacks Components Value ble_core_task BLE_CORE_TASK 1024 1			
Objects ummary Propert LE_COF		Stacks Components Value ble_core_task BLE CORE_TASK 1024 1 NULL			
Objects iummary Propert LE_COF		Stacks Components Value ble_core_task BLE CORE TASK 1024 1 NULL Static			

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。

BLE_COP	RE_TASK	
Settings	Property	Value
	✓ Thread	
	Symbol	ble_core_task
	Name	BLE CORE TASK
	Stack size (bytes)	0x3000
	Priority	2
	Thread Context	NULL
	Memory Allocation	Static
	Allocate Secure Context	Enable

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	32	10
(configTIMER_QUEUE_LENGTH)		
Common > Optional Functions > <i>xTimerPendFunctionCall()</i> Function (INCLUDE_xTimerPendFunctionCall)	Enabled	Disabled

 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。





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。

*[sample_freertos] FSP Configuration ×	
Stacks Configuration	Generate Project Content
Threads 📀 New Thread 🗟 Remove 🗉	BLE_CORE_TASK Stacks 🔊 New Stack > 🏝 Extend Stack > 🔊 Remove
<ul> <li>AL/Common</li> <li>g_ioport I/O Port (r_ioport)</li> <li>FreeRTOS Port (rm_freertos_port)</li> <li>BLE_CORE_TASK</li> </ul>	BLE Mesh Bearer Platform (rm_mesh_bearer_platform)
BLE Mesh Bearer Platform (rm_mesh_bearer_platform)	• • • • • • • • • • • • • • • • • • •
	BLE Mesh Bearer (rm_ble_mesh_bearer)      BLE Mesh (rm_ble_mesh)
Objects 🕢 New Object > 🛍 Remove	
	Add BLE Mesh OS     Add BLE Mesh Timer Module     Add BLE Mesh Timer Module     Add Timer Driver     # g_flash0 Flash (r_flash_lp)     # Add BLE Mesh Driver       New     # BLE Mesh OS on Baremetal (rm_mesh_os_baremetal)       # BLE Mesh OS on FreeRTOS (rm_mesh_os_freertos)
Summary BSP Clocks Pins Interrupts Event Links @ Stacks Con	ponents

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。

🏶 *[sample_freertos] FSP Configuration 🗙		
Stacks Configuration		Generate Project Content
Threads 📀 New Threads	ead 🔊 Remove 😑	BLE_CORE_TASK Stacks  New Stack >  Extend Stack >  Remove
		BLE Mesh Bearer Platform (rm_mesh_bearer_platform)
d BLE Mesh Bearer Platform (rm_mes	h_bearer_platform)	
		BLE Mesh Bearer (rm_ble_mesh_bearer)
Objects 📀 New O	bject > 💼 Remove	
		Image: Second
Summary BSP Clocks Pins Interrupts Event I	inks 🥹 Stacks Comp	onents

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。

*[sample_freertos] FSP Configuration ×	
Stacks Configuration	Generate Project Content
Threads 🔄 New Thread 🗟 Remove	□     BLE_CORE_TASK Stacks       ● New Stack > ▲ Extend Stack > ▲ Remove
<ul> <li>AL/Common</li> <li>g_ioport I/O Port (r_ioport)</li> <li>FreeRTOS Port (rm_freertos_port)</li> <li>BLE_CORE_TASK</li> </ul>	sh Bearer Platform (rm_mesh_bearer_platform)
d <sup>#</sup> BLE Mesh Bearer Platform (rm_mesh_bearer_platform	sh Bearer (rm_ble_mesh_bearer)
Objects	sh (rm_ble_mesh) sh OS on DS sh_os_freetOS (m_mesh_timer_freett os)
Summary BSP Clocks Pins Interrupts Event Links @ Stacks Co	mponents

