RX23W グループ

Bluetooth Mesh モジュール Firmware Integration Technology

要旨

本書は Firmware Integration Technology(FIT)を使用した Bluetooth® Mesh モジュールについて説明します。本モジュールは Bluetooth Mesh Networking 仕様に準拠した多対多の無線通信機能を提供します。

以後、本書では本モジュールを Mesh FIT モジュールと称します。

対象デバイス

RX23W グループ

関連ドキュメント

- Bluetooth Core 仕様 (bluetooth.com/specifications/)
- Bluetooth Mesh Networking 仕様 (bluetooth.com/specifications/)
- CC-RX コンパイラ ユーザーズマニュアル (R20UT3248)
- e² studio ユーザーズマニュアル 入門ガイド (R20UT4374)
- RX スマート・コンフィグレータ ユーザーガイド e² studio 編 (R20AN0451)
- Firmware Integration Technology ユーザーズマニュアル (R01AN1833)
- RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)
- RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)
- RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- RX ファミリ フラッシュモジュール Firmware Integration Technology (R01AN2184)
- RX23W グループ BLE モジュール Firmware Integration Technology (R01AN4860)
- RX23W グループ Bluetooth Mesh スタック スタートアップガイド (R01AN4874)
- RX23W グループ Bluetooth Mesh スタック 開発ガイド (R01AN4875)

目次

1. 概要

1.1 提供機能

Mesh FIT モジュールは 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。本モジュール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

1.2　ソフトウェア構成

図 1-1 に Mesh FIT モジュールを使用するためのソフトウェア構成を示します。

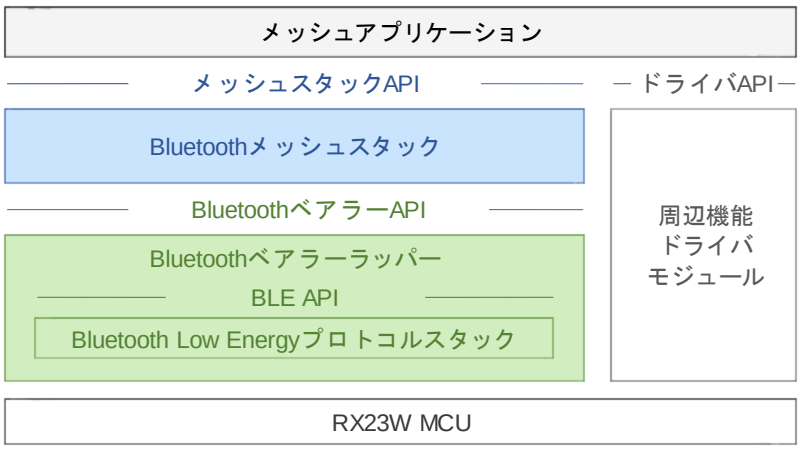

図 1-1　ソフトウェア構成

Mesh FIT モジュールを使用するためのソフトウェア構成は下記のソフトウェアで構成されます。

- **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**
  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、Mesh FIT モジュールパッケージ(R01AN4930)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、Mesh FIT モジュール(R01AN4930)に含まれます。

Bluetooth Low Energy プロトコルスタックは、BLE FIT モジュール(R01AN4860)に含まれます。

1.3　ファイル構成

Mesh FIT モジュール(r_mesh_rx23w)に含まれるファイル構成を以下に示します。

```
r_mesh_rx23w
   |   r_mesh_rx23w_if.h                          Mesh スタック API ヘッダファイル
   |   readme.txt                                 Mesh FIT モジュール情報ファイル
   |
   +---doc\
   |   |   blemesh_api.chm                        Mesh スタック API 仕様書
   |   +---en\
   |   |       r01an4930ej0130-rx23w-blemesh.pdf  Mesh FIT モジュール説明書(英)
   |   +---ja\
   |           r01an4930jj0130-rx23w-blemesh.pdf  Mesh FIT モジュール説明書(日)
   |
   +---json\                                      QE for BLE 向けサービス定義ファイル
   |       mesh_provisioning.service.json         Mesh Provisioning Service 定義
   |       mesh_proxy.service.json                Mesh Proxy Service 定義
   |
   +---lib\
   |       lib_ble_ms_ccrx.lib                    Mesh スタックライブラリ
   |
   +---src\
       +---bearer\                                Bluetooth ベアラー
       +---drivers\                               Mesh ドライバ
       +---include\                               Mesh スタックヘッダファイル
```

Mesh FIT モジュールが提供する機能を利用するには、Mesh FIT モジュールをプロジェクトに組み込む必要があります。本モジュールの組み込み方法は、本書の 4 章を参照してください。

1.4　API 仕様

Mesh FIT モジュールの機能を実行するには、Mesh FIT モジュールに含まれる Mesh スタックの API を利用します。Mesh スタック API の仕様は、Mesh Stack API 仕様書(doc\blemesh_api.chm)を参照してください。

1.5　API ヘッダファイル

Mesh FIT モジュールを利用するには、r_mesh_rx23w_if.h をインクルードしてください。Mesh スタックの API は src\include\に含まれる複数のヘッダファイルで定義されていますが、r_mesh_rx23w_if.h によって全てのヘッダファイルをインクルードすることができます。

2. 要件

Mesh FIT モジュールを使用したアプリケーション開発の要件について示します。

2.1 ハードウェア要件

ご使用になる MCU は下記のハードウェア機能をサポートしている必要があります。

- Bluetooth Low Energy (BLE)
- Compare Match Timer (CMT)
- 8-Bit Timer (TMR)
- E2 Data Flash

2.2 ソフトウェア要件

Mesh FIT モジュールは下記の FIT モジュールを必要とします。

- **r_bsp**: Board Support Package (BSP FIT モジュール バージョン 5.66 以降)
- **r_ble_rx23w**: Bluetooth Low Energy (BLE FIT モジュール バージョン 2.11 以降)
- **r_flash_rx**: Data Flash memory (フラッシュ FIT モジュール バージョン 4.60 以降)

BLE FIT モジュールは下記の FIT モジュールを必要とします。

- **r_lpc_rx**: Low Power Control (LPC FIT モジュール)
- **r_cmt_rx**: Compare Match Timer (CMT FIT モジュール バージョン 4.70 以降)　注
- **r_sci_rx**: Serial Communication Interface (SCI FIT モジュール)
- **r_byteq**: Byte Queues/Circular Buffers (BYTEQ FIT モジュール)
- **r_gpio_rx**: General Purpose I/O (GPIO FIT モジュール)
- **r_irq_rx**: Interrupt Request (IRQ FIT モジュール)

注:　BLE FIT モジュールは CMT2 と CMT3 を直接使用するため、CMT FIT モジュールは CMT0 と CMT1 のみ使用できます。

2.3 サポートするツールチェーン

Mesh FIT モジュールは下記のツールチェーンで動作を確認済みです。

- **統合開発環境**:　ルネサスエレクトロニクス製 e² studio 2022-10
- **コンパイラ**:　ルネサスエレクトロニクス製 C/C++ Compiler for RX Family (CC-RX) V2.08.01
- **エンディアン**:　リトルエンディアン
- **開発ボード**:　Target Board for RX23W (RTK5RX23W0C00000BJ)
  Target Board for RX23W module (RTK5RX23W0C01000BJ)
  Renesas Solution Starter Kit (RSSK) for RX23W (RTK5523W8AC00001BJ)

2.4　セクション

Mesh FIT モジュールに含まれる Mesh スタックは表 2-1 に示すセクション名で配置されます。

表 2-1　Mesh スタックのセクション名

プログラム領域の名称	Mesh スタックのセクション		
	名称	属性	アライメント数
プログラム領域	MESH_P	code	1byte
定数領域	MESH_C	romdata	4byte
	MESH_C_2	romdata	2byte
	MESH_C_1	romdata	1byte
初期化データ領域	MESH_D	romdata	4byte
	MESH_D_2	romdata	2byte
	MESH_D_1	romdata	1byte
	MESH_R	data	4byte
	MESH_R_2	data	2byte
	MESH_R_1	data	1byte
未初期化データ領域	MESH_B	data	4byte
	MESH_B_2	data	2byte
	MESH_B_1	data	1byte
switch 文分岐テーブル領域	MESH_W	romdata	4byte
	MESH_W_2	romdata	2byte
	MESH_W_1	romdata	1byte
リテラル領域	MESH_L	romdata	4byte

属性:　code　実行命令を格納
　　data　変更可能なデータを格納
　　romdata　固定データを格納

セクション仕様の詳細は「CC-RX コンパイラ ユーザーズマニュアル」(R20UT3248)の 6 章「セクション仕様」を参照してください。

Mesh FIT モジュールを使用するプログラムは、ROM の初期化データを RAM の初期化データセクションに転送する必要があります。初期化データの転送のための設定は 5.6.1 項を参照してください。

2.5　プログラムサイズ

表 2-2 に Mesh FIT モジュールのプログラムサイズを示します。なお使用されない変数や関数がある場合、リンカの最適化処理によって削除されるため、実際にリンクされる Mesh FIT モジュールの ROM サイズは減少します。また Mesh FIT モジュールの設定変更により、Mesh FIT モジュールが必要とする RAM サイズは変動します。

表 2-2　Mesh FIT モジュールの全プログラムサイズ

デバイス	コンパイラ	分類	サイズ
RX23W グループ	CC-RX V2.08.01	ROM	67,466byte
		RAM	9,974byte
条件			
Mesh FIT モジュール 　デフォルト設定 (r_mesh_rx23w\ref\r_mesh_rx23w_config_reference.h) コンパイルオプション 　最適化レベル　レベル 2 全体的に最適化を実施する (-optimize=2) 　最適化方法　コード・サイズ重視の最適化を実施する (-size) リンクオプション 　最適化方法　すべての最適化を行わない (-nooptimize)			

表 2-3 に Mesh FIT モジュールパッケージに含まれるデモプロジェクトのプログラムサイズを示します。デモプロジェクトの詳細は「RX23W グループ Bluetooth Mesh スタック 開発ガイド」(R01AN4875)を参照してください。

表 2-3　Mesh FIT モジュールパッケージに含まれるデモプロジェクトのプログラムサイズ

デバイス	コンパイラ	分類	サイズ
RX23W グループ	CC-RX V2.08.01	ROM	306,876byte (Mesh FIT モジュール 62,622byte)
		RAM	45,795byte (Mesh FIT モジュール　9,970byte)
条件			
プロジェクト 　Target Board for RX23W 向け Server Models プロジェクト (rsskrx23w_mesh_server) コンパイルオプション 　最適化レベル　レベル 2 全体的に最適化を実施する (-optimize=2) 　最適化方法　コード・サイズ重視の最適化を実施する (-size) リンクオプション 　最適化方法　一度も参照のない変数/関数を削除する (-optimize=symbol_delete)			

3. FIT モジュール設定

3.1 Mesh FIT モジュール

Mesh FIT モジュールには、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表 3-1 に示す設定マクロとして r_ble_rx23w_config.h で定義されます。

スマート・コンフィグレータを使用する場合は、GUI でこれらの設定マクロの値を設定でき、プロジェクトへの Mesh FIT モジュールの追加時に r_ble_rx23w_config.h に反映されます。

設定マクロの設定により Mesh FIT モジュールが必要とする RAM サイズが変化します。ファームウェアの使用する RAM サイズがデバイスの RAM サイズを超過した場合は、Mesh FIT モジュールの設定値を見直してください。

表 3-1 Mesh FIT モジュールの設定マクロ

設定マクロ	詳細
MESH_CFG_NUM_NETWORK_INTERFACES *デフォルト値: 2	Mesh ネットワークに使用するベアラー数 最小値: 1 最大値: (1 + BLE_CFG_RF_CONN_MAX) 最初のベアラーは ADV Bearer、残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。
MESH_CFG_NUM_PROVISIONING_INTERFACES *デフォルト値: 2	プロビジョニングに使用するベアラー数 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。
MESH_CFG_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。
MESH_CFG_NET_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。
MESH_CFG_NET_SEQNUM_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
MESH_CFG_MAX_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

MESH_CFG_MAX_DEV_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。
MESH_CFG_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
MESH_CFG_NET_SEQ_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。
MESH_CFG_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
MESH_CFG_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
MESH_CFG_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
MESH_CFG_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
MESH_CFG_PROXY_SUBNET_NETID_ADV_TIMEOUT *デフォルト値: 300	Proxy Advertisement with Network ID の送信間隔[msec] 最小値: 20
MESH_CFG_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
MESH_CFG_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
MESH_CFG_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
MESH_CFG_MAX_LPNS *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
MESH_CFG_REPLAY_CACHE_SIZE *デフォルト値: 10	Replay Protection Cache サイズ 最小値: 2
MESH_CFG_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

MESH_CFG_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
MESH_CFG_LTRN_SAR_CTX_MAX *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
MESH_CFG_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
MESH_CFG_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
MESH_CFG_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
MESH_CFG_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
MESH_CFG_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
MESH_CFG_FRIEND_MESSAGEQUEUE_SIZE *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
MESH_CFG_FRIEND_SUBSCRIPTION_LIST_SIZE *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
MESH_CFG_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
MESH_CFG_LPN_CLEAR_RETRY_COUNT *デフォルト値: 5	Low Power Node が Friend Clear メッセージを送信後、Friend Node から Friend Clear Confirmation メッセージを受信できなかった場合の最大再送回数 最小値: 1
MESH_CFG_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
MESH_CFG_ACCESS_ELEMENT_COUNT *デフォルト値: 4	エレメントの最大数 最小値: 1
MESH_CFG_ACCESS_MODEL_COUNT *デフォルト値: 20	モデルの最大数 最小値: 1
MESH_CFG_MAX_APPS *デフォルト値: 8	Application Key の最大数 最小値: 1
MESH_CFG_MAX_VIRTUAL_ADDRS *デフォルト値: 8	バーチャルアドレスの最大数 最小値: 1
MESH_CFG_MAX_NON_VIRTUAL_ADDRS *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
MESH_CFG_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
MESH_CFG_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

MESH_CFG_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
MESH_CFG_HEALTH_SERVER_MAX *デフォルト値: 2	Health Server Model の最大数 最小値: 1
MESH_CFG_LIGHT_LC_SERVER_MAX *デフォルト値: 1	Light LC Server Model の最大値 最小値: 1
MESH_CFG_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 カンパニーID は Assigned Numbers を参照
MESH_CFG_DEFAULT_PID *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
MESH_CFG_DEFAULT_VID *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
MESH_CFG_DATA_FLASH_BLOCK_ID *デフォルト値: 1	Mesh 情報を格納するデータフラッシュの先頭ブロック ID 最小値: 0 最大値: 7
MESH_CFG_DATA_FLASH_BLOCK_NUM *デフォルト値: 5	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