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。

Stacks Configuration         Threads       New Thread       Remove       HAL/Common Stacks       New N <ul> <li></li></ul>	Generate F Iow Stark & Extend Stark Analog Artificial Intelligence	Project Content
Threads     New Thread     Remove     HAL/Common Stacks <ul> <li></li></ul>	Analog Artificial Intelligence	Remove
	Audio Bootloader CapTouch Connectivity DSP Graphics Input Monitoring Motor Networking Power	> > > > > > > > > > > > > > > > > > >
Objects <ul> <li>PreeRTOS Buffer Allocation 2</li> <li>FreeRTOS Buffer Allocation 2</li> <li>FreeRTOS Heap 1</li> <li>FreeRTOS Heap 2</li> <li>FreeRTOS Heap 3</li> <li>FreeRTOS Heap 4</li> <li>FreeRTOS Heap 5</li> </ul>	RTOS Security Sensor Storage System Timers Transfer	> > > > > > > > > > > > > > > > > > >

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	7	1
(MESH_BEARER_PLATFORM_CFG_RF_CONNECTION_MAXIMUM)		
Maximum advertising data length	1650	31
(MESH_BEARER_PLATFORM_CFG_RF_ADVERTISING_DATA_MAXIMUM)		
Maximum advertising set number	4	1
(MESH_BEARER_PLATFORM_CFG_RF_ADVERTISING_SET_MAXIMUM)		
Maximum periodic sync set number	2	1
(MESH_BEARER_PLATFORM_CFG_RF_SYNC_SET_MAXIMUM)		

[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。

BLE Mesh Settings	n Bea	way Diatfourne (une maash is sever u		
Settings	BLE Mesh Bearer Platform (rm_mesh_bearer_platform)			
	Prope	erty	Value	
-	✓ Co	ommon		
-		Debug Public Address	FF:FF:FF:50:90:74	
		Debug Random Address	FE:FE:FE:FE:FE:FE	
		Maximum number of connections	1	
		Maximum connection data length	251	
		Maximum advertising data length	31	
		Maximum advertising set number	1	
		Maximum periodic sync set number.	1	
		Store Security Data	Disable	
		Data Flash Block for Security Data	0	
		Remote Device Bonding Number	7	
		Connection Event Start Notify	Disable	
		Connection Event Close Notify	Disable	
-		Advertising Event Start Notify	Disable	
		Advertising Event Close Notify	Disable	
		Scanning Event Start Notify	Disable	
		Scanning Event Close Notify	Disable	
		Initiating Event Start Notify	Disable	
		Initiating Event Close Notify	Disable	
		RF Deep Sleep Start Notify	Disable	
		RF Deep Sleep Wakeup Notify	Disable	
-		Bluetooth dedicated clock	6	
		Maximum number of connections.		

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	1	6
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		
Memory Pool → Memory Pool Size	0x4000	0x3000

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

🔲 Properti	ies 🗙 🎋 Debug 🛷 Search 📱 Stack Analysis	🔋 Memory Usage
BLE Mes	h (rm_ble_mesh)	
Settings	Property	Value
	> Common	
	<ul> <li>Module BLE Mesh (rm_ble_mesh)</li> </ul>	
	> General	
	✓ Storage	
	Block Number	6
	<ul> <li>Memory Pool</li> </ul>	
	Memory Pool Size	0x3000
	> Logging	
	Channel Number	0
	Network Interface Number	2
	Provisioning Interface Number	2
	Network Cache Size	10
	Network Sequence Number Cache Size	32
	Maximum Number of Subnet	4
	Maximum Number of Device Key	4
	Proxy Filter List Size	2
	Maximum Number of LPN	1
	Reassembled Cache Size	8
	Maximum Number of Segmentation and	8
	Maximum Number of Friend Message Q	15
	Maximum Number of Friend Subscriptio	8
	Maximum Number of Element	4
	Block number.	

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	Mesh ネットワークに使用するベアラー数
(network_interfaces_num)	最小值: 1
*デフォルト値: 2	最大値: (1 +
	MESH_BEARER_PLATFORM_CFG_RF_CONNECTIO
	N_MAXIMUM)
	   最初のベアラーは ADV Bearer 弾るベアラーは同時に
	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	プロビジョニングに使用するベアラー数
(provisioning_interfaces_num)	最小值: 1
*デフォルト値: 2	最大値: 2
	木設定を1とした場合 PB-ADV Bearer のみ使用でき
	ふした。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	Unprovisioned Device Beaconの送信間隔[msec]
Milliseconds	最小值: 20
(unprov_device_beacon_timeout)	
*デフォルト値: 200	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
	9 る場合、Connectable Advertising PDU を本設定间隔 で送信する DP ADV ト DP CATT の五ちた使用する提
	C医信9 る。FB-ADV と FB-GATT の両方を使用9 る場 会 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 Message Cache が保持できる最大ノード数
(network_cache_size)	最小值: 2
*デフォルト値: 10	
	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
Natural Matural Ormana Muscher Oraba Oira	れる。 Natural Massars Osaka が得せてたてをす。 ドの OFO
Network → Network Sequence Number Cache Size	Network Message Cache か保持でさる谷ノートの SEQ
(network_sequence_nun_cache_size)	留亏奴 是小值·32
Network→Maximum Number of Subnet	取り値 02 Network Key NID などのサブネット情報の最大数
(maximum subnets)	最小值:1
/  *デフォルト値: 4	
Network→Maximum Number of Device Key	Device Key の最大数
(maximum_device_keys)	最小值: 1
*デフォルト値: 4	
	Configuration Client Model を使用しない場合、本設定
Notwork->Drovy Eilter List Size	は1じよい。 タ Dravy Eilter Liet に発行可能たフビレスの早十数
(provy filter list size)	合 PTOXY FILLEF LISE I〜 豆球可能などトレムの取入数   畠小値・1
(pi o,y_iiitei_iist_size)   *デフォルト値: 2	1



Network ->Network Sequence Number Block Size	Data Flach にまき込む SEO 悉号の問隔
(not sequence number block size)	
	取い唱・「
*テフオルト値: 2048	
	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、SFQ 番号が 2048、4096
	と 2018 の 仲粉に 到達する 毎に Data Flack へ まきひま
	と 2040 の旧数に到廷 9 の母に Data Tidsii 、音とどよ
	れる。もし SEQ 备ちか 3000 の時点 C MCU かりセッ
	トされた場合、SEQ 番号は 4096 から冉開する。
	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	Network Transmit Count ステートのデフォルト値
(net tx count)	□ 最小値·0
(hot_ot_ot_otant)    *デフォルト店・1	□ 块 1 / / / / / / / / / / / / / / / / / /
Network Network Interval Stere for Network Deckets	取八に./
	Network Transmit Interval Steps ステートのテノオルト
(net_tx_interval_steps)	
*デフォルト値: 4	最小値: 0
	最大値: 31
Network→Network Transmit Count for Relayed Packets	Relay Retransmit Count ステートのデフォルト値
(net relay tx count)	最小值·0
(!!!!_!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!	□ 缺 1 № 0   是 士 値 · 7
Network Metwork Interval Stone for Deleved Deckets	取入値. / Rolov Detronomit Interval Stone フェートのデフナルト
	Relay Retransmit Interval Steps $XT - FOT JT VF$
(net_relay_tx_interval_steps)	
*デフォルト値: 9	最小値: 0
	最大値: 31
Network→Proxy ADV Network ID Timeout for Each Subnet	Proxy Advertisement with Network ID の送信間隔[msec]
in Milliseconds	最小値: 20
(proxy subnet netid adv timeout)	
(====================================	
Network -> Provy ADV Nede Identity Timeout for Each	 Draxy, Advartisement with Nede Identityの送信問院
Subnet in Milliseconds	
(proxy_subnet_nodeid_adv_timeout)	最小值: 20
*デフォルト値: 300	
Network→Proxy ADV Node Identity Overall Time Period in	Proxy Advertisement with Node Identity の送信期間
Milliseconds	[sec]
(proxy nodeid adv timeout)	   最小值·1
(picky_inducia_adv_anicodd)   *デフォルト値: 60	
	Natural PDU の送信さっ サイブ
	Network PDU の送信キューサイス
Iransmission	最小値: 2
(net_tx_queue_size)	
*デフォルト値: 64	
Transport→Maximum Number of LPN	Friend Node としてフレンドシップを確立する対向
(maximum lpn)	Low Power Node の最大数
│、/ / │*デフォルト値・1	↓ 最小值·1
Transport-Poplay Protection Cache Size	Poplay Protection Cache # $ \vec{ x} $
(replay apple aize)	- Ropiay Floteolion Caone サイス - 具小店・2
(replay_cache_size)	□ 取小唱. 2
_ *テフオルト値: 10	
Transport→Reassembled Cache Size	Segmentation and Reassembly(SAR)処理の受信メッセ
(reassembled_cache_size)	ージキャッシュサイズ
*デフォルト値: 8	最小值: 2
Transport→Friend Poll Retry Count	Low Power Node が Friend Update メッヤージを受信で
(frnd poll retry count)	きなかった場合の Friend Poll メッセージ再送回数
*デフォルト値: 10	