3.2　BSP FIT モジュール

BSP FIT モジュールの表 3-2 に示す設定マクロは、Mesh FIT モジュールが動作するために変更が必要です。スマート・コンフィグレータでの設定方法は 5.4.1 項を参照してください。

注:　Mesh FIT モジュールを使用する場合、本変更を必ず行ってください。

表 3-2　BSP FIT モジュールの変更が必要な設定マクロ

設定マクロ	デフォルト値	Mesh FIT 向け設定値
BSP_CFG_HEAP_BYTES	0x400	0x1000
BSP_CFG_CLOCK_SOURCE	4	1
BSP_CFG_USB_CLOCK_SOURCE	1	0
BSP_CFG_PCKB_DIV	2	1
BSP_CFG_FCK_DIV	2	1
BSP_CFG_CONFIGURATOR_SELECT	0	1

3.3　BLE FIT モジュール

BLE FIT モジュールの表 3-3 に示す設定マクロは、使用するリソースを削減するために変更します。スマート・コンフィグレータでの設定方法は 5.4.2 項を参照してください。

表 3-3　BLE FIT モジュールの変更すべき設定マクロ

設定マクロ	デフォルト値	Mesh FIT 向け設定値
BLE_CFG_RF_CONN_MAX	7	1
BLE_CFG_RF_ADV_DATA_MAX	1650	31
BLE_CFG_RF_ADV_SET_MAX	4	1
BLE_CFG_RF_SYNC_SET_MAX	2	1
BLE_CFG_CMD_LINE_CH	1	8
BLE_CFG_BOARD_TYPE	0	1: Target Board for RX23W, Target Board for RX23W module 2: RSSK for RX23W

4. FIT モジュールの追加方法

各 FIT モジュールはプロジェクト毎に追加する必要があります。推奨される FIT モジュールの追加方法は、**スマート・コンフィグレータ**を使用する下記の(1)または(2)です。

(1) e² studio 上で**スマート・コンフィグレータ**を使用して追加する場合
e² studio の**スマート・コンフィグレータ**を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は「RX スマート・コンフィグレータ ユーザーガイド e² studio 編」(R20AN0451)を参照してください。また 5 章も併せて参照してください。

(2) CS+上で**スマート・コンフィグレータ**を使用して追加する場合
CS+上で**スタンドアロン版スマート・コンフィグレータ**を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は「RX スマート・コンフィグレータ ユーザーガイド e² studio 編」(R20AN0451)を参照してください。

(3) e² studio 上で **FIT コンフィグレータ**を使用して追加する場合
e² studio の **FIT コンフィグレータ**を使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology」(R01AN1723)を参照してください。

(4) CS+上で手動追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加することができます。詳細は「RX ファミリ CS+に組み込む方法 Firmware Integration Technology」(R01AN1826)を参照してください。

5.　使用方法

e² studio 上でスマート・コンフィグレータを使用して、新規プロジェクトに Mesh FIT モジュールを追加する方法について説明します。

5.1　新規プロジェクトの作成

[ファイル]メニューから[新規]→[C/C++ Project]を選択します。[Templates for New C/C++ Project]ダイアログの左側で[Renesas RX]、右側で[Renesas CC-RX C/C++ Executable Project]を選択し、[次へ]ボタンをクリックします。

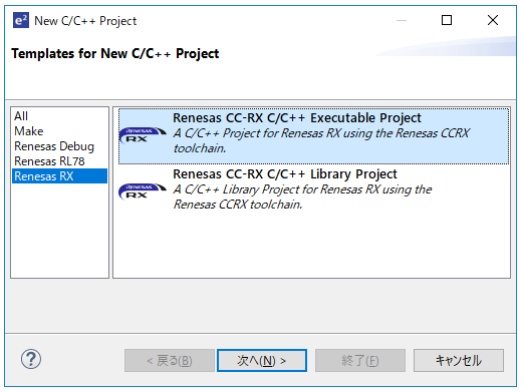

図 5-1　プロジェクトテンプレートの選択

[New Renesas CC-RX Executable Project]ダイアログでプロジェクト名を入力し、[次へ]ボタンをクリックします。

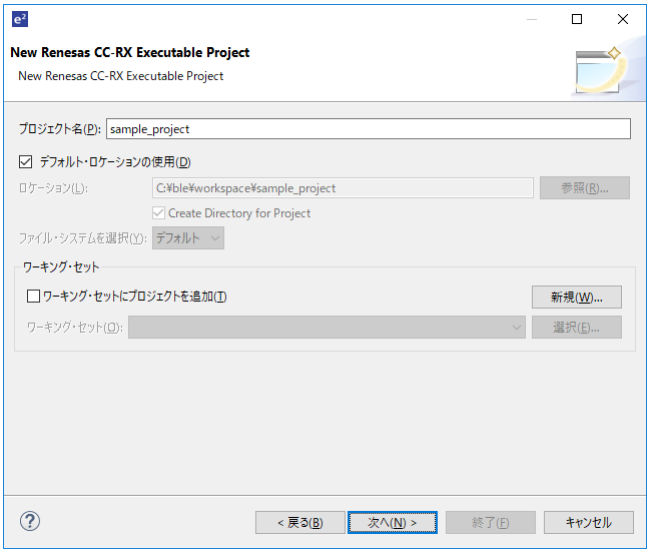

図 5-2　新規プロジェクトの設定

Device Setting でエンディアンを[Little]に設定します。またターゲット・デバイスで使用する RX23W のデバイス型名を選択し、[次へ]ボタンをクリックします。

Target Board for RX23W を使用する場合は"R5F523W8AxNG"を選択します。Target Board for RX23W module を使用する場合は"R5F523W8CxLN"を選択します。RSSK for RX23W を使用する場合、RSSK の型名が"RTK5523W8AC00001BJ"であれば"R5F523W8AxBL"、"RTK5523W8BC00001BJ"であれば"R5F523W8BxBL"を選択します。

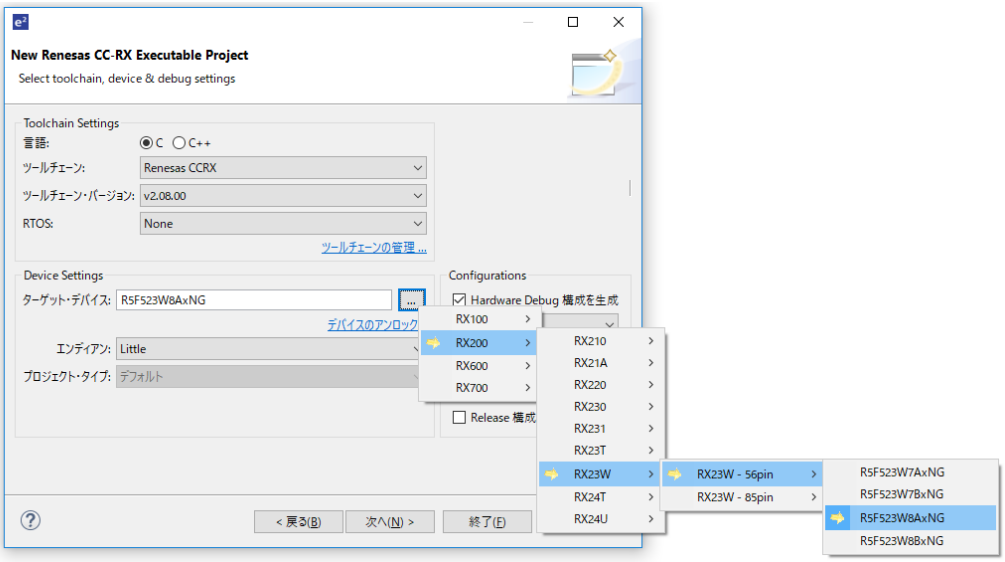

図 5-3　ツールチェーン、デバイス、デバッグの設定

[コーディング・アシストツールの選択]ダイアログで[スマート・コンフィグレータを使用する]にチェックを入れ、[終了]ボタンをクリックします。

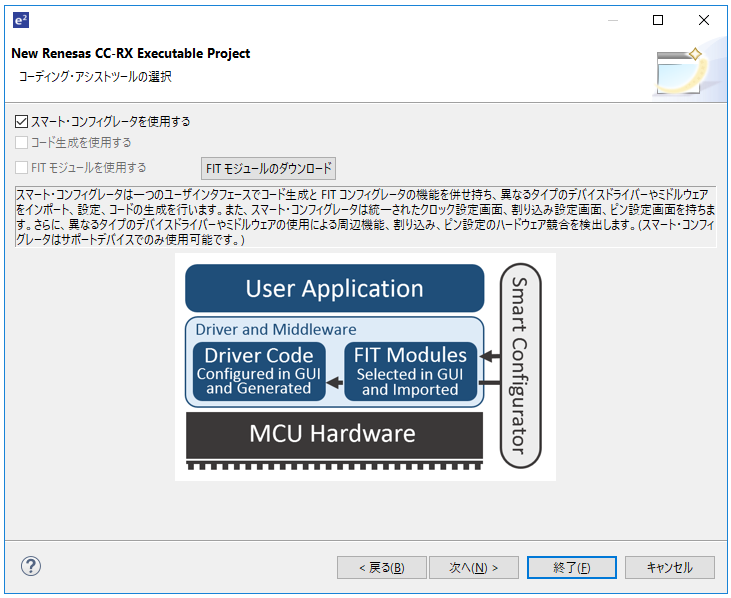

図 5-4　コーディング・アシストの選択

e² studio に新規のプロジェクトが作成されます。またスマート・コンフィグレータはプロジェクトに含まれる{プロジェクト名}.scfg をクリックすることで使用できます。

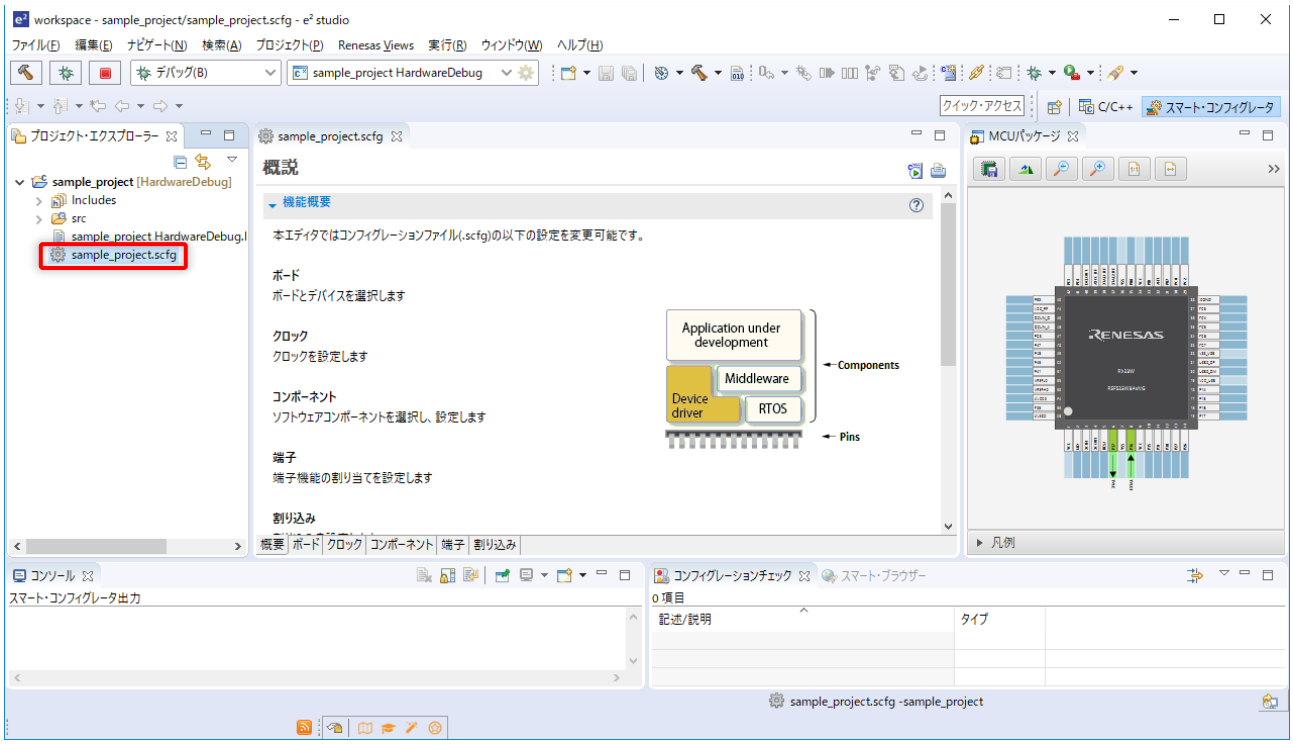

図 5-5　新規プロジェクトの作成完了

5.2　クロックの設定

スマート・コンフィグレータの[クロック]タブで、クロックの選択と周波数を設定します。Mesh FIT モジュールを使用する場合、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 B(PCLKB):　8MHz 以上

BLE FIT モジュールに含まれる Bluetooth Low Energy プロトコルスタックは、ICLK と PCLKB 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 PCLKB の周波数が 32MHz となるように設定することを推奨します。

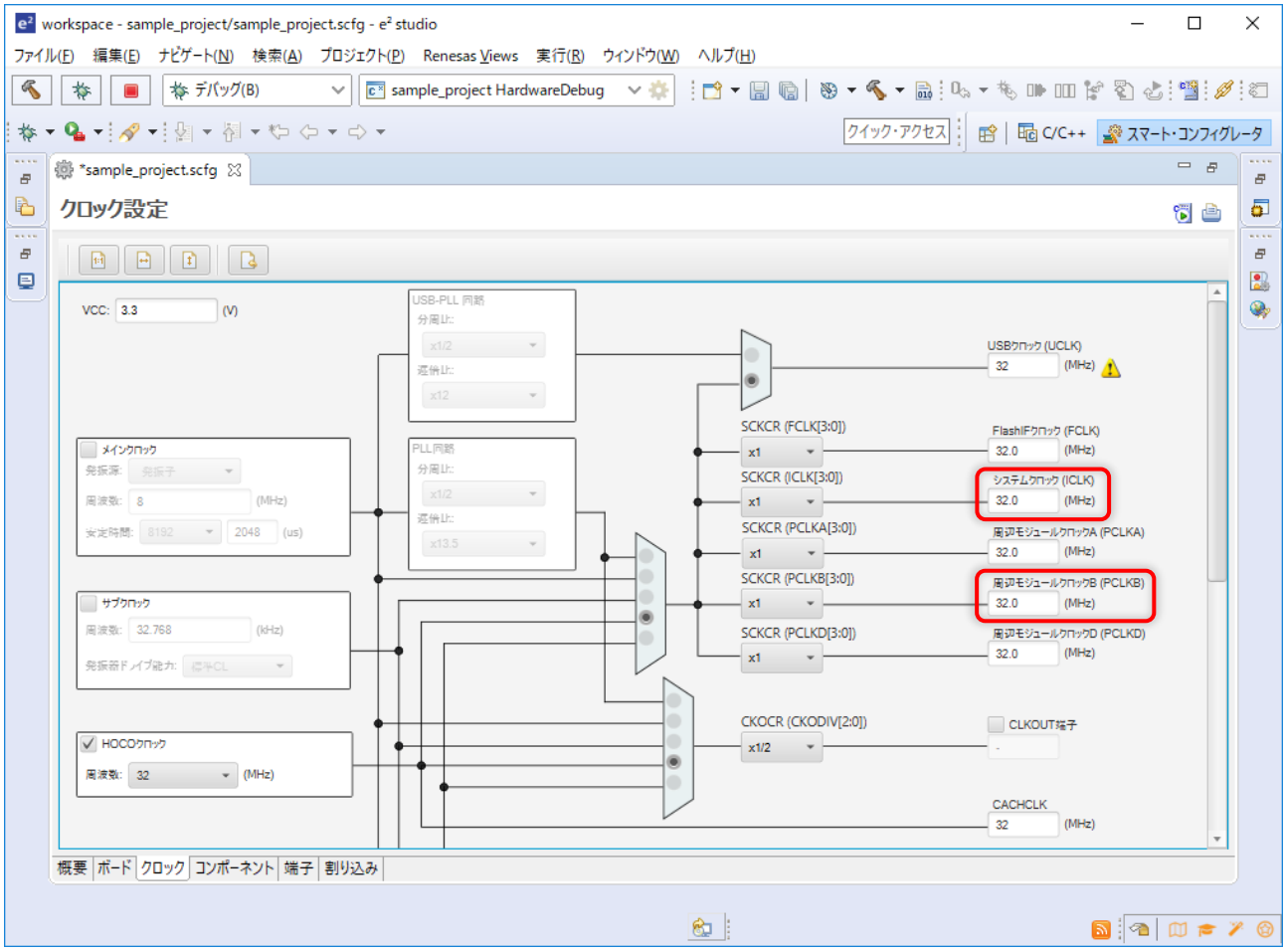

図 5-6　クロックの設定

5.3　コンポーネントの追加

スマート・コンフィグレータの[コンポーネント]タブで、Mesh FIT モジュールとその他の必要な FIT モジュールを追加します。必要な FIT モジュールは 2.2 節を参照してください。

[コンポーネントの追加]ボタン をクリックします。[ソフトウェアコンポーネントの選択]ダイアログで、必要な FIT モジュール(r_mesh_rx23w, r_bsp, r_ble_rx23w, r_byteq, r_cmt_rx, r_flash_rx, r_gpio_rx, r_irq_rx, r_lpc_rx, r_sci_rx)を選択し、[終了]ボタンをクリックします。

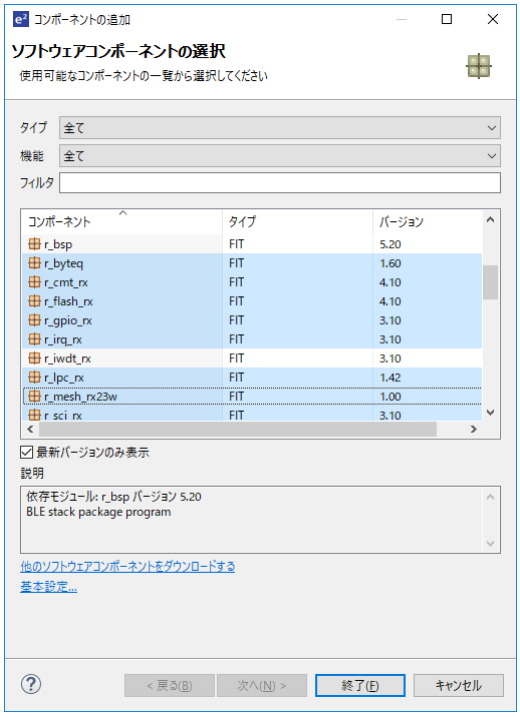

図 5-7　ソフトウェアコンポーネントの選択

注:　必要な FIT モジュールが見つからない場合は、[他のソフトウェアコンポーネントをダウンロードする]をクリックし、[FIT モジュールのダウンロード]ダイアログの手順に従って FIT モジュールをダウンロードしてください。

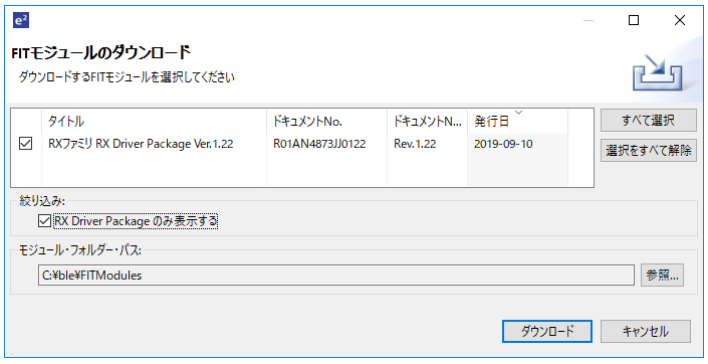

図 5-8　FIT モジュールのダウンロード

注:　ダウンロードした FIT モジュールが見つからない場合は、[基本設定…]をクリックし、[設定]ダイアログの[すべての FIT モジュールを表示する]にチェックを入れてください。

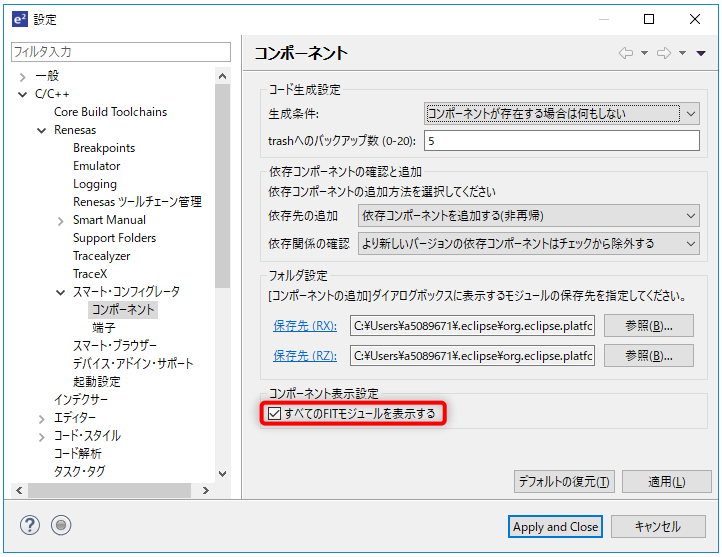

図 5-9　すべての FIT モジュールを表示

選択した FIT モジュールが[コンポーネント]タブに追加されます。

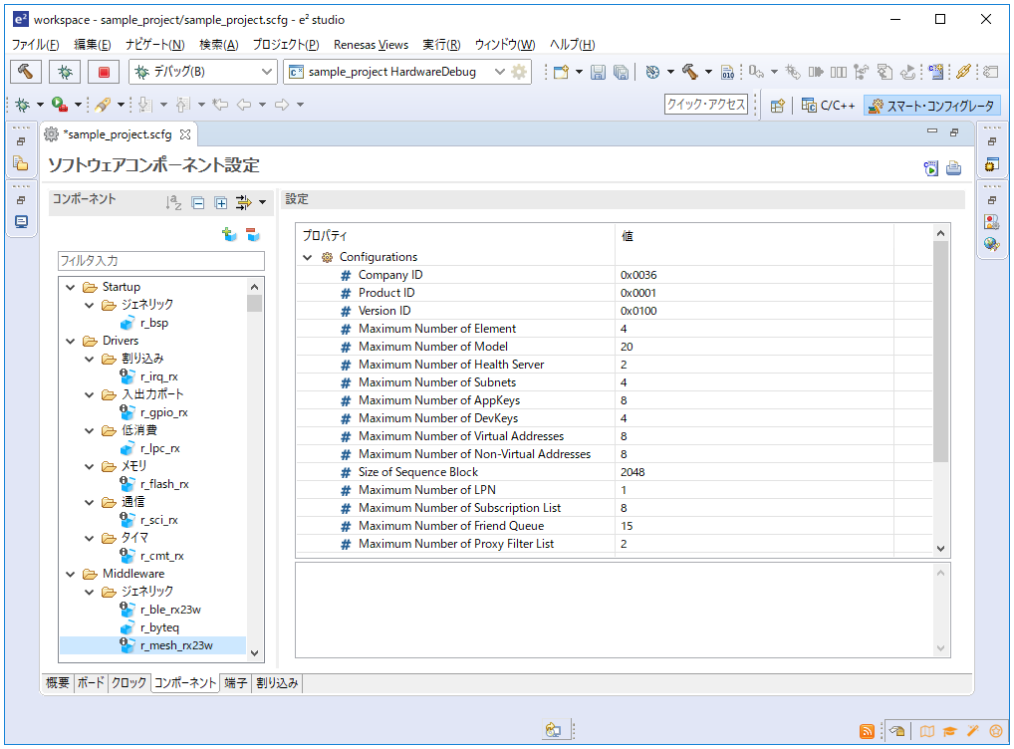

図 5-10　ソフトウェアコンポーネント

5.4　コンポーネントの設定

スマート・コンフィグレータの[コンポーネント]タブで、Mesh FIT モジュールが必要とする FIT モジュールの設定を変更します。

5.4.1　r_bsp

Mesh FIT モジュールの動作に必要なヒープ領域を確保するため、[コンポーネント]タブで[r_bsp]を選択して[Heap size]を"0x1000"に設定します。

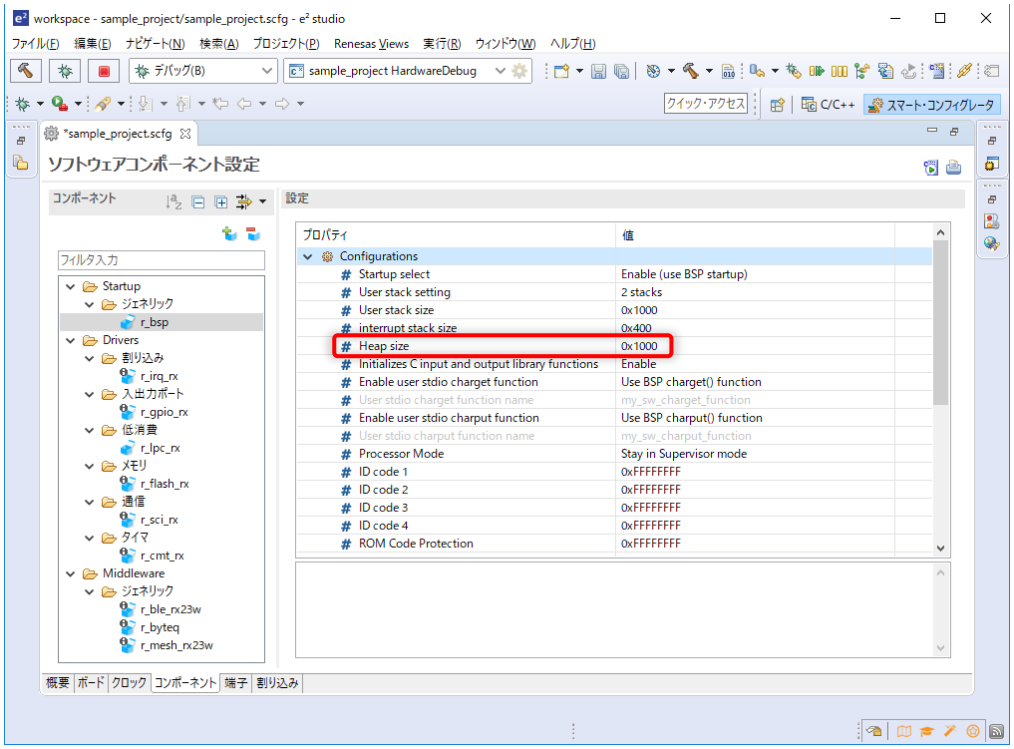

図 5-11　r_bsp 設定

注:　FIT モジュールバージョンの依存関係でワーニングが発生する場合、コンポーネントタブに表示された FIT モジュールの右クリックメニューで FIT モジュールバージョンを変更してください。

5.4.2　r_ble_rx23w

BLE FIT モジュールが使用するリソースを削減するため、[コンポーネント]タブで[r_ble_rx23w]を選択して[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。

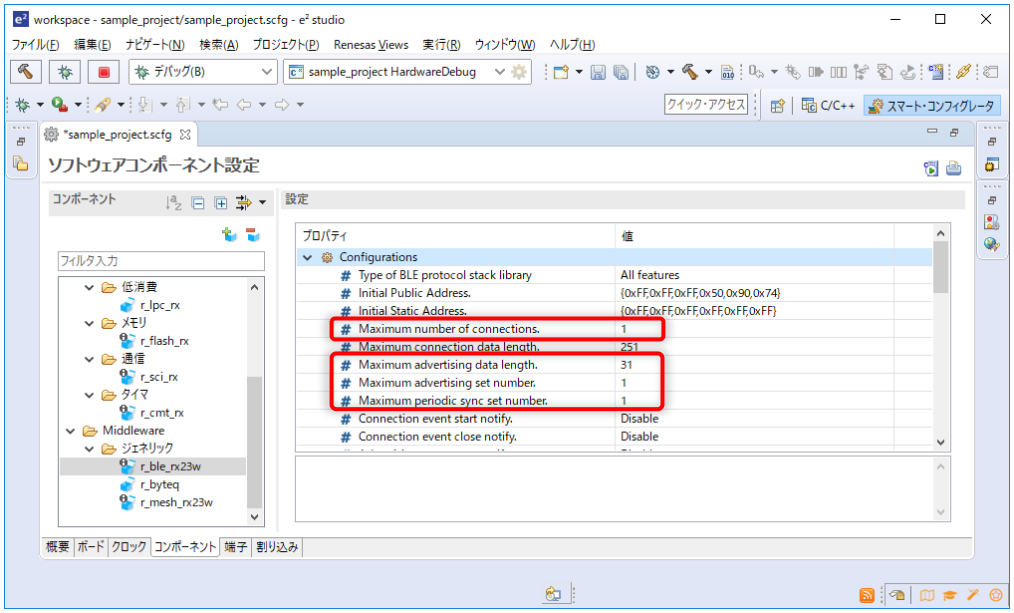

図 5-12　r_ble_rx23w 設定(1)