Transport→Maximum Number of Segmentation and	Segmented メッセージの送受信に使用する
Reassembly Context	Segmentation and Reassembly(SAR)処理のコンテキス
(maximum ltrn sar context)	L数
	□□ 数 □ = 小店: 0
テノオルト値.o	
	Segmented メッセーンの再达间隔[msec]
I imeout in Milliseconds	最小值:200
(ltrn_rtx_timeout)	
*デフォルト値: 300	
Transport->Lower Transport Segment Re-Transmission	Segmented メッセージの再送回数
Count	最小值: 2
(ltrn_rtx_count)	
*デフォルト値: 2	
Transport->Lower Transport Acknowledgement Timeout in	Segmented Acknowledgement メッセージの送信間隔
Milliseconds	[msec]
(ltrn ack timeout)	
、 / デフォルト値: 200	
Transport->Lower Transport Incomplete Timeout in	Segmented メッヤージの受信キャンヤルタイムアウト
Milliseconds	時間 [sec]
(Itrn incomplete timeout)	最小值: 10
((((())))) ((()))) (() () (() () () () () () () () () () ()	
Transport->Friendship Receive Window	Iow Power Node の母信ウィンドウサイブ [msec]
(frad receive window)	EUW FOWEI Node の受信ワインドワッイス [IIISEC] 早小店: 100
	取小但.100
	各 Low Power Node に対するメッセーシキュー数
(maximum_friend_message_queue)	最小值:2
*デフォルト値: 15	
Transport→Maximum Number of Friend Subscription List	各 Low Power Node に対するフレンドサブスクリプシ
(maximum_friend_subscription_list)	ョンリストの最大数
*デフォルト値: 8	最小値: 1
Transport→Friend Clear Confirmation Timeout in	Low Power Node が Friend Clear Confirmation メッセー
Milliseconds	ジを受信できなかった場合の Friend Clear メッセージ
(lpn_clear_retry_timeout_initial)	の再送間隔[msec]
*デフォルト値: 1000	最小値: 1000
Transport→Friend Clear Retry Count	Low Power Node が Friend Clear Confirmation メッセー
(lpn clear retry count)	ジを受信できなかった場合の Friend Clear メッセージ
*デフォルト値:5	再送回数
	最小值:1
Transport→Friendship Retry Timeout in Milliseconds	Low Power Node による Friend Request メッセージの
(trn_frndreg_retry_timeout)	送信期間 [msec]
(m_maroq_rony_mood)    *デフォルト値: 1200	是小值·1100
Access Maximum Number of Element	
(maximum_access_element_num)	<b>取小胆</b>
	モナルの最大致
(maximum_access_model_num)	最小値: 1
*デフォルト値: 60	
Access→Maximum Number of Application	Application Key の最大数
(maximum_application)	最小値: 1
*デフォルト値: 8	
Access→Maximum Number of Virtual Address	バーチャルアドレスの最大数
(maximum_virtual_address)	最小值: 1
*デフォルト値: 8	
Access→Maximum Number of Non-Virtual Address	非バーチャルアドレス(ユニキャストアドレス、グルー
(maximum_non_virtual_address)	プアドレス)の最大数
*デフォルト値: 8	最小值: 1



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

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



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

## 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	Network 処理のコールバック
(p_callback)	
*デフォルト値: NULL	

#### 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	Upper Transport 処理のコールバック	
(p_callback)		
*デフォルト値: NULL		

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

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



# 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。

*[sample_appl] FSP Conf	figuration ×					- 8
Stacks Configuratio	on					Generate Project Content
Threads	🐑 New Thread ඬ Remove 📋	HAL/Common Stacks			🔊 New Stack >   ≜	Extend Stack > 🔞 Remove
<ul> <li>✓          <sup>®</sup> HAL/Common         <sup>⊕</sup> g_ioport I/O Port         <sup>⊕</sup> BLE Mesh Bearer</li> </ul>	t (r_ioport) r Platform (rm_mesh_bearer_platform)	BLE Mesh Bearer Platform	(rm_mesh_bearer_platform	n)		
		BLE Mesh Bearer (rm_ble_	mesh_bearer)			
					<u> </u>	
		BLE Mesn (m_ble_mesn)		1 <u></u>	1	
		BLE Mesh OS on Baremetal	BLE Mesh Timer on Baremetal	Add Timer Driver	g_tlash0 Flash (r_flash_in)	BLE Driver (r_ble_ext
Objects	New Object > R Remove	(rm_mesh_os_baremet	(rm_mesh_timer_bare		(Current Ob)	
		al)	metal)		New ) @ Timer G	eneral PWM (r. ont)
				,	<ul> <li>Timer, Lo</li> </ul>	ow-Power (r_agt)
					·	(r_flash_lp)
		<				>
Summary BSP Clocks Pins	Interrupts Event Links @ Stacks Component	its				

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。

g_timer	) Timer, General PWM (r_gpt)			
Settings	Property ✓ Common	Value		
AFTINIO	Parameter Checking	Default (BSP)		
	Pin Output Support	Disabled		
	Write Protect Enable	Disabled		
	Clock Source	PCLKD		
	<ul> <li>Module g_timer0 Timer, General PWM (r_gpt)</li> </ul>			
	> General			
	> Output			
	> Input			
	✓ Interrupts			
_	Callback	NULL		
_ L	Overflow/Crest Interrupt Priority	Priority 3		
_	Capture A Interrupt Priority	Disabled		
	Capture B Interrupt Priority	Disabled		
	Underflow/Trough Interrupt Priority	Disabled		
	<			

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。

_flash0	Flash (r_flash_lp)	
ettings	Property ✓ Common	Value
PLINTO	Parameter Checking	Default (BSP)
	Code Flash Programming	Disabled
	Data Flash Programming	Enabled
	<ul> <li>Module g_flash0 Flash (r_flash_lp)</li> </ul>	
	Name	g_flash0
	Data Flash Background Operation	Disabled
	Callback	NULL
	Flash Ready Interrupt Priority	Disabled
	Enabling allows Flash API calls that ref o	peration continuing in the background

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。
  - ベアメタル環境

Priority (C Priority 0

Propert	ies × 🦗 スマート・ブラウザー				
g_extern	g_external_irq0 External IRQ (r_icu)				
Settings	Property ✔ Common	Value			
7411110	Parameter Checking	Default (BSP)			
	<ul> <li>Module g_external_irq0 External IRQ (r_icu)</li> </ul>				
	Name	g_external_irq0			
	Channel	8			
	Trigger	🔒 Falling			
	Digital Filtering	Disabled			
	Digital Filtering Sample Clock (Only valid whe	PCLK / 64			
	Callback	🔒 r rf ble interrupt			
	Pin Interrupt Priority	Priority 0 (highest)			
	✓ Pins				
	IRQ08	<unavailable></unavailable>			
L	Select the PIN interrupt priority.				

図 3-27 ICU の設定(ベアメタル環境)

• FreeRTOS 環境

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

■ Properties × 🏟 スマート・ブラウザー				
g_exterr	nal_irq0 External IRQ (r_icu)			
Settings	Property ✔ Common	Value		
AFTINO	Parameter Checking	Default (BSP)		
	<ul> <li>Module g_external_irq0 External IRQ (r_icu)</li> </ul>			
	Name	g_external_irq0		
	Channel	8		
	Trigger	🔒 Falling		
	Digital Filtering	Disabled		
	Digital Filtering Sample Clock (Only valid whe	PCLK / 64		
	Callback	<pre>f_ble_interrupt</pre>		
	Pin Interrupt Priority	Priority 1		
	✓ Pins			
	IRQ08	<unavailable></unavailable>		
L	Select the PIN interrupt priority.			