Target Board for RX23W / Target Board for RX23W module または RSSK for RX23W を使用する場合は、[Enabled/Disabled command line function.]を"Enable"に設定し、[SCI CH for command line function]を"8"に設定します。[Enabled/Disabled board LED and Switch control support.]を"Enable"に設定し、[Board Type]を"Target Board"または"RSSK"に設定します。

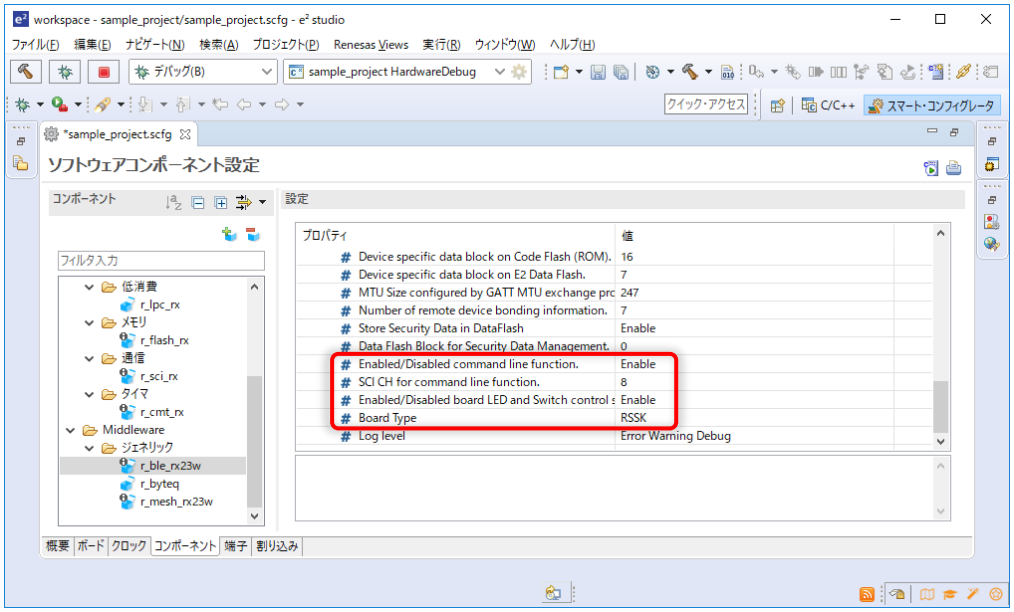

図 5-13　r_ble_rx23w 設定(2)

注:　[Type of BLE protocol stack library]には"All features"または"Balance"を選択することができます。"Compact"は Scan 動作に対応していないため選択できません。

5.4.3　r_sci_rx

Target Board for RX23W / Target Board for RX23W module または RSSK for RX23W でシリアル通信機能を使用する場合は、[コンポーネント]タブで[r_sci_rx]を選択して[Include software support for channel 8]を"Include"に設定します。またその他の SCI チャネルを"Not"に設定します。

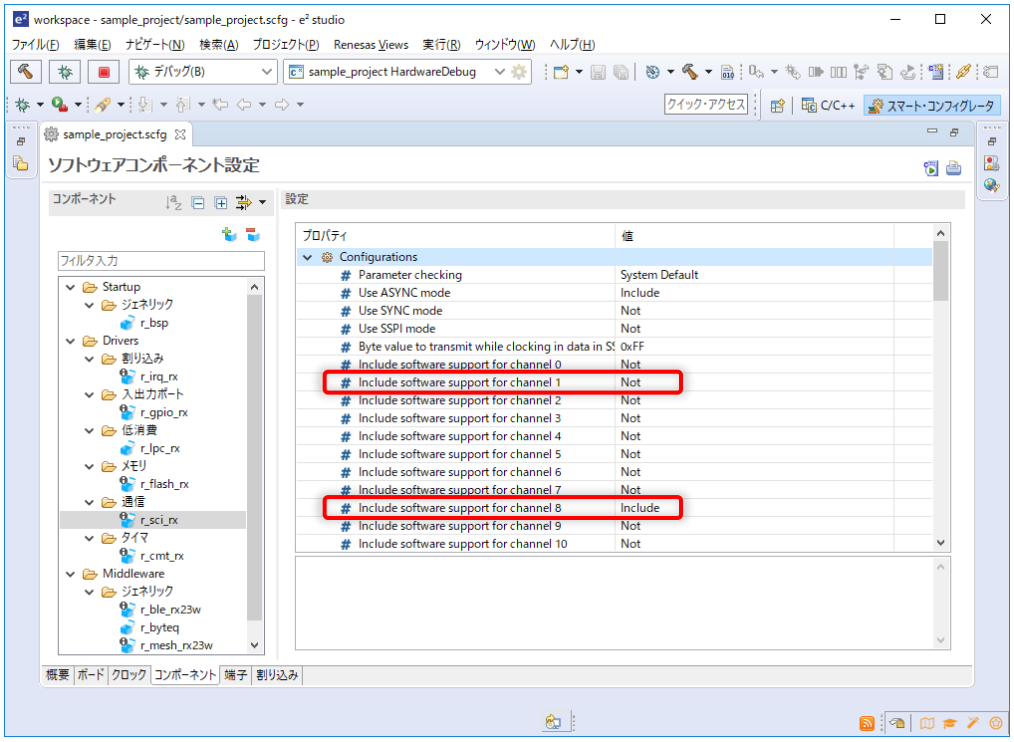

図 5-14　r_sci_rx 設定(1)

さらに[リソース]→[SCI8]を選択後、[RXD8/SMISO8/SSCL8 端子]と[TXD8/SMOSI8/SSDA8 端子]を"使用する"に設定します。

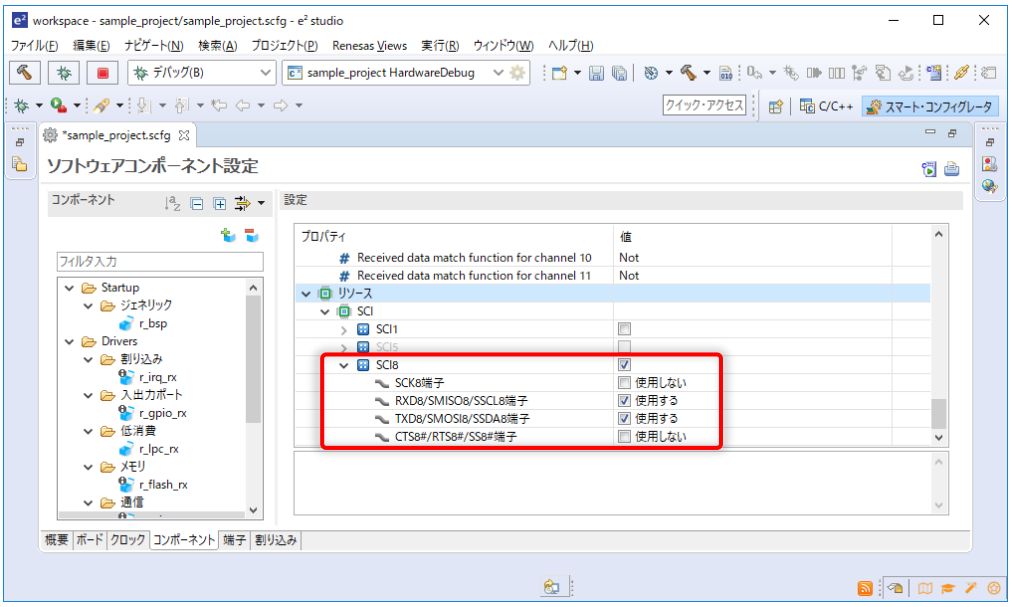

図 5-15　r_sci_rx 設定(2)

BLE FIT モジュールは RX23W の開発ボード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)を提供します。

本機能を使用する場合は、[Transmit end interrupt]を"Enable"に設定します。また本機能を Target Board for RX23W / Target Board for RX23W module または RSSK for RX23W で使用する場合は、[ASYNC mode TX queue buffer size for channel 8]を"180"に設定します。

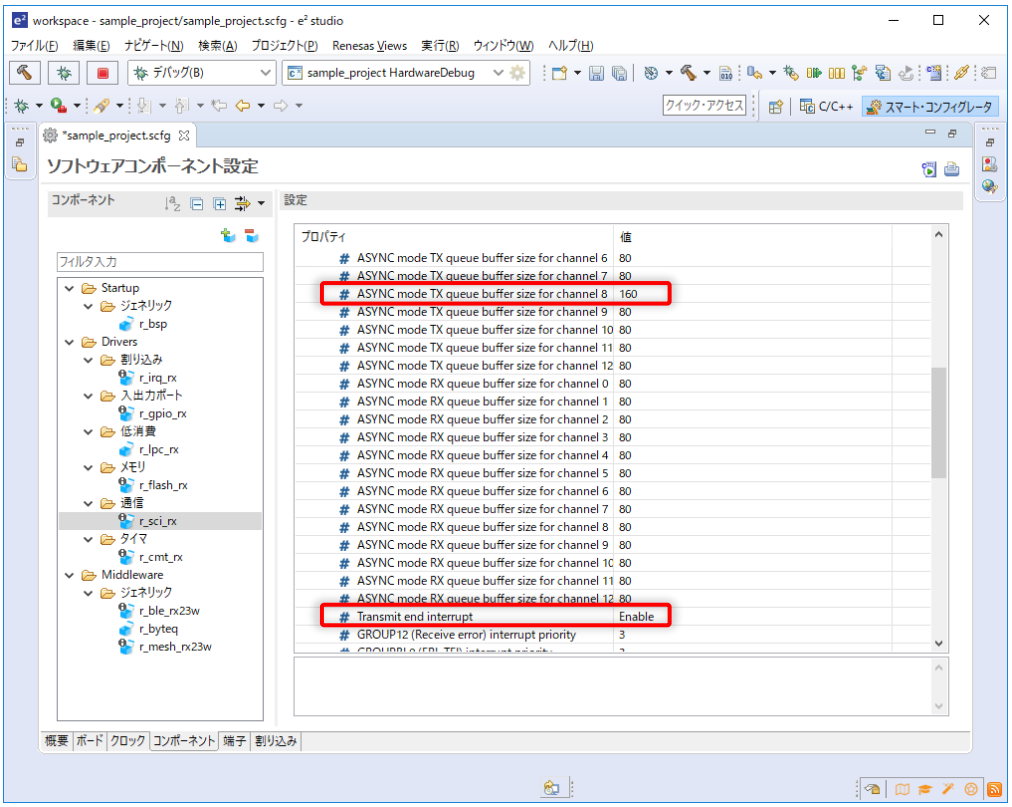

図 5-16　r_sci_rx 設定(3)

5.4.4　r_irq_rx

Target Board for RX23W or Target Board for RX23W module に実装されたスイッチを使用する場合は、[コンポーネント]タブで[r_irq_rx]を選択し、[IRQ5 端子]を"使用する"に設定します。

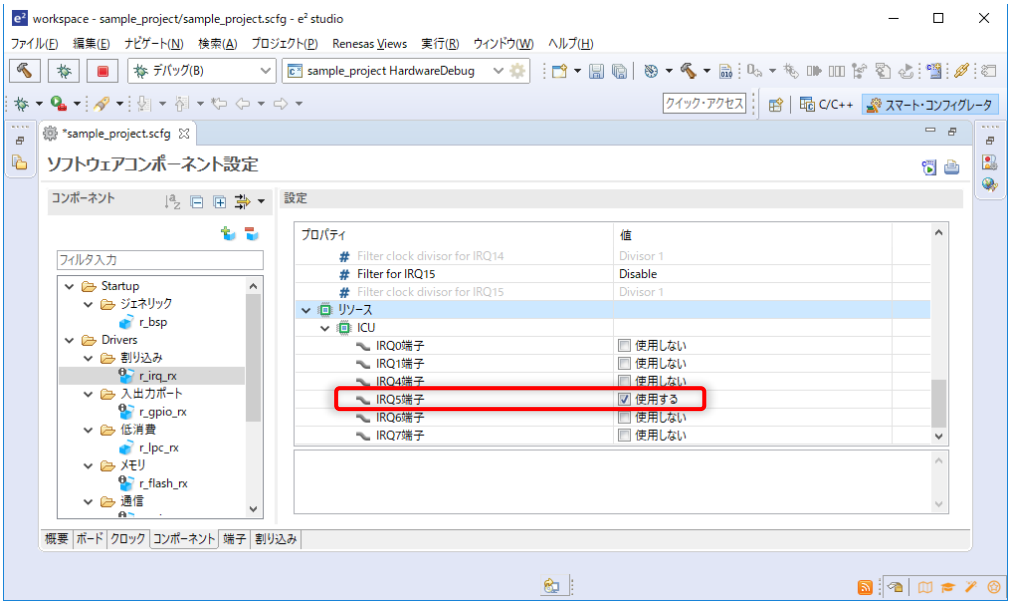

図 5-17　Target Board 向け r_irq_rx 設定(1)

さらに[Filter for IRQ5]を"Enable"に設定し、[Filter clock divisor for IRQ5]を"Divisor 1"に設定します。

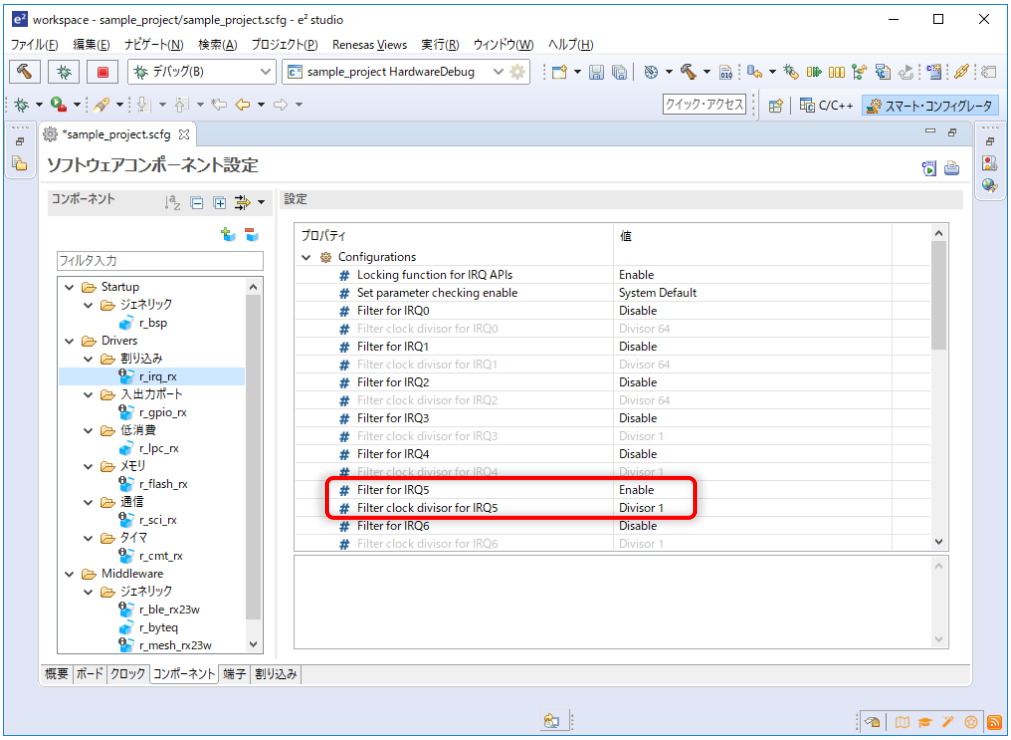

図 5-18　Target Board 向け r_irq_rx 設定(2)

RSSK for RX23W に実装されたスイッチを使用する場合は、[コンポーネント]タブで[r_irq_rx]を選択し、[IRQ0 端子]と[IRQ1 端子]を"使用する"に設定します。

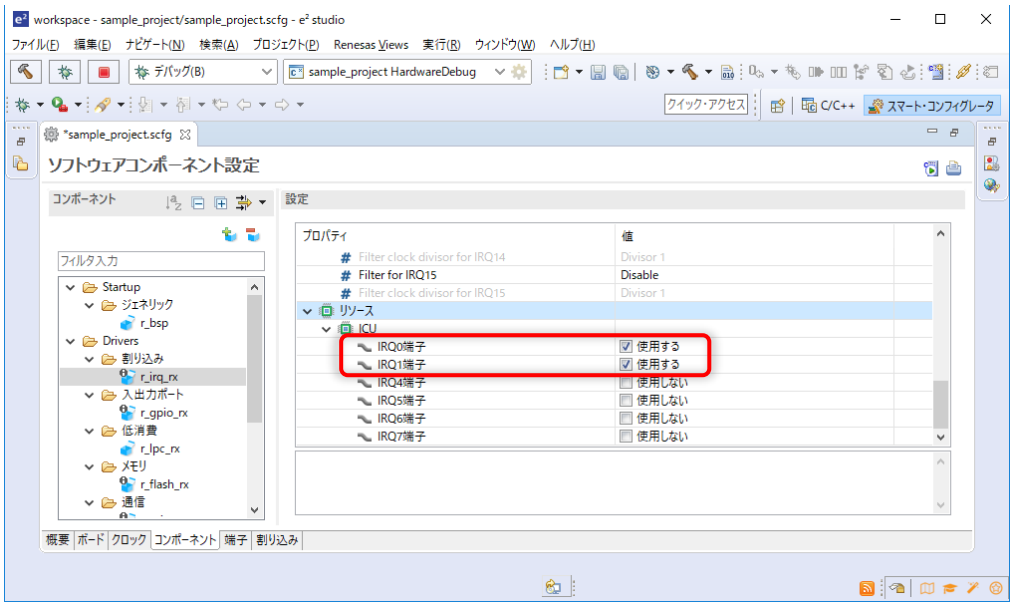

図 5-19　RSSK 向け r_irq_rx 設定(1)

さらに[Filter for IRQ0]と[Filter for IRQ1]を"Enable"に設定し、[Filter clock divisor for IRQ0]と[Filter clock divisor for IRQ1]を"Divisor 1"に設定します。

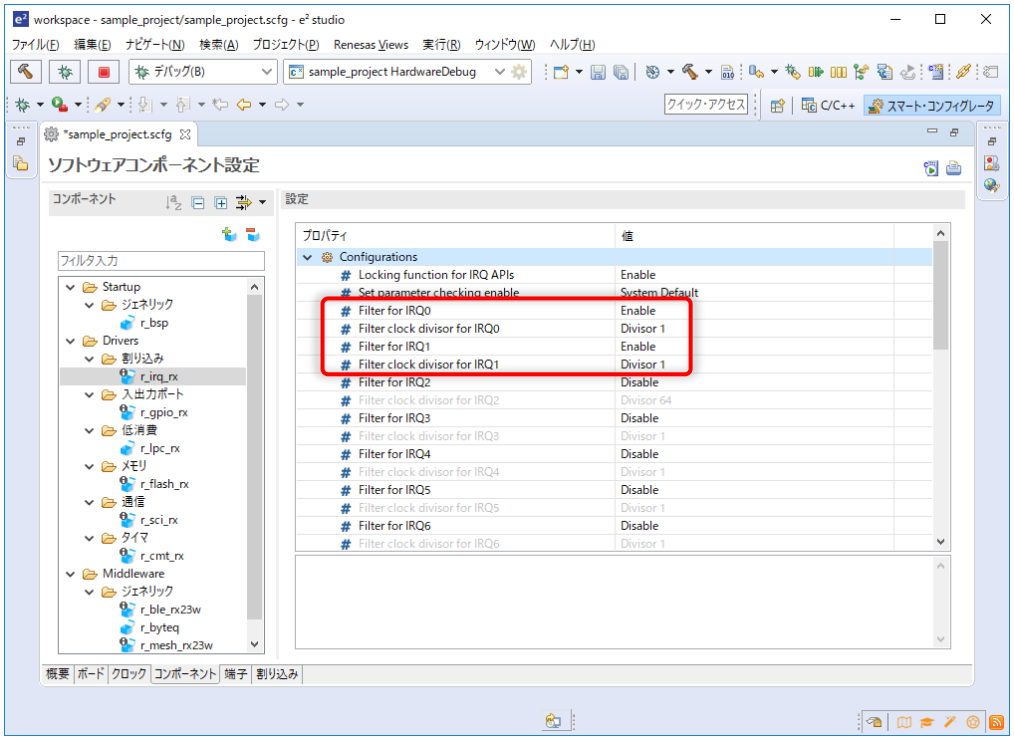

図 5-20　RSSK 向け r_irq_rx 設定(2)

5.5　コードの生成

スマート・コンフィグレータの[コンポーネント]タブで、[コードの生成]ボタン  をクリックします。プロジェクトの smc_gen フォルダに FIT モジュールのコードが生成されます。

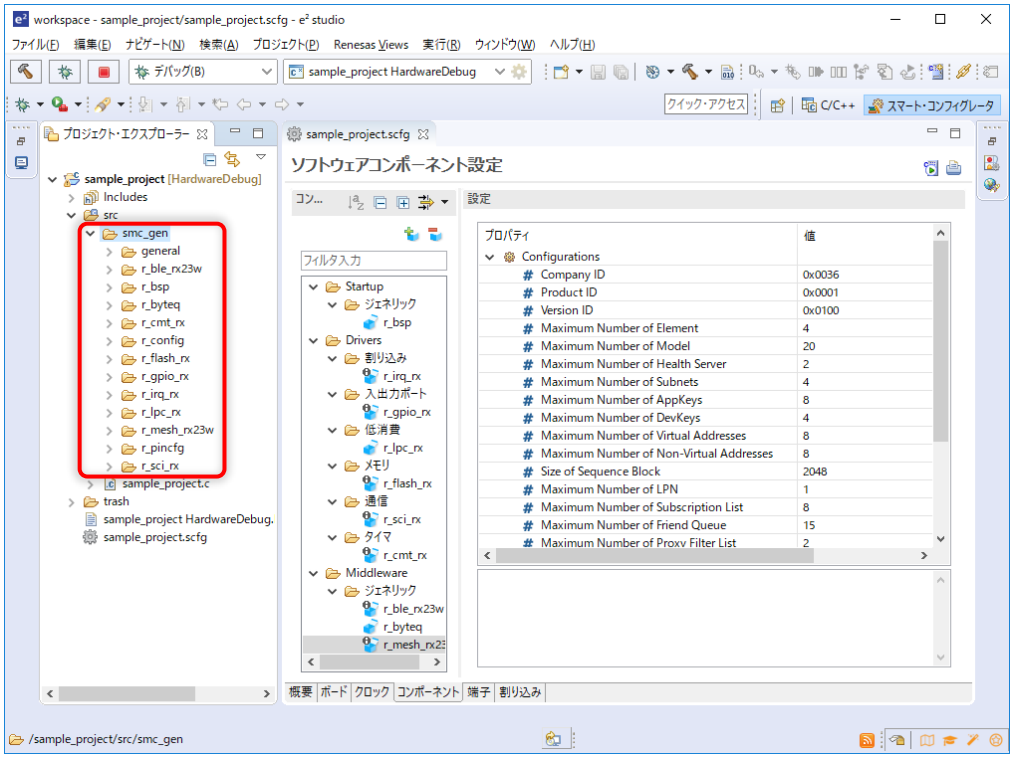

図 5-21　コードの生成結果

5.6　リンカ設定

5.6.1　セクション

Mesh FIT モジュール、BLE FIT モジュールを使用するプロジェクトでは、リンクオプションに各モジュールのセクションを追加し、ROM の初期化データを RAM の初期化データセクションに転送するための設定を行う必要があります。

(1)　セクションの追加

[プロジェクト]メニューで[プロパティ]を選択します。[プロパティ]ダイアログの左側で[C/C++ ビルド]→[設定]を選択後、右側の[ツール設定]タブで[Linker]→[セクション]を選択します。[...]ボタンをクリックすると表示されるセクション・ビューアーで、下記のセクションを追加します。

[RAM]

BLE FIT モジュールセクション　BLE_B*, BLE_R*

Mesh FIT モジュールセクション　MESH_B*, MESH_R*

[ROM]

BLE FIT モジュールセクション　BLE_C*, BLE_D*, BLE_W*, BLE_L*, BLE_P*

Mesh FIT モジュールセクション　MESH_C*, MESH_D*, MESH_W*, MESH_L*, MESH_P*

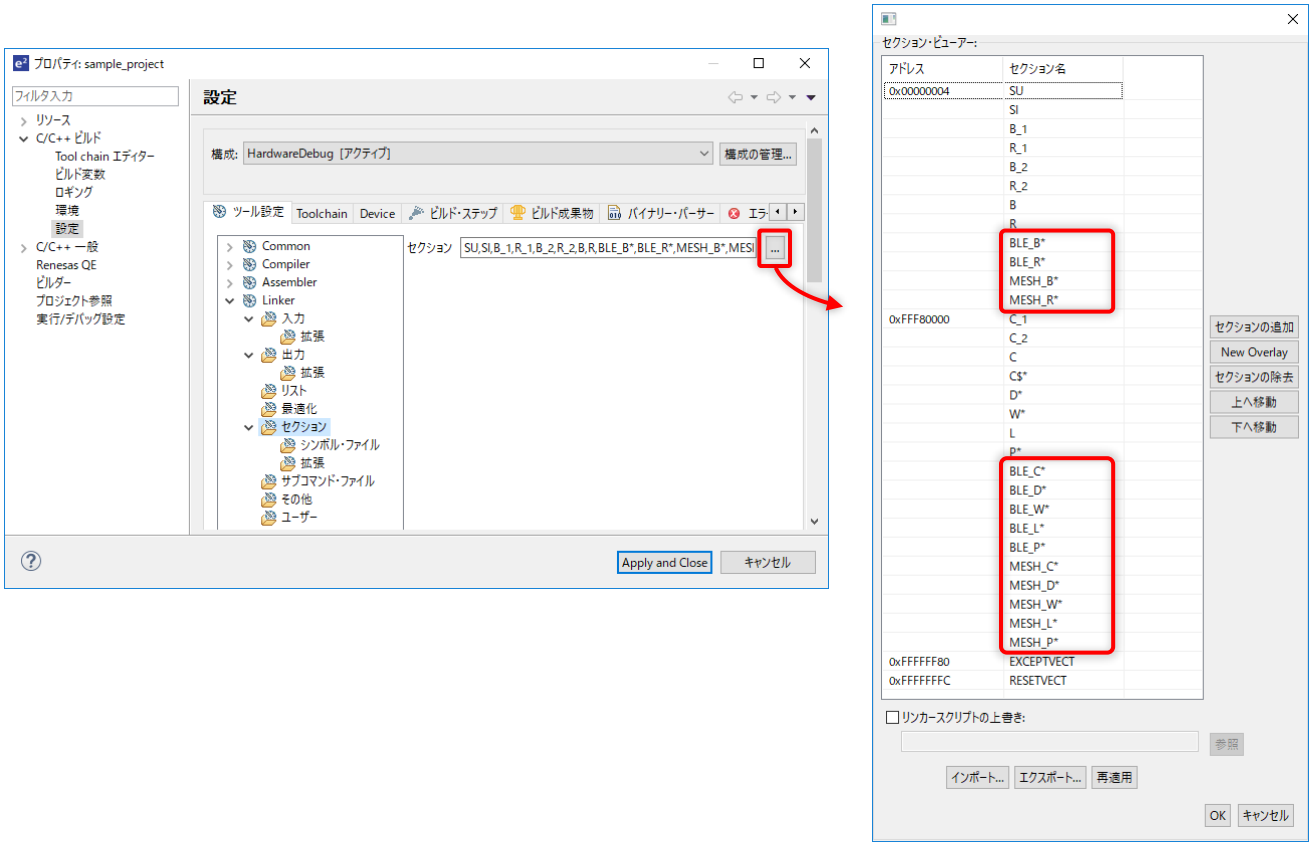

図 5-22　セクションの追加

(2) ROM から RAM へのマッピング

[プロパティ]ダイアログの[ツール設定]タブで[Linker]→[シンボル・ファイル]を選択します。[ROM から RAM へマップするセクション]に下記を追加します。