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。

* *(sample_appl] FSP Configuration ×				
Stacks Configuration		Generate Project Content		
Threads 📀 New Thread 🗟 Remove 🗉	BLE Mesh Bearer Platform (rm_ble_mesh_bearer_platform) Stacks	🔁 New Stack > 🐣 Extend Stack > 🛍 Remove		
<ul> <li>✓</li></ul>	E Mesh Bearer (rm_ble_mesh_bearer)	^		
	The sh (rm_ble_mesh)			
	Mesh OS on eRTOS mesh_os_freertos)	ded_freertos)		
Objects  New Object > Remove	<ul> <li> <b>g_flash0 Flash</b>             (r_flash_lp)</li></ul>			

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。

imer	0 Timer, General PWM (r_gpt)	
tinas	Property	Value
nfo	> Common	
<u> </u>	<ul> <li>Module g_timer0 Timer, General PWM (r_gpt)</li> </ul>	
	> General	
	> Output	
	> Input	
	✓ Interrupts	
	Callback	🔒 r rf host timer interrupt
	Overflow/Crest Interrupt Priority	Priority 2
	Capture A Interrupt Priority	Disabled
	Capture B Interrupt Priority	Disabled
	Underflow/Trough Interrupt Priority	Disabled
	> Extra Features	
	✓ Pins	
	GTIOCA	<unavailable></unavailable>

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。

🔅 *[sample_appl] FSP Configuration 🗙		- 0
Stacks Configuration	Generate Project Cor	ntent
Stacks Configuration         Threads       New Thread Remove         Image: Stack	Cenerate Project Cor Cenerate Project Cor Antificial Intelligence Audio Bootloader CapTouch Connectivity DSP Graphics Input Monitoring Motor Networking	ntent
	Power RTOS	> >
Objects  New Object >  Remove	Security Sensor Storage System Timers Transfer	> > > > >



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] :
- [Trigger]
- [Callback]
- : Callback\_ble\_sw\_irq

Priority 2

Falling

4

• [Pin Interrupt Priority] :

Propert	ies × 🏶 スマート・ブラウザー				
g_exterr	nal_irq1 External IRQ (r_icu)				
Settings	Property ✓ Common	Value			
AFTINO	Parameter Checking V Module g_ble_sw_irq External IRQ (r_icu)	Default (BSP)			
	Name	g_ble_sw_irq			
	Channel	4			
	Trigger	Falling			
	Digital Filtering	Disabled			
	Digital Filtering Sample Clock (Only valid whe PCLK / 64				
	Callback	Callback_ble_sw_irq			
	Pin Interrupt Priority	Priority 2			
	✓ Pins				
	IRQ04	<unavailable></unavailable>			
	Module name				

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。

Priority 2

P205

P206

• [General]→[Channel]

4

:

1

:

:

• [Interrupts]→[Callback]

user\_uart\_callback\_ble\_cli

- [Interrupts]→[xxx Interrupt Priority] :
- [Pins]→[TXD]
- [Pins]→[RXD]

Propert	ties × 🛞 スマート・ブラウザー		
g_uart0	UART (r_sci_uart)		
Settings API Info	Property > Common → Module g_uart0 UART (r_sci_uart)	Value	
	✓ General		
_	Name	g_uart0	
	Channel	4	
_	Data Bits	8bits	
	Parity	None	
	Stop Bits	1bit	
	> Baud		
	> Flow Control		
	> Extra		
_	✓ Interrupts		
	Callback	user_uart_callback_ble_cli	
	Receive Interrupt Priority	Priority 2	
	Transmit Data Empty Interrupt Priority	Priority 2	
	Transmit End Interrupt Priority	Priority 2	
	Error Interrupt Priority	Priority 2	
	✓ Pins		
	TXD	P205	
L	RXD	P206	
	CTS	<unavailable></unavailable>	
	Select the SCI channel.		

#### 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。

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<sup>®</sup> 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。

• <u>crackle;</u> 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<sub>L</sub>(Max.)から V<sub>H</sub>(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<sub>L</sub>(Max.)からV<sub>H</sub> (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。

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

- 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。
- 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 ロ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。