BLE_D=BLE_R

BLE_D_1=BLE_R_1

BLE_D_2=BLE_R_2

MESH_D=MESH_R

MESH_D_1=MESH_R_1

MESH_D_2=MESH_R_2

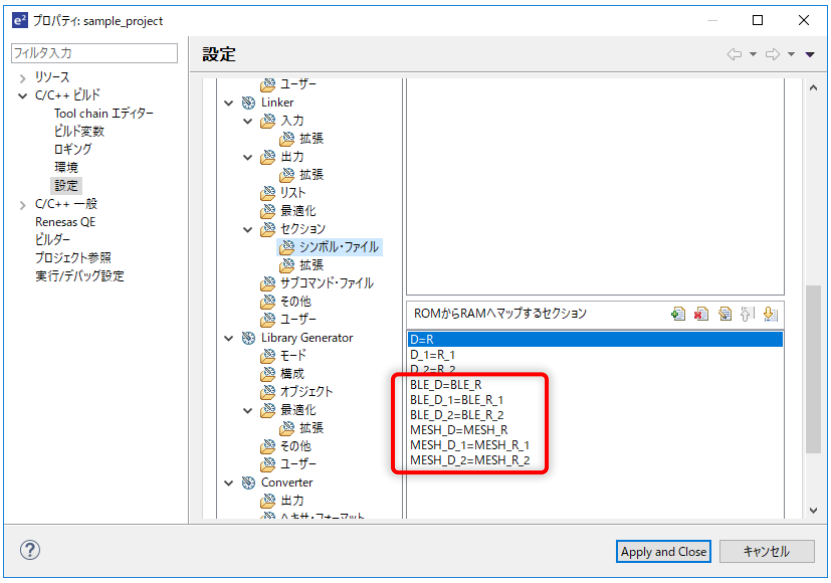

図 5-23　ROM から RAM へマップするセクションの設定

5.6.2　ライブラリ

Mesh FIT モジュールの Mesh スタックライブラリ、BLE FIT モジュールの Bluetooth Low Energy プロトコルスタックライブラリをリンクオプションに追加する必要があります。

(1)　ライブラリの追加

[プロパティ]ダイアログの[ツール設定]タブで[Linker]→[入力]を選択します。[リンクするリロケータブル・ファイル、ライブラリ・ファイルおよびバイナリ・ファイル]に下記が追加されていることを確認します。

"${workspace_loc:/${ProjName}/src/smc_gen/r_mesh_rx23w/lib/lib_ble_ms_ccrx.lib}"

"${workspace_loc:/${ProjName}/src/smc_gen/r_ble_rx23w/lib/lib_ble_ps_ccrx.lib}"

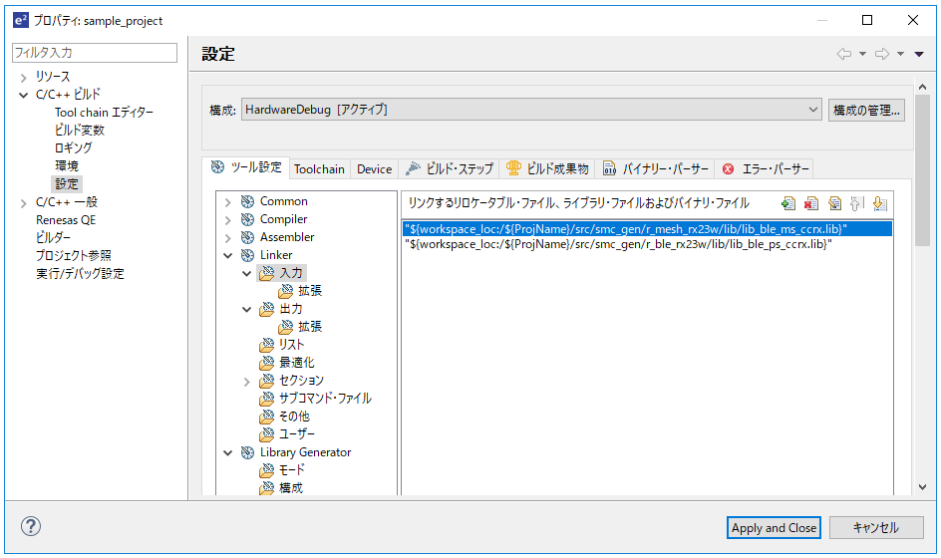

図 5-24　ライブラリ・ファイルの設定

(2)　ビルド前コマンドの追加

BLE FIT モジュールの Bluetooth Low Energy プロトコルスタックライブラリを使用するため、[プロパティ]ダイアログの[ビルド・ステップ]タブで、[ビルド前のステップ]→[コマンド]に下記のコマンドを追加します。

..\src\smc_gen\r_ble_rx23w\lib\ble_fit_lib_selector.bat

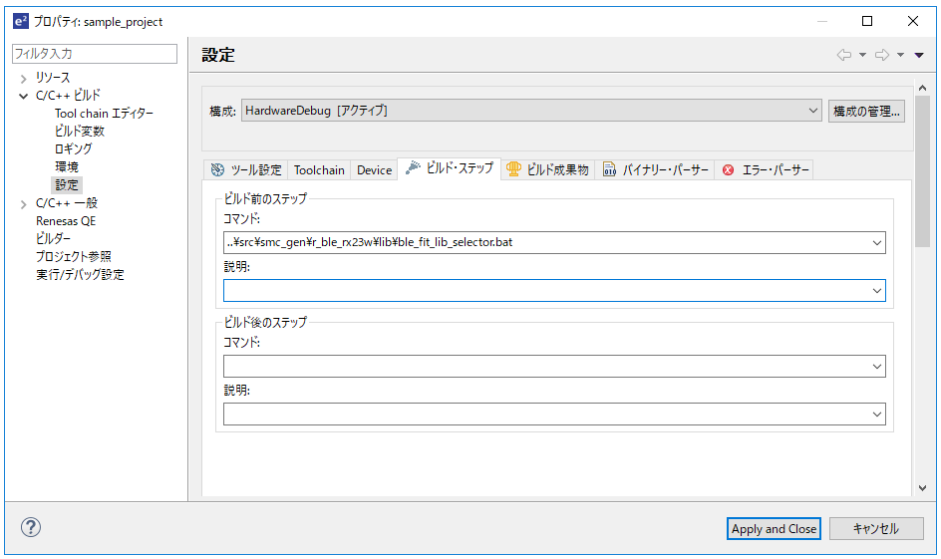

図 5-25　ビルド前コマンドの追加

リンカ設定の完了後は、[プロパティ]ダイアログの[Apply and Close]ボタンをクリックします。

5.7　デバッグ設定

[実行]メニューの[デバッグの構成]を選択します。[Renesas GDB Hardware Debugging]で{プロジェクト名} HardwareDebug を選択し、RX23W でのソフトウェアデバッグに必要な設定を行います。

5.7.1　デバッガの接続

[Debug hardware]で使用するデバッガを選択します。Target Board for RX23W / Target Board for RX23W module のオンボードエミュレータを使用する場合は"E2 Lite (RX)"を選択してください。

[Target Device]で使用するデバイスを選択します。Target Board for RX23W / Target Board for RX23W module または RSSK for RX23W を使用する場合は、"RX"→"RX23W"→"R5F523W8"を選択してください。

Target Board for RX23W / Target Board for RX23W module または RSSK for RX23W を使用する場合は、[クロック]→[メイン・クロック・ソース]を"HOCO"に設定します。また[電源]→[エミュレーターから電源を供給する]に"いいえ"を設定します。

注:　Target Board for RX23W は出荷時、RX23W の ID コードプロテクト機能が有効になっています。ファームウェアの書き込み時に「id コードの設定に失敗しました」と表示される場合は、[Debugger]タブの[Connection Settings]タブで[フラッシュ]→[ID コード]に”45FFFFFFFFFFFFFFFFFFFFFFFFFFFF”を設定してください。

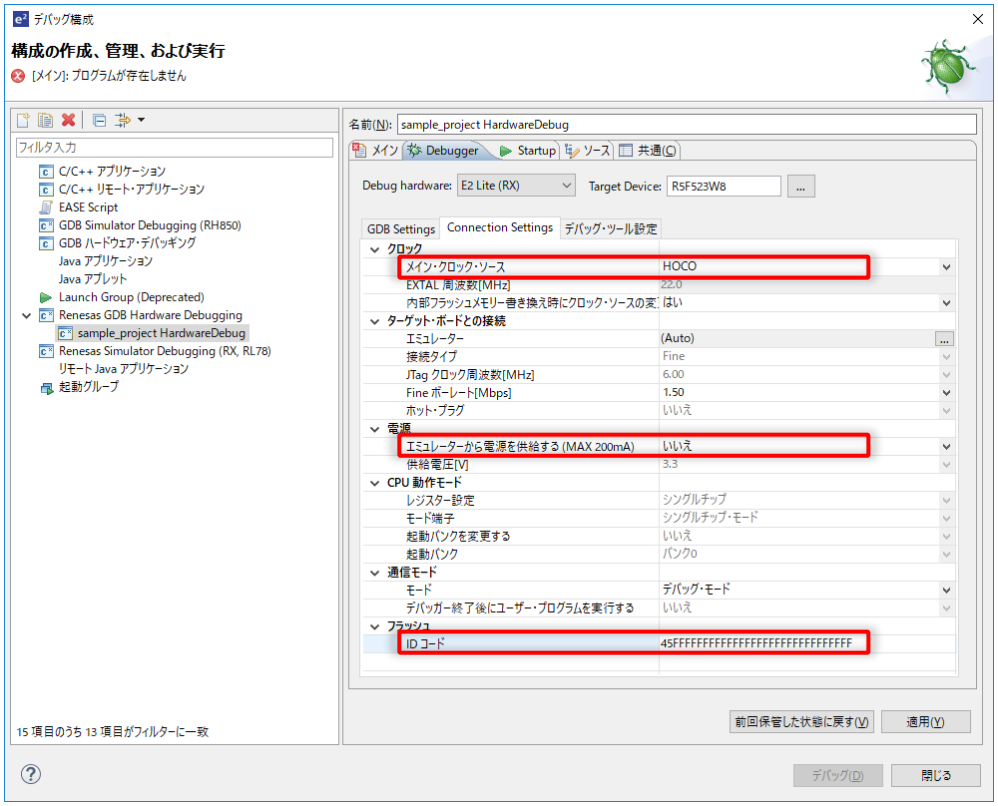

図 5-26　デバッガ接続の設定

5.8　プロジェクトのビルド

プロジェクトをビルドするには、[プロジェクト]メニューの[プロジェクトのビルド]を選択します。[コンソール]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。

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。

6. Mesh アプリケーションの実装方法

Mesh FIT モジュールを使用する Mesh アプリケーションの実装方法は「RX23W グループ Bluetooth Mesh スタック 開発ガイド」(R01AN4875)を参照してください。

また Mesh FIT モジュールパッケージ(R01AN4930)には Mesh FIT モジュールを使用したデモプロジェクトが含まれます。デモプロジェクトの実行方法は「RX23W グループ Bluetooth Mesh スタック スタートアップガイド」(R01AN4874)を参照してください。

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。

RX23W 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.

既知の制限事項

Bluetooth Mesh スタック	– Low Power Node として動作時、Mesh スタックが CMT(Compare Match Timer)を使用するため、MCU は低消費電力状態のソフトウェアスタンバイモードには遷移できません。
	– Sensor Setup Server として動作時、コールバック関数が受信した Sensor Cadence Set / Sensor Cadence Set Unacknowledged メッセージのパラメータを正しく通知しない場合があります。 Sensor Setup Server のコールバック関数の引数 `msg_raw` から Sensor Cadence Set / Sensor Cadence Set Unacknowledged メッセージのパラメータを参照してください。
Bluetooth ベアラー	– Proxy Server として動作時、複数の Proxy Client と接続した状態で、ペイロードサイズが大きいマルチキャストアドレス宛てのメッセージを送信した場合、メッセージが一部の Proxy Client に到達しない場合があります。 例) Proxy Server が 7 台の Proxy Client と接続中に 95 文字を含む Vendor Set メッセージ送信した場合、本メッセージは 1~2 台の Proxy Client にのみ到達する。
Mesh サンプルプログラム	– Generic OnOff Server または Vendor Server として動作時、SET メッセージまたは SET Unacknowledged メッセージを受信すると、モデルステートが変化していない場合も STATUS メッセージをパブリッシュします。

プログラム更新内容 (MESH FIT モジュール)

■ **Rev1.01**

Bluetooth ベアラー	**blebrr.c** - アドバタイジング送信タイミングにランダムディレイを追加

■ **Rev1.10**

Bluetooth Mesh スタック	**Common** - MESH FIT モジュール設定に下記のマクロを追加。詳細は 3.1 節を参照。 MESH_CFG_NET_SEQNUM_CACHE_SIZE - システムタイムタイマを登録する関数を追加。 MS_systemtime_init_pl() - メッシュモニタ機能を有効化する関数を追加。 MS_monitor_register_pl() **Provisioning** - 関数名にプレフィックス(MS_)を追加。 MS_prov_set_device_oob_pubkey_pl() MS_prov_set_static_oob_auth_pl() MS_prov_clear_device_oob_pubkey_pl() MS_prov_clear_static_oob_auth_pl() MS_mempool_init_pl() MS_storage_register_pl() - MS_prov_setup()の引数 role と bearer の実装順序を修正。 **Network** - Network メッセージキャッシュの実装を改善。 - Network レイヤーの送信状態を確認する関数を追加。 MS_net_register_tx_state_access() - IV Update の 96 時間チェックを無効化する関数を追加。 MS_access_set_iv_update_test_mode() - IV Index 更新時に Network レイヤーと Lower Transport レイヤーの送信状態チェックを追加。 - IV Update 完了時に SEQ 番号をリセットする処理を追加。 - MCU リセット時に Secure Network Beacon 送信を復帰する処理を追加。 - Proxy Advertisement を即時停止するように MS_proxy_server_adv_stop()の実装を修正。 - Proxy 接続の切断と再確立時の Proxy フィルタリスト管理の実装を修正。 **Lower Transport** - Lower Transport レイヤーの送信状態を確認する関数を追加。 MS_ltrn_register_tx_state_access() - マルチキャストアドレス向けセグメントメッセージ受信に対して Acknowledgement メッセージを送信しないよう修正。 **Access** - MS_access_cm_set_model_publication()に Publication アドレスの有効チェックを追加。

	− MS_access_cm_set_iv_index()が返却するエラーコードを追加。 ACCESS_IV_VAL_NOT_PERMITTED ACCESS_IV_UPDATE_TOO_SOON ACCESS_IV_INCORRECT_STATE ACCESS_IV_UPDATE_DEFERRED_IN_BUSY − MS_access_cm_reset()の実行で下記機能の設定をクリアするように修正。 Relay Proxy Friend Low Power Secure Network Beacon **Model** − SIGClient モデルコールバックの第 1 引数を MS_ACCESS_MODEL_HANDLE から MS_ACCESS_MODEL_REQ_MSG_CONTEXT 構造体に変更。 MS_CONFIG_MODEL_CB MS_HEALTH_CLIENT_CB MS_GENERIC_ONOFF_CLIENT_CB MS_GENERIC_LEVEL_CLIENT_CB MS_GENERIC_DEFAULT_TRANSITION_TIME_CLIENT_CB MS_GENERIC_POWER_ONOFF_CLIENT_CB MS_GENERIC_POWER_LEVEL_CLIENT_CB MS_GENERIC_BATTERY_CLIENT_CB MS_GENERIC_LOCATION_CLIENT_CB MS_GENERIC_PROPERTY_CLIENT_CB MS_SENSOR_CLIENT_CB MS_TIME_CLIENT_CB MS_SCENE_CLIENT_CB MS_SCHEDULER_CLIENT_CB MS_LIGHT_LIGHTNESS_CLIENT_CB MS_LIGHT_CTL_CLIENT_CB MS_LIGHT_HSL_CLIENT_CB MS_LIGHT_XYL_CLIENT_CB MS_LIGHT_LC_CLIENT_CB − 周期的なメッセージパブリケーション処理の登録チェックを追加。 − MS_config_server_init()にコールバック関数を登録するための引数 appl_cb を追加。
Bluetooth ベアラー	**blebrr.c/h** − Advertising 送信制御に使用するタイマを Bluetooth Low Energy プロトコルスタックの内部タイマからソフトウェアタイマ(R_BLE_TIMER)に変更。 − 送信キューサイズ(BLEBRR_QUEUE_SIZE)を 32 から 64 に変更。 − Advertising 送信回数(BLEBRR_ADVREPEAT_COUNT)を 5 回から 3 回に変更。 − Advertising 送信ランダムディレイ幅(BLEBRR_ADVREPEAT_RAND_DELAY)を 7msec から 10msec に変更。 − Advertising と Scan の動作状態を独立管理に変更。 − malloc()によるデータバッファの確保に失敗した場合、送信キューがリークする問題を解消。 **Blebrr_pl.c/h** − Static Random アドレスの取得処理を追加し、デフォルトのデバイスアドレスタイプ(BLEBRR_VS_ADDR_TYPE)を Public アドレスから Static Random アドレスに変更。 − GATT インタフェースを持つデバイスとの複数同時接続に対応するための実装を追加。 − Mesh GATT Client として接続確立後に Service Discovery を開始して、対向デバイスが Mesh GATT サービスを持つ場合、Notification を有効化する機能を追加。 − Mesh GATT サービスの UUID を含むデバイスをスキャンする関数を追加。 R_MS_BRR_Scan_GattBearer() − 関数名にプレフィックス(R_MS_BRR_)を追加。

	R_MS_BRR_Init() R_MS_BRR_Setup() R_MS_BRR_Register_GattIfaceCallback() R_MS_BRR_Set_GattMode() R_MS_BRR_Get_GattMode() R_MS_BRR_Disconnect() R_MS_BRR_Set_ScanRspData() R_MS_BRR_Create_Connection() R_MS_BRR_Discover_Service() R_MS_BRR_Config_Notification() – GATT インタフェースイベントを追加。 BLEBRR_GATT_IFACE_NOT_FOUND BLEBRR_GATT_IFACE_SCAN – GATT インタフェースコールバック(BLEBRR_GATT_IFACE_CB)に引数 conn_hdl と引数 peer_addr を追加。 **gatt_db_prov.c/h, gatt_db_proxy.c/h** – QE for BLE で生成された以下のサービスを持つ GATT データベースに置換。 GAP サービス GATT サービス Mesh Provisioning サービスまたは Mesh Proxy サービス **gatt_clients.c** – 対向デバイスの Mesh GATT サービスから Client Characteristic Configuration ディスクリプタ(CCCD)を探索する処理を追加。 – 対向デバイスの Mesh GATT サービスから取得した Attribute Handle の有効性を確認する処理を追加。
ドライバ	**mesh_resource.h** – メモリプールサイズマクロ(MESH_MEMPOOL_SIZE)の定義を更新。 – ストレージサイズマクロ(MESH_STORAGE_SIZE)を追加。 **Mesh_dataflash.h** – データフラッシュ有効化コンパイルスイッチ(DATAFLASH_EN)のデフォルト設定を 0 から 1 に変更。 **Mesh_systemtime.c/h** – 8 ビットタイマ(TMR)で 1msec 単位の 32bit システムタイムを生成するドライバを追加。

■ **Rev1.20**

Bluetooth Mesh スタック	**Common** – MS_CONFIG 構造体に下記の設定パラメータを追加。詳細は 3.1 節を参照。 MESH_CFG_NET_TX_COUNT MESH_CFG_NET_TX_INTERVAL_STEPS MESH_CFG_NET_RELAY_TX_COUNT MESH_CFG_NET_RELAY_TX_INTERVAL_STEPS MESH_CFG_CONFIG_SERVER_SNB_TIMEOUT MESH_CFG_PROXY_SUBNET_NETID_ADV_TIMEOUT MESH_CFG_PROXY_SUBNET_NODEID_ADV_TIMEOUT MESH_CFG_PROXY_NODEID_ADV_TIMEOUT MESH_CFG_FRND_POLL_RETRY_COUNT MESH_CFG_LTRN_RTX_TIMEOUT MESH_CFG_LTRN_RTX_COUNT MESH_CFG_LTRN_ACK_TIMEOUT MESH_CFG_LTRN_INCOMPLETE_TIMEOUT MESH_CFG_FRND_RECEIVE_WINDOW MESH_CFG_LPN_CLEAR_RETRY_TIMEOUT_INITIAL MESH_CFG_TRN_FRNDREQ_RETRY_TIMEOUT MESH_CFG_UNPROV_DEVICE_BEACON_TIMEOUT MESH_CFG_NET_TX_QUEUE_SIZE MESH_CFG_MAX_NUM_TRANSITION_TIMERS MESH_CFG_MAX_NUM_PERIODIC_STEP_TIMERS – Transition Time 値をミリ秒に変換する関数を追加。 MS_common_get_transition_time_in_ms() – Mesh モニタ機能のログ構造体を更新。(MS_logger.h) **Bearer** – Mesh ビーコンの送信シーケンスを変更して MS_brr_bcast_end()を削除。 **Provisioning** – Provisioning 中に無効な Public Key と Confirmation Value をリジェクトする処理を追加。(脆弱性 CVE-2020-26560) – Public Key を取得・設定する関数を追加。 MS_prov_access_local_public_key() **Network** – Network Message キャッシュを修正して、メッセージを意図せず破棄する問題を解消。 – Proxy Server が特定条件で Proxy Client にメッセージを送信できない問題を解消。 – Proxy Advertising with Node Identity 送信が MESH_CFG_NODEID_ADV_TIMEOUT マクロで指定した時間の経過後に停止しない問題を解消。 – Network Transmit ステートと Relay Retransmit ステートに従ってメッセージ送信するように実装を改善。 **Transport** – ローパワーノードが Friend Clear Confirm を受信できない場合に Friendship を再確立できない問題を解消。 – ローパワーノードが特定の条件でフレンドシップの切断時に Scan を再開しない問題を解消。 – ローパワーノードとフレンドノードの送受信処理を改善。

	Model − サーバーモデルの Status メッセージ送信関数に下記の引数を追加して、Publish Address が設定されている場合の挙動を改善。 reply: Unicast Address 宛に送信するかを指定するフラグ publish: Publish Address 宛に送信するかを指定するフラグ MS_generic_battery_server_state_update() MS_generic_level_server_state_update() MS_generic_location_server_state_update() MS_generic_location_setup_server_state_update() MS_generic_onoff_server_state_update() MS_generic_power_level_server_state_update() MS_generic_power_level_setup_server_state_update() MS_generic_power_level_server_state_update() MS_generic_power_level_setup_server_state_update() MS_generic_power_onoff_server_state_update() MS_generic_power_onoff_setup_server_state_update() MS_generic_user_property_server_state_update() MS_generic_admin_property_server_state_update() MS_generic_manufacturer_property_server_state_update() MS_generic_client_property_server_state_update() MS_light_ctl_server_state_update() MS_light_ctl_temperature_server_state_update() MS_light_hsl_server_state_update() MS_light_hsl_hue_server_state_update() MS_light_hsl_saturation_server_state_update() MS_light_lc_server_state_update() MS_light_lightness_server_state_update() MS_light_lightness_setup_server_state_update() MS_light_xyl_server_state_update() MS_scheduler_server_state_update() MS_sensor_server_state_update() MS_sensor_setup_server_state_update() MS_time_server_state_update() − 複数エレメントが同一クライアントモデルを使用する場合に Mesh モデルのインスタンスを変更するための関数を追加。 MS_generic_battery_client_set_model_handle() MS_generic_default_transition_time_client_set_model_handle() MS_generic_level_client_set_model_handle() MS_generic_onoff_client_set_model_handle() MS_generic_power_level_client_set_model_handle() MS_generic_power_onoff_client_set_model_handle() MS_generic_property_client_set_model_handle() MS_light_ctl_client_set_model_handle() MS_light_hsl_client_set_model_handle() MS_light_lc_client_set_model_handle() MS_light_lightness_client_set_model_handle() MS_light_xyl_client_set_model_handle() MS_scene_client_set_model_handle() MS_scheduler_client_set_model_handle() MS_sensor_client_set_model_handle() MS_time_client_set_model_handle()
Bluetooth ベアラー	**blebrr.c/h** − BLE_GAP_EVENT_ADV_ON イベントから R_BLE_GAP_StartAdv()をコールするまでの時間を 4msec から 10msec に延長して、Advertising 送信を途中停止する問題を解消。(BLEBRR_ADV_TIMEOUT) − Bluetooth ベアラーで Advertising 送信を繰り返す処理を削除。(BLEBRR_ADVREPEAT_COUNT) − Advertising 送信のタイムアウト処理を追加。(BLEBRR_ADV_ABORT_TIMEOUT, blebrr_advscan_timeout_handler) − ADV ベアラーを無効化する関数を追加。(blebrr_adv_disable) − Mesh ビーコン送信を繰り返す処理を削除。(blebrr_clear_bcon, blebrr_get_next_beacon, blebrr_bcon_send, blebrr_update_advdata)

	- Advertising 送信処理を更新。(blebrr_adv_setup, blebrr_pl_advertise_setup, blebrr_advscan_timeout_handler) - 特定のタイミングで ADV ベアラーの Scan が再開されない問題を解消。(blebrr_adv_sleep, blebrr_adv_wakeup) - サービスディスカバリで Mesh GATT サービスの発見を通知するイベントを追加。(BLEBRR_GATT_IFACE_FOUND) - GATT Service Changed の受信を通知するイベントを追加。(BLEBRR_GATT_IFACE_CHANGED) - Create Connection のキャンセルを通知するイベントを追加。(BLEBRR_GATT_IFACE_CANCEL) **blebrr_gatt.c** - GATT クライアントが接続を切断後に Scan が再開されない問題を解消。(blebrr_pl_gatt_disconnection) **blebrr.c** - 接続中に More Data ビットでデータを連続送信できるように設定を変更。(BLEBRR_CONN_MIN_CE_LEN, BLEBRR_CONN_MAX_CE_LEN) - Scan Response データをクリアする処理を追加して、Scan Response データが設定されている場合に Non-connectable Advertising が送信されない問題を解消。(blebrr_advertise_data_pl, blebrr_adv_param_set_comp_hdlr, blebrr_adv_data_upd_comp_hdlr, blebrr_gap_cb) - GATT Service Changed の受信を通知する処理を追加。(BLEBRR_GATT_IFACE_CHANGED) - Create Connection をキャンセルする API 関数を追加。(R_MS_BRR_Cancel_CreateConnection) - GATT Service Changed の CCCD 値を設定する API 関数を追加。(R_MS_BRR_Config_ServChanged) - GATT データベースの構成を変更した場合に GATT Service Changed を送信する処理を追加。(blebrr_set_gattmode_pl) - GATT クライアントとして接続を確立後に MCU サイズを拡張する処理を追加。(blebrr_gap_cb) **gatt_client.c** - MTU Exchange を実行する関数を追加。(mesh_client_expand_mtu) - GATT Service Changed の CCCD 値を設定する関数を追加。(mesh_client_config_serv_changed) - サービスディスカバリで GATT Service Changed を探す処理を追加。(mesh_client_common_disc_cb) **gatt_services.c/h** - MTU Exchange 要求に応答するための処理を追加。(mesh_serv_gatts_cb) - MTU サイズを R_BLE_GATT_GetMtu()で取得する処理を追加。(mesh_serv_get_mtu) - GATT データベースの Mesh Provisioning サービスと Mesh Proxy サービスを切り替える処理を追加。(mesh_serv_prov_init, mesh_serv_proxy_init, mesh_serv_prov_deinit, mesh_serv_proxy_deinit) - GATT Service Changed を送信する関数を追加。(mesh_indicate_serv_changed) **gatt_db.c** - Mesh Provisioning サービスと Mesh Proxy サービスを 1 つの GATT データベースに統合(gatt_db_prov.c/h, gatt_db_proxy.c/h**)**
ドライバ	**mesh_resources.h** - メモリプールサイズマクロの定義を更新。(MESH_MEMPOOL_SIZE) - ストレージサイズマクロの定義を更新。(MESH_STORAGE_SIZE)

	mesh_systemtime.c/h - 8 ビットタイマ(TMR)のレジスタに設定する値の計算式を更新。 - システムタイム有効化マクロのデフォルト値を(1)から(0)に変更。(SYSTEMTIME_EN)

■ **Rev1.30**

Bluetooth Mesh スタック	**Common** - MESH FIT モジュール設定に下記のマクロを追加。詳細は 3.1 節を参照。 MESH_CFG_LPN_CLEAR_RETRY_COUNT MESH_CFG_LIGHT_LC_SERVER_MAX - Mesh スタックを初期化して実行結果のステータスコードを返却する関数を追加。 MS_init_ext() - DataFlash に Provisioning 情報がない状態で Mesh スタックを初期化すると、Data Flash ドライバが Open 状態のままとなる問題を修正。 - ROM セクションのマッピング変更で、受信した Model メッセージをアプリケーションに通知できなくなる問題を修正。 - 各 Mesh Stack API のエラーチェックを強化。 **Provisioning** - Provisioning で使用される ECDH 鍵を生成し、Public 鍵を返却する関数を追加。 MS_prov_generate_ecdh_key_pl() **Network** - Proxy Advertisement with Node Identity の停止時に Config Node Identity ステートを更新する処理を追加。 - Proxy feature の無効時に Config Node Identity ステートを 0x02(not supported)に設定する処理を削除。 **Lower Transport** - 任意のタイミングで Friend Poll メッセージを送信する関数を追加。 MS_trn_lpn_poll() **Access** - 任意のエレメントを有効化する関数を追加。 MS_access_register_element_ext() - 登録されたモデル数を取得する関数を追加。 MS_access_get_model_count() MS_access_get_total_model_count() **Model** - Server Model と Setup Server Model の初期化処理を統合した関数を追加。

	MS_generic_location_server_init_ext() MS_generic_power_level_server_init_ext() MS_generic_power_onoff_server_init_ext() MS_light_ctl_server_init_ext() MS_light_hsl_server_init_ext() MS_light_lc_server_init_ext() MS_light_lightness_server_init_ext() MS_light_xyl_server_init_ext() MS_scheduler_server_init_ext() MS_sensor_server_init_ext() MS_time_server_init_ext() - Server Model と Setup Server Model の Status メッセージ送信を統合した関数を追加。 MS_generic_location_server_state_update_ext() MS_generic_power_level_server_state_update_ext() MS_generic_power_onoff_server_state_update_ext() MS_light_ctl_server_state_update_ext() MS_light_hsl_server_state_update_ext() MS_light_lc_server_state_update_ext() MS_light_lightness_server_state_update_ext() MS_light_xyl_server_state_update_ext() MS_scheduler_server_state_update_ext() MS_sensor_server_state_update_ext() MS_time_server_state_update_ext() - コールバック関数が Periodic Publication のタイムアウトを通知するように Server Model を更新。 - Health Server Model の終了処理を実行する関数を追加。 MS_health_server_deinit() - 使用する Health Client Model のモデルハンドルを設定する関数を追加。 MS_health_client_set_model_handle() - Scene Server Model の Scene Status メッセージを送信する関数を追加。 MS_scene_server_state_update() - Light LC Server Model に Light LC State Machine を実装し、Light LC Server Model の初期化関数に状態遷移イベントを通知するコールバック関数を下記の関数に追加。 MS_light_lc_server_init_ext() - Light LC Server Model の終了処理を実行する関数を追加。 MS_light_lc_server_deinit() - Light LC Server Model に Light LC Property ステートを設定する関数を追加。 MS_light_lc_server_set_time_property() MS_light_lc_server_set_fade_time() MS_light_lc_server_set_fade_on_time() MS_light_lc_server_set_fade_standby_auto_time() MS_light_lc_server_set_fade_standby_manual_time() MS_light_lc_server_set_occupancy_delay_time() MS_light_lc_server_set_prolong_time() MS_light_lc_server_set_run_on_time() - Light LC Server Model の Light LC State Machine ステートにアクセスする関数を追加。 MS_light_lc_server_set_lc_state_info() MS_light_lc_server_get_lc_state_info() MS_light_lc_server_report_occupancy() MS_light_lc_server_report_light_onoff() MS_light_lc_server_report_light_on() MS_light_lc_server_report_light_off()

Bluetooth ベアラー	**blebrr**.c – R_MS_BRR_Setup()で確保したリソースを解放する関数を追加。 R_MS_BRR_Close() – MS_prov_bind()で設定した Encoded URI を送信する処理を追加。
ドライバ	**mesh_resources.h** – メモリプールサイズマクロの定義を更新。(MESH_MEMPOOL_SIZE) – ストレージサイズマクロの定義を更新。(MESH_STORAGE_SIZE)

プログラム更新内容 (Mesh サンプルプログラム)

■ **Rev1.01**

mesh_cli	− Command Line Interface(CLI)プログラムを追加。
Mesh_model.c	− ステート操作関数にエラーチェックを追加。 Mesh_model_onoff_server_state_get() mesh_model_onoff_server_state_set()

■ **Rev1.10**

vendor_model	− 可変長のデータを送受信する Vendor モデルを追加。
mesh_core.c	− Bluetooth ベアラーの関数の仕様変更に伴って実装を更新。 R_MS_BRR_Set_GattMode() R_MS_BRR_Get_GattMode() R_MS_BRR_Register_GattIfaceCallback() − Bluetooth ベアラーのコールバック関数の仕様変更に伴って実装を更新。 BLEBRR_GATT_IFACE_CB − BLEBRR_GATT_IFACE_CB GATT インタフェースを持つデバイスとの複数同時接続に対応。 − GATT インタフェースイベントの BLEBRR_GATT_IFACE_NOT_FOUND で接続を切断する処理を追加。 − GATT インタフェースイベントの BLEBRR_GATT_IFACE_SCAN で Mesh GATT サービスを持つデバイスをログ表示する処理を追加。 − Provisioning の完了後、Proxy Advertisement の Identification Type が Node Identity となるように修正。 − 送受信するメッセージの SEQ が閾値(CORE_NET_IV_UPDATE_INIT_THRESHOLD)を超えると、IV Update プロシージャを開始する処理を追加。 − Low Power ノードとして Friend ノードと Friendship を確立し、Friend Subscription List を追加する処理を追加。 − Proxy Client として接続確立後、Subscription アドレスを対向 Proxy Server の Proxy Filter List に登録する処理を追加。 − Bluetooth Mesh スタックの全レイヤーの送受信 PDU と SNB をログ出力する処理を追加。 − LED 点滅のタイミングをプロビジョニング中から GATT インタフェースの接続中に変更。
mesh_model.c	− Composition Data ステートの Element Location を設定するためのマクロを追加。 ELEMENT_DESC_LOCATION − Generic OnOff Set メッセージ送信関数の引数に TID(Transaction ID)を追加。 − Vendor Client モデルと Vendor Server モデルのメッセージ送信関数、コールバック関数を追加。 − Bluetooth Mesh スタックの関数の仕様変更に対応。 MS_config_server_init() − Bluetooth Mesh スタックのコールバック関数型の仕様変更に伴って実装を更新。 MS_GENERIC_ONOFF_CLIENT_CB

main.c	– Bluetooth ベアラーの関数の仕様変更に伴って実装を更新。 R_MS_BRR_Init() R_MS_BRR_Setup() – Bluetooth ベアラーのコールバック関数型の仕様変更に伴って実装を更新。 BLEBRR_INIT_CB – Vendor Client モデルを使用し、コンソールで入力された文字列を送信する処理を追加 – Vendor Server モデルを使用し、受信した文字列をコンソールに表示する処理を追加。 – MCU リセット直後のオンボードスイッチ(SW1)の押下で、ノードコンフィグレーションをリセットする処理を追加。 – Config Node Reset メッセージの受信で MCU をリセットする処理を追加。
mesh_appl.h	– コンパイルスイッチを追加。 IV_UPDATE_INITIATION_EN IV Update プロシージャ LOW_POWER_FEATURE_EN Low Power 機能 CONSOLE_MONITOR_CFG Mesh モニタ機能 ANSI_CSI_EN ANSI エスケープシーケンス CPU_USAGE_EN CPU 使用率測定 – Mesh モデルのメッセージ受信コールバック関数を追加。 onoff_server_set_cb Generic OnOff Set メッセージ onoff_client_status_cb Generic OnOff Status メッセージ vendor_server_set_cb Vendor Set メッセージ vendor_client_status_cb Vendor Status メッセージ config_server_node_reset_cb Config Node Reset メッセージ – コンソールへのログ出力マクロを追加。 – CPU 使用率測定マクロを追加。

■ **Rev1.20**

mesh_cli	– コマンドパラメータから Static OOB AuthValue を取得する処理を追加。(root->core->provision->auth_act) – コマンドパラメータから OOB Public Key を取得する処理を追加。(root->core->provision->dev_pkey) – プロビジョニングで割り当てるユニキャストアドレスを変更するためのコマンドを追加。(root->core->provision->next_addr) – Mesh モニタ機能のログ出力処理を更新。(root->pkt_log) – BLEBRR_GATT_IFACE_FOUND イベント、BLEBRR_GATT_IFACE_CHANGED イベント、BLEBRR_GATT_IFACE_CANCEL イベントのログ出力を追加。(cli_gatt_bearer_iface_event_pl_cb) – GATT Service Changed の CCCD 値を制御するコマンドを追加。(root->brr->config_servchg) – Create Connection をキャンセルするコマンドを追加。(root->brr->cancel) – コマンドパラメータから Scan Response データに設定するローカルネームを取得する処理を追加。(root->brr->srsp_set) – Mesh モデルの STATUS メッセージ送信関数の仕様変更に従ってサーバーモデルコールバック関数の実装を更新。(appl_model_server_callback.c) – Light LC サーバーモデルの初期化関数に Generic OnOff サーバーモデルと Generic Power OnOff サーバーモデルを追加する処理を追加。(main_light_lc_server_operations) – Provisioning Capabilities の設定を変更し、OOB による Public Key 交換と Authentication を有効化。(appl_prov_capab) – GATT ベアラーによる Provisioning 完了時に切断する処理を追加。(appl_prov_callback)

	- Provisioning 完了後にプロビジョナーとして動作できるように実装を更新。(appl_prov_callback) - Network Message Cache の挙動を変更するテストコマンドを追加。(root->core->network->cache_wrap)
main.c	- ヘルスモデルのテスト ID を定義。 e_mesh_health_test_id_t - Bluetooth ベアラーに Scan Response データを設定。 blebrr_init_cb() - スイッチの長押しで Proxy Advertisement with Node Identity を送信する処理を追加。 sw_long_press_timer_cb()
mesh_appl.h	- メモリをダンプ表示する関数マクロを追加。(CONSOLE_DUMP_BYTES) - NetKey を生成する関数マクロを追加。(NETKEY_GEN) - AppKey を生成する関数マクロを追加。(APPKEY_GEN)
mesh_core.c	- 接続したデバイスのコネクションハンドルを保持する処理を追加。 mesh_core_gatt_iface_cb() - Proxy Advertisement の送信処理を更新。 mesh_core_prov_setup() mesh_core_prov_bind() mesh_core_proxy_setup() mesh_core_proxy_start() - 接続したデバイスの接続を切断する処理を追加。 mesh_core_proxy_disconnect() - Proxy 接続の確立時に全サブネットの Secure Network Beacon を送信する処理を追加。 mesh_core_proxy_cb() - フレンドサブスクリプションリストにアドレスを登録する関数を追加。 mesh_core_lpn_add_addresses() - IV Update 開始処理を更新。 mesh_core_monitor_net_pdu() - Mesh モニタ機能のログ出力処理を更新。 mesh_core_monitor_access_pdu() mesh_core_monitor_trans_pdu() mesh_core_monitor_ltrans_pdu() mesh_core_monitor_net_pdu() mesh_core_monitor_generic_log)
mesh_model.c	- MS_generic_onoff_server_state_update()の仕様変更に伴って処理を更新。 mesh_model_onoff_server_cb() - MS_vendor_server_state_update()の仕様変更に伴って処理を更新。 mesh_model_vendor_server_cb() - コンフィグレーションモデルで Friend 機能が無効化された場合に GATT ベアラーの全接続を切断する処理を追加。

	mesh_model_config_server_cb() - コンフィグレーションモデルで追加されたサブスクリプションアドレスを Friend Subscription List に追加する処理を追加。 mesh_model_config_server_cb() - Health Fault Status メッセージを送信する関数を追加。 mesh_model_health_server_fault_status() - MCU リセット後に Periodic Publishing を再開する関数を追加。 mesh_model_trigger_publishing()
vendor_server.c	- MS_vendor_server_state_update()の引数に reply と publish を追加。 - Periodic Publishing に対応。 vendor_server_publish_timout_cb()

■ **Rev1.30**

FITDemos	- 下記の Target Board for RX23W module 向けデモプロジェクトを新規追加。 tbrx23wmodule_mesh_cli tbrx23wmodule_mesh_client tbrx23wmodule_mesh_server
mesh_mobile	- MeshMobile のプロジェクトを更新。
rsskrx23w_mesh_cli **tbrx23w_mesh_cli**	- Friend Poll メッセージを送信する frndpoll コマンドを追加。 - Scene Status メッセージを送信する scene_update コマンドを追加。 - LC Time Property ステートを一括で設定する lcprop_set コマンドを追加。 - Mesh Stack API の更新に対応、コメントの更新など。
rsskrx23w_mesh_client **rsskrx23w_mesh_server** **tbrx23w_mesh_client** **tbrx23w_mesh_server**	- UIDR レジスタから生成したデバイス固有の Device UUID を使用するように変更。 - OOB Public Key と Static/Output/Input OOB Authentication を使用した Provisioning プロシージャに対応。 - Proxy 機能の無効時にすべての GATT コネクションを切断する処理を削除。 - Config Node Reset メッセージを受信後、MCU をリセットせず Provisioning を再開するシーケンスに変更。 - Mesh Stack API の更新に対応、コンソールログの更新、コメントの更新など。

改訂記録

Rev.	発行日	改定内容	
1.00	2019.10.11	-	初版発行
1.01	2019.11.29	P.8	表 2-2、表 2-3 のプログラムサイズを更新
		P.13	表 3-2 から BSP_CFG_ID_CODE_LONG_1 を削除
		P.21	5.4.1 項から ID コードの設定手順を削除
		P.32	5.7.1 項に ID コードの設定に関する注記を追加
1.10	2020.09.30	P.6	ハードウェア要件に 8-Bit Timer を追加
		P.6	ソフトウェア要件の r_cmt_rx バージョンを 4.5 以降に変更
		P.8	表 2-2、表 2-3 のプログラムサイズを更新
		P.9	表 3-1 に MESH_CFG_NET_SEQNUM_CACHE_SIZE マクロを追加
		P.25	5.6 節から「r_cmt_rx」項を削除
		P.25	5.6.1 項「r_ble_rx23w」を追加
1.20	2021.09.30	P.6	1.3 節「ファイル構成」に記載されたファイル構成を更新
		P.7	2.2 節「ソフトウェア要件」に記載された FIT モジュールのバージョンを更新
		P.9	表 2 2、表 2 3 のプログラムサイズを更新
		P.10	表 3 1に設定マクロを追加
		P.31	5.6節「コードの変更」を削除
1.30	2022.12.22	P.6	1.3節「ファイル構成」に"json"フォルダ以下を追加。
		P.7	2.3節「サポートするツールチェーン」のe^2 studioバージョンを更新し、Target Board for RX23W moduleを追加。
		P.9	表 2-2、表 2-3のプログラムサイズを更新。
		P.11	表 3-1のMESH_CFG_PROXY_SUBNET_NETID_ADV_TIMEOUTマクロとMESH_CFG_PROXY_SUBNET_NODEID_ADV_TIMEOUTマクロのデフォルト値を100から300に変更。
		P.12~P.13	表 3-1にMESH_CFG_LPN_CLEAR_RETRY_COUNTマクロとMESH_CFG_LIGHT_LC_SERVER_MAXマクロを追加。
		P.14	表 3-3にTarget Board for RX23W moduleの設定を追加。
		P.16~P.33	5.1節、5.4節、5.7節にTarget Board for RX23W module使用時の設定を追加。

製品ご使用上の注意事項

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

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

ご注意書き

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

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

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

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

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

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

(Rev.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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